리눅스 클러스터 기술 백서

클루닉스/서진우(alang@clunix.com)

차례

Chapter 1.

철학이 있는 리눅스 운영체제 구성

- Page 003

Chapter 2.

리눅스 기반 인터넷 서비스 구성

- Page 058

Chapter 3.

부하분산(Load Balancing) 클러스터 구성

- page 112

Chapter 4.

고가용성(High Availability) 클러스터 구성

- page 206

Chapter 5.

고성능(High Performance Computer) 클러스터 구성

- Page 256

To install Linux with philosophy

(철학이 있는 리눅스 운영체제 구성)

클루닉스 기술부 1차 세미나 자료

클루닉스 서 진우(alang@clunix.com)

철학이 있는 리눅스 설치

주요 내용

==========

1. 리눅스 설치
   ① 설치 전 고려 사항
      i. 파티션 정책
      ii. SCSI Adapter Driver Patch

   ② 리눅스 설치
      i. 파티션 정책
      ii. 패키지 정책
      iii. 설치 완료 후 기본 확인 작업

   ③ 설치 완료 후 부가 작업
      i. 주요 서비스 파티션 구성 확인 및 파일 시스템
      ii. 시스템 구성 확인

2. 패키지 업데이트
   ① S/W 패키지 업데이트

   ② H/W 패키지 업데이트

3. OS 관련 기본 보안 정책 및 보안 패키지 설치
   ① OS관련 기본 보안 정책
      i. 데몬 관리 정책
      ii. 기본 시스템 계정 정책
      iii. 시스템 퍼미션 정책
      iv. 일반 사용자 관리 정책
      v. TCP 보안 정책

   ② 보안 패키지 설치
      i. Fcheck 설치 후 관리 하기

서 진우(alang@clunix.com)

4. OS 설치 후 중요 데이터 백업 정책
   ① 백업 정책 설정 시 고려할 점
   ② 백업의 방식
   ③ 2차 및 3차 백업에 대해
      i. FTP를 이용한 2차 백업
      ii. RSYNC를 이용한 2차 백업
   ④ 복구의 방식
   ⑤ 백업 및 복구 시 주의점
   ⑥ 실전 백업 예제

5. 문제 발생 시 응급 복구 방법
   ① 긴급 부팅 시키기
   ② 부팅 디스켓 만들기
   ③ 파일 시스템 점검 수리
   ④ RPM 패키지 상태 초기화 하기

==========

서 진우(alang@clunix.com)

1. 리눅스 설치

① 설치 전 고려 사항

i. 파티션 정책

자동파티션 설치를 선택하면 기본적으로 /boot , / , swap 3개의 파티션으로 자동 활당되어진다. 하지만 정식 서비스를 위해서는 가지고 있는 하드디스크를 계산하여 성능 및 보안등을 고려하여 현 서비스에 가장 적절한 정책을 세워야 한다.

리눅스의 대표 파티션들은 모두 보안적인 측면이나 성능적인 측면에서 의미를 부여하고 있다. 리눅스의 대표적인 파티션을 살펴 보면 다음과 같다.

/boot
/
/etc
/home
/usr
/usr/local
/tmp
/var
swap

각 대표적인 모든 파티션을 다 나누어 설치를 하게 되면 가장 안정적일수 있지만 제한된 하드디스크로 모든 파티션을 나누게 되면 각 파티션에서 사용할 수 있는 용량이 고정되어 버리기 때문에 실제 서비스 도중 특정 파티션에 용량이 초과할 경우 추가 하드 디스크를 사용해야 하는 경우가 발생하기 쉽다.

뿐만 아니라 많은 파티션을 나누어 놓게 되면 성능면에서는 Disk i/o 를 분산 시킬수 있어 바람직 할수 있지만 관리 측면을 고려 해 본다면 필요에 의한 적절한 파티션 분배가 이루어 져야 할것이다.

각 파티션별로 그 의미에 대해 알아보자.

/boot :

/boot 파티션은 리눅스 부팅에 관련된 파일이나 이미지들이 모여 있는 파티션 이다. 만일 /boot 파티션을 나누어 놓은 경우는 크게 세 가지 의미가 있다.

첫째는 안정성이다. 실제 /boot 파티션을 나누지 않고 그냥 / 에 포함 시킬 경우 시스템이 과도한 사용이나 해킹등의 이유로 / 파티션이 망가질 경우 실제 / 포함 된 /boot 디렉토리의 booting image 들 역시 깨어질 위험성이 크다. /boot 가 나누어져 있을 경우 / 가 망가지더라도 새로운 / 파티션을 만들고 /boot 에 있는 부팅 이미지를 가지고 어느 정도 까지는 복구가 가능하다.

둘째는 기능성(?)이라 볼수 있다. 실제 /boot 에 들어 있는 kernal image 가 즉 리눅스 OS 라 볼수도 있다. 만일 당신이 개발자나 리눅스 전문 엔지니어라서 여러개의 배포판을 설치 할 경우 각 배포판 마다 /, swap 등의 파티션을 별도로 나누어서 설치해야 하는 어리석음을 갖지 말길 바란다. 실제 배포버전별로 /boot 파티션을 나누어 준다고 하면 각각의 다른 리눅스 배포판을 설치 하고도 실제 /etc /usr/local 등에 있는 설정이나 패키지를 통합해서 관리 할수가 있다.

셋째는 리눅스 OS의 부트로더의 기능적인 제한때문에 /boot 를 나누는 경우가 많다. 리눅스의 대표적인 부트로더로 Lilo 가 있는데 이는 실제 1024 실린더 밖에 설치 될 경우 치명적인 에러가 발생한다. 그래서 대부분이 하드디스크의 MBR 영역에 설치를 하게 되는데 만일 MBR 에 다른 부트 로더를 설치하고 Lilo 는 리눅스 파티션의 첫번째 파티션에 설치를 한다고 할때 /boot 파티션을 첫 번째 파티션으로 만들어 이곳에 Lilo를 설치 할수 있다.

/ :

루트 파티션은 실제 리눅스 파티션의 최 상위 파티션으로 실제 / 파티션과 Swap 파티션만으로도 리눅스를 설치 하여 서비스를 할 수도 있다. 하지만 이렇게 설치를 하게 되면 앞에서 언급하였고 뒤에서 설명하는 바와 같이 부적절한 상황을 초래하기 쉽다. 일단 커널 이미지에서 실제 / 파티션을 찾아서 / 파티션을 최상위로 해서 하위의 파티션들을 찾기 때문에 / 파티션이 없으면 리눅스를 부팅 시킬수 없게 된다. 아래에서 설명하는 바와 같이 여러개의 파티션을 나눌 경우 실제 별도의 파티션을 나누지 않는 상위 디렉토리들이 모두 이 / 파티션안에 포함 되게 될것이다.

/etc :

/etc 는 리눅스 시스템의 모든 설정 파일이 모여 있는 디렉토리이다. 만일 /etc

를 / 파티션에 포함한 경우 변수에 의해 / 파티션이 깨어진 경우 별도의 백업이 없으면 관리자로써 상당히 난처한 상황에 빠지게 딘다.

즉 아무리 데이터를 백업을 하고 있다 하더라도 설정 파일이 없으면 서비스를 구동 할수 없게 된다. 그렇기 때문에 /etc 를 별도의 파티션으로 나누는 것도 고려해야 할것이다. 하지만 /etc 의 용량은 매우 작기 때문에 백업만 적절히 한다고 하면 굳이 나눌 필요는 없다. 파티션이 너무 많이 나누어 지면 관리측면에서 부적절 할수도 있다.

/usr :

/usr 는 실제 레드헷 리눅스의 RPM 패키지가 설치되는 파티션이다. 이 파티션을 나누는 이유는 크게 성능과 패키지 관리 측면에서 볼수 있다.
일단 레드헷 리눅스의 모든 프로그램이 이곳에 설치가 된다고 볼수 있다.
즉 서비스 구동 시에 프로그램 성능에 가장 큰 영향을 주는 파티션인 만큼 다른 Disk i/o 과 많은 다른 파티션과 동일 파티션으로 구성하면 성능에 많은 지장을 주게 된다.
관리 측면으로 보면 최근 배포판 버전으로 업그레이드를 하고자 할때 /usr 파티션이 나누어져 있을 경우 설치 시 /usr 파티션만 새로 포맷하고 새 패키지를 설치하면 실제 redhat 7.3 에서 redhat 9 로 업그레이드를 한다고 해서 서버의 모든 자료를 백업하고 새로 OS 설치 후 복구하는 수고를 들수 있다.

/usr/local :

/usr/local 는 응용프로그램을 Source 로 설치 할 경우 파일이 위치 하는 디렉토리이다. 기본적으로 배포판의 패키지는 기본적이 서버 구성에 해당하는 패키지만 설치 하고 나머지는 모두 Source 설치 하는걸 권장 한다. 리눅스 배포판의 패키지를 무분별하게 설치 할 경우에는 보안/ 관리 측면에서 상당히 부적절 할 수 있다.

그러므로 최소 설치 후 필요한 프로그램을 Source 로 설치 하는 것을 권장하는 바이다. 즉 Source 로 설치하는 프로그램을 관리하는 파티션이다. /usr 과 같은 의미를 가진다. 성능과 패키지 관리 측면이다. 만일 / , /usr 등의 문제로 인해 다시 설치 할 경우 /usr/local 파티션이 / , /usr 에 속해 있으면 재 설치 할때 마다.. 일일이 Source 설치도 다시 해야 한다. 하지만 /usr/local 이 별도로 나누어져 있으면 OS를 다시 깔아도 실제 서비스 프로그램을 다시 깔 필요는 없게 된다.

/var :

/var 는 시스템 log 파일이 저장 되는 곳이다. /var 파티션을 분리하는 이유는 실제 log 를 기록 하면 그만큼 많은 Disk i/o 병목이 시스템에 발생하게 된다. 실제 서비스가 이루어 지는 / 혹은 /usr 와 사용자 데이터가 저장되는 /home 등과 같은 파티션이 존재 하게 되면 /var 에 일어나는 병목으로 인해 시스템의 성능이 저하되게 될것이다. 이를 방지하기 위해 /var 파티션을 별도로 활당하던지 네트워크 드라이브를 이용하여 원격디스크를 활용하는 방법을 사용한다.
(log 서버와 같이..)

/var 는 로그 및 메일 큐등이 쌓이는 곳이기 때문에 해커들이 고의적으로 시스템에 나쁜 코드를 심어 놓거나 mail 서버의 spam를 무수히 보냄으로 해서 /var의 디스크 용량이 무지커지는 경우가 발생한다. 이때 /var의 파티션이 나누어 있지 않는 경우 실제 / 파티션의 disk 용량이 가득차게 되고, 이로 인해 / 파티션의 중요한 파일들이 손상을 입게 된다. 이런 일들을 방지 하기 위해서도 /var 파티션을 나누는 것을 권장한다.

이밖에 해커들이 / 파티션을 공격하여 시스템을 깨어버린다 하더라도 실제 /var 파티션이 나누어져 있으면 이런 해커의 작업등이 시스템 로그에 남게 됨으로 시스템은 망가졌지만 이후 /var 의 로그를 이용하여 시스템에 무슨 문제가 있었는지를 확인할 수도 있다.

/tmp :

/tmp 는 실제 시스템 서비스가 가동될때 임시 파일이나 세션 파일들이 생기는 곳이다. 보통 nobody 권한의 파일들이 생김으로 해서 보안이 약한 곳이라 볼수 있는데 해커들이 주고 이 /tmp 를 통해 원격지에서 시스템을 공격하는 경우가 많다. 그러므로 해서 해커의 공격을 받더라도 /tmp 디렉토리만 영향을 받게 하기 위해 파티션을 분리하는것을 권장한다.

swap :

은 물리적인 메모리가 부족한 경우 하드디스크를 메모리 처럼 사용할 수 있게 하는데 이용되는 파티션이다.
보통 시스템의 물리적인 메모리의 1.5배에서 2배로 잡는것이 정석이다.
시스템의 주요 사용 용도 중에 매우 큰 용량의 파일을 다루는 작업일 경우 보통 swap 을 크게 잡아야 한다. 대표적으로 Oracle 의 경우 보통 DB file 하나가 작게는 500M 에서 크게는 2Gbyte 정도 하는 실제 DB 엔진에서 메모리에 이 큰 파일을 올려 놓고 작업을 하게 되는데 실제 물리적 메모리가 1.5 G 일 경우 swap 이 없는 경우에는 작업을 할수 없게 되거나 페이징이 일어나 성능이 급속

도록 떨어지게 된다. 이를 방지하기 위해서 서비스에 이용되는 메모리 수치를 예상하여 swap 을 잡으면 된다. swap 은 상황에 따라서 여러가지 방법으로 추가 할수 있기 때문에 초기에는 권장방법대로 1.5배에서 2배 정도로 잡으면 된다.

ii. SCSI Adapter Driver Patch

최신 Ultra320 SCSI Controllor 의 경우 기본 레드헷 지원 드라이버를 사용하면 i/o error 를 발생 하는 경우가 있다. 대표적인 제품으로 Adatec 의 AIC-29320 제품의 경우 그러함. 설치 시에 그냥 설치 하면 SCSI 장치 인식 중에 멈처 버리는 증세가 발생함.

boot : linux apic

와 같은 방식으로 부팅하면 장치는 인식하지만 레드헷 기본 커널에 있는 드라이버로 서비스 가동 시에 조금 과도한 Disk Access 작업을 할 경우 kernel panic 상태가 된다.

이에 설치 전에 각 장치 공급사의 홈페이지에서 최신 드라이버를 다운 받아서 설치 시에 최신 드라이버를 인식하도록 해주어야 한다.

http://www.adaptec.com/ 에서 최신 드라이버를 다운 받을 수 있음

먼저 각 하드웨어 공급사의 최신 드라이브를 다운 받은 후 설치 시 적용할 수 있는 이미지 디스켓을 만든다.

==========

```
# fdformat /dev/fd0
# dd if=aic79xx-2.0.2-i686-rh90.img of=/dev/fd0 bs=1024
```

==========

그런후 설치 CD 로 부팅 후 boot prompt 가 뜨면 < linux dd > 란 command 로 설치 시 추가 드라이브를 적용 시킨다.

==========

boot : linux dd

==========

그럼 설치 CD 의 커널 이미지로 부팅을 진행하면서 하드디스크 혹은 기타 장치를 인식하기 전에 추가 드라이브 디스켓을 삽입하라는 메세지가 뜬다. 이때 위에서 만든 드라이브 디스켓을 삽입한다.

그러면 설치 CD 에 있는 장치 드라이브를 이용하는것이 아니라 다운 받은 최신 드라이브로 장치를 인식하여 설치를 하게 된다.

② 레드헷 리눅스 서버 설치

서버용으로 설치를 할 경우 앞에서 설명한 바와 같이 이 서버가 서비스할 서비스 내역을 충분히 이해 한 후 반드시 필요한 패키지 만을 설치 하는 것을 권장한다.

사용하지 않는 패키지를 설치 할 경우 사용하지 않는 패키지의 보안적 문제가 생겼을때 관리자는 이를 신경쓰지 않게 되고, 이 패키지를 통한 해킹의 위험을 가지게 된다.

기본적은 설치 과정에 대한 설명은 생략 하고 파티션과 패키지 구성에 대한 정책만을 설명하겠다. 이는 저의 개인적인 견해를 설명한것이니 이가 완변한 설치 정책이라 말하지는 않겠다.

i. 파티션 정책

실제 서비스를 대상으로 하는 서버를 설치 할 경우에는 전 기본적으로 아래와 같이 파티션을 나눈다.

/boot
/
/usr
/usr/local

/var
/tmp
/home
swap

/boot : 100M ~ 200M
/ : 1G
/usr : 3G
/usr/local : 2G
/var : 2G ~ 3G
/tmp : 500M
/home : 필요한 만큼
swap : 물리적인 메모리의 2배

여기서 /boot /tmp 는 그 용량이 작기 때문에 적절한 백업을 할 경우에는 관리상의 이유로 / 에 포함하는 경우도 많다.

/usr/local 역시 관리상의 이유로 /usr 에 포함하는 경우도 많다.
대신 이와 같이 포함하는 경우에는 반드시 포함 시키는 파티션의 크기에 신경을 써야 한다.

/var 파티션의 경우 로그의 중요성이 낮거나 메일 서비스등을 하지 않는 경우에는 관리상의 이유로 하여 / 에 포함하는 경우도 많다. 하지만 메일 서버등을 중점으로 하는 서비스에서는 반드시 /var 의 파티션을 나누어 주고 용량도 충분히 주길 바란다.

/home /usr/local 의 파티션을 별도로 나눌 경우 성능을 고려 한다면 설치 시 파티션을 지정 하지 말고 기본 OS 설치 후 fdisk 를 통해 수동으로 파티션을 나눈 후 레드헷 기본 파일 시스템인 ext3 대신 reiserfs 혹은 xfs 파일 시스템을 사용하는 것을 권장 한다. ext3 에 비해 reiserfs 나 xfs 파일 시스템이안정성에서 우수함을 인증 받은 상태이고 성능에서도 30% 정도의 성능 향상을 가져 올수 있다.

(hdperm 으로 디스크 성능을 체크 해 보면 ext3 의 경우 초당 40M 정도의 read buffer 의 성능을 나타내지만 reiserfs 의 경우 초당 55M 의 성능을 발휘한다.)

/home 이나 /usr/local 파티션의 성격 상 디스크 사용량이 많거나 실제 사용하는 프로그램이 설치 되는 곳인만큼 시스템의 Disk I/O 성능의 영향을 많이 받는 곳이기에 보다 안정스럽고 좋은 성능의 파일 시스템을 사용하는게 좋다.

swap 역시 oracle 과 같은 큰 파일을 다루는 서버라고 하면 1G 용량으로 여러개의 swap 파티션을 잡아 두는 것을 권장한다.
(그냥 큰 용량으로 swap을 잡지 않고 적절한 용량으로 나누어 여러개를 잡는 이유는 swap 파티션의 크기 역시 일반 IA32 시스템에서는 2G 제한이 있고 2G의 스왑파티션 보다는 1G의 스왑파티션 2개가 I/O 분산으로 인해 성능이 좋기 때문이다.

OS 의 설치는 초보 엔지니어나 경력 엔지니어나 다 기본으로 하기 마련이다.
하지만 시스템 엔지니어의 시작이 OS 설치 라고 하면 마지막 역시 OS 설치라고 말 할수 있는 만큼 자신만의 설치 방법을 세워 두길 바란다.

ii. 패키지 정책

실제 OS 구동에 필요한 기본 프로그램 (부팅관련 프로그램, 라이브러리)은 배포판에 있는 패키지를 선택하지만 서비스에 필요한 프로그램은 대도록이면 Source 로 직접 설치 하는 것을 권장한다.

이는 패키지 관리 및 보안성격상에도 큰 의미를 가지고 있지만 시스템 성능에 미치는 영향도 크다

실제 배포판은 대표적인 아키텍처에 해당하는 시스템에서 해당 프로그램을 build 하여 패키지를 만들어 놓은 것이다. 유사한 아키텍처를 가진 시스템에서 사용하는데는 지장이 없지만 그래도 자기 시스템에 최적화된 프로그램을 사용하기 위해서는 실제 Source 로 해당 시스템에서 직접 설치 하는 것이 좋다.

(이와 같은 패키지 관리를 한다고 하면앞에서 설명한 바와 같이 /usr/local 파티션을 별도로 나누어 관리하는 것이 시스템 문제 발생하여 재 설치 시 오랜 시간이 소요되는 Source rebuiling 작업을 생략할 수 있게 된다.)

레드헷 리눅스에서의 패키지 선택은 기본적으로 패키지 선택 단계에서

/ 편집기 / DNS 서버 / 네트워크 서버 / 개발 도구 / 커널 개발 /시스템 도구

정도를 기본 설치 하고 기타 서비스 관련 프로그램 및 기타 관리 도구는 필요한 프로그램만을 수동으로 설치 하면 된다.

iii. 설치 완료 후 기본 확인 작업

기본적으로 설치가 완료 된 후 부팅이 정상적으로 되는지와 부팅 시 에러 내용이 있는
지, 파티션이 정책대로 적용이 되었는지 등을 확인 합니다.

dmesg -> 부팅 메세지 확인
df -h -> 파티션별 용량 확인

부팅 시 bootloader 에서 문제가 발생하는 경우가 많다.

특히 SCSI 하드와 IDE 하드를 같이 장착하고 설치 한 경우 LILO 혹은 GRUB 등의 bootloader
에서 정상적으로 커널 이미지를 Loading 하지 못하는 경우가 생긴다.
기본적인 해결 방법을 설명하겠다.

// LILO Bootloader 로 부팅 시 문제 발생 했을때 해결 방법

==========
==========
리눅스설치CD로 부팅해서 rescue 모드로 들어갔다.
df 쳐 보니 /dev/hda2 가 /mnt/sysimage 에 mount 되어 있다.
하드디스크에 있는 lilo.conf 는 /mnt/sysimage/etc/lilo.conf 에 잘 있다.
이제 이 파일을 잘 고쳐서 lilo 를 실행하면 된다. rescue 모드에서도 vi는 실행된다.

vi /mnt/sysimage/etc/lilo.conf
/mnt/sysimage/etc/lilo.conf 를 수정 후

/etc 에 복사한다
cp /mnt/sysimage/etc/lilo.conf /etc
여기까지만 하고 lilo 를 실행시키면 에러가 난다.

/etc/lilo.conf 에서 /boot 라고 되어있는 곳을 /mnt/sysimage/boot 로 바꾸고 저장한다.

이제 lilo 를 실행시킨다.

/mnt/sysimage/sbin/lilo

/boot/chain.b 를 못 찾는다는 메시지가 나타나면,

cp /mnt/sysimage/boot/chain.b /boot/ 한 다음에
#/mnt/sysimage/sbin/lilo

혹은

chroot /mnt/sysimage
/sbin/lilo

하면 된다.

==========

// GRUB Bootloader 로 부팅 시 문제 발생 했을때 해결 방법

==========

긴급시 grub 으로 부팅을 시키는 과정입니다.

grub >

먼저 root 지정을 해 줍니다.

grub > root (hd0,0)

-> 실제 /boot 파티션의 위치를 지정한다. 만일 별도의 /boot 파티션
이 없는 경우에는 / 위치를 지정합니다.

grub > kernel /vmlinuz-2.4.20-8smp ro root=/dev/sda2

-> 실제 / 파티션 위치를 적어 줍니다.

grub > initrd /initrd-2.4.20-8smp.img

-> SCSI 하드인 경우 initrd 이미지가 없으면 하드 인식을 못함.

grub > boot

하면 부팅 됩니다.

그런 후 /etc/grub.conf 를 수정하거나 쉽게 LILO 를 수정하여 다시 적용하고 실행 하면 된다.

참고로 LILO 로 SCSI 와 IDE 를 병행 할때 sda 가 첫번째 디스크가 아니라는 에러가 발생 하는 경우가 있다.

이는 LILO라는 부트로더는 실제 BIOS 의 장치에 관련된 정보를 참조 하지 않기 때문에 발생하는 문제이다.

디스크에 관련해서 이런 문제가 발생하면..

lilo.conf 에 다음 내용을 추가해 주면 된다.

```
disk=/dev/sda
bios=0x80
disk=/dev/hda
bios=0x81
```

위의 설정 내용은 SCSI 가 첫번째 하드라는 것을 LILO 에서 인식하도록 해주는 것이다. 이렇게 해야 부팅시 정상적으로 LILO 가 띄워 질 것이다.

③ 설치 완료 후 부가 작업

i. 주요 서비스 파티션 구성 확인 및 파일 시스템 설정

앞의 파티션 정책에서 설명 했듯이 주요 서비스 파티션 중 데이터 관련 파티션인 /home 과 서비스 프로그램 관련 파티션인 /usr/local 을 구성해야 하고 이 두개의 파티션을 안정성 및 효율이 높은 reiserfs 혹은 xfs 로 파일 시스템을 설정 하는 하도록 한다.

```
# fdisk /dev/sda

-> /usr/local    : 3G
-> /home                  : 나머지 전부 활당

# mkfs.reiserfs   /dev/sda8   -> /dev/sda? 에는   해당 파티션의 디바이스명을 적어준다.
# mkfs.reiserfs   /dev/sda9

# vi /etc/fstab
================================================================================
=============
LABEL=/                 /                       ext3      defaults           1 1
LABEL=/boot             /boot                   ext3      defaults           1 2
none                    /dev/pts                devpts    gid=5,mode=620   0 0
none                    /proc                   proc      defaults           0 0
none                    /dev/shm                tmpfs     defaults           0 0
LABEL=/tmp              /tmp                    ext3      defaults           1 2
LABEL=/usr              /usr                    ext3      defaults           1 2
LABEL=/var              /var                    ext3      defaults           1 2
/dev/sda6               swap                    swap      defaults           0 0
/dev/cdrom              /mnt/cdrom              udf,iso9660 noauto,owner,kudzu,ro 0 0
/dev/fd0                /mnt/floppy             auto      noauto,owner,kudzu 0 0
# 아래를 추가해줌
    /dev/sda8               /usr/local              reiserfs   defaults          1 2
    /dev/sda9               /home                   reiserfs   defaults          1 2

================================================================================
=============

# tar czvf local.tgz /usr/local    ->   /usr/local 의 디렉토리 구조를 백업해 둠
# cp local.tgz /
# mount -a
# cd /
# tar xzvf local.tgz
```

실제 파티션과 파일 시스템 구성이 정상적으로 되었는지를 확인 한다.

```
# df -h
```

```
================================================================================
=============
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda2             981M  121M  811M  13% /
/dev/sda1             198M   15M  173M   8% /boot
none                 1009M     0 1009M   0% /dev/shm
/dev/sda7             487M  8.1M  454M   2% /tmp
/dev/sda5             2.9G  898M  1.9G  33% /usr
/dev/sda3             2.9G   58M  2.7G   3% /var
/dev/sda8             2.9G   33M  2.8G   2% /usr/local
/dev/sda9              22G   33M   22G   1% /home
================================================================================
=============
```

ii. 시스템 구성 확인

마지막으로 현재 시스템의 H/W, S/W 구성 정보를 확인 및 기록 해 두도록 합니다.
이는 현재 시스템 장치가 재대로 인식을 하는지 확인과 동시에 이후 유지 관리를 위한 시스템 정보를 기록하는 의미를 가지고 있습니다. 이는 시스템의 정보를 확인 하는 방법으로 이후 관리 시에 큰 도움이 될것입니다.

- booting message 확인
dmesg

- CPU 정보
cat /proc/cpuinfo

- 메모리 정보
cat /proc/meminfo

- SCSI 정보 HDD 정보
cat /proc/scsi/scsi

- SCSI controllor 정보
cat /proc/scsi/aic7xxx/0

- IDE 하드 정보

cat /proc/ide/hda/model
cat /proc/ide/hda/setting

- 네트워크 정보
ifconfig
route -n

- 파티션 정보
fdisk –l /dev/sda

- 디스크 용량
df –h

이로써 대략적인 설치 과정이 완료 되었습니다.
다음 장에서는 설치 이후 설치 된 패키지의 최신 패키지 업데이트 및 기타 하드웨어의 최신 드라이브 업데이트에 대한 설명을 하겠습니다.

2. 패키지 업데이트

① S/W Package Update

리눅스 시스템은 Open 된 OS 인 만큼 보안에 대한 취약점이 있다고 생각할 수 있지만 이와 달리 Open 된 시스템인 만큼 보안의 취약점을 미리 사전에 파악하여 패치 함으로 관리자의 관심만 유지되면 보안에 더 강력하다고 말할 수 있다.

레드헷 리눅스의 경우 up2date 란 프로그램을 통해 윈도우 처럼 자동으로 패키지를 업데이트 할 수 있는데 기본적으로 배포판에 설치되는 up2date 로 업그레이드를 하면 SSL 인증 에러가 발생하게 될것이다. 이는 현재 배포 버전인 Redhat9 이 발표된 이후 레드헷사의 up2date 인증 방식에 변경이 있었기 때문에 정상적인 up2date를 이용하여 전체적인 패키지 업데이트를 하기 위해서는 먼저 up2date 를 업그레이드 해야 한다.

wget ftp://updates.redhat.com/9/en/os/i386/up2date*

위의 구문으로 바로 up2date 에 관련된 패키지를 다운 받을 수 있을 것이다.
up2date / up2date-gnome 두개의 패키지를 다운 받게 될것인데 이중 up2date-gnome

은 Xwindows 의 gnome 이 설치 된 경우에만 설치 하도록 한다.

rpm -Uvh up2date

/etc/rc.d/init.d/rhnsd restart
rhn_register

최초 rhn_register 를 실행하면 rhn_register 구성 편집 화면이 나오는데 그냥
q 를 치면 빠저 나온다.

rpm --import /usr/share/rhn/RPM-GPG-KEY
rhn_register
gpg --import /usr/share/rhn/RPM-GPG-KEY

레드헷 사에 현 시스템의 정보 및 간단한 인증 정보를 기록합니다.
정식 레드헷 배포판을 구매하면 up2date 의 지원을 계속 받을 수 있지만 공개 배포판의 경우에는 3개월 기간 제한이 있기 때문에 설치 시에 한번 지원 받는다는 의미를 가진다.

up2date -p -> 현 시스템이 RPM 패키지 정보를 정리한다.
up2date -u -d -> 업데이트를 시킬 패키지를 다운 받는다.

up2date 에서 -d 옵션을 사용하면 업데이트 대상 패키지를 다운 받아 /var/spool/up2date 안데 저장된다.

-u 옵션만 사용하면 다운 받아서 업데이트 패키지를 바로 설치 하고 설치가 완료된 rpm 을 삭제 하게 됩니다. 만일 업데이트 대상 시스템이 여러대 일 경우 계속 같은 방법으로 업데이트 시킬 필요 없이 한곳에서 다운 받아놓고 그 패키지를 이용하여 다른 시스템들의 업그레이드를 시키는 방법이 편할 것이다. 단 위 경우는 시스템의 패키지 구성이 유사한 경우에 의미가 있다.

② H/W Driver Update

dmesg message 를 살펴 보면 커널이 로딩한 장치의 드라이브 버전을 확인 할 수 있다.
흔히 문제가 되는 것이 SCSI Controllor 와 Intel Gigabit Card 가 대표적이다.
먼저 현재 시스템에 설치 시 인식된 장치를 확인합니다.

```
# cat /etc/modules.conf
================================================================================
===========
alias eth0 e1000
alias eth1 e100
alias scsi_hostadapter aic7xxx
alias usb-controller usb-uhci
================================================================================
===========
```

```
# dmesg | grep Driver
# dmesg | grep DRIVER
# dmesg | grep driver
```

scsi0 : Adaptec AIC7XXX EISA/VLB/PCI SCSI HBA DRIVER, Rev 6.2.8
Intel(R) PRO/1000 Network Driver - version 4.4.19-k1
Intel(R) PRO/100 Network Driver - version 2.1.29-k2

대표적인 버전은 AIC79XX -> 2.0.2 이상
e1000 -> 5.0.43 이상
e100 -> 2.2.21 이상

의 버전을 설치 하도록 한다.

rpm 으로 만들어진 드라이브 버전은 현 시스템 커널 버전에 의존함으로 드라이버 설치 후 kernal upgrade를 할 경우에는 반드시 다시 장치 드라이브 버전을 확인하셔야 합니다.

3. OS 관련 기본 보안 정책 및 보안 패키지 설치

① OS 관련 기본 보안 정책

i. 데몬 관리 정책

* 불필요한 데몬 제거

```
/sbin/chkconfig --del anacron
/sbin/chkconfig --del atd
/sbin/chkconfig --del autofs
/sbin/chkconfig --del echo
/sbin/chkconfig --del finger
/sbin/chkconfig --del gpm
/sbin/chkconfig --del nfs
/sbin/chkconfig --del nfslock
/sbin/chkconfig --del portmap
/sbin/chkconfig --del rlogin
/sbin/chkconfig --del rsh
/sbin/chkconfig --del rsync
/sbin/chkconfig --del lpd
/sbin/chkconfig --del xfs
```

* 기본 데몬

```
/sbin/chkconfig --add crond
/sbin/chkconfig --add network
/sbin/chkconfig --add sshd
/sbin/chkconfig --add syslog
```

* 서비스 데몬

```
/sbin/chkconfig --add named
/sbin/chkconfig --add proftpd
/sbin/chkconfig --add sendmail
/sbin/chkconfig --add nfs
/sbin/chkconfig --add portmap
/sbin/chkconfig --add rlogin
/sbin/chkconfig --add rsh
/sbin/chkconfig --add rsync
```

* 불필요한 xinetd 데몬 제거

```
# rm -f chargen
# rm -f chargen-udp
# rm -f daytime
# rm -f daytime-udp
# rm -f echo
# rm -f echo-udp
# rm -f finger
# rm -f ntalk
# rm -f rexec
# rm -f rlogin
# rm -f rsh
# rm -f rsync
# rm -f servers
# rm -f services
# rm -f sgi_fam
# rm -f talk
# rm -f telnet
# rm -f time
# rm -f time-udp
# rm -f proftpd-inetd
```

ii. 기본 시스템 계정 정책

* 불필요한 시스템 계정 삭제

```
# userdel adm
# userdel lp
# userdel sync
# userdel shutdown
# userdel halt
# userdel news
# userdel uucp
# userdel operator
# userdel games
# userdel gopher
```

* 불필요한 시스템 그룹 삭제

```
# groupdel adm
# groupdel lp
# groupdel news
# groupdel uucp
# groupdel games
# groupdel dip
```

iii. 시스템 퍼미션 정책

* suid sgid 퍼미션의 명령어 검색 후 퍼미션 변경

```
# find / -type f ₩( -perm -4000 -o -perm -2000 ₩)

chmod 700 /usr/bin/chage
chmod 700 /usr/bin/gpasswd
chmod 700 /usr/bin/wall
chmod 700 /usr/bin/chfn
chmod 700 /usr/bin/chsh
chmod 700 /usr/bin/newgrp
chmod 700 /usr/bin/write
chmod 700 /usr/bin/passwd
chmod 700 /usr/bin/at
chmod 700 /usr/bin/lockfile
chmod 700 /usr/bin/rcp
chmod 700 /usr/bin/rlogin
chmod 700 /usr/bin/rsh
chmod 700 /usr/bin/slocate
chmod 700 /usr/bin/crontab
chmod 700 /usr/libexec/openssh/ssh-keysign
chmod 700 /usr/sbin/ping6
chmod 700 /usr/sbin/traceroute6
chmod 700 /usr/sbin/usernetctl
chmod 700 /usr/sbin/userhelper
chmod 700 /usr/sbin/lockdev
chmod 700 /usr/sbin/userisdnctl
chmod 700 /usr/sbin/sendmail.sendmail
chmod 700 /usr/sbin/traceroute
```

```
chmod 700 /usr/sbin/utempter
chmod 700 /bin/ping
chmod 700 /bin/mount
chmod 700 /bin/umount
chmod 700 /bin/su
chmod 700 /sbin/pam_timestamp_check
chmod 700 /sbin/pwdb_chkpwd
chmod 700 /sbin/unix_chkpwd
chmod 700 /sbin/netreport

chmod 4750 /usr/bin/rcp
chmod 4750 /usr/bin/rsh
chmod 4750 /usr/bin/rlogin
chmod 4111 /usr/bin/sudo
chmod 2750 /usr/sbin/sendmail.sendmail
chmod 4750 /bin/su
```

* 중요 시스템 설정 파일 보호 정책

```
chattr +i /etc/fstab
```

* 관리자 그룹에 포함된 계정에게 시스템 관리 명령어 사용 허용 정책

```
chmod 750 /bin/ps
chmod 750 /bin/netstat
chmod 750 /bin/dmesg
chmod 750 /bin/df
chmod 750 /usr/bin/w
chmod 750 /usr/bin/who
chmod 750 /usr/bin/last
chmod 750 /usr/bin/top
chmod 750 /usr/sbin/lsof

chgrp wheel /bin/ps
chgrp wheel /bin/netstat
chgrp wheel /bin/dmesg
chgrp wheel /bin/df
chgrp wheel /usr/bin/w
```

```
chgrp wheel /usr/bin/who
chgrp wheel /usr/bin/last
chgrp wheel /usr/bin/top
chgrp wheel /usr/sbin/lsof
chgrp wheel /usr/bin/rcp
chgrp wheel /usr/bin/rsh
chgrp wheel /usr/bin/rlogin
```

```
# vi /etc/group
=====================================================================
.
.
wheel:x:10:root,clunix,alang
.
=====================================================================
```

* 시스템 최 상위 디렉토리 퍼미션 정책

```
chmod 711 /
chmod 711 /home
chmod 711 /var
chmod 711 /var/log
chmod 711 /etc
chmod 700 /root
```

iv. 일반 사용자 관리 정책

* 일반 shell 사용자 작업 내용 감시 하기

/etc/profile 파일의 제일 밑 부분에 아래 내용 추가

```
=====================================================================

if [ $LOGNAME != "admin" ]
then
HISTFILE=/var/log/user_history
```

```
TMOUT=200
echo -n "
==========================================================================
userhistory 로그인 ID: $LOGNAME 접속시간 : `/bin/date`
==========================================================================
" >> /var/log/user_history
fi

==========================================================================
```

/root/.bashrc 파일 밑 부분에 다음 라인 추가

```
HISTFILE=/root/.bash_history
TMOUT=-1
```

마지막으로 /var/log/user_history 파일의 퍼미션을 662 으로 해줍니다.

v. TCP 보안 정책

* TCP Wrapper 로 TCP 서비스 접근 권한 정책

vi /etc/hosts.deny

```
sshd : ALL : twist ( /root/bin/hostchk Y Y %a %c %d %h %n %p %s %u ) &
in.proftpd : ALL : twist ( /root/bin/hostchk Y Y %a %c %d %h %n %p %s %u ) &
```

vi /etc/hosts.allow

```
sshd : localhost 127.0.0.1 211.241.202. 192.168.1.
```

```
in.proftpd : localhost 127.0.0.1 211.241.202. 192.168.1.
```

vi /root/bin/hostchk

```
#!/bin/sh

################################  변수정의부문

# 메일 수신자
mailto=alang@clunix.com

# 화면출력 여부, 메일전송 여부
dsp=$1; msg=$2

# 접속자 정보 등
a=$3; c=$4; d=$5; h=$6; n=$7; p=$8; s=$9; u=$10

# 현재 시간
time=`date`

# 접속시도자 소속 서버의 finger 정보
finger=`/usr/bin/finger -l @$h 2> /dev/null`

################################  화면 출력부문

if [ $dsp = Y ]
      then

/bin/echo -n "
                        ====================================
                              접속이 허용되지 않습니다.
                        ====================================

                  Access Time                : $time
                  Client host address        : $a
```

```
                Client information         : $c
                Client host name(or IP) : $h
                Client host name            : $n
                Client user name            : $u
"
fi


################################  메일 송신부문

if [ $msg = Y ]
     then

/bin/echo -n "
                          ================================
                                 접속 거부자 상세정보
                          ================================

          Access Time                          : $time
          Access client host address       : $a
          Access client information        : $c
          The daemon process name            : $d
          Access client host name(or IP) : $h
          Access client host name           : $n
          The daemon process id              : $p
          Server information                 : $s
          Access client user name           : $u
                    ------------------------------------
                       Access client finger information
                    ------------------------------------
                              $finger
               ---------------------------------------------
 " | ₩
       /bin/mail -s "tcp_wrapper report [$d]" $mailto

fi
```

* Syn Flooding DOS 공격 방지 정책

vi /root/bin/synfl

```
sysctl -w net.ipv4.tcp_max_syn_backlog=1024
sysctl -w net.ipv4.tcp_syncookies=1
sysctl -w net.ipv4.icmp_destunreach_rate=1
sysctl -w net.ipv4.icmp_echo_ignore_broadcasts=1
sysctl -w net.ipv4.icmp_echoreply_rate=1
sysctl -w net.ipv4.icmp_ignore_bogus_error_responses=1
sysctl -w net.ipv4.icmp_paramprob_rate=1
sysctl -w net.ipv4.icmp_timeexceed_rate=1
sysctl -w net.ipv4.igmp_max_memberships=1
sysctl -w net.ipv4.ip_default_ttl=64
sysctl -w net.ipv4.ip_forward=0
sysctl -w net.ipv4.ipfrag_time=15
sysctl -w net.ipv4.tcp_syn_retries=3
sysctl -w net.ipv4.tcp_retries1=3
sysctl -w net.ipv4.tcp_retries2=7
sysctl -w net.ipv4.conf.eth0.rp_filter=2
sysctl -w net.ipv4.conf.lo.rp_filter=2
susctl -w net.ipv4.conf.default.rp_filter=2
sysctl -w net.ipv4.conf.default.rp_filter=2
sysctl -w net.ipv4.conf.all.rp_filter=2
sysctl -w net.ipv4.conf.eth0.accept_redirects=0
sysctl -w net.ipv4.conf.lo.accept_redirects=0
sysctl -w net.ipv4.conf.default.accept_redirects=0
sysctl -w net.ipv4.conf.all.accept_redirects=0
sysctl -w net.ipv4.conf.eth0.accept_source_route=0
sysctl -w net.ipv4.conf.lo.accept_source_route=0
sysctl -w net.ipv4.conf.default.accept_source_route=0
sysctl -w net.ipv4.conf.all.accept_source_route=0
sysctl -w net.ipv4.conf.eth0.bootp_relay=0
sysctl -w net.ipv4.conf.lo.bootp_relay=0
sysctl -w net.ipv4.conf.default.bootp_relay=0
sysctl -w net.ipv4.conf.all.bootp_relay=0
```

```
sysctl -w net.ipv4.conf.eth0.log_martians=1
sysctl -w net.ipv4.conf.lo.log_martians=1
sysctl -w net.ipv4.conf.default.log_martians=1
sysctl -w net.ipv4.conf.all.log_martians=1
sysctl -w net.ipv4.conf.eth0.secure_redirects=0
sysctl -w net.ipv4.conf.lo.secure_redirects=0
sysctl -w net.ipv4.conf.default.secure_redirects=0
sysctl -w net.ipv4.conf.all.secure_redirects=0
sysctl -w net.ipv4.tcp_keepalive_time=30
sysctl -w net.ipv4.tcp_fin_timeout=30
sysctl -w net.ipv4.tcp_tw_buckets=1440000
sysctl -w net.ipv4.tcp_tw_buckets=1440000
sysctl -w net.ipv4.tcp_keepalive_probes=2
sysctl -w net.ipv4.tcp_max_ka_probes=100
```

* 시스템 시작 시 자동 실행 코드 추가

vi /etc/rc.d/rc.local

```
rdate -s time.bora.net
/root/bin/synfl
```

* 최종적으로 Open port scan 확인

기본적인 OS 설정 및 보안 설정이 완료되면 최종적으로 현재 서비스가 이루어 지는 TCP/IP Port 에 대해 시스템 초기 상태에서 알아 두어야 한다.

이후 이때 파악 되지 않는 서비스 포트가 열려 있을 경우 보안에 의심을 해보 아야 할것이다.

Port Scan 도구로는 가장 많이 사용하는것이 nmap,nc,netstat,lsof 등 이다.
사용법은 다음과 같다.

nmap,nc,netstat 등을 이용해서 시스템의 열린 포트를 점검한다.

```
# nmap localhost -p 1-65535
# nc -w 3 -v -z localhost 1-65535
# netstat -ant | grep LISTEN

21/tcp      open        ftp
22/tcp      open        ssh
23/tcp      open        telnet
25/tcp      open        smtp
53/tcp      open        domain
80/tcp      open        http
3306/tcp    open        mysql
```

② 보안 패키지 설치

i. fcheck 설치 및 관리

** 파일 무결성 체크 프로그램 (Fcheck 설치..)

지금까지의 시스템 설치후 기본 보안에 신경을 써야 할부분에 대해서 알아보았고 마지막으로 현재 기본 보안 처리된 시스템의 무결성에 대해 앞으로 감시해서 유지 해야 할것이다. 무결성 체크 프로그램으로는 대표적인게 Tripwire 가 있다.
하지만 여기서 소개하는 Fcheck 는 가볍고 간단하면서 Tripwire 의 역활을 충분히 할수 있다.

먼저 프로그램을 다운 받도록 하자.

http://www.geocities.com/fcheck2000/
가면 최신 프로그램이 있을것이다. 다운을 받고 /usr/local 및에 둔다.
압축을 풀고..fcheck,fcheck.cfg 파일의 설정값을 몇개 수정하자.

tar xzvf FCheck_2.07.59.tar.gz

cd fcheck

vi fcheck

```
#################################################################
##############
#                                                               #
#                    User modifiable variable definitions:      #
#                                                               #
#################################################################
##############

# This should be passed through the command line, but hard coding still works
$config="/usr/local/fcheck/fcheck.cfg";
-------------------------------------- 이부분을 시스템 환경에 맞게 수정

----------------------------------------------------------------

# vi fcheck.cfg
----------------------------------------------------------------
Directory     = /etc/
Directory     = /bin/
Directory     = /usr/bin/
Directory     = /sbin/
Directory     = /usr/sbin/
Directory     = /usr/local/bin/
Directory     = /usr/local/sbin/
Directory     = /lib/
Directory     = /usr/lib/
Directory     = /usr/local/lib/

#
# Directory 설정은 fcheck 가 체크할 파일들이 들어 있는 디렉토리로 시스템에
# 중요한 명령어나 설정파일등이 있는 곳을 정해 주면 된다.
#

Exclusion         = /etc/passwd
Exclusion          = /etc/shadow
```

```
#
# Exclusion 설정은 Directory 설정에서 정한 디렉토리중 자주 변경이 되는 파일
# 이 있어서 체크때마다 걸리므로 체크 생략을 요하는 파일을 정해 주면 된다.
# 호스팅 업체나 기타 자주 사용자를 추가하는 서버라면 위와 같이 해주면 된다.
#

DataBase            = /usr/local/fcheck/data/data.dbf

#
# DataBase 설정은 리눅스 파일 정보를 DB 파일로 저장해서 다음 체크시 비교
# 분석할때 사용되어진다.
#

TimeZone            = GMT-9

#
# 한국 시간을 적용한다. GMT-9
#

#File     = /usr/local/admtools/logs/sol.dbf
```

File 설정은 필요 없으니 주석 처리 해준다.
기타 여러 설정값이 있으나 크게 작동하는데 영향을 주지 않는다.기본값을 적용한다.

이와 같이 적용후 /usr/local/fcheck/data 디렉토리를 만들어 준다.

```
# mkdir /usr/local/fcheck/data
# /usr/local/fcheck/fcheck -ac
```

이와 같이 fcheck -ac 로 파일 무결성 체크를 시작한다. 그럼 data 디렉토리 안에 data.dbf 파일이 생성되어 진다.

이제 Directory 설정에 해당하는 디렉토리 안에다가 파일을 하나 생성하고 재대로 점검을 하는지 테스트를 하여 보자..

touch /etc/test
/usr/local/fcheck/fcheck -a

그럼 아마 아래와 같은 결과를 출력할것이다.
PROGRESS: validating integrity of /etc/
STATUS:
ADDITION: [zzang911.net] /etc/test
Inode Permissons Size Created On
19508 -rw-r--r-- 0 Sep 19 20:13 2001

자 이제 이 명령어를 이용하여 주기적으로 시스템 파일 무결성을 체크하고 관리
자에게 통보하는 프로그램을 만들어 보도록 하자.

vi fcheck.sh

```
#!/bin/sh


CHECK=`/usr/local/fcheck/fcheck -a | grep Inode`
HOSTNAME=`hostname`

if [ -n "$CHECK" ]
    then
    /usr/local/fcheck/fcheck -a > fcheck_result
    mail -s "$HOSTNAME File 무결성 체크 결과" admin@clunix.com < fcheck_result
    rm -f fcheck_result
fi
```

간단한 이정도 스크립터로 보다 효율적인 관리가 가능해 질것이다.
이제 cron 에 등록시켜 놓고..매일 파일 무결성 체크를 간단히 메일로 받아서 관리
할수 있게 된다. 만일 변화된 파일이 정당한 변화라면..

/usr/local/fcheck/fcheck -ac

로 DBfile 를 업데이트 시켜 줘야 한다. 아님..계속 메일이 날아 오게 된다.

이로써 시스템 설치후 점검해야 할 보안 설정에 대해서 마치겠다. 시스템 보안은 초기의 보안 설정이 아주 큰 부분을 차지 하고 있다. 하지만..지속적은 감시와 관리역시 초기 보안 상태를 유지하는 가장 중요한 작업이라고도 할수 있다.

4. OS 설치 후 중요 데이터 백업 정책

이른바 "정보보호" 의 마인드가 확산되면서 해킹이나 크래킹에 대비한 정보보호의 측면으로 많은 논의가 되고 정보가 공유되고 있지만 정작 매우 중요한 백업에 대한 구체적 방식 및 주의점에 대해서는 많은 정보가 공유되고 있지 않는 실정이다. 실무적으로 시스템을 담당하는 이들은 사전에 백업의 중요성을 인식하지 못하다가 중요한 데이터를 훼손당하거나 긴급하게 데이터를 복구하여야 할 필요성이 있을 경우에서야 비로소 백업의 중요성을 인식하여 소 잃고 외양간 고치는 격으로 백업을 실시하거나, 설사 백업을 하더라도 사전 지식이나 올바른 정책 없이 실시하여 매우 비효율적인 방법으로 백업을 하는 경우가 많이 있다.

따라서 이 섹션에서는 일반적으로 폭넓게 사용하고 있는 리눅스를 중심으로 보안이라는 화두와 결코 따로 갈 수 없는 백업의 방식과 백업시 주의점에 대해 알아보도록 하겠다.

① 백업정책 설정 시 고려할 점.

백업을 실시할 경우 어떤 방법으로 어떤 매체를 이용해 어떤 주기로 할 것인지 이른바 "백업정책" 을 설정하여야 하는데. 이때 고려하여야 할 점에 대해 알아보기로 하자.
(1) 호환성 - 자신이 운영하는 시스템에서 백업의 대상이 되는 데이터가 각기 다른 운영체제와 다른 시스템과 호환이 될 필요가 있다면 백업 후 데이터의 호환성이나 이식성을 고려하여야 한다.

(2) 자동 백업 - 사실 백업이라는 절차가 만일의 사고에 대비하여 꼭 필요한 절차이기는 하지만 손이 많이 가는 작업이기는 하다. 특히 백업을 하여야 할 대상이 여러 시스템이거

나 하나의 시스템내에서도 일괄 백업이 아닌 일부의 데이터에 대해서만 백업을 할 필요가 있다면 일련의 백업작업의 자동화는 필수라고 할 수 있을 것이다.
(3) 비용 - 데이터의 유실이나 사고 발생시에는 백업의 소중함을 더 없이 인식하지만 평소에는 백업이 마치 사치라고 생각하는 사람이 있을 수도 있다. 따라서 가급적이면 적은 비용으로 백업을 할 수 있다면 좋을 것이다.

(4) 작업의 편리성 - 요즘에는 편리하게 백업정책을 설정하고 복잡한 설정도 쉽게 할 수 있으며 백업 상황등을 GUI 환경으로 볼 수 있는 백업유틸리티나 상용 솔루션들이 많이 나와 있다. 그러나 이러한 X-Windows 기반의 프로그램의 경우 일반 데스크탑에서는 유용하나 보안을 중요시하는 서버 시스템의 경우 X-Windows 자체를 사용하지 않는 경우가 있으므로 이 부분은 신중하게 검토할 필요가 있다.

(5) 네트워크 백업 - 이 부분은 자동백업과 함께 가장 중요한 부분중의 하나라고 판단된다. 일반적으로 백업은 백업매체의 자체백업(또는 1차 백업이라 한다.) 뿐만 아니라 일정 주기로 2차,3차 백업을 하게 되는데, 이러한 2차,3차 백업이 자동으로 되려면 네트워크를 통한 백업이 가능하여야 한다.
(6) 저장 매체의 선택 - 백업의 매체에 따라 저장방식이 다른 경우가 있으므로 신중히 선택되어야 한다. 고전적인 방식의 테이프나 추가의 하드 디스크, 다시쓸 수 있는 CD등 다양한 매체가 있을 수 있다. 위의 여러 가지 사항 및 자신이 관리하는 시스템의 특성을 고려하여 저장매체를 선택하여야 할 것이다.

권장할 만한 방식.

위의 여러 가지 상황을 고려하여 이식 성이 높고 여러 옵션이 지원되며 백업 파일을 이용해 복구 시에서도 강력한 기능을 제공하는 tar를 이용하여, 하드 디스크에 저장하는 방식을 권장한다. tar는 "Tape ARchiver" 에서 나온 말에서도 알 수 있듯이 초기에 주로 테이프 백업을 위해 자주 쓰였던 유틸리티인데, 이는 리눅스를 포함한 변종의 유닉스 계열에서도 모두 사용이 가능하고 Shell Script 등을 이용하면 복잡해 보이는 백업정책도 쉽고 단순하게 세울 수 있으며 백업후 복구가 필요한 시점에서도 강력한 기능을 발휘할 수 있다.

그리고, 예전에는 주로 테이프 백업을 선호하였지만 요즘에는 고속의 고용량, 저가의 하드 드라이브가 보편화되면서 테이프보다는 하드 드라이브를 추가 장착하여 백업매체로 써 널리 이용하는 추세이다.

② 백업의 방식

그럼, 이제 구체적으로 어떻게 백업할 것인지 백업의 방식에 대해 이야기 해 보도록 하자.
먼저, 어떤 데이터를 어떤 주기로 백업할 것인가를 선택하여야 할 것이다.
모든 데이터를 매일 백업하는 것은 비효율적이므로 꼭 백업이 필요한 데이터만 적당한 주기로 백업하는 것이 불필요한 백업으로 인한 디스크 공간을 절약할 수 있고 백업작업으로 이한 필요이상의 프로세스를 줄일 수 있다.
백업을 할 대상은, 다른 서버에서 복사(이식)가 가능한 바이너리나 기본설정화일들은 굳이 백업할 필요가 없으며 각 시스템마다 고유한 시스템 설정(Configuration) 과 실제 데이터는 반드시 백업하여야 할 것이다.

일반적으로 리눅스의 경우 대부분의 설정은 /etc 디렉토리에 있으며, 각종 데이터는 /home 이하에, 그리고 기타 부가적으로 설치하는 프로그램은 /usr/local 에 있으니 자신의 시스템 고유의 설정을 고려하여 각각의 디렉토리이하에 대해 적절히 백업을 하면 된다. 이를테면 메일 서버의 경우 추가적으로 /var/spool/mail 이하의 데이터가 매우 중요하므로 백업시 추가하여야 할 것이다. 백업 디렉토리 설정시 주의할 점은 backup 이 되는 파일을 다시 백업하여 무한루프에 빠지는 실수를 주의하여야 하고 실시간으로 시스템내 프로세스에 대한 정보가 저장되는 /proc 역시 백업을 하는 실수를 범하지 말아야 한다.

그럼. 예제를 통해 실제로 백업을 하는 명령어를 알아보도록 하자.

```
cd /backup
tar cvpfz /backup/home.tar.gz /home --exclude=/home/test
```

[스크립트 1]

위의 명령에서 사용한 옵션에 대해 하나씩 알아보도록 하자.
먼저 백업파일은 /backup 이라는 디렉토리에서 생성하기를 원하므로 /backup 으로 이동한후 tar 명령을 이용해 백업을 시작한다.

"c" 는 Create a new archive 의 뜻으로 백업시 새로운 파일을 생성하며

"v" 는 Verbosely list files processed 의 뜻으로 백업시 백업이 진행되고 있는 상황
및 디렉토리 리스트를 보여주며
"f" 는 archive file is local 의 뜻이며 명령어 이후의 이름이
생성될 파일의 이름이라는 뜻이며
"p" 는 Preserve-Permissions 의 뜻으로 이전 데이터의 Permission 정보를
그대로 보존한다는 뜻이며
"z" 는 filter the archive through gZip의 뜻으로 gzip 으로 압축한다는 뜻이다.
일반적으로 tar로 백업시 사용되는 확장자 규칙은 압축되지 않고 단순히 하나의 파일로 패킹(packing) 한 파일의 경우에는 "tar" 를, gzip 으로 압축된 파일의 경우에는 "tar.gz" 나 "tgz" 를 확장자로 한다.
따라서 확장자가 home.tar.gz 나 home.tgz 일 경우 tar 로packing
(즉 단순히 하나의 파일로 묶기만 함) 된후 gzip 으로 압축되었음을 뜻한다.

[스크립트 1] 과 같이 실행되었을 경우에는
/backup 디렉토리에 /home/test 이하를 제외한 /home 이하의 모든 파일들이 home.tar.gz 라는 이름으로 생성된다. 제외할 디렉토리나 파일이 더 있을 경우에는 --exclude 옵션을 계속 이어서 쓰면 된다.
그리고 백업시 ``z'' (압축) 옵션의 사용에 유의하여야 한다. 단순히 하나의 파일로 압축백업시에는 때에 따라 데이터의 일부가 손상되는 경우가 있는데, 이때에는 백업된 파일 전체가 깨어져 복구를 할 수 없게 되는 경우가 있다. 백업시 압축을 하지 않고 단순히 tar로 Packing한(묶은) 경우에는 설사 파일이 깨어지거나 손상되었다 하더라도 파일의 복구가 가능하기 때문에 백업시 압축을 할 것인지 여부를 신중히 선택하여야 한다.
아울러 현재까지 최신 안정 커널인 Kernel 2.2 리눅스의 경우, 하나의 파일은 최대 2G가 최대 사이즈인데, 특정 파티션 이하에 많은 파일이 있어 하나의 파일로 백업하였는데, 2G를 넘을 경우에는 적당히 분산하여 백업을 하여야 한다.
이때에는 아래와 같은 방법이 가능할 것이다.

```
tar cvfpz home_a_h.tar.gz /home/[a-h]*
tar cvfpz home_i_p.tar.gz /home/[i-p]*
tar cvfpz home_q_z.tar.gz /home/[q-z]*
```

/home 이하의 디렉토리에 많은 하부 디렉토리 및 파일이 있을 때에는 전체 디렉토리를 백업할 경우 2기가를 초과하여 백업파일이 제대로 생성되지 않으므로, 이를 /home 이하의 디렉토리명이나 피일 이름의 이니셜로 구분하여 3등분하면서 백업을 하는 것이다.

그럼, 실제로 백업을 하는 스크립트를 분석해 보자.

```
#!/bin/sh                                    ..... (1)
cd /backup                                   ..... (2)
rm -f *.tar.gz                               ..... (3)
tar cvfpz apache.tar.gz /usr/local/apache/* ..... (4)
tar cvfpz home.tar.gz /home/* --exclude=/home/test    ..... (5)
tar cvfpz etc.tar.gz /etc/*                  ..... (6)
tar cvfpz mail.tar.gz /var/spool/mail/*      ..... (7) ,
chown backup.backup *.tar.gz                 ..... (8)
chmod 700 *.tar.gz                           ..... (9)
ls -alh | mail -s www50_backup backup@clunix.com ..... (10)

df -sh | mail -s www50_df backup@clunix.com      ..... (11)
```

[스크립트2]

(1) 번은 시작되는 내용이 Shell Script 임을 정의한다.
(2) 백업작업이 이루어지는 파티션인 /backup 으로 이동한다.
(3) 새롭게 백업을 하여야 하므로 기존의 백업된 파일을 모두 삭제한다.
(4) - (7) 번까지는 백업을 해야 할 디렉토리 이하에 대해 데이터를 각각 백업한다.
(8) 백업된 파일의 소유권을 backup 으로 설정한다.
(9) 백업된 파일의 권한을 700 으로 설정한다.
(10) 백업이 제대로 되었는지 확인하기 위해 현재의 디렉토리내 백업 파일의 리스트등의 정보를 백업담당자인 backup@clunix.com 에게 메일로 발송한다.
(11) 혹 파티션이 가득 찰수도 있으니 현재의 파티션 사용 정보를 백업담당자에게 메일로 발송한다.

별도의 설명이 필요하지 않을 정도로 간단한 스크립트이다.
Shell 스크립트는 실행되어야 할 명령어를 순서대로 나열하기만 하면 된다.
위 예에서는 Full Backup을 보여주고 있는데, 필요에 따라 적절히 "증가분 백업" 도 이용하기 바란다. 증가분(변경분) 백업은 풀백업을 받은 후 기존의 백업 데이터와 비교하여 이미 백업된 이후 변경되거나 추가된 부분만 백업하는 방식이다.

예를 들어 설명해 보도록 하자.
아래의 예는 백업을 단순화 하기 위해 /home 에 대해서만 백업하는 예를 들겠다.
백업정책은 일주일을 단위로 풀백업을 받고 이후 매일 증가분 백업을 하기로 한다.

```
#!/bin/sh                                    .....(1)
TODAY=`date +"%d %b %Y"`                     ..... (2)
PREVIOUSDAY=`cat FULL_BACKED_DAY`            ..... (3)
tar cvfpzG _modified_home.tgz -N "$PREVIOUSDAY" /home .....(4)
```

[스크립트3]

먼저 위의 스크립트를 실행하기에 앞서 [스크립트 2]를 참고하여 일주일을 단위로(정기적으로 백업을 하는 방법은 이후에 설명할 CRON을 이용하면 된다.) /home 에 대해 Full Backup을 실행하기로 한다.(이때 백업 스크립트 제일 하단에
echo `date +"%d %b %Y"` > FULL_BACKED_DAY 를 추가한다)
이후 매일 위의 [스크립트3] 스트립트를 실행하면 풀백업이후 추가 및 변경된 파일과 디렉토리만 선별하여 별도로 백업을 하게 되는 것이다.

(1) 실행되는 스크립트가 Shell 스크립트임을 정의한다.
(2) 현재의 시각을 -N 옵션이 이해할 수 있는 13 Jan 2001 형태로 TODAY 라는 변수에 대입한다.
(3) 이전에 Full 백업시의 시각인 FULL_BACKED_DAY를 읽어 PREVIOUSDAY 라는 변 수에 대입한다.
(4) /home 디렉토리 이하에 대해 Full 백업시의 시각 이후로 변경, 추가된 파일에 대해서만 modified_home.tgz 라는 파일로 저장한다.

증가분 백업 스크립트는 다소 복잡하기는 하지만 각자의 환경에 따라 [스크립트 2] 스크립트와 적절히 혼합하여 사용한다면 프로세스의 유발을 최소화하여 유용하게 사용이 가능할 것이다.
그럼, 이의 스크립트를 매주 또는 매일 일일이 실행할 필요 없이 자동으로 실행할 수 있는 방법은 무엇일까? 이 물음에 해답을 줄 수 있는 것이 바로 cron을 이용하는 것이다.

cron 은 일종의 일정관리 데몬으로서 기본 설정파일인 crontab 에서 정해진 시각에 따라 주기적으로 명령을 수행한다. crontab 은 아래와 같이 7개의 구성요소로 이루어지는데, 6번째 필드(User)는 생략되어도 무방하다.

분 | 시 | 날짜 | 달 | 요일 | 사용자 | 명령어
각 항목은 정수로 나타낼 수도 있고 또한 몇 개의 항목은 와일드카드 문자로 인식되는 ``*'' 문자로 표현이 가능한데, * 는 "매" 의 의미이다. 즉 시간 항목에 * 이 있다면 매시간이라는 뜻이고 날짜 항목에 * 이 있으면 매일이라는 뜻이 되는 것이다. 그리고 하나의 필드에 중복된 시간을 나타내고자 하면 콤마로 구분하면 되고, 연속된 시간을 나타내고자 하면 하이픈(-)을 이용하여 일정 기간을 나타낼 수 있다.
참고로 분은 0-59, 시는 0-23, 날짜는 0-31, 달은 0-12(0또는 12는 12월, 1은 1월), 요일은 0-7(0과 7은 일요일, 1은 월요일)로 나타낸다.
그럼, crontab 파일을 열어보자.

```
01 * * * * root run-parts /etc/cron.hourly
02 2 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
0 0 * * 1 root rdate -s time.bora.net && clock -w
```

위 설정에서 각각의 의미를 알아보도록 하자.
첫줄은 매시 1분에 /etc/cron.hourly 디렉토리내의 스크립트를 실행한다는 의미이고
두번째줄은 매일 02시 2분에 /etc/cron.daily 내의 스크립트를 실행한다는 의미이며
세번째줄은 매월, 매일 새벽 4시 22분에 /etc/cron.weekly 디렉토리내 스크립트를 실행하는 것처럼 보이지만 5번째 필드인 <요일> 필드값이 *이 아닌 0 이므로 1주일에 1회 일요일(0이므로) 에 /etc/cron.weekly 디렉토리내의 스크립트를 실행하라는 의미이다.
네번째줄은 같은 방식으로 직접 해석해 보기 바란다.
다섯번째줄과 같이 매일, 매주 0시0분, 매주 월요일(1) 에 rdate -s time.bora.net &&

clock -w 라는 명령어를 실행하라는 형식으로 직접 명령어를 지정해 줄 수도 있다.
각 디렉토리내 실행 스크립트는 정기적으로 시스템에 의해 실행이 되어야 하므로 실행(eXecute) 권한이 있어야 함은 당연하다. 따라서 위에서 작성한 스크립트를 700 정도의 적당한 실행권한을 부여하여 해당 디렉토리에 설정해 주면 될 것이다.
또는 직접 /etc/crontab를 편집하여 실행시간을 정의할 수도 있다.

기타 tar나 crontab 에 대한 부가적인 명령어 옵셥은 man 페이지나 HOWTO 문서를 참고하기 바란다.

③ 2차 및 3차 백업에 대해

서버 자체에서의 1차 백업은 짧은 기간에 정기적으로 이전의 백업을 삭제하고 새롭게 백업을 받으므로 데이터의 훼손이나 삭제사실을 늦게 발견하였을 경우에는 이미 삭제된 데이터로 백업을 받아 복구할 수 없는 경우가 발생한다. 또는 자체 백업이 되는 서버에 장애가 있거나 백업 데이터 자체가 훼손이 되는 경우도 있다. 이러한 경우에 대비하여 반드시 2차백업을, 가능하다면 3차백업까지 설정 하여 운영하는 것이 필요한데, 2차 백업은 1차 백업의 내용을 ftp로 전송하거나, rsync를 이용하여 특정 디렉토리 이하에 대해 동기화를 하는 방법등이 있다. 일반적으로 백업 주기를 타이트하게 설정시 1차백업은 월,수,금 새벽에 진행하고, 2차 백업은 화,목,토 새벽에, 그리고 3차백업은 2차백업서버에 접속하여 수,일 새벽정도에 하는 것이 권장할 만한 방법이다.

i. FTP 를 이용한 2차 백업

ftp를 이용한 2차 백업서버로의 전송은 expect 스크립트를 이용해 ftp 작업을 자동화 할 수 있으며 ncftpget을 이용, 스크립트를 작성하는 방법도 있다.

```
## Expect script를 이용하는 방법

#!/usr/bin/expect                    .....(1)
spawn ftp data.tt..co.kr             .....(2)
expect "Name :"                      .....(3)
send "backup₩r"                      .....(4)
expect "Password:"                   .....(5)
send "tomorrow₩r"                    .....(6)
expect "ftp>"                        .....(7)
sleep 1                              ..... (8)
send "bin₩r"                         .....(9)
expect "200 Type set to I."          .....(10)
sleep 1                              .....(11)
expect "ftp>"                        .....(12)
```

```
send "get /backup/home.tar.gz /home/pr/data_home.tar.gz₩r" .....(13)
expect " Kbytes/sec)"                                      .....(14)
sleep 1                                                    .....(15)
expect "ftp>"                                              .....(16)
send "quit₩r"                                              .....(17)
expect "221 Goodbye."                                      .....(18)
interact                                                   .....(19)
```

(1)먼저 expect 스크립트라는 것을 정의한후
(2)백업을 받으려는 서버인 data.clunix.com 에 ftp로 접속한다.
(3)"Name :" 이라는 응답이 오기를 기대(expect)한다.
(4) 데이터 서버에 접속하기 위해 username 인 backup 을 입력후 엔터를 입력한다.
(5) username을 입력했으므로 Password: 라는 반응이 오기를 기다린다.
(6) username 에 해당하는 암호인 tomorrow를 입력한다.
(이 부분은 각자의 환경에 따라 다를 것이다)
(7) ftp> 라는 반응이 오기를 기다린다.
(8) 1초간 대기를 한후
(9) binary 로 전송을 받아야 하므로 bin 입력을 한다.
(10) 서버로부터 200 Type set to I 라는 반응이 오기를 기다린후
(11) 1초간 대기를 한후
(12) ftp> 라는 반응이 오기를 기대(expect) 한후
(13) 원격지 데이터 서버의 /home/pr/data_home.tar.gz를 백업서버의
/backup디렉토리에 home.tar.gz 라는 파일로 저장한다.
(14) 사버로부터 Kbytes/sec) 라는 메시지가 출력되면
(15) 1초간 대기를 (sleep)한 후
(16) ftp> 메시지가 오기를 기다린 후
(17) quit를 입력하여 접속을 끊고
(18) 221 Goodbye 메시지가 나오기를 기대한 후
(19) 이후 명령어입력 정보를 기다린다.

만약 전송받는 파일의 사이즈가 클 경우에는 sleep 1 대신에
set timeout -1 를 지정하여야 정상적으로 ftp 전송이 가능하다.

그러나 expect 보다는 ncftpget을 이용하는 방법이 더 간단하고 빠르게 전송할 수 있다.

```
#!/bin/sh

rm -f *.tar.gz              .....(1)
ncftpget -u backup -p 'root///' data.clunix.com /home/data_backup ₩
 '/backup/home.tar.gz' .....(2)
```

이 두줄로 간단하게 처리가 되는데 이는
(1) 기존의 백업받았던 데이터를 삭제후
(2) ncftpget을 이용해 data.clunix.com 에 접속하여 /backup 디렉토리의 home.tar.gz 라는 파일을 백업서버의 /home/data_backup 디렉토리이하에 home.tar.gz 라는 파일로 ncftp전송한다는 뜻이다.
expect 보다는 ncftpget 이 좀더 빠르고 간단하므로 ncftpget을 사용하기를 권장한다.

ii. RSYNC를 이용한 2차 백업

ncftpget 과 비슷한 방법으로 rsync 라는 패키지를 이용하는 방법이 있는데, 이는 rpm 으로도 기본 제공되므로 쉽게 이용이 가능하며 백업뿐만이 아니라 Clustering 으로도 적용 가능하다.

설정방법은
(1) 먼저 데이터 서버의 /etc/inetd.conf 에
rsync stream tcp nowait root /usr/bin/rsync rsyncd --daemon를 추가 설정후
killall -HUP inetd 로 INETD를 재설정한다.
rsync 는 포트 873을 이용하며 --daemon은 데몬 모드로 작동한다는 뜻이다.

(2) /etc/rsyncd.conf 에 아래와 같이 설정한다.

```
[mail]
path = /backup
comment = mail 서버
uid = root
gid = root
```

```
use chroot = yes
read only = yes
hosts allow = 211.11.22.33
max connections = 1
```

[mail] rsync의 서비스명으로 어떠한 이름을 설정하여도 된다.
단, 백업서버에서 데이터 전송시 필요한 설정이다.
path 데이터를 동기화할(즉, 백업으로 전송할) 디렉토리를 설정한다.
여기에서는 /backup 이하의 디렉토리에 대해 2차 백업을 할 것이다.
comment 서비스명에 대한 설명이다.
uid 파일을 전송하는 사용자의 uid로 기본값은 nobody이다.
gid 파일을 전송하는 사용자의 gid로 기본값은 nobody이다.
use chroot 위의 path를 최상위 디렉토리로 사용하여 이외의 디렉토리는 접근할 수 없도록
하는 설정으로 보안상 필요하므로 꼭 설정한다.
read only 읽기전용모드로 설정한다는 의미이다.
hosts allow 기본값은 모든 접속을 받아들이므로 보안을 유지하려면 반드시 백업서버의 IP 를 설정하여야 한다.
max connections 백업서버에서 전송시 동시접속자수를 뜻한다. 1정도면 적당하다.
timeout 클라이언트에서의 접근시 타임아웃시간이다.

아래는 데이터를 백업할 2차 백업서버에서의 설정 내용이다.

```
#!/bin/sh ..... (1)
cd /mail_backup ..... (2)
rm -rf * ..... (3)
/usr/bin/rsync -av data.clunix.com::mail /mail_backup ..... (4)
chown root.root * ..... (5)
chmod 700 * ..... (6)
```

(1) shell 스크립트임을 정의한다.

(2) 백업을 받을 디렉토리인 /mail_backup 으로 이동후
(3) 기존의 백업받았던 데이터를 삭제한다.
(4) rsync을 이용해 서버에 접속하여 데이터를 받아온다.
이때 -a는 아카이브 모드로서 파일의 속성이나 퍼미션 및 소유권등의 정보
를 보존하여 동기화한다는 의미이며 -v 는 verbose 의 의미로 동기화의 진행상황을
자세하게 보여준다. 데이터 서버에서 서비스명을 [mail] 로 했으므로
data.clunix.com::mail 로 설정되었으며 동기화한 데이터는 /mail_backup 이라는
디렉토리에 저장된다.
(5),(6)은 위에서 설명되었으며 보다 세부적인 rsynhc에 대한 내용은
HOWTO를 참고하기 바란다.

2차 백업을 좀더 효율적으로 하기 위해서는 데이터 서버에 이더넷 카드를 2장 설치하여 eth0 은 일반 서비스로 이용하고 eth1은 사설 IP를 할당하여 백업서버와 Direct 로 연결하여 백업시 이용하는 방법도 있다. 이 경우에는 2차 백업시 별도의 이더넷 카드로 전송이 되므로 백업 작업시 발생하는 이더넷 카드에서의 트래픽이 서비스에 영향을 주지 않아 병목현상을 피할수 있고, 사설 IP를 이용하므로 백업서버의 보안도 일정 정도 강화할 수 있다.
단, 이러할 경우 백업서버는 데이터서버와 근거리에 위치하여야 하고, 사설 IP를 할당하므로써 백업서버에 직접 접근시 사설 IP로 접근하여야 하는 단점이 있기는 하다.

④ 복구의 방식

이제 어렵게 백업한 파일을 복구할 필요가 생기게 되었다.
백업 파일로부터의 복구방법은 그리 복잡하지 않은데, 복구시에는 백업때와는 달리 c 옵션대신 eXtract 의 의미인 x를 사용하는 것만 주의하면 된다.
예를 들어보면,
tar zxvpf /backup/etc.tar.gz 와 같이 할 경우 현재의 디렉토리에
압축된 파일에 들어있는 모든 파일을 복구하며 p옵션을 주었으므로 원래 파일의 소유자와 퍼미션도 그대로 복구된다.

