ISBN 978-89-6211-281-8

가시화 시스템의 프로그램 설치 현황

(Software installation status of visualization system)

구 기 범 (voxel@kisti.re.kr)

한 국 과 학 기 술 정 보 연 구 원
Korea Institute of Science & Technology Information

목차

1. 배경 ····· 1

2. 시스템 부팅 ····· 2

가. Power on/off ····· 2

1) Power on 순서 ····· 2

2) Power-off 순서 ····· 3

나. 파일시스템 관리 ····· 5

1) Lustre의 mount 방법 ····· 5

2) OST 서버의 관리 ····· 6

3. OS/디바이스 드라이버 수준 설정 ····· 8

가. 비디오 카드 드라이버의 설치 ····· 8

나. Ganglia ····· 9

다. 네트워크 드라이버 설치 ····· 10

1) 10G NIC ····· 10

2) Infiniband drivers(Obsolete) ····· 12

라. Torque 스케줄러 설치 ····· 14

1) Building & installing torque ····· 14

2) Post setup ····· 16

4. 응용 프로그램 설치 및 설정 ····· 19

가. 프로그램 설치에 관한 일반사항 ····· 19

1) RPM 패키지 ····· 19

2) non-RPM 패키지 ····· 19
나. Chromium ····· 21
1) 컴파일 ····· 21
2) 설치 후 설정 ····· 22
다. VTK ····· 23
1) 설치 조건 ····· 23
2) 세부 설정 ····· 23
라. GoogleEarth ····· 28

5. 결론 ····· 30

표 차례

[표 2-1] 각 노드의 부팅이 완료된 후 실행 중이어야 하는 데몬 ····· 3

[표 3-1] 노드별로 필요한 Torque RPM 파일 ····· 15

소스 차례

[소스 2-1] 정상적인 부팅 과정의 확인 ····· 3
[소스 2-2] MDS01에서의 Lustre(/home) 마운트 ····· 5
[소스 2-3] MDS02에서의 Lustre(/xtmp) 마운트 ····· 5
[소스 2-4] OST에서의 Lustre 마운트 (/home, /xtmp) ····· 5
[소스 2-5] OST를 다운 시킬 경우 MDS에서의 OST 설정 변경 ····· 6
[소스 2-6] OST 개별 마운트 ····· 7

[소스 3-1] /etc/init.d/gmond의 수정 ····· 9
[소스 3-2] nx_nic의 src.rpm 설치 ····· 10
[소스 3-3] spec 파일을 이용한 nx_nic 빌드 ····· 11
[소스 3-4] nx_nic의 RPM 파일 확인 ····· 11
[소스 3-5] nx_nic RPM 파일의 설치 ····· 11
[소스 3-6] spec 파일을 이용한 nx_lsa 빌드 ····· 12
[소스 3-7] nx_lsa의 RPM 파일 확인 ····· 12
[소스 3-8] Infinicon 드라이버 설치 ····· 13
[소스 3-9] Infinicon 드라이버 설치 ····· 13
[소스 3-10] Torque 스케줄러의 configure 스크립트 실행 ····· 14
[소스 3-11] Torque 스케줄러의 RPM 만들기 ····· 14
[소스 3-12] Torque RPM 파일의 확인 ····· 15
[소스 3-13] /var/spool/torque/server_priv/nodse ····· 16
[소스 3-14] qmgr의 설정 ····· 17
[소스 3-15] pbs_server와 pbs_sched의 서비스 등록 ····· 18
[소스 3-16] pbs_mom의 서비스 등록 ····· 18
[소스 3-17] 프로그램 설치 디렉토리 ····· 20
[소스 3-18] Chromium의 util/sdp.c 확인 ····· 21