백업 파일에 있는 모든 파일을 복구하는 것이 아니라 특정 디렉토리나 파일에 대해서만 복구하려면 아래와 같이 하면 된다.

tar zxvpf /backup/home.tar.gz etc/passwd home/clunix/public_html

현재의 디렉토리가 /home/user 라면 위와같은 경우 현재의 디렉토리에 원래의 /etc/passwd 파일이 /home/user/etc/passwd 파일로 복원이 되고 원래의 /home/clunix/public_html 이하의 파일들이 /home/user/home/clunix/public_html 로 복원되는 것이다. 특정 파일을 지정할 경우에는 특정파일이 복구되고, 디렉토리를 지정할 경우에는 디렉토리 이하의 디렉토리 및 파일들이 복원된다. 이때 주의할 점은 백업이 풀릴 파일을 지정시 /etc/passwd 와 같이 원래의 절대경로를 설정하면 백업에서 풀리면서 현재의 설정 파일에 백업본이 그대로 overwrite를 되므로 주의하여야 한다는 점이다. 따라서 복구시 / 를 입력하지 않는다.

⑤ 백업 및 복구 시 주의점

(1) 데이터의 변화주기 및 경중을 고려하여 백업주기를 다르게 설정한다.

백업할 데이터가 홈페이지 관련 데이터일 경우 일반적인 HTML 이나 관련 GIF등의 이미지 파일들은 자주 변경이 되지는 않으며 주로 작업자의 PC에서 작업후 서버에 업로드를 하므로 작업자의 PC에 저장되어 있는 경우가 많다. 그러므로 이의 데이터들은 매일 또는 수시로 백업을 받을 필요는 없으며 변경된 내용만을 백업하거나 일정주기마다 백업을 하면 된다. 그리고 백업이 동작시 일반 데이터뿐만 아니라 zip 이나 asf 또는 mpeg 등과 같이 파일자체가 압축되어 있는 형식의 경우에는 파일자체의 사이즈가 크고 백업시 많은 부하를 유발하게 되므로 이러한 파일의 경우에는 꼭 백업할 필요가 없다면 데이터의 경중을 고려하여 백업에서 제외하거나 백업일정을 적당히 조정할 필요가 있다.
요즘은 회원제 운영이 보편화되면서 대부분의 홈페이지에서 각종 데이터베이스를 연동하여 사용하는 경우가 많은데, 데이터 베이스의 경우 회원의 ID / PW 등의 각종 회원 데이터가 보관되고 정보가 자주 변경되므로 수시로 백업을 할 필요가 있다.
각 데이터베이스 프로그램의 경우 별도의 백업 명령어(msqldump., mysqldump등) 가 있으므로 tar 가 아닌 dump 명령어를 이용하는 것도 권장할 만 한다.

(2) 1차 백업시 백업 파티션은 별도의 디스크로 구성한다.

시스템 인스톨시 하나의 물리적인 디스크를 여러 파티션으로 논리 분할하여 이용하는 경우가 많은데, 이때 백업 파티션의 경우에는 반드시 별도의 물리적인 파티션 분할을 이용하여 별도로 디스크를 구성하는 것이 중요하다. 이는 만약 데이터가 보관된 하드 디스크가 물리적인 결함으로 복구가 불가능할 경우 이 디스크에 있던 데이터는 물론이고 복구시 필요한 백업까지도 함께 복구가 불가능하여 백업으로서의 의미가 상실되기 때문이

다.
일반적인 데이터는 빠른 연산이 필요하므로 SCSI 를 이용하고 백업디스크는 비용부담이 적은 고용량의 IDE 방식도 무난하다.

(3) 백업된 데이터에 대한 보안설정도 중요하다.
일반적으로 백업 스크립트는 root 소유로 작동하므로 백업된 파일이 생성될 때에는 root 소유로 백업 화일이 생성된다. 그러나 일반적으로 umask 는 022 로 설정되어 있으므로 생성된 백업파일은 644로 저장된다. 644일 경우 아래와 같이 일반 사용자가 백업된 파일에 접근하여 심지어는 접근이 불가능한 shadow 파일에도 접근이 가능하게 되어 보안상 문제가 될 수 있다.
따라서 백업 파티션을 별도로 구성하여 백업이 끝난 후에는 반드시 백업 파티션을 umount하여 일반 사용자가 접근할 수 없도록 하고 백업을 시작하기 전에 mount를 하면 될 것이다. 또는 생성된 파일의 권한을 700 정도로 제한하여도 된다. 그리고 백업작업 중간에 일반 사용자가 백업 파일을 복사할 수도 있으므로 백업시작 스크립트에 umask를 066으로 설정하여 생성되는 파일의 소유권을 600 으로 하고 백업이 끝난후에 umask를 다시 022로 재설정하여 일반 사용자의 비정상적인 접근을 차단할 수 있다.

```
[user@www user]$id

uid=500(user) gid=500(user)
[user @www user]$ cat /etc/shadow
cat: /etc/shadow: 허가 거부됨
[user@www user]$ ls -la /backup/etc.tar.gz
-rw-r--r--    1 root   root      954039 12월  12 04:22 /backup/etc.tar.gz
[user@www user]$ cp /backup/etc.tar.gz .
[user@www user]$ tar zxvfp etc.tar.gz etc/shadow
etc/shadow
[user@www user]$ cat etc/shadow
root:$1$V.120Zso3$cD/aUqQr2EBY0:11295:0:99999:7:-1:-1:134540356
bin:*:11266:0:99999:7:::
daemon:*:11266:0:99999:7:::
```

(4) 2차, 3차 백업서버는 가급적 네트워크에서 격리할 필요가 있다.

만약 1차 백업이 되고 있는 데이터 서버가 해킹을 당해 데이터가 유실되었다면 2차나 3차 백업이 되고 있는 서버의 데이터로 복원을 하여야 한다. 그런데, 만약 2차,3차 백업서버도 기존의 서버와 같은 설정으로 되어 있어 같은 방식으로 해킹을 당해 데이터가 유실되었다면? 해결책이 없는 것이다. 따라서 최후의 보루라 할 수 있는 2차,3차 백업 서버는 기존의 서버와 환경설정을 다르게 하고 관리자 암호등 접근권한도 다르게 설정 할 필요가 있다.
그리고 백업서버는 일체의 제공 서비스 없이 백업만을 담당하므로 불필요한 서비스는 띄워놓을 필요가 없으며 백업이 작동할 때 이외에는 네트워크에 연결되어 있을 필요가 거의 없으므로 백업이 끝난 후에는 네트워크에서 격리할 필요가 있는데, 이는 ipchains 등을 이용하여 자체 방화벽을 구성하여 외부에서의 접근을 일체 차단할 수도 있겠지만 다음과 같이 아예 네트워크에서 격리를 할 수도 있다.

백업 시작전 : /etc/rc.d/init.d/network start 로 Netwok 연결
백업 종료후 : /etc/rc.d/init.d/network stop 로 Network 차단.

2차,3차 백업서버는 네트워크에서의 격리뿐만 아니라 물리적으로도 격리 할 필요가 있다.
화재나 천재지변등 피치 못할 사정으로 사고 발생시 백업서버가 원 DATA가 위치한 서버와 공동으로 위치할 경우 돌이킬 수 없는 문제가 벌생할 수 있으므로 가능한대로 백업 서버는 다른 건물로, 같은 건물의 경우라면 다른층에 위치하는 것이 좋다.
부득이 같은 층에 있다면 멀리 위치시켜 예기치 못한 서고의 가능성을 분산시키는 것이 필요하다.

(5) 백업 담당자를 지정, 운영한다.

보안담당자도 없는데 백업담당자가 필요하겠냐고 생각할 수 있겠지만, 전담자까지는 없더라도 담당자는 지정을 하여 정기적으로 체크를 함으로써 보다 책임성 있고 신뢰성 있는 백업체계를 유지할 필요가 있다. 백업이 되고 있는줄 알고 복구를 할 일이 있어 백업서버를 살펴 보니 corn 데몬이 죽어 있다거나 설정한 스크립트가 실행권한이 없어 실행이 되고 있지 않는 경우도 있으니 백업 담당자가 수시로 체크를 하여야 한다. 백업 체크일지를 두거나 정기적으로 체크하는 스크립트를 만들어 자동으로 체크여부를 알려주는 것도 좋은 방법이 될 것이다.

(6) tar 이외 cp를 이용하는 방법도 고려할 만 하다.
TAR를 이용하여 Packing 이나 압축없이 cp를 이용하는 방법도 있는데,
이는 일주일에 1회정도
cp -ap /etc /backup/ 와 같이 /etc 디렉토리 전체를 풀복사한후
cp -aufp /etc/ /backup/etc/ 로 풀복사 이후 새롭게 생성이나 변경, 수정된 파일이나

디렉토리만 복사하는 방법도 있다.

(7) 라우터와 스위칭의 설정파일도 백업을 잊지 않는다.

서버뿐만이 아니라 라우터와 스위칭의 Configuration File 도 설정 변경시 백업을 할 필요가 있다. 특히 라우터나 스위칭의 경우 설정 변경시 원치 않던 설정까지 변경되는 경우가 있고 라우터의 장애시에는 라우터 밑단에 연결되어 있는 모든 네트워크가 영향을 받으므로 반드시 백업을 받아둘 필요가 있다. 다만 설정이 자주 변경되지는 않으므로 정기적으로 백업을 할 필요는 없고, 설정이 변경될 때만 그때 그때마다 설정파일을 Copy & Paste 로 복사를 해 두면 된다.

⑥ 실전 백업 예제

이 백업 스크립터는 총 3개의 스크립터가 필요하다.
shell script 에 대한 준비가 부족하다 생각하면 스크립터 경로를 변경하지 말길 바란다.

```
/root/bin/backup-sh
/root/bin/sysbkp
/root/mesg/backupdisplay
```

이 스크립터는 매일 중요 데이터를 백업 하되 이틀간의 자료만을 보관하는 정책의 스크립터입니다.

vi /root/bin/backup-sh

```
#!/bin/sh
```

```
rdate -s time.bora.net

bkdir_name=`date +%Y%m%d`

# /dev/hdd 하드는 백업 하드를 의미한다. 즉 백업 하드 디스크가 계속 mount 되어있다면
# 실제 해커가 들어왔을때 백업 파티션에 있는 백업 자료까지도 손상시킬수 있기 때문에
# 백업 바로 전에 mount 시켜 주고 백업 작업이 완료 된후 반드시 umount 시켜 주도록 한다.

mount -t ext3 /dev/hdd   /backup

# 기존의 백업 자료를 삭제 한다. 즉 오늘 백업을 하면 실제 2일 전 자료를 삭제 하는 것이다.

rm -rf /backup/bk_*-1

# 어제 자료를 다른 이름으로 하루 더 보관하기 위한 방법으로 Name Label 에다 -1 를 붙여
# 디렉토리 명을 변경 한다.
mv /backup/* /backup/bk_${bkdir_name}-1

# 실제 시스템 백업 리스트 입니다. 각 해당 백업이 저장될 폴더를 생성함.

mkdir /backup/bk_${bkdir_name}
mkdir /backup/bk_${bkdir_name}/home
mkdir /backup/bk_${bkdir_name}/named
mkdir /backup/bk_${bkdir_name}/named/zone
mkdir /backup/bk_${bkdir_name}/passwd
mkdir /backup/bk_${bkdir_name}/apache
mkdir /backup/bk_${bkdir_name}/sendmail
mkdir /backup/bk_${bkdir_name}/mysql
mkdir /backup/bk_${bkdir_name}/real
mkdir /backup/bk_${bkdir_name}/etc
mkdir /backup/bk_${bkdir_name}/cron
mkdir /backup/bk_${bkdir_name}/root

# 사용자의 개별 데이터 백업 ( 홈디렉토리 백업 )

tar cvfpz /backup/home_a_h.tar.gz /home/[a-h]*
tar cvfpz /backup/home_i_p.tar.gz /home/[i-p]*
```

```
tar cvfpz /backup/home_q_z.tar.gz /home/[q-z]*

# 설정 파일 및 시스템 중요 파일 백업

# DNS 설정 파일 백업

cp -a /etc/named.conf /backup/bk_${bkdir_name}/named
cp -a /etc/hosts /backup/bk_${bkdir_name}/named
cp -a /etc/resolv.conf /backup/bk_${bkdir_name}/named
cp -a /var/named/* /backup/bk_${bkdir_name}/named/zone

# 메일 설정 파일 백업

cp -a /etc/mail/* /backup/bk_${bkdir_name}/sendmail

# 시스템 계정 설정 파일 백업

cp -a /etc/passwd* /backup/bk_${bkdir_name}/passwd
cp -a /etc/shadow* /backup/bk_${bkdir_name}/passwd
cp -a /etc/group* /backup/bk_${bkdir_name}/passwd

# Apache web service 설정 파일 백업

cp -a /usr/local/apache/conf/httpd.conf /backup/bk_${bkdir_name}/apache

# real media server 설정 파일 백업

cp -a /usr/local/real/rmserver.cfg /backup/bk_${bkdir_name}/real

# 시스템 자동설정(cron)에 대한 백업

cp -a /var/spool/cron/root /backup/bk_${bkdir_name}/cron

# 사용자 디스크 쿼터에 대한 설정 백업

cp -a /home/aquota.user* /backup/bk_${bkdir_name}/home

# 시스템 전체 설정 파일 백업
```

```
tar czvfp /backup/bk_${bkdir_name}/etc/etc.tgz /etc

# Mysql 데이터 백업 ( 근래에는 /usr/local/mysql/data 디렉토리를 사용하는 경우 있음 )

tar czvfp /backup/bk_${bkdir_name}/mysql/mysqldata.tgz /usr/local/mysql/var

# root 디렉토리 백업

tar czvfp /backup/bk_${bkdir_name}/root/root.tgz /root

# 1일 동안 수정된 파일 점검 -> 보안적인 의미

find /home  ₩! -type d -atime +1 -exec ls -l {} ₩; >
 /backup/bk_${bkdir_name}/home/today_access_file

umount /backup
```

sysbkp 스크립터는 위의 backup-sh 스크립터를 실행하고 백업 중에 발생한 오류 및 백업 결과 보고를 관리자에게 메일로 해주는 스크립터이다.

vi /root/bin/sysbkp

```
#!/bin/sh

data=`date +%Y%m%d`

/root/bin/backup-sh 2> /root/mesg/err_all
cat /root/mesg/err_all | grep -v "Removing leading" > err_chk
/root/mesg/backupdisplay > /root/mesg/backupnotice
mail -s "$data 백업 결과 통보" alang@clunix.com < /root/mesg/backupnotice
```

backupdisplay 스크립터는 백업 결과 보고서의 내용을 보고서 형태로 정리해 주는 스크립터이다.

vi /root/mesg/backupdisplay

```
#!/bin/sh

bk_data=`date +%Y년%m월%d일%H시%M분`

echo -n "
------------------------------------------------------
*                                                    *
*       WELCOM TO SYSTEM MANAGER WEB SITE !!!!!!          *
*       SECURITY !! FAST !! GOOD !! WEBSERVER !!          *
*                                                    *
*       ------- http://www.clunix.com   --------          *
*                                                    *
------------------------------------------------------

♥♥♥  ♡♥:*:.....:*:♥♡♥:*:.....:*:♥♡♥
:*:.....:*:♥♡  ♥:*:.....:*:♥♡

////////////////////// System user notice ////////////////////

"
if [ ! -s /root/mesg/err_chk ]
then
    echo "시스템 백업이 $bk_data 에 무사히 마쳤습니다."
else
    echo "$bk_data 이루어진 백업에 오류가 발생하였습니다."
fi

echo -n "
//////////////////////////////////////////////////////////

////////////////// what's error ////////////////////////////
```

```
"
echo
/bin/cat /root/mesg/err_chk

echo -n "
////////////////////////////////////////////////////////

:*:♥♥:*:.....:*:♥♡♥:*:.....:*:♥♡♥:*:..
♡♥:*:.....:*:♥♡♥:*:.....:*:♥  ♡♥:*:....*:
```

;; 워낙 옛날에 만든 스크립터이지만 지금에 와서 보니 너무 유치하다.

마지막으로 root 계정에서 /root/bin/sysbkp 스크립터를 자동 실행 하게 cron 에 등록한다.

crontab -e

```
0 3 * * * /root/bin/sysbkp
```

/etc/rc.d/init.d/crond restart

5. 문제 발생 시 응급 복구 방법

파일 시스템이 깨졌거나 해킹등의 문제로 정상 부팅이 안되는 경우 복구 방법 입니다.
실제 리눅스 커널로 부팅을 하는 것이 가장 중요하다. 부팅을 해서 시스템 상태를 점검 해야 이 다시 설치를 할지 아님 일부 중요 파일의 백업을 이용하여 복구를 할지 결정이 가능하다. 먼저 시스템 부팅이 안된다. 이제 부팅을 시켜 보자

① 긴급 부팅 시키기

1장에서 설정한 bootloader 설정 내용도 참조하세요.

i. Linux 배포 CD를 통한 부팅

단 이때는 기본적으로 시스템의 하드의 파티션 중 / 파티션의 디바이스 명을 알고 있어야 한다. 1장에서 언급한바와 같이 그래서 초기 셋팅 후 시스템 정보를 보관하는것은 상당히 중요하다.

Linux Install CD 를 넣고 부팅 한다. boot 프롬프트에 다음내용을 기재한다.

boot : linux initrd= root=/dev/hda2 -> /dev/hda2 가 실제 / 파티션의 device name

이는 Linux Install CD 에 있는 기본 커널로 시스템을 부팅하여 현 시스템의 root mount point (/) 에 접근하는 것이다.

이경우는 IDE 하드에 리눅스가 설치 된 경우 주로 사용한다. SCSI 의 경우 Install CD 커널에 SCSI 모듈 정보가 포함안되어져 있어서 root device 를 못찾는다면서 실패할 가능성이 많다.

주로 Lilo 가 깨졌을때 많이 사용한다.

ii. rescue mode booting

역시 Linux Install CD 를 이용하여 부팅할때 boot 프롬프트에서 rescue 라는 옵션과 같이 부팅

boot : linux rescue

그럼 text 설치 과정과 비슷하게 넘어가다가 bash prompt (#) 바로 떨어진다.

하드디스크에 크게 문제가 없을 경우에는 /mnt/sysimage 폴더에 system root 파티션이 mount 되어져 있을것이다.

chroot /mnt/sysimage

하면 마치 정상적으로 부팅 한것 처럼 mount 가 되어진다.
이상태에서 정검하면 됩니다. 진짜 중요한 파일에 문제가 있어 이것이 안된다고 하면

수동으로 mount 하여 중요파일의 손상 정도를 파악한다.
rescue 방식은 주로 SCSI 하드 디스크 일 경우 많이 사용함.

주로 Lilo 깨졌을때, 파일 시스템 깨졌을때, 완전히 망가 졌을 때

iii. single mode 로 부팅하기

파일 시스템에 문제가 발생하여 파일 시스템을 체크하기 위해 부팅하는 모드로 LILO Prompt 가 뜨면 ..

Lilo : linux single

위와 같이 부팅한다. 그런 후 안정스럽게 파일 시스템을 점검 및 복구 한다.

이밖에 파일 시스템 복구 목적으로 긴급 부팅 하는 방법으로는

lilo: linux init=/bin/sh

도 있다.

② 부팅 디스켓 만들기

먼저 부팅디스켓으로 사용할 커널 이미지를 선택한다.
그리고 선택한 커널 이미지가 root 장치(root mount point)가 재대로 잡혀 있는지 확인한다.

```
# rdev /boot/vmlinuz
Root device /dev/hda1
```

만일 루트 장치가 현재 잡혀 있는거랑 틀리다면 아래와 같이 수정한다.

```
# rdev /boot/vmlinuz /dev/hda3
#
```

이제 깨끗한 플로피 디스켓 하나 준비하고 포맷한다.

fdformat /dev/fd0
dd if=/boot/vmlinuz of=/dev/fd0 bs=8192

③ 파일시스템 점검 수리

fsck -t [type] [device]

fsck 파일 점검은 가능한 umount 한 후에 실행한다. 하지만 / 을 mount 하지 않은 면 당근 부팅이 제대로 될수 없기에 / 를 체크할때는 플로피로 부팅을 하던지 아님 single mode 로 부팅한후 체크 한다.

체킹후 mount -t -w -o remount /

해주면 된다.

④ RPM 패키지 상태 초기화 하기

누가 해킹 혹은 실수로 RPM 패키지의퍼미션이나 권한등을 변경하여 정상적인 작동을 안할 경우 이를 설치 시 상태로 복구 하는 방법이다.

rpm --setperms -a

이로써 "철학이 있는 리눅스 설치" 의 모든 내용을 마치도록 하겠습니다.

To construct

internet service system

in Redhat Linux 9

(리눅스 기반 인터넷 서비스 환경 구성)

클루닉스 기술부 2차 세미나 자료

클루닉스 서 진우(alang@clunix.com)

==========

Redhat 9 Linux Internet Service System 구축하기

==========

* 주요 내용

---

1. Bind 를 이용한 DNS 서비스
2. Sendmail 을 이용한 SMTP/POP3 메일 서비스
3. Proftpd 를 이용한 FTP 서비스
4. RealServer 를 이용한 real media Streaming 서비스
5. Apache + PHP + Tomcat(JSP) + mysql 를 이용한 웹 서비스

---

* 사용된 패키지

OS : Redhat Linux 9
bind
sendmail
proftpd
realserver
apache
php
mysql
tomcat
tomcat-connectors
Zend
zlib
tiff
libpng
clibpdf
pdflib
swf

freetype

jpeg

libungif

gd

---

1. BIND9 를 이용한 DNS 서버 구축 하기

[root@ns /etc]# rndc-confgen > /etc/rndc.conf

[root@ns /etc]# cat /etc/rndc.conf

```
Key "rndc-key" {
        algorithm hmac-md5;
        secret "n+h/daGNTmmEKimn25/h4g==";
  };

  options {
        default-key "rndc-key";
        default-server 127.0.0.1;
        default-port 953;
  };
```

[root@ns root]# vi /etc/rndc.conf

```
key "syszone-key" {
    algorithm hmac-md5;
    secret "e05yfVyzq/Y8HN+fmk7g9w==";
};

options {
    default-key "syszone-key";
    default-server 127.0.0.1;
    default-port 953;
};
```

rndc-key 값을 자기 사이트에 맞게 수정한다.(보안상 문제)

[root@ns root]# vi /etc/named.conf

```
options {
        directory "/var/named";
};

controls {
        inet 127.0.0.1 allow { localhost; } keys { "syszone-key"; };
};


key "syszone-key" {
                algorithm hmac-md5;
                secret "e05yfVyzq/Y8HN+fmk7g9w==";
};

logging {
        category lame-servers { null; };
        category unmatched { null; };
        category network { null; };
        category notify { null; };
        };


zone "." IN {
        type hint;
        file "named.ca";
};

zone "localhost" IN {
                type master;
                file "localhost.zone";
                allow-update { none; };

};


zone "41.238.211.in-addr.arpa" IN {
```

```
        type master;
        file "syszone.rev";
};

zone "0.0.127.in-addr.arpa" IN {
        type master;
        file "named.local";
        allow-update { none; };
};

zone "syszone.co.kr" IN {
    type master;
    file "syszone.zone";
    allow-update { none; };
};
```

[root@ns root]# vi /var/named/syszone.zone

```
$TTL    86400
@               IN      SOA     ns.syszone.co.kr. root.ns.syszone.co.kr.  (
                                        1997092700 ; Serial
                                        28800      ; Refresh
                                        900        ; Retry
                                        360000     ; Expire
                                        3600 )     ; Minimum
                IN      NS      ns.syszone.co.kr.
                IN      MX      10      mail
                IN      A       211.238.41.180
;
ns              IN      A       211.238.41.180
;
www             IN      CNAME   ns
mail            IN      CNAME   ns
db              IN      CNAME   ns
ftp             IN      CNAME   ns
```

[root@ns root]# vi /var/named/syszone.rev

```
$TTL    86400
@       IN      SOA     ns.syszone.co.kr. root.ns.syszone.co.kr.  (
```

```
                                        1997022700 ; Serial
                                        28800      ; Refresh
                                        14400      ; Retry
                                        3600000    ; Expire
                                        86400 )    ; Minimum
                IN      NS      ns.syszone.co.kr.
        IN  MX  10  mail.syszone.co.kr.

180     IN      PTR     syszone.co.kr.
180     IN      PTR     mail.syszone.co.kr.
```

[root@ns root]# cat /etc/resolv.conf

```
search syszone.co.kr
nameserver 211.238.41.180
```

;;; search 의 의미 ;;;

resolv.conf 파일에 search domain 을 적어 주는 이유는
여기에 적힌 도메인은 보통 자기 자신의 도메인을 주로 적고
여기에 적힌 도메인의 sub name 은 그냥 hostname 만으로도
찾을수 있게 한다.

[root@ns root]# nslookup mail

Server: 211.238.41.180
Address: 211.238.41.180#53

mail.syszone.co.kr canonical name = ns.syszone.co.kr.
Name: ns.syszone.co.kr
Address: 211.238.41.180

이제 설정한 도메인을 검색해 보자

[root@ns root]# nslookup syszone.co.kr

Server: 211.238.41.180

Address: 211.238.41.180#53

Name: syszone.co.kr
Address: 211.238.41.180

[root@ns root]# nslookup 211.238.41.180

Server: 211.238.41.180
Address: 211.238.41.180#53

180.41.238.211.in-addr.arpa name = mail.syszone.co.kr.
180.41.238.211.in-addr.arpa name = syszone.co.kr.

;; 주의 점 ;;

DNS 서버가 방화벽 안쪽에 있을 경우 반드시 tcp 53 port 와 udp 953 port 를 open 시켜 두어야 한다.

2. Sendmail 로 메일 서버 구축 하기

인터넷이 대중화되면서 편지를 주고받는 것보다 E-Mail(전자우편)을 주고 받는 것이 일상적인 일이 되었습니다. 이번 주 테마에서는 E-Mail(전자우편)이 어떻게 전달되는지의 개념과 메일서버로 가장 많이 사용되고 있는 Sendmail의 설치방법과 사용법에 대해서 알아보도록 하겠습니다.

E-mail(전자우편)과 메일서버

Sendmail의 설치와 사용법을 배우기 전에 E-Mail(전자우편)이 어떠한 경로를 통해 전달되고 어떻게 사용되는지에 대해 살펴보겠습니다. 메일서버란 인터넷에서 E-Mail(전자우편)을 주고 받는 기능을 수행하는 서버를 말하며, E-Mail(전자우편)은 크게 두 종류의 프로그램과 프로토콜에 의해 전달됩니다. 여기서 두 종류의 프로그램이란 MUA(Mail User Agent)와 MTA(Mail Transfer Agent)를 말합니다. MUA로는 사용자가 직접 메일을 작성하거나 보낼 때 사용하는 넷스케이프 메신저, 마이크로소프트 아웃룩, 유도라 등의 프로그램이 있고, MTA는 실제로 메일을 전송해주는 Sendmail, Qmail등의 프로그램이 있습니다.

아래의 그림과 같이 Sendmail이 설치된 메일서버는 SMTP(Simple Mail Transfer Protocol) 프로토콜을 사용하여 메일을 전송하고, 호스트로부터 받은 메일을 클라이언트로 전달할 때는 POP3 또는 IMAP등의 프로토콜을 사용합니다. 예를 들어 aroma@mail.linuxul.com이라는 메일주소를 가진 사람이 다른 서버에 계정을 가진 nea@yahoo.co.kr이라는 친구에게 메일을 보낸다고 가정하면, 아래와 같은 과정을 거쳐 메일이 전달됩니다.

1. aroma@mail.linuxul.com이라는 사람이 넷스케이프 메신저와 같은 MUA 프로그램에서 보낸 메일은 먼저 SMTP 프로토콜을 통해 mail.linuxul.com 메일서버에 전달됩니다.

2. mail.linuxul.com에서 대기하고 있던 MTA데몬(Sendmail)은 메일을 메일큐 디렉토리(/var/spool/mqueue)에 임시 저장한 후 SMTP 프로토콜을 통해 다시 yahoo.co.kr메일서버로 전송하고 큐디렉토리에 저장했던 메일을 삭제합니다. yahoo.co.kr에서는 sendmail 데몬이 대기하고 있다가 mail.linuxul.com 메일서버가 보내온 메일의 도착지가 자신의 것인지 확인 한 후 맞으면 메일박스(/var/mail 디렉토리)에 nea라는 파일로 저장합니다.

3. nea@yahoo.co.kr의 사용자는 마이크로소프트 아웃룩이나 유도라와 같은 MUA를 통해 메일을 확인하며, 이때에는 POP3나 IMAP등의 프로토콜을 통해 메일이 전송됩니다.

Sendmail

Sendmail은 메일서버로 가장 많이 사용되고 있는 MTA프로그램으로 공식사이트는 http://www.sendmail.org입니다. 대부분의 리눅스 배포판에 기본적으로 설치가 되어 있으므로 자신의 시스템에 맞는 환경설정을 한 후 사용하시면 됩니다. Sendmail의 최신 버전은 sendmail 8.11.2이며 아래의 사이트에서 다운받을 수 있습니다.

Sendmail 공식 미러사이트 - ftp://ftp.sendmail.org
Sendmail 미러사이트 - http://www.sendmail.org/mirrors.html
국내 미러사이트 : ftp://ftp.linux.co.kr/server/sendmail

POP3와 IMAP

앞에서도 설명했듯이 넷스케이프 메신저나 마이크로소프트 아웃룩과 같은 클라이언트 프로그램에서 메일을 가져오려면 메일서버에 POP3 또는 IMAP등의 데몬이 설치되어 있어야 합니다. POP3(Post Office Protocol version 3)와 IMAP(Internet Message Access Protocol)데몬은 사용자의 인증절차를 거친 후 메일을 클라이언트에 보내는 기능을 수행합니다. RedHat 리눅스 배포판에서는 IMAP과 POP 데몬을 하나의 패키지(UW IMAP)로 제공하고 있어 누구나 쉽게 설치할 수 있으며 최신버전은 아래의 사이트에서 다운받으시면 됩니다.

UW IMAP 공식 홈페이지 - http://www.washington.edu/imap/
UW IMAP 미러사이트 - ftp://ftp.cac.washington.edu/imap
국내 미러사이트 - ftp://ftp.linux.co.kr/server/imap

Sendmail과 IMAP 설치하기

대부분의 리눅스 배포판에는 Sendmail이 기본적으로 내장되어 있으므로 최신버전이나 보안문제로 인해 다시 설치해야 하는 경우를 제외하면 소스를 사용해 설치하는 경우는 드물고 배포판에 들어있는 RPM 패키지를 사용해 설치합니다.

Sendmail 설치하기

와우 리눅스 7.0 까치버전을 기본 OS로 하여 Sendmail을 설치하는 방법에 대해 알아보겠습니다. 먼저 CD-ROM을 마운트한 후 /RedHat/RPMS 디렉토리에 있는 sendmail을 확인합니다. 기본적으로 제공하는 Sendmail은 Sendmail-8.11.0입니다.

```
# mount /dev/cdrom
# cd /mnt/cdrom/RedHat/RPMS
# ls sendmail*

sendmail-8.11.0-8.i386.rpm
sendmail-cf-8.11.0-8.i386.rpm
sendmail-doc-8.11.0-8.i386.rpm
```

Sendmail에 관련된 rpm패키지를 모두 설치합니다.

```
# rpm -Uvh sendmail*
sendmail                ##############################
sendmail-cf             ##############################
sendmail-doc        ##############################
```

Sendmail을 실행하기전에 /etc/services 파일을 열고 smpt 프로토콜에 주석처리가 되어 있는지 확인한 후 Sendmail 서버를 실행합니다.

```
# cat /etc/services | grep smtp
smtp25/tcp      mail
smtp25/udp      mail
smtps       465/tcp # SMTP over SSL (TLS)
```

Sendmail 서버가 정상적으로 실행되었는지 확인하기 위해 telnet을 사용해 25번 포트(smpt)로 접속해봅니다. 아래와 같은 메시지가 출력되면 Sendmail이 정상적으로 실행된 것입니다.

Sendmail 서버가 정상적으로 실행되었는지 확인하기 위해 telnet을 사용해 25번 포트(smpt)로 접속해봅니다. 아래와 같은 메시지가 출력되면 Sendmail이 정상적으로 실행된 것입니다.

```
# telnet 0 25
Trying 0.0.0.0...
Connected to 0 (0.0.0.0).
Escape character is '^]'.
220 xxxxxx.net ESMTP Sendmail 8.11.2/8.11.2; Sat, 21 Jul 2001 14:28:12 +0900
```

만약 레드헷 버젼이 7.1 이상부터는 relay 보안 기능이 강화 되어서 초기 셋팅상태에서는 내부 사용자 끼리만 메일을 보내고 받을수 있다. 원격지에서는 이 서버를 통해서 메일을 주고 받을수가 없다. 일단..새로 sendmail.cf 파일을 만들어야 한다.

cd /etc/mail

/etc/mail/sendmail.mc 파일을 수정한후 m4 를 이용하여 새로운 sendmail.cf
파일을 만든다. (redhat 7.1 부터 변경된 사항)

vi sendmail.mc

.
.
아래 설정 부분 제일 앞에 "dnl" 을 붙여준다.
dnl DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')
.
.
---

m4 sendmail.mc > /etc/sendmail.cf
/etc/rc.d/init.d/sendmail restart

확인 하도록 하자.

telnet 25
Trying x.x.x.x...
Connected to 0.
Escape character is '^]'.
220 xxxxxxx.net ESMTP Sendmail 8.11.2/8.11.2; Wed, 27 Jun 2001 01:29:51 +0900

Washington IMAP설치하기

Washington IMAP 역시 RPM 패키지로 제공하고 있으므로 CD-ROM을 마운트한 후 쉽게
설치할 수 있습니다. 먼저 CD-ROM에 있는 UW IMAP 패키지를 확인 해 보겠습니다.

 # mount /mnt/cdrom
 # cd /mnt/cdrom/RedHat/RPMS
 # ls imap*

imap-2000-3.i386.rpm

Washington IMAP는 imap-2000-3이라는 파일명으로 패키징되어 있으며, POP3와 IMAP 데몬이 모두 설치됩니다.

```
# rpm -Uvh imap-2000-3.i386.rpm
```

이제 설치한 imap과 pop3 데몬을 클라이언트에서 접속할 수 있도록 실행시켜야 합니다. 먼저 /etc/services 파일을 열어 pop3와 imap의 주석을 제거하고 아래와 같이 설정되었는지 확인합니다.

```
pop3            110/tcp   pop-3  # POP version 3
pop3            110/udp   pop-3
imap2   143/tcp   imap     # Interim Mail Access Proto v2
imap2   143/udp   imap
```

레드햇 7.0 버전부터는 슈퍼데몬이 inetd에서 xinetd로 변경되었고, inetd.conf 설정파일이 xinetd.conf 파일로 변경되었습니다. xinetd의 기본적인 설정파일들은 /etc/xinetd.d에 위치하므로 설치한 ipop3와 imaps 파일을 열고 disable = no를 추가하여 아래의 그림과 같이 설정합니다.

```
# vi /etc/xinetd.d/ipop3
# default: on
service pop3
{
                disable = no
                socket_type = stream
                protocol = tcp
                wait = nowait
                user = root
                server = /usr/sbin/ipop3d
```

```
                log_on_success += USERID
                log_on_failure += USERID
}
```

다음에는 슈퍼 데몬을 다시 재시작합니다.

```
# /etc/rc.d/init.d/xinetd restart
```

이제 마지막으로 설치한 POP3와 IMAP이 제대로 가동되었는지 확인해 보겠습니다. telnet을 이용하여 25번 포트(POP3)와 143번 포트(IMAP)로 접속한 후 아래와 같이 테스트 해봅니다. 여기까지 제대로 되었다면 설치는 성공한 것입니다.

이밖에 pop 프로그램으로 많이 사용하는 qpopper 의 설치에 대해 알아보자.

pop 데몬으로 qpopper 를 사용할것이기 때문에 최신 소스를 다운 받는다.
다운 받은 소스를 풀고 그 디렉토리로 이동한 다음...

```
# ./configure   --enable-specialauth --enable-servermode --enable-shy
# make
# cp popper/popper /usr/local/lib
```

inetd.conf 파일을 redhat 7.1 부터는 쓰지 않으므로 xinetd 방식으로 전환 한다.
먼저 inetd.conf 파일을 /etc 밑에 만들고 pop 설정 부분을 적어준다.

```
# vi inetd.conf
-------------------------------------------------------------------------

pop-3     stream   tcp        nowait   root       /usr/local/lib/popper

-------------------------------------------------------------------------

# inetdconvert -d /etc/xinetd.d pop-3
# /etc/rc.d/init.d/xinetd restart
```

확인 하자.

telnet 0 110
Trying 0.0.0.0...
Connected to 0.
Escape character is '^]'.
+OK ready

이것으로 센드메일에 관련되어 smtp, pop 모든 셋팅을 마쳤다.

Sendmail 설정하기

Sendmail에서 가장 중요하고도 어려운 부분이 sendmail.cf 파일의 설정입니다. O'Reilly사에서 Sendmail에 대한 전문서적이 나올 정도로 방대하고 다양한 기능을 가지고 있으며 사용법도 매우 다양합니다. Sendmail을 사용하기 위해 기본적으로 알아야 할 설정 파일들에 대해 살펴보겠습니다.

/etc/sendmail.cf

Sendmail의 가장 중요한 설정파일로 /etc 또는 /etc/mail 디렉토리에 자동으로 설치되어 있습니다. Sendmail.cf에 대한 자세한 내용은 Bryan Costales 와 Eric Allman이 집필한 O'Reilly의 Sendmail을 참고하시거나 이상로님의 홈페이지 http://trade.chonbuk.ac.kr/~leesl/mail/을 참고하시기 바랍니다. 여기서는 간단히 메일서버를 관리하기 위한 몇 가지 설정에 대해서만 다루겠습니다.

Fw/etc/mail/local-host-names

메일을 수신할 호스트 이름을 명시한 파일의 위치를 설정합니다.

FR-o /etc/mail/relay-domains

relay-domains파일에는 Relay를 허용할 호스트의 이름을 설정합니다. 주석으로 처리하면 모든 IP에 대해서 Relay가 허용되므로 스팸메일과 같은 문제가 발생할 수도 있으므로 주의하시기 바랍니다.

DnMAILER-DAEMON

Sendmail 서버가 에러메시지를 보내야 할 경우 보낸 사람의 이름을 결정합니다. 잘못된 메일이 되돌아 온 경우 FROM : Mail Delivery Subsystem

&ltMAILER-DAEMON>과 같은 메시지를 보신적이 있을 것입니다.

Kaccess hash /etc/mail/access

Relay를 허용하거나 거부할 특정 IP와 도메인을 설정하는 파일입니다.
relay-domains보다 사용이 편리하므로 많이 사용됩니다.
O ForwardPath=$z/.forward.$w:$z/.forward

여러 개의 메일을 가진 경우, 특정 메일계정으로 들어온 메일을 다른 메일로 곧바로 보내주는 포워딩파일을 설정합니다. 사용자의 홈 디렉토리에 .forward라는 파일을 만들고 포워딩시킬 메일 주소를 입력하면 됩니다.

O MaxMessageSize=1000000

메일의 최대 크기를 결정합니다. 주석을 제거하면 설정한 크기(Byte단위)보다 큰 메일은 전송할 수 없게 됩니다. 지금 써준 1000000은 1메가로 제한한 메일 용량입니다.

O QueueDirectory=/var/spool/mqueue

큐 디렉토리를 설정합니다.

O Timeout.queuereturn=5d

메일을 보내려는 호스트에 문제가 생기면 메일은 큐 디렉토리에 저장됩니다. Sendmail 서버는 쌓인 메일을 상대방 호스트에 보내기 위해 주기적으로 접속을 시도하며, 일정한 기간이 지나면 메일을 다시 발송한 사람에게 되돌려 보냅니다. Sendmail이 메일을 보내려고 시도하는 기간을 설정하는 옵션으로 5d는 5일을 의미합니다.

O Timeout.queuewarn=4h

큐 디렉토리에 쌓인 메일이 지정한 시간안에 전송되지 못할 경우 메일을 보낸 사람에게 경고 메일을 보냅니다. 기본값은 4h로 4시간안에 전송되지 못하면 보낸 사람에게 경고의 메일을 보냅니다.

Mlocal,
P=/usr/bin/procmail, F=lsDFMAw5u:/|@qSPfhn9,
S=EnvFromL/HdrFromL, R=EnvToL/HdrToL,
T=DNS/RFC822/X-Unix,

A=procmail -Y -a $h -d $u

사용자계정에 대문자가 있는 경우에도 메일을 받을 수 있도록 설정하려면 Mlocal로 시작하는 부분을 찾아 F= 부분에 'u'를 추가합니다.

/etc/mail/access
스팸메일을 방지하기 위해 Relay를 허용할 호스트의 IP와 도메인을 설정하는 매우 중요한 파일입니다. Relay에 대한 자세한 내용은 [Relay에 대하여]를 참고하시기 바랍니다.

먼저 vi에디터나 emacs를 사용해 /etc/mail/access파일을 열고 릴레이를 허용하거나 거부할 IP 주소를 아래와 같이 입력합니다.

```
[root@zzang911 /root]# cat /etc/mail/access
# Check the /usr/share/doc/sendmail-8.11.2/README.cf file for a description
# of the format of this file. (search for access_db in that file)
# The /usr/share/doc/sendmail-8.11.2/README.cf is part of the sendmail-doc
# package.
#
# by default we allow relaying from localhost...
localhost.localdomain        RELAY
localhost        RELAY
127.0.0.1        RELAY
xxx.xxx.xxx.xxx                                        RELAY
spam.com                                                    REJECT
```

위는 localhost 나 xxx.xxx.xxx.xxx 의 특정 IP 에서는 릴레이를 허용하나 spam.com 에서는 오는 모든 메일은 릴레이를 거부한다는 뜻이다.

이와 같이 허용이나 거부 대역을 지정후 access.db 파일을 갱신시켜주면 바로 적용이 된다.

```
[root@zzang911 /root]# makemap hash /etc/mail/access < /etc/mail/access
```

[Relay에 대하여]

그럼 간단하게 Relay에 대한 개념을 알아보겠습니다.
예를 들어 A라는 사람은 IP주소가 203.243.88.21이라는 컴퓨터에서 aroma@mail.linuxul.com이라는 메일을 사용한다고 가정하고, B라는 사람은 IP주소가 168.211.106.34인 컴퓨터에서 nea@yahoo.co.kr이라는 메일을 사용한다고 가정합니다.
B가 A에게 메일을 보내면 메일은 203.249.88.21로 가는 것이 아니라 mail.linuxul.com서버의 /var/mail 디렉토리 밑에 aroma라는 파일로 복사가 됩니다. 그러면 A는 MUA 프로그램을 사용해 메일을 확인할 수 있지요. 그런데 여기서 중대한 문제점이 생겨나게 됩니다. 아무나 주소를 맞게 보내면 그냥 /var/mail에 쌓이게 되는 것이죠. 그래서 하드디스크가 메일로 꽉 차버리거나, 네트워크 전송량의 증가로 네트워크가 마비되는 경우가 생겨나게 됩니다. 이 문제를 막기 위해서 고안된 방법이 바로 Relay라는 방법입니다.

Relay에는 두 가지 방식이 있습니다.
o 첫 번째는 메일을 보내는[송신] 컴퓨터의 제한
o 두 번째는 메일을 받는[수신] 컴퓨터의 제한

Relay로 주로 쓰이는 방법은 메일을 확인하는 컴퓨터는 제한하지 않고 메일을 보내는 것을 제한하는 것입니다. 즉, 위의 그림과 같이 /etc/mail/access파일에 203.243.88의 C클래스를 허용 가능하게 해주면 203.243.88.21처럼 허용 그룹에 들어 있는 컴퓨터에서는 메일을 보내고/받을 수가 있지만, 그룹에 들지 않은 168.211.106.34라는 컴퓨터에서는 메일을 확인할 수만 있습니다. 또한 spam.com이라는 도메인에 속한 호스트에서 오는 메일은 보내기와 받기가 모두 거부됩니다.
/etc/mail/relay-domains을 사용해도 됩니다. 하지만, 차이점은 relay-domains을 사용하면sendmail데몬을 다시 실행시켜 주어야 하지만 /etc/mail/access파일을 사용하면 다시 실행할 필요 없이 makemap hash /etc/mail/access < /etc/mail/access라고 실행만 시켜주면 됩니다.

/etc/mail/local-host-names
메일을 수신할 호스트의 이름을 입력하는 파일로, Sendmail 8.9.x 버전이하에서 사용되었던 sendmail.cw 파일의 명칭이 Sendmail 8.10.x 버전부터는 local-host-names로 변경되었습니다. Sendmail 서버는 이곳에 적힌 호스트의 이름으로 메일이 들어오면 더 이상 다른 서버로 메일을 보내지 않고 자신의 메일박스에 저장합니다. 아래와 같이 메일서버의 호스트이름을 입력하거나 여러 개의 가상호스트를 사용한다면 모두 입력해야 합니다.

etc/aliases

특정 사용자에게 온 메일을 다른 사람에게 보내주거나, 메일링리스트를 작성해야 하는 경우에 사용되는 파일로 보안에 주의하여 사용해야 합니다. aliases 파일을 열어보면 시스템 계정들이 아래와 같이 root로 alias되어 있습니다. 또한 특정 사용자를 다른 사용자로 alias 시킬 수도 있습니다.

메일링 리스트 작성하기

메일링리스트란 동일한 메일을 여러 사람에게 보내야 할 경우 사용되며, 그룹을 지정해 구성원들의 목록을 써주거나 특정 파일을 지정하는 방법 등으로 사용됩니다. 예를 들어 aroma, bulpeng, bibi와 같은 사용자들을 test1라는 그룹으로 지정해 보겠습니다.

vi편집기로 /etc/aliases 파일을 열고 아래의 라인을 추가한 후 저장합니다. 앞으로 test1에게 메일을 보내면 지정된 모든 사용자들에게 메일이 도착하게 됩니다.
test1: aroma, bulpeng, bibi

다른 하나는 특정파일을 지정해 주는 방법으로 include 지시자를 사용한 후에 파일의 경로와 파일이름을 입력합니다.
test2: :include:/etc/maillist/test2

cat /etc/maillist/test2
aroma
bulpeng
bibi

파일을 수정한 후에는 newaliases 명령으로 aliases.db를 만들어 줍니다.
newaliases
/etc/aliases: 40 aliases, longest 28 bytes, 434 bytes total

이제 test1 또는 test2로 메일을 보내면 지정된 사용자들(aroma, bulpeng, bibi)이 모두 메일을 받아볼 수 있습니다.

sendmail 기본 보안

설정파일 속성 변경하기

chattr +i /etc/mail/sendmail.cf or /etc/sendmail.cf

chattr +i /etc/mail/local-host-nameds

chattr +i /etc/mail/aliases or /etc/aliases

chattr +i /etc/mail/access

Sendmail deamon을 실행하는 쉘을 바꾸어주기

Sendmail의 보안을 위해서는 보통 프로그램의 /bin/sh을 smrsh program으로 바꾸어 주어야 합니다. /bin/sh로 실행이 되었을 경우에 해킹의 우려가 있으므로, 이를 제한하기 위해서 Sendmail 프로그램의 실행을 제한할 필요가 있기 때문입니다.

하지만, 레드햇의 RPM으로 설치했을 경우에는 Default로 설정이 되어 있습니다.

/etc/mail/sendmail.cf 또는 /etc/sendmail.cf 파일에서

Mprog,

P=/bin/sh, F=lsDFMoqeu9, S=EnvFromL/HdrFromL,

R=EnvToL/H drToL, D=$z:/,

T=X-Unix/X-Unix/X-Unix,

A=sh -c $u

의 굵게 쓴 파일을 아래와 같이 바꾸어주고

Mprog,

P=/usr/sbin/smrsh, F=lsDFMoqeu9, S=EnvFromL/HdrFromL,

R=EnvToL/H drToL, D=$z:/,

T=X-Unix/X-Unix/X-Unix,

A=smrsh -c $u

sendmail.cf 파일을 저장하고 아래와 같이 sendmail deamon을 재시작합니다.

/etc/rc.d/init.d/sendmail restart

인증되지 않은 사용자의 Sendmail Deamon사용을 제한하기
Sendmail에는 Anti-Spam Feature를 포함하고 있습니다. 이 기능을 제대로 사용하기 위해서는 /etc/mail/sendmail.cf 또는 /etc/sendmail.cf파일에서 옵션을 바꾸어 주어야 합니다.
privacy flags
O PrivacyOptions=authwarnings, goaway

Sendmail에서 goaway 옵션을 사용할 경우에 SMTP의 EXPN, VRFY명령어를 제한하게 됩니다. 이로 인해서 VERB명령어도 더불어 제한하게 됩니다. 처음 SMTP에 접하시는 분들은 이 명령어에 대해서 생소할 수 있으므로 약간의 설명을 덧붙이겠습니다. STMP에 접속하여서 MAIL을 전달하기 위해서는 여러 가지 명령어를 사용하게 되어 있습니다.
아래와 같이 접속 후 HELP 명령어를 타이핑하면 HELO, EHLO, MAIL, RCPT, DATA, RSET, NOOP, QUIT, HELP, VRFY, EXPN, VERB ETRN, DSN, AUTH, STARTTLS등의 사용 가능한 명령어를 보실 수 있습니다.

자세한 사용법과 내용은 HELP &ltCommand>하시면 알 수 있습니다.
HELP AUTH <- 입력
214-2.0.0 AUTH mechanism [initial-response]
214-2.0.0 Start authentication.
214 2.0.0 End of HELP info

EXPN aroma <- 입력 (Mail list의 관계된 메일 계정을 보여줄 수 있는 명령어입니다.)
250 2.1.5 &ltsweetness@linux.co.kr>
VRFY aroma <- 입력(Alias에 관계된 메일 계정을 보여줄 수 있는 명령어입니다.)
250 2.1.5 &ltaroma@mail.linuxul.com>

보안설정 후에는 아래와 같은 메시지가 나타납니다.
EXPN aroma <- 입력
502 5.7.0 Sorry, we do not allow this operation

mailq command의 제한하기
/usr/bin/mailq 명령어를 제한하기 위해서는 /etc/mail/sendmail.cf or /etc/sendmail.cf파일에서 옵션을 바꾸어 주어야 합니다.
privacy flags
O PrivacyOptions=authwarnings, goaway, restrictmailq

또한, queue디렉토리를 root권한으로 막아주고 재시작합니다.
#chmod 0700 /var/spool/mqueue
#/etc/rc.d/init.d/sendmail restart

mailq명령어를 실행시키면 아래와 같은 메시지가 나타납니다.
aroma$ mailq
You are not permited to see the queue

Queue 프로세스를 root권한으로 제한하기
Queue프로세스를 root권한으로 제한하기 위해서는 /etc/mail/sendmail.cf 또는 /etc/sendmail.cf파일에서 아래와 같이 옵션을 바꾸어 줍니다.
privacy flags
O PrivacyOptions=authwarnings, goaway, restrictmailq,
restrictqrun

파일을 저장후 재시작합니다.
#/etc/rc.d/init.d/sendmail restart

SMTP Greeting Message를 수정하기
Sendmail의 Version 정보라든지 여러 가지 이유로 Connected중에 보여주는 아래와 같은 메시지들을 바꾸어주므로, 불필요한 정보를 숨길 수 있습니다.
[root@mail /root]# telnet mail.linuxul.com 25

Trying 211.106.100.89...
Connected to mail.linuxul.com.
Escape character is '^]'.
220 mail.linuxul.com ESMTP Sendmail 8.11.0/8.11.0; Wed, 21 Feb 2001
22:50:51 +0900

/etc/mail/sendmail.cf 또는 /etc/sendmail.cf파일에서 옵션을 바꾸어 주어야 합니다.
SMTP initial login message (old $e macro)
O SmtpGreetingMessage=$j Sendmail $v/$Z; $b

를 아래와 같이 수정합니다.
SMTP initial login message (old $e macro)
O SmtpGreetingMessage=$j

파일을 저장후 Sendmail을 재시작합니다.
#/etc/rc.d/init.d/sendmail restart

수정후에는 아래와 같은 메시지가 출력됩니다.
telnet mail.linuxul.com 25
220 localhost.localdomain ESMTP

Sendmail 모니터링

[Sendmail의 프로세스 상태보기]
ps aux|grep sendmail
root 21094 0.0 0.3 3244 1504 ? S 08:43 0:00 sendmail: accepti

[Sendmail의 mail 상태를 모니터링하기]

[root@ns /root]# mailstats

```
Statistics from Wed Jan 23 04:02:04 1980
M    msgsfr   bytes_from    msgsto      bytes_to   msgsrej msgsdis   Mailer
5    361145    14357435K     89312      3671483K     16983 0   esmtp
8    236772    11571389K    661786     20775066K      8687 0   local
=============================================================
T    597917    25928824K    751098     24446549K     25670 0
```

[메일큐값 모니터링 하기]

```
[root@ns /root]# mailq
                     Mail Queue (5 requests)
--Q-ID-- --Size-- -----Q-Time----- ------------Sender/Recipient------------
LAA21970        2375 Sat Jul 21 11:40 MAILER-DAEMON
        8BITMIME     (Deferred: Connection refused by web.kdlp.org.)

QAA29807       11059 Fri Jul 20 16:58 MAILER-DAEMON
        8BITMIME     (Deferred: Connection refused by tmail.simmani.com.)

OAA05202        3153 Wed Jul 18 14:43 MAILER-DAEMON
        8BITMIME     (Deferred: Connection refused by web.kdlp.org.)

PAA19364* (no control file)
```

praliases명령은 현재 메일 시스템의 Aliases상태를 보여줍니다.

```
# praliases
```

3. Proftpd Server 구축하기

ftp://ftp.oops.org 에서 proftpd 를 다운 받는다

```
# rpm -Uvh proftpd-1.2.9-xkr.ixx.rpm --force
# rpm -Uvh proftpd-doc-1.2.9-xkr.ixx.rpm --force

[root@ns root]# vi /etc/proftpd/proftpd.conf

-----------------------------------------------------------------------
ServerName              "ALANG1 FTP Server For CLUNIX"
```

```
#
ServerType              standalone
#ServerType             inetd
DefaultServer               on
ServerAdmin             root@localhost

# 요즘에는 inverse domain 이 거의 지원되지 않으니, 이 설정은 필수라고 생각
# 하고 수정하지 않는다.
#
UseReverseDNS                   off
IdentLookups                    off

# Server 에 접속했을 경우 보내주는 Defualt 메세지를 설정한다. Off 로 지정
# 하였을 경우 Proftpd Server Ready ServerName 이 출력된다.
#
ServerIdent                     On      "ProFTPd Account Server ready .. "

# User login 을 했을시에 user 들이 자신의 홈상위 디렉토리들을 마음대로 돌
# 아다니지 못하게 chroot() 를 설정한다. group 별로 설정을 하게 되며 "!"는
# 제외하라는 의미를 가지게 된다.
#
DefaultRoot             ~   !wheel

# ServerType이 standalone이면 이 항목의 주석을 풀어 줘야 한다. inetd로 작
# 동을 시킬시에는 /etc/services에서 port를 지정한다.
#
Port                    21

# root login 을 허락할지의 여부를 지정한다. PAM modules를 사용한다면 먼저
# /etc/proftpd/ftpusers에서 root를 삭제해야 한다.
# http://www.oops.org/SERVICE/jsboard/read.php?table=ProftpdTip&no=10 참조
#
RootLogin               off

# Global section은 proftpd의 전체적인 설정에 모두 적용시킨다.
<Global>
  # group 과 world writable 로 부터 새로운 dir 과 file 들을 생성하는 것을
```

```
  # 막기위하여 기본적으로 umask는 022로 설정을 한다.
  Umask                 022

  # ftpusers file을 이용하여 PAM 인증을 하기를 원하면 이 지시자의 값을 on
  # 으로 한다. default 값은 off 이다.
  AuthPAMAuthoritative          on

  # service를 시작하고 마칠 시간을 24시간 표기법으로 지정을 한다. 이 설정
  # 은 Korea User Group 의 time limit 패치가 적용이 되어 있어야 사용이 가
  # 능하다.
  #
  # UpTime              10
  # DownTime                23

  # Server 의 Ftpd 시간을 지역시간으로 고정한다. on 으로 했을 경우 GMT 시
  # 간을 표시 하기 때문에 한국의 경우 9시간의 오차가 발생한다.
  TimesGMT              off

  # 1.2.1 이하 버젼에서의 버그를 위한 설정
  DenyFilter                ₩*.*/
</Global>

# 회선의 Bandwidth를 특정 속도로 제한을 한다. 단위는 Byte per Sec 이다.
# 다운로드와 업로드를 16KB/s (128Kb/s) 로 제한
TransferRate                RETR,STOR     102400

# 최대 접속 인원수를 지정한다.
MaxClients              10

# 하나의 호스트로 부터 동시에 접근할 수 있는 수를 지정한다. 아래의 기본
# 설정으로는 하나의 호스트에서 한번의 접근만 허용한다.
MaxClientsPerHost           3 "Sorry, %m connection allow per one host"

# 하나의 계정 ID 로 동시에 접근할 수 있는 호스트 수를 지정 한다. 아래의
# 기본 설정으로는 하나의 계정 하나 호스트에서만 접근만 허용한다. 하나의
# 계정에 하나의 접근만을 허락하려면 위의 MaxClientsPerHost의 값과
# MaxHostsPerUser의 값이 둘다 1이면 된다.
#MaxHostsPerUser            1 "Sorry, %m hosts allow per one user"
```

```
# 접속 대기시간을 설정한다. user 가 접속후 아무 작동도 안할때 일정 시간후
# 에 접속이 종료되게 한다
TimeoutIdle             900
TimeoutNoTransfer       900
TimeoutLogin            300

# DeferWelcome 는 client가 인증을 하기 전에 servername을 display하는 것을
# 방지한다.
DeferWelcome            off

# 'welcome.msg 는 login 시에 보여지고, 'message' 는 각 하위 디렉토리에 접
# 속 했을때 보여지게 된다.
DisplayLogin            /etc/proftpd/welcome.msg
DisplayFirstChdir       .message

# DoS(Denial Of Service) 공격을 막기 위해, 자식 process 의 maximun number
# 를 30 으로 설정한다. 만약 30이상의 접속을 허락할 필요가 있다면 간단하게
# 이 치수를 증가 시키도록 한다. 이것은 오직 standalone mode 에서만 가능하
# 다. inetd mode 에서는 service 당 maximun number를 제한 하는 것을 허락하
# 는 inetd server에서 설정을 해야 한다.(xinetd 역시 마찬가지 이다)
# MaxInstances            30

User                    nobody
Group                   nobody

# 일반적으로 file들을 overwrite를 가능하게 한다.
<Directory /*>
  AllowOverwrite        on
</Directory>

# ls 명령어의 -a option으로 hidden file을 볼수 있게 한다.
# LsDefaultOptions      "-a"

<Anonymous ~ftp>
  User                  ftp
  Group                 ftp
  # /etc/shells 에 등록되어 있는 shell 이 지정되어 있는 유저들만 로그인이
```

```
  # 가능하게 한다. ftp user 의 shell 이 보통 /bin/false 로 지정이 되어 있
  # 으므로 이 값을 off 로 해준다.
  RequireValidShell       off

  # 익명 접근시의 패스워드 체크정도 여부를 결정을 한다. AnonPassType 지시
  # 자는 Korean User Group 의 독자적인 패치이다.
  # none              -> 아무런 체크를 하지 않는다.
  # trivial           -> 패스워드에 @ 문자가 존재하는지만 체크한다.
  # complete-email -> 패스워드가 완전한 형태의 이메일 주소형식을 가지는지
  #                   체크한다.
  AnonPassType            none

  # 익명 접근시 사용하지 못하게 할 패스워드를 정규표현식으로 설정한다. 이
  # 지시자는 Korean User Group 의 독자적인 패치이다.
  #
  #AnonRejectPasswords       ^(IEUser|mozilla|username|test)@?

  # 익명 접근을 할때 특정 password를 지정할수 있다. 단 위의 User 지시자의
  # name이 passwd file에 등록이 되어져 있어야 한다. 이 지시자가 on일 경우
  # 이메일 주소로 login을 할수 없다.
  #
  # 이 지시자가 설정되면, AnonPassType 이 무시된다.
  #
  # AnonRequirePassword       on

  # 링크된 실 경로 출력 여부
  # ShowSymlinks          off

  # User name "ftp"로 anonymous login을 할수 있도록 한다
  UserAlias           anonymous     ftp

  <Limit LOGIN>
    Order allow,deny
    Allow from 211.238.41.
    Allow from 192.168.123.
    Deny from all
  </Limit>
```

```
DisplayLogin            welcome.msg
DisplayFirstChdir       .message

# 최대 접속 인원수를 지정한다.
MaxClients              10

# 하나의 호스트로 부터 동시에 접근할 수 있는 수를 지정한다. 아래의 기본
# 설정으로는 하나의 호스트에서 한번의 접근만 허용한다.
MaxClientsPerHost       3 "Sorry, %m connection allow per one host"

# 소유권이 root인 file이나 directory들을 보여주지 않는다
#HideUser               root

# 그룹권한이 root인 file이나 directory들을 보여주지 않는다
#HideGroup              root

# upload/download 비율을 지원한다.
# http://www.oops.org/SERVICE/jsboard/read.php?table=ProftpdTip&no=9 참조
#
# Ratios                on
# HostRatio             foobar.net 100 10 5 100000

# 서버를 시간대로 운영하는 것을 지정한다. 아래의 예는 오후 3 시 부터 오
# 후 6시까지만 서버의 접속을 가능하게 한다.
#
# Korean User Group 의 time limit 패치가 적용이 되어 있어야 한다.
#
# UpTime                15
# DownTime              18

<Limit WRITE>
  DenyAll
</Limit>
<Limit READ>
  DenyALL
 </Limit>
```

```
  # anonymous 계정으로 접근할수 있는 디렉토리별로의 허가권 설정..
  # 일단은 모든 디렉토리에 쓰기 권한은 주지 않는설정이다.
<Directory *>
      <Limit WRITE>
      DenyAll
      </Limit>
</Directory>

# incoming 디렉토리에 읽기권한을 주기않고..upload 권한만 부여.
<Directory upload>
      <Limit READ>
      DenyAll
      </Limit>
      <Limit STOR MKD>
      AllowAll
      </Limit>
</Directory>

<Directory pub>
   <Limit READ>
      AllowALL
   </Limit>
   <Limit STOR MKD>
      Order allow,deny
      Allow from 211.238.41.
      Allow from 192.168.123.
      Deny from all
   </Limit>
</Directory>
</Anonymous>

---------------------------------------------------------------------------

[root@ns root]# ncftp -u alang syszone.co.kr
Connecting to 211.238.41.180...
ProFTPd Account Server ready ..
Logging in...
```

```
Password requested by 211.238.41.180 for user "alang".

    Password required for alang.

Password:

 ######################################################################
 #                                                                    #
 #                    ProFTPD Korea User Groups                       #
 #                     http://proftpd.oops.org                      #
 #                                                                    #
 #       This file is printed when account ftp user login          #
 #       from /etc/proftpd/welcome.msg                                #
 #                                                                    #
 ######################################################################
User alang logged in.
Logged in to syszone.co.kr.
ncftp /home/alang >
```

4. Real Server 셋팅법

먼저 일정한 디렉토리에 realserver 프로그램을 옮겨놓는다.

그런뒤 실행파일인 g2p3-linux-c6.bin 에 실행 퍼미션을 준다.

```
[root@alang realserver]# chmod 755 g2p3-linux-c6.bin
```

실행 시킨다.

```
[root@alang realserver]# ./g2p3-linux-c6.bin

Welcome to the realserver Setup for UNIX
Setup will help you get realserver running on your computer.

Setup will step you through the installation process by displaying
informational screens.  Please follow the navigational controls
below:
```

Key	Behavior
===	========
N	Next
P	Previous
X	Exit
F	Finish (Express Installation)

Each input requires the execution of the key above
followed by the [ENTER] key. Enter [N]ext to continue:(그냥 엔터 친다.)

If a realserver license key file has been sent to you,
please enter its directory path below. If you have not
received a realserver license key file, then this server
will default to a Basic RealServer. /usr/local/src/apmj/RealPlayServer/license.txt
(여기는 라이센스키가 있는 위치를 적어준다.)

Installation and use of realserver requires
acceptance of the following terms and conditions:
Press [Enter] to display the license text... (엔터 친다.)

그럼 라이센스 정보가 나온다.

REALNETWORKS, INC.
END USER LICENSE AGREEMENT
REALNETWORKS BASIC SERVER VERSION G2
REDISTRIBUTION NOT PERMITTED
Software License for RealNetworks Basic Server
Version G2
IMPORTANT -- READ CAREFULLY: This RealNetworks
License Agreement ("License Agreement") is a legal
agreement between you (either an individual or an
entity) and RealNetworks, Inc. and its suppliers
and licensors (collectively "RN") for RN's Basic
Server Version G2 software ("Software"). You may
install only ONE copy of the Software. By
clicking on the "I Accept" button, installing,
copying or otherwise using the Software, you agree
to be bound by the terms of this License

Agreement. If you do not agree to the terms of
this License Agreement, click on the "I Do Not
Accept" button and/or do not install the Software.
ANY THIRD PARTY SOFTWARE, INCLUDING ANY NON-RN
PLUG-IN, THAT MAY BE PROVIDED WITH THE SOFTWARE IS
INCLUDED FOR USE AT YOUR OPTION. IF YOU CHOOSE TO
USE SUCH SOFTWARE, THEN SUCH USE SHALL BE GOVERNED
BY SUCH THIRD PARTY'S LICENSE AGREEMENT, AN
ELECTRONIC COPY OF WHICH WILL BE INSTALLED IN THE
"realserver" DIRECTORY ON YOUR COMPUTER, UPON
INSTALLATION OF THE SOFTWARE.

1. SOFTWARE OWNERSHIP. This is a license
agreement and NOT an agreement for sale. This
Software and the Documentation is proprietary to
RN, and is protected by the copyright and other
intellectual property laws of the United States
and international treaties. The Software is
licensed, not sold. RN continues to own the
Software, and other content provided or
--More--(13%)

중간 생략 .....

rbitration Association. The arbitrator's award
shall be binding and may be entered as a judgment
in any court of competent jurisdiction. This
License Agreement will not be governed by the
United Nations Convention of Contracts for the
International Sale of Goods, the application of
which is hereby expressly excluded.

Basic Server Gold EULA 11-17-981

1
Choose "Accept" to accept the terms of this
license agreement and continue with realserver setup.
If you do not accept these terms, enter "No"
and installation of realserver will be cancelled.
(라이센스 동의을 묻는곳이다...그냥 엔터)

Enter the complete path to the directory where you want
to be installed. You must specify the full
pathname of the directory and have write privileges to
the chosen directory: /usr/local/real
(리얼서버가 인스톨되는 디렉토리를 적어준다.)

Please enter a username and password that you will use
to access the web-based RealSystem Administrator, the
RealSystem monitors, and RealSystem live encoders:
Username: admin
(설치후 웹상에서 관리할 수 있는데 사용할 관리자 아이디를 적는다.)

Password: (패스워드)

Please enter a port on which realserver will listen for
PNA connections. These connections have URLs that begin
with "pnm://": (Default: 7070)
(PNA 접속에 port를 적어준다. - 디폴트 권장)

Please enter a port on which realserver will listen for
RTSP connections. These connections have URLs that begin
with "rtsp://": (Default: 554)
(RTSP 접속에 사용할 port - 디폴트 권장)

Please enter a port on which realserver will listen for
HTTP connections. These connections have URLs that begin
with "http://": (Default: 8080)

(HTTP 접속에 사용할 port - 디폴트 권장 -> 만일 tomcat 설치 시에는 다른
포트를 사용한다. 8880)

realserver will listen for RealSystem Administrator
requests on the port shown. This port has been
initialized to a random value for security. Please
verify now that this pre-assigned port will not interfere
with ports already in use on your system; you can
change it if necessary. (Default: 1278) 1975

(웹상에서 관리자 페이지로 접속할때 사용할 포트 번호)

You have selected the following realserver configuration:

Admin Username: admin
Admin Password: muchun
Monitor Password: muchun
Encoder Username: admin
Encoder Password: muchun
PNA Port: 7070
RTSP Port: 554
HTTP Port: 8080
Admin Port: 1975

Enter [F]inish to begin copying files, or [P]revious
to go back to the previous prompts: F

(환경설정사항을 확인한다. F 를 눌려서 계속 진행)

Copying RealNetworks program files.....

realserver installation is complete.
The RealSystem Administrator allows you to configure
and maintain realserver through an intuitive
web-based interface. Please note that realserver
must be running in order to use the Administrator.
Would you like to start realserver now and launch
the RealSystem Administrator? (Default: Yes)

(리얼서버 설치 완료 . 서버를 실행하고 관리자 모드를 실행하겠냐는
메세지 입니다. - 엔터)

설치가 완료 되었습니다.
웹에서 http://도메인:관리자포트번호/admin/index.html
으로 접속 하면 관리 도구가 나옵니다. 여기서 서비스를 관리하시면
됩니다.

부팅시 실행방법은 /etc/rc.d/rc.local 파일 안에 다음을 추가
/usr/local/real/Bin/rmserver /usr/local/real/rmserver.cfg -m 64 &

설정 추가는 웹에서도 할수 있고 바로 설정 파일을 수정할수도 있다.
vi /usr/local/real/rmserver.cfg

```
..
      </List>
      <List Name="syszone">
            <Var MountPoint="/syszone/"/>
            <Var ShortName="pn-local"/>
            <Var BasePath="/home/www/www"/>
      </List>
..
```

cd /home/www/www

vi rmtest.html

```
<html>
<body>

<a href=./realtest.ram>테스트</a>

</body>
</html>
```

vi realtest.ram

```
rtsp://211.238.41.180/syszone/realdb/Yesterday.rm
pnm://211.238.41.180/syszone/realdb/Yesterday.rm
```

http://211.238.41.180/rmtest.html 로 접속해서 확인할수 있습니다.

5. Mysql + Apache + PHP + JSP + Library

5.1. mysql setting

```
cd /usr/local/src/apmj
tar xzvf mysql-4.0.20.tar.gz
cd mysql-4.0.20
./configure --prefix=/usr/local/mysql --localstatedir=/usr/local/mysql/data --with-mysqld-user=mysql --with-charset=euc_kr
make
make install

/usr/local/mysql/bin/mysql_install_db
userdel -r mysql
groupdel mysql
groupadd mysql
adduser -g mysql -d /usr/local/mysql/data -s /bin/false mysql
chown -R mysql. /usr/local/mysql/data

cp /usr/local/src/apmj/mysql-4.0.20/support-files/my-medium.cnf /etc/my.cnf

vi /usr/local/mysql/share/mysql/mysql.server
-----------------------------------------------------------------
$bindir/safe_mysqld --datadir=$datadir --pid-file=$pid_file
```

위 행을 찾아서 뒤에 아래 옵션을 추가로 붙여준다.

```
--language=korean --safe-show-database &

-----------------------------------------------------------------

cp /usr/local/mysql/share/mysql/mysql.server /etc/rc.d/init.d/mysqld
ln -s /etc/rc.d/init.d/mysqld /etc/rc.d/rc3.d/S90mysqld

/etc/rc.d/init.d/mysqld start
```

5.2. lib setting

- zlib

rpm 으로 zlib 가 설치 되었다면 삭제

rpm -e --nodeps zlib
rpm -e --nodeps zlib-devel

http://www.gzip.org/zlib/ 에서
zlib 을 다운 받습니다.

cd /usr/local/src/apmj

tar zxvf zlib-1.2.1.tar.gz
cd zlib-1.2.1

./configure -s // libz.so 생성
make

./configure && make test && make install //libz.a 생성

cp libz.so* /usr/local/lib

cd /usr/local/lib
rm -rf libz.so
rm -rf libz.so.1
ln -s libz.so.1.2.1 libz.so
ln -s libz.so.1.2.1 libz.so.1

vi /etc/ld.so.conf

/usr/local/lib
/usr/lib

...

추가 함

ldconfig // 추가된 LIB 환경설정을 적용함

- tiff

cd /usr/local/src/apmj

tar xzvf tiff-v3.5.7.tar.gz
cd tiff-v3.5.7
./configure && make && make install

- libpng

rpm -e --nodeps libpng
rpm -e --nodeps libpng-devel

http://www.libpng.org/pub/png/libpng.html 에서 다운..

tar zxvf libpng-1.2.5.tar.gz
cd libpng-1.2.5
cp scripts/makefile.linux Makefile
make test && make install

- clibpdf

http://www.fastio.com 에서 다운..

cd /usr/local/src/apmj

mv clibpds202r1.tar.tar clibpds202r1.tar.gz
tar zxvf clibpds202r1.tar.gz
cd ClibPDF/source
mv Makefile Makefile.orig
cp Makefile.Linux Makefile
make lib && make install

- pdflib

http://www.pdflib.com 에서 다운..

```
tar zxvf pdflib-4.0.3.tar.gz
cd pdflib-4.0.3
./configure
make && make install

cd /usr/local/src/apmj
```

- swf

ftp://ftp.sgi.com/sgi/graphics/grafica/flash 에서 다운..

```
cd /usr/local/src/apmj
tar zxvf dist.99.linux.tar.Z
mkdir /usr/local/swf
cd dist
mkdir /usr/local/swf/include
mkdir /usr/local/swf/lib
mkdir /usr/local/swf/fonts
mkdir /usr/local/swf/psfonts
mkdir /usr/local/swf/bin
cp swf.h /usr/local/swf/include
cp libswf.a /usr/local/swf/lib
cp bin/* /usr/local/swf/bin
cp fonts/* /usr/local/swf/fonts
cp psfonts/* /usr/local/swf/psfonts
```

- freetype

http://freetype.sourceforge.net

```
cd /usr/local/src/apmj
tar xzvf freetype-2.1.3.tar.gz
cd freetype-2.1.3
```

./configure && make && make install

- jpeg

cd /usr/local/src/apmj
tar xzvf jpegsrc.v6b.tar.gz
cd jpeg-6b
./configure --enable-shared --enable-static
make && make test && make install

- libungif

rpm -e --nodeps libungif
rpm -e --nodeps libungif-devel

ftp://sunsite.unc.edu/pub/Linux/libs/graphics 에서 다운..

cd /usr/local/src/apmj
tar zxvf libungif-4.1.0.tar.gz
cd libungif-4.1.0
./configure && make && make install

- gd

rpm -e --nodeps gd
rpm -e --nodeps gd-devel

http://www.boutell.com/gd/http/ 에서 다운 ..

cd /usr/local/src/apmj

tar zxvf gd-2.0.21.tar.gz
cd gd-2.0.21
./configure
make && make install

- imap

rpm -qa |grep imap

설치 확인후 설치 되 있다면 삭제...

ftp://ftp.cac.washington.edu
에서 다운..

cd /usr/local/apmj

tar xzvf imap-2004.tar.Z
mv imap-2004 /usr/local/imap
make slx

//ssltype 관련 error 발생 시 ..
Makefile 의 SSLTYPE을 nopwd 에서 none 으로 변경한다.

vi Makefile
vi src/osdep/unix/Makefile

컴파일이 무사히 끝났다면...
cp imapd/imapd /usr/sbin
cp ipopd/ipop3d /usr/sbin
vi /etc/xinetd.d/imapd <=아래의 내용을 새로 적던지.. 복사 붙여넣기 한다..

```
service imap
{
disable = no
socket_type = stream
wait = no
user = root
server = /usr/sbin/imapd
log_on_success += DURATION USERID
log_on_failure += USERID
}
```

저장후 빠져나옴

vi /etc/xinetd.d/ipop3d <= 아래의 내용을 추가..

```
service pop3
{
disable = no
socket_type = stream
wait = no
user = root
server = /usr/sbin/ipop3d
log_on_success += USERID
log_on_failure += USERID
}
```

```
cp c-client/c-client.a /usr/lib
cp c-client/mail.h  /usr/local/include
cp c-client/rfc822.h /usr/local/include
cp c-client/linkage.h /usr/local/include
/etc/rc.d/init.d/xinetd restart
```

// test //

```
telnet localhost 110
telnet localhost 143
```

5.3 J2SDK 설치

```
cd /usr/local/src/apmj
rpm -Uvh j2sdk-1_4_1_01-fcs-linux-i586.rpm
cd /usr/java
ln -s j2re1.4.2_03 jre
ln -s j2sdk1.4.1_01 java
```

vi /etc/profile

```
-------------------------------------------------
.
.
# j2sdk config
PATH="$PATH:/usr/java/java/bin"
```

```
export JAVA_HOME="/usr/java/java"
-------------------------------------------------

source /etc/profile
```

// 확인 //

```
# java -version

java version "1.4.1_01"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.1_01-b01)
Java HotSpot(TM) Client VM (build 1.4.1_01-b01, mixed mode)
```

5.4 apache 1차 설치

```
cd /usr/local/src/apmj
tar xzvf apache_1.3.31.tar.gz
cd apache_1.3.31
./configure --prefix=/usr/local/apache
```

5.5 jakarta-tomcat 설치

```
cd /usr/local/src/apmj
tar xzvf jakarta-tomcat-4.1.30.tar.gz
mv jakarta-tomcat-4.1.30 /usr/local/tomcat

tar xzvf jakarta-tomcat-connectors-4.1.30-src.tar.gz
cd jakarta-tomcat-connectors-4.1.30-src/jk/native
./buildconf.sh
./configure --with-apache=/usr/local/src/apmj/apache_1.3.31
make && make install
```

// 확인 //

```
ls -al /usr/local/src/apmj/apache_1.3.31/src/modules
drwxr-xr-x       3 root         root              4096    5월  31 21:35 jk
```

5.6 PHP 설치 하기

cd /usr/local/src/apmj
tar xzvf php-4.3.7.tar.gz
cd php-4.3.7

./configure --with-apache=/usr/local/src/apmj/apache_1.3.31 --with-mysql=/usr/local/mysql --with-imap=/usr/local/imap --with-jpeg-dir=/usr/local/lib --with-png-dir=/usr/local/lib --with-gif-dir=/usr/lib --with-zlib-dir=/usr/local/lib --with-gd --with-freetype-dir=/usr/include/freetype2 --with-zlib --with-tiff-dir=/usr/local/lib --with-pdflib --with-cpdflib --with-gettext --with-swf=/usr/local/swf --with-mod_charset --with-language=korean --with-charset=euc_kr --with-xml --enable-ftp --enable-sockets --disable-debug --enable-system --enable-track-vars --enable-calendar --enable-magic-quotes

make && make install
cp libs/libphp4.a /usr/local/src/apmj/apache_1.3.31/src/modules/php4/
cp php.ini-dist /usr/local/lib/php.ini

5.7 apache 2차 셋팅

cd /usr/local/src/apmj/apache_1.3.31

./configure --prefix=/usr/local/apache --activate-module=src/modules/php4/libphp4.a --activate-module=src/modules/jk/lib/jk.a --enable-module=so --enable-rule=SHARED_CORE --enable-shared=max

make && make install

vi /usr/local/apache/conf/httpd.conf

간단히 ServerName 의 주석을 풀고 현 시스템의 도메인이나 IP 를 적어준다.
800 줄 근처에 AddType allplcation/x-httpd-php 로 시작하는 부분을 찾아서..
AddType application/x-httpd-php .html .php3 .php4 .inc .phtml .php .ph <= 요렇게
바꿉니다.

/usr/local/apache/bin/apachectl restart

5.8 Zend 설치 하기

```
cd /usr/local/src/apmj
tar xzvf ZendOptimizer-2₩[1₩].5.2-Linux_glibc21-i386.tar.gz
cd ZendOptimizer-2.5.2-Linux_glibc21-i386
./install
```

5.9 Apache httpd.conf 설정

```
ServerType standalone
ServerRoot "/usr/local/apache"

#LockFile /usr/local/apache/logs/httpd.lock

PidFile /usr/local/apache/logs/httpd.pid

ScoreBoardFile /usr/local/apache/logs/httpd.scoreboard

#ResourceConfig /usr/local/apache/conf/srm.conf
#AccessConfig /usr/local/apache/conf/access.conf
Timeout 300

KeepAlive On

MaxKeepAliveRequests 100

KeepAliveTimeout 15

MinSpareServers 20
MaxSpareServers 40

StartServers 20

MaxClients 150

MaxRequestsPerChild 0

#Listen 3000
#Listen 12.34.56.78:80

#BindAddress *
```

```
# Dynamic Shared Object (DSO) Support
# LoadModule foo_module libexec/mod_foo.so
LoadModule env_module              libexec/mod_env.so
LoadModule config_log_module    libexec/mod_log_config.so
LoadModule mime_module             libexec/mod_mime.so
LoadModule negotiation_module libexec/mod_negotiation.so
LoadModule status_module          libexec/mod_status.so
LoadModule includes_module       libexec/mod_include.so
LoadModule autoindex_module     libexec/mod_autoindex.so
LoadModule dir_module              libexec/mod_dir.so
LoadModule cgi_module              libexec/mod_cgi.so
LoadModule asis_module             libexec/mod_asis.so
LoadModule imap_module             libexec/mod_imap.so
LoadModule action_module          libexec/mod_actions.so
LoadModule userdir_module        libexec/mod_userdir.so
LoadModule alias_module           libexec/mod_alias.so
LoadModule access_module          libexec/mod_access.so
LoadModule auth_module             libexec/mod_auth.so
LoadModule setenvif_module       libexec/mod_setenvif.so
#LoadModule php4_module            libexec/libphp4.so
#LoadModule jk_module              libexec/libjk.so

ClearModuleList
AddModule mod_env.c
AddModule mod_log_config.c
AddModule mod_mime.c
AddModule mod_negotiation.c
AddModule mod_status.c
AddModule mod_include.c
AddModule mod_autoindex.c
AddModule mod_dir.c
AddModule mod_cgi.c
AddModule mod_asis.c
AddModule mod_imap.c
AddModule mod_actions.c
AddModule mod_userdir.c
AddModule mod_alias.c
```

```
AddModule mod_access.c
AddModule mod_auth.c
AddModule mod_so.c
AddModule mod_setenvif.c
AddModule mod_php4.c

#ExtendedStatus On

Port 80

User nobody
Group nobody

ServerAdmin alang@syszone.co.kr

ServerName 211.238.41.180

DocumentRoot "/usr/local/apache/htdocs"

<Directory />
    Options FollowSymLinks
    AllowOverride None
</Directory>

<Directory "/usr/local/apache/htdocs">

    Options Indexes FollowSymLinks MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>

<Directory "/home">
    Options FollowSymLinks ExecCGI
    AllowOverride All
    Order allow,deny
    Allow from all
    Deny from env=go_out
```

```
    Deny from export=go_out
</Directory>

<IfModule mod_userdir.c>
    UserDir www
</IfModule>

#<Directory /home/*/public_html>
#     AllowOverride FileInfo AuthConfig Limit
#     Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
#     <Limit GET POST OPTIONS PROPFIND>
#          Order allow,deny
#          Allow from all
#     </Limit>
#     <LimitExcept GET POST OPTIONS PROPFIND>
#          Order deny,allow
#          Deny from all
#     </LimitExcept>
#</Directory>

<IfModule mod_dir.c>
    DirectoryIndex index.html index.htm index.php index.phtml index.php3 index.jsp
</IfModule>

AccessFileName .htaccess

<Files ~ "^₩.ht">
    Order allow,deny
    Deny from all
    Satisfy All
</Files>

<Files ~ "^₩.ht">
    Order allow,deny
    Deny from all
</Files>
<Files ~ "*Chat*">
    Order allow,deny
```

```
    Deny from all
</Files>
<Files ~ "*CHAT*">
    Order allow,deny
    Deny from all
</Files>

#CacheNegotiatedDocs

UseCanonicalName On
<IfModule mod_mime.c>
    TypesConfig /usr/local/apache/conf/mime.types
</IfModule>

DefaultType text/plain

<IfModule mod_mime_magic.c>
    MIMEMagicFile /usr/local/apache/conf/magic
</IfModule>

HostnameLookups Off
ErrorLog /usr/local/apache/logs/error_log
LogLevel warn

LogFormat "%h %l %u %t ₩"%r₩" %>s %b ₩"%{Referer}i₩" ₩"%{User-Agent}i₩"" combined
LogFormat "%h %l %u %t ₩"%r₩" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

CustomLog /usr/local/apache/logs/access_log common

ServerSignature On

# > AddType           text/html .ahtml
# > EBCDICConvert Off=InOut .ahtml
#
# EBCDICConvertByType    On=InOut text/* message/* multipart/*
# EBCDICConvertByType    On=In       application/x-www-form-urlencoded
```

```
# EBCDICConvertByType   On=InOut application/postscript model/vrml
# EBCDICConvertByType Off=InOut */*

<IfModule mod_alias.c>

    Alias /icons/ "/usr/local/apache/icons/"

    <Directory "/usr/local/apache/icons">
        Options Indexes MultiViews
        AllowOverride None
        Order allow,deny
        Allow from all
    </Directory>

    Alias /manual/ "/usr/local/apache/htdocs/manual/"

    <Directory "/usr/local/apache/htdocs/manual">
        Options Indexes FollowSymlinks MultiViews
        AllowOverride None
        Order allow,deny
        Allow from all
    </Directory>

    ScriptAlias /cgi-bin/ "/usr/local/apache/cgi-bin/"

    <Directory "/usr/local/apache/cgi-bin">
        AllowOverride None
        Options None
        Order allow,deny
        Allow from all
    </Directory>

</IfModule>

<IfModule mod_autoindex.c>

    IndexOptions FancyIndexing
    AddIconByEncoding (CMP,/icons/compressed.gif) x-compress x-gzip
```

```
AddIconByType (TXT,/icons/text.gif) text/*
AddIconByType (IMG,/icons/image2.gif) image/*
AddIconByType (SND,/icons/sound2.gif) audio/*
AddIconByType (VID,/icons/movie.gif) video/*

AddIcon /icons/binary.gif .bin .exe
AddIcon /icons/binhex.gif .hqx
AddIcon /icons/tar.gif .tar
AddIcon /icons/world2.gif .wrl .wrl.gz .vrml .vrm .iv
AddIcon /icons/compressed.gif .Z .z .tgz .gz .zip
AddIcon /icons/a.gif .ps .ai .eps
AddIcon /icons/layout.gif .html .shtml .htm .pdf
AddIcon /icons/text.gif .txt
AddIcon /icons/c.gif .c
AddIcon /icons/p.gif .pl .py
AddIcon /icons/f.gif .for
AddIcon /icons/dvi.gif .dvi
AddIcon /icons/uuencoded.gif .uu
AddIcon /icons/script.gif .conf .sh .shar .csh .ksh .tcl
AddIcon /icons/tex.gif .tex
AddIcon /icons/bomb.gif core

AddIcon /icons/back.gif ..
AddIcon /icons/hand.right.gif README
AddIcon /icons/folder.gif ^^DIRECTORY^^
AddIcon /icons/blank.gif ^^BLANKICON^^

DefaultIcon /icons/unknown.gif

#AddDescription "GZIP compressed document" .gz
#AddDescription "tar archive" .tar
#AddDescription "GZIP compressed tar archive" .tgz

ReadmeName README
HeaderName HEADER

IndexIgnore .??* *~ *# HEADER* README* RCS CVS *,v *,t
```

```
</IfModule>

<IfModule mod_mime.c>

    AddLanguage da .dk
    AddLanguage nl .nl
    AddLanguage en .en
    AddLanguage et .ee
    AddLanguage fr .fr
    AddLanguage de .de
    AddLanguage el .el
    AddLanguage he .he
    AddCharset ISO-8859-8 .iso8859-8
    AddLanguage it .it
    AddLanguage ja .ja
    AddCharset ISO-2022-JP .jis
    AddLanguage kr .kr
    AddCharset ISO-2022-KR .iso-kr
    AddLanguage nn .nn
    AddLanguage no .no
    AddLanguage pl .po
    AddCharset ISO-8859-2 .iso-pl
    AddLanguage pt .pt
    AddLanguage pt-br .pt-br
    AddLanguage ltz .lu
    AddLanguage ca .ca
    AddLanguage es .es
    AddLanguage sv .sv
    AddLanguage cs .cz .cs
    AddLanguage ru .ru
    AddLanguage zh-TW .zh-tw
    AddCharset Big5          .Big5    .big5
    AddCharset WINDOWS-1251 .cp-1251
    AddCharset CP866         .cp866
    AddCharset ISO-8859-5    .iso-ru
    AddCharset KOI8-R        .koi8-r
    AddCharset UCS-2         .ucs2
```

```
    AddCharset UCS-4            .ucs4
    AddCharset UTF-8            .utf8

    <IfModule mod_negotiation.c>
        LanguagePriority kr en da nl et fr de el it ja no pl pt pt-br ru ltz ca es sv tw
    </IfModule>

    AddType application/x-tar .tgz

    AddEncoding x-compress .Z
    AddEncoding x-gzip .gz .tgz
    #AddType application/x-compress .Z
    #AddType application/x-gzip .gz .tgz
    AddType application/x-httpd-php .html .php3 .php4 .inc .phtml .php .ph .jsp .xtm
    AddType application/x-httpd-php-source .phps


    AddHandler cgi-script .cgi

    #AddType text/html .shtml
    #AddHandler server-parsed .shtml
    #AddHandler send-as-is asis
    #AddHandler imap-file map
    #AddHandler type-map var

</IfModule>
#MetaDir .web

#MetaSuffix .meta

# Customizable error response (Apache style)
#   these come in three flavors
#
#      1) plain text
#ErrorDocument 500 "The server made a boo boo.
#   n.b.   the single leading (") marks it as text, it does not get output
#
#      2) local redirects
```

```
#ErrorDocument 404 /missing.html
#   to redirect to local URL /missing.html
#ErrorDocument 404 /cgi-bin/missing_handler.pl
#   N.B.: You can redirect to a script or a document using server-side-includes.
#
#       3) external redirects
#ErrorDocument 402 http://www.example.com/subscription_info.html
#   N.B.: Many of the environment variables associated with the original
#   request will *not* be available to such a script.

#
# Customize behaviour based on the browser
#
<IfModule mod_setenvif.c>

      BrowserMatch "Mozilla/2" nokeepalive
      BrowserMatch "MSIE 4₩.0b2;" nokeepalive downgrade-1.0 force-response-1.0
      BrowserMatch "RealPlayer 4₩.0" force-response-1.0
      BrowserMatch "Java/1₩.0" force-response-1.0
      BrowserMatch "JDK/1₩.0" force-response-1.0
      BrowserMatch "Webzip" go_out
      BrowserMatch "WebZip" go_out
      BrowserMatch "Teleport" go_out
      BrowserMatch "GetRight" go_out
      BrowserMatch "Wget" go_out
</IfModule>

#<Location /server-status>
#       SetHandler server-status
#       Order deny,allow
#       Deny from all
#       Allow from .example.com
#</Location>

# Allow remote server configuration reports, with the URL of
# http://servername/server-info (requires that mod_info.c be loaded).
# Change the ".example.com" to match your domain to enable.
#
```

```
#<Location /server-info>
#       SetHandler server-info
#       Order deny,allow
#       Deny from all
#       Allow from .example.com
#</Location>

# MOD_JK Config

<IfModule mod_jk.c>
JkWorkersFile /usr/local/tomcat/conf/workers.properies
JkLogFile /usr/local/tomcat/logs/jk.log
JkLogLevel info
</IfModule>


NameVirtualHost 211.238.41.180

<VirtualHost 211.238.41.180>
    ServerAdmin alang@syszone.co.kr
    DocumentRoot /home/syszone/www
    ServerName syszone.co.kr
    ServerAlias syszone.co.kr www.syszone.co.kr
    JkMount /*.jsp ajp13
    JkMount /webapps/* ajp13
    JkMount /ROOT/* ajp13
</VirtualHost>
```

5.10 Tomcat 설정

vi /usr/local/tomcat/conf/workers.properies

```
workers.tomcat_home=/usr/local/tomcat
workers.java_home=/usr/java/java
worker.list=ajp12, ajp13
worker.ajp13.port=8009
worker.ajp13.host=localhost
worker.ajp13.type=ajp13
```

5.11 Tomcat 과 Apache 연동 및 가상 호스트 설정

vi /usr/local/tomcat/conf/server.xml

```
.
.
            </Host>
        <Host name="syszone.co.kr">
            <Context path="" docBase="/home/www/www" reloadable="true"/>
            <Alias>www.syszone.co.kr</Alias>
        </Host>

        </Engine>
    </Service>
</Server>
```

/usr/local/apache/bin/apachectl restart
/usr/local/tomcat/bin/strtup.sh
// 확인 //

vi /home/syszone/www/phpinfo.php

```
<?

            phpinfo ();

?>
```

vi /home/syszone/www/hello.jsp

```
<%

            out.println("Hello, JSP");

%>
```

http://www.syszone.co.kr/phpinfo.php
http://www.syszone.co.kr/hello.jsp
http://211.238.41.180:8080

Introduction to

Load Balance Clustering

Using Linux

(부하분산 클러스터 구성)

클루닉스 기술부 3차 기술 세미나 자료

클루닉스

서 진우(alang@clunix.com)

목 차

1. 시스템 구축 준비 하기

1.1. 리눅스 부한 분산 클러스터 개론

1.2. 시스템 구축 환경

2. Open Source 를 이용한 리눅스 부하분산 클러스터 구축

2.1. 리눅스 클러스터 커널 환경 구축

2.2. ipvsadm 설치 및 사용 방법

2.3. LB 서버와 Real 서버의 네크워크 구성

2.4. Direct Routing 방식에서의 ARP 문제 해결

2.5. 부하 분산 테스트

2.6. Mon 을 이용한 Load Balance 에 Fail Over 기능 추가하기

2.7. LB 서버간의 이중화 구축 하기 (Heartbeat)

3. Encluster 를 이용한 리눅스 부하분산 클러스터 구축

3.1. Encluster 1.6을 이용한 리눅스 부하분산 클러스터 구축

3.2. Encluster 2.0을 이용한 리눅스 부하분산 클러스터 구축

4. Data 동기화 (Data Sync or Data Mirroring)

4.1. rsync 를 이용한 data sync

4.2. dutils 를 이용한 data sync

4.3. NFS + Automount 를 이용한 data sync

4.4. Intermezzo 를 이용한 data sync

4.5. Pvfs 를 이용한 data sync

5. Benchmark Tool로 시스템 성능 체크 하기

5.1. 웹 서버 벤치 마크 개론

5.2. 웹 서버 벤치마크 툲과 사용 방법

5.3. 벤치마크 시나리오 수립 및 분석 하기

5.4. 웹 시스템 튜닝 하기

6. 리눅스 클러스터를 이용한 대용량 웹 클러스터 시스템 구성 하기 (시스템 컨설팅)

6.1. 인터넷 웹 클러스터 개론

6.2. 대규모 인터넷 사이트 구축 설계

1. 시스템 구축 준비 하기

1.1. 리눅스 부하 분산 클러스터 개론

1.1.1 소 개

인터넷의 발달과 더불어 생활의 정보 전달 매체로 웹이란 수단을 선택하기 시작한지 10 년에 다되어가면서 매년 인터넷의 사용 증가는 100%를 넘고 있다. 뿐만 아니라 근래에 들어 인터넷 초기 시절의 단순한 정보 공유 매개체로서의 역할에서 사업상의 중요한 업무 시스템 구성의 중요한 요소까지 차지하게 되었다. 인터넷의 발달과 더불어 하드웨어와 소프트웨어 그리고 네트워크의 발달 속도 역시 빠른 추세로 증가 하고 있지만 기하급수적으로 늘어난 서비스 사용량을 처리하기에 보편적인 시스템의 성능과 시스템 환경의 부족함이 많이 존재하고 있다.

이에 보다 높은 가용성 (Availability) 과 확장성 (scalable)을 가진 서버의 요구가 절실히 높아지고 있다. 이에 고성능 SMP 서버를 이용하는 경우도 있지만 비 경제적인 가격 문제와 실제 시스템 병목 원인이 꼭 서버의 성능에만 있는 것이 아니기 때문에 고성능 SMP 서버만이 진정한 해결 방법은 아니다.

빠른 네트워크의 발달과 함께 분산된 서버의 자원을 하나의 작업에 이용하는 클러스링 기술이 발달하게 되고 인터넷 서비스 역시 클러스터링 기술이 부각되기 시작하였다.

클러스터란 여러 대의 컴퓨터를 네트워크를 통해 연결하여 하나의 단일 컴퓨터처럼 작동하게 하는 기술을 말한다. 클러스터는 사용 목적에 따라 구현 방법 및 소프트웨어가 달라지며, 클러스터의 기능과 운영이 달라진다. 따라서 클러스터는 그 사용 목적에 따라 베어울프와 같은 과학 계산용 클러스터와 인터넷 서비스에서 사용되는 부하분산 클러스터로 나눌 수 있다.

이런 클러스터 시스템의 장점으로 추가 확장성과 높은 가용성 그리고 비용-효율성을 두루 갖추고 있어 인터넷 서비스에 사용되는 시스템의 요구 조건을 충족 시켜 줄 수 있다.

부하 분산 클러스터는 이전부터 L4 Switch 란 하드웨어를 이용하여 시스템 구축을 해 왔지만 워낙 고가의 시스템이여서 일반 사이트에서는 적용이 힘든 방식으로 알려져 있었다. 하지만 Linux 의 커널에서 L4 Switch 의 기능과 동일하고 더 월등한 성능의 LVS(Linux virtual server) 기능을 지원함에

따라 일반 사이트에서도 손쉽게 클러스터 기술을 이용하여 고성능의 서버를 이용할 수 있게 되었다. 리눅스의 LVS 커널은 커널 2.0.x 때부터 지원 되었고, 사용 된지도 7~8 년에 이르고 있다.

이에 지속적인 안정성과 성능이 개선되어져 리눅스의 수많은 기능 중 가장 선호하는 기능으로 손꼽고 있다.

이런 리눅스 부하 분산 클러스터도 여러가지 방식이 있고 리눅스 LVS 커널과 다른 리눅스 응용 프로그램을 조합하여 고성능 뿐만 아니라 고가용성의 기능까지 탑재하게 된다.

본문 에서는 네트워크 구성에 따른 부하 분산 방식과 작업 활당 방식 그리고 고 가용성을 구현하는 방법에 대해 자세히 알아보도록 하겠다.

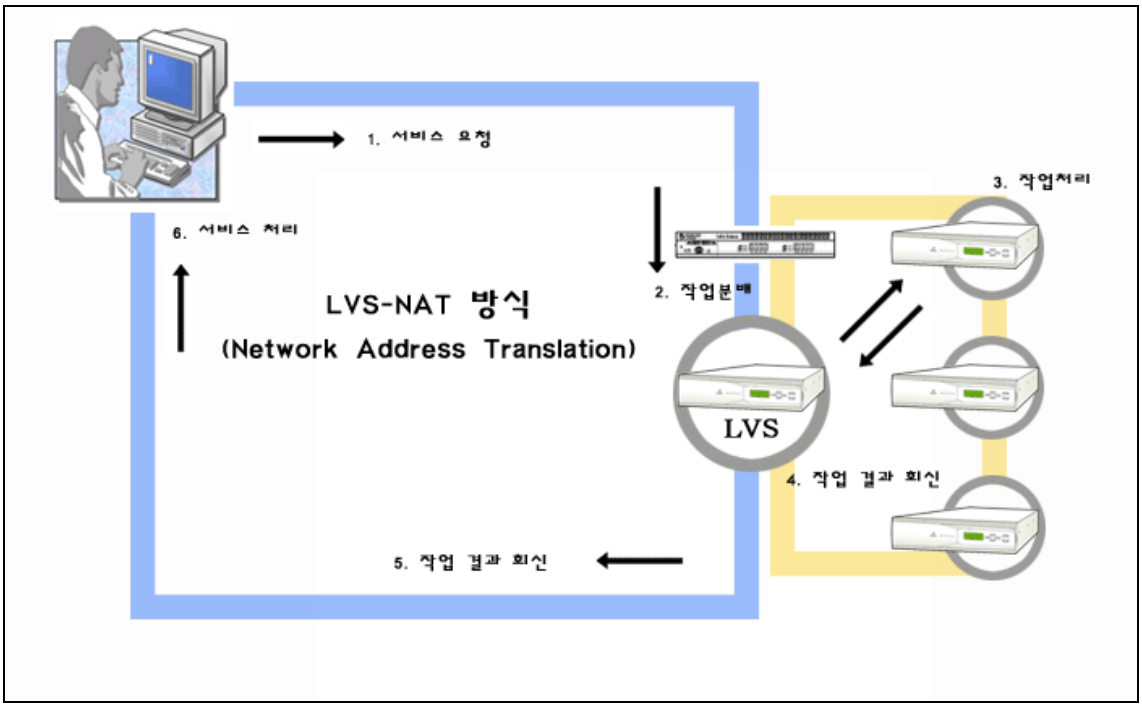
1.1.2 네트워크 구성에 따른 부하 분산 클러스터 종류

부하 분산 클러스터를 생각하면 대부분 Load Balancer 을 앞 단에 두고 backend 의 Real Server 에 작업을 분산하는 서버 측면의 부하 분산 방식을 알고 있다. 하지만 클라이언트에 부하분산 모듈을 넣어 서버의 상황을 클라이언트가 서버로부터 얻어내어 클라이언트가 직접 서버로 분산해서 들어가는 클라이언트 측면의 부하분산 방식도 있다. (예: Clunix 의 LB Less , 버클리의 스마트 클라이언트, 보스턴 대학의 클라이언트) 서버 측면의 부하분산 방식으로는 DNS 의 호스트를 이용한 Round-Robin DNS 방법과 Reverse-Proxy 와 같은 응용프로그램 계층의 부하분산 그리고 L4 Switch 와 같은 IP 계층의 부하분산 방식이 있다. 리눅스 가상 서버 역시 IP 계층의 부하 분산 방식에 속한다.

리눅스 가상 서버에서도 네트워크 구성에 따라 주소 변환 방식 (NAT)과 IP tunneling 에 의한 방식 그리고 Direct Rounting dispatching 기술을 이용한 방식 등이 있다.

1.1.2.1 Network address translation 부하 분산 방식 (네트워크 주소 변환 방식)

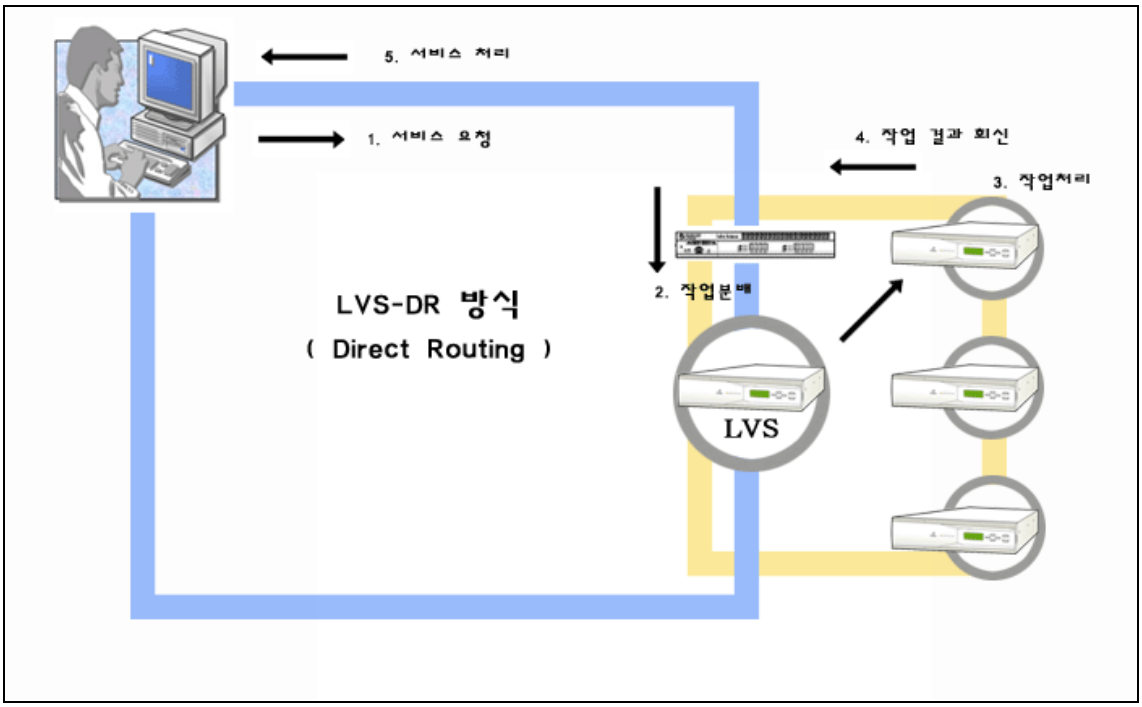
IPv4 에서 IP 주소가 부족하고 보안상에 몇가지 문제가 있어서 인터넷에서 사용할 수 없는 사설 IP (10.0.0.0/255.0.0.0 , 172.16.0.0/255.240.0.0 , 192.168.0.0 /255.255.0.0)를 사용하는 경우가 많아 지고 있다. 이때 인터넷 망에서 사설망의 호스트에 접근하기 위해서 네트워크 주소 변한 (NAT) 기능이 필요하게 된다. 이런 리눅스에서의 NAT 방식의 부하분산 방식은 초기의 리눅스 IP 마스커레이딩 코드와 Steven Clarke 의 포트 포워딩 코드를 재사용 하여 구현하고 있다.



사용자가 LVS 에서 가지고 있는 VIP 를 향해 서비스 요청을 하면 LVS 에서는 그 서비스 요청된 IP 와 Port 가 가상 서버 정책에 있는지를 확인하고 있다면 이 정책에 정해진 스케줄에 따라 실제 작업 서버로 패킷을 포워딩 시키게 된다. 이때 LVS 서버는 사용자가 요청한 패킷의 네트워크 주소 부분을 공인 IP 에서 사설 IP 로 주소를 변환 시켜 같은 사설망에 존재하는 작업 서버로 보내게 된다. 작업서버는 LVS 서버로부터 패킷을 전달 받고 이에 대한 요청 처리된 내용을 다시 LVS 에 보내게 된다. 그럼 LVS 가 사설 IP 로 된 패킷을 다시 받아 다시 공인 IP 로 주소를 변환시켜서 사용자에게 돌려 보내는 방식이다.

1.1.2.2 Direct Routing 부하 분산 방식

실제 서버와 부하 분산 서버 사이에서 가상 IP 를 공유하는 방식으로 부하 분산 서버와 실제 작업 서버에 모두 동일한 가상 IP 를 가지도록 네트워크 구성해야 한다. 가상 IP 가 설정된 인터페이스를 통해 사용자의 요청을 받아 들여 이 요청이 가상 서버 정책에 적용되는 요청인지를 확인 후 적용되는 패킷이면 정책에 따라 실제 작업 서버로 패킷을 포워딩하게 된다. 이때 실제 서버 역시 별도의 Arp 캐싱을 남기지 않는 Alias 인터페이스를 통해 부하분산서버로부터 포워딩되는 패킷을 받아 들이게 된다.
만일 가상 IP 가 설정된 인터페이스에 arp caching 이 남는다고 하면 사용자가 부하 분산 서버를 통해 초기에 연결된 실제 작업와 연결되면 그 이후부터는 초기 접속한 작업 서버와만 통신이 이루어 지게 된다. 그렇기 때문에 반드시 커널 차원에서의 arp hidden patch 를 시켜 줘야 한다.



DR 방식은 패킷의 데이터 프레임 부분의 MAC 주소만을 변경하여 패킷을 포워딩 하기 때문에 물리적인 세크먼트 (같은 subnet)안에서만 동작이 가능하다. DR 방식의 arp 문제 해결 방법에 대해서는 실제 DR 구축 문서를 참조하길 바란다.

1.1.2.3 IP Tunneling 에 의한 부하 분산 방식

IP 터너링은 IP 화된 정보 (IP datagram)안에 IP 화된 정보를 감싸 넣는 (encapsulate) 기술이다. 리눅스 커널에서 지원하고 이를 이용하여 보다 WAN 을 대상으로 작업 서버 노드를 구성할수 있게 된다.

실제 가상 IP 주소로 향하는 요구 패키이 부하분산 서버로 전달 되면 부하분산 서버에서 패킷의 목적지 주소와 포트를 검사하여 가상 서버 정책과 일치 하면 스케줄링에 따라 실제 작업 서버로 전달하게 된다. 이때 패킷을 일반적인 스트링 방식으로 전달하는 것이 아닌 캣슐 형식으로 싸서 전달하게 된다. 이리 전달 되면 작업 서버에서는 감싸진 패킷을 다시 풀고 요청을 처리한 다음 실제 서버의 라우팅 테이블에 따라 사용자에게 직접 결과를 돌려주는 방식이다.

IP 터너링 방식은 DR 방식의 물리적인 세그먼트의 영향을 받지 않고 어디든지 작업을 분산 할 수 있는 장점이 있다. 그래서 거의 무한한 확장성을 가지게 된다. 하지만 모든 서버가 IP 터널링 전송 규약을 지원해야 하기에 지원하지 못하는 OS 에서는 문제가 발생하게 된다. .

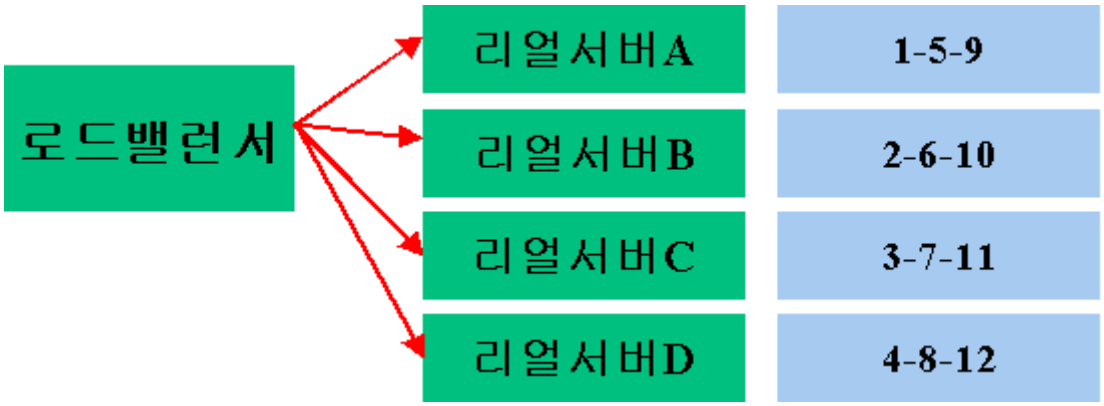
1.1.3 작업 활당 방식 (Scheduling Algorithms)

클러스터로부터 서버를 선택하기 위한 네 개의 작업 할당 방식들(scheduling algorithms)이 준비되어있다: Round-Robin, 가중치가 있는(weighted) Round-Robin, 최소-접속(Least-Connection) 그리고 가중치가 있는 최소-접속의 방식들이다. 처음의 두 방식들은 서버에 대한 어떠한 부하 정보(load information)도 가지고

있지 않기 때문에 자명(self-explanatory)하다. 뒤의 두 방식들은 각 서버에 대한 활동 접속 수(active connection number)를 세어 이러한 접속 수에 의해 그들의 부하를 추정한다.

1.1.3.1 Round-Robin Scheduling (라운드 로빈 스케쥴링)

말그대로 라운드-로빈 방식을 이용해 네트웍 연결을 서로 다른 서버에 연결하는 것을 말한 다. 이경우 실제서버의 연결갯수나 반응시간등은 고려를 하지 않는다. 그렇지만 약간의 차이 가 있다. 라운드 로빈 DNS 는 단일한 도메인을 서로 다른 IP 로 해석을 하지만, 스케쥴링의 기초는 호스트 기반이며 캐싱때문에 알고리즘을 효율적으로 사용하기 힘들다. 그래서 실제 서버사이에 동적인 부하 불균형이 심각해 질수 있다. 가상 서버의 스케쥴링 기초는 네트웍 기반이며 라운드 로빈 DNS 에 비해 훨씬 더 훌륭하다.



1.1.3.2 Weighted Round-Robin Scheduling (가중치기반 라운드 로빈 스케쥴링)

가중치기반 라운드 로빈 스케쥴링은 실제 서버에 서로 다른 처리 용량을 지정할 수 있다. 각 서버에 가중치를 부여할 수 있으며, 여기서 지정한 정수값을 통해 처리 용량을 정한다. 기본 가중치는 1 이다. 예를 들어 실제 서버가 A,B,C 이고 각각의 가중치가 4,3,2 일 경우 스케쥴링 순서는 ABCABCABA 가 된다. 가중치가 있는 라운드 로빈 스케쥴링을 사용하면 실제 서버에서 네트웍 접속을 셀 필요가 없고 동적 스케쥴링 알고리즘보다 스케쥴링의 과부하가 적으므로 더 많은 실제 서버를 운영 할 수 있다. 그러나 요청에 대한 부하가 매우 많을 경우 실제 서버 사이에 동적인 부하 불균 형 상태가 생길 수 있다. 라운드 로빈 스케쥴링은 가중치기반 라운드 로빈 스케쥴링의 특별한 한 종류이며 모든 가중 치가 동일한 경우이다. 가상 서버의 규칙을 변경하고나서 스케쥴링 순서를 생성 하는 데는 거의 과부하가 걸리지 않으며 실제 스케쥴링에 어떠한 과부하도 추가하지 않는다. 그러므로 라운드 로빈 스케쥴링만 단독으로 실행하는 것은 불필요한 일이다.

1.1.3.3 Least-Connection Scheduling (최소 접속 스케쥴링)

최소 접속 스케쥴링은 가장 접속이 적은 서버로 요청을 직접 연결하는 방식을 말한다. 각 서버에서 동적으로 실제 접속한 숫자를 세어야하므로 동적인 스케쥴링 알고리즘중의 하나이다. 비슷한 성능의 서버로 구성된 가상 서버는 아주 큰 요구가 한 서버로만 집중되지 않기 때문에, 접속부하가 매우 큰 경우에도 아주 효과적으로 분산을 한다.
가장 빠른 서버에서 더 많은 네트웍 접속을 처리할 수 있다. 그러므로 다양한 처리 용량을 지닌 서버로 구성했을 경우에도 훌륭하게 작동 한다는 것을 한눈에 알 수 있을 것이다. 그렇지만 실제로는 TCP 의 TIME_WAIT 상태때문에 아주 좋은 성능을 낼수는 없다. TCP 의 TIME_WAIT 는 보통 2 분이다. 그런데 접속자가 아주 많은 웹 사이트는 2 분동안에 몇천개의 접속을 처리해야 할 경우가 있다. 서버 A 는 서버 B 보다 처리용량이 두배일 경우 서버 A 는 수천개의 요청을 처리하고 TCP 의 TIME_WAIT 상황에 직면하게 된다. 그렇지만 서버 B 는 몇천개의 요청이 처리되기만을 기다리게 된다. 그래서 최소 접속 스케쥴링을 이용할 경우 다양한 처리용량을 지난 서버로 구성되었을 경우 부하분산이 효율적으로 되지 못할 수 있는 것이다.

1.1.3.4 Weighted Least-Connection Scheduling (가중치 기반 최소 접속 스케쥴링)

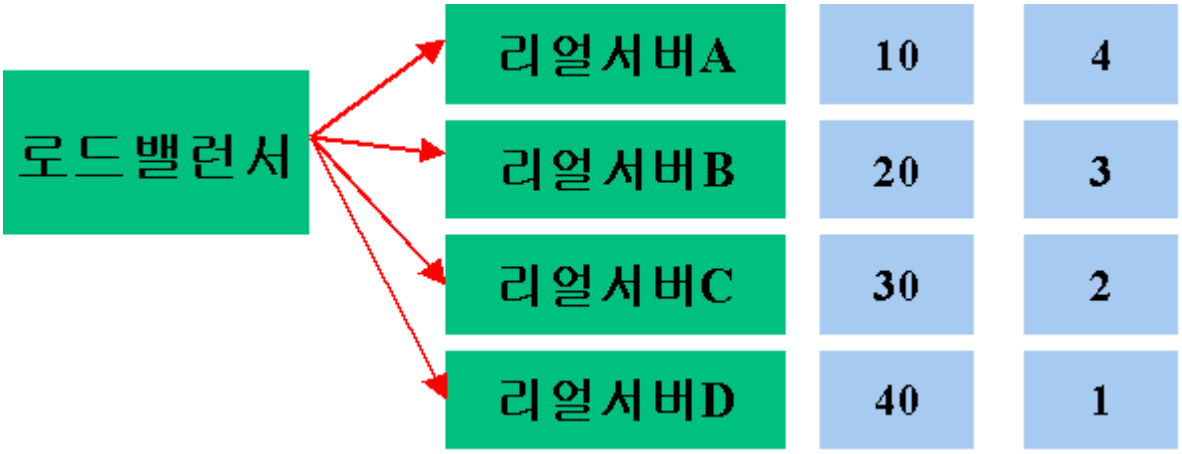
가중치 기반 최소 접속 스케쥴링은 최소 접속 스케쥴링의 한 부분으로서 각각의 실제 서버에 성능 가중치를 부여할 수 있다. 언제라도 가중치가 높은 서버에서 더 많은 요청을 받을 수 있다. 가상 서버의 관리자는 각각의 실제 서버에 가중치를 부여할 수 있다. 가중치의 비율인 실제 접속자수에 따라 네트웍 접속이 할당된다. 기본 가중치는 1 이다. 가중치가 있는 최소 접속 스케쥴링은 다음과 같이 작동한다:
n 개의 실제 서버가 있는 경우 각 서버 i 는 가중치 Wi (i = 1, ... , n)를 가진다고 가정하자. 서버 i 의 활동 접속(active connection)은 Ci (i = 1, ... , n)이고 모든_접속 은 Ci (i = 1, ... , n)의 합이다. 서버 j 로 가는 넷트웍 접속은 아래와 같다.

(Cj/ALL_CONNECTIONS)/Wj = min { (Ci/ALL_CONNECTIONS)/Wi } (i=1,..,n)

이 비교에서 ALL_CONNECTIONS 는 상수이므로 Ci 를 모든_접속 으로 나눠줄 필요가 없다.그러면 다음과 같이 최적화될 것이다.

Cj/Wj = min { Ci/Wi } (i=1,..,n)

가중치가 있는 최소 접속 스케쥴링 알고리즘은 최소 접속 스케쥴링 알고리즘에 비해 부가적인 배분작업이 필요하다. 서버들이 같은 처리 용량을 가졌을때는 작업 할당의 간접 비용을 최소화하기위해 최소 접속 스케쥴링과 가중치가 있는 최소 접속 스케쥴링 알고리즘 둘 다 사용할 수 있다.



1.1.4 고 가용성 (High Availability)

중요한 상업적 응용 분야가 인터넷으로 점점 더 많이 옮아옴에 따라 가용성이 높은 서버들을 제공하는 것이 점점 더 중요해지고 있다. 클러스터화된 시스템의 장점중 하나는 하드웨어와 소프트웨어의 여유분이 있다는 것이다. 높은 가용성은 노드(node)나 데몬(deamon)의 장애들(failures)이나 시스템의 재구성(reconfiguring)을 적절히 감지해 클러스터에 남아있는 노드로 작업 부하(workload)를 이전하는 것으로 이룰 수 있다. 가상 서버의 높은 가용성을 현재 "mon"과 "fake" 소프트웨어를 사용해 제공한다. "mon"은 넷트웍 서비스 가용성과 서버 노드를 모니터할 수 있는 범용 자원 모니터링 시스템(general-purpose resource monitoring system)이다. "fake"는 ARP 속이기(spoofing)를 사용해 IP 를 이전하는 소프트웨어다.

서버의 장애 극복(failover)은 다음과 같이 제어한다: 부하 분산기(load balancer)에 클러스터의 서비스 데몬들과 서버 노드들을 모니터하기 위한 "mon" 데몬이 운영된다. 매 t 초마다 서버 노드들이 살아있는지 감지하도록 fping.monitor 를 구성하고 매 m 분마다 모든 노드의 서비스 데몬들을 감지하기 위해 연관된 모니터 프로그램 역시 구성한다. 예를 들어 http 서비스를 점검하기위해 http.monitor 를, ftp 서비스를 위해 ftp.monitor 를 또, 기타 다른 모니터 프로그램을 사용할 수 있다. 서버 노드나 데몬이 새로 사용 불능 상태(down)가 되거나 사용 가능 상태(up)가 되면 가상 서버 테이블에 규칙을 제거하거나

추가하기위한 경보가 작성된다. 따라서 부하 분산기가 서비스 데몬이나 서버를 제거하거나 복구되었을 때 서비스에 추가하는 것을 자동으로 할 수 있다

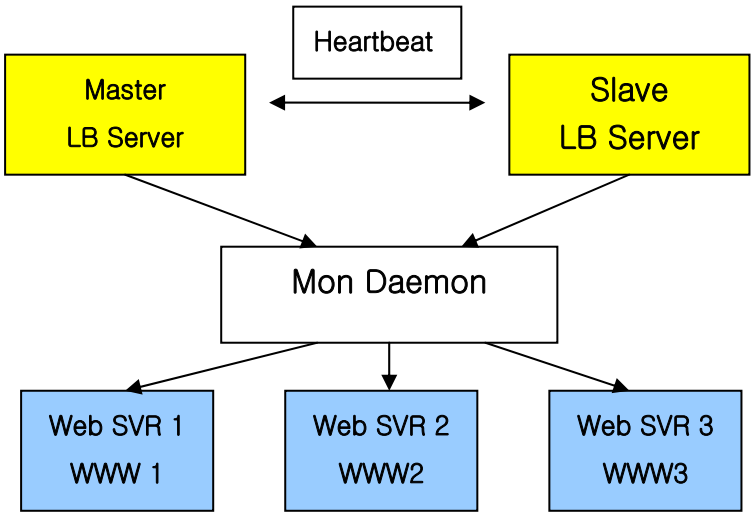
1.2 시스템 구축 환경

1.2. 시스템 구축 환경

앞으로 진행할 리눅스 클러스터 시스템 구축을 위해 테스트 환경으로 4 대 (최소 3 대) 서버가 필요합니다. Load Balance Server 2 대와 일반 웹서버 2 대로 구성하여 리눅스 부하분산 클러스터 시스템을 구축할 것 입니다.

시스템 구성은 Master LB 서버와 Slave LB 서버를 두어 Heartbeat 를 이용하여 LB 서버의 이중화 (High availabiltiy)를 구현할 것이고 Mon 을 이용하여 2 대의 웹서버에 작업 부하 분산 및 Service Fail Over 가 될수 있도록 구현할 것이다.

1.2.1 시스템 구성도



1.2.2 시스템 기본 정보

Operating System : Redhat Linux 9 (Kernel Version : 2.4.27)

Master LB ip : 211.238.41.167

Slave LB ip : 211.238.41.166

Web Server-1 : 211.238.41.165

Web Server-2 : 211.238.41.164

Virtual ip : 211.238.41.170

Cluster ip (Encluster 시 필요) : 211.238.41.168

LB 서버간의 heartbeat 와 웹서버간의 Data Sync, LB 와 웹서버간의 Mon server monitoring 에 사용되는 네트워크 대역을 Private Network 로 분리할 수도 있다.
이 방법에 대해서는 아래 내용을 충분히 숙지하고, 기본적인 리눅스 네트워킹에 대한 이해만 있으면 충분히 구축할 수 있을 것이다.

1.2.3 설치 시 필요 패키지

Kernel : 2.4.26 이상 (본 문서에서는 2.4.27 사용)
ftp://ftp.kernel.org/pub/linux

ipvs : Kernel 2.4.23 이하 버전에만 해당함
http://linuxvirtualserver.org/software

ipvsadm-1.21
http://linuxvirtualserver.org/software

arp hidden patch : 해당 커널에 맞는 패치를 다운 받는다.
http://www.ssi.bg/~ja

Mon 관련 Perl 모듈
ftp://ftp.bora.net/pub/CPAN
ftp://ftp.kornet.net/pub/CPAN
ftp://ftp.nuri.net/pub/CPAN
CPAN 을 이용한 다운 방식도 있음 -> 본문 참조

Mon
ftp://ftp.kernel.org/pub/software/admin/mon

Mon.cgi
http://www.nam-shub.com/files/

fping
ftp://ftp.kernel.org/pub/software/admin/mon

heartbeat
http://linux-ha.org/download

rsync

http://rsync.samba.org/ftp/rsync

Total package

ftp://syszone.co.kr/pub/linux/server/lvspkg

2. Open Source 를 이용한 리눅스 부하분산 클러스터 구축

2.1. 리눅스 클러스터 커널 환경 구축

이전에는 리눅스 커널에서 기본적으로 ipvs 기능을 지원 하지 않았기에 별도의 커널 패치를 해주어야 한다. ipvs 를 정상적으로 수행하기 위해서는 ipvs 자체 커널 패치와 direct routing 방식을 지원하기 위해서 arp 문제를 해결해 주는 non-arp alias 인터페이스를 만들 때 사용되는 arp hidden 패치를 해주어야 한다.

패치는 각 커널 버전에 맞는 패치를 해야 커널 컴파일 시 에러가 발생하지 않는다.

ipvs 는 커널 2.4.23 까지 ipvs-1.0.10 대 Patch 로 사용이 가능하다.
2.4.26 이후 부터는 ipvs-1.0.11 버전이 커널 속에 기본적으로 포함되어 있다.

2.1.1 커널 2.4.23 이하 버전일 경우 커널 패치 방법

ipvs 패치는 로드밸랜싱 서버에만 해주면 되지만 arp 패치는 로드,리얼 서버 모두 해준다. 여기서는 시스템 초기 셋팅은 동일하게 해야 한다. 두 대가 모두 lvs, real_server 역활을 해야 하기 때문이다.

IPVS module 의 설치 방법에는 두가지가 있다. 우선 하나는 모듈로 올리는 방법과 커널에 추가 하는 방법이다.

- 모듈에 올리는 방법 :

// <path-name> 은 ipvs patch directory

```
# tar xzvf ipvs-0.9.6.tar.gz
# cd ipvs-0.9.6/ipvs
# make
# make -C ipvsadm
```

```
# make install
# insmod ip_vs_wlc.o
```

- 커널에 적재하는 방법 :

```
# cd /usr/src
# tar xjvf linux-2.4.23.tar.bz2
# ln -sf linux-2.4.23 linux
# cd linux
# make menuconfig
->          save -> exit

# zcat ../linux-2.4.23-ipvs.patch.gz | patch p1
# cat ../hidden-2.4.23-1.diff | patch p1
```

만일 커널 버전에 맞지 않는 Patch 를 할경우 에러가 발생할 수 있으니 반드시 커널 버전에 맞는 patch 를 적용하도록 한다.

2.1.2 커널 컴파일 하기

아래 커널 컴파일 환경은 Kernel 2.4.27 환경이고, 커널 컴파일 관련 옵션 선택 작업은 2.1.1 에서 다룬 kernel 2.4.23 이전 커널의 ipvs 패치 이후 진행 단계와 동일하게 작업 하면 된다.

이후 모든 설명은 커널 2.4.26 이상 환경을 기준을 설명한다.

```
# tar xjvf linux-2.4.27.tar.bz2
# ln -sf linux-2.4.27 linux
# cd linux
# make menuconfig
```

아래와 같은 초기 메뉴 화면이 나온다.

```
        Code maturity level options  --->
        Loadable module support  --->
        Processor type and features  --->
        General setup  --->
        Memory Technology Devices (MTD)  --->
        Parallel port support  --->
        Plug and Play configuration  --->
        Block devices  --->
        Multi-device support (RAID and LVM)  --->
        Networking options  --->
        Telephony Support  --->
```

```
                ATA/IDE/MFM/RLL support   --->
```

기본적은 커널 옵션은 해당 시스템 사항에 맞게 적용하고 여기서는 ipvs 에 관련된 부분만 적용하도록 하겠다.

먼저 Networking options 메뉴로 가서 아래와 같이 옵션을 체크 한다.

```
            <*> Packet socket
            [*]    Packet socket: mmapped IO
            < > Netlink device emulation
            [*] Network packet filtering (replaces ipchains)
            [ ]    Network packet filtering debugging
            [*] Socket Filtering
            <*> Unix domain sockets
            [*] TCP/IP networking
            [*]    IP: multicasting
            [ ]    IP: advanced router
            [ ]    IP: kernel level autoconfiguration
            <M>    IP: tunneling
            <M>    IP: GRE tunnels over IP
            [*]       IP: broadcast GRE over IP
            [ ]    IP: multicast routing
            [*]    IP: ARP daemon support (EXPERIMENTAL)
            [ ]    IP: TCP Explicit Congestion Notification support
            [*]    IP: TCP syncookie support (disabled per default)
              IP: Netfilter Configuration   --->
              IP: Virtual Server Configuration   --->
            < >    The IPv6 protocol (EXPERIMENTAL)
            < >    Kernel httpd acceleration (EXPERIMENTAL)
               SCTP Configuration (EXPERIMENTAL)   --->
            < > Asynchronous Transfer Mode (ATM) (EXPERIMENTAL)
            < > 802.1Q VLAN Support
            ---
            < > The IPX protocol
            < > Appletalk protocol support
            Appletalk devices   --->
            < > DECnet Support
            < > 802.1d Ethernet Bridging
            < > CCITT X.25 Packet Layer (EXPERIMENTAL)
```

```
< > LAPB Data Link Driver (EXPERIMENTAL)
[ ] 802.2 LLC (EXPERIMENTAL)
[ ] Frame Diverter (EXPERIMENTAL)
< > Acorn Econet/AUN protocols (EXPERIMENTAL)
< > WAN router
[ ] Fast switching (read help!)
[ ] Forwarding between high speed interfaces
QoS and/or fair queueing  --->
Network testing  --->
```

IP: Netfilter Configuration ---> 로 가서 iptables 에 관련된 옵션을 아래와 같이 체크 한다. 실제 iptables 의 경우 패킷의 포워딩 및 NAT, DR 방식에서 주요 한 요소로 사용되게 된다.

```
<*> Connection tracking (required for masq/NAT)
<M>  FTP protocol support
<M>  Amanda protocol support
<M>  TFTP protocol support
<M>  IRC protocol support
<M> Userspace queueing via NETLINK (EXPERIMENTAL)
<*> IP tables support (required for filtering/masq/NAT)
<M>  limit match support
<M>  MAC address match support
<M>  Packet type match support
<M>  netfilter MARK match support
<M>  Multiple port match support
<M>  TOS match support
<M>  recent match support
<M>  ECN match support
<M>  DSCP match support
<M>  AH/ESP match support
<M>  LENGTH match support
<M>  TTL match support
<M>  tcpmss match support
<M>  Helper match support
<M>  Connection state match support
<M>  Connection tracking match support
<M>  Unclean match support (EXPERIMENTAL)
<M>  Owner match support (EXPERIMENTAL)
<*>  Packet filtering
```

```
        <M>     REJECT target support
        <M>     MIRROR target support (EXPERIMENTAL)
        <*>   Full NAT
        <M>     MASQUERADE target support
        <M>     REDIRECT target support
        [*]     NAT of local connections (READ HELP)
        <M>     Basic SNMP-ALG support (EXPERIMENTAL)
        <*>   Packet mangling
        <M>     TOS target support
        <M>     ECN target support
        <M>     DSCP target support
        <M>     MARK target support
        <M>   LOG target support
        <M>   ULOG target support
        <M>   TCPMSS target support
        <*> ARP tables support
        <M>   ARP packet filtering
```

```
      IP: Virtual Server Configuration  ---> 로 가서 ipvs 옵션을 체크한다.
    <*> virtual server support (EXPERIMENTAL)
    [*]   IP virtual server debugging
    (12)   IPVS connection table size (the Nth power of 2)
    --- IPVS scheduler
    <*>   round-robin scheduling
    <*>   weighted round-robin scheduling
    <*>   least-connection scheduling
    <*>   weighted least-connection scheduling
    <*>   locality-based least-connection scheduling
    <*>   locality-based least-connection with replication scheduling
    <*>   destination hashing scheduling
    <*>   source hashing scheduling
    <*>   shortest expected delay scheduling
    <*>   never queue scheduling
    --- IPVS application helper
    <*>   FTP protocol helper
```

이밖에 Reiserfs, NFS, Coda, XFS, Samba 등등 클러스터 파일 시스템 구성에 관련된 유용한 기능을 추가적을 체크해 주면 된다.

저장하고 나온다. 이런 일련의 작업이 귀찮을 경우 ..

ftp://syszone.co.kr/pub/linux/server/lvspkg/xeon-p4-aic-usb 를 다운 받아

/usr/src/linux 에 놓아두고

--> Load an Alternate Configuration File 에서 불려와서 적용해도 된다.

단 위 커널 설정 파일은 Intel Xeon CPU, Aic79xx SCSI Adapter 환경에 맞는 설정이다.

```
# make dep
# make clean
# make bzImage
# make modules
# make modules_install
```

커널 컴파일 완료 후

```
# cp System.map  /boot/System.map-2.4.27-lvs
# cp arch/i386/boot/bzImage /boot/vmlinuz-2.4.27-lvs
# cd /boot
# ln -sf System.map-2.4.27-lvs  System.map
# ln -sf vmlinuz-2.4.27-lvs vmlinuz
```

lilo 설정

```
# vi /etc/lilo.conf
----------------------------------------------------------------
prompt
timeout=50
default=linux-lvs
boot=/dev/sda
map=/boot/map
install=/boot/boot.b
message=/boot/message
image=/boot/vmlinuz-2.4.20-8smp
label=linux
initrd=/boot/initrd-2.4.20-8smp.img
```

```
read-only
append="root=LABEL=/"
image=/boot/vmlinuz
label=linux-lvs
read-only
root=/dev/sda2
-----------------------------------------------------------------
```

```
# lilo
```

리부팅을 한다.

2.2. ipvsadm 설치 및 사용 방법

ipvsadm 은 LB 서버에서 클라이언트의 서비스 요청을 받아 실제 작업 서버로 요청 패킷을 포워팅 시키는 정책을 정의하는 명령어 이다. 부하분산 서버의 Virtual Server 정책은 모두 이 명령어로 정의하게 된다.

2.2.1 ipvsadm 설치

ipvsadm-1.21.tar.gz 파일을 /usr/local/src 밑에 옮겨 놓는다.

```
# cd /usr/local/src
# tar xzvf ipvsadm-1.21.tar.gz
# cd ipvsadm-1.21
# make
# make install
# ln -sf /sbin/ipvsadm /usr/sbin/ipvsadm          -> Encluster 사용 시 필요함.
# ipvsadm
---------------------------------------------------------------------
IP Virtual Server version 1.0.11 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port              Forward Weight ActiveConn InActConn
```

위와 같은 내용이 나오면 정상적으로 설치가 완료 된것이다.

2.2.2 ipvsadm 사용 방법

ipvsadm 옵션 설명

-A : 서비스를 추가 한다.

-D : 서비스 삭제

-E : 서비스 수정

-C : 정의된 모든 정책 삭제

-s : 스케줄러 선택 (wlc : weight least connection scheduling)

-a : add server (-e : EDIT , -d : Delete)

-t : TCP service 추가 (-u : UDP , -f : Firewall)

-r : 옵션 다음에 Real Server ip address 를 적는다.

-i : 패킷전달 방식 (ip tunneling 방식) , -g : Direct Routing 방식, -m : NAT

-w : wlc 방식에서 가중치 (기본 가중치는 1 로 주어짐)

--set tcp tcpfin udp : 연결 시간 제한 설정 (set connection timeout value)

-p : persistent 시간 지정

예) 정책 설정 방법

```
# ipvsadm -A -t 211.238.41.170:80 -s wlc -p 360
# ipvsadm
---------------------------------------------------------------------
IP Virtual Server version 1.0.11 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  211.238.41.170:http wlc persistent 360
---------------------------------------------------------------------

# ipvsadm -a -t 211.238.41.170:80 -r 211.238.41.165 -g -w 100
# ipvsadm -a -t 211.238.41.170:80 -r 211.238.41.166 -g -w 100
# ipvsadm
---------------------------------------------------------------------
IP Virtual Server version 1.0.11 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  211.238.41.170:http wlc persistent 360
  -> 211.238.41.166:http          Route   100    0          0
  -> 211.238.41.165:http          Route   100    0          0
---------------------------------------------------------------------
```

2.3 LB 서버와 Real Server 의 네트워크 구성 설정

2.3.1 LB 서버 네트워크 구성

LB 서버에는 특정 인터페이스에 VIP 를 설정하여야 한다. 여기서는 eth0:0 인터페이스에 VIP 를 설정하도록 한다.

-> eth0 : real IP

-> eth0:0 virtual IP

-> VIP : 211.238.41.170

-> Web Server-1 : 211.238.41.165

-> Web Server-2 : 211.238.41.164

ifconfig eth0:0 211.238.41.170 netmask 255.2555.255.255 broadcast 211.238.41.170 # route add -host 211.238.41.170 dev eth0:0

route -n 으로 확인하여 해당 flags 값이 UH 면 된다.

route –n

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
211.238.41.170	0.0.0.0	255.255.255.255	UH	0	0	0	eth0
211.238.41.160	0.0.0.0	255.255.255.224	U	0	0	0	eth0
169.254.0.0	0.0.0.0	255.255.0.0	U	0	0	0	eth0
127.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0	lo
0.0.0.0	211.238.41.190	0.0.0.0	UG	0	0	0	eth0

---

kernel forward parameter 값을 1 로 수정한다.

echo 1 > /proc/sys/net/ipv4/ip_forward

매번 부팅때마다 적용하기 위해서는 /etc/sysctl.conf 에서 해당 부분 설정을

0 에서 1 로 수정한다.

net.ipv4.ip_forward = 1

*** 만일 NAT 구성 시에는 /etc/sysconfig/network 파일을 열어 아래 내용을 추가해 준다.

IP_FORWARD(대문자)=true(소문자)

- **ipvsadm 정책 설정**

ipvsadm -A -t 211.238.41.170:80 -s wlc

ipvsadm -A -t 211.238.41.170:22 -s 짖

ipvsadm -a -t 211.238.41.170:80 -r 211.238.41.165 -g

ipvsadm -a -t 211.238.41.170:80 -r 211.238.41.164 –g

ipvsadm -a -t 211.238.41.170:22 -r 211.238.41.165 –g

```
# ipvsadm
---------------------------------------------------------------------- IP Virtual Server version 1.0.11
(size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  211.238.41.170:ssh 짖
  -> 211.238.41.165:ssh           Route   1      0          0
TCP  211.238.41.170:http wlc
  -> 211.238.41.165:http          Route   1      0          0
  -> 211.238.41.164:http          Route   1      0          0
----------------------------------------------------------------------
```

2.3.2 Real Server 네트워크 구성

Real Server 에서는 LB 로 부터 분배 되어지는 패킷을 받기 위해 DR, NAT, Tun 별로 모두 네트워크 구성이 달라진다.
NAT 의 경우는 별도의 VIP 에 대한 Alias Device 를 가질 필요가 없다. 단지 네트워크 구성에서 Gateway 를 LB 서버의 통신 가능한 IP 로 정해주면 된다.
DR 이나 Tun 의 경우에는 Real Server 에서도 VIP 에 대한 IP 를 Non arp alias Device 에 설정을 해주어야 한다. 그래야 LB 서버에서 포워딩되는 패킷을 받아 들일수가 있다

여기서는 Direct routing 의 방식으로 설정을 하도록 하겠다.
-> eth0 : Real IP
-> lo:0 : VIP
ifconfig lo:0 211.238.41.170 netmask 255.255.255.255 broadcast 211.238.41.170
route add -host 211.238.41.170 dev lo:0
여기서 별도의 non arp alias device 를 잡지 않아도 iptables 의 rediect 기능을 이용하여 Direct routeing 방식으로 네트워크를 구성할 수 있다.
Encluster 에서는 이 방법을 사용하고 있다.
iptables -t nat -A PREROUTING -j REDIRECT -d 211.238.41.170 -p tcp
iptables -t nat -A PREROUTING -j REDIRECT -d 211.238.41.170 -p u 에
Real Server 네트워크 구성에서 VIP 가 설정 되는 Device 의 경우 반드시 arp caching 남으면 안된다.
arp caching 남게 되면 vip 를 통해 초기 접속된 Real Server 로만 계속 접속을 시도하게 됨으로 LB 서버의 분배 스케줄링에 문제가 발생하게 된다.
다음 장에서 DR 과 tun 방식에서의 Arp 캐싱 문제 해결 방법에 대해 알아보자.

2.4. Direct Routing 방식에서의 ARP 문제 해결

arp caching 문제 해결은 2.1 장에서 얘기한 hidden patch 를 통해 해결 할수 있다.
하지만 커널 2.4.26 이상부터는 별도의 hidden patch 없이 arp_ignore , arp_announce 값을 이용하여 arp 문제를 해결 할수 있다

**** 참고 ****

하지만 http://www.ssi.com/~ja 에는 2.4.26~27 버전대의 hidden patch 가 올려져 있지만 hidden patch 를 한경우와 arp_ignore , arp_announce 통해 처리한 경우와 다른 점은 찾지 못했당.

- hidden patch 를 통해 arp 캐싱 문제 해결

```
# echo 1 > /proc/sys/net/ipv4/conf/all/hidden
# echo 1 > /proc/sys/net/ipv4/conf/lo/hidden
```

- arp_ignore , arp_announce 통해 해결하는 방법

```
# echo 1 > /proc/sys/net/ipv4/conf/all/arp_ignore
# echo 2 > /proc/sys/net/ipv4/conf/all/arp_announce
# echo 1 > /proc/sys/net/ipv4/conf/lo/arp_ignore
# echo 2 > /proc/sys/net/ipv4/conf/lo/arp_announce
```

매 부팅때 마다 위 설정을 유지 하기 위해서는 위의 명령 내용을 /etc/rc.d/rc.local 에 적어 주던지 아님 /etc/sysctl.conf 파일 내용에 해당 내용을 아래와 같이 추가하도록 한다.

```
# cat /etc/sysctl.conf
-----------------------------------------------------------------
net.ipv4.ip_forward = 1        ## 0 에서 1 로 수정
# 아래는 추가
net.ipv4.conf.lo.arp_ignore = 1
net.ipv4.conf.lo.arp_announce = 2
net.ipv4.conf.all.arp_ignore = 1
net.ipv4.conf.all.arp_announce = 2
# hidden patch 적용 경우에는 아래 추가
net.ipv4.conf.all.hidden = 1
net.ipv4.conf.lo.hidden = 1
-----------------------------------------------------------------
```

2.5. 부하 분산 테스트

앞장에서 설정한 정책에 따라 잘 분산이 되는지 확인해 보도록 하자.

확인에 앞서 먼저 정확한 분산 확인을 위해 apache 의 httpd.conf 의 설정 값을 수정하도록 한다.

KeepAlive = Off

MaxKeepAliveRequests 1

그리고 웹 페이지 테스트를 할수 있는 index 페이지를 만들어 보도록 하자.

vi index.html

```
<html>
<head>
<title>node01</title>
</head>
<body>
<font size=5 color=red> node01 </font>
</body>
</html>
```

그런 후 브라우저를 열어서 VIP 로 웹 접속을해보도록 하자.

웹 접속이 이루어 졌으면 브라우저 새로고침을 계속 눌려보아, 화면이 서버 별로 변경이 되는지를 확인하면 된다.

이밖에 아래 Perl 스크립터를 이용하여 확인을 해보아도 된다. 아래 스크립터로 테스트를 하기 위해서는 index 웹페이지의 title 을 반드시 적어 주어야 한다.

아래 스크립터는 웹페이지의 title 내용을 출력하라는 내용의 스크립트다.

vi get_title.pl

```
#!/usr/bin/perl –w
#
#클러스터 서버들의 타이틀을 가져오는 프로그램.
#
    use LWP::UserAgent;
    use HTTP::Request;
    use HTTP::Response;
    use URI::Heuristic;
    my $raw_url = shift or die "usage: $0 url₩n";
    for ( $i = 0 ; $i <= 100 ; $i++){      # 이곳의 숫자를 변경해 반복횟수를 변경한다.
    my $url = URI::Heuristic::uf_urlstr($raw_url);
    $| = 1;
    printf "%s =>₩n₩t", $url;
```

```
        my $ua = LWP::UserAgent->new();
        $ua->agent("schmozilla/v9.14 Platinum");
        my $req = HTTP::Request->new(GET=>$url);
        $req->referer("http://wizard.yellowbrick.oz");
        my $response = $ua->request($req);
        if ($response->is_error()){
            printf "%s₩n", $response->status_line;
            }
        else{
            my $count;
            my $bytes;
       my $content = $response->content();
       $bytes = length $content;
       $count = ($content =~ tr/₩n/₩n/);
       printf "%s (%d lines, %d bytes)₩n", $response->title(), $count, $bytes;
          }
```

```
    # chmod 755 get_title.pl
    # ./get_title.pl http://211.238.41.170

          http://211.238.41.170 =>
        node02 (12 lines, 109 bytes)
          http://211.238.41.170 =>
        node01 (12 lines, 108 bytes)
          http://211.238.41.170 =>
        node02 (12 lines, 109 bytes)
          http://211.238.41.170 =>
        node01 (12 lines, 108 bytes)
          http://211.238.41.170 =>
        node02 (12 lines, 109 bytes)
          http://211.238.41.170 =>
        node01 (12 lines, 108 bytes)
          http://211.238.41.170 =>
        node02 (12 lines, 109 bytes)
```

2.6. Server failed 를 감지하는 MON 적용 하기

지금까지 기본적인 로드 밸런싱(작업분배)은 설정에 대해 알아보았다.
하지만 real_server(앞으로 real 이라하겠다.)중 하나가 죽었도 master_lvs 에서는 스케줄 규칙에 따라 패킷을 죽은 real 에도 정해진 규칙대로 분배하게 된다.
그렇기 때문에 패킷이 죽은 real 로 보내질때는 접속이 되지 않는다.
이를 방지하기 위해서 mon 이란 프로그램을 설치한다. mon 은 정해진 시간마다 각 real 서버들을 감시 하면서 죽은 real 서버가 발견대면 ipvsadm 에서 죽은 real 서버로 분배대는 작업 규칙을 제거하여 더이상 작업을 분배하지 않도록 한다.
그리고 죽었던 real 서버가 살아나면 이또한 감지하고 다시 정해진 작업분배를 계속 하게 된다.
이제 mon 설치 방법에 대하여 알아보도록 하자.

2.6.1 Mon 설치 준비 하기

mon 은 기본적으로 perl 로 만들어진 각 서비스 체크 모듈을 가지고 서비스의 상태를 파악하여 해당 서비스가 죽은 경우 특정 액션을 취할 수 있게 해주는 프로그램이다.
이를 이용하면 ipvsadm 정책을 새로이 만들 수도 있고, 기타 다른 곳에도 응용할 수 있다.

mon 을 이용하기 위해서는 기본적으로 perl ver 5.x 이상이 설치 되어져 있어야 한다.

또 몇가지 기본 perl modules 과 fping 정도의 프로그램을 먼저 설치 해 두어야 한다.
perl modules 은 CPAN 을 이용하여 쉽게 설치 할수 있고, 직접 다운 받아 설치할 수도 있다.
perl modules 설치 전에 perl 용 시스템 header 파일을 만들어야 한다.

```
# cd /usr/include
# h2ph *.h sys/*.h asm/*.h
```

- CPAN 이용방법

```
# perl-MCPAN -e shell
```

처음 이용할 경우 다음과 같은 기본 설정 단계가 나온다.

```
---------------------------------------------------------------------
/usr/lib/perl5/5.8.0/CPAN/Config.pm initialized.
CPAN is the world-wide archive of perl resources. It consists of about
100 sites that all replicate the same contents all around the globe.
Many countries have at least one CPAN site already.
The resources found on CPAN are easily accessible with the CPAN.pm module. If you
```

want to use CPAN.pm, you have to configure it properly. If you do not want to enter a dialog now, you can answer 'no' to this question and I'll try to autoconfigure.

(Note: you can revisit this dialog anytime later by typing 'o conf init' at the cpan prompt.)

Are you ready for manual configuration? [yes] -> yes

The following questions are intended to help you with the
configuration. The CPAN module needs a directory of its own to cache
important index files and maybe keep a temporary mirror of CPAN files.
This may be a site-wide directory or a personal directory.
First of all, I'd like to create this directory. Where?
CPAN build and cache directory? [/root/.cpan] .

기본 내용으로 설정한다.

cpan shell -- CPAN exploration and modules installation (v1.61)
ReadLine support available (try 'install Bundle::CPAN')
cpan>

---------- cpan> install Time::Period
cpan> install Time::HiRes
cpan> install Convert::BER
cpan> install Mon::Client
cpan> install Mon::Protocol
cpan> install Mon::SNMP

- 직접 다운 받아 설치 하기

ftp://ftp.bora.net/pub/CPAN 에서 원하는 perl modules 을 다운 받는다.

tar xzvf Time-Hires-01.20.tar.gz

cd Time-Hires-01-20

perl Makefile.PL

make
make test
make install

- fping 설치 하기

```
# cd /usr/local/src
# tar xjvf fping-2.2b1.tar.bz2
# cd fping-2.2.b1
# ./configure
# make

CONFIG_FILES= CONFIG_HEADERS=./config.h ./config.status
creating ./config.h
gcc -c -DHAVE_CONFIG_H -I. -I. -I.     -g -O2 fping.c
fping.c:222: conflicting types for `sys_errlist'
/usr/include/bits/sys_errlist.h:28: previous declaration of `sys_errlist'
make: *** [fping.o] 오류 1
```

-> 이와 같이 오류가 발생하면 fping.c 파일의 222번 줄을 수정한다.

```
/* externals */

extern char *optarg;
extern int optind,opterr;
extern char *_sys_errlist[];   -> *sys_errlist[];
extern int h_errno;

#ifdef __cplusplus
}
```

.. 그런 후 ..

```
# make
# make install
```

2.6.2 Mon 설치 하기

```
# tar xzvf mon-0.99.2.tar.gz -c /usr/lib
# cd /usr/lib
# mv mon-0.99.2 mon
# cd mon
# mkdir /etc/mon   /var/log/mon
# install m644 doc/mon.8 /usr/man/man8
```

```
# install m644 doc/moncmd.1 /usr/man/man1
# install m644 doc/monshow.1 /usr/man/man1
# install m644 etc/auth.cf /etc/mon
# install m644 etc/example.cf /etc/mon
# install m700 etc/S99mon /etc/rc.d/init.d/mon
# vi /etc/rc.d/init.d/mon 을 수정한다.
-------------------------------------------------------------
#!/bin/sh
#
    # start/stop the mon server
#
# You probably want to set the path to include
# nothing but local filesystems.

 #
# chkconfig: 2345 99 10
# description: mon system monitoring daemon
# processname: mon
# config: /etc/mon/mon.cf
# pidfile: /var/run/mon.pid
#
PATH=/bin:/usr/bin:/sbin:/usr/sbin
export PATH

# Source function library.
. /etc/rc.d/init.d/functions

# See how we were called.
case "$1" in
    start)
    echo -n "Starting mon daemon: "
    daemon /usr/lib/mon/mon -f -c /etc/mon/mon.cf   --> 이부분 -f 추가
-------------------------------------------------------------------

# vi /etc/service
-------------------------------------------------------------------
맨 아래다가..

mon                 2583/tcp            # MON
```

```
mon                 2583/udp          # MON traps
```

추가 ..

```
# cp /etc/mon/example.cf /etc/mon/mon.cf
```

이것으로 mon 설치를 완료한다.
간단한 테스트를 해보도록 하자
mon 의 fping modules 을 이용한 서버 상태 확인 하는 테스트이다.

```
# /usr/lib/mon/mon.d/fping.monitor syszone.co.kr

start time: Thu Aug 26 18:11:37 2004
end time   : Thu Aug 26 18:11:37 2004
duration   : 0 seconds

-------------------------------------------------------------------------
reachable hosts                                 rtt
-------------------------------------------------------------------------
syszone.co.kr                                   0.19 ms
```

다음장에서는 실제 mon 을 이용하여 LB 서버에 Fail Over 기능을 추가 해보자.