[소스 3-19] /var/log/messages의 protocol family 등록 확인 ····· 21
[소스 3-20] Chromium의 util/sdp.c 수정 ····· 21
[소스 3-21] CR_SDP_SUFFIX의 설정 ····· 22
[소스 3-22] Mothership 설정 파일 내에서의 서버 추가 ····· 23
[소스 3-23] Mothership 설정 파일의 서버 추가부부분 수정 ····· 23
[소스 3-24] VTK 컴파일을 위한 C 컴파일러 설정 ····· 24
[소스 3-25] VTK 컴파일을 위한 LD_LIBRARY_PATH 설정 ····· 24
[소스 3-26] Picasso에 설치된 MVAPICH2의 mpicxx.h ····· 25
[소스 3-27] VTK 컴파일 과정에서 발생하는 에러 메시지 ····· 26
[소스 3-28] VTK 소스의 수정 ····· 27
[소스 3-29] VTK 소스의 수정 ····· 27
[소스 3-30] Picasso의 /usr/local 내 디렉토리 구조 ····· 28
[소스 3-31] Chromium으로 GoogleEarth를 실행하기 위한 CRMOTHERSHIP의 설정 ····· 28
[소스 3-32] Chromium으로 GoogleEarth를 실행하기 위한 mothership 설정 파일 ····· 29

1. 배경

본 보고서는 KISTI 슈퍼컴퓨팅센터의 주력 가시화 시스템인 Picasso의 소프트웨어 설치 현황과 신규 소프트웨어 설치 시의 특이사항 등에 대해 설명한다.

Picasso는 고해상도 프로젝터가 연결된 클러스터로, 데이터 visualization 용도로는 사실상 국내 최대 규모의 시스템이다. 따라서 기존의 소프트웨어도 가능하면 Picasso의 고성능, 고해상도 출력장치를 충분히 활용하도록 설치하는 것이 유리하다.

본 보고서는 다음과 같이 구성돼있다. 먼저 2장에서는 시스템을 구성하는 각 노드의 부팅 순서와 부팅 과정에서 가장 중요한 공유 스토리지의 제어 방법에 대해 설명한다. 그리고 3장에서는 OS와 디바이스 드라이버 등 다른 어플리케이션을 실행하기 위해 필요한 기본 프로그램의 설치 및 설정방법에 대해 설명한다. 마지막으로 4장에서는 Picasso에 설치된 주요 어플리케이션 일부를 선정해서 Picasso의 환경에 맞도록 설정하는 방법에 대해 다룬다.

2. 시스템 부팅

가. Power on/off

1) Power on 순서

Picasso를 구성하는 노드에 전원을 켜는 순서는 다음과 같다.

- 스토리지 디바이스
- OST 서버
- MDS 서버
- MDS 서버에서 Lustre 수동 마운트
- mgmt01, mgmt02
- gw01 ~ gw09
- input, scheduler, license, web 서버
- master, display, render

이 때 주의해야 할 점은 각 단계가 완전히 종료된 후 다음 단계로 넘어가야 한다는 사실이다. 그리고 render와 display 노드는 모든 노드를 동시에 켜면 마운트 단계에서 스토리지에 과부하가 걸릴 수 있으므로 8대~16대 단위로 묶어서 부팅하는 것이 바람직하다.

모든 노드의 부팅이 끝난 후 display와 render 노드는 display라는 계정으로 정상적인 로그인이 끝나 있어야 한다. 이 때 모든 노드에 일일이 로그인해서 확인하려면 너무 긴 시간이 소요된다. 대신 display라는 계정으로 실행되고 있는 프로세스의 수가 33개 전후면 정상적인 로그인이 된 것으로 간주해도 큰 무리가 없다.

```
$ pdsh -w display[...],render[...] 'ps axu | grep display | wc -l'
| sort
display01: 33
display02: 33
display03: 33
display04: 33
display05: 33
display06: 33
display07: 34
display08: 33
display09: 35
display10: 33
display11: 33
```

[소스 2-1] 정상적인 부팅 과정의 확인

마지막으로 [표 2-1]은 각 노드에 따라서 부팅이 종료된 시점에 실행 중이어야 하는 프로그램을 보여주고 있다.

노드 명	프로그램 명	특이사항
display	ClusterDaemon, nvidia-settings	nvidia-settings는 render01에만 해당
input	trackdserver	
master01	trackd	

[표 2-1] 각 노드의 부팅이 완료된 후 실행 중이어야 하는 데몬

Display와 render 노드는 부팅이 정상적으로 완료되면 display / render 노드에 정상적으로 로그인이 돼있는지 확인하는 방법 : display라는 계정으로 실행되고 있는 프로세스의 수가 33개 전후면 정상적인 로그인이 된 것으로 간주해도 무리가 없다.

2) Power-off 순서

Picasso의 각 노드의 power-off 순서는 다음과 같다.