2.6.3 Mon 을 이용한 Load Balance 에 Fail Over 기능 추가하기

/usr/lib/mon/mon.d 디렉토리 안의 file 들은 각 시스템의 해당 서비스를 모니터링 하는 perl scripts 이다. mon 의 장점과 단점이 될수 있는 부분이 기본적으로 포함되어져 있는 monitor scripts 가 없는 경우 일반인들이 싶게 개별 사이트의 특수한 프로그램을 감시 할수 없다는 것이다. 하지만 약간의 Perl 지식만 갖춘 상태라고 하면 어떤 형태의 moniter modules 도 쉽게 만들 수 있을 것이다.
이장에서는 mon을 이용하여 virtual address 정책을 실제 작업가 죽어 있을 경우 정책에서 자동 제거하고 다시 살아나면 다시 포함하는 기능을 추가하는 방법에 대해 알아 보겠다.

mon 의 기본 설정 파일은 한개이다. /etc/mon/mon.cf 가 그것이다.
이 설정 파일의 내용만 이해하면 시스템 엔지니어가 원하는 어떤 형태의 감시

툴도 만들 수 있을 것이다.

```
# vi /etc/mon/mon.cf
--------------------------------------------------------------------------
# global options
#
cfbasedir     = /etc/mon            # mon.cf 위치
logdir        = /var/log/mon
alertdir      = /usr/lib/mon/alert.d
mondir        = /usr/lib/mon/mon.d
maxprocs      = 20
histlength = 100
randstart = 30s

# authentication types:
#     getpwnam          standard Unix passwd, NOT for shadow passwords
#     shadow            Unix shadow passwords (not implemented)
#     userfile          "mon" user file


authtype = userfile       # mon.cgi 사용시 로그인 가능 기능
userfile = /etc/mon/mon.user

dtlogging = yes
dtlogfile = downtime.log

############## 호스트 그룹 지정 ####################
# hostgroup 은 실제 모니터링을 해야할 서버와 그 서버의 별칭을 정하는 곳
# 아래에서 HTTP-server1 은 서비스 별칭이고 www1 은 호스트네임이다.
# 서비스 별칭은 실제 서비스에 대한 모니터링 관련 설정을 하는 watch 에서
# 사용된다.

hostgroup HTTP-server1 www1

hostgroup HTTP-server2 www2

watch HTTP-server1
    service ping
    description ping servers in www1
```

```
        interval 5s
        monitor fping.monitor
        period wd {Sun-Sat}
    #       alert mail.alert alang@clunix.com
    #       alert page.alert alang@clunix.com
            alertevery 10m
            alertafter 2 30m
            alert test.alert "www1 Server is DOWN"
            upalert test.alert "www1 Server is UP"
        service HTTP
        description http to servers in www1
        interval 3s
        monitor http.monitor
     #   depend HTTP-server1:ping
        period wd {Sun-Sat}
            alertevery 30m
    #       alert mail.alert alang@clunix.com
    #       alert page.alert alang@clunix.com
            alert test.alert "HTTPd is DOWN"
            upalert test.alert "HTTPd is UP"
            alert lvs.alert -P tcp -V 211.238.41.170:80 -R 211.238.41.165:80 ₩
                -W 100 -F dr -X down
            upalert lvs.alert -P tcp -V 211.238.41.170:80 -R 211.238.41.165:80 ₩
                -W 100 -F dr

    watch HTTP-server2
        service ping
        description ping servers in www2
        interval 5s
        monitor fping.monitor
        period wd {Sun-Sat}
     #       alert mail.alert alang@clunix.com
     #       alert page.alert alang@clunix.com
            alertevery 10m
            alertafter 2 30m
            alert test.alert "www2 Server is DOWN"
            upalert test.alert "www2 Server is UP"
        service HTTP
        description http to servers in www2
```

```
        interval 3s
        monitor http.monitor
    #   depend HTTP-server2:ping
        period wd {Sun-Sat}
            alertevery 30m
    #         alert mail.alert alang@clunix.com
    #         alert page.alert alang@clunix.com
            alert test.alert "www2 HTTPd is DOWN"
            upalert test.alert "www2 HTTPd is UP"
           alert lvs.alert -P tcp -V 211.238.41.170:80 -R 211.238.41.166:80 ₩
                -W 100 -F dr -X down
           upalert lvs.alert -P tcp -V 211.238.41.170:80 -R 211.238.41.166:80 ₩
                -W 100 -F dr
```

이제 설정을 저장 후 Mon 데몬을 시작해 보자

```
# /etc/rc.d/init.d/mon start
```

restart 는 불안하니 stop , start 만 사용하도록 하자.

2.6.4 Mon.cgi 를 이용하여 웹에서 Mon 제어하기

moncgi란 test-mode가 아닌 그래픽 모드에서 서버를 모니터링 할 수 있게끔 해주는 툴이다. 물런 설치 하지 않아도 LB Fail over 기능에는 전혀 상관 없다.

설치 과정에 대해 알아보자

Mon 웹프로그램을 관리할 시스템 계정 생성

```
# adduser -u 90 -c "MonCGI Apache User" -d /usr/local/moncgi ₩
    -s /sbin/nologin moncgi
```

mon 웹프로그램에 로그인할 계정 생성

```
          # /usr/local/apache/bin/htpasswd -c /etc/mon/mon.user moncgi
# cd /usr/local/src
# tar xzvf apache_1.3.31.tar.gz
# cd apache_1.3.31
# vi config.layout      -> 아래와 부분을 추가함
```

```
<Layout moncgi>
    prefix:          /usr/local/moncgi
    exec_prefix:     $prefix
    bindir:          $exec_prefix/bin
    sbindir:         $exec_prefix/bin
    libexecdir:      $exec_prefix/libexec
    mandir:          $prefix/man
    sysconfdir:      /etc/mon/moncgi
    datadir:         $prefix
    iconsdir:        $datadir/icons
    htdocsdir:       $datadir/htdocs
    manualdir:       $htdocsdir/manual
    cgidir:          $datadir/cgi-bin
    includedir:      $prefix/include
    localstatedir: /var/log/mon
    runtimedir:      $localstatedir/logs
    logfiledir:      $localstatedir/logs
    proxycachedir: $localstatedir/proxy
</Layout>
```

```
# ./configure --with-layout=moncgi --show-layout
# ./configure --with-layout=moncgi --with-port=9000 --server-uid=moncgi  ₩
    --server-gid=moncgi --disable-module=autoindex --disable-module=imap ₩
    --disable-module=include --disable-module=negotiation ₩
    --disable-module=status --disable-module=userdir
# make
# make install
# cd /etc/mon/moncgi
# rm -rf access* srm* magic*
# grep -v "#" httpd.conf | grep ^. > tt      --> 주석  처리된 내용을 삭제함.
# mv ./tt ./httpd.conf
# vi httpd.conf
```

```
######################### 전체 환경 설정 ####################
ServerType                  standalone
ServerRoot                  "/usr/local/moncgi"
PidFile                     /var/log/mon/moncgi/httpd.pid
ScoreBoardFile              /var/log/mon/moncgi/httpd.scoreboard
```

```
        Timeout                 300
        KeepAlive               On
        MaxKeepAliveRequests    100
        KeepAliveTimeout        15
        MinSpareServers         1
        MaxSpareServers         2
        StartServers            0
        MaxClients              10
        MaxRequestsPerChild     0

        Listen          211.238.41.167:9000

        ######################## 주 서버 설정 ######################

        Port 9000
        User moncgi
        Group moncgi
        ServerAdmin root@www3.clunix.org

        DocumentRoot "/usr/local/moncgi/cgi-bin"

        <Directory "/usr/local/moncgi/cgi-bin">
                Options ExecCGI
                AllowOverride None
                AddHandler cgi-script .cgi
        #       Order deny,allow
        #       Deny from all
        #       Allow from 211.238.41.180
        </Directory>

        DirectoryIndex mon.cgi

        UseCanonicalName On

        TypesConfig /etc/mon/moncgi/mime.types
        DefaultType text/plain

        HostnameLookups Off
```

```
ErrorLog /var/log/mon/moncgi/error_log
LogLevel warn
LogFormat "%h %l %u %t ₩"%r₩" %>s %b" common
CustomLog /var/log/mon/moncgi/access_log common

ServerSignature On

BrowserMatch "Mozilla/2" nokeepalive
BrowserMatch "MSIE 4₩.0b2;" nokeepalive downgrade-1.0 force-response-1.0
BrowserMatch "RealPlayer 4₩.0" force-response-1.0
BrowserMatch "Java/1₩.0" force-response-1.0
BrowserMatch "JDK/1₩.0" force-response-1.0

-----------------------------------------------------------------

# cd /usr/local/moncgi 로 이동
# rm -rf htdocs icons libexec man cgi-bin/*
# cd /etc/rc.d/init.d/
# ln -sf /usr/local/moncgi/bin/apachectl moncgi
# /etc/rc.d/init.d/moncgi restart

이제 moncgi 가 재대로 동작하는지 확인

# lsof -i | grep 9000
-------------------------------------------------------------------------
httpd       3805     root     16u   IPv4 419398          TCP www3.clunix.org:9000 (LISTEN)
httpd       3816 moncgi     16u   IPv4 419398          TCP www3.clunix.org:9000 (LISTEN)
httpd       3817 moncgi     16u   IPv4 419398          TCP www3.clunix.org:9000 (LISTEN)
httpd       3818 moncgi     16u   IPv4 419398          TCP www3.clunix.org:9000 (LISTEN)
httpd       3819 moncgi     16u   IPv4 419398          TCP www3.clunix.org:9000 (LISTEN)
           httpd       3820 moncgi     16u   IPv4 419398          TCP www3.clunix.org:9000 (LISTEN)

# chkconfig --add moncgi
# chkconfig --level 3 moncgi on

moncgi를 설치 하기 위해서는 몇가지 perl modules 이 더 필요하다.
필요한 모듈로는 Crypt::TripleDES , Math::TrulyRandom  이 있다.
앞서 설명한 CPAN 을 이용하도록 하자.
```

```
# perl -MCPAN -e shell
cpan > install Crypt::TripleDES
cpan > look Math::TrulyRandom
# perl Makefile.PL
# make && make install
# exit
# exit
#
```

mon.cgi-1.52.tar.gz 을 다운 받는다. (http://www.nam-shub.com/files/)

```
# mkdir /etc/mon/moncgi
# /usr/local/src
# tar xzvf mon.cgi-1.52.tar.gz
# cd mon.cgi-1.52
# install m755 mon.cgi /usr/local/moncgi/cgi-bin
# install m644 config/mon.cgi.cf /etc/mon/moncgi
# install m700 util/moncgi-appsecret.pl /usr/sbin/moncgi-appsecret
# cd .. ; rm -rf mon.cgi*
# cd /usr/local/moncgi/cgi-bin
```

mon.cgi 를 수정한다.

```
# vi mon.cgi
```

$moncgi_config_file = "/etc/mon/moncgi/mon.cgi.cf" 176 줄

/etc/mon/moncgi/mon.cgi.cf 에서 Reloas_time, must_login , Login_expire_time 을 설정이 되어 있는지 확인을 해본다. 환경에 맞게 수정하라.
그런후 http://IP:9000/mon.cgi
(접근후 계정과 password를 넣는 창이 있을 것이다. 이것은 MON에서 추가해준 유저를 넣으면 된다.)

Mon 데몬을 자동으로 실행하는 init script 를 만들어 보자

```
# vi /etc/rc.d/init.d/puck
```

```
#!/bin/sh
```

```
#
# load balancer daemon scripts
#
PATH=/bin:/usr/bin:/sbin:/usr/sbin
export PATH
IPVSADM=/sbin/ipvsadm
MON=/usr/lib/mon/mon
RETVAL=0
#Source function library.
. /etc/rc.d/init.d/functions
case "$1" in
 start)
   if [ -x $IPVSADM ]
   then
#
# ipvs system 설정 부분
#
   echo 1 > /proc/sys/net/ipv4/ip_forward
   ifconfig eth0:0 211.238.41.170 netmask 255.255.255.255 ₩
      broadcast 211.238.41.170 up
   route add -host 211.238.41.170 dev eth0:0
#   echo 1 > /proc/sys/net/ipv4/conf/all/arp_ignore
#   echo 2 > /proc/sys/net/ipv4/conf/all/arp_announce
#   echo 1 > /proc/sys/net/ipv4/conf/eth0/arp_ignore
#   echo 2 > /proc/sys/net/ipv4/conf/eth0/arp_announce

   $IPVSADM -A -t 211.238.41.170:80 -s wlc
   $IPVSADM -a -t 211.238.41.170:80 -r 211.238.41.165:80 -g -w 100
   $IPVSADM -a -t 211.238.41.170:80 -r 211.238.41.166:80 -g -w 100
#
# 추가되는 real_server 는 아래에 같은형식으로 추가하면 된다.
# $IPVSADM -a -t 211.238.41.170:80 -R 211.238.41.165 -g -w 2
# $IPVSADM -a -t 211.238.41.170:80 -R 211.238.41.166-g -w 2
#


   echo -n "started loadbalancer daemon:"
   daemon   $MON -f -c /etc/mon/mon.cf
   RETVAL=$?
```

```
    echo
    [ $RETVAL = 0 ] && touch /var/lock/subsys/punk
    echo
    fi
    ;;
    stop)
    if [ -x $IPVSADM ]
then
echo -n "puck daemon stopping..."
$IPVSADM -C
ifconfig eth0:0 down
killproc mon
rm -f /var/lock/subsys/punk
killall http.monitor
echo -n "puck daemon killed"
echo
    fi
    ;;
    *)
    echo "Usage : puck {start|stop}"

exit 1
esac
exit 0
```

이제 /etc/rc.d/init.d/puck initscripts 로 mon 을 손쉽게 제어할 수 있다.
LB 서버에서 /etc/rc.d/init.d/puck start 한 후 Web Server-1, Web Server-2의 httpd 데몬을 stop, start 시키면서 ipvsadm tables 에 상황에 맞게 적용되는지를 확인한다.

2.7. LB 서버간의 이중화 HA 구축 하기 (Heartbeat)

Load Balance 방식의 최대 단점 중 하나가 LB 서버가 다운 될을 경우 모든 서비스가 정지된다. 그러므로 LB 서버의 이중화는 필수적인 사항이다. 이를 위해 heartbeat 란 프로그램을 이용하여 LB 서버를 이중화 해야 한다.

heartbeat 프로그램을 다운 받아 /usr/local/src 에 놓아둔다.

heartbeat 에 이용되는 프로그램은 다음과 같다.
http://www.linux-ha.org/ 에서 heartbeat 최신버전을 다운 받는다.

heartbeat-pils-1.2.2-8.rh.9
heartbeat-stonith-1.2.2-8.rh.9
heartbeat-1.2.2-8.rh.9

2.7.1 Heartbeat 설치 및 기본 설정

Heartbeat는 Master_lvs 와 Slave_lvs 에 모두 설치한다.

rpm -Uvh heartbeat-*

설정파일은 /etc/ha.d 에 있다.

cd /etc/ha.d

heartbeat 설정은

ha.cf
haresources
authkeys

rpm -q heartbeat -d 라고 하면 설정파일의 샘플파일이
/usr/share/doc/packages/heartbeat 란 디렉토리에 있다는 것을 알수 있다.

샘플 파일을 /etc/ha.d 로 복사한다.

cp ha.cf /etc/ha.d
cp haresources /etc/ha.d
cp authkeys /etc/ha.d

복사가 완료되었으면 다시 /etc/ha.d 디렉토리돌아온다.

/etc/ha.d/ha.cf 설정 내용 ----------

debugfile /var/log/ha-debug
logfile /var/log/ha-log

```
logfacility        local0
keepalive 2
deadtime 5
hopfudge 1
udpport 1001
udp        eth0
node       www3        # Master LB Server
node       www2        # Slave LB Server
```

/etc/ha.d/authkeys 설정 내용 ----------

```
#auth 1
#1 crc
#2 sha1 HI!
#3 md5 Hello!
auth 1
1 sha1 HI!
```

/etc/ha.d/haresource 설정 내용 ----------

```
# 설정 형식에 맞게 설정해야 합니다. 설정형식은...
# masternode_name     VIP                  DAEMON_name
# DAEMON_name 은 /etc/rc.d/init.d 밑에 있는 initscripts 를 적으면 된다

www3       211.238.41.170    puck
```

여기까지 Master_LB 에 대한 설정은 모두 끝났다.

2.7.2 Slave LB 서버가 Director 와 Real Service 의 두가지 역활 담당 시 Slave LB 서버 설정

만일 Slave_LB 서버를 LB 역활로만 둔다라고 하면 Master_LB 서버와 동일하게

Mon 설정과 Heartbeat 설정을 해주면 된다.
하지만 Slave_LB 서버를 평소에 RealServer 역활을 수행하다가 Master LB 서버에
문제 발생 시 Real Server 에서 LB 서버로 역활을 전환하는 방식으로 한다고 하면
Slave_LB 서버의 Mon 과 Heartbeat 설정에서 몇가지 수정을 해주어야 한다.

먼저 Slave_LB 서버가 Real Server 역활을 수행 시 lo:0 를 추가할때 쓰이는
스크립터를 만들도록 한다.

/etc/ha.d/lvs-r.sh----------

```
#!/bin/sh

VIP=211.238.41.170

ifconfig lo:0 $VIP netmask 255.255.255.255 broadcast $VIP up
route add -host $VIP dev lo:0

  /sbin/sysctl -w net.ipv4.conf.all.arp_ignore=1
  /sbin/sysctl -w net.ipv4.conf.all.arp_announce=2
  /sbin/sysctl -w net.ipv4.conf.lo.arp_ignore=1
  /sbin/sysctl -w net.ipv4.conf.lo.arp_announce=2

echo 1 > /proc/sys/net/ipv4/ip_forward
```

---

lvs-r.sh 의 퍼미션에 실행 권한을 준다.

```
# chmod 755 /etc/ha.d/lvs-r.sh
```

/etc/rc.d/init.d/puck 을 수정한다.

/etc/rc.d/init.d/puck ----------

```
case "$1" in
  start)
  if [ -x $IPVSADM ]
  then
#
```

```
        # ipvs system 설정 부분
        #
          echo 1 > /proc/sys/net/ipv4/ip_forward
          ifconfig eth0:0 211.238.41.170 netmask 255.255.255.255 ₩
                broadcast 211.238.41.170 up
          route add -host 211.238.41.170 dev eth0:0
          /sbin/sysctl -w net.ipv4.conf.all.arp_ignore=0
          /sbin/sysctl -w net.ipv4.conf.all.arp_announce=0
          /sbin/sysctl -w net.ipv4.conf.eth0.arp_announce=0
          /sbin/sysctl -w net.ipv4.conf.eth0.arp_announce=0
          $IPVSADM -A -t 211.238.41.170:80 -s wlc
          $IPVSADM -a -t 211.238.41.170:80 -r 211.238.41.165:80 -g -w 100
          $IPVSADM -a -t 211.238.41.170:80 -r 211.238.41.166:80 -g -w 100
        #
        # 추가되는 real_server 는 아래에 같은형식으로 추가하면 된다.
        # $IPVSADM -a -t 211.238.41.170:80 -R 211.238.41.165 -g -w 2
        # $IPVSADM -a -t 211.238.41.170:80 -R 211.238.41.166-g -w 2
        #


          echo -n "started loadbalancer daemon:"
          daemon   $MON -f -c /etc/mon/mon.cf
          RETVAL=$?
          echo
          [ $RETVAL = 0 ] && touch /var/lock/subsys/punk
          echo
          fi
          ;;
          stop)
          if [ -x $IPVSADM ]
        then
        echo -n "puck daemon stopping..."
        $IPVSADM -C

        ######## HA 설정 추가 내용 ###################################

        ifconfig eth0:0 down
        /etc/ha.d/lvs-r.sh
```

```
        ############################################################

        killproc mon
        rm -f /var/lock/subsys/punk
        killall http.monitor

        ------------------------------------------------------------------
```

그담에 /etc/rc.d/init.d/heartbeat 파일에 한줄만 추가해 준다.

/etc/rc.d/init.d/heartbeat 수정 내용 ----------

.

```
        case "$1" in
          start)
                StartHA
                RC=$?
        .
        .


          stop)
            RunStartStop "pre-stop"
            StopHA
            RC=$?
            Echo
                if
                  [ $RC -eq 0 ]
                then
                  rm -f $LOCKDIR/$SUBSYS
                fi
                RunStartStop post-stop $RC

        ######### Slave-HA 추가 사항 ######################

                /etc/ha.d/lvs-r.sh

        ####################################################
```

;;

여기까지가 설정의 모든 것이다.

이제 Master_LB 와 Slave_LB 의 heartbeat 데몬을 가동시킨다.

/etc/rc.d/init.d/heartbeat start

이제 High Availability 가 이루어 지는지 테스트를 해보자.

A. 정상적으로 작업분배가 이루어 지는지 테스트를 한다.
B. Master_LB 의 Lan 선을 뽑아 버린다. 그럼..Slave_LB 가 Master_LB 의 권한을 무사히 위임 받는지를 확인한다.
C. 다시 Master_LB 를 살려 보도록 하자. 그럼..Slave_LB 가 Master_LB의 역활을 다시 반납하고 다시 real 서버로 돌아가는지를 확인한다.
D. 다른 노드들도 각각 Lan선을 뽑고 난뒤에도 서비스에 지장이 없는지를 확인 하도록 한다.

위의 4가지 테스트가 모두 성공적으로 완료 된다면 heartbeat 와 mon 을 이용한 고가용성 시스템 구축이 완성된것이다.

마지막으로 서버 환경에 맞추어서 heartbeat 와 mon 의 데몬 감시 주기 시간을 조절하면 된다.

3. Encluster 를 이용한 리눅스 부하분산 클러스터 구축

3.1. Encluster 1.6을 이용한 리눅스 부하분산 클러스터 구축

3.1.1 Encluster 1.6 의 구성 및 기능 소개

3.1.2 Encluster 1.6 설치 및 기본 설정

ftp://clunix@technet.clunix.com/EnCluster 에서 최신 버전을 다운 받는다.
(EnCluster-1.6.5h-9-i386 버전 권장)

클루닉스에서 제공하는 Kernel rpm 혹은 kernel.org 에서 제공하는 Kernel Source 에 ipvs 패치 및 적용을 한 커널을 준비 하여야 한다.

ipvsadm 은 클루닉스에서 제공하는 rpm 패키지를 설치하길 권장 한다.
source 설치 시는 반드시 /usr/sbin 밑에 ipvsadm 링크를 걸어 주어야 한다.

```
# ln -s /sbin/ipvsadm /usr/sbin/ipvsadm
# rpm -Uvh cmdb-1.6.5-9.i386.rpm
# rpm -Uvh cluman-1.6.5h-5.i386.rpm
# rpm -Uvh ipvsadm-1.21-8.enc.i386.rpm
```

- Cluster IPAddress 입력

```
# vi /usr/clx/etc/clusterip
--------------------------------------------------------------------------

211.238.41.168    # cluster ip, Master Node 가 가지고 있는 관리 IP

--------------------------------------------------------------------------
```

- License Key 입력

```
# vi /usr/clx/etc/license.key
--------------------------------------------------------------------------

ENC16-XXXX-XXXX-XXXX-XXXX-XXXX  # 라이센트 입력

--------------------------------------------------------------------------
```

- 기본 시스템 설정 확인

```
# vi /usr/clx/sbin/cmctl
--------------------------------------------------------------------------
```

```
# Network interface device
CMDB_NETDEV=eth0

# Special Runing Mode
EXTRA='-e masq -e linux24'
```

CMDB_NETDEV=eth0 는 실제 클러스터 군이 서로 통신을 하는 Network Device 명을 적어 주어야 한다. 실제 클러스터 구축 시 Service Network 와 Management Network 로 분리 하는 경우가 많은데 이때는 실제 Encluster 클러스터 노드간의 통신을 진행 하는 장치명을 적어 주면 된다.

EXTRA='-e masq' 부분을 EXTRA='-e masq -e linux24' 으로 변경 해줌
-e linux24 는 Direct Routing 방식에서의 Real 서버에 lo alias device 에 VIP 를 설정하는 역활을 iptables 를 이용하여 대신 해주는 부분임
/usr/clx/sbin/vipctl 에 보면 -e linux24 설정이 되어져 있는 경우 아래와 같은 설정이 적용된다.

```
iptables -t nat -D PREROUTING -j REDIRECT -d $VIP -p tcp > /dev/null 2>&1
iptables -t nat -D PREROUTING -j REDIRECT -d $VIP -p udp > /dev/null 2>&1
iptables -t nat -A PREROUTING -j REDIRECT -d $VIP -p tcp
iptables -t nat -A PREROUTING -j REDIRECT -d $VIP -p udp
```

이 설정은 반드시 커널에서 iptables 에 nat 와 redirect 기능이 추가 되어야 한다.
이 부분의 내용이 정상적으로 적용 되지 않은 경우 VIP 를 통해 접속 할 경우 접속이 이루어 지지 않는다.

정상적인 동작을 확인 하는 방법은 Real Node 에서 아래 Command Line 을 입력하면 확인이 가능하다.

```
# iptables -L -t nat -n
```

- Encluster 데몬 시작

```
# /etc/rc.d/init.d/cmctl restart
```

정상적으로 시작이 이루어 지면 웹 브라우저를 열어 Encluster 1.6 Web UI인 Cluman 에 접속을 한다.

http://CIP:777

이후 클러스터 설정은 Cluman 에서 설정을 하면 된다.

- 설치 관련 팁

1) 라이센스 관련 팁
- 라이센스 정상 확인
/usr/clx/sbin/cmdb -V << License Key >>

- 라이센스 변경
/usr/clx/etc/license.key 에 변경된 라이센스 정보를 입력 후

/etc/rc.d/init.d/cmctl reloadlicense

2) Cluman 패스워드 분실 시

- Cluman (Web UI) 관리 패스워드 분실 시 패스워드 초기화

/usr/clx/sbin/initpasswd.sh
위를 실행하면 초기 패스워드인 root/root 로 변경이 된다.

/usr/clx/sbin/cmitool set svc sysinfo var0 ""
cp /usr/clx/cluman/bin/.htpasswd.default /usr/clx/cluman/bin/.htpasswd
위 같은 방법으로 해도 패스워드가 초기화 되어진다.

3) Cluman 접속 불가능 한 상태에서의 master node 확인하기

/usr/clx/sbin/cmitool get cluster self
---
entry_type: cluster
num_entry: 1

name: default

ip: 211.238.41.168

master_node_name: 211.238.41.167

master_node_ip: 211.238.41.167

run: up

status: up

num_node: 3

num_nset: 1

num_vnode: 1

num_svc: 4

update_seq: 69

config_seq: 46

3.1.3 Encluster 1.6 제거

- Encluster 1.6 제거

```
# rpm -e cmdb-x.x.i386.rpm
# rpm -e cluman-x.x.i386.rpm
# rpm -e ipvsadm-1.21.x.i386.rpm
# rm -rf /usr/clx/cluman
# rm -rf /usr/clx/log
```

/usr/clx/etc 밑에 있는 clusterip, license.key, nodelist, snapshot.cdb 등은
영구적인 삭제가 아닌 경우 재설치를 위해 반드시 놓아 둔다.

snapshot.cdb 는 cluman 의 설정 저장 파일로 snapshot.cdb 를 백업 해 두었다가
재 설치 및 업그레이드 이후 다시 /usr/clx/etc 밑에 놓고 /etc/rc.d/init.d/cmctl
을 restart 하면 이전의 설정을 그대로 유지 할 수 있다.

3.1.4 Cluman 을 이용한 Encluster 1.6 관리 하기

3.1.5 cmitool, cmish 을 이용한 command 방식 관리

3.1.6 Encluster 1.6 을 이용한 HA 구성

Encluster 의 원리 중 director election 상태에서 특정 스크립트를 실행하고
director election 해제 시에 특정 스크립트를 실행하는 원리가 있다.
이를 이용하면 간단한 HA 구성을 만들 수 있다.

apache 웹서버를 부하분산 하면서 Mysql DB 를 HA 시키는 구조의 시스템을
구축해 보도록 하자.

- svr1, svr2 서버에 apache 와 mysql 을 설치 한다. mysql 버전은 3.23 이상으로 한다.

- Encluster 로 svr1, svr2의 서버들은 DR 형태로 클러스터 구성을 한다.

- httpd 와 mysqld 서비스를 등록하다.

- mysql replication 설정하기

svr1 서버를 master 서버로 지정 함.
svr2 서버를 slaver 서버로 지정 함.

- master 서버에 DB 복제를 해주는 역활의 User 를 만든다.

mysql > GRANT FILE ON *.* TO cluster@"%" IDENTIFIED BY '<password>';

여기서 % 대신에 slave 서버 주소를 적어 주어도 상관없음.

master 의 /etc/my.cnf

```
---------------------------------------------------------------
[mysqld]
log-bin
binlog-do-db=syszone
server-id=1
---------------------------------------------------------------
binlog-do-db='DB 이름'
```

slaver 의 /etc/my.cnf

```
---------------------------------------------------------------------
[mysqld]
server-id=2
master-host=211.238.41.165
master-user=repli
master-password=root///
master-port=3306
---------------------------------------------------------------------
```

master-user=repli 가 잘 안되면 그냥 root 로 한다.
단..root 역시 원격 DB 접속을 가능토록 해주어야 한다.

```
mysql> select User,Host from user;
+---------+-----------+
| User    | Host      |
+---------+-----------+
| root    | %         |
| www     | %         |
|         | localhost |
| root    | localhost |
| sahak21 | localhost |
| www     | localhost |
| www     | svr2      |
+---------+-----------+
```

svr1, svr2 서버의 mysql deamon 을 차례로 start 시킨다.

확인 절차 --master server (svr1)

```
mysql > show master status;
+--------------+----------+--------------+------------------+
| File         | Position | Binlog_do_db | Binlog_ignore_db |
+--------------+----------+--------------+------------------+
| svr1-bin.023 | 79       | syszone      |                  |
+--------------+----------+--------------+------------------+
```

확인 절차 --slave server (svr2)

Master_Host	211.238.41.165
Master_User	root
Master_Port	3306
Connect_retry	60
Master_Log_File	svr1-bin.023
Read_Master_Log_Pos	79
Relay_Log_File	svr2-relay-bin.025
Relay_Log_Pos	271
Relay_Master_Log_File	svr1-bin.023
Slave_IO_Running	Yes
Slave_SQL_Running	Yes
Replicate_do_db	
Replicate_ignore_db	
Last_errno	0
Last_error	
Skip_counter	0
Exec_master_log_pos	79
Relay_log_space	271

여기서 master pos 에 관련된것이 master status 의 position 값과 일치해야함.

- Encluster 상에서 Apache Virtual Host 와 Mysql DB host 지정 문제를 해결해야 한다.

- apache virtualhost 문제 해결

실제 apache 에서 virtualhost 를 사용할 경우 현 서버 구성중 자기 자신이 direct 로 되어져 있는 경우 httpd.conf 의 서버 주소 및 virtualhost 주소가 모두 vip 로 되어져야 virtualhost 가 되어진다. 그리고 real server 인 경우에는 httpd.conf 의 Ip address 가 모두 real ip로 되어져 있어야 virtualhost 가 이루어 진다.

/usr/local/apache/conf 디렉토리에 httpd.conf-real , httpd.conf-vip 의 두개의 설정파일을 만들어 둔다.

- Mysql connect address 문제 해결

Encluster 의 클러스터 구성에서 web 과 같은 서비스와 Mysql DB 서비스를 같이 할 경우 mysql db connection address 가 vip 로 되어져야 하는데 web의 경우는 vip 를 인식하는 클라이언트가 클러스터 구성의 외부에 있는 client brower 에서 찾아 들어가기 때문에 vip를 통한 부하분산에 문제가 없다.
하지만 mysql 처럼 DB 서버의 경우에는 db client 가 클러스터 구성 외부의 client 가 아닌 web server가 db client 역할을 하고 그 web server 가 db server와 같은 서버일 경우 client 가 클러스터 구성 안에 존재 하게 된다.
이때 클러스터 구성안에서는 vip 를 direct server 로 인식하기 때문에 부하분산 방식에 문제가 생긴다. 그래서 이를 해결 하기 위해서는 direct 서버에 별도의 db connection 전용의 ip address 를 활당하는 방법으로 이를 해결한다.

- Encluster 구성중 /usr/clx/sbin/vipctl 의 script 를 이용하여 위의 두가지 문제를 해결한다.

vipctl 의 역활은 direct 가 변경되는 경우 이 스크립터를 실행한다.

////// master (svr1) 의 /usr/clx/sbin/vipctl script 수정

script fuction 중 honor_vip () 는 direct 변동으로 인해 자기가 director 가 안 되었을 경우 실행하는 함수이다.

```
honor_vip () {
  if ifconfig $DEV:$ALIAS | grep "inet addr:$VIP" &> /dev/null
  then
    # The VIP is already taken
    exit 0;
  fi

  if [ -e /proc/sys/net/ipv4/conf/all/hidden ]
  then
    # Linux-2.2.x with hidden patch
    # Do ARP disabling on the interface
    # This line may be moved to system init script
    echo 1 > /proc/sys/net/ipv4/conf/all/hidden
    echo 1 > /proc/sys/net/ipv4/conf/lo/hidden
    # Add interface alias on the vip
    ifconfig lo:$ALIAS $VIP netmask 255.255.255.255 broadcast $VIP up
  elif [ -e $IPTABLES ]
```

```
    then
      # Linux-2.4.x with iptables
      $IPTABLES -t nat -D PREROUTING -j REDIRECT -d $VIP -p tcp > /dev/null 2>&1
      $IPTABLES -t nat -D PREROUTING -j REDIRECT -d $VIP -p udp > /dev/null 2>&1
      $IPTABLES -t nat -A PREROUTING -j REDIRECT -d $VIP -p tcp
      $IPTABLES -t nat -A PREROUTING -j REDIRECT -d $VIP -p udp

      이부분 추가
-----------------------------------------------------------------------
      ifconfig $DEV:777 down
      /usr/local/apache/bin/apachectl stop
      ln -sf /usr/local/apache/conf/httpd.conf-real /usr/local/apache/conf/httpd.conf
     /usr/local/apache/bin/apachectl start
-----------------------------------------------------------------------

    fi
}
```

take_vip () 함수는 자기가 director 가 되었을 경우 실행하는 함수이다.

```
take_vip () {
        # 디렉터가 되는 경우

    # Remove vip honoring
    dishonor_vip

    # Add interface alias on the vip
    ifconfig $DEV:$ALIAS $VIP netmask 255.255.255.255 broadcast $VIP up &> /dev/null
    route add -host $VIP dev $DEV:$ALIAS &> /dev/null

        이부분 추가
-----------------------------------------------------------------------
    ifconfig $DEV:777 211.238.41.174 netmask 255.255.255.224 up &> /dev/null
    /usr/local/apache/bin/apachectl stop
    ln -sf /usr/local/apache/conf/httpd.conf-vip /usr/local/apache/conf/httpd.conf
    /usr/local/apache/bin/apachectl start
-----------------------------------------------------------------------
```

```
    # Do ARP spoofing; do several times
    if [ $DOARPSPOOFING = true ]
    then
      HWADDR=`ifconfig $DEV | grep HWaddr | cut -d' ' -f11`
      BADDR='ff:ff:ff:ff:ff:ff'
      ( ₩
      $SENDARP $DEV $VIP $HWADDR $VIP $BADDR &> /dev/null; usleep 400; ₩
      $SENDARP $DEV $VIP $HWADDR $VIP $BADDR &> /dev/null; usleep 400; ₩
      $SENDARP $DEV $VIP $HWADDR $VIP $BADDR &> /dev/null; ₩
      ) &
    fi
}
```

////// slave (svr2) 의 /usr/clx/sbin/vipctl 수정

honor_vip () 의 수정 부분에 아래 내용을 추가

```
ifconfig $DEV:777 down
/usr/local/apache/bin/apachectl stop
ln -sf /usr/local/apache/conf/httpd.conf-real /usr/local/apache/conf/httpd.conf
/usr/local/apache/bin/apachectl start
/usr/clx/sbin/dush -n svr1 /etc/rc.d/init.d/mysql stop
/etc/rc.d/init.d/mysql stop
/usr/clx/sbin/dua /usr/local/mysql/data/syszone
/usr/clx/sbin/dush -n svr1 /etc/rc.d/init.d/mysql start
/etc/rc.d/init.d/mysql start
```

take_vip () 의 수정 부부에 아래 내용을 추가

```
ifconfig $DEV:777 211.238.41.174 netmask 255.255.255.224 up &> /dev/null
/usr/local/apache/bin/apachectl stop
ln -sf /usr/local/apache/conf/httpd.conf-vip /usr/local/apache/conf/httpd.conf
/usr/local/apache/bin/apachectl start
```

이는 웹 서버는 Encluster 를 이용하여 정상적으로 부하분산을 시키고 Mysql 의 경우는

replication 을 이용하여 평소에는 master 서버의 내용을 slave 에서 실시간 복제를 하고 있다가 master 서버에 문제가 발생했을 경우 slave 서버가 director 가 되면서 mysql connection address 를 slave 가 가져가면서 slave 에서 db service를 하게 된다.

그런 이후 master 가 살아나면 slave db server 새로 갱신된 내용을 master 서버로 file dump 방식으로 다시 전달하고 master 서버에서 db connection ip address 를 가져 오게 되어 master 서버에서 다시 서비스를 하게 된다.

slave 서버에서 master 서버로 다시 db service server가 변경되는 액션은 관리자가 수동으로 처리하게 하고 있다. 자동으로 할 경우엔 slave server 의 honor_vip () 함수의 추가 내용 마지막에 /etc/rc.d/init.d/cmctl restart 를 추가해 주면 된다.

;;; 추가 설정 시 주의 사항

rsh, rlogin 을 이용한 dua, dush util 를 설치하고 설정해야 함.
direct 가중치를 master 노드에 200 , slave 노드에 100 을 주도록 한다.

3.1.7 Encluster 1.6 의 작동 원리

3.2. Encluster 2.0을 이용한 리눅스 부하분산 클러스터 구축

3.2.1 Encluster 2.0 구성 및 기능 소개

3.2.2 Encluster 2.0 설치

ftp://clunix@technet.clunix.com/EnCluster 에서 최신 버전을 다운 받는다.

- IPVS Kernel 및 ipvsadm 설치
ipvs 가 패치, 적용된 rpm kernel 를 설치 한다.
ipvsadm 역시 rpm 으로 설치 한다. Source 설치 시에는 반드시 /usr/sbin 밑에 링크를 건다.

ln -s /sbin/ipvsadm /usr/sbin/ipvsadm

- RPM 설치

rpm -ivh ecm-base-1.0.0-1.i386.rpm
rpm -ivh ecm-mon-1.0.0-1.i386.rpm
rpm -ivh ecm-ui-1.0.0-1.i386.rpm
rpm -ivh ecm-l4-1.0.0-1.i386.rpm
rpm -ivh ecm-mon-1.0.0-1.i386.rpm
rpm -ivh ecm-tsp-1.0.0-1.i386.rpm

ecm-base RPM을 가장 먼저 설치해야 함

- Encluster 2.0 기본 설정

vi /usr/clx/etc/econf

---

cl=(name=clunix-cluster dsc='cluster test cluster' ip=<ip>192.168.0.101)

---

name, dsc는 http://192.168.0.101:910/ecx/에 접속, 왼쪽 메뉴의 Cluster를 선택하였을 때 나오는 이름, 설명에 해당하는 값임

ip는 CIP(Cluster IP)로서 같은 Cluster에서는 이 값이 동일해야만 서로 통신이 가능하며, UI 접속 주소이기도 함.

vi /usr/clx/etc/license.key

- license.key 파일이 없거나 잘못된 키이면, 실행되지 않음

vi /usr/clx/bin/ecmctl
- ECM channel로서 어떤 network interface를 사용할 것인지 결정 (default: eth0)
- 파일 앞부분에서 아래라인 수정
ECM_NET_DEV=${ECM_NET_DEV:=eth0}

/usr/clx/rc/snapshot 파일이 존재하면, 기존의 설정을 읽게 됨

- 깔끔하게 시작하고 싶은 경우, snapshot 파일을 삭제

/usr/clx/bin/ecmctl start
tail -f /usr/clx/log/edbd.log

edbd log를 주시하면서, wake up slept services 라는 메시지가 나오면 ready to serve상태임

http://CIP:910/ecx로 접속

3.2.3 Encluster 2.0 제거

- 기존 install 제거 (ecm-base를 가장 나중에 제거)
rpm -e ecm-ui (HPC: # rpm -e ecmhpc-ui)
rpm -e ecm-mon
rpm -e ecm-base
rm -rf /usr/clx

* /usr/clx/etc/license.key나 /usr/clx/rc/snapshot 백업할 필요있으면 백업
* /usr/clx를 지우면 dutil과 nodelist 파일도 같이 삭제됨
* 따라서, dutil rpm도 제거하고, 다시 설치하거나,

/usr/clx/sbin과 /usr/clx/etc/nodelist, /usr/clx/etc/nosynclist 만을 남기고
/usr/clx를 삭제하는 것을 권장

3.2.4 Web UI (ecm-ui)를 이용한 Encluster-2.0 관리 하기

3.2.5 Edb-tool 사용하기

3.2.6 Encluster-2.0 작동 원리

4. Data 동기화 (Data Sync or Data Mirroring)

4.1. rsync 를 이용한 data sync

클러스터에서 디스크를 동기화하는 방법은 많이 소개가 되어져 있다.
예전에는 거의 NFS 를 이용해 왔었는데 보안과 시스템 로드에서 몇가지 문제점이 있어서 현재는 많이 사용하진 않는다. 현재에 대두되는 걸로는 rsync,GFS, pvfs, Soft RAID, Intemezzo 등이 있다. 여기서는 가장 간단하면서 경제적이고 부하도 크게 문제되지 않는 rsync 에 대해서 알아보도록 하겠다.

rsync 는 NT 의 미러링과 같이 특정 하드디스크의 데이타를 그 속성을 유지한체 다른 디스크로 동일하게 복제해주는 역활을 하는 프로그램이다.

rsync 로 이용할수 있는것은 아주 다양한데..주로 클러스터 웹서버의 디스크 동기화, 미러링 서버의 디스크 동기화, 백업서버의 데이터 백업 등이 있다.

rsync는 rcp와 비슷한 동작을 하는 프로그램으로 rcp보다 더 다양한 옵션이 있고, 더 효율적으로 데이터를 전송합니다. (출발지와 목적지 사이에다른 부분만을 전송)
파일크기의 변화나 시간의 변화등을 이용 동기화를합니다.
주요 특징은 다음과 같습니다.

- 링크, device, 소유자, 그룹, 허가권 복사 지원
- GNU tar와 비슷한 exclude, exclude-from 옵션 지원
- rsh 또는 ssh 등 사용가능
- root 권한이 필요없음
- anonymous 또는 인증 rsync 서버 지원(미러링에 유용함)

이제 설치로 넘어가 보도록 하자..

4.1.2. rsync 설치

rsync 는 Source 를 이용하여 설치하는 방법과 RPM을 이용하여 설치 하는 방법이 있다.

http://rsync.samba.org/ftp/rsync/ 에서 최신판을 받아 설치 하면된다.

rpm -Uvh rsync-2.5.7-0.9.i386.rpm 혹은 ..

```
# tar xzvf rsync-2.5.7.tar.gz
# ./configure
# make
# make install
```

4.1.3 rsync 설정 방법

rsync 가 사용하는 프로토콜로는 rsh 나 ssh를 사용하거나 특정 포트를 이용하여 xinetd 데몬으로 제어도 가능하다. 보통 873 포트를 사용한다.
ssh 를 이용한 사용방법과 873 port 를 이용한 사용방법은 다소 차이가 있다.

873 port 사용방법
rsync 를 사용할 리눅스 버젼이 6.x 일경우에는 /etc/inetd.conf 에 다음줄을 추가한다.

```
rsync stream tcp nowait root /usr/bin/rsync rsyncd --daemon
```

만일 리눅스 버젼이 Redhat 7.x ~ 9 이면...

/etc/xinetd.d/rsync 파일을 만들어 준다.

```
---------------------------------------------------------------------
service rsync
{
disable = no
socket_type = stream
wait = no
user = root
server = /usr/bin/rsync
server_args = --daemon
log_on_failure += USERID
}
---------------------------------------------------------------------
```

그리고 /etc/services 에 다음 내용이 있는지 확인하고 없으면 추가한다.

```
rsync           873/tcp                 # rsync
rsync           873/udp                 # rsync
```

그런후 inetd 혹은 xinetd 데몬을 restart 해준다.

이제 /etc/rsyncd.conf 설정 파일을 만들어 준다.

/etc/rsyncd.conf ----------

[webdata]
path = /usr/local/apache/htdocs
comment = web data
uid = root
gid = root
use chroot = yes
read only = yes
hosts allow = 211.238.41.164, 211.238.41.165
max connections = 3
timeout 600

[home] : 서비스명
path : 전송될 data가 들어 있는 경로
uid : data 를 전송하는 사용자의 uid 기본값은 nobody
gid : data 를 전송하는 사용자의 gid 기본값은 nobody
use chroot : path를 root 디렉토리로 사용. 보안상 필요함.
read only : 읽기전용
(클라이언트에서 서버로 올리는 경우에는 read only= no 로 설정을 해야됩니다.)

hosts allow : 호스트별 접속허용. 기본값은 all host이므로 보안을 유지하
려면 반드시 설정함
max connections : 동시접속자수.

timeout : 클라이언트에서 접근시 타임아웃시간.
anonymous 로 운영하는 경우 설정을 해야 클라이언트가 죽었을 때 서버에서 접속을
해체할 수 있습니다.

이렇게해서 873포트를 쓰는 rsync서버의 설정은 끝났습니다.

4.1.4 rsync 클라이언트 사용방법

이제 rsync 클라이언트에서 data 를 동기화 하는 방법을 알아보자.
Mater_LB(www3) 서버를 데이터 관리 서버라고 하자. 즉 Master_LB 서버에 실제 서비스 데이터를 올려 두고 Webserver-1(www1), Webserver-2(www2) 에서 rsync 를 통해 자료를 동기화시키는 구성으로 셋팅을 해보자.

www1, www2 서버에서 아래 명령을 수행한다.

```
# rsync -avz --delete www3::webdata   /usr/local/apache/htdocs
```

ssh 나 rsh 를 이용한 방법은 rsyncd 를 설치하고 xinetd 설정이나 rsyncd.conf 같은 설정은 하지 않아도 된다. 그냥..

```
# rsync -avz -e ssh www3:/usr/local/apache/htdocs   /usr/local/apache/htdocs
```

rsh 를 사용하면.. -e rsh 하면 된다.

ssh 를 이용하면 패스워드를 입력해야 하는데 스크립터를 디스크 동기화를 자동화 할때는 다소 불편한 점이 있다. 이때는 --password-file 옵션을 사용하면 암호파일의 위치를 지정해 줄수 있다.

이와 같이 rsync 클라이언트 사용법을 cron 등에 등록시켜 주기적으로 data 를 업데이트 시킴으로써...두 호스트 간의 데이타 동기화를 할수 있다.

4.1.5 실제 적용 예제

[Maste LB 의 /etc/rsyncd.conf 설정]

/etc/rsyncd.conf ----------

```
[webdata]
path = /usr/local/apache/htdocs
comment = web data
uid = root
gid = root
use chroot = yes
read only = yes
hosts allow = 211.238.41.165, 211.238.41.166
max connections = 3
```

```
timeout 600

[dbdata]
path = /usr/local/mysql/data
comment = mysql
uid = root
gid = root
use chroot = yes
read only = yes
hosts allow = 211.238.41.165, 211.238.41.166
max connections = 3
timeout 600
```

[WebServer-1, WebServer-2 의 자동 Sync Script]

/root/bin/rsync.sh ----------

```
rsync -avz --delete www3::webdata /usr/local/apache/htdocs
rsync -avz --delete www3::dbdata /usr/local/mysql/data
```

```
# chmod 755 rsync.sh
```

이 스크립터 파일에 실행 권한을 주고 실행하면 된다.
그런뒤...cron 에 5분 간격으로 위의 스크립터가 실행되도록 한다.

```
# crontab -e
```

```
*/5 * * * * /root/bin/rsync.sh
```

```
# /etc/rc.d/init.d/crond restart
```

이것으로 Rsync 를 이용한 데이터 동기화 내용을 마치도록 하겠다.

4.2. dutils 를 이용한 data sync

4.2.1 dua, dush 은 무엇 인가 ?

dua, dush 는 Encluster-HPC 의 다중 서버 관리 도구이다. 즉 여러대의 서버에 file 및 작업 명령을 일괄적으로 처리하도록 해주는 프로그램이다.
dua, dush 를 정상적으로 사용하기 위해서는 앞에서 설명한 rsh, rlogin 서비스 설정이 완료되어 있어야 한다.

4.2.2 rsh, rlogin 설정 하기

rlogin, rsh 관련 서비스를 구동하기 위해서는 rsh, rsh-server 두개의 패키지가 설치가 되어져 있어야 한다.

먼저 xinetd 데몬에서 rsh, rlogin 서비스를 사용할 수 있도록 설정을 변경한다.

```
# vi /etc/xinetd.d/rsh
--------------------------------------------------------------------------
service shell
{
        disable = no
        socket_type             = stream
        wait                    = no
        user                    = root
        log_on_success          += USERID
        log_on_failure          += USERID
        server                  = /usr/sbin/in.rshd
}
--------------------------------------------------------------------------

# vi /etc/xinetd.d/rlogin
--------------------------------------------------------------------------
```

```
service login
{
        disable = no
        socket_type             = stream
        wait                    = no
        user                    = root
        log_on_success          += USERID
        log_on_failure          += USERID
        server                  = /usr/sbin/in.rlogind
}
```

설정을 변경한 후 xinetd 데몬을 restart 한다.

```
# /etc/rc.d/init.d/xinetd restart

# vi /etc/securetty
```

```
.
..
...
rsh
rlogin
```

---

rsh, rlogin 서비스를 허용할 hosts 와 users 설정을 한다.

```
# vi /etc/hosts
```

```
127.0.0.1                       localhost

# Data Sync 가 필요한 nodelist
211.238.41.165                  www1
211.238.41.166                  www2
211.238.41.167                  www3
```

```
# vi /etc/hosts.equiv
```

```
---------------------------------------------------------------------------
www1
www2
www3
.
---------------------------------------------------------------------------

# vi $HOME/.rhosts
---------------------------------------------------------------------------
www1
www2
www3
---------------------------------------------------------------------------
```

.rhosts 설정에서는 각 노드간의 rsh 접속이 동일한 계정 사이에서만 기본적으로 이루어 진다. 만일 clunix 계정으로 root 의 권한으로 rsh 접속을 하기 위해서는 root 의 .rhosts 에 아래와 같이 해 주어야 한다.

```
# vi /root/.rhosts
---------------------------------------------------------------------------
www1
www2
www3
www1            clunix
www2            clunix
www3            clunix
---------------------------------------------------------------------------
```

위 설정에서는 root 계정에서는 www node 간에 rsh, rlogin 으로 별도의 옵션이나 인증없이 login 이 된다.

clunix 계정에서는 -l 옵션으로 www node 간에 rsh, rlogin 으로 root login 이 허용된다.

```
www1 root # rsh www2
www2 root #

www1 clunix $ rsh -l root www2
www2 root #
```

4.2.3 dutils 설치 및 기본 설정

dutils 는 dua, dush 로 구성 되어진 Encluster 의 다중 서버 관리 도구이다.
즉 여러대의 서버에 file 및 작업 명령을 일괄적으로 처리하도록 해주는 프로그램이다.
dua, dush 를 정상적으로 사용하기 위해서는 앞에서 설명한 rsh, rlogin 서비스 설정
이 완료되어 있어야 한다.

먼저 Master Node 에 dutil-1.2.1-1.noarch.rpm 를 설치 한다.

```
# rpm -Uvh dutil-1.2.1-1.noarch.rpm
```

실제 dua, dush 를 이용하여 일괄 괄리할 서버 리스트를 작성한다.

```
# vi /usr/clx/etc/nodelist
--------------------------------------------------------------------------
www1
www2
www3
--------------------------------------------------------------------------
```

** dua : file 을 nodelist 에 포함된 모든 Node 에게 일괄적으로 sync 시키는 도구
** dush : nodelist 의 포함된 node 에게 일괄적으로 내려진 작업을 수행한다.

공통 옵션 설명 :

-l : /usr/clx/etc/nodelist 이외의 다른 nodelist 를 참조할 경우 -l 옵션으로 호출
할수 있다.

예) dua -l /etc/nodelist /root

-n : nodelist 에 포함된 모든 node 가 아닌 특정 node 에만 작업을 수행할 경우 사용됨

예) dua -n www3 /root

-p : 작업을 순차적으로 처리하는 것이 아닌 backgroud mode 로 동시에 작업이 수행된다

예) dush -p "/etc/rc.d/init.d/ecmctl restart"

-s : 노드간 수행 결과를 구분하기 위한 ###executing..." 이란 메세지가 나타나지 않고 수행 결과만을 출력해 주는 옵션이다.

-d : dua 에서 사용되는 옵션으로 디렉토리간의 완전한 sync 를 할때 사용된다. 즉 www1 와 www2 의 디렉토리 내용을 완전히 sync 시킬때 사용되는 옵션으로 그냥 # dua /root 라 수행하면 www1 의 /root 내용이 www2 의 /root 로 파일이 복제 되지만 # dua -d /root 라 수행하면 www2가 www1 보다 더 많은 파일을 가지고 있다면 www1 에 없는 파일은 모두 삭제해 버리고 완전 www1 와 동일한 자료만을 가지게 된다.

4.2.4 dua 를 이용한 Data sync

이제 dua 를 이용한 웹 서비스 데이터 컨덴츠를 동기화 해 보도록 하자.

/home/clunix/www 밑에 html 및 php,jsp 와 같은 웹 소스 파일이 위치 한다면 www1 을 file master 서버로 정해 두고 www1 에서만 파일 업로드를 시키도록 정책을 정한다. 그런 후

www1 에서

dua /home/clunix/www

라고 정해 주시면 /home/clunix/www 밑의 모든 파일이 www2, www3 로 sync 가 되어진다.

만일 www1, www2, www3 에서 모두 file upload 가 이루어 진다면 ..

www1 에 아래와 같은 스크립트를 하나 만든다.

vi /root/bin/datasync

```
---------------------------------------------------------------------------
#!/bin/sh
```

```
DUTILS=/usr/clx/sbin

$DUTILS/dua -n www2 /home/clunix/www/
$DUTILS/dush -n www2 "$DUTILS/dua -n www3 /home/clunix/www/"
$DUTILS/dush -n www3 "$DUTILS/dua -n www1 /home/clunix/www/"
$DUTILS/dush -n www3 "$DUTILS/dua -n www2 /home/clunix/www/"
---------------------------------------------------------------------------
```

위 명령어를 www1 에서 실행하면 www1 의 data 가 www2 로 전달 되고
www2 의 데이터가 www3 로 전달 되고 www3 의 data 가 www1, www2 로 전달되어
3개의 노드의 파일이 모두 동일해 진다.

단 위 구성에서는 www1 서버에 이상이 생기면 파일 동기화에 문제가 발생 할수
있다는 단점이 있다.

dua 는 관리자의 편위를 위한 data sync tool 이다. 모든 active 서버에 데이터
를 multi active 상태로 sync 를 시키기 위해서는 아래의 Intermezzo 나 Pvfs 를
사용해야 한다.

요즘에는 Open GFS File System 이 주요 이슈로 떠 오르고 있다.

4.2.5 dush를 이용한 일괄 관리

dush 를 이용하여 3개의 서버에 일괄적으로 시스템을 관리 할 수 있다.
예를 들면 시스템의 프로그램 일괄 설치 및 시스템 상의 명령어 일괄 수행등에
활용 될수 있다.

프로그램 일괄 설치

- dua 로 프로그램 일괄 배포

```
[root@test01 root]# dua check-utils-2.0.1.rpm
### synchronizing test01
building file list ... done
wrote 71 bytes  read 20 bytes  182.00 bytes/sec
total size is 109608  speedup is 1204.48
### synchronizing test02
building file list ... done
```

check-utils-2.0.1.rpm

wrote 54927 bytes read 36 bytes 109926.00 bytes/sec

total size is 109608 speedup is 1.99

- dush 로 프로그램 일괄 설치

[root@test01 root]# dush rpm -Uvh check-utils-2.0.1.rpm

- dush 로 프로그램 일괄 수행

[root@test01 root]# dush /usr/local/bin/chkbandwidth

executing in test01

[test01] [I N] 163.52 Kb/s [OUT] 306.26 Kb/s

executing in test02

[test02] [I N] 45.66 Kb/s [OUT] 1.18 Kb/s

위와 같이 dush 를 잘 이용하면 여러대의 서버를 손쉽게 관리 할수 있다.

4.3. NFS + Automount 를 이용한 data sync

여러대의 서버가 같은 data 로 서비스를 할 경우 Sync 방식 외에 Data Shared 방식으로 서비스를 진행 할수도 있다. 가장 대표적인 서비스가 NFS 이다. NFS 의 경우 시스템 부하, 보안, NFS Lock 같은 문제로 많은 유저가 사용을 주저 하고 있지만 사이트의 시스템 구성과 서비스 방식에 따라 NFS 서비스는 클러스터 시스템구성에서 가장 적절한 파일 시스템 환경을 제공할 수도 있다.

- nfs 설정

Nfs Server 가 되는 File Server 의 exports 설정 파일을 아래와 같이 설정한다.

```
# vi /etc/exports
--------------------------------------------------------------------------
# NFS 원격 설치 시 사용되는 경로
/data/clunix/www      *(rw,no_root_squash)
--------------------------------------------------------------------------
```

```
# /etc/rc.d/init.d/portmap restart
# /etc/rc.d/init.d/nfs restart
```

이것을 각 부하분산 시스템에 적절한 Nfs 설정이 완료된다.

```
# mount -t nfs file:/data/clunix/www    /home/clunix/www
```

```
[root@www1 root] df
Filesystem            1K-blocks      Used Available Use% Mounted on
/dev/hda3             10080520    148852   9419600   2% /
/dev/hda1               202220     14518    177262   8% /boot
none                    514940         0    514940   0% /dev/shm
/dev/hda2             10080520   1547244   8021208  17% /usr
/dev/hda6              2016016    200712   1712892  11% /var
file:/data/clunix/www 234429320     32840 234396480   1% /home/clunix/www
```

- automount 를 이용한 NFS 서비스 구동하기

automount 는 nfs 서비스와 같이 활용되어 nfs lock 이나 nfs 로 인한 connection 을 효율적으로 관리해 주는 프로그램이다.

nfs 는 항상 node 간의 네트워크 mount 상태가 유지 되는데 automount 는 nfs 의 사용이 없을 경우에는 자동으로 mount 를 해제하고, 사용자가 nfs 해당 경로에 접근을 하면 자동으로 mount 를 시켜 주는 역활을 한다. 그러다 특정 시간 동안 접속이 없으면 다시 Mount 를 해제하게 된다. 이로써 보안과 성능, Lock 같은 문제를 어느 정도 해결 할수 있다.

automount 에는 am-utils (amd), autofs 두가지의 패키지가 있다.
Redhat9 에서는 기본적으로 autofs 를 체택하고 있다.

간단한 설정에 대해 알아보자.

주요 설정 파일은 ..

/etc/auto.master
/etc/auto.misc

```
# vi /etc/auto.master
```

```
==========================================================================
/home           /etc/auto.clunix     --timeout=5


# vi /etc/auto.clunix
==========================================================================
clunix/www          -fstype=nfs,rw,soft,bg      file:/data/clunix/www

# /etc/rc.d/init.d/portmap restart
# /etc/rc.d/init.d/autofs restart
```

/home/clunix/www 로 이동하면 자동으로 file 의 /data/clunix/www
로 nfs mount 가 된다.

automount 를 사용하게 되면 실제 사용자가 지정된 mount point 에 접근 시에만 자동으로 nfs mount 하고 일정 시간 사용을 하지 않으면 자동으로 umount 를 시키기 때문에 nfs 서비스로 인한 resource 를 최대한 절약 할 수 있고, 여러명이 지속저으로 Nfs 를 사용할 경우 1개의 곳에서 lock 이 결려도 전체 서비스에 문제가 생길 가능성이 있는데 automount 로 상당 부분 해소할 수 있다.

autofs-4.x 버전에서는 multiple hostname 기능이 지원한다.
이는 먼저 master 로 Connection 요청을 해서 정상적인 요청이 이루어 지지 않으면 slave 에 연결이 된다. Redhat9 에서는 기본적으로 autofs-3.x 임으로 autofs-4.x 로 업그레이드를 해야 한다.

http://www.kernel.org/pub/linux/daemons/autofs/

4.4. Intermezzo 를 이용한 data sync

4.5. Pvfs 를 이용한 data sync

4.5.1 pvfs (Parallel Virtual File System) 소개

4.5.2 pvfs 설치

- Source 다운 로드

ftp://ftp.parl.clemson.edu/pub/pvfs/

pvfs-1.6.2.tgz

pvfs-kernel-1.6.2-linux-2.4.tgz

- 네트워크 설정 준비 환경

Pvfs Meta Server : www3.clunix.org

I/O Server : www1.clunix.org, www2.clunix.org

File Clients : www1.clunix.org, www2.clunix.org

PVFS 시스템 구성은 크게 3가지로 나누어 진다. 파일의 변경 정보가 저장되는 Meta 서버와 각 파일의 inode 정보들이 생성되는 I/O 서버, 그리고 파일 시스템 동기화가 되어지는 클라이언트 노드로 구성되어진다. 테스트 환경에서 각 성격 별 서버가 결정 나면 모든 구성 노드들의 /etc/hosts 에 반드시 해당 서버들의 hosts 정보를 입력해야 한다.

vi /etc/hosts

211.238.41.165 www1.clunix.org www1

211.238.41.166 www2.clunix.org www2

211.238.41.167 www3.clunix.org www3

vi /etc/sysconfig/network

HOSTNAME=xxx.clunix.org ->(각 서버에 해당하는 호스트네임)

- PVFS Source 설치

mv pvfs-1.6.2.tgz pvfs-kernel-1.6.2-linux-2.4.tgz /usr/local/src

cd /usr/local/src

tar xzvf pvfs-1.6.2.tgz

tar xzvf pvfs-kernel-1.6.2-linux-2.4.tgz

cd pvfs-1.6.2

./configure && make && make install

cd ../pvfs-kernel-1.6.2-linux

./configure --with-libpvfs-dir=../pvfs-1.6.2/lib

make && make install

- Meta Server (www3) 설정

```
# mkdir /pvfs-meta
# cd /pvfs-meta
# /usr/local/bin/mkmgrconf
---------------------------------------------------------------------

This script will make the .iodtab and .pvfsdir files
in the metadata directory of a PVFS file system.

Enter the root directory:
/pvfs-meta   //metadata를 저장할 디렉토리 이름 (바로 위에서 만든 디렉토리이다.)
Enter the user id of directory:
root
Enter the group id of directory:
root
Enter the mode of the root directory:
777
Enter the hostname that will run the manager:
www3.clunix.org   (mgr을 실행시킬 서버, 메타서버의 호스트이름을 쓰면 된다.)
Searching for host...success
Enter the port number on the host for manager:
(Port number 3000 is the default)
3000
Enter the I/O nodes: ( can use form node1, node2, ...or nodename#-#,#,#)
www1.clunix.org, www2.clunix.org -> i/o 서버에 해당하는 서버의 호스트 네임을 적는다.

Searching for hosts...success
I/O nodes : www1.clunix.org, www2.clunix.org
Enter the port number for the iods:
(Port number 7000 is the default)
7000
Done!
---------------------------------------------------------------------

# mgr     ->    meta server daemon running
# ps aux | grep mgr -> 프로세스 확인

- I/O Server 설정

# mkdir /pvfs-data
```

```
# chmod 700 /pvfs-data
# chown -R nobody.nobody /pvfs-data
# cp /usr/local/src/pvfs-1.6.2/system/iod.conf /etc/iod.conf
# iod    ->   I/O server daemon running
# ps aux | grep iod   -> 프로세스 확인

// pvfs-data 디렉토리의 소유권이 nobody 가 아닌 경우 iod 데몬이 실행 되지 않는다.

- 클라이언트 설정

# mkdir /lib/modules/2.4.27/misc
# cp /usr/local/src/pvfs-kenel-1.6.2/pvfs.o /lib/modules/2.4.27/misc
# mkdir /mnt/pvfs
# vi /etc/pvfstab
-----------------------------------------------------------------
www3.clunix.org:/pvfs-meta /mnt/pvfs pvfs port=3000 0 0

# pvfsd   -> pvfsd client daemon running
# mknod /dev/pvfsd c 60 0
# insmod pvfs
# mount.pvfs www3.clunix.org:/pvfs-meta /mnt/pvfs pvfs port=3000 0 0
# df -h      ->   mount 확인

// 이것으로 PVFS 설치 및 설정이 완료되었다.
// I/O 서버와 동시에 클라이언트 노드를 사용할 경우 각 해당 설정을 모두 다 해주면
// 된다.  실제 로컬 파일 시스템을 사용하는 성능과 비교 했을때 단일 파일의 크기에는
// 크게 상관이 없지만, 파일 수에 대해서는 상당한 영향을 받는 것을 확인했다.
// 즉 500M 파일 하나를 복사 하는 속도는 로컬 시스템과 PVFS 차이는 없지만
// 100개 정도의 파일 ( 총 크기 10M )을 복사하는데는 500M 1개 파일 복사하는데 비해
// 8배 정도 더 속도가 느리게 나왔다. 이런 기능 상의 특성을 이용해서 적절한 곳에
// 사용해야 할것이다.
```

5. Benchmark Tool로 시스템 성능 체크 하기

지금까지 LVS 기능을 이용하여 고성능 웹서버를 구축하는 방법에 대해 알아 보았다.
이장에서는 이렇게 구축한 시스템이 어느 정도의 성능을 나타낼수 있는지를 확인하는
방법에 대해 알아보도록 하겠다.

5.1. 웹서버 벤치마크 개론

웹서버 벤치마크는 웹서버처리과정을 자동으로 프로그램한 툴을 통해 웹서버의 처리율과 응답시간을 측정하는 것으로 단일 서버의 최대 처리 용량을 산정하여 전체 서비스에 필요한 자원 산정을 하는 기준을 제시하기 위해서다.

뿐만 아니라 초기의 벤치마크 자료는 서비스 이후 발생하는 병목에 대해 문제 원인 및 S/W 및 H/W 의 업그레이드등의 튜닝의 객관적 기준을 제시해 주는 역할을 한다.

그러므로 반드시 대형 사이트 구축 시에는 초기 구축 이후 구축 시스템의 객관적 성능을 체크해 두어야 한다.

- 웹 벤치마크를 위해 기본적인 HTTP 메커니즘

클라이언트(브라우저) -> 서비스 요청 (네트워크 연결 요청)
서버 -> 네트워크 연결 응답 -> TCP 연결
클라이언트 -> HTTP 요청
서버 -> HTTP 응답
클라이언트 -> HTTP 요청
서버 -> HTTP 응답
클라이언테(브라우저) -> HTTP 수용 (브라우저 분석)
서버 -> TCP 연결 해제

위의 방식이 일반적인 웹 서비스 메커니즘 (HTTP/1.1) 이라 보면 된다.

- 웹 벤치 마크 시 고려 대상

Single Server 일때의 성능 수치 분석
Multi Server (Cluster) 일때의 성능 수치 분석
-> Single node 일 경우 기본 시스템 상태에서의 성능값과 최적 튜닝 이후의 성능값을 파악해 둔다. 그런 후 Cluster Server 시에 성능을 체크하고 서버 수에 따른 성능 향상 손실값을 구한다.

Multi 서버 최대 성능 / (Single 서버 최대 성능 * n) ;; n 은 Multi node 수
손실값이 큰 경우 손실이 발생하는 원인에 대해 파악해 두어야 한다.

Static page 중심의 성능과 Dynamic page 중심의 성능 파악

방화벽이나 Proxy 서버등이 있을 경우 이 환경에서의 테스트 역시 필요하다.

벤치 마크 당시 시스템 모니터링으로 통해 시스템 상의 자원 활용률 역시 파악을 하여 특정 리소스 (CPU,MEM,I/O,NET)에 발생 하는 병목을 파악해 두도록 한다.

그리고 벤치 마크 시에는 반드시 웹 로그를 off 시키도록 한다. 수많은 요청을 하기 때문에 웹 로그의 크기가 짧은 시간에 급작스럽게 크지므로 디스크의 병목이 발생 할수 있다. (SPECweb 에서는 웹로그 사용해야함)

그리고 한대의 클라이언트 PC에서 테스트 경우 일정 부하 이후에 성능 증가가 없을 경우 두대의 클라이언트 PC로 벤치 마크 수행하고 만일 서버측에서 더 처리가 가능하면 이는 클라이언트 병목임으로 클라이언트에서의 최대 요청 수도 확인해 봐야 한다.

- 벤치 마크 순서

+ 테스트 환경 준비 및 확인
    - 장비 설치, 네트워크 연결
    - 테스트 환경에 대한 벤치 마크
    (network 벤치 마크 : netperf, 시스템 불필요 데몬 제거, 튜닝)
+ 벤치 마크 수행
    - 벤치 마크 실행 (반복 수행 후 평균 냄)
    - 서버 모니터링 (vmstat, process account)
    단위 테스트 별로 최소 10분 이상 테스트 수행
    SPECWeb 의 경우 20분 정도 수행
    부하를 높여 가며 반복 테스트 -> 자동화 된 툴 사용 가능 (autobench)

+ 결과 분석

- 벤치 마크 도구의 리포트 분석

Request per second : Maximum hit per dya 등을 구할때 기준 척도
Response time : 응답 시간에 민감한 경우 (쇼핑몰)
Max. concurrency : 동시 접속 수
Transfer rate : 네트워크 자원 소모량 측정

5.2. 웹서버 벤치 마크 툴과 사용 방법

5.2.1. ab 를 이용한 벤치마킹

http://httpd.apache.org 사이트에서 구할 수 있다. apache package 에 포함되어져 있으며 single, fixed 웹 환경에 사용 가능하다. 즉 단일 서버 성능 체크 시 유용하다. 사용이 간편하고 기본 apache 웹서버에 내장이 되어져 있어 별도의 설치 부담이 적다는 장점이 있다.

사용 옵션으론
-n : 웹 요청 수
-c : 동시 유저수
-k : KeepAlive 사용
-h : help

사용 예제)

Single Server Test

ab -n 50000 -c 1000 http://192.168.123.165/

```
==========================================================================
==
        Server Software:        Apache/1.3.31
        Server Hostname:        192.168.123.165
        Server Port:            80

        Document Path:          /
        Document Length:        7 bytes

        Concurrency Level:      1000
        Time taken for tests:   36.852 seconds
        Complete requests:      50000
        Failed requests:        0
        Broken pipe errors:     0
        Total transferred:      8601548 bytes
        HTML transferred:       350063 bytes
        Requests per second:    1356.78 [#/sec] (mean)
        Time per request:       737.04 [ms] (mean)
        Time per request:       0.74 [ms] (mean, across all concurrent requests)
        Transfer rate:          233.41 [Kbytes/sec] received

        Connnection Times (ms)
                      min  mean[+/-sd] median   max
        Connect:        0  142  774.9      2    9003
```

```
Processing:    18   166 1016.0     93 28055
Waiting:        3   166 1016.0     93 28055
Total:         18   308 1406.3     95 31060
```

==========

Cluster Server Test (2node)

ab -n 50000 -c 1000 http://192.168.123.175/

==========

```
Server Software:        Apache/1.3.31
Server Hostname:        192.168.123.175
Server Port:            80

Document Path:          /
Document Length:        7 bytes

Concurrency Level:      1000
Time taken for tests:   19.276 seconds
Complete requests:      50000
Failed requests:        0
Broken pipe errors:     0
Total transferred:      8600860 bytes
HTML transferred:       350035 bytes
Requests per second:    2593.90 [#/sec] (mean)
Time per request:       385.52 [ms] (mean)
Time per request:       0.39 [ms] (mean, across all concurrent requests)
Transfer rate:          446.20 [Kbytes/sec] received

Connnection Times (ms)
min  mean[+/-sd] median   max
Connect:        0  177  831.6      2    9001
Processing:     4  174  863.6     93   13966
Waiting:        3  174  863.6     93   13966
Total:          4  351 1281.1     95   16451
```

==========

unable access_log, keepalive on, max process modify 등등 수정 후 다시 시도

5.2.2. httperf 를 이용한 벤치마킹

ftp://ftp.hpl.hp.com/pub/httperf 에서 구할 수 있다.
비교적 단순하면서 다양한 통계 결과를 출력 할수 있다.
(https, 수행시간별 통계,access log relay 가능, CPU, Net, cookie, session..)

주요 사용 옵션

--hog : 웹 시뮬레이션 시 ephemeral port (1024~5000)를 사용할수 있게 한다.
기본적으로는 위 port 사용의 제한이 있음.

--server : 시뮬레이션을 할 웹서버 주소
--num-conns : Connection 수
--num-calls : connection 이 Closeing 되기 전에 다시 요청하는 수
--rate : session 과 connection 비율, --wsess 와 같이 사용 가능
--uri : request page
--wsess : session 요청 및 session 지연 시간 등을 지정
--ssi : https 사용

사용 예)
httperf --hog --server www --num-conns 100 --rate 10 --timeout 5
www 서버에 100 개의 connection 을 만들고 fixed rate 비용을 10 으로 한다.