- X-Windows를 실행하고 있는 모든 노드에서 runlevel을 3으로 전환

- Lustre를 사용하고 있는 모든 노드에서 lustre umount
- master, display, render 노드 종료
- input, scheduler, license, web 서버 종료
- gw01 ~ gw09 종료
- mgmt01, mgmt02 종료
- MDS 서버 종료
- OST 서버 종료
- 스토리지 디바이스 전원 차단

이 때 주의해야 할 사항은 Lustre를 umount 할 때 파일시스템의 부담을 최소로 줄이기 위해 8~16대 단위로 묶어서 순서대로 umount를 진행한다는 점이다.

나. 파일시스템 관리

1) Lustre의 mount 방법

Picasso의 MDS와 OST에서 Lustre filesystem mount는 부트 시점에 자동으로 되지 않는다. 따라서 스토리지 서버를 리부트 한 후에는 다음과 같이 수동으로 마운트를 해줘야 한다.

- MDS01의 lustre 마운트 방법 : 여기서 /MDSHOMEDATA는 lustre 파일시스템에 대한 메타데이터를 의미하고, lfs_mds는 OASIS 파일시스템을 위한 메타데이터 서비스다.

```
[root@mds01 ~]mount -t lustre /dev/cciss/c0d0p2 /MDSHOMEDATA
[root@mds01 ~]/opt/lake-vfs/bin/lfs_mds start
```

[소스 2-2] MDS01에서의 Lustre(/home) 마운트

- MDS02의 lustre 마운트 방법 : MDS02는 /xtmp를 총괄한다.

```
[root@mds02 ~]mount -t lustre /dev/cciss/c0d0p2 /MDSXTMPDATA/
```

[소스 2-3] MDS02에서의 Lustre(/xtmp) 마운트

- 각 OST는 다음과 같은 방법으로 파일시스템을 마운트하면 된다.

```
[root@mds01 ~]# pdsh -w ost[01-19] mount -t lustre /dev/cciss/c0d
0p1 /HOMEDATA
[root@mds01 ~]# pdsh -w ost[01-20] mount -t lustre /dev/cciss/c1d
0 /XTMPDATA1
[root@mds01 ~]# pdsh -w ost[01-20] mount -t lustre /dev/cciss/c2d
0 /XTMPDATA2
[root@mds01 ~]# pdsh -w ost[01-19] /opt/lake-vfs/bin/lfs_ds start
```

[소스 2-4] OST에서의 Lustre 마운트 (/home, /xtmp)

위의 모든 작업이 성공한 후 다른 노드(master, render 등)에서 /home과 /xtmp를 마운트하면 된다. 여기서 주의해야 할 사항은 ost20번의 c0d0은 백업용으로 사용하기 위해 ost20번의 로컬 드라이브 /DSDATA로 마운트 되어있고(따라서 /home으로 사용하지 않는다), ost[01-19]번의 c0d0p2는 /home의 백업용으로 mgmt02에 /homebackup으로 마운트 되어 있다.

2) OST 서버의 관리

OST 서버 한 대를 다운시켜야 할 경우 mds서버에서 해당 ost 서버를 deactive 시키면 해당 OST 서버가 다운돼있는 상황에서도 지속적인 파일시스템 운영이 가능하다.

```
[root@mds01~]lctl conf_param homefs-OST0006.osc.active=0
```

[소스 2-5] OST를 다운 시킬 경우 MDS에서의 OST 설정 변경

OST 서버를 다시 active 시키려면 끝의 0을 1로 바꾼다. 그러면 약간의 시간이 걸리며 MDS와 OST간의 time sync가 맞춰지면서 OST가 활성화된다.

homefs-OST0006은 타깃 스토리지 넘버로, 클라이언트에서 lfs check osts를 하면 나오는 번호들이다. OST에서 문제가 생겼을 경우 MDS에서 Lustre error 메세지와 함께 해당 OST의 IB(인피니밴드) 주소와 타깃 스토리지 번호가 출력된다.