Single Server TEST

httperf --hog --server=192.168.123.165 --rate=100 --num-conns=50000 ₩
--num-calls=10

==========

Maximum connect burst length: 1

Total: connections 1000 requests 2000 replies 1000 test-duration 9.993 s

Connection rate: 100.1 conn/s (10.0 ms/conn, <=1 concurrent connections)
Connection time [ms]: min 1.2 avg 1.7 max 2.4 median 1.5 stddev 0.2

Connection time [ms]: connect 0.4

Connection length [replies/conn]: 1.000

Request rate: 200.1 req/s (5.0 ms/req)

Request size [B]: 76.0

Reply rate [replies/s]: min 100.0 avg 100.0 max 100.0 stddev 0.0 (1 samples)

Reply time [ms]: response 1.2 transfer 0.0

Reply size [B]: header 193.0 content 7.0 footer 2.0 (total 202.0)

Reply status: 1xx=0 2xx=1000 3xx=0 4xx=0 5xx=0

CPU time [s]: user 3.35 system 6.63 (user 33.5% system 66.3% total 99.9%)

Net I/O: 34.4 KB/s (0.3*10^6 bps)

Errors: total 1000 client-timo 0 socket-timo 0 connrefused 0 connreset 1000

Errors: fd-unavail 0 addrunavail 0 ftab-full 0 other 0

==========

Cluster Server TEST (2node)

```
#   httperf --hog --server=192.168.123.175 --uri=/index.html --rate=100 ₩
     --num-conns=1000 --num-calls=10
```

==========

Maximum connect burst length: 1

Total: connections 1000 requests 2000 replies 1000 test-duration 9.994 s

Connection rate: 100.1 conn/s (10.0 ms/conn, <=1 concurrent connections)

Connection time [ms]: min 1.2 avg 1.9 max 2.8 median 1.5 stddev 0.4

Connection time [ms]: connect 0.5

Connection length [replies/conn]: 1.000

Request rate: 200.1 req/s (5.0 ms/req)

Request size [B]: 76.0

Reply rate [replies/s]: min 100.0 avg 100.0 max 100.0 stddev 0.0 (1 samples)

Reply time [ms]: response 1.3 transfer 0.0

Reply size [B]: header 193.0 content 7.0 footer 2.0 (total 202.0)

Reply status: 1xx=0 2xx=1000 3xx=0 4xx=0 5xx=0

CPU time [s]: user 3.03 system 6.97 (user 30.3% system 69.7% total 100.1%)

Net I/O: 34.4 KB/s (0.3*10^6 bps)

Errors: total 1000 client-timo 0 socket-timo 0 connrefused 0 connreset 1000

Errors: fd-unavail 0 addrunavail 0 ftab-full 0 other 0

==========

====

5.2.4. http-load 를 이용한 벤치마킹

http://www.acme.com/software/http_load 에서 구할 수 있다. ab 에 비해 오랜 시간 테스트가 가능하고 ab 보다 높은 성능 수치를 보이는 우수한 성능의 벤치 마크 도구이다.

사용 예)

http_load -parallel 1000 -seconds 30 url http://192.168.123.165/index.html

-parallel : 동시 접속 수
-seconds : 테스트 수행 시간
url : 테스트 페이지 경로

5.2.4. SPECweb을 이용한 벤치마킹

http://www.spec.org 에서 구할 수 있다. SPECWeb 벤치 마크 툴은 공인 상용 벤치 마크툴로 실제 웹 사이트와 유사한 환경을 자동 생성한 후 통계적 기법으로 시뮬레이션 하는 툴로 객관적인 평가 자료로 많이 활용 되어진다.

5.3. 벤치마크 시나리오 수립 및 분석 하기

준비 작업

(1) 클라이언트/서버 벤치마크 작업에서 반드시 검사해야 할 부분 중의 하나는 벤치마크 대상이 서버가 되어야 한다는 점이다. 잘못된 벤치마크를 수행하게 되면 서버의 성능이 아닌 클라이언트나 네트워크의 성능을 벤치마크 하게 되므로 클라이언트의 수를 충분하게 확보해주고, 클라이언트 기계 자체도 튜닝을 해주어야 한다. 가장 간단한 튜닝으로 클라이언트에서 사용 가능한 포트 수를 늘려주는 것이 있다.

```
echo "1100 61000" >/proc/sys/net/ipv4/tcp_local_port_range
```

(2) 테스트 될 웹 서버의 로그 기능을 끄도록 한다. Apache의 경우 httpd.conf에서

CustomLog /var/log/httpd/access_log common

부분을

CustomLog /dev/null common

로 바꾼다. 이렇게 하지 않으면 테스트 도중 파일 시스템이 가득 차거나 웹 서버에서 더 이상 로그를 기록하지 못하게 될 수도 있다.

(3) 클라이언트 기계를 네트워크에 연결시킬 위치(스위치, 허브 등)를 확인하고 어떠한 네트워크 구성을 갖는지 미리 확인한다.

이번 실험에서의 테스트 위치는 다음과 같다.

- 서버와 같은 LAN 상
- 방화벽 바깥쪽
- 인터넷 연결을 통해서 (optional)

테스트 순서

Single server LAN (step 1)

개별 서버가 최고의 성능을 발휘할 수 있도록 튜닝되어 있는지 여부 검사

Cluster server LAN (step 2)

클러스터링을 통한 성능 향상 여부 검사

Single server Firewall (step 3)

Step 1과 비교하여 방화벽이 성능에 문제를 일으키는지 검사

Cluster server Firewall (step 4)

Step 2와 비교하여 방화벽과 엔클러스터가 같이 동작하는데 문제가 있는지 검사

Single server Internet (step 5)

Step 1과 비교하여 인터넷 접속에 문제가 있는지 검사

Cluster server Internet (step 6)

최종 사용 환경과 유사한 테스트

실험 당일 현장 상황의 제약으로 인해서 실제 실험은 step 1, 2, 3에 대해서만 진행되었다.

테스트 수행

벤치마크 파라메터는 다음과 같다.

- Apache benchmark

  ab –n 50000 –c 1000 http://target_host/index.html > $log_file
- http_load

  http_load –parallel 1000 –seconds 120 url_file > $log_file
- httperf

  httperf --hog --server=target_host --uri=/index.html --rate=500 --num-conns=50000 --num-calls=10 > $log_file

Target host는 single server의 경우 211.189.1.40, cluster server의 경우 211.189.1.46을 사용했고, url로 사용된 index.html의 크기는 2504 bytes이다.

ab나 http_load의 경우 동시 접속 사용자 1,000을 기준으로 서버의 성능을 테스트하였고, httperf의 경우 connection rate을 500으로 하고 request / connection을 10으로 하였다. 결과적으로 5,000 request / sec의 부하가 서버에 걸리게 된다.

Workload의 특성으로 살펴보면, ab, http_load는 HTTP/1.0 with no keep-alive (1 request / 1 connection)이며, httperf는 HTTP/1.1 with keep-alive (10 request / 1 connection)이다. 이론적으로 보면 connection overhead가 작은 httperf의 성능이 다른 것들에 비해 잘 나와야 하는데 결과에서 볼 수 있듯이 그렇지 못했다. 이는 실험에 문제가 있었던 것으로 생각된다.

모든 BMT 수행 결과를 로그로 기록한다. 로그 파일의 이름은 다음과 같이 배정한다.

$(method)-$(server_type)-$(network).log

method: ab(apache bench), hl(http_load), hp(httperf)

server_type : sg(single), cl(cluster)

network: lan(LAN), fwl(Firewall), int(Internet)

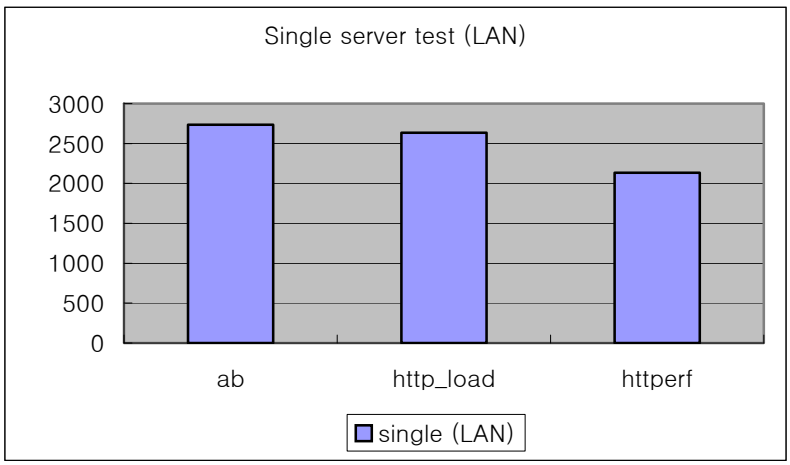
벤치마크가 수행되는 기계의 대수는 서버의 대수와 같거나 많게 하는 것이 좋다. 하지만 실험 기자재의 부족으로 single server의 경우 client machine 1대, cluster server의 경우 client machine 3대가 사용되었다.

실험 결과

(1) Single server LAN

실험 결과에서 볼 수 있듯이 단일 서버의 경우 2,500 request/sec 이상의 성능을 보이고 있다. 이는 dual CPU를 가진 서버에서 적절히 튜닝 되어 있는 경우 나타나는 성능이라 얘기할 수 있다(정적 문서 서비스). httperf의 경우 session 당 10개의 서비스 요청을 하는 구조임에도 불구하고 성능이 안 좋은 것을 볼 수 있는데, 실험에 사용된 httperf에 문제가 있었다는 것이 차후에 밝혀졌다. 그래프를 해석할 때, 도구간의 성능 비교는 무의미하다(도구별로 특성이 다르기 때문).

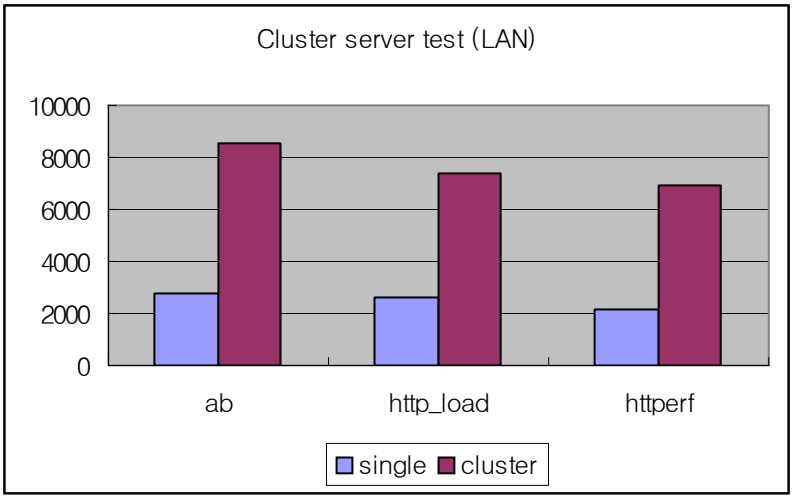
네트워크 전송률로 보면 전체 100Mbps 중에서 60Mbps를 사용한다. TCP 자체가 아닌 HTTP라는 프로토콜을 사용한 점을 감안한다면 상당히 높은 수치가 나온 것임을 할 수 있다.



(2) Cluster server LAN

다음 결과는 클러스터 서버에 대한 성능을 (1)의 단일 서버의 성능과 비교하여 보인 것이다. 단일 서버의 성능에 비해 3배가량의 성능(8,000 request/sec 안팎)이 나오는 것을 확인할 수 있다. 실제 클러스터링에 사용된 서버는 4대(작업 분산용 서버 1대 제외)이지만, 벤치마크에 사용된 클라이언트 기계가 3대뿐이었기 때문에 이러한 결과가 나온 것이라 생각할 수 있다.

클러스터링 사용 시 네트워크 전송률은 189Mbps 정도이다. 외부 라인의 대역폭이 100Mbps라는 점을 감안한다면 충분한 대역폭이라 할 수 있다.



서버가 동적 문서 처리(CGI, JSP 등)로 인한 부하가 크거나 외부 대역폭이 커지게 된다면 지금보다(4대) 더 많은 작업 서버가 필요하게 된다. 이러한 경우 다음과 같은 점들이 고려되어야 한다.

- 동적 문서 처리로 인한 부하 증가
  서버 부하의 증가로 인해 서비스가 느려지게 되면 단순히 작업 노드의 수를 증가시켜 주는 방법으로 해결이 가능하다.
- 외부 대역폭 증가
  외부 대역폭이 증가하고 이를 충분히 활용하기 위해서는 작업 노드의 수 뿐만 아니라 작업 분배 서버의 수도 증가시켜주어야 한다. 작업 노드 수와 작업 분배 서버 수의 비율은 서비스의 내용에 따라 다르기는 하지만 정적 문서 서비스의 경우 5:1 정도가 적합하다. 동적 문서 서비스의 경우 한 대의 작업 분배 서버에서 지원 가능한 작업 노드의 수는 더욱 커지게 된다. 이는 작업 분배 서버의 필요 대수가 실제로 서버가 서비스하는 내용의 복잡도 보다는 서비스를 위해 사용되는 네트워크 대역폭과 밀접한 관련이 있기 때문이다.

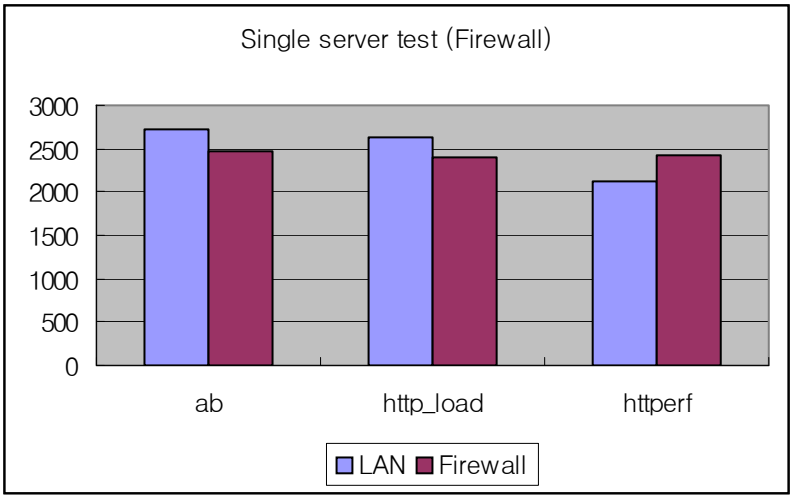
현재 상황으로 본다면, 작업 분배 서버 한 대만으로도 충분히 외부 접속 라인(100Mbps)을 감당할 수 있기 때문에 서비스가 느려지는 경우 작업 노드만 추가함으로써 이를 감당할 수 있으리라 예상할 수 있다.

(3) Single server Firewall

다음 결과는 방화벽을 통한 단일 서버의 성능 측정 결과이다. 일반적으로 알려져 있듯이 방화벽을 사용하게 되면 보안성은 좋아지지만 방화벽 자체의 성능 한계로 인해 전체적인 서비스 성능은 약간 떨어지게

된다. 그래프에서도 볼 수 있듯이 LAN 환경에서 직접 테스트를 한 것에 비해 10% 정도의 성능 감소가 있다는 것을 알 수 있다(httperf의 경우 실험 오류로 인해 비교 불가능). 네트워크 전송률의 경우 55Mbps를 사용한다.

아쉽게도 클러스터링이 되었을 때의 방화벽 성능은 측정할 수가 없었다. 이는 방화벽 외부에 연결 가능한 클라이언트 기계의 대수 제한으로 인한 문제였는데, 보다 많은 대수의 클라이언트 기계를 사용한다면 방화벽이 외부 100Mbps 접속 라인을 얼마만큼 활용할 수 있는지 파악할 수 있을 것이다.



결론

이상으로 현재 SDS에 설치되어 있는 서버 시스템에 대한 벤치마크를 수행하고 그 결과를 간단히 분석해 보았다. 비록 준비가 부족하기도 했고 실험 시간이 부족하기도 했지만, 복잡하게 구성되어있는 전체 시스템에 대한 전반적인 평가를 해볼 수 있었다는 것에 대해 의미를 부여하고 싶다.

벤치마크 결과는 실제로 서비스 환경이 제대로 spec 상의 성능을 내어주는지를 확인하고 성능상에 문제가 있을 때 이를 분석하는데 유용한 자료로 활용될 수 있다. 뿐만 아니라 시스템의 어떠한 부분이 차후 성능 개선에 있어서 우선적으로 업그레이드 되어야 하는지도 밝혀낼 수 있는 back data를 제공한다는 측면에서도 그 의미를 찾아볼 수 있다.

실제로 이번 BMT 과정을 통해서 클루닉스의 서버 및 클러스터링 솔루션의 성능을 확인할 수 있었고 방화벽의 성능 튜닝이 이루어졌으며, 시스템 전반의 성능에 대한 이해를 증진할 수 있는 좋은 기회였다고 생각된다.

차후 고려 사항

(1) httperf 등의 복잡한 벤치마크 수행

이번 실험에서는 httperf의 수행 환경에 문제가 있어서 제대로 된 테스트 결과를 얻을 수 없었다. 하지만, httperf는 수행 시간 조정 및 다양한 환경에서의 측정을 가능하게 하므로 차후 setup을 제대로 해서 다시 수행해볼 필요가 있다.

(2) 실험 시간의 증가 (최소 30분 정도)

실험 시간이 짧으면 결과물의 편차가 크게 되어 의미 있는 데이터를 얻기 힘들다. 최소 30분 이상의 실험을 3회 이상 반복하여 평균값으로 결과를 구해보는 것이 보다 의미 있는 데이터를 얻을 수 있는 방법이다.

(3) 실험 환경 벤치마크

제대로 된 방식으로 서버 벤치마킹을 실시하기 위해서는 클라이언트 기계 및 네트워크 자체에서 성능에 문제를 일으키는 부분이 없는지 검사를 하는 부분이 선행되어야 한다. 이번 실험의 경우 시간 및 장비 제약상 네트워크 환경 자체에 대한 검사가 부족했으며 클라이언트 기계의 수도 부족한 편이었다.

Appendix. 서버 모니터링

테스트 진행 중에 서버의 상태를 체크 하여 서버의 성능에 문제가 있는지를 파악한다. 가장 간단한 방법으로는 테스트 되는 웹 서버와 디렉터에서 vmstat을 수행하는 것이다. 다음은 vmstat을 수행했을 때의 결과 예를 보인 것이다.

```
vmstat 1 (1초 간격으로 성능 검사)
    procs                  memory      swap          io     system
cpu
 r  b  w   swpd   free   buff  cache   si   so    bi    bo   in    cs  us
sy  id
 0  0  0   3384 354612 469580  37352    0    0     1     1   11     9
0   0   8
 0  0  0   3384 354612 469580  37352    0    0     0     0  116    13
0   1  99
```

특히, 여기에서 눈 여겨 볼 것은 in, cs, id 이다. In은 interval 내의 interrupt 회수, cs는 context switch, id는 CPU idle을 나타낸다. 웹 서버의 경우 cs의 수치가 급증(수만 단위)하면서 id이 0에 가까우면 더 이상의 성능 증가는 기대하기 힘들고, 디렉터의 경우 in의 값이 3~4만 정도이면 더 이상의 성능 증가를 기대하기 힘들다.

드문 경우이기는 하지만 free의 값이 0에 가까우면서 si, so의 값이 증가하는 경우가 있다. 메모리가 부족해서 swap이 활발하게 사용되는 경우이니 메모리를 늘리라고 권하는 수 밖에 없다.

한가지 첨언하자면, in값이 매우 크고 id가 비교적 큰(40~50 정도) 경우가 있는데, 디렉터와 같은 환경에서 자주 발견되는 현상이다. 대부분 PCI 버스 자체의 병목현상이 원인이므로 이런 경우 디렉터를 여럿 두는 것이 안전한 해결 방안이다. 이 경우 CPU를 업그레이드하는 것은 별 도움을 주지 못한다. 경우에 따라서는 NIC tuning을 통해 약간의 성능 개선을 얻을 수 있기도 하다(특히 gigabit NIC의 경우).

5.4. 웹 시스템 튜닝 하기

규모 서비스를 준비하는 경우 운영체제의 제한사항을 먼저 확인해야한다. 동시에 열수 있는 총파일수, 한 프로세스가 열수 있는 파일수 등등.

예를 들어 대형 웹서버를 아파치로 서비스하는 경우를 생각해보자.
아파치는 기본적으로 프로세스 방식으로 서비스를 처리한다. 이건사용자의 요구가 올때마다 하나의 프로세스를 띄우므로 만약 동시에 10명의 사용자가 접속을 하면 10개의 프로세스가 떠야한다는 것이다.
최근의 아파치 서버는 MaxClients 150 이라고 설정되어있다. 이건 동시에 150개의 프로세스를 띄울수 있으며 결국 동시에 150명을 받아 들일 수 있다는 것이다.
(실제로 이정도만 하더라도 절대로 작은 규모는아니다) 그런데 만약 nobody가 만들어낼 수 있는 최대 프로세스 개수가 그 이하라면? 당연히 문제가 생길 것이다. 물론 최근 레드햇 6.0 이상 버전은 그 이상으로 맞추어져 있어서 문제가 생기지는 않겠지만.

문제는 프로세스가 많이 뜨면 프로세스뿐만이 아니라 열 수 있는 파일 수에서도 문제가 된다.

그러면 먼저 프로세스의 자원 한도에 대해서 알아보자.

5.4.1 서버 튜닝

- 프로세스의 자원한도
프로세스의 자원한도를 리눅스에서는 ulimit 를 통해서 알 수 있다.
(Redhat 6.0, PowerLinux 등)

```
# ulimit -a (또는 ulimit -Sa)   --> soft 한도
core file size (blocks)  0
data seg size (kbytes)    unlimited
file size (blocks)        unlimited
max memory size (kbytes) unlimited
stack size (kbytes)       8192
cpu time (seconds)        unlimited
max user processes        2048
pipe size (512 bytes)     8
open files 1024
virtual memory (kbytes)  2105343
```

```
# ulimit -Ha    ------>> hard 한도
core file size (blocks)  unlimited
data seg size (kbytes)    unlimited
file size (blocks)        unlimited
max memory size (kbytes) unlimited
```

stack size (kbytes) unlimited
cpu time (seconds) unlimited
max user processes 2048
pipe size (512 bytes) 8
open files 1024
virtual memory (kbytes) 4194302

소프트한도는 새로은 프로세스가 만들어졌을때 디폴트로 적용되는 자원의 한도입니다. 이것을 하드한도까지 증가시킬 수 있습니다.
그렇지만 이 한도를 넘어서 확장하는것은 슈퍼유저만이 가능합니다.
하드한도는 절대적인 선이지요.
그렇다면 하드한도는 슈퍼유저라고 무한대로 늘릴수 있는가?
절대 아니지요. 이건 커널차원에서 지정을 해야합니다.
이에 대해서는 뒤에서 설명합니다.

코어파일의 최대크기
프로세스의 데이타 세그먼트 최대크기
쉘에서 생성되는 파일ㅢ 최대크기
resident set size의 최대크기(메모리 최대크기)
프로세스의 스택 최대크기
총 누적된 CPU시간(초)
단일 유저가 사용가능한 프로세스의 최대갯수
512-바이트 블락의 파이프 크기
open file descriptors의 최대 숫자(열수있는 최대파일수)
쉘에서 사용가능한 가상 메모리의 최대용량

각 설정을 수정할 수 있는데 학교에서 실습용 워크스테이션으로 쓰는게 아니라면 보통 이런작업을 할 일은 없을듯.
여기서 각 항목에 대해서 잘 모른다면 프로그래밍, OS 관련책을 보셔야할듯. (실은 저도 다 까먹었음)
위에서 하나의 프로세스가 열 수 있는 파일의 최대값은 ulimit 명령을 이용해서는 수정이 되지 않습니다.
단지 보여주기만 할 뿐이죠. 이건 커널소스를 뜯어고치든지 /proc 에서
직접 수정하든지 해야합니다.

최근의 리눅스배포판에서는 유저당 총 2048개의 프로세스를 띄울 수 있고 하나의 프로세스가 총 1024개의 파일을 열 수 있습니다. 실제로 이건 커널 2.0대에서 2.2대로 올라가면서
가장 크게 변한 부분중의 하나입니다. 실제로 이정도만해도 일반적인 서비스에서는 충분하고 이걸 바꾸어야 할 경우는 거의 없을 것입니다. 그런데 우리의 목표는 대규모 서비스를 하는 것이잖아요?

- 파일, 프로세스 갯수 조정

주요하게 살펴야할 것듯.

리눅스에서 동시에 열 수 있는 파일수 : NR_FILE , 4096
한 프로세스가 열 수 있는 파일수 : NR_OPEN, 1024

```
# vi /usr/src/linux/include/linux/fs.h

# per user max open file
#define INR_OPEN 1024            /* Initial setting for nfile rlimits */

# system max open file
#define NR_FILE  4096    /* this can well be larger on a larger system */
```

여기서 한 프로세스가 열수 있는 파일수를 수정하려면
/usr/src/linux/include/linux/limits.h 에서

```
# fs.h에서의 per user max open file(INR_OPEN)과 동일하게 지정
#define NR_OPEN          1024
```

이부분도 수정을 해주어야합니다.

수정을 하였으면 컴파일을 해서 테스팅을 해야겠지요.
여기서 주의할 것은 메모리가 적은 시스템의 경우라면 부팅이 되지
않을 수도 있습니다.

그런데 굳이 컴파일을 하지 않아도 /proc 를 이용 변경할 수 있습니다.

```
# cat /proc/sys/fs/file-max
4096

# echo 8192 > /proc/sys/fs/file-max
# cat /proc/sys/fs/file-max
8192
```

```
# cat /proc/sys/fs/file-nr
591     184     8192
```

여기서 591는 현재 할당된 파일핸들, 184는 그중사용된 파일핸들,
8192는 파일핸들의 최대숫자입니다. 할당된 파일핸들이 최대치로
가더라도 실제 사용된 파일핸들의 숫자가 여유가 있다면 걱정할
필요는 없습니다.
(만약 시스템에서 "running out of file handles" 라는 메시지가
나온다면 열수 있는 파일에 문제가 있다는 것입니다)

그런데 안타깝게도 한 프로세스당 열수 있는 파일수(NR_OPEN)은
쉽게 바꿀 수가 없습니다. 기본 1024인데 이건 위에서 말을 하대로
컴파일을 하세요. 그다음 시스템이 뻗을지 계속 유지될지는...

여기서 좀만 더 살펴보지요.
(다시 NR_FILE 이 4096이라 하구요)

```
# cat /proc/sys/fs/inode-max
8319
```

```
# cat /proc/sys/fs/inode-nr
8340    1006
```

파일 핸들에 따라 커널에서 동적으로 inde structures를 할당합니다.
inode-max는 inode 핸들러의 최대값입니다.
여기서 inode-max 값은 3-4배이상으로 하라고 추천합니다.

/usr/src/linux/Documentation/proc.txt 에 /proc 에 대한
상세한 설명이 나와있는데 이에 대해서 살펴보지요.

This value should be 3 to 4 times larger than the value
in file-max, since stdin, stdout, and network sockets also need an
inode struct to handle them.

그러니깐 stdin(표준입력), stdout(표준출력), 네트웍 소켓에서 파일을 다루기 위해 inode struct 가 필요하다고 하네요.

메일서버가 200개 이상 실행되거나 웹서버가 동시에 400개 이상 실행되는 대형 사이트에는 최대값을 크게 만드는 것이 좋다고 하네요. 그런데 중요한건 무조건 늘린다고 좋은것은 아니겠지요?
최대값이 늘어날 수록 더 많은 메모리가 필요하고 자원할당도 더 많이 필요하기요.

그래서 가능하면 어떤 프로그램에서 쓸데없는 파일은 열지 않도록 하면 파일핸들을 할당하지 않기때메 열린 파일수가 줄어들겠지요. 하나의 열린 파일수를 줄이면 268개의 파일을 닫는 효과가 있다고 하는데 제가 이부분은 아직 정확히 이해가 가지 않습니다.

이번엔 시스템의 최대 프로세스수를 생각해볼까요?
예를 들어 텔넷으로 접근하는 경우 최소한 로그인프로그램과 쉘프로그램 두개의 프로세스가 생성되겠지요? 256명이 접근하면 최소 512. 그렇다면 시스템의 최대 프로세스 갯수를 설정하는
것도 위와 비슷합니다.

/usr/src/linux/include/linux/tasks.h

여기서보면

```
#define NR_TASKS        2560    /* On x86 Max 4092, or 4090 w/APM configured. */

#define MAX_TASKS_PER_USER          2048
```

x86 시스템에서는 최대 4092까지 가능하다고 되어있고 유저당 최대 프로세스수는 2048입니다. 이건 ulimit 에서
보았지요?

자 이제 마지막으로 프로세스에서 열어놓은 파일을 어떻게 확인할 것인가?

lsof 프로그램을 이용하면 됩니다.

5.4.2 apache 튜닝

- apache 튜닝
고성능의 아파치 서버를 셋팅 하는법

(1) 기본값으로 되어 있는 MaxcClient 254 를 1024 까지 높인다.
--> 추가적으로 커널 컴파일과 아파치 컴파일 과정이 필요하다.

커널 컴파일
/usr/src/linux/include/linux/아래의
fs.h / tasks.h / limits.h 세 화일안에서 NR_OPEN(files/1PS),NR_FILE(files/SYS)
NR_TASKS(PSs/SYS)등을 다음과 같이 조정.

files/process NR_OPEN 1024
NR_SUPER 256 (파일 시스템 마운트수)
NR_INODE 24576
NR_FILE 8192 (동시에 열수 있는 파일수)
files/user CHILD_MAX 4096
OPEN_MAX 1024
NR_TASKS 2048 (프로세스 총수)
NR_TASKS/8
MIN_TASKS FOR_ROOT 16

아파치 컴파일
src/include/httpd.h 의 HARD_SERVER_LIMIT 를 256 에서 1024 로 조절.

예) 기본 설치값에서 MaxClient 를 1240 으로 설정했을 때...
WARNING: MaxClients of 1240 exceeds compile time limit of 256 servers,
lowering MaxClients to 256. To increase, please see the
HARD_SERVER_LIMIT define in src/include/httpd.h.
(2) port 를 80 번외에 추가로 설정.
- cp /www/conf/http.conf /www/conf/httpd8080.conf
- cp /www/bin/httpd /www/binhttpd8080
- adduser www
- httpd8080.conf 의 user, group 를 nobody 에서 www 로 변경
- /www/bin/httpd8080 -f /www/conf/httpd8080.conf 로 추가 데몬 설치
- 홈페이지내에 접속자수가 많은 링크를 http://domain 명:8080/filename.html 로 수정.
- 이런 방식으로 8081, 8082, 8083.... 등으로 여러 port 를 둘 수 있다...

(3) DNS 를 이용한 Round Robin 방식(Load Balancing 효과)
웹서버 자체를 여러개의 서버로 분산.
IP Based Virtual Host : 하나의 도메인에 여러 IP 할당.

(4) 서버의 설정 조정

MaxClients : 1024

HostNameLookups off

MaxSpareServers 40

MinSpareServers 20

StartServers 20

MaxRequestsPerChild 30

(적절히 설정하여 새로운 자식 프로세스가 생성될 때의 오버헤드를 줄일 수 있다.)

KeepAlive 5

(5) apache 컴파일시 불필요한 모듈을 첨가하지 않는다.(예: php)

ServerAlias 등을 이용하여 httpd.conf 의 사이즈를 최소화한다.

access_log / error_log 를 생성되지 않도록 한다.

(6) 튜닝 시 고려 사항

웹서버 설정시에 가장 중요한 부분이 Timeout 부분입니다.

MaxProcess 수를 무작정 늘이는 것은 결코 좋은 일이 아닙니다.

단순한 static file(html, image 등)을 serving하는 일이라면, 보통 하나의 파일을 serving하는데 5~10ms 정도 시간이 소모되는데, 이렇게 되면 1초에 100~200request 밖에 처리하지 못합니다. 불필요한 system call을 줄이고(이건 httpd source에서 불필요한 부분을 제거해야 합니다 (불필요apache modules, 불필요 기능), 불필요한 network data(dns lookup 등) 모으는 작업을 줄이면 1 CPU에서 초당 수백 request를 처리할 수 있습니다.

CPU가 2개가 되면, static file을 serving하는 경우, 거의 2배로 용량이 늘어난다고 보면 됩니다.

이런 경우에는 Max Process수를 수백개로 설정하는 것이 가능하지만, dynamic file을 생성하는 경우에는 그렇지 않습니다.

보통 웬만한 경우, 아무리 빨리 처리를 해도, 어느정도 기능을 갖춘 dynamic html의 경우, 50~200ms 정도의 시간이 소요됩니다. 이렇게 되면 기껏 초당 5~20 request를 처리하게 됩니다.

(CPU개수가 많아지면, 그만큼 늘어납니다.)

이때 만일 max process수를 초당 처리할 수 있는 건수 이상으로 늘여주게 되면, request만 받은 상태에서 답을 해주지 못하는 httpd의 수가 늘어나게 되고, 웹서버가 버벅거리게 됩니다.

이 때에는 서버를 추가하여 load를 분산시키는 수밖에 없습니다.

PentiumII 500MHz Dual 서버라 하더라도, dynamic page는 초당 50~100건 이상 처리하기 힘들고, max process수를 이 이상 설정 하게 되면 순식간에 load가 10이상으로 올라갑니다.

그리고 가장 중요한 것. Timeout 설정입니다. 아래에서는 keep-alive 를 설정해 놓은 경우, 하나의 connection
에서 계속해서 다음 request를 처리할 수 있기 때문에 효율적 이라고 하지만, 실제로는 그렇지 않습니다. keep-alive 를 허용하고 그 timeout을 5초로만 설정해도, 하나의 request를 처리한 후 적어도 5초동안은 그 httpd가 다른 작업을 하지 못하고 다음 request를 기다리게 됩니다.

보통 웹브라우저들은 서버로 동시에 4개의 connection을 만들게 됩니다. 한 페이지를 보는데 이미지 등등 해서 보통 4개의 connection을 만드는 것은 기본이죠. 이렇게 되면 httpd가 100개 떠 있다고 해도, 실제로는 동시에 25명의 방문자밖에 처리하지 못합니다.

그리고 keep-alive timeout이 5초인 경우, 한 명의 방문자를 처리한 후 적어도 5초동안은 계속해서 기다리면서 httpd가 놀게 됩니다.
(그렇다고 해서 httpd의 수를 늘여주면 앞의 문제 때문에 load가 몰릴 때 순간적으로 부하가 지나치게 많이 걸리게 됩니다. 어떤 request는 수초가 지난 후 답을 받는 등 quality of service가 많이 떨어지죠.)

결국 한 명의 방문자를 처리하는데 4개의 httpd가 5초동안 작업 한다는 뜻이고, 100개의 httpd를 띄워봐야 1초에 5명의 방문자밖에 처리하지 못하는 셈입니다.
(1 명 / 5 sec / 4 httpd = 5 / 1 sec / 100 httpd)

하지만 상반된 경우도 있습니다. 실제 페이지의 크기 자체가 클 경우 keepalive 를 off 했을 경우 서비스 전송이 재대로 처리 되지 못해 TIME_OUT 상태로 빠지는 경우가 있습니다. 클라이언트 브라우저에서는 글자와 그림 몇 개만 나오고 계속 대기 상태로 있는 증세가 나타 날수 있습니다. 그렇기 때문에 Timeout 설정은 현재 서비스 하는 사이트 특성에 맞게 적절히 지정해야 합니다.

대부분의 페이지가 text 중심으로 되어진 검색엔진 서비스 등 traffic이 많은 사이트에서는 keep-alive옵션을 반드시 꺼 놓게 됩니다. 그리고 connection timeout도 상당히 짧게 설정해 놓죠. 4~5초 이내로 말입니다. 하지만 쇼핑몰과 같이 다수 이미지 중심의 사이트의 경우는 반드시 KeepAlive 와 timeout 설정을 넉넉히 잡아 주셔야 합니다.

또 dynamic file 을 serving하는 httpd 앞에 빠른 proxy server를 두어서, 반복해서 요청되는 dynamic file의 경우, proxy server에 caching되어 있는 정보를 보여주게 하고, dynamic file을 serving 하는 서버에서 static file을
serving하느라 httpd를 잡아먹지 않도록 static file을 serving하는 웹서버는 모두 별도로 분리시키게 됩니다.

6. 리눅스 클러스터를 이용한 대용량 웹 시스템 구성

(시스템 컨설팅)

6.1. 인터넷 웹 클러스터 개론

6.1.1 부하 분산 클러스터 표준 시스템 구성

클러스터는 크게 과학 계산 클러스터와 부하 분산 클러스터로 나누어 지며 네트워크 구성에 따라 NAT (Network Address Translation) 방식과 DR (Direct Route) 방식으로 나누어 집니다. 인터넷 웹 서버 클러스터 시스템은 부하 분산 클러스터에 해당되며, 그 구성으로는 크게 웹 서버군, 파일 서버군, DB 서버군으로 나눌 수 있습니다. 웹 서버군은 다시 LB 서버, 정적 웹 서버(html 형식), 동적 웹 서버(Cgi, php, jsp, perl, c 형식) 로 나눌 수 있습니다.

네트워크 구성은 클러스터 시스템 구성 내에서 데이터 보안 중요도에 따라 공인 망과 사설 망으로 분리 할 수 있습니다. 실질적인 서비스 요청을 처리하는 웹 서버 군은 공인망에 연결하며, 데이터가 저장되는 파일 서버(NAS)와 DB 서버는 사설 망으로 구성하는 것이 바람직합니다.

1) LB 서버

LB 서버에서는 서비스 요청을 부하 분산 알고리즘에 따라 실제 작업 서버로 전달함으로 LB 서버가 제어하는 작업 서버의 수에 따라 LB 서버의 수를 결정할 수 있습니다.
클러스터 구성이 NAT 구성일 경우 한 작업 서버에서 소요되는 네트워크 대역폭이 10M 라 가정하고, LB의 NIC(네트워크 카드)가 100M 라 할 때 이론적으로 하나의 LB에 10대 이상의 작업 노드가 연결되면, LB 서버에 네트워크 병목 현상이 일어 납니다. 이는 언제나 이론적인 계산일 뿐입니다. 서비스 별로 사용되는 네트워크 사용량이 틀리기 때문에 실제 LB에서 측정되는 네트워크 대역폭을 측정하고, LB서버의 NIC 가용 대역폭의 60%가 초과 할 시에는 LB서버를 증설하거나 LB의 NIC를 1G 급으로 전환하는 방법을 고려해야 합니다. 비용적인 부분을 절감하는 방법으로 LB에서 여러 개의 NIC를 추가하여 Static routing table로 네트워크 환경을 구성하여, 프로그램(html,php,jsp등..) 혹은 DNS서버에서 적절히 Network Class 를 분리 시키는 방법이 있습니다. 하지만 초기 사이트 개발 시 이를 고려 하지 않은 경우에는 엄청난 프로그램 코드를 수정해야 하는 경우가 발생할 수 있으니 상황에 맞게 처리 하면 됩니다.

2) 웹 서버

웹 서버에는 시스템 파일만을 저장하도록 하고, 모든 데이터 파일은 파일 서버에 보관하여, 웹 서버에서는 단지 웹 프로세스 처리 하는 방식으로 구성하는 것이 시스템 성능이나 보안 측면에서 효율적으로 운영할 수 있습니다. 웹 서버는 기본적으로 NIC 두 개 이상 설치 하여 하나는 공인 망으로 웹 서비스를 담당하고, 다른 하나는 사설 망으로 파일서버와 DB서버와의 통신만을 하게 함으로써 외부에서 바로 데이터에 접근할 수 없도록 구성함으로 보안을 강화하고, 또한 외부 통신 네트워크 대역과 내부 데이터 전송 전용 네트워크 대역을 나눔으로 하여, 네트워크 성능을 향상 시킬 수 있습니다.

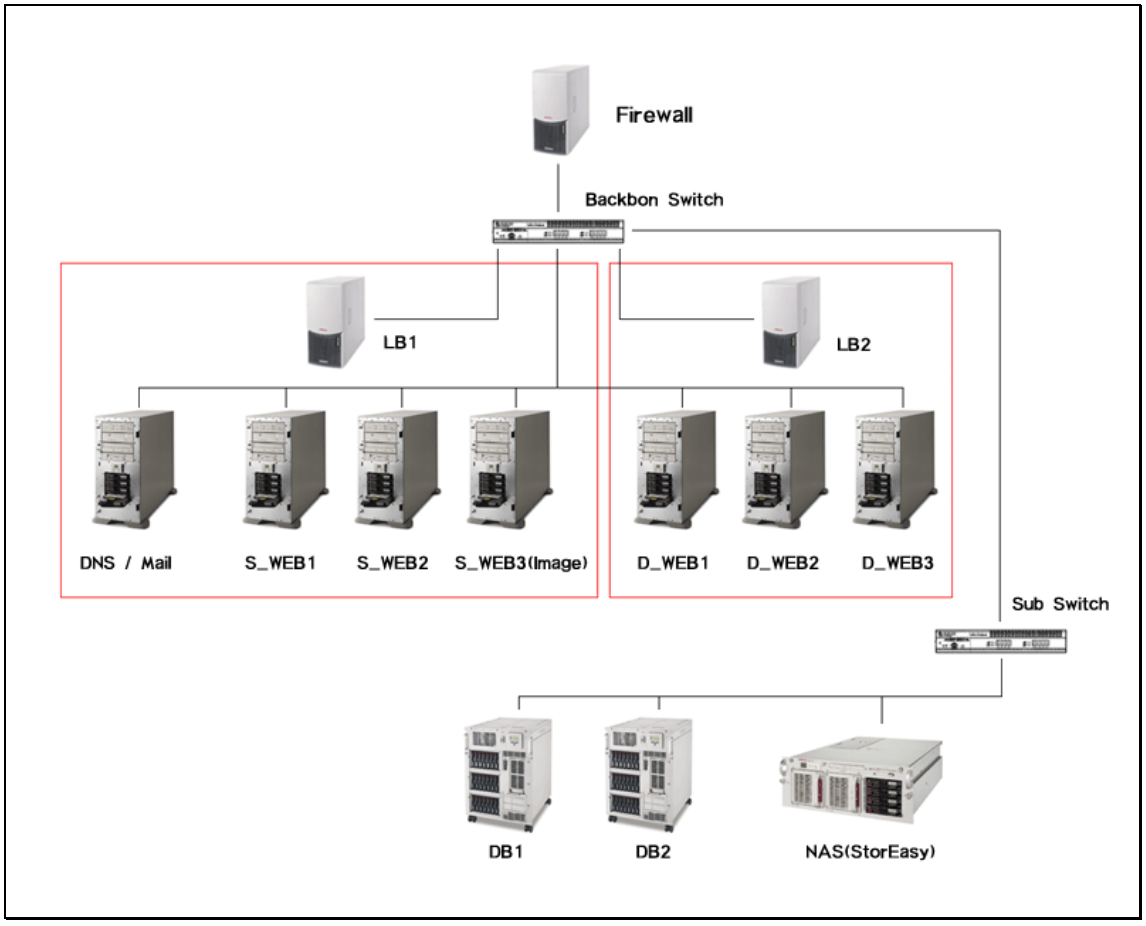
3) 파일 서버

파일 서버 구성은 사이트 특성에 맞게 Raid level 을 정하고, 두 개 이상의 NIC를 설치 하여 웹 서버 호스트 별로 다른 네트워크 인터페이스를 이용할 수 있도록 하는 것이 바람직합니다.

4) DB 서버

DB서버는 초기 프로그램 개발 시에 데이터 스키마를 어떻게 구성하는 가에 따라 크게 성능 차이가 납니다. 모든 프로그램 상의 DB 내용을 하나의 DB 파일에 여러 개의 테이블로 나누어 개발하게 되면 운영 시 성능이나 시스템 확장 시 여러 가지 문제가 발생 할 수 있습니다. 프로그램의 종류에 따라 데이터베이스 파일을 나누어서 DB를 구성하는 것이 효율적입니다.

6.1.2 인터넷 웹 서버의 서비스 별 시스템 구성



위 그림은 국내 유명 경매 사이트의 클러스터 구축 시스템 구성도 입니다.
앞에서 소개한 웹 서버 군을 서비스 별로 다시 분리하면 DNS, Mail. Web, FTP, DB, File 서비스 등으로 나눌 수 있습니다.

Web 서비스도 그 성격에 따라서 정적인 웹 서비스(일반 html 페이지), 와 동적인 웹 서비스 (WAS) 로 나누어 지고, 여기서 정적인 웹 서비스는 text 기반과 image 기반으로 세분화 되어집니다.

위 구성도를 보면 먼저 Firewall 하단에 부하 분산 시스템이 구성되어져 있습니다.
LB1 서버에서는 DNS 서버와 메일 서버, 그리고 정적인 웹 서버(S_WEB)를 제어하고, 다른 LB2 서버에서는 WAS 서버 역할을 하는 동적인 웹 서버(D_WEB) 로 구성되어 집니다.

정적인 웹 서버 중 S_WEB3은 이미지 전용 서버로 역할이 나누어져 있는데, 이는 특별히 이미지를 많이 사용하는 사이트에서는 이미지 로딩으로 인해 네트워크 손실을 막기 위해 추가 구성 하는 방식입니다.

Backbone Switch 하단에 있는 웹 서버 군은 공인 IP와 사설 IP 두 개의 네트워크 대역을 가지는데, 사설 IP는 Sub Switch의 DB서버와 NAS 서버와의 직접 통신에 이용되어 집니다.

일반 데이터 자료는 NAS 서버에만 존재하면 되는데, 이를 구현하는 방법으로 주로 NFS 를 많이 이용하고 있습니다.

이와 같이 분산 시스템에서는 DNS에서 서비스 별로 적절한 호스트를 나누어 주고, 프로그램 상에서 URL 절대경로 등을 사용함으로 실제 서비스 상에서는 하나의 대용량 메인 프레임 서버와 같은 성능을 나타낼 수 있습니다. (경우에 따라서는 더 나은 성능이 나타납니다. -> 서비스 별로 여러 개의 네트워크 대역을 나누기 때문에 시스템의 네트워크 인터페이스 자체의 네트워크 제한에서 오는 손실을 상당 수 방지할 수 있습니다.)

위 구성은 서비스 별로 시스템을 분산 시켜 클러스터 시키는 일반적인 방식입니다. 클러스터 관리 서버, SMS, NMS, IDS 서버 등을 추가 하여, 이 보다 더 세부적으로 분산 시킬 수도 있고, 위 구성에서 나누어진 서비스들을 몇 개로 통합하여 더 단순한 구성을 만들 수도 있습니다.

앞에서 언급한 바와 같이 단독 시스템에서 분산 시스템으로 시스템을 재 구성할 때는 현재 시스템의 서비스 별 자원 할당량을 파악해야 하고, 적절히 서비스를 그룹화 함으로 해서 이 후 일부 서비스의 성능 저하 시 해당 서비스에 Node 만 병렬적으로 추가 함으로 해결할 수 있고, 새로운 서비스를 Open 시 에도 새로운 서비스 그룹을 만들어 LB에 서비스 그룹 추가함으로, 기존 시스템 구성에 쉽게 적용할 수 있어야 합니다.

그리고 일부 서비스가 중지 되더라도 전체 서비스 중지가 되지 않도록 구성해야 합니다.

6.2. 대규모 인터넷 사이트 구축 설계

(별첨 참조)

Introduction to

Making High Availability Linux System

Using DRBD

(고가용성 클러스터 구성)

클루닉스 기술부 4차 세미나 자료

- 차례 -

클루닉스 서 진우(alang@clunix.com)

1. DRBD 를 이용한 HA 시스템 구축 준비

1.1 DRBD 최신 버전 다운 받기
1.2 커널 환경 준비 하기

2. DRBD 구축 하기

2.1 DRBD 설치 하기
2.2 DRBD 설정 하기
2.3 DRBD status 수동 전환하기
2.4 DRBD Multi Instance 구성하기

3. Heartbeat 구축 하기

3.1 Heartbeat 설치 하기
3.2 Heartbeat 설정 하기
3.3 Multi Heartbeat 구성하기

4. Heartbeat 와 DRBD 를 이용한 File System HA 하기

5. Apache + DRBD + Heartbeat 를 이용한 고가용성 웹서버 구축하기

6. Mysql + DRBD + Heartbeat 를 이용한 고가용성 Mysql DB 서버 구축하기

7. Oracle + DRBD + Heartbeat 로 고가용성 Oracle DB 서버 구축하기

7.1 DRBD 를 이용한 Oracle HA 구축 시 고려 사항
7.2 Oracle File Data Structure 설계 시 고려 사항
7.3 Oracle HA Migration 을 위한 Oracle File Data Control 하기
7.4 Oracle 과 Webprogram 연동하기
7.5 Oracle HA 에 필요한 Scripts
7.6 DRBD + Heartbeat 를 이용한 Oracle HA 구축하기
7.7 두 개의 Instance 를 이용한 Oracle Both HA 구축하기

1. DRBD 를 이용한 HA 시스템 구축 준비

1.1 DRBD 최신버전 다운 받기

http://oss.linbit.com/drbd/

당시 최신 버전 drbd-0.7.5.tar.gz 를 받는다

;; drbd 는 high availabilty cluster 를 구현하는데 필요한 network block device 를 제공하는 kernel modules 과 scripts 로 구성되어져 있다. network 를 통한 block device 를 통해 remote disk 간의 mirroring 기능을 제공한다. 그러므로 현 시스템의 kernel version 과의 의존성이 크다. 설치 되는 Makefile 에서 현 시스템의 kernel version 과 modules version 을 check 하게 되는데 Redhat 기본 커널에서 잘 안되는 경우가 발생한다.

original kernel source version 2.4.27 로 테스트를 했다.

1.2 커널 환경 준비 하기

먼저 linux-2.4.27.tar.bz2 를 /usr/src 밑에 놓고 kernel compile 를 한다.

커널 컴파일 시 꼭 포함해 줘야 할 것은

커널 모듈 인식

CONFIG_MODULES=y
CONFIG_MODVERSIONS=y
CONFIG_KMOD=y

NBD 모듈 인식

CONFIG_BLK_DEV_LOOP=y
CONFIG_BLK_DEV_NBD=y
CONFIG_BLK_DEV_RAM=y
CONFIG_BLK_DEV_RAM_SIZE=4096
CONFIG_BLK_DEV_INITRD=y
CONFIG_BLK_STATS=y

reiserfs 파일 시스템 인식

CONFIG_REISERFS_FS=y

정도 이다.

cd /usr/src/linux

make dep && make clean && make bzImage && make modules && make modules_install
cp System.map /boot/System.map-2.4.27
cp arch/i386/boot/bzImage /boot/vmlinuz-2.4.27
cd /boot
ln -sf System.map-2.4.27 System.map
ln -sf vmlinuz-2.4.27 vmlinuz

cd /lib/modules
depmod -a 2.4.27

/etc/lilo.conf 에 new building kernel 을 인식할수 있도록 적용 후 reboot

---

2. DRBD 구축 하기

2.1 DRBD 설치 하기

먼저 /usr/src 및에 drbd source 를 copy 해 둔다.

```
# cp drbd-0.7.5.tar.gz /usr/src
# tar xzvf drbd-0.7.5.tar.gz
# cd drbd-0.7.5
# cd drbd
# make clean all
# cd ../user
# make
# cd ..
# make all
# make install
```

이로써 drbd 컴파일이 완료된다. 아무런 error 없이 컴파일이 되었으면 정상적으로 drbd modules 생성되었는지 확인한다.

```
# ls -al /lib/modules/2.4.27/kernel/drivers/block/drbd.o
/lib/modules/2.4.27/kernel/drivers/block/drbd.o
```

그런후 drbd device 를 생성한다.

```
# vi mkdrbddev
-----------------------------------------------------------------
#!/bin/sh

for i in $(seq 0 15)
do mknod /dev/drbd$i b 147 $i
done

-----------------------------------------------------------------

# sh mkdrbddev
# ls -al /dev/drbd0
brw-r--r--      1 root       root       147,     0 Oct 31 15:47 /dev/drbd0
```

이로써 설치가 완료 되었다. 이제 지금껏 생성한 DRBD 장치를 이용하여 설정을 해보도록 하자.

먼저 DRBD 시스템 설정 환경을 정하도록 하자.

master node :

hostname : test01
ip address : 192.168.123.165

slave node :

hostname : test02
ip address : 192.168.123.166

2.2. DRBD 설정하기

vi /etc/drb3. drbd.conf 생성하기

vi /etc/drbd.conf

```
---------------------------------------------------------------------------

skip {
  As you can see, you can also comment chunks of text
  with a 'skip[optional nonsense]{ skipped text }' section.
  This comes in handy, if you just want to comment out
  some 'resource <some name> {...}' section:
  just precede it with 'skip'.

  The basic format of option assignment is
  <option name><linear whitespace><value>;

  It should be obvious from the examples below,
  but if you really care to know the details:

  <option name> :=
        valid options in the respective scope
  <value>   := <num>|<string>|<choice>|...
               depending on the set of allowed values
               for the respective option.
  <num>     := [0-9]+, sometimes with an optional suffix of K,M,G
```

```
  <string> := (<name>|₩"([^₩"₩₩₩n]*|₩₩.)*₩")+
  <name>   := [/_.A-Za-z0-9-]+
}


resource drbd0 {
  protocol C;
  incon-degr-cmd "echo '!DRBD! pri on incon-degr' | wall ; sleep 60 ; halt -f";
  startup {
    degr-wfc-timeout 120;
  }

  disk {
    on-io-error     detach;
  }

  net {
    # sndbuf-size 512k;
    # timeout           60;     #  6 seconds  (unit = 0.1 seconds)
    # connect-int     10;     # 10 seconds  (unit = 1 second)
    # ping-int        10;     # 10 seconds  (unit = 1 second)
    # max-buffers       2048;
    # max-epoch-size   2048;
    # ko-count 4;
    # on-disconnect reconnect;

  }

  syncer {
    rate 10M;
    group 1;
    al-extents 257;
  }

  on test01 {
    device      /dev/drbd0;
    disk        /dev/sda7;
    address     192.168.123.165:7788;
    meta-disk   internal;
```

```
  }

  on test02 {
    device     /dev/drbd0;
    disk       /dev/sda7;
    address    192.168.123.166:7788;
    meta-disk internal;
  }
}
```

위의 drbd.conf 파일을 test01, test02 의 /etc 에 copy 한다.

test01, test02 Node 에서 아래 command 를 실행한다.

;; test01

```
# modprobe drbd; drbdadm up all; dmesg | tail; cat /proc/drbd
```

;; test02

```
# modprobe drbd; drbdadm up all; dmesg | tail; cat /proc/drbd
```

```
drbd: SVN Revision: 1578 build by root@test02, 2004-10-31 15:47:05
drbd: registered as block device major 147
drbd0: resync bitmap: bits=5535754 words=172994
drbd0: size = 21 GB (22143016 KB)
drbd0: 17 GB marked out-of-sync by on disk bit-map.
drbd0: Found 2 transactions (2 active extents) in activity log.
drbd0: Marked additional 1024 KB as out-of-sync based on AL.
drbd0: drbdsetup [1364]: cstate Unconfigured --> StandAlone
drbd0: drbdsetup [1366]: cstate StandAlone --> Unconnected
drbd0: drbd0_receiver [1367]: cstate Unconnected --> WFConnection
version: 0.7.5 (api:76/proto:74)
SVN Revision: 1578 build by root@test02, 2004-10-31 15:47:05
 0: cs:WFConnection st:Secondary/Unknown ld:Inconsistent
    ns:0 nr:0 dw:0 dr:0 al:0 bm:2 lo:0 pe:0 ua:0 ap:0
```

test01, test02 Node 에서 차례로 drbd 데몬을 실행한다.

;; test01

```
# /etc/rc.d/init.d/drbd start

 DRBD's startup script waits for the peer node(s) to appear.
 - In case this node was already a degraded cluster before the
    reboot the timeout is 120 seconds. [degr-wfc-timeout]
 - If the peer was available before the reboot the timeout will
    expire after 0 seconds. [wfc-timeout]
    (These values are for resource 'drbd0'; 0 sec -> wait forever)
 To abort waiting enter 'yes' [     8]: To abort waiting enter 'yes' [ -- ]:
```

위의 메세지는 test02 노드의 drbd 데몬이 실행이 안되어서 통신이 안되기 때문에 test02 노드가 drbd 통신을 할때까지 대기 하고 있다는 메세지다. test02 의 drbd 데몬을 실행하면 자동으로 없어진다.

;; test02

```
# /etc/rc.d/init.d/drbd start
```

이제 drbd 가 정상적으로 작동하는지 확인 하자

```
[root@test01 root]# cat /proc/drbd
version: 0.7.5 (api:76/proto:74)
SVN Revision: 1578 build by root@test01, 2004-10-31 16:13:21
 0: cs:SyncSource st:Secondary/Secondary ld:Consistent
    ns:824336 nr:0 dw:0 dr:825060 al:0 bm:51 lo:74 pe:36 ua:181 ap:0
        [>.....................] sync'ed:   4.6% (16891/17696)M
        finish: 0:42:23 speed: 6,716 (5,720) K/sec

[root@test02 root]# cat /proc/drbd
version: 0.7.5 (api:76/proto:74)
SVN Revision: 1578 build by root@test02, 2004-10-31 15:47:05
 0: cs:SyncTarget st:Secondary/Secondary ld:Inconsistent
    ns:0 nr:1016336 dw:1016336 dr:0 al:0 bm:65 lo:6 pe:256 ua:6 ap:0
```

```
        [=>..................] sync'ed:   5.7% (16703/17696)M
        finish: 0:35:38 speed: 7,996 (5,840) K/sec
```

현재 두 Node 모두 Secondary 상태로 실행되고 있다는 것을 알수 있다.
이제 test01 노드를 Primary 로 전환해 보도록 하자.

;; test01

```
# drbdadm primary all
```

만일 아래와 같은 메세지가 출력 되면

```
ioctl(,SET_STATE,) failed: Input/output error
Local replica is inconsistent (--do-what-I-say ?)
```

```
# drbdadm -- --do-what-I-say primary all
```

같이 실행하면 된다.
다시 확인 해보도록 하자.

```
[root@test01 root]# cat /proc/drbd
version: 0.7.5 (api:76/proto:74)
SVN Revision: 1578 build by root@test01, 2004-10-31 16:13:21
 0: cs:SyncSource st:Primary/Secondary ld:Consistent
    ns:2204520 nr:0 dw:0 dr:2205544 al:0 bm:136 lo:256 pe:0 ua:256 ap:0
        [==>..................] sync'ed: 12.3% (15543/17696)M
        finish: 11:03:10 speed: 36 (5,696) K/sec
```

;; test02

```
[root@test02 root]# cat /proc/drbd
version: 0.7.5 (api:76/proto:74)
SVN Revision: 1578 build by root@test02, 2004-10-31 15:47:05
 0: cs:SyncTarget st:Secondary/Primary ld:Inconsistent
    ns:0 nr:2263064 dw:2263064 dr:0 al:0 bm:141 lo:7 pe:256 ua:7 ap:0
        [==>..................] sync'ed: 12.6% (15486/17696)M
        finish: 0:40:39 speed: 6,404 (5,640) K/sec
```

이제 Primary Node 에 drbd 를 device 에 mount 를 시켜 보도록 하자.

;; test01

```
# mkreiserfs /dev/drbd0
# mkdir /data1
# mount /dev/drbd0 /data1
```

-> /dev/drbd0 device 에 파일 시스템을 재생성 할때 물리적 device 가 format 되어 버린다.

이제 test01 에 /data1 에서 발생하는 모든 file system 의 writing 작업은 drbd dev 를 통해서 test02 의 /dev/sda7 에도 일어나게 된다.

2.3. DRBD 수동 전환 방식

위의 과정으로 DRBD 실제 셋팅이 완료 되었다. 이제 실제 노드에 failed status 를 가정할때 Node 전환 방식을 수동으로 확인해 보자. 실제 HA 프로그램을 이용하여 이런 일련의 과정을 자동화 할수 있을 것이다.

- Primary/Secondary Switching

test01 Node 를 primary , test02 Node 를 secondary 로 가정. test01, test02의 역활을 수동으로 전환 하는 방식이다.

;; test01

```
[root@test01 root]# umount /data1
[root@test01 root]# /etc/rc.d/init.d/drbd stop
```

- 즉 test01 node 가 정상 작동 안하는 것을 가정한다.

;; test02

```
[root@test02 /]# cat /proc/drbd
version: 0.7.5 (api:76/proto:74)
SVN Revision: 1578 build by root@test02, 2004-10-31 15:47:05
 0: cs:WFConnection st:Secondary/Unknown ld:Consistent
      ns:0 nr:696 dw:696 dr:0 al:0 bm:0 lo:0 pe:0 ua:0 ap:0
```

test02 상태를 보면 현재 Secondary status 이고 primary 상태의 노드는 인식할 수 없는 상태이다.

test02 를 primary 로 전환 한다.

```
[root@test02 /]# drbdadm primary all

[root@test02 /]# cat /proc/drbd
version: 0.7.5 (api:76/proto:74)
SVN Revision: 1578 build by root@test02, 2004-10-31 15:47:05
 0: cs:WFConnection st:Primary/Unknown ld:Consistent
      ns:0 nr:696 dw:696 dr:0 al:0 bm:0 lo:0 pe:0 ua:0 ap:0

[root@test02 /]# mount /dev/drbd0 /data1

[root@test02 /]# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/sda2              2016044    149888   1763744   8% /
/dev/sda1               202220     17642    174138  10% /boot
none                    515760         0    515760   0% /dev/shm
/dev/sda3              7052496   2001624   4692624  30% /usr
/dev/sda6              2016016    242608   1670996  13% /var
/dev/drbd0            22142336     67440  22074896   1% /data1
```

- test02 node 는 test01 node 를 주기적으로 checking 하다 문제 발생 시 위 일련의 가정을 거쳐서 Secondary 상태에서 Primary 상태로 전환되어 파일 서비스를 진행하게 된다.

이제 test01 node 가 다시 살아 나서 정상적인 역활 상태로 전환 한다고 가정해 보자

;; test01

```
[root@test01 root]# /etc/rc.d/init.d/drbd start

[root@test01 root]# cat /proc/drbd
version: 0.7.5 (api:76/proto:74)
SVN Revision: 1578 build by root@test01, 2004-10-31 16:13:21
 0: cs:Connected st:Secondary/Primary ld:Consistent
    ns:0 nr:40 dw:40 dr:0 al:0 bm:3 lo:0 pe:0 ua:0 ap:0
```

test01 node 가 살아 나게 되면 drbd status 가 Secondary 상태로 있게 된다.
이를 다시 Primary 로 전환 한다.

```
;; test02

[root@test02 /]# umount /data1
[root@test02 /]# drbdadm secondary all
[root@test02 /]# cat /proc/drbd
version: 0.7.5 (api:76/proto:74)
SVN Revision: 1578 build by root@test02, 2004-10-31 15:47:05
 0: cs:Connected st:Secondary/Primary ld:Consistent
    ns:80 nr:696 dw:780 dr:848 al:0 bm:8 lo:0 pe:0 ua:0 ap:0

;; test01

[root@test01 root]# drbdadm primary all
[root@test01 root]# cat /proc/drbd
version: 0.7.5 (api:76/proto:74)
SVN Revision: 1578 build by root@test01, 2004-10-31 16:13:21
 0: cs:Connected st:Primary/Secondary ld:Consistent
    ns:0 nr:80 dw:80 dr:0 al:0 bm:3 lo:0 pe:0 ua:0 ap:0
[root@test01 root]# mount /dev/drbd0 /data1
```

2.4 DRBD Multi Instance 구성하기

/etc/drbd.conf 를 아래와 같이 구성한다.

```
---------------------------------------------------------------------
resource drbd0 {
  protocol C;
  incon-degr-cmd "echo '!DRBD! pri on incon-degr' | wall ; sleep 60 ; halt -f";
  startup {
    degr-wfc-timeout 120;      # 2 minutes.
  }


  disk {
    on-io-error     detach;
  }

  net {

  }

  syncer {
    rate 10M;
    group 0;
    al-extents 257;
  }

  on test01 {
    device        /dev/drbd0;
    disk          /dev/sda7;
    address       192.168.123.165:7788;
    meta-disk     internal;
  }

  on test02 {
    device       /dev/drbd0;
    disk         /dev/sda7;
    address      192.168.123.166:7788;
    meta-disk internal;
  }

resource drbd1 {
```

```
  protocol C;
  incon-degr-cmd "echo '!DRBD! pri on incon-degr' | wall ; sleep 60 ; halt -f";
  startup {
    degr-wfc-timeout 120;      # 2 minutes.
  }


  disk {
    on-io-error     detach;
  }

  net {
  }

  syncer {
    rate 10M;
    group 0;
    al-extents 257;
  }

  on test01 {
    device       /dev/drbd1;
    disk         /dev/sda8;
    address      192.168.123.165:7799;
    meta-disk    internal;

  }

  on test02 {
    device      /dev/drbd1;
    disk        /dev/sda8;
    address     192.168.123.166:7799;
    meta-disk internal;
  }

}
```

drbd0, drbd1 두개의 resource 설정을 한후 /etc/rc.d/init.d/drbd start
를 해주면 된다.

각 노드에 대한 primary, secondary 설정은 해당 노드의 정책에 맞게 구성해 주면 된다. 즉.
.

	test01	test02
drbd0	primary	secondary
drbd1	secondary	primary

와 같은 구성을 위해서는..

;; test01

drbdadmin primary drbd0

;; test02

drbdadmin primary drbd1

명령을 해주면 된다.

3. Heartbeat 구축 하기

3.1 Heartbeat 설치 하기

http://www.linux-ha.org

test01, test02 node 에 heartbeat package 를 설치 한다.

heartbeat-stonith-1.2.2-8.rh.9.i386.rpm
heartbeat-pils-1.2.2-8.rh.9.i386.rpm
heartbeat-1.2.2-8.rh.9.i386.rpm

[root@test01 src]# rpm -Uvh heartbeat-*

3.2 Heartbeat 설정 하기

heartbeat 의 설정 파일은 크게 3개로 나누어 진다. 이 3개의 설정파일의 설정만 완료 되면 정상적인 HA 가 이루어 진다.

ha.cf -> ha 의 기본 환경 설정 파일이다.
authkeys -> ha 노드간의 인증 방식을 지정하는 파일이다.
haresources -> heartbeat 구성에서 vip 와 실제 node failed 시 실행 스크립트를 지정하는 설정 파일이다.

[root@test01 src]# cd /etc/ha.d

[root@test01 ha.d]# vi ha.cf
---

debugfile /var/log/ha-debug
logfile /var/log/ha-log
logfacility local0
keepalive 2

```
deadtime 5
hopfudge 1
udpport 1001
udp     eth0
node    test01
node    test02
```

```
[root@test01 ha.d]# vi authkeys
```

```
auth 1
1 sha1 HI!
```

```
[root@test01 ha.d]# chmod 600 authkeys
```

;; 반드스 인증 파일은 퍼미션 600 으로 해주어야 한다.

```
[root@test01 ha.d]# vi haresources
```

```
test01 192.168.123.170 mysqld
```

---

haresources 설정 파일의 형식은 다음과 같다.

<< mater node >> << vip >> << init scripts >>

3.3 Multi Heartbeat 구성 하기

2.4장에서 multipul Drbd instance 구성하기에 대해 잠시 알아 보았다. 여기서는 multipul instance 로 구성된 DRBD 를 Heartbeat 에서도 dual 로 구성하여 사용할수 있는 방법을 알아보도록 하자.

Heartbeat 를 multi 로 구성하는 방법은 아주 간단하다.

[root@test01 root]# vi /etc/ha.d/haresources

---

test01 192.168.123.170 drbddisk::drbd0 Filesystem::/dev/drbd0::/data1::reiserfs mysqld

- 아래 추가 -

test02 192.168.123.171 drbddisk::drbd1 Filesystem::/dev/drbd1::/data2::reiserfs mysqld

---

와 같이 해주면 된다.

이제 5장과 이 단원의 방법으로 both HA on both Drbd System 이 구축이 되는 것이다.
간단히 위 시스템 설게에 대해 설명하면..

test01, test02 시스템에 각각 두개의 data volum (data1, data2) 이 존재하고 이 두개의 volum 을 drbd0, drbd1 로 구성한다.

그런 후 아래와 같이 각 시스템의 drbd 권한을 준다.

test01 ->

drbd0(/data1) - primary

drbd1(/data2) - secondary

test02 ->

drbd0(/data1) - secondary

drbd1(/data2) - primary

그런후 mysql datafile path 를 아래와 같이 구성하여 시스템을 작동시킨다.

test01 ->

/data1 -> mysql datafile of test01 node

test02 ->

/data2 -> mysql datafile of test02 node

이제 시스템을 test01 에 /data1 에 datafile 을 둔 mysql 이, test02 에 /data2 에 datafile 을 둔 mysql 이 각각 작동하면서 test01 에서는 /data2 에 test02 의 mysql datafile 을 real time volum mirroring 을 되어지고, test02 에서는 /data1 에 test01 의 mysql datafile 이 real time volum mirroring 이 되어진다.

test01 노드에 이상이 생기면 test02 에서 자동으로 /data1 을 mount 시키며
test01 에서의 mysql 서비스가 구동되어진다.

test02 노드에 이상이 생기면 test01 에서 자동으로 /data2 을 mount 시키며
test02 의 mysql 서비스가 지속적으로 구동이 되어지는 것이다.

주의 할점은 위의 구성으로 시스템을 구성하기 위해서 mysql 의 기본 instance
가 두개가 될수 있는 설정을 해야 한다. --datefile --port 등의 옵션을 주고
컴파일을 해서 한 시스템에 두개의 mysql 을 동작 시켜야 할것이다.

이 방법의 각자의 엔지니어링 역량에 맡기도록 하겠다.

4. Heartbeat 와 DRBD 를 이용한 File System HA 하기

test01, test02 node 에 heartbeat package 를 설치 한다.

heartbeat-stonith-1.2.2-8.rh.9.i386.rpm
heartbeat-pils-1.2.2-8.rh.9.i386.rpm
heartbeat-1.2.2-8.rh.9.i386.rpm

[root@test01 src]# rpm -Uvh heartbeat-*

[root@test01 src]# cd /etc/ha.d

[root@test01 ha.d]# vi ha.cf
---

debugfile /var/log/ha-debug
logfile /var/log/ha-log
logfacility local0
keepalive 2

```
deadtime 5
hopfudge 1
udpport 1001
udp        eth0
node       test01
node       test02
```

```
[root@test01 ha.d]# vi authkeys
```

```
auth 1
1 sha1 HI!
```

```
[root@test01 ha.d]# chmod 600 authkeys
```

```
[root@test01 ha.d]# vi haresources
```

```
test01 192.168.123.170 drbddisk Filesystem::/dev/drbd0::/data1::reiserfs mysqld
```

위 설정을 test02 에도 동일하게 해준다. 그냥 copy 해도 된다.
haresources 설정을 간단히 설명하면 다음과 같다.

test01 -> ha active node
192.168.123.170 -> service ip address

Filesystem::/dev/drbd0::/data1::reiserfs ->

/etc/ha.d/resource.d/Filesystem 이란 scripts 가 존재한다.
이 스크립트의 역할은 /dev/drbd0 device 를 mount point 에 mount 시켜 주는 역할을 한다.

사용 방법은 ./Filesystem <device> <directory> <fstype> {start|stop|status}

이다.

drbd 에서는 "./Filesystem /dev/drbd0 /data1 reiserfs start"
하면 drbd device 에 mount 가 되는 것이다.
"./Filesystem /dev/drbd0 /data1 reiserfs stop" 하면 drbd device 에 mount 된 경로
가 umount 가 되어진다.

drbddisk ->

drbddisk scripts 는 실제 drbd 의 역할을 drbd node 간의 역할을 정하는 것이다.
drbd Node 는 각각 primary 와 secondary 로 각각의 역할이 있다. 만일 특정 노드에서
"drbddisk start" 를 하면 이 node 가 primary 로 존재하게 된다. stop 시는 secondary
로 존재하게 된다.

이제 drbd 의 역할 전환 수동 방식에 대해서는 위에서 자세히 설명하였다.
단계를 보면 ..

A. umount test01 drbd filesystem
B. apply test01 "drbdadm secondary all"
C. mount test02 drbd filesystem mounting
D. applay test02 "drbdadmin primary all"

위와 같다.

heartbeat 의 working logic 과 위 과정을 자동화 해줄수 있는 scripts 로 drbd 를
ha 할수 있다.

정확한 작동을 위해서 ha 에 필요한 scripts 를 /etc/rc.d/init.d 밑에 copy 한다.

[root@test01 rc3.d]# cd /etc/ha.d/resource.d/
[root@test01 resource.d]# cp Filesystem drbddisk /etc/rc.d/init.d/

node reboot 시에도 이상없이 작동 할 수 있게 heartbeat 와 drbd 를 booting 시 자동으로
실행하게 설정한다.

[root@test01 rc3.d]# cd /etc/rc.d/rc3.d/
[root@test01 rc3.d]# ln -s /etc/rc.d/init.d/drbd S70drbd

이로써 DRBD + heartbeat 를 이용한 클러스터 파일 시스템 환경이 준비가 되었다.

5. Apache + DRBD + Heartbeat 를 이용한 고가용성 웹서버 구축하기

test01, test02 Node 에 apache + mysql + php + tomcat 을 설치한다.
설치 관련 자료는 클루닉스 기술부 2차 세미나 자료를 참고 한다.

httpd.conf 에서 ServerName , NameVirtualHost 를 Heartbeat 의
haresources 에 정의된 service ip address 로 지정한다.

DocumentRoot 는 drbd device 에 mount 된 /data1 아래에 생성하도록 한다.

test01, test02 node 의 apache init script 를 /etc/rc.d/init.d 밑에 복사한다.

[root@test01 root]# cp /usr/local/apache/bin/apachectl /etc/rc.d/init.d/
[root@test02 root]# cp /usr/local/apache/bin/apachectl /etc/rc.d/init.d/

heartbeat 의 haresources 설정에 apache init script 를 추가한다.

[root@test01 root]# cd /etc/ha.d
[root@test01 ha.d]# vi haresources

---

test01 192.168.123.170 drbddisk Filesystem::/dev/drbd0::/data1::reiserfs ₩
 mysqld apachectl

---

haresources 를 test02 node 에도 copy 한다.

test01, test02 node 의 heartbeat 를 다시 시작한다.

6. Mysql + DRBD + Heartbeat 를 이용한 고가용성 Mysql DB 서버 구축하기

mysql 을 DRBD 에 적용하는 방법에 대해 간단히 설명한다.

일단 mysql 을 클루닉스 기술부 2차 세미나 방식으로 설치 했을 경우에는 mysql db file 이 /usr/local/mysql/data 밑에 설치 되게 된다. 이 db file path를 DRBD 가 적용되는 /data1 밑에 path 로 변경해 주어야 한다.

먼저 실행 중인 mysql daemon 을 stop 시킨다.

```
[root@test01 data1]# vi /etc/rc.d/init.d/mysqld stop
```

기존에 생성된 기본 DB file 을 모두 DRBD 가 적용된 volum 으로 Copy 한다.

```
[root@test01 data1]# cp -a /usr/local/mysql/data /data1/mysqldb
[root@test01 data1]# chown mysql. mysqldb
```

그런후 mysql init script에서 --datadir 의 부분을 변경해 준다.

```
[root@test01 data1]# vi /etc/rc.d/init.d/mysqld
---------------------------------------------------------------------------

...

/// --datadir 로 해당 문자열을 찾는다 //

...

$bindir/mysqld_safe --datadir=/data1/mysqldb --pid-file=$pid_file ₩
 --language=korean --safe-show-database >/dev/null 2>&1 &

..
---------------------------------------------------------------------------
```

```
[root@test01 data1]# vi /etc/rc.d/init.d/mysqld start
```

이로써 DRBD 를 이용한 mysql database HA 가 구현된다.

7. Oracle + DRBD + Heartbeat 로 고가용성 Oracle DB 서버 구축하기

7.1 DRBD 를 이용한 HA 구성으로 Oracle 설치 시 유의할 점

oracle 을 DRBD 로 HA 설계를 하기 위해서는 몇 가지 초기 Oracle 시스템 설계 시 고려 해야 할 사항이 있다. 오라클 DB 의 구동 방식은 Mysql 가 같은 file 위주 형식의 작동 방식이 아니다.

즉 시스템의 file data 에 관련된 설정과 oracle 내부 엔진에 관련된 설정들이 자체 dba db tables 에서 관리 하도록 구성이 되어져 있기 때문에 DB 의 주요 구성 요소의 정보를 HA 백업을 시킬 노드에도 동일하게 적용이 되어져 있어야 한다.

만일 실제 DB file 의 내용과 시스템의 구조, 오라클의 내부 구성등의 정보가 하나라도 일치 하지 않으면 정상적으로 oracle daemon 을 start 시킬 수 없다.

이렇기에 오라클을 설치 시 control file, db file, log file, paramate file 등의 경로 및 정보들을 모두 DRBD 가 작동하는 volum 에 위치 시켜야 한다.

오라클 DB 의 File structure 에 대해 이해만 하고 있으면 초기 설치 시에 HA에 필요한 file 의 경로를 DRBD Volum 으로 target 을 정해 생성할 수 있다.

하지만 기존에 이미 만들어진 DB 인 경우는 수동으로 data path를 모두 수정해야
할것이다.

그럼 오라클 설정 시 필요한 파일에 대해 알아 보도록 하자

* control db file (default path : {ORACLE_BASE}/oradata)
* data db file (non system db, system db) (default path : {ORACLE_BASE}/oradata)
* redo log db file (default path : {ORACLE_BASE}/oradata)
* spfile<SID>.ora (default path : {ORACLE_HOME}/dbs)
* orapw<SID> (default path : {ORACLE_HOME}/dba)

위 DB 파일들은 초기 DB 생성 프로그램인 dbca 를 통해 DB 생성 시 위치를 정해 줄수도
있고 DB Create scripts 수정등을 통해서 변경 할 수도 있다.

spfile<SID>.ora 파일은 서버 초기화 파일로 db file 의 기본 정보 및 변경된 정보
기타 parameter 정보를 관리하는 파일이다. oracle 8i 에서는 pfile 이라 했는데
oracle 9i 부터 pfile 과 spfile 두가지 방식으로 관리가 가능하다.

orapw<SID> 파일은 오라클에 sysdba 권한으로 접속을 할 수 있는 계정을 관리 하는 파일
이다. 만일 이 문제로 oracle db mount 시 error 가 발생 할때는 그냥 primary 시스템의
orapw<SID> 파일을 옮겨 그대로 사용하면 된다.

위에서 언급한 내용으로 단순 active / standby 형식의 oracle HA 구성이 가능하다.
하지만 oracle multi instance 구성으로 두대의 노드가 각 instance 의 both HA 을
구성할때는 oracle 의 file architecture 뿐만 아니라 process, log architecture
즉 .. Oracle Instance Structure 에 대해 이해를 해야 한다.
이 문서는 Oracle 을 설명하는 문서가 아니니 oracle 에 관련된 사항들은 개인적인
노력이 필요할 것이다.

7.2 Oracle File Data Structure 설계 시 고려 해야 할 사항

Control File, Redo Log File, Data File
분류 시 고려해야 할 사항들

Control Files

- Control File은 크기가 작으므로 중요하므로 여러 디스크에 저장 되면 좋다.
- Control File은 디스크 마다 이름이 같은 것이 좋다. (추 후 문제시 복구 용이)

Redo Log Files

- 그룹 단위이며 한 그룹에는 적어도 2개 이상의 Redo Log File(Member)이 있어야 한다.
- 같은 그룹에 속한 Redo Log File 들은 다른 디스크에 저장되는 것이 좋다.
- Datafile 과 Redo Log File은 같은 디스크에 위치하지 않도록 한다.
  (복구 시 필요하므로 따로 둔다 - 같이 오류,에러시 복구 불가능)

Data Files

- System, Undo, Temporary 및 오라클 디폴트 제공 테이블 스페이스 영역은 같은 디스크에 존재하는 것이 좋다.
- System, Undo, Temporary 등 테이블 스페이스 영역은 오라클이 사용하므로 사용자 영역과 다른 디스크에 위치하는 것이 좋다.
- 같은 테이블 스페이스에 존재하는 Data File도 다른 디스크에 위치하게 만드는 것이 좋다.
- Datafile 과 Redo Log File은 같은 디스크에 위치하지 않도록 한다.
  (복구시 필요하므로 따로 둔다 - 같이 오류, 에러시 복구 불가능)

예) disk1 ~ disk5까지의 디스크가 존재한다.

각각의 디스크에 Control File, Reod Log File, Data File을 배치하라.

단, 사용자 Tablespace는 happy 한 개 존재한다.

DISK 1

---

control.ctl

happy01.dbf

DISK 2

---

control.ctl

happy02.dbf

DISK 3

---

control.ctl

system01.dbf, undotbs01.dbf, temp01.dbf, cwlite01.dbf, drsys01.dbf,

example01.dbf, indx01.dbf, odm01.dbf, tools01.dbf, users01.dbf, xdb01.dbf

---

DISK 4

---

control.ctl

redo01.log, redo02.log, redo03.log

DISK 5

---

control.ctl

redo01.log, redo02.log, redo03.log

오라클 성능을 위해서는 위 사항을 충분히 고려 해서 설계하는 것이 좋을 것이다. 하지만 HA 설계 시에는 Data File Managing Point 가 많아 진다는 단점이 있을 것 이다. 이는 성능과 관리의 상반된 관점에서의 설계자가 결정해야 할 문제라 생각한다.

7.3 Oracle HA Migration 을 위한 Oracle File Data Control 하기

Oracle Install 단계에서 미리 HA를 고려한 DB Structure 로 구성을 했다면 필요 없겠지만.. 이미 서비스를 하고 있는 Oracle 을 HA 구성으로 변경할 경우 반드시 알아야 하는 기본 Oracle DBA 기술이기에 간단히 설명하겠다.

기본적으로 오라클 DB 를 생성했다면 {ORACLE_BASE}/oradata 밑에 아래와 같은 DB file 이 있을 것이다.

---------- control file

control01.ctl

control02.ctl

control03.ctl

---------- data file

----------non-system data file

cwmlite01.dbf

drsys01.dbf

example01.dbf

indx01.dbf

odm01.dbf

tools01.dbf

users01.dbf

xdb01.dbf

oem_repository.dbf

----------system data file

system01.dbf

undotbs01.dbf

----------temporary data file

temp01.dbf

---------- log file

redo01.log

redo02.log

redo03.log

s----------

여기에는 크게 control file, data file, log file 로 분류를 할수 있다.
여기서 data file 의 경우에는 다시 system data file, non-system data file,
temporary data file 로 나눌 수 있다.

각각의 파일 별로 데이터 경로를 변경하는 방법이 조금씩 다르다.

+ Control File Data Path Modify

기본적인 Control file 위치는 초기화 파라메터 파일 에서 확인 할수 있다.

$ORACLE_HOME/dbs/init<SID>.ora 파일 안에 아래과 같이 존재

```
control_files=("/oracle/oradata/disk3/control.ctl",
               "/oracle/oradata/disk4/control.ctl",
               "/oracle/oradata/disk5/control.ctl")
```

Oracle 9i 에서는 기본적으로 init<SID>.ora 파일이 없을 것이다.
초기 설치 시에는 관련 정보가 모두 server parameter 파일인 spfile<SID>.ora 에 저장이
되기 때문에 별도로 초기화 파라메터 파일을 생성해야 한다.

생성 구문은 다음과 같다.

$ sqlplus /nolog
SQL> conn / as sysdba
SQL> create pfile from spfile;
SQL> shutdown normal

그럼 {ORACLE_HOME}/dbs 밑에 init<SID>.ora 파일이 생성이 되어져 있을 것이다.
이제 각 control file 의 현재 위치가 확인 되면 옮겨 보도록 하자

먼저 init<SID>.ora 파일의 열어서 control_file 의 control file 경로를 옮길 위치로
변경을 한다.

그런 후 기존의 control0X.ctl file 을 OS 상에서 물리적으로 옮길 경로로 copy 한다.

그런 후 DB 를 변경된 pfile 로 start 시킨다.

$ sqlplus /nolog
SQL> conn / as sysdba
SQL> startup pfile='{ORACLE_HOME}/dbs/init<SID>.ora'

-- 수정된 정보 확인

SQL> col NAME format a35
SQL> select * from v$controlfile;

정상적으로 변경이 완료되면 기존 control file 을 삭제한다.

-- 변화된 정보를 spfile 에 새로 적용 시킨다.
기존 spfile 을 잠시 다른곳을 백업해 두고 지운다.

SQL> create spfile from pfile;

이로써 control file 변경이 완료된다.

+ DB DATA File Path Modify

DB DATA file 은 system file, non-system file, temp file 로 나누어진다.

-- system file 변경 방법

- Database가 Mount 단계이어야 한다.
- 이동하고자하는곳에 이동하고자 하는 Data File이 존재해야 한다.
- 4 번은 Control File을 갱신시키는 역할을 한다.
- GAD는 Global Area Database 명이다.

a. 오라클을 shutdown 시킨다.

b. startup mount상태로 만든다.

c. !mv $ORACLE_BASE/oradata/<GAD>/system01.dbf
$NEW_VOL/oradata/<GAD>/system01.dbf

d. ALTER DATABASE RENAME
FILE '$ORACLE_BASE/oradata/<SID명>/system01.dbf'
TO '$NEW_VOL/oradata/<SID명>/system01.dbf'

e. alter database open;

-- non system file 변경 방법

- Tablespace는 offline 이어야한다.
- 이동하고자하는곳에 이동하고자 하는 Data File이 존재해야 한다.
- 4 번은 Control File을 갱신시키는 역할을 한다.
- GAD는 Global Area Database 명이다.

a. startup 상태에서 수행한다.

b. alter tablespace <tablespace명> offline; (mount 상태과 같다)

c. SQL> !mv $ORACLE_BASE/oradata/<GAD>/userdata01.dbf
$NEW_VOL/oradata/<GAD>/userdata01.dbf

d. ALTER TABLESPACE userdata RENAME
DATAFILE '$ORACLE_BASE/oradata/<GAD>/userdata01.dbf'

TO '$NEW_VOL/oradata/<GAD>/userdata01.dbf';

e. alter tablespace <tablespace명> online;

+ Temporary File Path Modify

Temporary Tablespace 관리시 주의사항

오라클 설치 시의 temp 테이블 스페이스는 Default Temporary Tablespace는 삭제되지 않는다. 이동 할수도 없다. 그렇기에 새로운 Temporary 를 원하는 위치에 만들고, 새로운 Temporary tablespace 를 default temporary 로 지정한 후 기존적을 삭제해 버리면 된다.

Temporary Tablespace 생성

---

temp1라는 이름(논리적인 이름)으로 '$ORACLE_BASE/oradata/<SID명>/temp1.dbf' 라는 (물리적) 위치에 생성 할 경우

-> 기본적으로 오라클 생성 시 temp 라는 이름으로 기본 temporary 가 생성 됨..

```
CREATE TEMPORARY TABLESPACE temp1
TEMPFILE '$NEW_VOL/oradata/<SID명>/temp1.dbf' SIZE 20M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 4M;
```

Default Tempory Tablespace로 바꾸기

---

temp1 라는 Temporary Tablespace를 생성 후 다음을 실행한다.

```
ALTER DATABASE
DEFAULT TEMPORARY TABLESPACE temp1;
```

Temporary Tablespace 삭제

```
DROP TABLESPACE temp
INCLUDING CONTENTS AND DATAFILES;
```

+ Log File Path Modify

Redo Log File 이름 바꾸기 및 이동 시키기

---

a. Database를 shutdown 한다.

b. 옮길 위치에 Redo Log File을 복사한다.

```
!cp ${ORACLE_BASE}/oradata/redo01.log
        ${NEW_VOL}/oradata/redo01.log
```

c. Database를 MOUNT 모드로 실행시킨다.

d. 다음의 명령어를 실행한다.

```
ALTER DATABASE RENAME FILE
'${ORACLE_BASE}/oradata/redo01.log'
TO '${NEW_VOL}/oradata/redo01.log';
```

e. Database를 Open 모드로 변경한다.

```
ALTER DATABASE open;
```

7.4 Oracle 과 Web Program 연동하기

오라클을 웹서버에 연동하는 방법으로 크게 Jsp 환경과 PHP 환경이 있다.
JSP 환경은 JDBC 드라이브를 설치하면 간단히 된다. 여기서는 PHP 연동에 대해
간단히 알아보도록 하겠다.

그냥 기존 APM 연동 시 PHP configure option 중 마지막에 아래 구문을 추가한다.

--with-oci8=${ORACLE_HOME}

7.5 Oracle HA 를 위해 필요한 스크립트

오라클 HA 를 위해서는 Oracle Daemon 을 자동으로 실행 시켜 주는 start script 와 shutdown script 그리고 이를 제어 할수 있는 init script 가 필요하다.

이 3가지 script 를 이용하여 heartbeat 의 적용 하면 된다. 물런 앞에서 설명한 DRBD 설정 역시 포함이 되어야 한다. 일단 여기서는 Heartbeat 에서 사용될 3개의 스크립트와 1개의 oracle profile 에 대해 알아보자

먼저 heartbeat 의 구동이 root 상에서 이루어 진다는 것을 알아야 한다.
그리고 대부분의 Oracle Profile 은 oracle 계정에만 적용하도록 셋팅을 한다.

오라클을 구동을 하기 위해서는 반드시 oracle profile 설정이 필요하다.

oracle dba 계정이 아닌 root 가 oracle 을 start 시키기 위해서는 root 역시 oracle profile 정보를 가지고 있어야 한다. 그냥 root 의 .bash_profile 에 정보를 입력해도 되지만, 구동 시키는 console 에서만 oracle profile 이 적용 되게 하는 방법으로 oracle init scripts 를 만들었다. - 이게 FM 이다.

[root@test01 root]# cat /etc/sysconfig/oracle

```
--------------------------------------------------------------------------
# Oracle Environment
ORACLE_BASE=/oracle/app
ORACLE_HOME=$ORACLE_BASE/product/9.2.0
ORACLE_TERM=xterm
TNS_ADMIN=$ORACLE_HOME/network/admin
NLS_LANG=AMERICAN_AMERICA.KO16KSC5601
ORA_NLS33=$ORACLE_HOME/ocommon/nls/admin/data
PATH=$PATH:$ORACLE_HOME/bin
export PATH ORACLE_BASE ORACLE_HOME ORACLE_TERM TNS_ADMIN NLS_LANG ORA_NLS33
--------------------------------------------------------------------------
```

// Oracle startup , shutdown Script //

[root@test01 root]# cat /etc/rc.d/init.d/ora_start.sql

```
connect sys/sys@TEST01 as sysdba
startup
quit
```

--> ㅋㅋ 민망하군.. 간단히 설명하면

connect <user_id>/<password>@DB명 as sysdba

이런 형식으로 사용하면 된다. 접속 이후필요한 여러 DBA 명령을 이곳에 상황에서 맞게 만들어 주면 된다.

[root@test01 root]# cat /etc/rc.d/init.d/ora_shutdown.sql

```
connect sys/sys@TEST01 as sysdba
shutdown
quit
```

// Oracle Init Scripts //

[root@test01 root]# cat /etc/rc.d/init.d/oractl

```
#!/bin/sh

. /etc/init.d/functions

if [ -f "/etc/sysconfig/oracle" ]; then
        source /etc/sysconfig/oracle
fi

RETVAL=0

start () {
        echo -n $"Starting Daemon of Oracle : "
        if [ -f "/etc/sysconfig/oracle" ]; then
                . /etc/sysconfig/oracle
                echo $"(Starting .....)"
                sqlplus /nolog @/etc/rc.d/init.d/ora_start.sql
```

```
        else
                echo $"(No oracle config file)"
                exit 0
        fi

        RETVAL=$?
        echo
        [ $RETVAL -eq 0 ] && touch /var/lock/subsys/oracle

        }

stop () {
        echo -n $"Shutdown Daemon of Oracle: "
        sqlplus /nolog @/etc/rc.d/init.d/ora_shutdown.sql

        RETVAL=$?
         echo
        [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/oracle
}

case "$1" in

  start)
        start
        ;;

  stop)
        stop
        ;;

  restart)
        stop
        start
        RETVAL=$?
        ;;

  *)
```

```
        echo $"Usage: $0 {start|stop|restart}"
        exit 1
esac
exit $RETVAL
```

핵심만 설명하면 아래 구문으로 Oracle 내부 scripts 파일을 include 할수 있다.

sqlplus /nolog @/etc/rc.d/init.d/ora_start.sql

7.6 DRBD + Heartbeart 를 이용하여 Oracle HA 하기

a. DRBD 설치한다.

b. Heartbeat를 설치한다.

c. Node01, Node02 Oracle 설치
(File Data Path, File Data Name, SID, DB_name, Oracle 환경 설정 등을 모두 동일하게)

d. oracle File Data 를 모두 DRBD Volum 상의 디렉토리로 이동한다.

e. oracle Control File 상의 Data Path 정보를 모두 DRBD 볼륨상의 경로로 변경
(설치 시 HA 설계를 고려 하여 경로 지정 권장)

f. DRBD 설정
설정이 완료된 이후 drbd device로 file system 을 새로 만들어야 한다. 그렇기 때문에 오라클 DATA 를 모두 다른곳에 백업 한 후 mkreiserfs /dev/drbd0 등으로 filesystem 생성 후 다시 그 위치로 Data File 을 옮겨 놓는다.

g. Heartbeat 설정

[root@test01 root]# vi /etc/ha.d/ha/haresources

---

test01 192.168.123.170 drbddisk::drbd0 Filesystem::/dev/drbd0::/data1::reiserfs oractl

---

h. 오라클 DBA 인증 파일 Sync

${ORACLE_HOME}/dbs/orapw<SID> 파일을 양 노드 모두 동일하게 해주어야 한다.
Primary Node 의 ${ORACLE_HOME}/dbs/orapw<SID> 파일을 Secondary 에 복사해 준다.

7.7 두개의 Instance 를 이용한 Oracle Both HA 구축하기

* 시스템 디자인 :

test01, test02 노드에 각각 TEST01, TEST02 라는 Oracle DB 를 별도의 Instance 로 운영을 하고 자 한다. test01 노드에 문제 발생 시 test02 노드에서 test01 의 TEST01 DB 를 HA 하고 test02 노드에서 문제 발생 시는 test01 에서 TEST02 노드를 HA 하여 서비스의 고가용성을 추구 한다.

* DRBD 설계 다자인 :

test01, test02 노드에 각각 /data1, /data2 라는 물리적 볼륨을 만든다.
/data1 볼륨은 test01 에선 primary, test02 에서는 secondary 로 설정
/data2 볼륨은 test01 에선 secondary, test02 에서는 primary 로 설정

test01 에서는 /data1 에 oracle file data 가 위치 하도록
test02 에서는 /data2 에 oracle file data 가 위치 하도록 설계한다.

-- drbd config

[root@test01 root]# vi /etc/drbd.conf

```
---------------------------------------------------------------------------
skip {
  As you can see, you can also comment chunks of text
  with a 'skip[optional nonsense]{ skipped text }' section.
  This comes in handy, if you just want to comment out
  some 'resource <some name> {...}' section:
  just precede it with 'skip'.
```

```
   The basic format of option assignment is
   <option name><linear whitespace><value>;

   It should be obvious from the examples below,
   but if you really care to know the details:

   <option name> :=
         valid options in the respective scope
   <value>   := <num>|<string>|<choice>|...
                depending on the set of allowed values
                for the respective option.
   <num>     := [0-9]+, sometimes with an optional suffix of K,M,G
   <string> := (<name>|₩"([^₩"₩₩₩n]*|₩₩.)*₩")+
   <name>    := [/_.A-Za-z0-9-]+
}


resource drbd0 {
   protocol C;
   incon-degr-cmd "echo '!DRBD! pri on incon-degr' | wall ; sleep 60 ; halt -f";
   startup {
      degr-wfc-timeout 120;
   }

   disk {
      on-io-error     detach;
   }

   net {
      # sndbuf-size 512k;
      # timeout           60;      #   6 seconds   (unit = 0.1 seconds)
      # connect-int     10;      # 10 seconds   (unit = 1 second)
      # ping-int          10;      # 10 seconds   (unit = 1 second)
      # max-buffers       2048;
      # max-epoch-size   2048;
      # ko-count 4;
      # on-disconnect reconnect;

   }
```

```
    syncer {
      rate 10M;
      group 1;
      al-extents 257;
    }

    on test01 {
      device     /dev/drbd0;
      disk       /dev/sda8;
      address    192.168.123.165:7788;
      meta-disk  internal;
    }

    on test02 {
      device     /dev/drbd0;
      disk       /dev/sda8;
      address    192.168.123.166:7788;
      meta-disk internal;
    }
}

resource drbd1 {

    protocol C;
    incon-degr-cmd "echo '!DRBD! pri on incon-degr' | wall ; sleep 60 ; halt -f";
    startup {
      degr-wfc-timeout 120;     # 2 minutes.
    }


    disk {
      on-io-error    detach;
    }

    net {
    }

    syncer {
```

```
    rate 10M;
    group 0;
    al-extents 257;
  }

  on test01 {
    device      /dev/drbd1;
    disk        /dev/sda9;
    address     192.168.123.165:7799;
    meta-disk   internal;

  }

  on test02 {
    device      /dev/drbd1;
    disk        /dev/sda9;
    address     192.168.123.166:7799;
    meta-disk internal;
  }

}
~
```

---

* HA 설계 디자인 :

test01 노드가 Fail 시 test02 노드에서는 /data1 volum 의 Drbd status 를 Secondary 에서 primary 로 전환하고 /data1 을 mount 시킨다. 그런 후 TEST01 Oracle Instance Start script 를 실행한다.

test02 노드가 fail 시엔 test01 노드에서 /data2 volum 의 Drbd status 를 secondary 에서 primary 로 전환하고 /data2 를 mount 한다. 그리고 TEST02 Oracle Instance start script 를 실행한다.

[root@test01 root]# vi /etc/ha.d/haresources

```
test01 192.168.123.170 drbddisk::drbd0 Filesystem::/dev/drbd0::/data1::reiserfs oractl
```

test02 192.168.123.171 drbddisk::drbd1 Filesystem::/dev/drbd1::/data2::reiserfs oractl1

[root@test01 root]# vi /etc/rc.d/init.d/drbddisk

```
#!/bin/bash
#
# This script is inteded to be used as resource script by heartbeat
#
# Jan 2003 by Philipp Reisner.
#
###

DEFAULTFILE="/etc/default/drbd"
DRBDADM="/sbin/drbdadm"

if [ -f $DEFAULTFILE ]; then
   . $DEFAULTFILE
fi

if [ "$#" -eq 2 ]; then
   RES="$1"
   CMD="$2"
else
   RES="all"
   CMD="$1"
fi

case "$CMD" in
      start)
            $DRBDADM primary $RES
            ;;
      stop)
            $DRBDADM secondary $RES
            ;;
      status)
            if [ "$RES" = "all" ]; then
                  echo "A resource name is required for status inquiries."
                  exit 10
```

```
        fi
        ST=$( $DRBDADM state $RES 2> /dev/null )
        ST=${ST%/*}
        if [ "$ST" = "Primary" ]; then
            echo "running"
        else
            echo "stopped"
        fi
        ;;
    *)
        echo "Usage: drbddisk [resource] {start|stop|status}"
        exit 1
        ;;
esac

exit 0
```

```
[root@test01 root]# vi /etc/rc.d/init.d/Filesystem
```

---#!/bin/sh

```
unset LC_ALL; export LC_ALL
unset LANGUAGE; export LANGUAGE

prefix=/usr
exec_prefix=/usr
#. /etc/ha.d/shellfuncs
. /etc/ha.d/shellfuncs

# Utilities used by this script
MODPROBE=/sbin/modprobe
FSCK=/sbin/fsck
FUSER=/sbin/fuser
MOUNT=/bin/mount
```

```
UMOUNT=/bin/umount
BLOCKDEV=/sbin/blockdev

check_util () {
    if [ ! -x "$1" ] ; then
        ha_log "ERROR: setup problem: Couldn't find utility $1"
        exit 1
    fi
}

usage() {

cat <<-EOT;
        usage: $0 <device> <directory> <fstype> [<options>] {start|stop|status}

        <device>    : name of block device for the filesystem. e.g. /dev/sda1, /dev/md0
                      OR -LFileSystemLabel OR -Uuuid or an NFS specification
        <directory> : the mount point for the filesystem
        <fstype>    : name of the filesystem type. e.g. ext2
        <options>   : options to be given as -o options to mount.

        $Id: Filesystem.in,v 1.10 2003/07/03 02:14:14 alan Exp $
        EOT
}

#
#       Make sure the kernel does the right thing with the FS buffers
#       This function should be called after unmounting and before mounting
#       It may not be necessary in 2.4 and later kernels, but it shouldn't hurt
#       anything either...
#
#       It's really a bug that you have to do this at all...
#
flushbufs() {
  if
    [ "$BLOCKDEV" != "" -a -x "$BLOCKDEV" ]
  then
    case $1 in
      -*|[^/]*:/*)      ;;
```

```
        *)                      $BLOCKDEV --flushbufs $1;;
      esac
   fi
}
# Check the arguments passed to this script
DEVICE=$1
MOUNTPOINT=$2
FSTYPE=$3

case $DEVICE in
   -*) # Oh... An option to mount instead...   Typically -U or -L
           ;;
   [^/]*:/*)       # An NFS filesystem specification...
           ;;
   *)      if [ ! -b "$DEVICE" ] ; then
              ha_log "ERROR: Couldn't find device $DEVICE. Expected /dev/??? to exist"
              usage
              exit 1
           fi;;
esac

if [ ! -d "$MOUNTPOINT" ] ; then
           ha_log "ERROR: Couldn't find directory   $MOUNTPOINT to use as a mount point"
           usage
           exit 1
fi



# Check to make sure the utilites are found
check_util $MODPROBE
check_util $FSCK
check_util $FUSER
check_util $MOUNT
check_util $UMOUNT

case $# in
   4)      operation=$4; options="";;
   5)      operation=$5; options="-o $4";;
   *)      usage; exit 1;;
```

```
esac


# Look for the 'start', 'stop' or status argument
case "$operation" in

#
# START: Start up the filesystem
#
start)

        # See if the device is already mounted.
        $MOUNT | cut -d' ' -f3 | grep -e "^$MOUNTPOINT$" >/dev/null
        if [ $? -ne 1 ] ; then
            ha_log "ERROR: Filesystem $MOUNTPOINT is already mounted!"
            exit 1;
        fi

        # Insert SCSI module
        $MODPROBE scsi_hostadapter >/dev/null 2>&1

        # Insert Filesystem module
        $MODPROBE $FSTYPE >/dev/null 2>&1
        grep -e "$FSTYPE"'$' /proc/filesystems >/dev/null
        if [ $? != 0   ] ; then
                ha_log "ERROR: Couldn't find filesystem $FSTYPE in /proc/filesystems"
                usage
                exit 1
        fi

        # Check the filesystem & auto repair.
        # NOTE: Some filesystem types don't need this step...  Please modify
        #           accordingly

        if
          case $FSTYPE in
            ext3|reiserfs|xfs|jfs|vfat|fat|nfs) false;;
            *)                                  true;;
          esac
```

```
        then
          ha_log "info: Starting filesystem check on $DEVICE"
          $FSCK -t $FSTYPE -a $DEVICE

          # NOTE: if any errors at all are detected, it returns non-zero
          # if the error is >4 then there is a big problem
          if
            [ $? -ge 4 ]
          then
            ha_log "ERROR: Couldn't sucessfully fsck filesystem for $DEVICE"
            exit 1
          fi
        fi

        flushbufs $DEVICE
        # Mount the filesystem.
        if
          $MOUNT -t $FSTYPE $options $DEVICE $MOUNTPOINT
        then
          : Mount worked!
        else
          ha_log "ERROR: Couldn't mount filesystem $DEVICE on $MOUNTPOINT"
          exit 1
        fi

# end of start)
;;

#
# STOP: Unmount the filesystem
#
stop)

        # See if the device is currently mounted
        if
          $MOUNT | grep -e " on $MOUNTPOINT " >/dev/null
        then
                # Kill all processes open on filesystem
                $FUSER -mk $MOUNTPOINT
```

```
                # Get the current real device name...
                # (specified devname could be -L or -U...)
                DEV=`$MOUNT | grep "on $MOUNTPOINT " | cut -d' ' -f1`
                # Unmount the filesystem
                $UMOUNT $MOUNTPOINT
                if [ $? -ne 0 ] ; then
                        ha_log "ERROR: Couldn't unmount $MOUNTPOINT"
                        exit 1
                fi
                flushbufs $DEV
        else
                ha_log "WARNING: Filesystem $MOUNTPOINT not mounted?"
        fi

# end of stop)
;;

#
# STATUS: is the filesystem mounted or not?
#
status)

        $MOUNT | grep -e "on $MOUNTPOINT " >/dev/null
        if [ $? = 0 ] ; then
                echo "$MOUNTPOINT is mounted (running)"
        else
                echo "$MOUNTPOINT is unmounted (stopped)"
        fi
# end of status)
;;



*)
    echo "This script should be run with a fourth argument of 'start', 'stop', or 'status'"
    usage
    exit 1
;;
```

```
esac

# If you got to this point, chances are everything is O.K.
exit 0;
```

끝~

본 문서는 클루닉스 기술부 2차, 3차 세미나 자료와 함께 보시길 권장합니다.

DRBD를 이용한 HA 시스템 구축에 대한 저자 견해

HA 는 실제 개별 application level 의 고가용성을 보장하지 않는다. 즉 Server 에 여러개의 application 이 탑재된 경우 일부 application 에 fail 이 일어나고 server system 이 active status 를 유지할 경우에는 백업 서버로 서비스가 전환되지 않는다.

물런 server load 로 인해 특정 application 만 서비스가 정지하는 경우는 극히 드물기에 이런 경우를 배재 할수도 있지만 application level 까지도 status check 를 하는 load balance 의 경우와 비교할 경우 단점은 분명 존재한다.

이를 해결 하기 위해서는 both node writing 이 가능한 volum mirroring 기술을 이용하여 클러스터 파일 시스템 환경을 구축하여야 한다. 이를 가능케 하는 기술로 대표적인 것이 GFS(global file system) 이다.
Encluster + DRBD + GFS 로 이루어진 Cluster suite 를 만들면 최적의 Cluster Solution 이 되지 않을까 생각한다.

Introduction to

High Performance Computer Cluster

using Linux

(고성능 HPC 클러스터 구성)

클루닉스 기술부 5차 기술 세미나 자료

서 진우(alang@clunix.com)

목차 –

1. Teragon HPC 운영체제 설치
   1.1 NFS, Kick Start 설치 하기
   1.2 PXE, NFS, Kick Start 설치 하기

2. Teragon HPC Network 설정하기
   2.1 Network channel 이중화
   2.2 Rsh, rlogin 설정
   2.3 ssh 설정
   2.4 Dutils 설정
   2.5 NFS 설정
   2.6 NIS 계정 통합 시스템

3. Teragon HPC 설정하기
   3.1 Time Sync 설정
   3.2 Compiler 설치 및 설정
   3.3 Library 설치
   3.4 병렬프로그램 설치 및 설정 (MPICH, LAM-MPI)
   3.5 분산 시스템 일괄 계정 관리
   3.6 Open PBS 설치 및 관리 하기
   3.7 HPC Management Tool 설치, 설정, 관리 (dutils2)

4. Teragon HPC Security 설정
   4.1 Command, Directory Security
   4.2 Service Security (rsh, rlogin, ssh, nfs, ftp)
   4.3 Network Security (network filtering)

5. HPC Benchmark 성능 분석 하기
   5.1 NAS Parallel Benchmark (NPB)
   5.2 High Performance Linpack Benchmark (HPL)
   5.3 Parallel add Benchmark (정수 연산)
   5.4 Stream Memory Benchmark
   5.5 Bonnie++ Disk I/O Benchmark
   5.6 Netpipe Network Benchmark (TCP, MPI)

1. Teragon HPC 운영체제 설치 하기

1.1 NFS, Kick Start 설치 하기

Teragon NFS 원격 설치 환경을 구축 하기 위해선 먼저 Master Node를 설치를 해야 한다.
Master Node 에 기본적인 관리 노드 구성에 맞게 패키지 구성을 완료한 뒤, 원격 설치
서비스를 하기 위해 기본적으로 NFS 서비스와 DHCP 서비스를 구동시킨다.

그런후 계산 노드들은 부팅 CD 나 부팅 디스켓으로 부팅 후 kick start 설치모드로
원격 설치 서버에 접속하여 설치 시 필요한 정보와 패키지를 다운받아 자동 설치 하
게 된다. 아래는 이와 같은 일련의 작업을 설명한 것이다.

- Linux 운영 체제 설치

Master Node의 Redhat Linux 운영체제 설치 시를 아래와 같은 Remote install 관련 Package를 포함해서 설치 해야 한다.

dhcp
dhcp-devel
anaconda-runtime
redhat-config-kickstart
rsh-server

그리고 기본적으로 설치가 되어지는 lam-mpi package는 package 구성 시 제거한다.
이밖에 package 는 기술부 1차 세미나를 참조 하여 설치 한다.

- Remote Install Service 설치

Master Node 에 원격 설치 서버를 만들기 위해 Redhat 배포판 CD 를 통합하여
원격 설치에 맞게 구성을 만들어 놓는다.

> 기본 디렉토리를 생성한다.

```
# mkdir -p /data/clunix/teragon
# mkdir /data/clunix/cd1
# mkdir /data/clunix/cd2
# mkdir /data/clunix/cd3
```

> cd1,cd2,cd3 디렉토리에 Redhat9.x CD 내용을 각각 복사한다

```
# mount -t iso9660 /dev/cdrom /mnt/cdrom
# cp -rf /mnt/cdrom/* /data/clunix/cd1
# cp -rf /mnt/cdrom/* /data/clunix/cd2
# cp -rf /mnt/cdrom/* /data/clunix/cd3
```

> Teragon Server 에 기본 설치되는 RPM Package List 정리하여 'kick' 이란 file 로 보관한다.
> 'kick' file 저장위치는 /data/clunix/ 안에 둔다.

```
# rpm -qa > /data/clunix/kick
```

> redhat 원본CD 에서 Teragon 에 맞는 CD 구성을 만든다.
> addrpm.sh 스크립터를 $teracd 에 만든다.

```
# cd /data/clunix
```

[root@teragon01 clunix]# vi addrpm.sh

```
==========================================================================
==========

#!/bin/sh

for file in `cat kick`
do
        cp -f ./cd1/RedHat/RPMS/${file}* ./teragon/RedHat/RPMS/
        cp -f ./cd2/RedHat/RPMS/${file}* ./teragon/RedHat/RPMS/
        cp -f ./cd3/RedHat/RPMS/${file}* ./teragon/RedHat/RPMS/
done
```

==========

chmod 700 addrpm.sh

> addrpm.sh 실행한다.

만일 배포판 전체 패키지를 통합할 경우 CD1 의 내용에 CD2, CD3 의 RedHat/RPMS 밑의 rpm package 들만을 CD1으로 복사하면 된다.

[root@teragon01 clunix]# ./addrpm.sh

이러면 실제 설치에 필요한 RPM 패키지가 하나의 폴더안에 집중하여 관리 할수 있다.
(원격 설치 시 자동 One step 설치 방식으로 구성하여 설치 과정 중에 관리자가 계속 대기 하고 있다가 설치에 필요한 정보를 입력하는 수고를 생략하기 위한 것이다.)

기본적인 Redhat rpm package 가 하나로 통합을 시켰으면 이제 HPC에 필요한 추가 패키지를 rpm 으로 만들어 Redhat 배포판 디렉토리 구성의 RPM 디렉토리 안에 추가해 둔다.

dutils, rsh-enable, ecm-base, ecm-mon, ecmhpc-ui 등등의 패키지를 추가한다.

cp dutil-1.2.1-2.i386.rpm /data/clunix/teragon/RPMS
cp rsh-enable-0.1-1.i386.rpm /data/clunix/teragon/RPMS
cp ecm-base-1.0.0-5.i386.rpm /data/clunix/teragon/RPMS
cp ecm-mon-1.0.0-5.i386.rpm /data/clunix/teragon/RPMS
cp ecmhpc-ui-1.0.0-5.i386.rpm /data/clunix/teragon/RPMS

> clunix 의 RPM List 를 다시 만들어 준다. anaconda-runtime 이 설치되어져 있어야 한다.

- 기본적인 RPM 구성 정보를 담는 파일을 변경된 구성의 정보로 업데이트 시켜 주어야 한다.

[root@teragon01 clunix]# /usr/lib/anaconda-runtime/genhdlist --withnumbers –hdlist ₩
teragon/RedHat/base/hdlist /data/clunix/teragon/

**** 레드헷 전체 RPM 구성으로 이미지 서버 구축 시에는 위 내용 대신 CD1 에 있는 기본 hdlist 를 사용한다. 아님 설치 되지 않는 패키지가 발생할 수 있다.**

- DHCP Server 설정

[root@teragon01 clunix]# vi /etc/dhcpd.conf

```
================================================================================
ddns-update-style interim;
ignore client-updates;
default-lease-time 600;
max-lease-time 7200;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.100.255;
option routers 192.168.100.1;
option domain-name-servers 192.168.100.1;
option domain-name "teragon01";


subnet 192.168.100.0 netmask 255.255.255.0 {
    range 192.168.100.150 192.168.100.253;
}
================================================================================
```

> dhcp 에 사용되는 dev 지정

[root@teragon01 clunix]# vi /etc/sysconfig/dhcpd

```
================================================================================
DHCPDARGS=eth0

================================================================================
```

======

> dhcpd deamon restart

[root@teragon01 clunix]# /etc/rc.d/init.d/dhcpd restart

- NFS Server 설정

[root@teragon01 clunix]# vi /etc/exports

==========

/data/clunix/teragon *(ro,no_root_squash)

==========

[root@teragon01 clunix]# /etc/rc.d/init.d/portmap restart
[root@teragon01 clunix]# /etc/rc.d/init.d/nfs restart

> nfs mount 테스트를 한다.

[root@teragon01 data]# mount -t nfs localhost:/data/clunix/teragon /tmp
[root@teragon01 data]# umount /tmp

***** nfs 서버 설정 후 mount 가 되지 않을 경우 다시 한번 portmap 과 nfs 관련 데몬이 정상적으로 작동하는지 확인 하고, iptables 가 적용되지 않는 지 확인 하도록 한다. 초기 OS 설치 시 firewall 설정이 되어지므로 설치 단계에서는 반드시 iptables 를 잠시 stop 하도록 한다.**

[root@teragon01 data]# /etc/rc.d/init.d/iptables stop

- Kick Start 설정 파일 생성 하기

Kick Start 는 Redhat Linux 배포판 버전 설치를 일괄 설치가 가능하게 해주는 구축 방식이다.
Kick Start 에서 배포판 설치에 필요한 정보를 하나의 설정 파일로 만들고, 이 설정 파일을 이용하여 설

치 시 설치자에서 필요한 정보를 응답식으로 물어보지 않고 자동으로 설치 하도록 한다.

Kick start 설정은 Redhat Linux 배포판의 버전 마다 패키지 구성 Class Name 이 다르기 때문에 다른 버전의 배포판에서 제작한 설정 파일을 이용하여 만들 경우에 설치 된 배포판에서 직접 생성하는 것이 좋다. 우선 Master Server 설치를 하였으니 이 서버에서 생성이 가능하다.
생성하는 방법으로는 여러 가지가 있는데 가장 쉬운 방법은 이미 수동으로 설치한 Master Server 의 /root 디렉토리 밑에 보면 anaconda-ks.cfg 파일이 있을 것이다. 이것이 Master Server 설치 시 설치자가 설치에 필요한 정보를 토대로 설정 파일을 만들어 놓은 것이다. 이 설정 파일을 이용하여 원격 설치에 맞게 수정하는 것이 제일 간편하다. 만일 계산용 서버의 운영체제 구성이 Master Server 와 상이하게 다른 경우는 Master Server 에서 redhat-config-kickstart 란 프로그램을 이용하여 임의적으로 생성할 수도 있다.

- redhat-config-kickstart를 이용하여 생성한 ks.cfg 파일

```
==========================================================================
=========
#Generated by Kickstart Configurator

#System  language
lang ko_KR.eucKR
#Language modules to install
langsupport  --default=ko_KR.eucKR
#System keyboard
keyboard us
#System mouse
mouse generic3ps/2
#Sytem timezone
timezone Asia/Seoul
#Root password
rootpw --iscrypted $1$iv0/Mp/1$uoKRMOmtvv/DAVYEnnrGK.
#Use text mode install
text
#Install Red Hat Linux instead of upgrade
install
#Use NFS installation Media
nfs --server=192.168.10.254 --dir=/data/clunix/teragon
#System bootloader configuration
bootloader --useLilo --linear --location=mbr
#Clear the Master Boot Record
```

```
zerombr yes
#Partition clearing information
clearpart --all --initlabel
#Disk partitioning information
part /boot --fstype ext3 --size 200
part / --fstype ext3 --size 2000 --asprimary
part /usr --fstype ext3 --size 5000
part /usr/local --fstype ext3 --size 5000
part /var --fstype ext3 --size 2000
part swap --size 2000
part /home --fstype ext3 --size 1 --grow
#System authorization infomation
auth  --useshadow  --enablemd5
#Network information
#network --bootproto=static --ip=192.168.1.1 --netmask=255.255.255.0 --gateway=192.168.1.254 --
nameserver=192.168.1.254 --device=eth0
#Firewall configuration
firewall --medium --trust=eth1 --ftp --ssh
#Do not configure XWindows
skipx
#Package install information
%packages --resolvedeps
@ X Window System
@ Editors
@ Engineering and Scientific
@ Graphical Internet
@ Text-based Internet
@ Graphics
@ Authoring and Publishing
@ Server Configuration Tools
@ DNS Name Server
@ Network Servers
@ Development Tools
@ Kernel Development
@ X Software Development
@ Administration Tools
@ System Tools
```

```
%post --interpreter /bin/bash

#### setting IP ADDR ############################
stringZ=`cat /proc/cmdline`
for service in $stringZ
do
  if [ `expr match "$service" "ksip="` -eq "5" ] ; then
    IPADDR=${service:5}
  fi
  if [ `expr match "$service" "ksnm="` -eq "5" ] ; then
    NAME=${service:5}
  fi
done

echo "DEVICE=eth1" > /etc/sysconfig/network-scripts/ifcfg-eth1
echo "BOOTPROTO=static"   >> /etc/sysconfig/network-scripts/ifcfg-eth1
echo "IPADDR=${IPADDR}"   >> /etc/sysconfig/network-scripts/ifcfg-eth1
echo "NETMASK=255.255.255.0" >> /etc/sysconfig/network-scripts/ifcfg-eth1
echo "ONBOOT=yes"   >> /etc/sysconfig/network-scripts/ifcfg-eth1
echo "NETWORKING=yes"          >   /etc/sysconfig/network
echo "HOSTNAME=${NAME}"        >> /etc/sysconfig/network

################################################

##### setting services off ###################################
chkconfig kudzu off
chkconfig reconfig off
chkconfig ipchains off
chkconfig nfslock off
chkconfig nfs off
chkconfig keytable off
chkconfig apmd off
chkconfig rawdevices off
chkconfig sendmail off
chkconfig gpm off
chkconfig anacron off
chkconfig atd off
```

==========

이와 같이 만들어진 ks.cfg 파일을 /data/clunix/teragon 에 복사해 두고 dhcp, nfs, portmap 등의 서비스가 정상 가동하는지 확인 후 계산 노드 들을 네트워크 원격 설치 방식으로 설치 한다.

- 계산 서버 설치 하기

계산 서버의 전원을 켜고 배포판 부팅 CD나 Master 서버에서 만든 booting floppy disk를 이용하여 부팅을 시킨다. "Boot :" 와 같은 부팅 프롬프트가 나타나면, 프롬프트 상에 아래 내용을 입력해 준다.

boot : ks ks=nfs:[nfs_server_ip 혹은 호스트명]:/data/clunix/teragon/ks.cfg ksip=[설치서버의 기본 IP] ksnm=[설치서버의 호스트명]

그럼 자동으로 위의 ks.cfg 파일에서 설정한 구성으로 운영 체제가 설치 될것이다.

***** 계산 노드의 랜카드가 두장 이상일 경우 실제 nfs 로 통신할 랜카드를 선택 해야 한다.**
실제 랜선이 연결된 Ethernet 측의 네트워크 장치를 사용하여 설치를 해야 한다.

1.2 PXE, NFS, Kick Start 설치 하기

- 기본 운영 체제 설치

PXE-ROM 이 있는 Lan Card 의 경우 PXE-protocol 을 이용하여 랜카드 자체에서 네트워크를 통해 부팅 이미지를 읽어 올 수가 있다.
이 기능을 이용하면 리눅스 설치 CD 와 같은 미디어 없이 네트워크를 통해 리눅스를 설치 하거나 하드디스크 없이 CPU, MEM 만 달려 있는 시스템을 부팅 이미지와 NFS 서버를 통해 시스템을 작동 시킬 수 있다. 이런 방식을 이용하여 Diskless-HPC를 구축 하기도 한다. 여기서는 설치에 관련된 내용만을 다루도록 한다.

Pxe를 이용한 리눅스 네트워크 설치 때 사용되는 기본 프로그램은 아래와 같다.
==========

==========

redhat-config-netboot

tftp-server

dhcpd

nfs

위의 프로그램이 원격 설치 서버 시스템에 기본적으로 설치가 되어 있어야 한다.

설치는 간단히 배포 판에서 제공하는 RPM으로 설치 하면 된다.

rpm -Uvh redhat-config-netboot

rpm -Uvh tftp-server-x.x.x.rpm

rpm -Uvh tftp-x.x.x.rpm

그럼 /tftpboot 가 생성된다.

- TFTP 설정 하기

/etc/xinetd.d/tftp 의 disable = yes를 no 변경 해주고 xinetd 를 재 시작해 준다.

vi /etc/xinit.d/tftp

==========

service tftp

{

disable = no

socket_type = dgram

protocol = udp

wait = yes

user = root

server = /usr/sbin/in.tftpd

server_args = -s /tftpboot

per_source = 11

cps = 100 2

flags = IPv4

}

==========

```
# /etc/rc.d/init.d/xinetd restart
# chkconfig --level 3 xinetd on
# chkconfig --level 3 tftp on
```

- DHCP 설정 하기

PXE 네트워크 설치 방식에서 가장 유의해야 할 설정은 dhcp 설정이다.
위의 Nfs 네트워크 설치 방식 처럼 단순히 ip 만 활당해 주는 것이 아니라 네트워크를 통해
커널 이미지를 로딩하여 부팅을 해야 하기 때문에 다소 복잡하다.

```
# vi /etc/dhcpd.conf
=============================================================================
=========
ddns-update-style interim;
ignore client-updates;
default-lease-time 600;
max-lease-time 7200;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.100.255;
option routers 192.168.100.254;
option domain-name-servers 192.168.100.254;
option domain-name "node00.tergon.hpc";

# 아래 추가
allow booting;
allow bootp;

class "pxeclients" {
     match if substring (option vendor-class-identifier, 0, 9) = "PXEClient";
     next-server 192.168.100.254;
     filename "linux-install/pxelinux.0";
}

# 특정 Mac Address를 가지고 접속하는 클라이언트에서 특정 IP 부여
# 이는 tftp 설정에서 클라이언트 시스템 별로 별도의 부팅 이미지 및 ks 파일을
# 적용할수 있기 때문에 지정을 해주면 보다 자동화된 시스템 구축이 가능하다.
```

```
group {
        next-server 192.168.100.254;
        filename "linux-install/pxelinux.0";

        host node01 {
                hardware ethernet 00:09:3d:10:8b:c6;
                fixed-address 192.168.100.1;
                }
        host node02 {
                hardware ethernet 00:09:3d:10:96:64;
                fixed-address 192.168.100.2;
                }
        host node03 {
                hardware ethernet 00:09:3d:10:98:3b;
                fixed-address 192.168.100.3;
                }
        host node04 {
                hardware ethernet 00:09:3d:10:93:ce;
                fixed-address 192.168.100.4;
                }
        host node05 {
                hardware ethernet 00:09:3d:10:98:02;
                fixed-address 192.168.100.5;
                }
        host node06 {
                hardware ethernet 00:09:3d:00:7a:6c;
                fixed-address 192.168.100.6;
                }
        host node07 {
                hardware ethernet 00:09:3d:10:93:d4;
                fixed-address 192.168.100.7;
                }
        }

subnet 192.168.100.0 netmask 255.255.255.0 {
    range 192.168.100.100 192.168.100.150;
}
```

- PXE 설정하기

PXE 설정의 redhat-config-netboot란 명령을 이용하거나 수동으로 설정을 할 수 있다.
Redhat-config-netboot를 이용할 경우는 Master Server 운영체제 설치 시 X 관련 package를 포함하여야 한다. 이 문서에서는 pxe 관련 command를 이용한 수동 설정과 실제 설정 파일을 바로 수정하여 설정을 하는 방법에 대해 설명하도록 하겠다.

***** pxe utile 을 이용하여 설정 하는 방법**

설치 OS 구분 추가 :

pxeos -a -i "설명" -p <install_protocol> -D 0 -s <server_name> ₩
-L <net-location> <os-identifer>

예)
pxeos -a -i "Redhat ES3" -p NFS -D 0 -s 192.168.100.254 -L /home/clunix/teragon teragon

클라이언트 구분 추가 :

pxeboot -a -O <os-identifer> -r <ramdisk_size> <host>

예)

pxeboot -a -O teragon -r 10000 192.168.100.1

클라이언트 환경은 /tftpboot/linux-install/pxelinux.cfg/ 밑에 생성이 된다.
각 접속될 클라이언트 별(구분은 클라이언트 IP 로 한다)로 부팅 이미지와 ks 파일을 지정할수 있게 설정 파일이 나누어진다. 지정된 클라이언트가 아닌 시스템에서 접속한 경우에는 default 에 지정된 부팅 이미지로 부팅을 하게 된다. 기본적인 설정은 부팅이미지와 ramdisk size 정도 지정되어져 있다. 만일 별도의 kickstart 파일을 이용하여 자동 설치를 원할 경우에는 아래와 같이 ks 관련 append 를 추가 해 주어야 한다. ks 관련 설정은 반드시 기존의 append 설정 뒤에 붙여서 사용하여야 한다.

/tftpboot/linux-install/pxelinux.cfg/default 를 열어보면..

```
default local
timeout 100
prompt 1
display msgs/boot.msg
F1 msgs/boot.msg
F2 msgs/general.msg
F3 msgs/expert.msg
F4 msgs/param.msg
F5 msgs/rescue.msg
F7 msgs/snake.msg

label local
  localboot 1

label 0
  localboot 1

label 1
  kernel linux-install/teragon/vmlinuz
  append initrd=linux-install/teragon/initrd.img ramdisk_size=10000 ₩
  ks=nfs:192.168.100.254:/home/clunix/teragon-pkg/ks.cfg ip=dhcp ksdevice=eth0
```

==========

특정 노드에 대한 자동 설정을 하고 쉽다면

/tftpboot/linux-install/pxelinux.cfg/C0A86401 와 같이 특정 호스트에서 접속할 경우
특정 부팅 이미지로 부팅을 시킬 수 있다.

```
# vi /tftpboot/linux-install/pxelinux.cfg/C0A86401
```

==========

```
default teragon
```

```
label teragon
      kernel linux-install/teragon/vmlinuz
append initrd=linux-install/teragon/initrd.img console=tty0 ramdisk_size=10000 ₩
ks=nfs:192.168.100.254:/home/clunix/teragon-pkg/ks.cfg ksip=192.168.200.1 ksnm=node01
ksdevice=eth0
```

==========

***** 설정 파일을 직접 수정하여 PXE 설정하는 방법**

```
# mkdir /tftpboot/linux-install/teragon
# cp /home/clunix/teragon-pkg/images/pxeboot/vmlinuz /tftpboot/linux-install/teragon
# cp /home/clunix/teragon-pkg/images/pxeboot/initrd.img /tftpboot/linux-install/teragon
```

tftpboot/linux-install/pxeconfig.cfg/default 편집

==========

```
default local
timeout 100
prompt 1
display msgs/boot.msg
F1 msgs/boot.msg
F2 msgs/general.msg
F3 msgs/expert.msg
F4 msgs/param.msg
F5 msgs/rescue.msg
F7 msgs/snake.msg


LABEL local
  localboot 1

LABEL node00
KERNEL teragon/vmlinuz
APPEND initrd=teragon/initrd.img ramdisk_size=10000 ₩
ks=nfs:192.168.100.254:/home/clunix/teragon-pkg/ks.cfg ksdevice=eth0 ₩
```

ksip=192.168.100.254 ksnm=node00

LABEL node01
KERNEL teragon/vmlinuz
APPEND initrd=teragon/initrd.img ramdisk_size=10000 ₩
ks=nfs:192.168.100.254:/home/clunix/teragon-pkg/ks.cfg ksdevice=eth0 ₩
ksip=192.168.100.1 ksnm=node01

==========

이제 클라이언트 시스템을 부팅하였을때 클라언트 시스템에 OS 가 설치된 환경이 아니면
자동으로 서버에 접속하여 서버의 부팅 이미지를 가져 오게 된다.

다른 부팅 매체 및 이미지가 존재 할 경우 PXE Network boot 관련 단축키 (F12) 를 부르면
PXE로 접속하게 된다.

****** 2005년 1월 17일 Sun-FZ440 (Opteron System) 설치 당시 문제점 :**

FZ440 시스템에 Redhat ES3-update 3 버전으로 설치를 하였다. PXE 접속하여 OS 설치까지는 무사히 잘 되었는데 설치 완료 후 리부팅 후 bootloader 를 불려 오지 못하는 문제 발생

kickstart file 수정등으로 재 시도를 해 보았지만.. 동일 증세 발견 ..

문제 원인은 Redhat ES3-update3 -amd 버전의 배포 CD 의 isolinux/ 밑의 부팅 관련 이미지 들의 문제임.

당초 CD1 으로 CDROM 부팅이 안되어서 CD1/images/boot.iso 파일의 이미지를 다시 풀어서 부팅이 가능한 문제가 있었는데 설치 이후 기본 부팅 이미지를 덮어 쓰는데 문제 있는 이미지를 덮어 쓰는 문제가 발생함.

어려움이 많았음

**그리고 bootscripts 에 관련된 패키지가 설치가 안 되는 문제가 발생하였다.
이는 잘못된 정보로 hdlist 를 생성해서 발생한 문제다. 그냥 기본 CD1 에 있는 hdlist 로 패키지리스트를 구성하면 된다**

****** ks config sample file ******

```
redhat-config-kickstart
================================================================================
=========
#Generated by Kickstart Configurator

#System  language
lang en_US
#Language modules to install
langsupport ko_KR.eucKR --default=en_US
#System keyboard
keyboard us
#System mouse
mouse generic3ps/2
#Sytem timezone
timezone Asia/Seoul
#Root password
rootpw --iscrypted $1$I4KvFDeP$nz5hBVDYcdWgfmQgljUEG0
#Reboot after installation
reboot
#Install Red Hat Linux instead of upgrade
install
#Use NFS installation Media
nfs --server=192.168.100.254 --dir=/home/clunix/teragon-pkg
#System bootloader configuration
bootloader --location=mbr
#Clear the Master Boot Record
zerombr yes
#Partition clearing information
clearpart --all --initlabel
#Disk partitioning information
part /boot --fstype ext3 --size 200 --ondisk sda
part / --fstype ext3 --size 3000 --asprimary --ondisk sda
part /usr --fstype ext3 --size 10000 --ondisk sda
part /var --fstype ext3 --size 3000 --ondisk sda
part swap --size 4000 --ondisk sda
```

```
part /home --fstype ext3 --size 1 --grow --ondisk sda
#System authorization infomation
auth  --useshadow  --enablemd5
#Firewall configuration
firewall --enabled
#Do not configure XWindows
skipx
#Package install information
%packages --resolvedeps
@ X Window System
@ GNOME Desktop Environment
@ Editors
@ Engineering and Scientific
@ Graphical Internet
@ Text-based Internet
@ Server Configuration Tools
@ DNS Name Server
@ Network Servers
@ Development Tools
@ Kernel Development
@ X Software Development
@ GNOME Software Development
@ Administration Tools
@ System Tools

%post

=================================================================
=========


custom kick start file
=================================================================
=========
lang ko_KR.eucKR
langsupport  --default=ko_KR.eucKR
keyboard us
mouse generic3ps/2
```

```
timezone Asia/Seoul
rootpw --iscrypted $1$nC947A.0$7k8ULqNdFZhOYDrnhz3ai1
text
install
nfs --server=192.168.100.254 --dir=/home/clunix/teragon-pkg
bootloader --location=mbr
zerombr yes
clearpart --all
part /boot --fstype ext3 --size 200 --ondisk sda
part / --fstype ext3 --size 3000 --ondisk sda --asprimary
part /usr --fstype ext3 --size 10000 --ondisk sda
part swap --size 4000 --ondisk sda
part /var --fstype ext3 --size 3000 --ondisk sda
part /home --fstype ext3 --size 100 --grow --ondisk sda
authconfig --enableshadow --enablemd5
#network --device eth0 --bootproto none
#network --device eth1 --bootproto none
firewall --enabled --trust=eth0
skipx
%packages
@ engineering-and-scientific
@ legacy-software-development
@ editors
@ system-tools
@ base-x
@ gnome-software-development
@ development-tools
@ text-internet
@ x-software-development
@ legacy-network-server
@ dns-server
@ gnome-desktop
@ admin-tools
@ compat-arch-development
@ authoring-and-publishing
@ korean-support
@ server-cfg
@ network-server
```

```
@ kernel-development
@ graphical-internet
@ compat-arch-support
krb5-server
rsh-server
-lam
kernel
kernel-smp
rsh-enable
dutil

%post --interpreter /bin/bash
#### setting IP ADDR ############################
stringZ=`cat /proc/cmdline`
for service in $stringZ
do
  if [ `expr match "$service" "ksip="` -eq "5" ] ; then
    IPADDR=${service:5}
  fi
  if [ `expr match "$service" "ksnm="` -eq "5" ] ; then
    NAME=${service:5}
  fi
done

echo "DEVICE=eth1" > /etc/sysconfig/network-scripts/ifcfg-eth1
echo "BOOTPROTO=static"   >> /etc/sysconfig/network-scripts/ifcfg-eth1
echo "IPADDR=${IPADDR}"   >> /etc/sysconfig/network-scripts/ifcfg-eth1
echo "NETMASK=255.255.255.0" >> /etc/sysconfig/network-scripts/ifcfg-eth1
echo "ONBOOT=yes"   >> /etc/sysconfig/network-scripts/ifcfg-eth1
echo "NETWORKING=yes"          >   /etc/sysconfig/network
echo "HOSTNAME=${NAME}"        >> /etc/sysconfig/network

################################################

chkconfig gpm off
chkconfig kudzu off
chkconfig netfs off
chkconfig random off
```

```
chkconfig rawdevices off
chkconfig atd off
chkconfig audit off
chkconfig acpid off
chkconfig isdn off
chkconfig iptables off
chkconfig ip6tables off
chkconfig autofs off
chkconfig portmap off
chkconfig nfslock off
chkconfig mdmonitor off
chkconfig cups off
chkconfig rhnsd off
chkconfig hpoj off
chkconfig xfs off
chkconfig arptables_jf off
/root/rsh.enable
=================================================================
=========
```

2. Teragon HPC Network 설정 하기

2.1 Network channel 이중화

HPC 시스템에서 Network resource를 주로 사용하는 용도는 운영체제 부분과 계산 부분으로 나누어 진다. 즉 NFS 파일 서비스, 데이터 동기화, 서버 관리 작업 등에 사용되는 운영체제 관련 부분과 실제 MPI 계산에 이용되는 계산 부분이 있다. 만일 하나의 네트워크 인터페이스를 이용하여 시스템을 구성할 경우 계산 시 운영체제 부분에 해당하는 작업에 인해 계산 성능이 저하될 수 있기에 운영체제 부분과 계산 부분의 네트워크 대역을 분리하여 구성한다.

분리 방식은 /etc/hosts 구성에서 운영체제에서 이용되는 hostname 과 계산에 이용되는 hostname 을 분리하는 방법을 이용한다.

```
# vi /etc/hosts
=========================================================================
=========
# using administration, file service

192.168.0.254           node00
192.168.0.1             node01
192.168.0.2             node02
.
.
.

# using math node - calculation, computation

192.168.10.254          real00
192.168.10.1            real01
192.168.10.2            real02
.
```

.

==========

nfs 설정 및 dua, dush등의 운영체제 관련 서비스에는 모두 nodeXX 형식의 hostname 을 사용하고 lammpi, mpich 등의 병렬 계산에 사용되는 hostname은 모두 realXX 형식으로 지정하여 실제 관리 및 파일 서비스에 이용되는 네트워크 채널과 계산에 이용되는 네트워크 채널을 분리하여 사용한다.

2.2 rsh, rlogin 설정 및 사용

rlogin, rsh 관련 서비스를 구동하기 위해서는 rsh, rsh-server 두 개의 패키지가 설치가 되어 있어야 한다. rlogin, rsh 는 MPI 와 같은 병렬 프로그램이 프로세스간 통신을 할 때 사용되는 주요 프로토콜이다. 계산 뿐만 아니라 여러 대로 구성된 HPC 시스템에서 다수의 노드를 관리 하기 위해서는 rsh 와 같은 인증 방식이 필요하다. 하지만 UNIX 시절 보안에 많은 취약점이 나타난 적이 있었고, 관리자의 부주의로 인한 보안 상의 문제도 많이 제기 되어 사용을 다소 주저하는 경우도 많다. 하지만 설정 시 보안에 신경만 써 주면 염려와 같은 문제는 발생하지 않는다. 대부분의 문제는 프로그램 자체의 취약점이 아닌 관리자의 부주의로 인해 발생되는 문제 이다. Rsh, rlogin 보안에 대한 자세한 내용은 후반부의 HPC 보안 설정 부분에서 상세히 설명하도록 하겠다.

- rsh, rlogin 설정

** xinetd 데몬에서 rsh, rlogin 서비스를 사용할 수 있도록 설정을 변경한다.

```
# vi /etc/xinetd.d/rsh
-------------------------------------------------------------------------
service shell
{
        disable = no
        socket_type             = stream
        wait                    = no
        user                    = root
        log_on_success          += USERID
        log_on_failure          += USERID
        server                  = /usr/sbin/in.rshd
```

* 리눅스 클러스터 기술 백서 (Linux Cluster technical white paper)
}

---

vi /etc/xinetd.d/rlogin

```
service login
{
        disable = no
        socket_type             = stream
        wait                    = no
        user                    = root
        log_on_success          += USERID
        log_on_failure          += USERID
        server                  = /usr/sbin/in.rlogind
}
```

설정을 변경한 후 xinetd 데몬을 restart 한다.

/etc/rc.d/init.d/xinetd restart

** 기본적으로 root 관리자 계정에서는 rsh, rlogin 사용이 금지 되어져 있다. 하지만 root 에서도 다수의 서버를 관리 하기 위해서 rsh, rlogin을 사용하는 경우가 있는데 이때는 아래 설정을 해준다.

vi /etc/securetty

```
.
..
...
rsh
rlogin
```

** rsh, rlogin 서비스를 허용할 hosts 와 users 설정을 한다.

vi /etc/hosts

```
127.0.0.1               localhost

# 운영체제 네트워크 대역
192.168.10.254          master
192.168.10.1            node01
192.168.10.2            node02
192.168.10.3            node03
192.168.10.4            node04
192.168.10.5            node05
192.168.10.6            node06
.
# 계산 네트워크 대역
192.168.20.254          real00
192.168.20.1            real01
192.168.20.2            real02
192.168.20.3            real03
192.168.20.4            real04
192.168.20.5            real05
192.168.20.6            real06
```

** 시스템의 모든 일반 유저 계정에서 rsh, rlogin을 사용하기 위해서는 아래 설정을 해준다.

vi /etc/hosts.equiv

```
master
node01
node02
node03
.....
real00
real01
real02
.....
```

** 만일 특정 계정에서만 rsh, rlogin 서비스를 이용할 경우는 각 계정의 홈폴더에 .rhosts 란 파일을 만들고 아래 설정을 한다.

```
# vi $HOME/.rhosts
-------------------------------------------------------------------------
master
node01
node02
node03
.....
Real00
Real01
Real02
......
-------------------------------------------------------------------------
```

.rhosts 설정에서는 각 노드 간의 rsh 접속이 동일한 계정 사이에서만 기본적으로 이루어 진다. 만일 clunix 계정으로 root 의 권한으로 rsh 접속을 하기 위해서는 root 의 .rhosts 에 아래와 같이 해 주어야 한다.

```
# vi /root/.rhosts
-------------------------------------------------------------------------
master
node01
node02
node03
node04
node05
node06
master          clunix
node01          clunix
node02          clunix
node03          clunix
node04          clunix
node05          clunix
node06          clunix
```

.rhosts 와 /etc/hosts.equiv 파일은 보안 상 중요한 파일임으로 반드시 퍼미션을 644 혹은 600으로 변경을 해주도록 한다.

chmod 600 $HOME/.rhosts
chmod /etc/hosts.equiv

- rsh 사용 방법

*** rsh 사용 방법
usage: rsh [-l username] host [command]

*** rsh를 이용한 접속 법

[root@client root]# rsh server
Last login: Fri Jul 26 15:34:43 from tech
You have new mail.
[root@server root]#

*** remote shell 사용법

[root@client root]# rsh server ps
  PID TTY   TIME CMD
    1 ?   00:00:04 init
    2 ?   00:00:00 keventd
    3 ?   00:00:00 kapmd
    4 ?   00:00:00 ksoftirqd_CPU0
    5 ?   00:00:00 kswapd
    6 ?   00:00:00 bdflush
    7 ?   00:00:00 kupdated
    8 ?   00:00:00 mdrecoveryd
   12 ?   00:00:00 kjournald
   91 ?   00:00:00 khubd
  247 ?   00:00:00 kjournald

위의 "rsh server ps" 명령어는 client host 에서 server host로 접속하지 않고도

server host 의 process check 명령어 ps를 이용하는 예제 이다.

2.3 SSH 설정

Rsh 계열 프로토콜의 보안적인 문제에 대한 거부감에 rsh 사용을 거부 하는 경우도 많다. 이런 경우 보안적 측면에서의 안정성이 강한 SSH를 패스워드 인증 방식이 아닌 인증키를 이용한 인증 방식으로 구축 하면 된다. RSH 가 무조건 보안에 취약한 것은 아니지만..사용자의 눈 높이에 맞추는 차원에서는 초기 구축 시 SSH를 이용하는 방식을 권장한다. 설정 방법은 아래와 같다.

ssh-keygen 으로 공개인증키를 생성하면 실제 시스템 패스워드를 이용하여 접속하는 것이 아니라
인증키 패스워드를 이용하여 접속이 가능하다.

인증키 생성시 패스워드를 공백으로 처리하면 전 노드를 ssh로 접속할 때 암호 없이 접속이 가능하다.

** node01 설정

[root@otn01 root]# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): - 엔터
Enter passphrase (empty for no passphrase):
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
8f:c6:f2:05:69:f4:2f:f0:35:b1:57:ac:ac:2b:ea:df root@otn01

그럼 /root/.ssh/id_rsa.pub 파일이 생성된다. 이 파일의 내용을 보면 아래와 같다.

ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAIEA0lAeGFDgSDPMJNeydJiWSuT2XVc7YO+i1o₩
6qSStY67jVHqxkCkvMJbWkohT0cLDJB9NAqjpVUBc8U7l/g6uSe9VyV/Aq2kuvAWTwpajZ₩
9B1XomvCauA1kahv3xESziYhJjEfKBdSRiDmlcKTOFcpKP0Z2Akyx+83N2RbhaC+mAU= root@otn01

이 내용을 node01 에서 접속할 노드의 /root/.ssh/authorized_keys 파일안에 복사해
두면된다.

클러스터 시스템처럼 여러 노드가 존재할 경우는 일단 모든 노드에 ssh-keygen 를
이용하여 공개 인증키를 생성한 후 각 노드의 공개 인증키 내용을 순차적으로 하나
의 authorized_keys 로 생성하고 이 키를 전 노드에 /root/.ssh 밑에 동기화 시켜

주는 방식으로 구현하면 노드 간의 ssh 접속이 암호 없이 가능하게 된다.

2.4 Dutils 설치 및 설정

dua, dush 는 Encluster-HPC 의 다중 서버 관리 도구이다. 즉 여러 대의 서버에 file 및 작업 명령을 일괄적으로 처리하도록 해주는 프로그램이다.
dua, dush 를 정상적으로 사용하기 위해서는 앞에서 설명한 rsh, rlogin 서비스 설정이 완료되어 있어야 한다.

먼저 Master Node 에 dutil-1.2.1-1.noarch.rpm 를 설치 한다.

rpm -Uvh dutil-1.2.1-1.noarch.rpm

실제 dua, dush 를 이용하여 일괄 관리 할 서버 리스트를 작성한다.

vi /usr/clx/etc/nodelist
---
master
node01
node02
node03
node04
node05
node06

** dua : file 을 nodelist 에 포함된 모든 Node 에게 일괄적으로 sync 시키는 도구
** dush : nodelist 의 포함된 node 에게 일괄적으로 내려진 작업을 수행한다.

공통 옵션 설명 :

-l : /usr/clx/etc/nodelist 이외의 다른 nodelist 를 참조할 경우 -l 옵션으로 호출 할수 있다.

예) dua -l /etc/nodelist /root

-n : nodelist 에 포함된 모든 node 가 아닌 특정 node 에만 작업을 수행할 경우 사용됨

예) dua -n node03 /root

-p : 작업을 순차적으로 처리하는 것이 아닌 backgroud mode 로 동시에 작업이 수행된다

예) dush -p "/etc/rc.d/init.d/ecmctl restart"

-s : 노드간 수행 결과를 구분하기 위한 ###executing..." 이란 메세지가 나타나지 않고
수행 결과만을 출력해 주는 옵션이다.

-d : dua 에서 사용되는 옵션으로 디렉토리간의 완전한 sync 를 할때 사용된다.
즉 master 와 node02 의 디렉토리 내용을 완전히 sync 시킬때 사용되는 옵션으로
그냥 # dua /root 라 수행하면 master 의 /root 내용이 node02 의 /root 로 파일이
복제 되지만 # dua -d /root 라 수행하면 node02가 master 보다 더 많은 파일을
가지고 있다면 master 에 없는 파일은 모두 삭제해 버리고 완전 master 와 동일한
자료만을 가지게 된다.

2.5 NFS 설정

MPI 병렬 프로그램 사용 시 모든 계산 서버의 같은 PATH에 동일한 실행 파일이 존재 해야 한다. 이런 이유로 인해 노드 별 별도의 하드 디스크를 가지고 있는 경우 dua 와 같은 디스크 동기화 프로그램을 사용하거나 PVFS, GFS 같은 클러스터 파일 시스템을 사용한다. 이밖에 경제적, 관리적 이슈로 인해 NFS 와 같은 방식으로 특정 서버의 디스크를 공유하여 방법도 많이 사용한다.
여기서는 NFS 설정 방법과 효율적인 관리 방법에 대해 다루도록 한다.

- NFS 기본 설정

vi /etc/exports

NFS 원격 설치 시 사용되는 경로

```
/data/clunix/teragon        *(ro,soft,no_root_squash)
# 사용자 Data 보관에 이용되는 경로
/data/users                                    *(rw,soft,no_root_squash)
```

```
# exportfs -a
# /etc/rc.d/init.d/portmap restart
# /etc/rc.d/init.d/nfs restart
```

이것을 HPC에 적절한 Nfs 설정이 완료된다.

```
# mount -t nfs master:/data/clunix/teragon   /mnt/teragon
# mount -t nfs master:/data/users   /mnt/users
```

등의 잘 nfs mount 가 되는 지를 확인한다.

- nfs 설정 후 사용자에게 nfs 디스크를 할당 해 주는 방안..

홈 디렉토리 자체를 nfs 상에 만들면 nfs 에 문제가 있을 때 로그인이 될 수가 없다.
이를 피하기 위해서는 홈 디렉토리는 로컬 하드 상에 만들고 홈 디렉토리 밑에 data 파일을 nfs 디스크 상에 링크 형태로 만들어 대용량 파일을 이곳에 보관하는 방식으로 설정한다.

먼저 /nfs 디렉토리의 퍼미션을 1777 로 하여 자신의 파일에만 제어 권한을 준다.

```
# chmod 1777 /nfs
```

```
# vi /etc/skel/.bashrc
```

```
userid=`whoami`
if [ ! -d /nfs/$userid/ ]
then
mkdir /nfs/$userid
ln -sf /nfs/$userid $HOME/data
chown -R $userid. /nfs/$userid
```

```
else
ln -sf /nfs/$userid $HOME/data
rm -f $HOME/data/$userid
fi
```

위 스크립터에서 /nfs 는 실제 nfs 디렉토리라고 생각해야 한다.

```
[root@node00 root]# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/hda3             10080520    148852   9419600   2% /
/dev/hda1               202220     14518    177262   8% /boot
none                    514940         0    514940   0% /dev/shm
/dev/hda2             10080520   1547244   8021208  17% /usr
/dev/hda6              2016016    200712   1712892  11% /var
/dev/hda7              9775216    512684   9262532   6% /usr/local
/dev/hda8             43534780     32848  43501932   1% /home
/dev/sda5            234429320     32840 234396480   1% /data1
node00:/data1        234429320     32840 234396480   1% /nfs
```

와 같이..

사용자 계정을 만들고 처음 로그 인을 하면 nfs 상에 자신의 이름으로 디렉토리
를 만든다. 그런 후 자신의 홈 디렉토리 밑에 data 란 파일로 링크를 건다.

만일 /nfs 파일 밑에 이미 자신의 이름과 동일한 디렉토리가 있으면 ..
그냥 자신의 퍼미션으로 퍼미션 변경만을 하게 된다.

- automount 를 이용하여 사용자에게 nfs 공간을 할당 하는 방안

automount 에는 am-utils (amd), autofs 두 가지의 패키지가 있다.
Redhat9 에서는 기본적으로 autofs 를 채택하고 있다.

간단한 설정에 대해 알아보자.