MDS에서 러스터 파티션을 마운트 하다가 커널 패닉이 발생하는 것은 MDS와 OST가 비정상적으로 끊어졌기 때문일 가능성이 높다. 이때에는 MDS에서 러스터 파티션을 마운트 할 때 -extent 옵션을 주고 마운트 한 후 OST를 하나씩 마운트하면 된다. OST를 마운트 할 때는 MDS가 정상적으로 OST를 인식할 때 까지 기다려야 한다.

```
[root@mds01 ~]# mount -t lustre -o extent /dev/ccciss/c0d0p2 /MDS
HOMEDATA
```

[소스 2-6] OST 개별 마운트

기타 스토리지 디바이스나 파일시스템에 대한 자세한 사항은 http://sv.ksc.re.kr/svwiki/MountingPicassoLustre를 참고한다.

3. OS/디바이스 드라이버 수준 설정

가. 비디오 카드 드라이버의 설치

Picasso의 master, display, render 노드에는 NVIIDA QuadroFX5600이 설치돼있다. 이 카드의 성능을 제대로 발휘하려면 전용 디바이스 드라이버를 설치해야 하는데, 이는 yum으로 관리가 안 되므로 수동으로 설치해야 한다. 여기서 주의해야 할 사항은 master와 render는 CUDA도 지원해야 하므로, CUDA를 지원하는 디바이스 드라이버를 설치해야 한다는 점이다.

X-Window를 업그레이드하면 OpenGL 라이브러리를 덮어쓰는 경우가 발생하는데, 이때는 반드시 NVIDIA 드라이버를 다시 설치해야 한다. 가끔 제대로 설치가 안 될 수도 있으므로 두 번 연속 설치하는 것이 확실하다. 그리고 64-bit 시스템에서 설치할 경우 32-bit OpenGL도 설치할지의 여부를 물어오는데, 이에 대해서는 Yes를 선택해서 32-bit 어플리케이션의 실행도 지원하도록 해야 한다.

프로그램을 그냥 실행하면 여러 가지를 물어오기 때문에 여러 노드에 동시에 설치할 경우에는 상당히 불편하다. 이 경우에는 -s 옵션을 이용해서 바로 설치할 수 있다.

나. Ganglia

일반적으로 gmond를 실행하면 239.2.11.71로 자신의 상태를 broadcast한다. 이 때 여러 개의 네트워크 카드가 붙어있는 호스트의 경우 어떤 호스트에 나가는지 확실히 정해주기 위해서 Picasso에서는 다음과 같이 /etc/init.d/gmond를 수정했다. (route add -host .. 부분을 주의해서 봐야 한다) 주의할 사항은 현재 gw05는 eth0가 아닌 eth1로 되어 있다는 점이다.

```
 생략

RETVAL=0

case "$1" in
   start)
      echo -n "Starting GANGLIA gmond: "
      [ -f $GMOND ] || exit 1

      route add -host 239.2.11.71 dev eth0

      daemon $GMOND
      RETVAL=$?
      echo
      [ $RETVAL -eq 0 ] && touch /var/lock/subsys/gmond
        ;;

  stop)
      echo -n "Shutting down GANGLIA gmond: "
      killproc gmond
      RETVAL=$?

      route del -host 239.2.11.71 dev eth0

      echo
      [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/gmond
        ;;

 생략
```

[소스 3-1] /etc/init.d/gmond의 수정

다. 네트워크 드라이버 설치

Picasso의 일부 노드에는 기가빗 이더넷 외에 10G NIC과 Infiniband HCA가 꽂혀있다. 기가빗 이더넷은 리눅스 커널이 기본으로 지원하지만 10G NIC과 IB HCA는 별도의 드라이버를 설치해야 한다.

1) 10G NIC

가) 일반사항

Picasso가 사용하는 10G 네트워크 카드의 정확한 모델명은 NC510F PCIe 10gigabit server adapter로, master, display, gw01 ~ gw09 등 모두 28대의 노드에 설치돼있다.

이 네트워크 카드의 디바이스 드라이버는 HP 홈페이지에서 다운받을 수 있는데, 받아야 할 프로그램은 device driver, socket acceleration driver, firmware utility의 세 개로 각각 nx_nic, nx_lsa, nx_tools라는 이름을 갖고 있다. 이 때 tar.gz이 아닌 src.rpm을 받아야 한다.

10G NIC이 꽂혀있는 노드를 살펴보면 master와 display에는 RHEL5가 설치돼있고, gw노드에는 CentOS5가 설치돼있다. 따라서 RHEL5가 설치된 노드 중 한 대에서 RHEL5용 RPM을 만들고, CentOS5가 설치된 노드 중 한 대에서 CentOS5용 RPM을 각각 만들어야 한다. 이 얘기는 nx_lsa와 nx_nic에만 적용되고, nx_tools는 noarch이므로 그냥 바로 설치하면 된다.