주요 설정 파일은 ..

/etc/auto.master
/etc/auto.misc

이 있다.

```
# vi /etc/auto.master
=========================================================================
==
/home              /etc/auto.user01     --timeout=5

# vi /etc/auto.user01
=========================================================================
==
user01/data              -fstype=nfs,rw,soft,bg          master:/data/nfs/user01

# /etc/rc.d/init.d/portmap restart
# /etc/rc.d/init.d/autofs restart
```

/home/user01/data 로 이동하면 자동으로 master 의 /data/nfs/user01
로 nfs mount 가 된다.

automount 를 사용하게 되면 실제 사용자가 지정된 mount point 에 접근 시에만 자동으로 nfs mount 하고 일정 시간 사용을 하지 않으면 자동으로 umount 를 시키기 때문에 nfs 서비스로 인한 resource 를 최대한 절약 할 수 있고, 여러명이 지속저으로 Nfs 를 사용할 경우 1개의 곳에서 lock 이 결려도 전체 서비스에 문제가 생길 가능성이 있는데 automount 로 상당 부분 해소할 수 있다.

autofs-4.x 버전에서는 multiple hostname 기능이 지원한다.
이는 먼저 master 로 Connection 요청을 해서 정상적인 요청이 이루어 지지 않으면 slave 에 연결이 된다. Redhat9 에서는 기본적으로 autofs-3.x 임으로 autofs-4.x 로 업그레이드를 해야 한다.

http://www.kernel.org/pub/linux/daemons/autofs/

- NFS mount 상세 옵션 설명

NFS Clients 에서의 nfs.mountd option으로 soft,hard,timeo,rsize,wsize... 등등 많은 option 이 있는데 이 중에서 가장 최적화된 옵션을 찾아 되는 것은 남겨진 숙제이다.

주요 옵션 설명이다.

rsize=n NFS 서버로부터 읽어들이는 바이트 수를 지정 한다. 기본값은 커널에 따라다른데 현재로서는1024 바이트이다.

wsize=n NFS 서버에 쓰기를 할 때 사용하는 바이트 수 를 정한다. 기본값은 커널에 따라다른데 현재 로서는 1024 바이트이다.

timeo=n RPC 타임아웃이 생기고 나서 첫번째 재전송 요 구를 보낼 때 사용되는 시간으로서 1/10 초 단 위이다. 기본값은 7 * 1/10 초이다. 첫번째 타 임 아웃이 생기고 나서는 타임아웃 시간이 최대 치인 60 초에 이르거나 너무 많은 재전송 요구 가 벌어 질 때 까지 타임아웃 시간이 2 배로 변화 한다. 만약 파일 시스템이 hard (hard 옵션을 참고) 마운트 되어있다면 각각의 타 임 아웃 시간은 2 배로 증가하고 재전송 시도 가 일어날 때도 2 배로 증가한다.
최대 타임아웃 시간은 60 초이다. 네트웍 속도가 느리거나 서버 자체가 느리다든지 여러 개의 라우터와 게 이트웨이를 거칠 때는 마운트 당시 타임 아웃 시간을 늘려 주는 것이 좋다.

retrans=n 주 타임아웃을 발생시키는 부 타임아웃과 재전 송 횟수를 정한다. 기본값은 3 번의 타임아웃이다. 주 타임 아웃이 일어나면 화일 작업이 중지 되거나 콘솔 상에 "서버가 반응하지 않음 "server not responding""이라는 메시지가 출력 된다.

retry=n 백그라운드에서 진행 중인 NFS 마운트 작업 이 포기하기 전까지 실행할 횟수를정한다. 기본값 은 10000 번이다.

namlen=n NFS 서버가 RPC 마운트 프로토콜의 버전 2 를 지원하지 않을 때 원격 파일 시스템에서 지원 되는 파일 명의 길이를 명시한다.
POSIX pathconf 함 수 를 지원 하기 위해서 사용된다.
기본값은 255 개의 문자이다.

port=n NFS 서버와 연결할 수 있는 포트 번호를 정 한다. 만약 0 이라면(기본값) 원격 호스트의 포트매퍼(portmapper) 에게 질의 하여 알아내도록 한다.
만약 포트매퍼에 NFS 데몬이 등록되어 있지 않은 경우에는 2049 라는 표준 NFS

포트 번호가 사용된다.

mountport=n mountd 포트 번호 지정하기.

mounthost=name mountd 를 실행 중인 호스트 명을 정한다.

mountprog=n 원격 호스트의 마운트 데몬과 접속하기 위해 사용할 수 있는 별도의 RPC 프로그램번호를 정 한다. 만약 여러 개의 NFS 서버를 운영하고 있을 때 사용한다. 기본값은 표준 RPC 마운트 데몬 프로그램 번호인 100005 이다.

bg 만약 첫 번째 NFS 마운트 시도가 타임아웃 걸리 면 백그라운드에서 실행을 계속 된다. 기본 값은 백그라운드로 실행하지 않고 그냥 포기한다.

fg 첫번째 NFS 마운트 시도에서 타임아웃이 걸 리 면 그 즉시 포기 해 버린다. 기본값이다.

soft NFS 화일 작업에서 주 타임아웃이 걸리면 프로 그램에게 I/O 에러를 보고한다. 기본값은 무한히 NFS 화일 작업을 재시도 하는 것이다.

hard NFS 화일 작업에서 주 타임아웃이 걸리면 콘솔 상에 "server not responding", "서버가 반응 하지 않음" 이라고 출력하고 무한히 재시도한다. 이것이 기본값이다.

intr 주 타임아웃이 생기고 하드 마운트 된 상태라면 화 일 작업을 중지하도록 시그널 을 보내도록 허용하고 EINTR 시그널을 보내다. 기본값은 화일 작업을 인터럽트 하지 않는 것이다.

tcp NFS 화 일 시스템을 기본값인 UDP 가 아니라 TCP 프로토콜을 사용하여 마운트 하도록 한다. 많은 NFS 서버들이 오로지 UDP 만을 지원한다.

udp NFS 화일 시스템을 UDP 프로토콜로 마운트 한다. 기본값.

2.6 NIS 계정 통합 시스템 설정

NIS 서버를 이용하여 통합 계정 인증 시스템을 만드는 대표적인 방법은 크게 정통적인 Sun 의 yellow page 에서 유래된 yp 와 ldap project 에서 나온 openldap 을 이용하는 방법이 있다.

openldap 의 경우에는 다른 application 과의 계정 연동 부분도 많이 지원 하고 있는 만큼 그 크기나 깊이가 아주 광범위 하다고 볼 수 있다.
여기서는 간단히 ypserv, ypbind 를 이용한 NIS로 통합 계정 인증 시스템을 구현하는 방법에 대해 알아보겠다.

- 서버,클라이언트 공통 설정 부분

일단 yp 관련 클라이언트 tool 이 설치 되어져 있어야 한다.

```
# rpm -Uvh ypbind-x.x.x.rpm
# rpm -Uvh yp-tools-x.x.x.rpm
# rpm -Uvh ypserv-x.x.x.rpm      -> 계정 서버일 경우에 설치
```

/etc/host.conf 파일에 multi on 설정을 추가 한다.

```
# vi /etc/host.conf
---------------------------------------------------------------------
order hosts,bind
multi on
---------------------------------------------------------------------
```

NIS 도메인 이름을 결정한다. (DNS 의 도메인 과 NIS 도메인은 다른 차원의 것이다.)

```
# nisdomainname < NISDOMAIN >
```

/etc/sysconfig/network 에 NISDOMAIN 내용을 추가 한다.

```
# vi /etc/sysconfig/network
---------------------------------------------------------------------
NETWORKING=yes
HOSTNAME=test03
GATEWAY=192.168.123.254
NISDOMAIN=< NISDOMAIN >
---------------------------------------------------------------------
```

- 서버 설정

그런후 NIS 시스템에서 shadow file 을 인식 할수 있게 설정을 변경한다.
/var/yp/Makefile 을 열어서 all: 로 문자열을 검색하면

```
# vi /var/yp/Makefile
-----------------------------------------------------------------
.
.
all: passwd group hosts rpc services netid protocols mail ₩
     # publickey shadow netgrp networks ethers bootparams printcap ₩
     # amd.home auto.master auto.home auto.local passwd.adjunct ₩
     # timezone locale netmasks
```

나온다. 여기서 주석내용중의 shadow 를 주석 밖으로 빼내준다

```
all: passwd group hosts rpc services netid protocols mail shadow ₩
     # publickey netgrp networks ethers bootparams printcap ₩
     # amd.home auto.master auto.home auto.local passwd.adjunct ₩
     # timezone locale netmasks
```

그런 후 ypserv 와 yppasswd 를 시작 해 준다. 기본적으로 yp 는 rpc 를 이용하는 서비스 임으로 portmap 을 먼저 실행하여야 한다.

```
# /etc/rc.d/init.d/portmap start
# /etc/rc.d/init.d/ypserv start
# /etc/rc.d/init.d/yppasswdd start
# chkconfig --level 345 portmap on
# chkconfig --level 345 ypserv on
# chkconfig --level 345 yppasswd on
```

계정 인증 서버의 계정 정보를 클라이언트 서버로 전달 가능한 Map 으로 만든다.

```
# make -C /var/yp
```

위 단계가 바로 인증 서버의 계정 정보를 다른 서버들과 공유 가능한 계정 정보로 만드는 작업니다. 그렇기에 인증 서버의 계정 정보의 변경이 있는 경우 반드시 위 과정을 실행해

야 클라이언트에서 변경된 계정 정보를 인식 할수 있다.

- 클라이언트 설정

클라이언트에서는 ypserver 의 계정 정보를 rpc 를 통해 사용하기 위해서 계정 서버의 정보를 쿼리를 할수 있는 ypbind 같은 프로그램이 필요하다.

ypbind 설정은 /etc/yp.conf 를 수정하면 된다.

```
# vi /etc/yp.conf
---------------------------------------------------------------------------
ypserver < NISDOMAIN server for HOSTNAME >
domain < NISDOMAIN server for HOSTNAME >
---------------------------------------------------------------------------
```

그런 뒤 ypbind 를 실행한다. ypbind 역시 portmap 이 먼저 실행 되어야 한다.

```
# /etc/rc.d/init.d/portmap restart
# /etc/rc.d/init.d/ypbind restart
```

그런 후 /etc/passwd, /etc/group 설정에서 nis 로 계정을 인증 하겠다는 형식이 추가 되어야 한다.
/etc/passwd 파일의 제일 하단에 < +:*:0:0::: > 을 추가해 준다.
/etc/group 파일의 제일 하단에 < +:*:0:0: > 을 추가해 준다.

```
# vi /etc/nsswitch.conf
---------------------------------------------------------------------------

.
.
passwd:       files
shadow:       files
group:        files
```

위 내용을 ..아래로 변경 ..

```
passwd:       files nisplus nis
shadow:       files nisplus nis
```

group: files nisplus nis

클라이언트 설정 완료 후 클라이언트 시스템은 리부팅을 해주어야 한다. nsswitch 설정 적용을 위해서~

이제 위 내용으로 계정 연동이 완료되어 진다. 즉 ypserv 에 계정을 생성하고 make -C /var/yp 로 생성 내용을 갱신 하면 ypserv 에 연동된 클라이언트 시스템으로 접속하여 새로 생성된 계정으로 ssh 나 telnet 접속을 하면 성공하여야 한다.

정상적으로 동작하는 것을 확인 하는 방법으로는 yp-tools 에 있는 명령어를 이용하면된다.

yptest
ypcat passwd

위 명령을 사용하면 실제 ypserv 측의 계정 정보와 패스워드 정보들이 출력될것이다.

클라이언트 서버에서 ypserv 계정의 패스워드 변경을 할때는 yppasswd 명령을 사용한다.

yppasswd < userid >

이로서 ypserv 를 이용한 계정 통합 인증 시스템에 대해 알아 보았다. yp 는 간단한 구성의 시스템 인증에 많이 사용되어진다. 복잡한 어플리케이션 인증 시스템으로는 yp 보다는 openldap 으로 구성하는것이 효율적일것이다.

yp 로 구성하면 앞에서 설명했듯이 ypserv 에서의 계정 정보가 변경되면 반드시 /var/yp 의 map 을 재 갱신 시켜주어야 한다. 그렇기에 실시간으로 계정 정보를 인식하지는 못한다.

어플리케이션 인증 측면에서는 이해 할수 없겠지만 OS 차원에서의 계정 인증에서는 그렇게 문제가 되는 부분은 아니라고 생각한다. 계정 정보를 변경하고 갱신 명령만 한번 실행하면 수십대의시스템의 계정을 한대의 서버에서 관리 할수 있기 때문에 상당히 효율적이라고 볼수 있다.

이런게 귀찮은 경우는 별도의 계정에 관련된 명령어 (adduser, useradd, userdel, passwd)등을 yp 관련 명령어와 적용하여 scripts 를 만들거나 아님 cron 등에 짧은 주기로 등록해 두면 효과적일거라 생각한다.

이런 관리적인 측면 말고 yp 나 ldap 같은 계정 인증 통합 시스템에서는 반드시 사용자의 홈 폴더가

로컬디스크 상에 있으면 안 되는 점이 있다. 즉 사용자 홈 폴더는 NFS 등으로 하나로 통합해야 하는 주의점이 있다. 즉 계정은 계정 서버를 통해 인증되어 접속을 가능하지만 홈폴더가 계정 서버 자체에만 생성이 됨으로 현재 접속하는 서버에 홈 폴더가 없기에 문제가 발생할 수 있다. 이러한 부분 때문에 홈 폴더가 NFS 등으로 구성된 시스템에서는 yp를 이용한 NIS 서버를 사용하고 홈 폴더가 시스템 별로 분리된 시스템 구성에서는 3장에서 설명할 HPC Management Tool의 분산 시스템 일괄 계정 관리 프로그램을 이용하여 계정을 관리하면 될 것이다.

3. Teragon HPC 설정 하기

3.1 Time Sync 설정

클러스터 시스템에서는 노드 간의 시스템 시간이 일치해야 하는 경우가 있다. 즉 병렬 프로그램에서 프로세스의 시간을 기준으로 많은 프로그램을 하게 되는데 이가 일치 하지 않으면..프로세스간 통신 상에 문제가 발생할 수 있다. 그렇기에 HPC 시스템에서는 구성 시스템 모두가 같은 시간을 유지해 줘야 한다. Master 노드를 시간 서버로 정하고 나머지 계산 노드들이 Master 노드의 시간과 일치 하도록 해주는 설정이다.

- master node 에 time 서비스를 켠다.

"disable = yes"를 "disable = no" 로 변경해 준다.

vi /etc/xinetd.d/time

```
==========================================================================
=========
service time
{
        disable = no
        type            = INTERNAL
        id              = time-stream
        socket_type     = stream
        protocol        = tcp
        user            = root
        wait            = no
}
==========================================================================
=========
```

Xinetd 데몬을 재시작하여 설정을 적용시킨다.

```
# /etc/rc.d/init.d/xinetd restart
```

- master node 의 시간을 표준 시간으로 맞춘다.

```
# hwclock -w
# rdate -s time.bora.net
```

- read node 에서 master node 시간과 일치 시킨다.

```
# rdate -s master
```

- master node 에서 시간이 일치하는지 확인한다.

```
# dush date
```

- master node 에서 time sync scripts를 하나 만들고 cron에 매 시간 마다

한번 씩 실행 시켜 준다.

시스템 시간을 BIOS 표준 시간과 일치 시킨다. 그런 후 표준 원자 시간 서버에 시간을 맞춘다.
dush를 이용하여 전 계산 노드에서 master 노드의 시간과 일치 시킨다.

```
# vi /root/bin/timesync
=========================================================================
=========
#!/bin/sh

hwclock -w
rdate -s time.bora.net

/usr/clx/sbin/dush rdate -s master_node
=========================================================================
=========
```

Dush를 이용하여 전 노드의 data 정보를 Master 노드에서 출력하도록 한다.

```
# vi /root/bin/timeview
=========================================================================
=========
#!/bin/sh

/usr/clx/sbin/dush –s date
=========================================================================
=========

# /root/bin/timeview
=========================================================================
=========
[node00] Tue May 11 18:49:39 KST 2004
[node01] Tue May 11 18:49:39 KST 2004
[node02] Tue May 11 18:49:39 KST 2004
[node03] Tue May 11 18:49:39 KST 2004
[node04] Tue May 11 18:49:39 KST 2004
[node05] Tue May 11 18:49:39 KST 2004
```

Cron 에 등록하여 매 시간 마다 시간을 일치 시킨다.

```
# crontab -e
================================================================================
0 * * * * /root/bin/timesync

================================================================================

/etc/rc.d/inint.d/crond restart
```

3.2 Compiler 설치 및 설정

주요 병렬 프로그램은 fortran 이나 C 로 만들어진다. 리눅스에서는 기본적으로 Gnu C/FC를 지원 하지만 최대 성능을 위해 상용 컴파일러를 많이 사용한다. 대표적으로 사용하는 컴파일러로는 Intel Compiler 와 Portland Group Compiler 가 있다. 여기서도 이 두 가지 컴파일러를 설치 하는 방법에 대해 설명한다.

- intel fortran/c++ Compiler 설치

intel Compiler 설치 방법은 모두 동일하다. 주의할것은 .lic 파일을 설치 디렉토리
에 넣어두시고 install.sh을 실행하면 된다.

ftp://clunix@technet.clunix.com/Hpc/intel

```
# tar xvf l_fc_p_8₩[1₩].0.055.tar
# mv l_cpp_63460285.lic l_cc_p_8.0.055
# cd l_cc_p_8.0.055
# ./install.sh

// compiler 설치 경로 -> /usr/local/intel/fc

# vi /etc/profile.d/intel.sh
================================================================================
```

```
#!/bin/sh
PATH=$PATH:/usr/local/intel/fc/bin
export PATH
================================================================================
=========
```

혹은 /usr/local/intel/fc/bin/ 및에 ifortvar.sh 파일이 존재한다.
이것을 /etc/profile.d/ 밑에 복사해 두면 편리하다.
icc의 경우에도 fortran과 설치 방법은 동일하다. 설치 경로만을 /usr/local/intel/cc 로 적용하면된다.
환경설정은 iccvar.sh 파일을 복사해 두면 된다.

- PGI Compiler 설치하기 (x86_64용)-(IA32도 동일)

ftp://clunix@technet.clunix.com/Hpc/pgi-amd64
ftp://www.pgroup.com

****** PGI 설치 전 주의 사항 ********

PGI 설치 전에 Kernel Upgrade 가 필요한 경우엔 반드시 Kernel Upgrade 이후 PGI를 설치 하도록 하여라. 아님 GLIBC Version 문제로 인해 문제 발생 할 수도 있다.

```
# cd /usr/local/src/pgi-amd64/
# tar xzvf linux86-64[1].tar.gz
# ./install

================================================================================
=========
End-User License

NOTICE: PLEASE READ THIS DOCUMENT CAREFULLY BEFORE DOWNLOADING,
COPYING OR USING THE SOFTWARE.  THIS END-USER LICENSE AGREEMENT
("ELA") IS A LEGAL AGREEMENT BETWEEN YOU, THE LICENSEE (A SINGLE
PERSON, INSTITUTION, OR LEGAL ENTITY), AND STMicroelectronics, INC. A
```

DELAWARE CORPORATION WITH ITS PRINCIPAL PLACE OF BUSINESS AT 1310
ELECTRONICS DRIVE, CARROLTON, TX 75006 ("ST") FOR THE SOFTWARE,
ASSOCIATED MEDIA, PRINTED MATERIAL, ELECTRONIC DOCUMENTATION OR ANY
PORTION THEREOF. BY INSTALLING AND USING THE SOFTWARE ACCOMPANYING
THIS ELA INDICATES YOUR ACCEPTANCE OF THESE TERMS AND
CONDITIONS. PLEASE NOTE THAT YOU MAY NOT USE, COPY, MODIFY OR TRANSFER
THE SOFTWARE OR DOCUMENTATION OR ANY COPY, EXCEPT AS EXPRESSLY
PROVIDED IN THIS ELA. IF YOU DO NOT AGREE TO THE TERMS AND CONDITIONS

.

.

Address:

The Portland Group Compiler Technology
STMicroelectronics
9150 SW Pioneer Court, Suite H
Wilsonville, OR 97070
USA

Do you accept these terms? [accept,decline]

-> **accept 를 입력한다.**

This release of PGI software includes the ACML, which is a tuned
math library designed for high performance on AMD64 machines,
including Opteron(TM) and Athlon(TM) 64, and includes both 32-bit
and 64-bit library versions.

More information about the ACML can be found at the ACML web site:
http://www.developwithamd.com/acml

Install the ACML? [y/n]

-> **y 를 입력한다.**

---

following license.
LICENSE AGREEMENT
AMD CORE MATH LIBRARY

IMPORTANT: This is a legal agreement ("Agreement") between you, either as an individual or an entity, (the "USER") and Advanced Micro Devices, Inc. ("AMD"). By loading the software or any portion thereof ("Software"), and any related documentation ("Documentation"), USER agrees to all of the terms of this Agreement. Additionally, USER remains subject to the original terms and conditions of any other software license agreements entered into by USER and a third party. USER is responsible for ensuring that use of the Software provided by AMD is not in violation of any such agreement.

.

.

Do you accept these terms? [accept,decline]

-> accept 를 입력한다.

Please specify the directory path under which the software will be installed. The default directory is /usr/pgi, but you may install anywhere you wish, assuming you have permission to do so.

Installation directory? [/usr/pgi] **/usr/local/pgi**

-> 그냥 Enter 혹은 PGI를 설치할 경로를 입력한다.

Installing software into /usr/local/pgi (this may take some time).

##

If you don't already have permanent keys for this product/release, a fifteen-day evaluation license can be created now.

Create an evaluation license? [y/n]

-> y 를 입력한다. (evaluation license 생성할건지 물어봄..) -> 15일 사용가능

.

.

PGI Software: PGI Fortran/C/C++ compilers and tools for 32-bit x86
and 64-bit AMD64 processor-based computer systems.

Do you accept these terms? [accept,decline]

-> accept 를 입력한다.

Creating temporary license.
Please enter your name: root **-> root 를 입력한다**.
Please enter your user name: root **-> root 를 입력한다.**
Please enter your E-mail address: root@localhost **-> root를 입력한다.**

You have entered the following information:

name	root
user name	root
E-mail address	root@localhost

Do you wish to change anything? [yes/no]: -> no

The above information was saved to /usr/local/pgi/license.info.

Do you want the files in the install directory to be read-only? [y,n]

-> y 를 선택하면 됩니다.

---

cd /usr/local/pgi 가셔서 정상적으로 설치가 완료되었는지를 확인함.

******* 라이센스 발급 주의 사항 **

evaluation license 에서 정식 license upgrade 계획이 있을 경우에는 반드시
일반 User로 evaluation license information 을 만들도록 하라.

HPC 의 MPI 로 많이 사용되는 Lammpi 의 경우 root 로는 사용할수 없다.
만일 root 로 PGI 사용권한이 주어지면 root 계정에서 compile 을 하고 다시
일반 계정으로 lam 을 돌려야 하는 번거로움이 생긴다. 초기 계획에서 잘 고려
해서 결정하도록 하라.

****

그런후 반드시 /etc/profile.d 밑에 다음 환경 설정을 넣어 둠.

```
# vi pgi.sh
=========================================================================
=========
export PGI=/usr/local/pgi
export PATH=$PGI/linux86/5.1/bin:$PATH
export MANPATH=$MANPATH:$PGI/linux86/5.1/man
#export LM_LICENSE_FILE= $PGI/license.dat
=========================================================================
=========

# cp pgi.sh /etc/profile.d
# source /etc/profile.d/pgi.sh
# pgf77 test.f
=========================================================================
=========
NOTE: your evaluation license will expire in 14 days, 23.9 hours.
For a permanent license, please read the order acknowledgement
that you received.  Connect to https://www.pgroup.com/License with
the username and password in the order acknowledgement.

        Name:    root
        er:    root
        Email:   root@localhost
        Hostid: PGI=000C76E0827FB855DE54B8
```

PGFTN-F-0002-Unable to open source input file: test.f

==========

같이 메세지가 떨어지면 정상적으로 설치가 되어진것이다.

간단한 테스트를 해보기 위해서는 소스에서 제공하는 샘플 소스를 컴파일
해 볼수 있다.

```
# cd /usr/local/pgi/linux86/5.1/EXAMPLES/linpack/UNIX
# make
# ./linpkrd
```

==========

```
norm. resid          resid                machep            x(1)-1          x(n)-1
   1.89264926E+00   8.39915160E-14   2.22044605E-16 -6.22835117E-14 -4.16333634E-14


times are reported for matrices of order     100
         sgefa         sgesl         total        Kflops          unit         ratio
times for array with leading dimension of 201
      0.00115       0.00004       0.00119       578917.       0.00345       0.02118
      0.00114       0.00004       0.00118       582486.       0.00343       0.02105
      0.00112       0.00004       0.00116       589944.       0.00339       0.02078
      0.00117       0.00000       0.00117       585947.       0.00341       0.02093

times for array with leading dimension of 200
      0.00113       0.00004       0.00117       587926.       0.00340       0.02086
      0.00113       0.00004       0.00117       586895.       0.00341       0.02089
      0.00113       0.00004       0.00117       587817.       0.00340       0.02086
      0.00117       0.00000       0.00117       585409.       0.00342       0.02095

ROLLED DOUBLE    PRECISION LINPACK PERFORMANCE          585409 KFLOPS
FORTRAN STOP
```

==========

그럼 이 시스템의 CPU Flops를 체크하는 linpack 체크를 할 수 있다.

3.3 계산용 Library 설치

기본적인 blas와 lapack 은 Redhat 에 기본 탑재 되어져 있는 것을 사용하면 된다.

rpm -Uvh blas-x.x.x.rpm
rpm -Uvh lapack-x.x.x.rpm

- ATLAS 설치 하기

ATLAS (Auto Tuning Linear Algebra Subprogram) 는 수치 연산에 사용되는 BLAS 라이브러리를 해당 시스템(CPU, Compiler) 에 맞게 자동으로 Tuning 하여 최적의 라이브러리 환경을 만들어 주는 프로그램이다. 기본적으로 사용하는 blas 라이브러리에 비해 월등하기 때문에 Linpack Benchmarking Test 를 할경우에도 위의 라이브러리를 Link 하여 사용한다.

ATLAS 를 설치 할때 고려해야 할것은 Compiler 이다. 실제 사용 컴파일러를 가지고 있는 경우는 상용 컴파일러를 이용하여 라이브러리를 만드는 것이 성능에 유리하다. 하지만 Compiler 별로 설치가 되는 경우와 그렇지 못한 경우가 있어 이는 각 시스템의 상황에 따라 적절히 고려 해야 한다.

여기서는 PGI 컴파일러와 Opteron 프로세서 환경에서 컴파일 하는 것에 대해 알아 보겠다.

먼저 최신 atlas를 다운 받는다.

http://math-atlas.sourceforge.net

[root@otn02 hpc]# tar xzvf atlas3.7.8.tar.gz
[root@otn02 hpc]# cd ATLAS/
[root@otn02 ATLAS]# make config CC=<ANSI C compiler>
(if you have other Compiler. Default is gcc Compiler)
make config CC=pgcc
==========
========
.
.

```
011
010
009
008
007
006
005
004
003
002
001
Enter number at top left of screen [0]: < enter >
================================================================================
=========
                                        IMPORTANT
================================================================================
=========
Before going any further, check
    http://math-atlas.sourceforge.net/errata.html
This is the ATLAS errata file, which keeps a running count of all known
ATLAS bugs and system problems, with associated workarounds or fixes.
IF YOU DO NOT CHECK THIS FILE, YOU MAY BE COMPILING A LIBRARY WITH KNOWN BUGS.

Have you scoped the errata file? [y]:
.
.
ATLAS can detect almost everything it needs to know, so choosing the default
or 'Other/UNKNOWN' will at worst simply extend the install time (if you tell
config the answer to something ATLAS can skip some tests).

Configure makes no changes to the state of things until all questions have
been asked and answered.   Therefore, if you get confused and want to start
over, feel free to break out of this program (CTRL-C, CTRL-BREAK, etc)
and start again.   Alternatively, if you make a mistake you can finish the
configure process, and then edit the created make include file by hand to fix
the mistake manually (the name and location of this file will be printed
out at the end of configure).
```

If you have problems during configure or installation, consult the file
'ATLAS/README/TroubleShoot.txt'.

Are you ready to continue? [y]:

Probing to make operating system determination:
Operating system configured as Linux

Probing for architecture:
Select pointer width and ABI for x86-64 system:
    1. 32 bit libraries
    2. 64 bit libraries
Enter bit number [2]:
Architecture is set to HAMMER64

Probing for supported ISA extensions:
    AltiVec: NO.
    AltiVec: NO.
    SSE3: NO.
    SSE2: NO.
    SSE1: NO.
    3DNow2: NO.
    3DNow1: NO.

ATLAS can provide SMP support for the Level 3 BLAS via Posix threads.
If you choose to build a threaded library, ATLAS will compile all
aspects of the library (including the serial components) with the
threaded compiler/link flags. Most machines can use the serial
library even when it is compiled with threaded options, but this
is not guaranteed to work, so if you want a true serial library,
answer no to threading below.
    enable Posix threads support? [y]:
Number of CPUs: 2

Required cache flush detected as : 2097152 bytes
Looking for compilers (this may take a while):
    /usr/bin/gcc   : v3.2.3

F77 = /usr/bin/g77 -fomit-frame-pointer -O -m64
CC = /usr/bin/gcc -fomit-frame-pointer -O -mfpmath=387 -m64
MCC = /usr/bin/gcc -fomit-frame-pointer -O -mfpmath=387 -m64

FINDING tar, gzip, AND gunzip
    tar   : /bin/tar
    gzip   : /bin/gzip
    gunzip : /bin/gunzip

ATLAS has default parameters for OS='Linux' and system='HAMMER64'.
If you want to just trust these default values, you can use express setup,
drastically reducing the amount of questions you are required to answer

    use express setup? [y]: n

--> 여기서 별도의 컴파일러를 사용하고 싶은 경우 n 을 선택하도록 한다.
그리고 해당 컴파일러와 flags 등을 입력하면 된다.

You need to choose a name which represents this architecture (eg. UltraSparc,
Dec21164, etc). Do not use a generic name (eg. solaris, linux), which might
apply to different hardware. This architecture name will be appended to the
name of the created make include file, and appear in all subdirectories, so
don't make it longer than you like to type. The name should follow the rules
for file names (so don't use punctuation and spaces, for instance).

    Enter Architecture name (ARCH) [Linux_HAMMER64_2]:

The ATLAS install process is heavily file-based, and this can cause major
reliability problems when interacting with an overloaded or malfunctioning
remotely mounted filesystem. ATLAS therefore has a mechanism in place to
allow for a delay before a file is declared to not be there, so that
slow NFS (i.e., waiting for amd timout) problems can be overcome, or for
handling slightly differing clocks between server/client. This problem is
magnified if doing cross-compilation. In the question below, we ask how
much of a delay, in seconds, ATLAS should tolerate between file creation
and appearance. If you are installing on a local filesystem (eg. /tmp) or
a smooth-running NFS system, answer 0; for a moderately loaded NFS server, you

may want a value in the 10 second range, and for cross-compiling systems or
NFS servers experiencing errors, you may want to go as high as a couple
of minutes (120).
    Enter File creation delay in seconds [0]:

    Enter Top level ATLAS directory [/usr/local/src/hpc/ATLAS]:
    Enter Directory to build libraries in [$(TOPdir)/lib/$(ARCH)]:
    Enter Library name for ATLAS primitives [libatlas.a]:
    Enter Library name for the C interface to BLAS [libcblas.a]:
    Enter Library name for the Fortran77 interface to BLAS [libf77blas.a]:
    Enter Library name for the C interface to BLAS [libptcblas.a]:
    Enter Library name for the Fortran77 interface to BLAS [libptf77blas.a]:

I'm going to ask you for information about your Fortran 77 compiler. ATLAS
does not need Fortran77 to build, so if you don't have a Fortran compiler,
the install can still be completed successfully. However, ATLAS built without
a Fortran compiler will not be callable from Fortran (i.e., the user should
use the C interface), and we will not be able to do full testing, since some of
the tester code is written in Fortran77.

    Enter F77 Linker [$(F77)]: **/usr/pgi/linux86-64/5.2/bin/pgf77**
    Enter F77 Link Flags [$(F77FLAGS)]: **-fast -tp=k8-64**

I am now going to ask for two C compilers, and their associated flags.
The first such set (CC & CCFLAGS) are used in compiling the non-generated
ATLAS code. This code is written in normal C, and responds well to high
levels of optimization. Typically, this is set to your default compiler,
and your highest levels of optimization.

The second set of C compilers (MCC & MMFLAGS) is used to compile the generated
ATLAS code. Generated codes are written at a very low-level (think of C used
as a kind of portable assembler). On many platforms, high levels of
optimization are detrimental, as the compiler tries to pipeline a perfectly
pipelined code, and succeeds in reducing performance substantially (this
occurs on DEC ALPHAs & Sun UltraSparcs, for instance). If the default does
not work for you, try a midrange optimization such as -O. The generated code
does not alias any output arguments, so aliasing optimizations should be OK.

Enter ANSI C compiler(CC) [/usr/bin/gcc]: **/usr/local/pgi/linux86-64/5.2/bin/pgcc**
Enter C Flags (CCFLAGS) [-fomit-frame-pointer -O -mfpmath=387 -m64]: **-fast -tp=k8-64**
Enter C Linker [$(CC)]: **/usr/local/pgi/linux86-64/5.2/bin/pgcc**
Enter C Link Flags [$(CCFLAGS)]: **-fast -tp=k8-64**

Enter Archiver [ar]:
Enter Archiver flags [r]:
Enter Ranlib [echo]:
Enter BLAS library []:
Enter General and system libs [-lpthread -lm]:

Finding F77 to C calling conventions (this may take a while):
Assertion system(ln) == 0 failed, line 884 of config.c
Unable to figure F2C data

Calculated F77/C interoperation conventions:
Unable to determine naming conventions
Unable to determine F77/C integer correspondence
Unable to determine F77/C string interoperation

Your architectural defaults do not include defaults for the
Level 1 BLAS. ATLAS now has the ability to tune the Level 1 BLAS to
your machine. However, this will add time to the install. If your
algorithm utilizes the Level 2 or Level 3 BLAS to any degree, the
the Level 1 BLAS will usually be a low order term, and thus only matter
for small problems. Therefore, if you don't think you need good performance
from the Level 1 BLAS, you can answer "no" to the question below, and ATLAS
will skip the Level 1 tuning. ATLAS will still provide Level 1 BLAS, but
their performance may be much worse than if tuning were allowed.

Tune the Level 1 BLAS? [y]:
Creating ATLrun.sh

Creating subdirectories:
Checking for already existing subdirectories ..... no
Subdirectories successfully created.

Storing L1 cache size of 64KB.

Moving config logfiles ConfSummary.log and ConfDump.log to bin/Linux_HAMMER64_2/INSTALL_LOG/

Configuration completed successfully. You may want to examine the make include
file (Make.Linux_HAMMER64_2) for accuracy before starting the install with the command:
    make install arch=Linux_HAMMER64_2

rm -f ./xconfig

==========

[root@otn02 ATLAS]# make install arch=Linux_HAMMER64_2

무사히 컴파일이 완료되면 정상적으로 현 시스템에 최적화된 blas 라이브러리를 사용할 수 있게 될 것이다.

~ATLAS/lib/Linux_HAMMER64_2/ 및에 libatlas.a, libf77blas.a, libptcblas.a,
libstatlas.a, libcblas.a, liblapack.a, libpf77blas.a 와 같은 라이브러리가 있음 정상적으로 컴파일이 되어진 것 이다.

- ATLAS 최적화 하기

Cache edge Size 값이 있는데 Pentium 4 의 경우는 기본적으로 209K로 되어져 있다. 이것을 적절히 변경하면 성능을 향상 시킬 수 있는데 P4 에서는 384K 가 이상적인 성능을 보였다.

vi ~ATLAS/include/<ARCH>/atlas_cacheedge.h

#ifndef ATLAS_CACHEEDGE_H
    #define ATLAS_CACHEEDGE_H

```
    #define CacheEdge 524288   -> 수정
#endif
```

```
# vi ~ATLAS/bin/l3blastst.c
```

---

line 79 줄쯤~

```
//#define USE_F77_BLAS -> 주석처리

#ifdef ATL_USEPTHREADS
#define USE_L3_PTHREADS
#endif
```

~/ATLAS/bin/<ARCH>/make xsl3blastst, xdl3blastst, xcl3blastst, xzl3blastst 실행

실제 ATLAS를 컴파일 하는 데는 오랜 시간과 인내가 필요하다. 거의 몇 시간은 자고 와야 할 것 이다. 플랫폼 별로 바이너리를 만들어 두는 것이 좋을 것이다.

3.4 병렬 프로그램 설치 및 설정 (MPICH, LAM-MPI)

- lammpi 설치/설정하기

http://lammpi.org/download

```
# tar xzvf lam-7.0.5.tar.gz
# cd lam-7.0.5
# ./configure --prefix=/usr/local/lam
```

// intel compiler 적용시 //

```
# ./configure --prefix=/usr/local/lam --with-fc=ifc --with-cc=icc --with-cxx=icc
```

// 단 /etc/ld.so.config 에 intel 관련 라이브러리 경로를 적어줘야 함. 아님 configure 시 에러남.

만일 PGI 컴파일러 설치 시는 아래 configure를 참조 한다.

./configure --prefix=/usr/local/lam --with-fc=/usr/local/pgi/linux86-64/5.1/bin/pgf77 ₩
--with-cc=/usr/local/pgi/linux86-64/5.1/bin/pgcc --with-CFLAGS=-DDEC_ALPHA

위는 AMD64bit Opteron 환경에서 적용한 것으로 32bit 환경에서는 마지막 --with-CFLAGS=-DDEC_ALPHA 옵션을 제거한다.

--with-cxx=pgCC (closs compile)를 추가할 필요가 있을 시는 추가하도록 한다.

lam-7.1.x 이상 부터는 configure 옵션의 변화가 있다.

--with-cc , --with-cxx 같은 옵션이 적용 되지 않는 문제가 있다. 그래서 실제 컴파일이 되어지는 Shell 의 환경 변수 부분에서 실제 CC, CXX, FC 등을 미리 지정 해 두고 컴파일 하여야 정상적으로 옵션을 적용 시킬 수 있다.

CC=/usr/local/pgi/linux86-64/5.2/bin/pgcc
CXX=/usr/local/pgi/linux86-64/5.2/bin/pgCC
FC=/usr/local/pgi/linux86-64/5.2/bin/pgf90
CFLAGS=-fast
FFLAGS=-fast
CXXFLAGS=-fast
export CC CXX FC CFLAGS FFLAGS CXXFLAGS
./configure --prefix=/usr/local/lam

혹은 ..

./configure --prefix=/usr/local/lam CC=/usr/local/pgi/linux86-64/5.2/bin/pgcc CXX=/usr/local/pgi/linux86-64/5.2/bin/pgCC FC=/usr/local/pgi/linux86-64/5.2/bin/pgf90 CFLAGS=-fast FFLAGS=-fast

이 해준다.

intel complier 연동 시 ..

CC=/usr/local/intel/cc/bin/icc
CXX=/usr/local/intel/cc/bin/icc
FC=/usr/local/intel/fc/bin/ifc

export CC CXX FC

./configure --prefix=/usr/local/lam-intel CFLAGS='-O3 -fast -unroll -axW -tpp7 -align' FFLAGS='-O3 -fast -unroll -axW -tpp7 -align'

lam-mpi 에서 intel compiler 와 연동하기 위해서는 Pentium 계열 프로세스를 사용하여야 한다.

make
make install

vi /etc/bashrc

LAMHOME=/usr/local/lam
PATH=$PATH:/usr/local/lam/bin
export LAMHOME PATH

---

lam-mpi 가 compiler 별로 여러 개 설치 될 경우에는 반드시 위 환경 설정을
해당 lam-mpi 에 맞추어 lam 실행 전에 적용을 시켜 주어야 한다.

만일 compiler 를 gcc, intel, pgi 를 모두 사용한다고 하면 lam-mpi 를
각 compiler 에 맞게 3번을 compiler 해 주어야 한다.

lam-mpi compiler 시...해당 연동 compiler 에 맞게..

./configure --prefix=/usr/local/lam-gcc
make && make install
./configure --prefix=/usr/local/lam-intel
make && make install
./configure --prefix=/usr/local/lam-pgi
make && make install
와 같이 각각 재 컴파일을 해주어야 한다.

vi /etc/lamhosts

```
real00
real01
real02
real03
real04
real05
.
.
```

Cpu 가 smp 인 경우에는 ..

```
# vi /etc/lamhost
```

```
real00 cpu=2
real01 cpu=2
real02 cpu=2
real03 cpu=2
real04 cpu=2
real05 cpu=2
.
.
```

---

lamd 을 실행한다. lamd 은 root 가 아닌 일반 계정에서 실행하여야 한다.
보안을 생각하면 lamd 의 퍼미션을 750 으로 주고 lamusers 란 그룹을 만들어서
hpcusers 란 그룹에 속한 유저만 실행 가능토록 한다.

```
$ lamboot -v -b /etc/lamhosts
```

// lamd 에 의해 계산에 참여한 노드 리스트

```
$ lamnodes
```

// lamd 중지

```
$ wipe -v -b /etc/lamhosts
```

```
$ mpicc -o ring ring.c
$ mpirun -np 8 ./ring
```

이런 식으로 테스트를 하면 된다.

만일 lamboot 수행 시 SSI boot modules 에러가 발생하면 configure 시 --with-boot= 에 사용된 프로토콜이 어떤 것인지 확인하고 해당 프로토콜 상의 통신이 이상이 없는지를 확인한다. MPI 통신에 이용되는 프로토콜은 rsh 와 ssh 가 있다. 2장에서 설정한 부분이 정상적으로 작동하는 지를 확인한다.

MPI 프로그램의 경우 MPI 프로그램이 실행되는 모든 시스템에 설치가 되어 있어야 한다. 이는 하나의 시스템에 설치를 하고 dutils 에서 제공하는 dua 란 명령어를 이용하여 일괄적으로 데이터 동기화를 시키면 될 것이다.

- mpich 설치 /설정하기

```
wget ftp://ftp.mcs.anl.gov/pub/mpi/mpich.tar.gz

# ./configure --prefix=/usr/local/mpich -cc=/usr/local/pgi/linux86/5.1/bin/pgcc ₩
--fc=/usr/local/pgi/linux86/5.1/bin/pgf90 -c++=/usr/local/pgi/linux86/5.1/bin/pgCC ₩
--with-device=ch_p4 --with-arch=LINUX -flinker=/usr/local/pgi/linux86/5.1/bin/pgf90

# make
# make install

# cd /usr/local/mpich/share
# vi machines.LINUX
-----------------------------------------------------------------
# host_name:cpu_num
otn1:2
otn2:2
otn3:2

# cd /usr/local/mpich/examples

# /usr/local/mpich/bin/mpicc -o cpi cpi.c
```

```
# /usr/local/mpich/bin/mpirun -np 2 cpi
----------------------------------------------------------------
Process 0 on otn1
Process 1 on otn2
pi is approximately 3.1416009869231241, Error is 0.0000083333333309
wall clock time = 0.000000
```

mpich 역시 Compiler 별로 여러 개를 사용하고자 할 때는 버전 별로 prefix를 다르게 주어서 설치 하도록 하라..

```
# ./configure --prefix=/usr/local/mpich-gcc
# make && make install
# ./configure --prefix=/usr/local/mpich-intel
# make && make install
# ./configure --prefix=/usr/local/mpich-pgi
# make && make install
```

이와 같이 설치를 하면 사용 시 마다 해당 MPI 컴파일러의 환경 설정을 해야 하는 번거로움이 있다. 하지만 Teragon HPC Management Tool 에서 제공하는 MPIsel 이란 프로그램을 이용하면 쉽게 관리 할 수 있다. 이는 나중에 자세히 설명하도록 하겠다.

3.5 분산 시스템 일괄 계정 관리

- HPC 계정 생성 방법

아래 스크립터는 dush를 이용하여 제작된 분산 시스템 일괄 계정 관리 스크립터들이다. 이 스크립터는 그냥 참고용으로 살펴 보길 바란다. Dutils2 에서 이 보다 향상된 일괄 계정 관리 기능을 제공함으로 실제 사용은 아래에 dutils2 에서 제공하는 duseradd, duserdel, dpasswd 등을 이용하면 보안측면에서 더 안전한 일괄 계정 관리가 가능하다.

```
# vi /root/bin/huseradd
--------------------------------------------------------------------------
#!/bin/sh

DUTIL=/usr/clx/sbin
DUTILNODE=/root/.nodelist
ADDUSER=/usr/sbin/adduser
```

```
CRTPASS=/root/bin/createpasswd

if [ $# -ne 2 ]
then
echo "Not enough input !! Input HPC User account and password   to add !!"
echo "Please Enter   HPC_user   Password"
   exit 1

else
            cat /etc/passwd |grep $1:x:
            cat /etc/passwd |grep $1:x: >file_chk1
            s=`ls -l file_chk1 |awk '{x+=$5};END{print x}'`

            if [ $s != "0" ]; then
             echo "NO, Can't create!! This account exists !"
             echo "Please Enter another ID"
            rm file_chk1
             exit 1
            else
            rm file_chk1

$DUTIL/dush -l $DUTILNODE $ADDUSER $1
$DUTIL/dush -l $DUTILNODE $CRTPASS $1 $2
fi
fi
--------------------------------------------------------------------------

# vi /root/bin/createpasswd
--------------------------------------------------------------------------
#!/bin/sh

if [ $# -ne 2 ]
then
echo "Among password creation, error occurrence !!"
echo "Please Enter   HPC_user   Password"

   exit 1
else
```

```
echo "$2"|passwd --stdin $1
fi
```

---

huseradd 는 HPC 와 같은 여러 개의 시스템으로 구성된 시스템의 경우 ypbind, openldap 같은 NIS 시스템을 가지지 않고도 일괄적인 계정 관리가 가능한 명령어이다.

ypbind 나 openldap 방식으로 시스템을 구현할 경우 사용자의 HOME_DIR 이 반드시 NFS 상에 있어줘야 하는 제한이 있다. 위 방식은 각 시스템의 환경을 분산 환경 그대로 사용하면서 효과는 NIS 서버에서 계정을 통합한듯한 효과를 주게 된다.
아래와 같은 환경만 만들어져 있음 편리하게 일괄 계정 관리가 가능합니다.

huseradd 는 반드시 /root/bin 에 위치하여야 한다.
huseradd 는 관리노드(master,node00)에서만 사용하여야 한다.
createpasswd 는 모든 노드의 /root/bin 밑에 위치하여야 한다.
Dutils 이 반드시 설치 되어져 있어야 한다.
/root/.nodelist 에 dutils 사용 nodelist 를 만들어야 한다.

사용 방법은 다음과 같다.

```
# huseradd <hpc_id> <password>
```

- HPC 계정 삭제 방법

```
# vi /root/bin/huserdel
```

```
#!/bin/sh

DUTIL=/usr/clx/sbin
DUTILNODE=/root/.nodelist
USERDEL=/usr/sbin/userdel

if [ $# -ne 1 ]
then
echo "Not enough input !! Input HPC User account to delete !!"
```

```
echo "Please Enter   HPC_user"
   exit 1
else

$DUTIL/dush -l $DUTILNODE $USERDEL -r $1

fi
```

---

huserdel 역시 huseradd 와 같은 환경을 요구한다.
사용방법은 아래와 같다.

```
# huserdel <hpc_id>
```

- 패스워드 변경 방법

패스워드 변경은 createpasswd 를 이용하여 할수 있다. 하지만 이는 dutils 와 같이 사용해야 한다.

```
# dush createpasswd <hpc_id> <password>
```

3.6 Open PBS 설치 및 관리하기

PBS(Portable Batch System)는 배치 작업 및 컴퓨터 시스템 자원을 관리하는 패키지 입니다.
배치 작업이란 작업의 순서를 정해 놓고 일괄적으로 처리하는 작업을 말합니다.
PBS에서는 작업을 실행하기 위해 Batch 작업 및 Shell Scripts와 제어 속성등을 전달하기
되며 그 작업이 종료될때 까지 작업은 유지 되고 보존되어집니다. PBS는 단일 시스템에 설치
할수 있고 여러 시스템을 그룹으로 묶어 설치 할수도 있습니다.

- Open PBS 의 설치

다음 홈페이지에서 해당 프로그램을 다운 받는다.

http://www.openpbs.org

다운 받은 소스를 특정 디렉토리에 풀어 놓는다.
OpenPBS 는 바이너리와 소스 형태로 제공되어 진다. 레드헷 계엘에서는 쉽게 바이너리로 설치 하면 된다. 일단 소스로 설치 하는 방법에 대해 알아 보겠다.

Openpbs 는 Master 서버와 일반 계산 서버에 설치 되는 패키지가 각각 다른다.
소스로 설치 할때 역시 사용되는 configure 옵션이 다르기 때문에 이에 주의해야 한다.

아래는 서버 측 설치 옵션이다.

```
[root@otn03 openpbs]# tar xzvf OpenPBS_2_3_16.tar.gz
[root@otn03 openpbs]# cd OpenPBS_2_3_16
[root@otn03 OpenPBS_2_3_16]# ./configure --prefix=/usr/local/openpbs --enable-docs ₩
--enable-server --enable-clients --enable-gui --set-tmpdir=/tmp ₩
--set-server-home=/var/spool/openpbs --set-server-name-file=server_name ₩
--set-default-server=HOSTNAME --enable-syslog --enable-shell-pipe
```

** --set-default-server에는 master 서버의 hostname 을 적여 주면 된다.
--enable-clients 는 xpbs 프로그램을 만들기 위해서는 tcl/tk 관련 프로그램이 필요하다.
없으면 에레가 발생한다.

아래는 일반 계산 서버의 설치 옵션이다.

```
[root@otn03 OpenPBS_2_3_16]# ./configure --prefix=/usr/local/openpbs --enable-docs ₩
--enable-server --enable-clients --enable-gui --set-tmpdir=/tmp ₩
--set-server-home=/var/spool/openpbs --set-server-name-file=server_name ₩
--set-default-server=HOSTNAME --enable-syslog --enable-shell-pipe
```

** master 서버와 일반 서버와의 차이는 --enable-server 나 --enable-mon 이나의 차이다.

```
[root@otn03 OpenPBS_2_3_16]# make && make install
```

이제 바이너리 RPM 으로 설치 하는 방법에 대해 알아보자. 소스에 비해 훨씬 간단한

설치 과정을 제공한다.

Master 서버 측

[root@otn03 openpbs]# rpm -Uvh --nodeps openpbs-exechost-2.3pl2-1.i386.rpm -> 클라이언트 패키지
[root@otn03 openpbs]# rpm -Uvh --nodeps openpbs-2.3pl2-1.i386.rpm -> 서버 패키지

** Master 서버 측에서는 서버관련 패키지와 클라이언트 관련 패키지를 모두 설치 한다.

작업 서버 측

[root@otn01 openpbs]# rpm -Uvh --nodeps openpbs-exechost-2.3pl2-1.i386.rpm

PBS 서버 프로그램과 클라이이언트 프로그램 모두 설치가 되면 /etc/profile.d/ 밑에 PBS 관련 환경 설정 파일을 만들어 준다.

[root@otn03 root]# vi /etc/profile.d/pbs.sh

PBS_HOME=/usr/spool/PBS
PBS_SERVER_HOME=/usr/spool/PBS
PATH=/usr/pbs/bin:/usr/pbs/sbin:$PATH

[root@otn03 root]# source /etc/profile.d/pbs.sh

- Open PBS 설정 방법

설정 하기 전에 먼저 PBS 서버 프로그램을 실행 하여야 한다.

서버 측 설정

[root@otn03 openpbs]# /usr/pbs/sbin/pbs_server -t create
Incorrectly built binary which accesses errno or h_errno directly. Needs to be fixed.

PBS_Server otn03: Create mode and server database exists,
do you wish to continue y/(n)?y

[root@otn03 openpbs]# cd /usr/spool/PBS/server_priv
[root@otn03 server_priv]# vi nodes

---

otn03 np=2
otn02 np=2
otn01 np=2

---

otn01 은 /etc/hosts 에 지정된 클라이언트 서버의 hostname 이고 np=2 는 SMP 머신의 경우 Process 수를 의미한다. 만일 hostname:ts 와 같이 :ts 붙여 주면 관련 노드를 time shared 형태로 사용하겠다는 의미이다.

[root@otn03 PBS]# cd /usr/spool/PBS
[root@otn03 PBS]# echo "server_hostname" > server_name

** 서버 측 설정은 서버 관련 설정 이외의 아래의 클라이언트 설정 역시 해주어야 한다.

클라이언트 측 설정

모든 클라이언트 서버에 아래 설정을 동일하게 해준다.

[root@otn01 root]# cd /usr/spool/PBS/
[root@otn01 PBS]# echo "server_hostname" > default_server
[root@otn01 PBS]# echo "server_hostname" > server_name
[root@otn01 PBS]# echo "$clienthost server_hostname" > mon_priv/config

이제 PBS 데몬을 master 서버와 작업 서버에서 각각 실행한다.

[root@otn03 root]# /etc/rc.d/init.d/pbs restart
[root@otn01 root]# /etc/rc.d/init.d/pbs restart

PBS Server 에서 PBS server 를 초기화 한다.

```
[root@otn03 PBS]# qmgr < pbs_server.conf
Max open servers: 4
```

정상적인 작동이 가능한지 확인한다.

```
[root@otn03 PBS]# pbsnodes -a
------------------------------------------------------------------------------
otn03
     state = free
     np = 2
     ntype = cluster

otn02
     state = free
     np = 2
     ntype = cluster

otn01
     state = free
     np = 2
     ntype = cluster
------------------------------------------------------------------------------
```

- PBS 서버 큐잉 설정 하기

PBS 서버 디렉토리에 보면 pbs_server.conf 란 파일이 있다. PBS 의 주요 설정 파일로 이곳에서 큐잉에 관련된 다양한 설정을 할 수 있다.

pbs_server.conf 의 기본 설정 내용은 다음과 같다.

```
[root@otn03 PBS]# vi /usr/spool/PBS/pbs_server.conf
------------------------------------------------------------------------------
#
# Create queues and set their attributes.
```

```
#
#
# Create and define queue workq
#
create queue workq
set queue workq queue_type = Execution
set queue workq enabled = True
set queue workq started = True
#
# Set server attributes.
#
set server scheduling = True
set server default_queue = workq
set server log_events = 511
set server mail_from = adm
set server query_other_jobs = True
set server scheduler_iteration = 600
```

설정 파일의 구성은 다음과 같다.

queue_type : execution 과 route 로 두가지가 있다. execution 은 실제 작업이 실행되는 큐이고 route 는 다른 큐로 작업을 넘길 때 사용되는 큐이다.

enabled : 큐를 사용할건지를 정의함
started : 큐를 시작할건지를 정의함
mail_from : 작업이 종료되거나 실행이 중단된 경우 메일로 메세지를 보내는데 이때 사용되어지는 메일 주소를 적는다.
query_other_jobs : 다른 사용자의 job 상태를 조회할 수 있는 지를 정의

이제 자기 시스템에 맞게 queue 설정을 해 보도록 하자

```
# vi queue.conf
```

```
# create and define queue verylong

create queue verylong
set queue verylong queue_type = Execution
```

```
set queue verylong Priority = 40
set queue verylong max_running = 10
set queue verylong resources_max.ncpus = 6
set queue verylong resources_min.ncpus = 1
set queue verylong resources_max.cput = 72:00:00
set queue verylong resources_min.cput = 12:00:01
set queue verylong resources_default.cput = 72:00:00
set queue verylong enabled = True
set queue verylong started = True


# create and define queue long

create queue long
set queue long queue_type = Execution
set queue long Priority = 60
set queue long max_running = 10
set queue long resources_max.ncpus = 6
set queue long resources_min.ncpus = 1
set queue long resources_max.cput = 12:00:00
set queue long resources_min.cput = 02:00:01
set queue long resources_default.cput = 22:00:00
set queue long enabled = True
set queue long started = True


# create and define queue medium

create queue medium
set queue medium queue_type = Execution
set queue medium Priority = 80
set queue medium max_running = 10
set queue medium resources_max.ncpus = 6
set queue medium resources_min.ncpus = 1
set queue medium resources_max.cput = 02:00:00
set queue medium resources_min.cput = 00:20:01
set queue medium resources_default.cput = 02:00:00
set queue medium enabled = True
```

```
set queue medium started = True

# create and define queue small

create queue small
set queue small queue_type = Execution
set queue small Priority = 100
set queue small max_running = 10
set queue small resources_max.ncpus = 6
set queue small resources_min.ncpus = 1
set queue small resources_max.cput = 00:20:00
set queue small resources_default.cput = 00:20:00
set queue small enabled = True
set queue small started = True

# create nad define queue default

create queue default
set queue default queue_type = Route
set queue default max_running =10
set queue default route_destinations = small
set queue default route_destinations += medium
set queue default route_destinations += long
set queue default route_destinations += verylong
set queue default enabled = True
set queue default started = True


# Set server attributes

set server scheduling = True
set server max_user_run = 6
set server acl_host_enable = True
set server acl_hosts = *
set server default_queue = default
set server log_events = 63
set server mail_from = alang@clunix.com
set server query_other_jobs = True
```

```
set server resources_default.cput = 01:00:00
set server resources_default.neednodes = 1
set server resources_default.nodect = 1
set server resources_default.nodes = 1
set server scheduler_iteration = 60
set server default_node = 1
```

**** 위 큐잉 설정 파일의 관리 정책 설명 ****

위 설정 파일은 verylong, long, medium, small 이름의 작업 시간으로 길이 별로 정의 된 실행 큐와 default 와 같이 특정 작업을 작업 시간에 따라서 다른 실행 큐로 전달한는 라우터 타입의 큐로 구성되어진다.

위 설정 파일의 Job 관리 정책은 다음과 같다.

사용자가 작업을 실행하면 처음 20분 동안은 small 이란 queue 에서 작업이 우선적으로 실행된다.

작업 시간이 20분이 넘어가면 이 작업은 medium 이란 queue 로 자동으로 넘어가게 된다. 이곳에서 small 보다는 낮은 우선 순위로 작업이 진행 되게 된다.

작업이 2시간이 넘어가면 long queue 로, 12시간이 넘어가면 verylong queue 로 넘어가서 처리 하게 되는 것이다.

이 파일을 적용 시켜 보도록 하자

```
[root@otn03 PBS]# qmgr < queue.conf
```

초기 큐 설정 상태로 돌아가고 싶다면 기본 pbs_server.conf 설정을 적용하면 된다.

```
[root@otn03 PBS]# qmgr < pbs_server.conf
```

큐의 기본 설정 상태를 확인 하기 위해서는 qmgr 명령을 이용해서 확인이 가능하다.
이 명령의 자세한 부분은 아래 "PBS 관련 명령" 에서 다루도록 하겠다.

```
[root@otn03 root]# qmgr
--------------------------------------------------------------------------------
Max open servers: 4
Qmgr: list queue default
Queue default
        queue_type = Route
        total_jobs = 0
        state_count = Transit:0 Queued:0 Held:0 Waiting:0 Running:0 Exiting:0
        max_running = 10
        resources_max.ncpus = 6
        resources_min.ncpus = 1
        route_destinations = small,medium,long,verylong
        enabled = True
        started = True

Qmgr: list queue small
Queue small
        queue_type = Execution
        Priority = 100
        total_jobs = 2
        state_count = Transit:0 Queued:1 Held:0 Waiting:0 Running:1 Exiting:0
        max_running = 10
        resources_max.cput = 00:20:00
        resources_max.ncpus = 6
        resources_min.ncpus = 1
        resources_default.cput = 00:20:00
        resources_assigned.ncpus = 6
        resources_assigned.nodect = 1
        enabled = True
        started = True
--------------------------------------------------------------------------------
```

현재의 설정 상태를 txt 로 저장 하기 위해서는 qmgr 의 -c 옵션으로 처리 할 수 있다.

```
[root@otn03 PBS]# qmgr -c 'print server' > queue.txt
```

- PBS 를 이용하여 작업 관리 하기

위 과정에서 정상적인 설정이 완료 되었다면 이제 바로 PBS 상에서 작업을 실행 할수 있을 것이다. PBS 상에서 Job 을 제어 관리하는 기본적인 방법에 대해 알아보자.

먼저 PBS 를 통해 job 처리는 qsub 란 명령어와 작업을 처리에 필요한 배치 스크립터를 만들어야 한다. 즉 job process script 를 qsub 란 명령어를 통해 실행 하는 것이다.

만일 HPL 벤치마킹 작업을 일반적인 방법으로 실행 하기 위해서는

```
$ lamboot -v -b ~/lamhost
$ mpirun -np 6 $HPL_PATH/xhpl
$ wipe -v -b ~/lamhost
```

위와 같은 일련의 과정이 필요할 것이다.

PBS 상에서 작업을 실행하기 위해서는 위의 과정을 하나의 배치 파일로 만든다.

```
$ vi hpl.sh
---------------------------------------------------------------------------
#!/bin/sh

cd $PBS_O_WORKDIR

lamboot -v -b $HOEE/lamhosts
mpirun -np 6 xhpl
wipe -v -b $HOME/lamhosts
---------------------------------------------------------------------------
```

* PBS_O_WORKDIR 은 실제 qsub 명령어를 실행하는 디렉토리의 절대 경로 이다.
* 즉 작업 디렉토리라고 보면됨

아래와 같이 qsub 명령어를 이용해서 hpl.sh 실행한다.

```
[clunix@otn03 Linux_ATHLON_BLAS]$ qsub hpl.sh
---------------------------------------------------------------------------
2.otn03
```

에러 없이 위와 같은 메세지가 출력하면 정상적으로 명령이 수행된것이다.

qstat 명령을 통해 작업 수행 상태를 확인해 보자

```
[clunix@otn03 Linux_ATHLON_BLAS]$ qstat
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
2.otn03                   hpl.sh           clunix          00:00:00 R small
```

위와 같이 2.otn03 작업은 clunix(User) 란 사용자가 small(Queue) 이란 큐에서 Running(S) 상태로 동작한다는 것이다.

만일 작업을 여러개 실행 하면..큐 설정에서 지정된 max_running 수 만큼 작업을 running 하고 이 이후의 작업은 Q 상태로 작업 대기 상태에 있게 된다.

즉 PBS 에서 시스템의 자원 현황에 맞게 PBS 큐잉 설정을 해 두면 사용자들이 시스템의 리소스를 Over 해서 사용하지 않게 자동으로 관리를 해주게 된다.

몇개의 job 이 동시 running 이 되게 하는 것은 max_running, resources_max.ncpus 등에 의해 제어 된다.

```
set queue long max_running = 10
set queue long resources_max.ncpus = 6
```

즉 위와 같이 long 큐에 max_running = 10 이고 resources_max.ncpus = 6 일 경우
Job 수가 10을 넘거나 qsub 를 통해 사용되는 process 의 수가 6개가 넘을 경우 그 이후의 job 은 모두 Queue 상태로 대기 하게 된다. 이 수치는 구축 시스템에 맞게 정의 하면 될것이다.

```
[clunix@otn03 Linux_ATHLON_BLAS]$ qstat
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
2.otn03                   hpl.sh           clunix          00:00:00 R small
3.otn03                   hpl.sh           clunix                 0 Q small

[clunix@otn03 Linux_ATHLON_BLAS]$ qstat -Q
Queue              Max Tot Ena Str Que Run Hld Wat Trn Ext Type
---------------- --- --- --- --- --- --- --- --- --- --- ----------
verylong             1     0 yes yes     0     0     0     0     0     0 Execution
```

```
long                1    0 yes yes    0    0    0    0    0    0 Execution
medium              1    0 yes yes    0    0    0    0    0    0 Execution
small               1    2 yes yes    1    1    0    0    0    0 Execution
default             1    0 yes yes    0    0    0    0    0    0 Route
```

작업이 완료되면 해당 Job id 번호의 출력 파일이 자동으로 생성 된다.

hpl.sh.e2
hpl.sh.e3
hpl.sh.o2
hpl.sh.o3

xxx.xx.ex 형식의 파일은 작업 중 발생한 에러 출력 파일이다.
xxx.xx.ox 형식의 파일은 작업 결과 출력 파일이다.

```
[clunix@otn03 Linux_ATHLON_BLAS]$ cat hpl.sh.e2
--------------------------------------------------------------------------------
n-1<22064> ssi:boot:base:linear: booting n0 (otn03)
n-1<22064> ssi:boot:base:linear: booting n1 (otn02)
n-1<22064> ssi:boot:base:linear: booting n2 (otn01)
n-1<22064> ssi:boot:base:linear: finished
n-1<22083> ssi:boot:base:linear: booting n0 (otn03)
n-1<22083> ssi:boot:base:linear: booting n1 (otn02)
n-1<22083> ssi:boot:base:linear: booting n2 (otn01)
n-1<22083> ssi:boot:base:linear: finished

[clunix@otn03 Linux_ATHLON_BLAS]$ cat hpl.sh.o2
--------------------------------------------------------------------------------
================================================================================
========
T/V                N      NB     P      Q               Time                 Gflops
--------------------------------------------------------------------------------
WR10R2C4       20000     256     1      6             359.48             1.484e+01
--------------------------------------------------------------------------------
||Ax-b||_oo / ( eps * ||A||_1  * N        ) =        0.0147722 ..... PASSED
||Ax-b||_oo / ( eps * ||A||_1  * ||x||_1  ) =        0.0142974 ..... PASSED
||Ax-b||_oo / ( eps * ||A||_oo * ||x||_oo ) =        0.0027248 ..... PASSED
```

```
================================================================================
========

Finished        1 tests with the following results:
                 1 tests completed and passed residual checks,
                 0 tests completed and failed residual checks,
                 0 tests skipped because of illegal input values.
--------------------------------------------------------------------------------

End of Tests.
================================================================================
========

LAM 7.1.1/MPI 2 C++/ROMIO - Indiana University
```

그리고 /usr/spool/PBS/server_priv/accounting 디렉토리 밑에 보면 그날의 날짜 이름으로 사용자 별 작업 시간 및 리소스 사용 현황을 로그로 남기게 된다. 이 account log 를 이용해서 계산서 발급등을 할 수 있게 된다.

```
[root@otn03 accounting]# vi 20050316
--------------------------------------------------------------------------------
.
.
03/16/2005 10:38:49;E;3.otn03;user=clunix group=clunix jobname=hpl.sh queue=small
ctime=1110936611 qtime=1110936611 etime=1110936611 start=1110936762
exec_host=otn03/0 Resource_List.cput=00:20:00 Resource_List.ncpus=6 Resource_List.neednodes=1
Resource_List.nodect=1 Resource_List.nodes=1 session=22094 end=1110937129 Exit_status=0
resources_used.cput=00:00:00 resources_used.mem=3280kb resources_used.vmem=85784kb
resources_used.walltime=00:06:07
--------------------------------------------------------------------------------
```

- PBS 관련 명령

여기서는 PBS 에서 주로 사용되는 명령어에 대해 다루어 보도록 하겠다. 기본적인 작동에 필요한 명령어는 이미 위 단락에서 사용해 보았을 것이다.

** PBS 주요 명령어

qsub : 작업을 사용자가 큐에 제출함
qalter : batch 작업의 속성을 변경
qdel : 큐에 있는 작업을 삭제함
qhold : 작업에 스케줄 되어 실행 되지 않도록 함
qmove : 작업을 다른 큐로 이동함
qmsg : 실행 중인 작업의 기본 출력 다음에 임의의 메세지를 추가함
qrerun : 실행 중인 작업을 장제 종료하고, 다시 큐로 돌려보냄

**** 주요 사용 방법**

$ qsub <pbs_job_scripts>

$ qdel <job_id>
* job_id 는 qsub 실행시 출력되는 8.otn03 같은 형태의 메세지의 숫자에 해당하는 부분이다.

```
[clunix@otn03 Linux_ATHLON_BLAS]$ qstat
Job id              Name             User             Time Use S Queue
---------------- ---------------- ---------------- -------- - -----
8.otn03             hpl.sh           clunix           00:00:00 R small
9.otn03             hpl.sh           clunix                  0 Q small
10.otn03            hpl.sh           clunix                  0 Q small
```

여기서 현재 큐에 대기중인 10.otn03 작업을 취소 해 보도록 하겠다.

```
[clunix@otn03 Linux_ATHLON_BLAS]$ qdel 10.otn03
[clunix@otn03 Linux_ATHLON_BLAS]$ qstat
Job id              Name             User             Time Use S Queue
---------------- ---------------- ---------------- -------- - -----
8.otn03             hpl.sh           clunix           00:00:00 R small
9.otn03             hpl.sh           clunix                  0 Q small
```

* qhold <job_id>

qhold 는 특정 작업이 Queuing 상태에서 작업 대기 중에 있을때 대기 상태에서 앞의 작업이 처리되어 실행 상태로 전환 되면 자동으로 Running 상태로 들어가는대 이것을 Running 으로 들어가지 않고 계속 Holding 상태에 있게 하는 명령어다.

```
[clunix@otn03 Linux_ATHLON_BLAS]$ qstat
Job id              Name             User             Time Use S Queue
---------------- ---------------- ---------------- -------- - -----
9.otn03             hpl.sh           clunix           00:00:00 R small
11.otn03            hpl.sh           clunix                 0 Q small

[clunix@otn03 Linux_ATHLON_BLAS]$ qhold 11.otn03

[clunix@otn03 Linux_ATHLON_BLAS]$ qstat
Job id              Name             User             Time Use S Queue
---------------- ---------------- ---------------- -------- - -----
9.otn03             hpl.sh           clunix           00:00:00 R small
11.otn03            hpl.sh           clunix                 0 H small
```

이렇게 11 번 Job 이 Queuing 상태에서 holding 상태로 넘어 가게 되면 9번 job 이 처리가 끝나게 되어도 11 번 Job 은 진행 되지 않는다.

* qrls <job_id>

qrls 명령은 qhold 명령으로 인해 hold 된 job 을 다시 풀어 놓는 명령이다.

```
[clunix@otn03 Linux_ATHLON_BLAS]$ qrls 11

[clunix@otn03 Linux_ATHLON_BLAS]$ qstat
Job id              Name             User             Time Use S Queue
---------------- ---------------- ---------------- -------- - -----
9.otn03             hpl.sh           clunix           00:00:00 R small
11.otn03            hpl.sh           clunix                 0 Q small
```

* qmsg [-E] [-O] <message_string> <job_id>

qmsg 는 Job 의 처리가 완료될 경우 생성되는 error 출력파일과 output 출력파일에 사용자의 임의의 메세지를 남기게 된다.

```
[clunix@otn03 Linux_ATHLON_BLAS]$ qmsg -E -O 'Alang Job End' 9
```

이와 같이 명령을 수행하면 9 번 job 이 완료된 후에 생성되는 출력파일에 Alang Job End 라는 메세지가 추가 되게 된다.

-E 옵션은 error 출력 파일에 사용자 메세지 추가
-O 옵션은 output 출력 파일에 사용자 메세지 추가
만일 아무런 옵션이 없을 경우는 표준 error 파일에 메세지가 추가 된다.

* qstat

현재 진행 중인 큐의 상태를 확인하는 명령어이다.

주요 옵션은

-Q : 큐의 상태 확인
-B : 서버의 상태 확인
-f : Job_queue 의 Detail 한 정보 확인

```
[clunix@otn03 Linux_ATHLON_BLAS]$ qstat
Job id              Name             User             Time Use S Queue
---------------- ---------------- ---------------- -------- - -----
11.otn03            hpl.sh           clunix          00:00:00 R small
13.otn03            hpl.sh           clunix                 0 Q small
14.otn03            hpl.sh           clunix                 0 Q small

[clunix@otn03 Linux_ATHLON_BLAS]$ qstat -Q
Queue              Max Tot Ena Str Que Run Hld Wat Trn Ext Type
---------------- --- --- --- --- --- --- --- --- --- --- ----------
verylong             1   0 yes yes   0   0   0   0   0   0 Execution
long                 1   0 yes yes   0   0   0   0   0   0 Execution
medium               1   0 yes yes   0   0   0   0   0   0 Execution
small                1   3 yes yes   2   1   0   0   0   0 Execution
default              1   0 yes yes   0   0   0   0   0   0 Route

[clunix@otn03 Linux_ATHLON_BLAS]$ qstat -B
Server             Max Tot Que Run Hld Wat Trn Ext Status
```

```
---------------- --- --- --- --- --- --- --- --- ----------
otn03                 0     3     2     1     0     0     0     0 Active


[clunix@otn03 Linux_ATHLON_BLAS]$ qstat -f 14
--------------------------------------------------------------------------------
Job Id: 14.otn03
    Job_Name = hpl.sh
    Job_Owner = clunix@otn03
    job_state = Q
    queue = small
    server = otn03
    Checkpoint = u
    ctime = Wed Mar 16 11:38:58 2005
    Error_Path = otn03:/home/clunix/hpl/bin/Linux_ATHLON_BLAS/hpl.sh.e14
    Hold_Types = n
    Join_Path = n
    Keep_Files = n
    Mail_Points = a
    mtime = Wed Mar 16 11:38:58 2005
    Output_Path = otn03:/home/clunix/hpl/bin/Linux_ATHLON_BLAS/hpl.sh.o14
    Priority = 0
    qtime = Wed Mar 16 11:38:58 2005
    Rerunable = True
    Resource_List.cput = 00:20:00
    Resource_List.ncpus = 6
    Resource_List.nodect = 1
    Resource_List.nodes = 1
    Variable_List = PBS_O_HOME=/home/clunix,PBS_O_LANG=en_US.UTF-8,
        PBS_O_LOGNAME=clunix,
        PBS_O_PATH=/usr/local/pgi/linux86-64/5.2/bin:/usr/pbs/bin:/usr/pbs/sbin:/usr
        /kerberos/bin:/opt/intel_fc_80/bin:/opt/intel_cc_80/bin:/usr/clx/sbin:/
        bin:/usr/bin:/usr/local/bin:/usr/bin/X11:/usr/X11R6/bin:/usr/local/lam-
        gcc/bin:/home/clunix/bin,PBS_O_MAIL=/var/spool/mail/clunix,
        PBS_O_SHELL=/bin/bash,PBS_O_HOST=otn03,
        PBS_O_WORKDIR=/home/clunix/hpl/bin/Linux_ATHLON_BLAS,
        PBS_O_QUEUE=default
    comment = Not Running: Queue job limit has been reached.
```

etime = Wed Mar 16 11:38:58 2005

- PBS 큐잉 관리 하기

PBS 의 큐의 관리를 위해서는 qmgr 명령어를 이용하여야 한다.
기본적인 queue 설정은 앞에서 설명한 바와 같이 pbs_server.conf 혹은 queue.conf 파일에 관리 정책에 맞게 설정을 하고 난 뒤 ..

qmgr < queue.conf

와 같이 실행하면 일괄 설정이 되어진다.

하지만 설정의 상태 확인 및 일부 수정등을 위해서 qmgr shell 상에서 큐잉 시스템을 관리하는 방법에 대해 알아보자

먼저 qmgr 을 실행한다.

[clunix@otn03 Linux_ATHLON_BLAS]$ qmgr
Max open servers: 4
Qmgr:

qmgr 명령 형식은 다음과 같다.

[command] [object] [obj_name] [attribute]

[command] 에는 다음과 같은 명령이 있다.

active(a) : 객체를 활성화 한다.
create(c) : 객체를 생성한다.
delete(d) : 객체를 삭제한다.
set(s) : 객체에 속성을 지정한다.
unset(u) : 객체의 속성을 제거 한다.
print : 객체의 설정 값을 출력한다.
list : 객체의 속성 값을 출력한다.

[object] 에는 다음과 같은 객체가 있다.

server
queue
node

[name] 은 실제 객체에 대한 이름이다. node 의 경우는 node_name 이 될것이고 queue 의 경우에는 앞서 큐잉 설정 파일에 create queue 로 생성한 queue 의 이름이 될것이다.

[attribute] 는 실제 해당 객체의 세부 속성이다.

* 현재 서버의 속성을 확인해 보자

```
Qmgr: list server
------------------------------------------------------------------------
Server otn03
        server_state = Active
        scheduling = True
        max_user_run = 6
        total_jobs = 1
        state_count = Transit:0 Queued:0 Held:0 Waiting:0 Running:1 Exiting:0
        acl_host_enable = True
        acl_hosts = *
        default_queue = default
        log_events = 63
        mail_from = alang@clunix.com
        query_other_jobs = True
        resources_default.cput = 01:00:00
        resources_default.nodect = 1
        resources_default.nodes = 1
        resources_assigned.ncpus = 6
        resources_assigned.nodect = 1
        scheduler_iteration = 60
        default_node = 1
        pbs_version = OpenPBS_2.3
```

* 현재 서버의 설정 값을 확인해 보자

```
Qmgr: print   server
------------------------------------------------------------------------
.
생략..
.
.
set queue default enabled = True
set queue default started = True
#
# Set server attributes.
#
set server scheduling = True
set server max_user_run = 6
set server acl_host_enable = True
set server acl_hosts = *
set server default_queue = default
set server log_events = 63
set server mail_from = alang@clunix.com
set server query_other_jobs = True
set server resources_default.cput = 01:00:00
set server resources_default.nodect = 1
set server resources_default.nodes = 1
set server scheduler_iteration = 60
set server default_node = 1
```

* queue 의 속성을 확인해 보자

```
Qmgr: list queue small
------------------------------------------------------------------------
Queue small
        queue_type = Execution
        Priority = 100
        total_jobs = 1
        state_count = Transit:0 Queued:0 Held:0 Waiting:0 Running:1 Exiting:0
        max_running = 1
        resources_max.cput = 00:20:00
```

```
        resources_max.ncpus = 6
        resources_min.ncpus = 1
        resources_default.cput = 00:20:00
        resources_assigned.ncpus = 6
        resources_assigned.nodect = 1
        enabled = True
        started = True
```

* small queue 의 resources_min.ncpus 값을 1 에서 2로 변경해 보자

```
Qmgr: set queue small resources_min.ncpus=2
Qmgr: list queue small
------------------------------------------------------------------------
Queue small
        queue_type = Execution
        Priority = 100
        total_jobs = 1
        state_count = Transit:0 Queued:0 Held:0 Waiting:0 Running:1 Exiting:0
        max_running = 1
        resources_max.cput = 00:20:00
        resources_max.ncpus = 6
        resources_min.ncpus = 2
        resources_default.cput = 00:20:00
        resources_assigned.ncpus = 6
        resources_assigned.nodect = 1
        enabled = True
        started = True
------------------------------------------------------------------------
```

** list, print 를 제외한 다른 명령은 반드시 PBS 관리자 권한에서 사용하여야 한다. 초기 PBS 관리자는 root 이다. 시스템 관리자와 PBS 관리자가 서로 다른 경우 PBS 관리자를 root 에서 다른 계정으로 변경해 주어야 한다.

*** PBS 관리자 변경 방법**

```
Qmgr: set server managers=clunix@otn03
Qmgr: print server managers
#
```

```
# Set server attributes.
#
set server managers = clunix@otn03
```

- PBS 스크리트 제작

qsub 에 사용되는 스크립트에 제작 시 유용한 환경 변수 및 기본 제작에 유용한 방법에 대해 간단히 설명하겠다.

qsub 에 사용되는 스크립트 제작시 다음과 같은 환경 변수를 이용할 수 있다.
시스템에서 지원하는 환경변수에 PBS_O_ 라는 접두사를 붙여 읽어 들일 수가 있다.

즉 ..

* PBS_O_HOST : qsub 명령이 실행된 호스트 이름
* PBS_O_QUEUE : 작업이 처음 제출되었던 큐의 이름
* PBS_O_WORKDIR : qsub 명령이 실행된 디렉토리의 절대 경로
* PBS_JOBID : 작업 아이디
* PBS_JOBNAME : 작업 이름 (qsub 로 실행하는 scripts 이름)
* PBS_NODEFILE : 작업에 활당된 노드 목록이 저장된 파일의 이름
* PBS_QUEUE : 작업이 실행되고 있는 큐의 이름

위의 변수를 이용하여 보다 효율적인 실행 스크립터를 만들 수가 있다.

만일 아래와 같은 스크립터를 qsub 로 실행하였다고 하자.

```
[clunix@otn03 Linux_ATHLON_BLAS]$ vi mpi.sh
---------------------------------------------------------------------------
#!/bin/sh

echo $PBS_O_HOST
echo $PBS_O_HOME
echo $PBS_O_QUEUE
echo $PBS_O_WORKDIR
echo $PBS_JOBID
echo $PBS_JOBNAME
```

```
echo $PBS_NODEFILE
echo $PBS_QUEUE
```

```
[clunix@otn03 Linux_ATHLON_BLAS]$ qsub mpi.sh
32.otn03

[clunix@otn03 Linux_ATHLON_BLAS]$ vi mpi.sh.o32
```

```
otn03
/home/clunix
default
/home/clunix/hpl/bin/Linux_ATHLON_BLAS
32.otn03
mpi.sh
/usr/spool/PBS/aux/32.otn03
small
```

이밖에 스크립트 안에 PBS 의 추가 기능을 포함 할수가 있는데 대표적인 사용 방법은 아래와 같다.

```
[clunix@otn03 Linux_ATHLON_BLAS]$ vi hpl.sh
```

```
#!/bin/sh
#PBS -j oe
#PBS -l nodes=3
#PBS -m ae

cd $PBS_O_WORKDIR

lamboot -v -b $PBS_NODEFILE
mpirun -np 6 xhpl
wipe -v -b $PBS_NODEFILE
```

위의 스크립터에서 사용된 #PBS 정의 부분이 있다.

* #PBS -j oe : 표준 error 출력 파일과 표준 output 출력 파일을 하나로 합니다.
합쳐진 출력 파일은 output 파일로 통일 된다

* #PBS -l nodes=3 : PBS 에 사용되어지는 node 수를 정의 하는 곳이다. 만일 이 정의를
생략하게 되면 time shared 상태로 동작하게 된다.

* #PBS -m ae : PBS 의 작업이 종료되면 자동으로 PBS server managers 에 지정된 사용자
에게 작업에 소요된 시스템 정보를 메일로 보내 주게 딘다. a 의 경우는
경고가 발생 했을 경우에 메일을 보내는 것이고 e 의 경우는 정상 종료된
경우에도 메일을 보내도록 하는 것이다.

***** time-shared 란 ..**

time-shared 란 용어는 프로세스 스케줄링에 주로 등장하는 용어로 여러개의 job 을 동시에 처리할때 CPU 를 시간 단위로 분활해서 사용하는 방식을 말한다.

즉 1, 2 의 두개의 Job 을 처리 할때 time-shared 방법으로 처리를 하면 두개의 job 을 동시에 수행하는데 프로세스 사용 시간을 분활하여 1번 job 을 처리하다가 분할 시간이 되면.. 2번 job 을 수행하고 또 시간이 되면 다시 1번 job 을 수행하는 방식을 의미 한다.

PBS queue 설정에 의해 예를 들어 보면 물리적인 CPU 수가 6이고 PBS 설정 상의 resources_max.ncpus = 12 일 경우 np 6 으로 mpirun 을 실행 하는 job 이 두개 일 경우 모두 running 상태로 작업이 실행 될 것이다. 이때 time-shared 방식에서는 Job 을 구분하여 해당 Job 에 활당된 CPU 요청 (job 당 6개, 총 12개)을 시간 별로 조금씩 활당 하여 물리적으로 처리 가능한 6개의 CPU 로 12개의 CPU 작업을 동시에 진행하게 된다.

non-time-shared 방식에서는 두개의 job 이 동시에 running 이 되더라도 우선 첫번째 Job 에서 요청한 6개의 CPU 작업을 실제 6개의 CPU 가 모두 처리 하고 난 뒤에 두번째 Job 에 6개 CPU가 모두 작업에 활당되어지게 된다.

time-shared 방식은 실제 작업의 특성에 따라 효율적일수도 있고 그렇지 못할 수도 있다. I/O 가 많은 작업의 경우 실제 Job 처리 중에 I/O 를 일으킬때 CPU 는 Idle 상태로 있기 때문에 이 시간 동안 다른 작업을 진행 할 수 있게 된다. 하지만..오직 CPU 자원만 활용하는 job 의 경우는 실제 batch 방식의 작업이 더 효율적일 수 있다.

3.7 Teragon HPC Management Tool 설치, 설정, 관리 (Dutils 2.0)

Dutils 2.0은 dutils 1.0 의 보안적 취약점과 기능을 개선한 제품으로 다양한 분산 시스템을 관리하는 프로그램을 제공한다. 각 기능과 사용 방법을 아래에 자세히 설명한다.

- Dutils 2.0 설치 및 설정 하기

먼저 RPM Package 를 설치 한다.

rpm -Uvh dutils-2.0.0.i386.rpm

그럼 /usr/local/dutils 이란 경로에 dutils 2.0 이 생성 될것이다.

dutils2.0 의 핵심 모듈은 dush2 란 프로그램이다. 일단 dush2를 초기화 하여 관리자 인증키를 생성 시켜 놓아야 dutils2.0 이 제공하는 모든 기능을 무난히 사용할 수 있을 것이다.

기본 설정은 /usr/local/dutils/etc/dutils.cfg 파일을 열어 DUTIL_ADM 에 관리자 리포트 메일 주소를 적어 주고 nodelist 파일에 클러스터 관리 노드의 hostname 을 적어 준다.

dush2에 대한 자세한 기능 설명은 아래 dutils 2.0 사용하기에서 설명하겠다.

- Dutils 2.0 사용 하기

* timesync, timeview

: **클러스터 시스템의 전 노드를 표준 시간에 맞게 동기화를 시키는 명령**

timesync 는 관리 노드에서 표준 시간을 맞추고, 계산 노드는 관리 노드와 시간을 동기화 시켜 주는 명령이다. 사용방법은 관리자 권한에서 timesync 명령을 수행하면 된다.

timeview 는 실제 클러스터 시스템의 현 시간을 확인하는 명령이다.
timeview 역시 그냥 timeview 란 명령을 입력하면 된다.

```
[root@otn03 bin]# timesync
* HPC system time synced ..
```

```
[root@otn03 bin]# timeview
---------------------------------------------------------------
[otn01] Tue Mar 22 13:42:57 KST 2005
[otn02] Tue Mar 22 13:42:57 KST 2005
[otn03] Tue Mar 22 13:42:57 KST 2005
```

* duseradd, duserdel, dpasswd

: **분산 시스템 계정 관리 방식에서 일괄적을 계정 생성, 삭제, 패스워드 변경등을 해주는 명령어**

duseradd 는 분산 시스템의 계정을 일괄적으로 생성하는 명령어입니다.
duserdel 는 분산 시스템의 계정을 일괄적으로 삭제하는 명령어입니다.
dpasswd 는 분산 시스템의 계정의 패스워드를 일괄적으로 변경하는 명령어입니다.

사용방법은 다음과 같다.

```
# duseradd [account] [password]
[root@otn03 root]# duseradd test pass
-------------------------------------------------------------------
[otn01] Changing password for user test.
[otn01] passwd: all authentication tokens updated successfully.
[otn02] Changing password for user test.
```

[otn02] passwd: all authentication tokens updated successfully.
[otn03] Changing password for user test.
[otn03] passwd: all authentication tokens updated successfully.

dpasswd [account] [password]
[root@otn03 bin]# dpasswd test root///

[otn01] Changing password for user test.
[otn01] passwd: all authentication tokens updated successfully.
[otn02] Changing password for user test.
[otn02] passwd: all authentication tokens updated successfully.
[otn03] Changing password for user test.
[otn03] passwd: all authentication tokens updated successfully.

duserdel [account]
[root@otn03 bin]# duserdel test

Do you delete home directory for this account? (yes/no)
* 사용자 홈디렉토리를 삭제할려면 yes, 보존할려면 no 를 입력

executing in otn01
executing in otn02
executing in otn03

* dalive

: **클러스터 시스템의 서버 활성 상태 및 네트워크 상태 모니터링**

dalive 는 클러스터 시스템 각각 서버의 활성 상태 및 네트워크 활성 상태를 한번에 모니터링 하는 것이다. 클러스터 구성 노드 수가 많은 경우 특정 노드에 이상이 있을 경우 별도의 모니터링 툴이 없거나 있어도 웹 브라우저나 특정 프로그램 상에서 파악이 가능하면 콘솔이나 터미널 작업 시 즉각적으로 확인이 힘든 경우가 발생한다. 이때 간단히 command 기반에서 이를 확인할수 있다.

사용방법은 다음과 같다

dalive [option] [hostname]

* option list

-n : 특정 노드에 대한 서버, 네트워크 상태 체크
-c : 활성 노드로 새로운 nodelist 생성
-d : 다운된 노드 리스트 출력
-h : 도움말
-r : 결과 리포트 작성
-m : 결과 리포트 메일 발송
null : nodelist 에 등록된 host 에 대한 서버, 네트워크 상태 체크

```
[root@otn03 bin]# ./dalive
-----------------------------------------------------
hpc system alive state ..

otn01 is alive
otn02 is alive
otn03 is alive
-----------------------------------------------------

[root@otn03 bin]# ./dalive -n otn01
-----------------------------------------------------
otn01 is alive
-----------------------------------------------------

[root@otn03 bin]# ./dalive -c
-----------------------------------------------------
hpc system alive state ..

otn01 is alive
otn02 is alive
otn03 is alive

update new nodelist..

otn01
otn02
otn03
-----------------------------------------------------
```

```
[root@otn03 bin]# ./dalive -d
-----------------------------------------------------
deathed node list...

otn01

[root@otn03 bin]# dalive -r
-----------------------------------------------------
creat report file 20050318-alive.txt to output..

[root@otn03 bin]# dalive -m
-----------------------------------------------------
send to mail of output result
```

* dload (uptime)

: **클러스터 시스템의 전체 시스템 부하 및 활성 상태를 일괄적으로 모니터링 하는 명령**

dload 사용 방법은 아래와 같다.

```
# dload [ option ]

        * option
                -r : 출력 결과 리포트 작성 하기
                -m : 출력 결과 리포트 메일 발송하기
                -h : 도움말

[root@otn03 bin]# dload
-----------------------------------------------------------------------
* hpc system load average info

=========================================================
 hostname   ||     date     || uptime ||   load average
=========================================================
   [otn01]         15:15:40     20days,     0.00, 0.00, 0.00
```

```
   [otn02]        15:09:27    23days,    0.00, 0.00, 0.00
   [otn03]        15:15:01    28days,    0.00, 0.00, 0.00,
```

* dcpu (ps,head)

: **클러스터 시스템의 전체 CPU 사용량을 모니터링 - 대량 CPU 사용 프로세스**

dcpu 는 클러스터 시스템의 프로세스별 CPU, MEM 사용량을 모니터링 하는 명령이다.
dcpu 는 시스템별로 CPU 사용량이 높은 10개의 프로세스에 대한 CPU, MEM 점유률을
보여 주며 특정 프로세스에 대한 모니터링도 가능하다.

dcpu 의 사용방법은 다음과 같다.

dcpu [process_name]

* 특별히 모니터링 할 process 가 없는 경우엔 그냥 dcpu 라고 입력하면 됩니다.
  그럼 각 서버별 CPU 상위 점유 프로세스를 출력한다.

* 특별히 모니터링 할 process 가 있는 경우엔 dcpu 뒤에 process 명을 적어준다.

```
[root@otn03 bin]# dcpu
================================================================================
 Hostname    ||  CPU    ||    MEM    ||    PROC
================================================================================
   [otn01]        84.2         14.9          xhpl
   [otn01]        77.7         15.8          xhpl
   [otn01]        0.1          0.1          bin/esmd
   [otn01]        0.1          0.0          bin/edbd -n eth0 -p 908 -d
   [otn01]        0.0          0.5          emacs
   [otn01]        0.0          0.1          /usr/clx/bin/egid -f /usr/clx/rc/http/conf/egid.conf
   [otn01]        0.0          0.1          /usr/clx/bin/egid -f /usr/clx/rc/http/conf/egid.conf
   [otn01]        0.0          0.1          /usr/clx/bin/egid -f /usr/clx/rc/http/conf/egid.conf
   [otn01]        0.0          0.1          /usr/clx/bin/egid -f /usr/clx/rc/http/conf/egid.conf
   [otn02]        82.7         16.4          xhpl
   [otn02]        79.0         16.4          xhpl
```

```
  [otn02]        1.0          0.0          bin/el4d
  [otn02]        0.1          0.1          bin/edbd -n eth0 -p 908 -d
  [otn02]        0.0          0.1          ssh otn03
  [otn02]        0.0          0.0          xinetd -stayalive -pidfile /var/run/xinetd.pid
  [otn02]        0.0          0.0          /usr/sbin/sshd
  [otn02]        0.0          0.0          /usr/pbs/sbin/pbs_mom
  [otn02]        0.0          0.0          /usr/local/lam-gcc/bin/lamd -H 192.168.123.167 -P 41831
  [otn03]        94.7          15.0           xhpl
  [otn03]        81.2          17.9           xhpl
  [otn03]        0.0          0.1          vim dload
  [otn03]        0.0          0.1          ssh -n otn03 cd /usr/local/dutils/bin; ps
  [otn03]        0.0          0.0          xinetd -stayalive -pidfile /var/run/xinetd.pid
  [otn03]        0.0          0.0          xfs -droppriv -daemon
  [otn03]        0.0          0.0          /usr/sbin/sshd
  [otn03]        0.0          0.0          /usr/sbin/named -u named
  [otn03]        0.0          0.0          /usr/sbin/atd

[root@otn03 bin]# dcpu xhpl
================================================================================
==
 Hostname    ||   CPU    ||    MEM    ||    PROC
================================================================================
==
  [otn01]        94.5          17.1          xhpl
  [otn01]        91.8          16.2          xhpl
  [otn02]        94.9          15.0          xhpl
  [otn02]        86.0          16.6          xhpl
  [otn03]        97.7          17.5          xhpl
  [otn03]        93.8          16.4          xhpl
```

* dmem (free)

: **클러스터 시스템의 전체 MEM 사용량을 모니터링**

dmem 은 클러스터 시스템의 전체 Memory 상태를 일괄적으로 모니터링 하는 명령이다.
dmem 의 사용방법은 다음과 같다.

```
                    Usage:
 dload [option]
      -r : create to output report
      -m : mail to output report
      -h : help message

[root@otn03 bin]# dmem
------------------------------------------------------------------------
* hpc system memory using info

=========================================================================
=========
   hostname   ||   total   ||   used   ||    free   ||   shared   || buffers || cached
=========================================================================
=========
[otn01] Mem:     1977          755          1221          0          199          103
[otn01] Swap:    2000          42          1957
[otn02] Mem:     1977          544          1433          0          126          97
[otn02] Swap:    2000          7          1992
[otn03] Mem:     1977          1959          17          0          20          1674
[otn03] Swap:    2000          9          1990


[root@otn03 bin]# dmem -r
------------------------------------------------------------------------
creat report file 20050321-dmem.txt to output..

[root@otn03 bin]# dmem -m
------------------------------------------------------------------------
send to mail of output result
```

* ddisk (df)

: **클러스터 시스템의 전체 디스크 사용량을 모니터링**

ddisk 는 클러스터 시스템의 파티션별 디스크 사용량과 파일시스템 속성을 일괄적으로 모니터링 하는 명령이다.

사용방법은 다음과 같다.

```
Usage:
dload [option]
    -r : create to output report
    -m : mail to output report
    -h : help message

[root@otn03 bin]# ddisk
--------------------------------------------------------------------------------
* hpc system disk using info

[otn01] Filesystem      Type      Size   Used Avail Use% Mounted on
[otn01] /dev/sda2       ext3      2.9G   406M   2.4G   15% /
[otn01] /dev/sda1       ext3      198M    23M   165M   13% /boot
[otn01] /dev/sda7       ext3       18G   6.1G    11G   38% /home
[otn01] /dev/sda3       ext3      9.7G   4.7G   4.6G   51% /usr
[otn01] /dev/sda5       ext3      2.0G    90M   1.8G    5% /var
[otn02] Filesystem      Type      Size   Used Avail Use% Mounted on
[otn02] /dev/sda2       ext3      2.9G   425M   2.4G   16% /
[otn02] /dev/sda1       ext3      198M    23M   165M   13% /boot
[otn02] /dev/sda7       ext3       18G   1.5G    15G    9% /home
[otn02] /dev/sda3       ext3      9.7G   4.6G   4.6G   51% /usr
[otn02] /dev/sda5       ext3      2.0G    89M   1.8G    5% /var
[otn03] Filesystem      Type      Size   Used Avail Use% Mounted on
[otn03] /dev/sda2       ext3      2.9G   435M   2.4G   16% /
[otn03] /dev/sda1       ext3      198M    23M   165M   13% /boot
[otn03] /dev/sda7       ext3       18G   8.9G   7.4G   55% /home
[otn03] /dev/sda3       ext3      9.7G   3.2G   6.1G   35% /usr
[otn03] /dev/sda6       ext3      2.0G   244M   1.6G   14% /var
```

* dhalt, dreboot

: **클러스터 시스템 일괄 종료 및 재시작 명령**

dhalt, dreboot 은 클러스터 시스템을 종료 혹은 리부팅 시키는 명령이다.
사용 방법은 다음과 같다.

Usage:
dhalt [hostname] [hostname] [hostname] ....

즉 dhalt 뒤에 종료를 원하는 시스템들의 hostname을 차례로 적어 주면 된다. 만일 hostname 을 별도로 정의 하지 않으면 dutils의 nodelist 에 정의된 모든 시스템에 종료된다. dreboot 역시 같다.

```
[root@otn03 bin]# dhalt
---------------------------------------------------------------------
 Halted otn01 System ..
 Halted otn02 System ..
 Halted otn03 System ..

 Broadcast message from root (pts/0) (Mon Mar 21 20:07:21 2005):

 The system is going down for system halt NOW!

[root@otn03 bin]# dreboot otn01 otn02
---------------------------------------------------------------------
 Rebooted otn01 System ..
 Rebooted otn02 System ..
```

* dsensor

: **클러스터 시스템의 전체 CPU 온도 및 Cooling Fan 온도 모니터링**

* dua2

: **클러스터 시스템의 데이터 일괄 동기화 명령**

dua2 는 분산 다중 시스템의 데이터 일괄 동기화 명령입니다. dua 와의 차이점을 수행 프로토콜을 기존의 rsh 에서 ssh 로 변경을 한것이다. 사용방법은 dua 와 동일 하다.

* dush2

: **클러스터 시스템의 일괄 명령 수행 명령**

dush2 는 분산 다중 시스템의 일괄 명령 수행 프로그램으로 관리 노드에서 dush2 를 통해 입력하는 명령을 클러스터 모든 시스템에 일괄 수행하도록 하는 프로그램이다.
기존의 dush 과 유사하나 몇가지 차이점이 있다.
기존의 dush 는 rsh 를 이용한 방법이라 적절한 보안 조치가 없을 경우 보안에 취약점을 내재 하고 있다. 하지만 dush2 는 ssh을 이용하여 보다 보안에 안전하다.
뿐만 아니라 dush 을 사용하기 위해서 rsh, rlogin 등의 운영체제 기반의 설정이 필요한데 dush2 는 이런 부분을 자동화 해주는 기능을 내재 하고 있다.
dush2 는 관리자 전용으로 사용하며, Teragon HPC Management Tool의 핵심 모듈이다.
사용방법은 dush 와 동일하다. 단 초기 관리자 인증키를 받을 때 --init 이란 옵션을 이용하면 된다.

```
[root@otn03 bin]# ./dush2 --init
 Generating public/private rsa key pair.
 Enter file in which to save the key (/root/.ssh/id_rsa):   <Enter>
 Enter passphrase (empty for no passphrase): <Enter>
 Enter same passphrase again: <Enter>

 root@otn01's password:
```

-> 최초의 인증키 생성 시 관리자 인증을 확인한다. 이는 최초에 한번 이루어 지며 그 이후에는 dush2 를 이용하여 관리자 인증 없이 인증키로 여러 시스템을 동시에 관리 할수 있다.

```
[root@otn03 bin]# ./dush2 -s uptime
[otn01]   23:14:49   up 19 days, 11:06,   2 users,   load average: 0.01, 0.00, 0.00
```

```
 [otn02]   23:08:43   up 23 days,   3:13,   4 users,   load average: 0.00, 0.00, 0.00
 [otn03]   23:14:13   up 27 days,   8:00,   3 users,   load average: 0.00, 0.00, 0.00
```

* dsysinfo

: **클러스터 시스템의 전체 시스템 기본 정보 (kernel, resource) - report**

* dnet

: **클러스터 시스템의 네트워크 사용량 모니터링**

dnet은 클러스터 시스템 별 네트워크 트래픽을 일괄적으로 모니터링 하는 명령이다.
사용법은 다음과 같다.

Usage:
dnet [netdev] [netdev]

즉 dnet 명령 뒤에 트래픽 체크를 할 네트워크 장치명을 적어 주면 된다.

```
[root@otn03 bin]# dnet eth0 eth1
--------------------------------------------------------------------------------
  ** network bandwidth in eth0 device
[otn01] [ I N ]       20.76 Kb/s       [ OUT ]      256 Bits/s
[otn02] [ I N ]       51.45 Kb/s       [ OUT ]    750.03 Kb/s
[otn03] [ I N ]         0 Bits/s       [ OUT ]      256 Bits/s
  ** network bandwidth in eth1 device
[otn01] [ I N ]       29.62 Kb/s       [ OUT ]     14.10 Kb/s
[otn02] [ I N ]      795.41 Kb/s       [ OUT ]     15.96 Kb/s
[otn03] [ I N ]       19.67 Kb/s       [ OUT ]      3.68 Kb/s
```

* pbsaccout

: **사용자별 CPU, MEM 사용 현황 결과 보고서 작성 명령**

- MPI Selector

MPI Selector 는 다양한 환경의 MPI Compiler 를 사용자 입장에서 효율적으로 모두 사용하고자 할때 필요한 툴이다.

즉 mpi 의 경우 리눅스에서 대표적으로 lam-mpi 와 mpich 를 사용하는데 mpi 와 compiler 연동시 개발자가 사용하는 대표적인 컴파일러 하나만을 연동해서 설치를 하게 된다.

**** 여러개 설치는 가능하나 기본적인 MPI 환경 설정(PATH, LIB, INC ..)을 시스템 환경 변수로 설정하여 사용을 하기 때문에 MPI 중 하나를 설치 한후 다른 기종의 MPI 를 사용할려면 사용자 레벨에서 일일이 수동으로 환경 설정을 변경하여 사용하여야 한다.

MPI Selector 는 lam-mpi(gcc), lam-mpi(pgi), lam-mpi(intel), mpich(gcc), mpich(pgi) mpich(intel) 중 원하는 MPI 환경을 사용하기 직전에 사용자가 임의로 선택해서 사용할 수 있다.

병렬 프로그램의 성격에 따라 MPI 종류별로 각각 성능의 차이가 있다. 사용자 입장에서 사용자의 어플리케이션에 최적의 MPI 환경을 손쉽게 찾을 수 있도록 되어져 있다.

사용 방법은 다음과 같다.

```
# mpisel   < MPI >   <Compiler>
```

여기서 < MPI > 에 해당하는 것으로 는 lam 과 mpich 가 있다.
<Compiler> 에 해당하는 것은 gcc, pgi, intel 이 있다.

즉 lam-mpi 에 pgi Compiler 가 연동된 MPI 환경에서 작업 하고 싶은 경우 아래와 같이 입력하면..

```
[root@otn03 bin]# ./mpisel lam pgi

lam-pgi development environment was applied.!!
```

```
otn03(lam-pgi) root # mpicc
pgcc-Warning-No files to process
```

위와 같이 pgi Compiler 와 연동된 MPI 를 사용할 수 있게 된다.
이 작업은 해당 쉘에서만 적용되어지며 logoff 후에는 다시 원래 환경 설정으로 돌아가게 된다.

- NFS Auto Mountor

4. Encluster 2.0 을 이용한 HPC System Monitoring

4.1 Encluster 2.0 설치, 설정

4.2 Encluster 2.0 관리

5. Teragon HPC Security

5.1 Command, Directory Security

5.2 Service Security (rsh, rlogin, ssh, nfs, ftp)

5.3 Network Security (network filtering)

6. HPC Benchmark 성능 분석 하기

6.1 NAS Parallel Benchmark

NAS Parallel Benchmarks (NPB)는 NASA Ames Research Center 로 개발 됐으며, 병렬 컴퓨터의 다양한 성능 수치를 측정 할 수 있는 대표적인 측정 프로그램 중 하나이다.

관련 정보는 다음 사이트에서 확인할 수 있다.

http://www.nas.nasa.gov/Software/NPB

다운로드를 위해서는 간단한 신상정보를 적은 후 사이트 가입 후에 다운을 받을 수 있다.

NPB 는 각각 다른 5개의 Class 로 정의 되어져 있다. 5개의 Class 란 A, B, C, W(orkstation), S(ample)이 있으며 그 차이점은 기본적으로 Problem size (Array size) 와 계산 반복 횟수의 차이에 있다. 일반적으로 대형 컴퓨터의 성능 평가에는 Class B 가 적합한것으로 알려져 있다.

NPB 는 5개의 Paralle kernel benchmarks 와 3개의 CFD (Computational Fluid Dynamics) Application benchmarks 로 구성되어져 있고 각각의 Benchmark를 실행한 결과를 Graph 로 출력할 수 있다.

* EP (The Embarrassingly Paralle Benchmark)

EP (The Embarrassingly Parallel) 벤치마크는 병렬 계산에 적합하고, 지정된 Scheme에 따른다. 다수의 Gaussian pseudorandom numbers 이용하고 , 2 차원의 통계 정보를 축적한 것이다 . 이 문제는 Monte Carlo applications 을 이용한 응용 프로그램에 잘 보여진다 .
이 문제를 풀 때에 전송을 대부분(거의) 필요로 하지 않는 이로부터 , 어느 의미에 있어 다음에 나오는 벤치마크는 시스템이 갖는 부동 소수점 연산의 최대 실효성능을 나타내는 것이라고 말할 수 있다.

* MG (Multigrid Benchmark)

MG(Multigrid Benchmark) 벤치마크는 3 -D Possion PDE 을 간략화한 Multi Grid Method로 풀고 있다 . Class B 는 Class A 와 동일한 크기의 격자를 사용하고 있지만 내부의 루프의 반복 수가 Class A 보다(부터) 커지고 있다 . 또한 MG의 경우 제곱개의 프로세서를 요구한다.

* CG (Conjugate Gradient Benchmark)

CG (Conjugate Gradient) 벤치마크는 노드간 통신성능에 크게 영향을 받는다.
따라서 전용 통신장치를 갖는 super computer에서도 프로세스 수의 증가에 따라 성능이 저하되는 특징이 보여진다. 또한 CG는 대규모적인 symmetric positive definite matrix의 고유의 최소근사치를 Conjugate gradient method를 이용하고 풀고 있다 . kernel 은 비 구조적 격자를 이용한 어플리케이션으로 자주 보여 진다 .

* FT (3-D FFT PDE benchmark)

FT (3-D FFT PDE) 벤치마크는 3-D partial differential equation을 FFT 이용하고 풀고 있다 . many spectral methods의 핵심이 이 kernel을 형성한다. FT 벤치마크는 라이브러리 루틴의 사용이 인정되고 있는 점에서 약간 취지를 달리 하고 있다 .

* LS (Interger Sort benchmark)

IS (Integer Sort) 벤치마크는 particle method codes을 사용한 에플리케이션에 있어 중요한 Sort를 평가한 것이다 . 이 타입의 application은 물리적인 면에서 particle를 잇는 Cell에 할당하고 , particle이 Cell로부터 흘러나오는지 아닌지를 본다. particle-in-cell Type의 에플리케이션과 비슷하다 . Sort는 particle을 다시 한번 적절한 Cell에 할당한 때에 사용된다 .

* LU (Simulated CFD Application (LU) Benchmark)

LU 벤치마크는 LU factorization를 실제로는 행하지 않고 , 5x5의 블록을 갖는 상하 삼각 행렬 시스템을 SSOR(Symmetric Successive Over-Relaxation)법으로 풀고 있다 .
LU는 다른 프로그램과는 달리 대단히 작은 크기의 메시지를 대량으로 주고받는다는 특징이 있다. 따라서 하드웨어 성능보다는 오히려 MPI자체의 작은 크기의 메시지 통신성능에 민감하게 반응하게된다. LU의 경우 하드웨어 성능비교도구보다는 MPI의 작은 크기의 메시지 통신 성능을 비교하기에 좋은 방법이다

* SP (Simulated CFD Application (SP) Benchmark)

SP (Scalar Pentadiagonal) 벤치마크는 X, Y, Z 방향으로 순차적으로 연립방정식을 푸는 것이다 .
각 방향의 연립방정식을 풀 때 multi-partition기법을 사용하는데 이는 큰 크기의 메시지를 소량으로 주고받는다는 점에서 LU와는 대조적이라고 할 수 있다.

* BT (Simulated CFD Application (BT) Benchmark)

BT (Block Tridiagonal) 벤치마크는 SP와 유사한 구조를 갖는다
BT의 경우 프로세서의 수가 증가에 따라서 리눅스 클러스터에서의 병렬화 효율이 감소하는 경향이 있다.

- NPB 설치 하기

NPB Source 파일을 특정 위치에 풀어 놓는다.
NPB 에는 여러 시스템 형태 별로 측정 프로그램이 따로 있는데 여기서는 MPI 관련 측정 툴을 이용하도록 한다.

[root@otn03 clunix]# tar xzvf NPB3.2.tar.gz
[root@otn03 clunix]# cd NPB3.2/NPB3.2-MPI/

make 에 앞서 시스템 별로 make 환경 설정 파일을 변경을 해 주어야 한다. make 환경 설정 파일로 make.def 가 있는데 대표적인 sample 파일이 config 폴더 안에 있다.
이것을 이용하여 make.def 파일을 만들도록 한다.

참고로 NPB 는 MPI 와 Fortran77 을 사용하여 개발되어져 있다.
컴파일을 할 시스템의 MPI 환경에 맞게 make.def 파일의 CLINK, FLINK 와 같은 변수를 수정 한다.

[root@otn03 NPB3.2-MPI]# cd config/
[root@otn03 config]# cp make.def.template make.def
[root@otn03 config]# vi make.def

.
.
#---
This is the fortran compiler used for MPI programs
#---
MPIF77 = /usr/local/lam-gcc/bin/mpif77
# This links MPI fortran programs; usually the same as ${MPIF77}
FLINK = $(MPIF77)
.
.
#---
These macros are passed to the linker to help link with MPI correctly
#---
FMPI_LIB = -L/usr/local/lam-gcc/lib -lmpi

#---
These macros are passed to the compiler to help find 'mpif.h'
#---
FMPI_INC = -I/usr/local/lam-gcc/include
.
.

.

```
#--------------------------------------------------------------------
# This is the C compiler used for MPI programs
#--------------------------------------------------------------------
MPICC = /usr/local/lam-gcc/bin/mpicc
# This links MPI C programs; usually the same as ${MPICC}
CLINK     = $(MPICC)

#--------------------------------------------------------------------
# These macros are passed to the linker to help link with MPI correctly
#--------------------------------------------------------------------
CMPI_LIB    = -L/usr/local/lam-gcc/lib -lmpi

#--------------------------------------------------------------------
# These macros are passed to the compiler to help find 'mpi.h'
#--------------------------------------------------------------------
CMPI_INC = -I/usr/local/lam-gcc/include
```

.

.

위와 같이 환경에 맞게 make.def 파일을 수정한 후 NPB-MPI Top 디렉토리에서 make 를 하면 된다.

NPB는 클래스, 벤치마크툴, CPU 개수 마다 다르게 실행 파일이 만들어 지므로 make 시 해당 실행 파일에 대한 정보를 적어 주어야 한다.

[root@otn03 NPB3.2-MPI]# make LU NPROCS=2 CLASS=S

LU 는 벤치마크 프로그램 이고 NPROCS 는 실행할 프로세스 수이다. CLASS 는 Problem size 에 해당한다.

[root@otn03 NPB3.2-MPI]# make LU NPROCS=4 CLASS=S

위와 같이 Process 2개 와 4개의 해당하는 실행 파일을 만들어 보자

***** 주의 점**

이때 NPROCS 정의 시 특정 프로세스 수를 입력하면 에러가 발생하는 경우가 있다.
NPB 는 8종류의 대상 문제가 정의 되어 있고 문제의 특성에 의해 실행 가능한 프로세스 수가 제한되어 있다.

CT, FT, MG, LU, IS 의 프로세스 수는 2의 승수(1, 2, 4, 8, 16 ..) 로 확장되어 지며,
SP, BT 의 프로세스 수는 n 의 2승 (1, 4, 9, 16, ..) 로 확정되어 진다.

벤치마크 종류 별, 문제 크기별, 프로세스 수별로 각각 컴파일을 하는것이 상당히 비효율적이기에 이것을 한번에 컴파일을 할수가 있다.

config 밑에 보면 suite.def 파일을 만들어 놓고 다음과 같은 형태로 컴파일 해당 변수를 적어 주어 필요한 실행 파일을 일괄적으로 컴파일이 가능 하다.

[root@otn03 config]# vi suite.def

```
---------------------------------------------------------------------------
# config/suite.def
# benchmark_name      class      process_number
ft          S          2
mg          S          2
sp          S          2
lu          S          2
bt          S          2
is          S          2
ep          S          2
cg          S          2
dt          S          2
ft          S          4
mg          S          4
sp          S          4
lu          S          4
bt          S          4
is          S          4
ep          S          4
cg          S          4
dt          S          4
---------------------------------------------------------------------------
```

위와 같이 정의 한 후 make suite 로 suite.def 에 정의 된 실행 파일을 일괄적으로 만들 수 있다.

```
[root@otn03 config]# cd ..
[root@otn03 NPB3.2-MPI]# make suite
```

위와 같이 컴파일하였을 경우 컴파일 된 실행 파일의 NPB-MPI Top 디렉토리 밑의 bin 에 생성 되게 된다.

- NPB 실행 하기

```
[root@otn03 NPB3.2-MPI]# cd bin
```

생성된 실행 파일을 클러스터 각 노드에 동기화를 시켜준 후 mpirun 을 실행하여 보자

```
[clunix@otn03 bin]$ dua *
[clunix@otn03 bin]$ mpirun -np 4 lu.S.4
 ----------------------------------------------------------------------------

  NAS Parallel Benchmarks 3.2 -- LU Benchmark

  Size:   12x 12x 12
  Iterations:   50
  Number of processes:      4

  Time step      1
  Time step     20
  Time step     40
  Time step     50
 .
 .LU Benchmark Completed.
  Class           =                        S
  Size            =                 12x  12x  12
  Iterations      =                       50
  Time in seconds =                     0.14
  Total processes =                        4
  Compiled procs  =                        4
  Mop/s total     =                   727.58
  Mop/s/process   =                   181.89
```

```
Operation type   =              floating point
Verification     =                  SUCCESSFUL
Version          =                          3.2
Compile date     =                16 Mar 2005

----------------------------------------------------------------------------
```

위에 출력값에서 Mop/s total , Mop/s/process, Time in seconds 등으로 성능 향상을 정도를 확인 할 수 있을 것이다.

6.2 HPL을 이용한 Linpack Test Benchmark

HPL(High-performance Linpack Benchmark)은 대용량 메모리 시스템을 벤치 마크 하는데 사용되는 벤치마크툴로 Top 500 Site 에서 사용하는 공식 프로그램으로 CPU 의 Flot 수를 측정하는 프로그램이다. HPL 은 Blas 같은 라이브러리를 이용하여 Ax=B 형태의 주어진 연립 방정식의 해를 구함으로서 컴퓨터의 성능을 측정하는 것이다.

HPL 은 BLAS, CBLAS, ATLAS 와 같은 계산 라이브러리와 MPI 병렬 프로그램을 이용하므로 HPL 를 설치 하기 전에 반드시 위 라이브러리와 MPI 프로그램 (mpich, lam-mpi) 을 설치 하도록 한다.

관련 자료는 http://www.netlib.org/benchmark/hpl/ 에서 hpl.tgz 파일을 다운 받으면 된다.

또 libgoto 라이브러리를 사용하여 실행할때는 http://www.cs.utexas.edu/users/kgoto 에 가서 플랫폼에 맞는 libgoto_opt64p-r0.96-2.so, xerbla.c or xerbla.f 를 다운 받는다.

```
# gcc -c xerbla.c
```

xerbla.o 파일이 생성 될것이다.

아래 예제는 모두 Opteron System 을 기준으로 한것이다.

```
# tar xzvf hpl.tgz
# cd hpl
# cp setup/Make.Linux_ATHLON_CBLAS .
# vi Make.linux_ATHLON_CBLAS
```

```
------------------------------------------------------------------------
SHELL        = /bin/sh
CD           = cd
CP           = cp
LN_S         = ln -s
MKDIR        = mkdir
RM           = /bin/rm -f
TOUCH        = touch

# 플래폼 아키텍처 명을 적어 준다. 즉 Make.XXXXXX 에서 XXXXXX 에 해당되는 명을 적어 준다.
ARCH         = Linux_ATHLON_BLAS

# TOPdir 이 기본적으로는 $(HOME) 으로 잡혀 있다. 이것을 hpl 이 설치된 절대 경로로 변경해준다.
# TOPdir       = $(HOME)/hpl

TOPdir       = /home/clunix/hpl
INCdir       = $(TOPdir)/include
BINdir       = $(TOPdir)/bin/$(ARCH)
LIBdir       = $(TOPdir)/lib/$(ARCH)

HPLlib       = $(LIBdir)/libhpl.a

# 해당 MPI 프로그램 환경 정보를 적어준다.
# MPICH 의 경우에는 MPlib 가 libmpi.a 가 아닌 libmpich.a 이다. 주의 해야함.
# MPdir        = /usr/local/mpi
# MPinc        = -I$(MPdir)/include
# MPlib        = $(MPdir)/lib/libmpich.a


MPdir        = /usr/local/lam-gcc
MPinc        = -I$(MPdir)/include
MPlib        = $(MPdir)/lib/libmpi.a


# 사용할 BLAS 라이브러리 경로를 적어 준다.
# 여기서는 ATLAS 를 사용할 것이므로 ATLAS 가 설치된 PATH를 적어준다.
# LAdir        = $(HOME)/netlib/ARCHIVES/Linux_ATHLON
# LAinc        =
```

```
# LAlib          = $(LAdir)/libcblas.a $(LAdir)/libatlas.a



LAdir          = /usr/local/atlas-gcc/lib/Linux_HAMMER64SSE2_2/
LAinc          =
LAlib          = $(LAdir)/libcblas.a $(LAdir)/libatlas.a

F2CDEFS        =
HPL_INCLUDES = -I$(INCdir) -I$(INCdir)/$(ARCH) $(LAinc) $(MPinc)
HPL_LIBS       = $(HPLlib) $(LAlib) $(MPlib)
HPL_OPTS       = -DHPL_CALL_CBLAS
HPL_DEFS       = $(F2CDEFS) $(HPL_OPTS) $(HPL_INCLUDES)

# 해당 Compiler 와 Linker 를 정의해 준다.
# CC             = /usr/bin/gcc

CC             = /usr/local/lam-gcc/bin/mpicc
CCNOOPT        = $(HPL_DEFS)
CCFLAGS        = $(HPL_DEFS) -fomit-frame-pointer -O3 -funroll-loops -W -Wall

#LINKER         = /usr/bin/gcc
LINKER         = /usr/local/lam-gcc/bin/mpicc
LINKFLAGS      = $(CCFLAGS)

ARCHIVER       = ar
ARFLAGS        = r
RANLIB         = echo
```

****** Xeon Nocona System 에서 문제 크기를 2000 을 넘기면 MPI 가 죽는 문제가 발생**
ATLAS 라이브러리를 Opteron 에서 컴파일한 라이브러리를 사용하였는데 이를 Nocona
에서 컴파일 한 라이브러리로 변경, HPL Makefile 의 C FLAGS 의 -W -Wall 를 빼고..
64bit 시스템을 정의 하는 -m64 를 포함하니 그 이상의 문제 사이즈를 처리 할 수 있었다.

****** HPL 테스트 하는 플랫폼 별로 ATLAS 라이브러리를 만들지 않으면 라이브러리에서 지원**
하는 Memory Size 같은 부분에서 시스템 별로 최적화 될수 없다.

****** 컴파일 역시 -m64를 붙여서 64bit 시스템에 맞게 Compile 이 될수 있게 해야 한다.**

그런 후 Make.top 와 Makefile 의 arch 부분도 위의 arch 부분과 동일하게 변경해 준다.

#

arch = Linux_ATHLON_CBLAS

#

컴파일을 한다.

make build arch=Linux_ATHLON_CBLAS

만일 컴파일 중 Error 가 발생하면 Make 설정 부분을 다시 점검해 보라
컴파일이 무사히 끝나면 bin/Linux_ATHLON_CBLAS 안에 HPL.dat 와 xhpl 파일이 생성되어져 있을 것이다.

xhpl 실행 파일은 계산 노드 전 노드에 동일한 경로에 복사를 해둔다.

dua xhpl

그런 후 HPL.dat 파일을 플랫폼에 맞게 수정하여 xhpl 를 실행 하면 linpack 수치를 구할 수 있다.

```
# default HPL.dat
# vi HPL.dat
--------------------------------------------------------------------------
HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL.out      output file name (if any)
6            device out (6=stdout,7=stderr,file)
4            # of problems sizes (N)
29 30 34 35  Ns
4            # of NBs
1 2 3 4      NBs
0            PMAP process mapping (0=Row-,1=Column-major)
```

```
1               # of process grids (P x Q)
2 1 4           Ps
2 4 1           Qs
16.0            threshold
3               # of panel fact
0 1 2           PFACTs (0=left, 1=Crout, 2=Right)
2               # of recursive stopping criterium
2 4             NBMINs (>= 1)
1               # of panels in recursion
2               NDIVs
3               # of recursive panel fact.
0 1 2           RFACTs (0=left, 1=Crout, 2=Right)
1               # of broadcast
0               BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
1               # of lookahead depth
0               DEPTHs (>=0)
2               SWAP (0=bin-exch,1=long,2=mix)
64              swapping threshold
0               L1 in (0=transposed,1=no-transposed) form
0               U   in (0=transposed,1=no-transposed) form
1               Equilibration (0=no,1=yes)
8               memory alignment in double (> 0)
----------------------------------------------------------------------------

# customizing HPL.dat
# vi HPL.dat
----------------------------------------------------------------------------HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL.out         output file name (if any)
6               device out (6=stdout,7=stderr,file)
1               # of problems sizes (N)
20000           Ns
1               # of NBs
104             NBs
0               PMAP process mapping (0=Row-,1=Column-major)
1                       # of process grids (P x Q)
1                        Ps
6               Qs
```

```
16.0            threshold
1               # of panel fact
1               PFACTs (0=left, 1=Crout, 2=Right)
1               # of recursive stopping criterium
4               NBMINs (>= 1)
1               # of panels in recursion
2               NDIVs
1               # of recursive panel fact.
2               RFACTs (0=left, 1=Crout, 2=Right)
1               # of broadcast
0               BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
1               # of lookahead depth
1               DEPTHs (>=0)
2               SWAP (0=bin-exch,1=long,2=mix)
64              swapping threshold
0               L1 in (0=transposed,1=no-transposed) form
0               U   in (0=transposed,1=no-transposed) form
1               Equilibration (0=no,1=yes)
8               memory alignment in double (> 0)
```

---

xhpl 실행파일을 실행 하기 위해서는 먼저 mpi 프로그램이 설치 되어져 있어야 한다.
이 문서에서의 설정은 lam-mpi 를 기준으로 작성되어져 있으니 lam-mpi 환경에서 구동하는 방법을 설명 하도록
하겠다.

lam-mpi 는 root 에서 실행 할수 없다. 그러므로 일반 계정에 lamboot 을 시키도록 한다.

```
$ cat lamhost
-------------------------------------------
otn01 cpu=2
otn02 cpu=2
otn03 cpu=2
-------------------------------------------
```

먼저 위 시스템은 opteron 244 (1.8G * 2) Process 의 3대 노드로 클러스터가 구성되어져 있다.

```
$ lamboot -v -b lamhost
LAM 7.1.1/MPI 2 C++/ROMIO - Indiana University

n-1<10965> ssi:boot:base:linear: booting n0 (otn03)
n-1<10965> ssi:boot:base:linear: booting n1 (otn02)
n-1<10965> ssi:boot:base:linear: booting n2 (otn01)
n-1<10965> ssi:boot:base:linear: finished

$ mpirun -np 6 xhpl > report
$ vi report
--------------------------------------------------------------------------


================================================================================
HPLinpack 1.0a  --  High-Performance Linpack benchmark  --  January 20, 2004
Written by A. Petitet and R. Clint Whaley,  Innovative Computing Labs.,  UTK
================================================================================

An explanation of the input/output parameters follows:
T/V    : Wall time / encoded variant.
N      : The order of the coefficient matrix A.
NB     : The partitioning blocking factor.
P      : The number of process rows.
Q      : The number of process columns.
Time   : Time in seconds to solve the linear system.
Gflops : Rate of execution for solving the linear system.

The following parameter values will be used:

N      :       29       30       34       35
NB     :        1        2        3        4
PMAP   : Row-major process mapping
P      :        2
Q      :        2
PFACT  :     Left     Crout     Right
NBMIN  :        2        4
```

```
NDIV    :          2
RFACT   :      Left      Crout      Right
BCAST   :     1ring
DEPTH   :          0
SWAP    : Mix (threshold = 64)
L1      : transposed form
U       : transposed form
EQUIL   : yes
ALIGN   : 8 double precision words

----------------------------------------------------------------------------

- The matrix A is randomly generated for each test.
- The following scaled residual checks will be computed:
     1) ||Ax-b||_oo / ( eps * ||A||_1   * N           )
     2) ||Ax-b||_oo / ( eps * ||A||_1   * ||x||_1   )
     3) ||Ax-b||_oo / ( eps * ||A||_oo * ||x||_oo )
- The relative machine precision (eps) is taken to be               1.110223e-16
- Computational tests pass if scaled residuals are less than                16.0

============================================================================
========
T/V                N      NB       P       Q                Time                 Gflops
----------------------------------------------------------------------------
WR00L2L2             29       1       2       2                0.04             4.506e-04
----------------------------------------------------------------------------
||Ax-b||_oo / ( eps * ||A||_1   * N           ) =           0.0674622 ..... PASSED
||Ax-b||_oo / ( eps * ||A||_1   * ||x||_1   ) =           0.0519667 ..... PASSED
||Ax-b||_oo / ( eps * ||A||_oo * ||x||_oo ) =           0.0174238 ..... PASSED
============================================================================
========
T/V                N      NB       P       Q                Time                 Gflops
----------------------------------------------------------------------------
WR00L2L4             29       1       2       2                0.04             4.520e-04
----------------------------------------------------------------------------
||Ax-b||_oo / ( eps * ||A||_1   * N           ) =           0.0674622 ..... PASSED
||Ax-b||_oo / ( eps * ||A||_1   * ||x||_1   ) =           0.0519667 ..... PASSED
||Ax-b||_oo / ( eps * ||A||_oo * ||x||_oo ) =           0.0174238 ..... PASSED
```

.

.

위와 같은 형식의 데이터가 만들어 진다. 위 데이터 수치는 기본 HPL.dat 파일로 실행한것으로 전혀 Customizing 이 되어져 있지 않다. HPL.dat 파일을 자신의 플랫폼에 맞게 최적화 시켜야 한다. 먼저 최적화 HPL.dat 파일을 만든 이후 네트워크 환경 이나 라이브러리, Compiler 와 같은 것을 변경해 가면서 최고의 flops 수를 낼수있는 환경을 찾아 내면 된다.

최적화 HPL.dat 로 실행 했을 경우

```
================================================================================
T/V                N      NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR10R2C4       25000     256     2     3             680.44             1.531e+01
--------------------------------------------------------------------------------
||Ax-b||_oo / ( eps * ||A||_1  * N        ) =        0.0614958 ..... PASSED
||Ax-b||_oo / ( eps * ||A||_1  * ||x||_1  ) =        0.0151841 ..... PASSED
||Ax-b||_oo / ( eps * ||A||_oo * ||x||_oo ) =        0.0026148 ..... PASSED
================================================================================

Finished      20 tests with the following results:
              20 tests completed and passed residual checks,
               0 tests completed and failed residual checks,
               0 tests skipped because of illegal input values.
--------------------------------------------------------------------------------

End of Tests.
================================================================================
```

100M Switch 환경에서 Opteron 6 CPU로 최고 7.5Gflops 까지 나온다. 이때 CPU 사용율이 전체적으로 70% 정도이다. 클러스터 최고의 환경이 완료되면 CPU 사용률이 100% 에 가깝게 나온다.
네트워크,메모리,라이브러리,컴파일러와 같은 환경을 변경해서 CPU 사용률이 100% 가깝게 나올때

측정한 flops 수가 가장 이상적인 수치일것이다. Gigabit 환경에서 CPU 평균 사용율 90% 와 최고 15Gflops 까지 나오는 것을 확인했다.

HPL.dat 튜닝은 http://www.netlib.org/benchmark/hpl/tuning.html 문서를 참고 하라.

6.3 Parall_add 를 이용한 간단한 정수 연산 Benchmark

parall_add 는 A=A+1 과 같은 계산 방식으로 사용자가 지정하는 num 값까지 순차적으로 + 시키는 역활을 병렬화 한 것이다. 정수 연산의 성능을 측정할 수 있는 간단한 MPI 프로그램이다.

```
[clunix@otn03 clunix]$ vi parall_add.c
----------------------------------------------------------------------------/*   This program is for MPI testing

    1. This program add from 1 to input_number.
    2. and report wall clock time and speedup.
*/

#include <mpi.h>
#include <stdio.h>

void notice()
{
    printf("₩n*******************************************************");
    printf("₩n                          Notice !!");
    printf("₩nIf input is not enough large,");
    printf("₩n     Parallel method is not efficient.");
    printf("₩n₩nThis program will add from 1 to your input.");
    printf("₩n*******************************************************");
    return;
}

int main(int argc, char **argv)
{
        unsigned long int i;
    int rank, size;
```

```
    unsigned long int SerialSum, sum;
    unsigned long int TDomain, SDomain;
    unsigned long int start, end;
    double stime1, stime2, ptime1, ptime2;
    float speedup;
    char temp[30];
    MPI_Status status;

    /* initialize MPI */
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD,&size);
    MPI_Comm_rank(MPI_COMM_WORLD,&rank);

    /* read input and distribute domain */
    if(rank == 0)
    {    notice();
         printf("₩nInput integer number : ");
         fscanf(stdin,"%s",&temp[0]);
         TDomain=0;
         TDomain=atoi(temp);
         if(TDomain <= 0)
         {    printf("₩nInput Error!!   TDomain=%lu",TDomain);
         return 1;
         }
         for(i=1; i<size; i++)
         MPI_Send(&TDomain,1,MPI_UNSIGNED_LONG,i,0,MPI_COMM_WORLD);
    }
    else
         MPI_Recv(&TDomain,1,MPI_UNSIGNED_LONG,0,0,MPI_COMM_WORLD,&status);

    /* initialize variables */
         sum=0;
    SerialSum=0;
    ptime1=MPI_Wtime();
    SDomain=TDomain/size;
    start=rank*SDomain+1;
    if(rank != size-1)
         end=start+SDomain-1;
```

```
    else
        end=TDomain;

    /* compute sum of sub_domain */
    for(i=start; i<=end; i++)
        sum=sum+i;

    /* compute sum of gloabl_domain */
    i=sum;
    MPI_Reduce(&i,&sum,1,MPI_UNSIGNED_LONG,MPI_SUM,0,MPI_COMM_WORLD);

    /* compute wall clock time */
    ptime2=MPI_Wtime();
    ptime1=ptime2-ptime1;

    /* compute sum by serial computing */
    stime1=MPI_Wtime();
    for(i=1; i<=TDomain; i++)
        SerialSum=SerialSum+i;

    /* compute wall clock time */
        stime2=MPI_Wtime();
    ptime1=ptime2-ptime1;


    /* report result and speed up */
    if(rank == 0)
    {   printf("₩n   Parallel SUM = %lu,  Wall clock time = %f",sum,ptime2-ptime1);
        printf("₩n   Serial   Sum = %lu,  Wall clcok time = %f",SerialSum,stime2-stime1);
        speedup=(stime2-stime1)/(ptime2-ptime1);
        printf("₩n   SPEED UP = %f₩n",speedup);
    }

    if(rank == 0) printf("₩nGoodbye!  : )₩n",rank);
    MPI_Finalize();
    return 0;
}
```

위 소스를 mpicc 로 컴파일 한다.

```
[clunix@otn03 clunix]$ mpicc -o parall_add parall_add.c
```

생성된 실행 파일을 계산 노드들의 동일한 위치에 복사한다.

```
[clunix@otn03 clunix]$ dua   parall_add

[clunix@otn03 clunix]$ lamboot -v -b lamhosts

LAM 7.1.1/MPI 2 C++/ROMIO - Indiana University

n-1<31303> ssi:boot:base:linear: booting n0 (otn03)
n-1<31303> ssi:boot:base:linear: booting n1 (otn02)
n-1<31303> ssi:boot:base:linear: booting n2 (otn01)
n-1<31303> ssi:boot:base:linear: finished

[clunix@otn03 clunix]$ mpirun -np 6 parall_add

****************************************************
                    Notice !!
If input is not enough large,
    Parallel method is not efficient.

This program will add from 1 to your input.
****************************************************
Input integer number :  10000000      -> 원하는 값을 적는다
```

그럼 아래와 같이 결과가 나온다.

```
    Parallel SUM = 5000000050000000,   Wall clock time = 0.075384
    Serial     Sum = 5000000050000000,   Wall clcok time = 0.507924
    SPEED UP = 6.737829

Goodbye!   : )
```

즉 1대 노드로 계산 한거 보다 6.7 배의 성능 향상이 있다는는 것을 알수 있다.

CPU 아키텍처 별로 성능 차이가 난다. 실제 AMD 계열 프로세스 보다 Pentium 계열 프로세서 시스템에서 성능이 더 월등한것을 알수 있다. 하지만 이는 정수 연산에 해당되는 것이다.
실제 실수 연산 및 부동 소수점 연산등의 성능을 알 수 있는 linpack 테스트 등에서는
AMD 계열이 성능이 우수함을 확인 할수 있다

6.4 Stream 메모리 Benchmark

Stream Memory Benchmark 는 Memory 의 bandwidth 를 측정하는 도구 이다.
HPC 클러스터 시스템과 같은 계산용 시스템에서는 실제 메모리 성능이 전체 성능을 좌우 하게 된다. 현재 시스템에 장착된 메모리의 대역폭이 어느 정도 있지를 측정하는 도구이다.

상세 정보는 다음 사이트에서 확인 할수 있다.

http://www.cs.virginia.edu/stream

* 아래는 local system 의 single cpu에 해당되는 메모리 대역폭을 측정하는 방법이다.

DDR Memory

```
[root@otn03 hpc]# tar xzvf stream_memory.tgz
[root@otn03 hpc]# cd StreamMemory/Code/
[root@otn03 Code]# gcc -O stream.c -o stream
[root@otn03 Code]# ./stream
-------------------------------------------------------------
This system uses 8 bytes per DOUBLE PRECISION word.
-------------------------------------------------------------
Array size = 2000000, Offset = 0
Total memory required = 45.8 MB.
Each test is run 10 times, but only
the *best* time for each is used.
-------------------------------------------------------------
Your clock granularity/precision appears to be 2 microseconds.
Each test below will take on the order of 13981 microseconds.
    (= 6990 clock ticks)
Increase the size of the arrays if this shows that
```

you are not getting at least 20 clock ticks per test.

WARNING -- The above is only a rough guideline.
For best results, please be sure you know the
precision of your system timer.

Function	Rate (MB/s)	Avg time	Min time	Max time
Copy:	1861.8853	0.0155	0.0172	0.0172
Scale:	1838.8761	0.0157	0.0174	0.0175
Add:	2066.1172	0.0209	0.0232	0.0233
Triad:	2045.4199	0.0211	0.0235	0.0235

Solution Validates

SDRAM 측정

This system uses 8 bytes per DOUBLE PRECISION word.

Array size = 2000000, Offset = 0
Total memory required = 45.8 MB.
Each test is run 10 times, but only
the *best* time for each is used.

Your clock granularity appears to be less than one microsecond.
Each test below will take on the order of 62057 microseconds.
(= -2147483648 clock ticks)
Increase the size of the arrays if this shows that
you are not getting at least 20 clock ticks per test.

WARNING -- The above is only a rough guideline.
For best results, please be sure you know the
precision of your system timer.

Function	Rate (MB/s)	Avg time	Min time	Max time
Copy:	320.1118	0.1065	0.1000	0.1890
Scale:	363.1081	0.1126	0.0881	0.2751

Add: 381.4946 0.1270 0.1258 0.1630

Triad: 351.4601 0.1414 0.1366 0.1793

Solution Validates

6.5 Bonnie++ 디스크 I/O Benchmark

Bonnie++ 는 하드디스크와 파일시스템의 성능을 측정하는 대표적인 벤치마크 툴로서 로컬 시스템에서 디스크 쓰기/읽기등을 시뮬레이션하여 디스크와 파일시스템의 성능을 측정하게 된다. 이전의 HPC 클러스터 시스템은 계산 전용으로 사용되어 CPU 와 Mem 그리고 네트워크 자원을 주로 이용하였는데 요즘에는 다양한 어플리케이션이 병렬화 되고, 특정 어플리케이션에서는 엄청난 량의 데이터를 계산 과정에서 생성하는 경우도 종종 있다. 즉 디스크 I/O 과 전체적인 성능에 중요한 역활을 하고 있기에 구축 시스템의 디스크와 파일시스템의 성능 역시 측정해 볼 필요가 있다.

```
[root@otn03 bonnie++-1.93c]# ./configure --prefix=/usr/local/bonnie
[root@otn03 bonnie++-1.93c]# make && make install
[root@otn03 bonnie++-1.93c]# cd /usr/local/bonnie/sbin
[root@otn03 bonnie++-1.93c]# ./bonnie++ -u root -d /data2/tmp -c 10 -s 500 -r 400
```

** bonnie++ 에는 다양한 옵션이 있는데 여기서는 실제 사용에 꼭 필요한 옵션에 대해서만 알아보도록 하겠다

-u : bonnie++ 를 실행할 유저 (bonnie++ 은 root 에서만 사용이 가능)
-d : bonnie++ 에서 시뮬레이션을 진행할 때 생성되는 임시 파일이 생기는 곳이다
-c : 동시 디스크 사용자
-s : 총 생성 디스크 용량
-r : 시뮬레이션때 사용되는 메모리량 (시스템의 실제 메모리 보다 크게 설정하면..
시스템에 문제 발생할수도 있음 - 실제 메모리와 비슷하거나 조금 작게 지정함)

그럼 다음과 같은 형태의 출력값이 나온다.

Using uid:0, gid:0.

Writing a byte at a time...done

Writing intelligently...done

```
Rewriting...done
Reading a byte at a time...done
Reading intelligently...done
start 'em...done...done...done...done...done...
Create files in sequential order...done.
Stat files in sequential order...done.
Delete files in sequential order...done.
Create files in random order...done.
Stat files in random order...done.
Delete files in random order...done.
Version 1.93c       ------Sequential Output------ --Sequential Input- --Random-
Concurrency   10     -Per Chr- --Block-- -Rewrite- -Per Chr- --Block-- --Seeks--
Machine     Size:chnk K/sec %CP K/sec %CP K/sec %CP K/sec %CP K/sec %CP  /sec %CP
alang        500M:1k    62  77 22891  52 13727  22   701  73 50382  29 +++++ +++
Latency                117ms     1176ms      949ms    32032us    30153us       46us
Version 1.93c       ------Sequential Create------ --------Random Create--------
alang                -Create-- --Read--- -Delete-- -Create-- --Read--- -Delete--
                files  /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP
                   16   537  82 +++++ +++ 16909  47   623  92 +++++ +++ 21049  60
Latency                111ms     5793us      115ms    62022us     6945us     8536us
1.93c,1.93c,alang,10,/data2/tmp/test,500M,1k,62,77,22891,52,13727,22,701,73,50382,29,ㅏ
+++++,+++,16,,,,,537,82,+++++,+++,16909,47,623,92,+++++,+++,21049,60,117ms,1176ms,949ms,
32032us,30153us,46us,111ms,5793us,115ms,62022us,6945us,8536us
------------------------------------------------------------------------
```

위와 같은 출력값을 보아 파악하기 힘듬으로 깔끔한 html 형태나 txt 형태로 변환 시켜 주는 프로그램에 bonnie++ 에 포함되어져 있다. bin/ 밑에 보면 bon_csv2txt, bon_csv2html 명령이 그것이다.

```
./bonnie++ -u root -d /data2/tmp -c 10 -s 500:1024 -r 400   | ../bin/bon_csv2txt > diskbench.txt
./bonnie++ -u root -d /data2/tmp -c 10 -s 500:1024 -r 400   | ../bin/bon_csv2html > diskbench.html
```

와 같이 사용하면 된다.

이제 하드디스크 방식과 (IDE, SATA, SCSI, SAN) 파일시스템 방식 (ext3, reierfs, xfs, jfs, nfs,) 의 시스템에서 어느 정도의 성능이 나오는지 판단 할 수 있을 것이다.

아래는 bonnie++ 출력값을 txt 로 변환한 것이다.

Version 1.93c ------Sequential Output------ --Sequential Input- --Random-
-Per Chr- --Block-- -Rewrite- -Per Chr- --Block-- --Seeks--
Machine Size K/sec %CP K/sec %CP K/sec %CP K/sec %CP K/sec %CP /sec %CP
alang 500M:1k 63 79 23631 54 13861 23 710 77 48775 23 +++++ +++
Latency 110ms 1446ms 640ms 15136us 33049us 47us
------Sequential Create------ --------Random Create--------
-Create-- --Read--- -Delete-- -Create-- --Read--- -Delete--
files:max:min /sec %CP /sec %CP /sec %CP /sec %CP /sec %CP /sec %CP
alang 16 735 88 +++++ +++ 20194 67 758 90 +++++ +++ 20888 63
Latency 120ms 5752us 5985us 64043us 5766us 6040us

실제 파일 쓰기를 진행 할때 작업 속도와 CPU 사용량을 알수 있다.

6.6 Netpipe 네트워크 성능 Benchmark

Netpipe(Network Protocol Independent Performaance Evaluator) 는 시스템 간의 네트워크 성능을 측정하는 벤치 마크 프로그램이다. TCP, MPI,PVM 등 다양한 통신 방식별로 네트워크 성능을 측정할 수 있는 툴이 있고..Myrinet, Infiniband, Fast Ethernet, SCI 등과 같은 다양한 디바이스별 네트워크 성능 측정이 가능하다.

관련 벤치마크 툴의 다운로드는 다음 사이트에서 가능하다.

http://www.scl.ameslab.gov/netpipe

먼저 다운 받은 소스를 적당한 위치에서 푼다.

```
[root@otn02 hpc]# tar xzvf NetPIPE_3.6.2.tar.gz
[root@otn02 hpc]# cd NetPIPE_3.6.2
```

makefile 을 수정한다.

makefile 에는 각 측정 대상의 프로그램이 설치된 경로를 적도록 되어져 있다.
각각 해당하는 프로그램 패스를 적절히 적도록 한다.
여기서는 tcp 와 MPI 관련 네트워크 성능 측정을 하겠다.

```
[root@otn02 NetPIPE_3.6.2]# vi makefile
---------------------------------------------------------------------------
CC            = cc
CFLAGS        = -O
SRC           = ./src

# 해당 MPICC 의 PATH 를 적어 넣는다
MPICC         = /usr/local/lam-gcc/bin/mpicc
.
.
MPI2CC     = /usr/local/lam-gcc/bin/mpicc
MPI2_LIB = /usr/local/lam-gcc/lib
MPI2_INC = /usr/local/lam-gcc/include
.
.
---------------------------------------------------------------------------
```

tcp 와 mpi 관련 측정 프로그램을 컴파일 한다.

```
[root@otn02 NetPIPE_3.6.2]# make tcp
[root@otn02 NetPIPE_3.6.2]# make mpi
```

테스트를 할 원격 시스템에 컴파일 된 실행 파일을 복사하던지 해당 시스템에서 동일하게
컴파일을 해준다.

***** TCP Network 성능 측정 방법 *****

```
[root@otn03 NetPIPE_3.6.2]# ./NPtcp  ( remote system )

[root@otn02 NetPIPE_3.6.2]# ./NPtcp -h otn03 ( local system )
.
.
```

```
109: 2097149 bytes      3 times -->     894.64 Mbps in    17884.17 usec
110: 2097152 bytes      3 times -->     894.26 Mbps in    17891.80 usec
111: 2097155 bytes      3 times -->     894.32 Mbps in    17890.65 usec
112: 3145725 bytes      3 times -->     895.49 Mbps in    26800.99 usec
113: 3145728 bytes      3 times -->     895.52 Mbps in    26800.00 usec
114: 3145731 bytes      3 times -->     895.67 Mbps in    26795.67 usec
115: 4194301 bytes      3 times -->     896.17 Mbps in    35707.67 usec
116: 4194304 bytes      3 times -->     896.22 Mbps in    35705.69 usec
117: 4194307 bytes      3 times -->     896.20 Mbps in    35706.32 usec
118: 6291453 bytes      3 times -->     896.78 Mbps in    53524.85 usec
119: 6291456 bytes      3 times -->     896.78 Mbps in    53524.65 usec
120: 6291459 bytes      3 times -->     896.79 Mbps in    53524.18 usec
121: 8388605 bytes      3 times -->     896.97 Mbps in    71351.17 usec
122: 8388608 bytes      3 times -->     896.99 Mbps in    71349.50 usec
123: 8388611 bytes      3 times -->     897.01 Mbps in    71348.35 usec
```

이와 같이 결과가 출력된다. 만일 결과 출력 파일을 파일로 자동 저장 하고 싶을 경우는 local system 에서 -o 옵션 뒤에 저장 파일명을 적어 주면 된다.

```
[root@otn02 NetPIPE_3.6.2]# ./NPtcp -h otn03 -o net_report.txt
```

위 두 시스템의 간의 네트워크 성능 측정 결과는 다음과 같다.

Message Size : 838.8611 Mbyte
Net Bandwidth : 897.01 Mbps
Latency : 71348.35 usec

***** MPI 네트워크 성능 측정 방법 *******

먼저 lamboot 로 mpi 해당 node에 lamd 데몬을 실행한다.

```
[clunix@otn03 clunix]$ lamboot -v -b lamhosts

LAM 7.1.1/MPI 2 C++/ROMIO - Indiana University

n-1<31167> ssi:boot:base:linear: booting n0 (otn03)
```

```
n-1<31167> ssi:boot:base:linear: booting n1 (otn02)
```

** 두 시스템의 동일한 경로에 NPmpi 실행 파일이 존재해야 한다.

```
[clunix@otn03 clunix]$ mpirun -np 2 /usr/local/bin/NPmpi
0: otn03
1: otn02
Now starting the main loop
   0:         1 bytes    2748 times -->        0.24 Mbps in       31.94 usec
   1:         2 bytes    3130 times -->        0.47 Mbps in       32.15 usec
   2:         3 bytes    3110 times -->
.
.
114: 3145731 bytes       3 times -->      893.14 Mbps in    26871.36 usec
115: 4194301 bytes       3 times -->      894.18 Mbps in    35786.83 usec
116: 4194304 bytes       3 times -->      894.05 Mbps in    35791.99 usec
117: 4194307 bytes       3 times -->      894.09 Mbps in    35790.80 usec
118: 6291453 bytes       3 times -->      895.49 Mbps in    53601.98 usec
119: 6291456 bytes       3 times -->      895.50 Mbps in    53601.50 usec
120: 6291459 bytes       3 times -->      895.53 Mbps in    53599.68 usec
121: 8388605 bytes       3 times -->      896.03 Mbps in    71425.84 usec
122: 8388608 bytes       3 times -->      896.04 Mbps in    71425.68 usec
123: 8388611 bytes       3 times -->      895.99 Mbps in    71429.13 usec
```

두 시스템의 MPI 네트워크 성능은 다음과 같다.

```
Message Size : 8388608 bytes
Net Bandwidth : 896.04 Mbps
Latency : 71425.68 usec
```

**** otn02, otn03 노드의 네트워크 환경은 Gigabit Ethernet 에 Gigabit Switch 환경이다.
**** Netpipe 성능 측정은 순수 네트워크 성능만을 측정한것으로 위 수치가 896.04 Mbps 의 성능이 나왔다고 해서 모든 네트워크 프로그램에서 위 성능이 나오는 것은 아니다.
실제 CPU, System I/O, Disk I/O 등등 여러가지 환경에 의해 실제 네트워크 성능이 결정된다.

본 설정은 순수 네트워크 장비의 성능을 측정하는 도구로 보면 된다.

****

이로써 1년 반에 걸친 리눅스 클러스터 관련 기술 정리가 완료가 되었습니다. 나름대로 오랜 시간 다섯 차례에 걸친 세미나를 통해 동료들에게 공유한 기술을 이렇게 하나의 책으로 만들게 되었습니다. 7년간의 리눅스 세계의 일선에서 엔지니어로 활동하면서 많은 것을 보고 배워오면서, 제가 추구하는 엔지니어의 정의와 이상을 실현 하고자 나름대로의 노력을 해왔습니다. 그 사이에 여러 동료를 만나게 되고, 함께 고민하고, 해결해 나가는 과정 속에 그들의 세계 역시 저에게 많은 도움으로 남게 되었습니다. 이제 그들과 함께 소박한 저의 이상 중에 하나인 이 기술을 같이 공유하고자 합니다. 클러스터에 대한 그 동안에 저의 애착이 담긴 서적입니다. 저의 애착을 이해해 주시고 옆에서 언제나 힘이 되어 주신 사장님과 기술부원 여러분께 다시 한번 감사 드립니다.

2005년 3월 29일 서 진우 (alang@clunix.com)