나) nx_nic의 설치

HP의 홈페이지에서 nx_nic-*-src.rpm을 받아와서 다음과 같이 설치한다.

```
$ rpm -i nx_nic-3.4.336-1.src.rpm
```

[소스 3-2] nx_nic의 src.rpm 설치

위의 명령이 성공적으로 끝났다고 해서 설치가 마무리된 것은 아니다. 다시 /usr/src/redhat/SPEC으로 들어가서 rpmbuild를 실행해야 한다.

```
$ cd /usr/src/redhat/SPECS/
$ rpmbuild -ba nx_nic-3.4.336.spec
```

[소스 3-3] spec 파일을 이용한 nx_nic 빌드

위의 명령이 성공적으로 끝나면 /usr/src/redhat/RPMS/x86_64에 RPM 파일이 만들어져 있다.

```
$ cd /usr/src/redhat/RPMS/x86_64
$ ls
nx_nic-3.4.336-1.x86_64.rpm
```

[소스 3-4] nx_nic의 RPM 파일 확인

각 노드에 만들어진 RPM을 설치한다. 앞에서도 설명했지만 RHEL5와 CentOS5를 반드시 구별해서 각각 build하고 설치한다.

다) nx_lsa의 설치

HP의 홈페이지에서 nx_lsa-*-src.rpm을 받아와서 다음과 같이 설치한다.

```
$ rpm -i nx_lsa-3.4.336-1.src.rpm
```

[소스 3-5] nx_nic RPM 파일의 설치

위의 명령이 성공적으로 끝났다고 해서 설치가 마무리된 것은 아니다. 다시 /usr/src/redhat/SPEC으로 들어가서 rpmbuild를 실행해야 한다.

```
$ cd /usr/src/redhat/SPECS/
$ rpmbuild -ba nx_lsa-3.4.336.spec
```

[소스 3-6] spec 파일을 이용한 nx_lsa 빌드

위의 명령이 성공적으로 끝나면 /usr/src/redhat/RPMS/x86_64에 RPM 파일이 만들어져 있다.

```
$ cd /usr/src/redhat/RPMS/x86_64
$ ls
nx_lsa-3.4.336-1.x86_64.rpm
```

[소스 3-7] nx_lsa의 RPM 파일 확인

각 노드에 만들어진 RPM을 설치한다. 앞에서도 설명했지만 RHEL5와 CentOS5를 반드시 구별해서 각각 build하고 설치한다.

라) nx_tools의 설치

nx_tools는 특별히 architecture 특성을 타지 않으니 RHEL5와 CentOS5의 구별이 필요 없고, 모든 노드에서 바로 RPM으로 설치하면 된다.

2) Infiniband drivers(Obsolete)

Picasso가 사용하는 인피니밴드 카드의 정확한 모델명은 InfiniHost III Lx HCA(MT25204)로, 웹 서버 등 극히 일부의 노드를 제외한 모든 노드에 장착돼있다. 여기서는 QLogic에서 제공하는 인피니밴드 드라이버의 설치법을 설명하는데, 2008년 8월 파일시스템을 업그레이드하면서 QLogic 드라이버를 OFED로 교체했다. (OFED의 설치방법은 다음 보고서에 추가할 예정이다)

QLogic 드라이버는 RPM을 제공하지 않는다. 따라서 설치된 리눅스 버전에 대응하는 tgz를 가져와서 설치한다.

```
$ tar zxvf InfiniServ.4.1.1.0.15G.tgz
   Pre-compiled binary의 경우 INSTALL만 실행하면 된다.
$ cd InfiniServ.4.1.1.0.15G
$ ./INSTALL
   아래의 메뉴에서 1번을 선택한다.
   1) Install/Uninstall Software
   2) Reconfigure IP over IB
   3) Reconfigure Driver Autostart
   4) Update HCA Firmware
   5) Generate Supporting Information for Problem Report
   6) Host Setup via Fast Fabric
   7) Host Admin via Fast Fabric
   8) Chassis Admin via Fast Fabric
   9) Externally Managed Switch Admin via Fast Fabric

   X) Exit
```

[소스 3-8] Infinicon 드라이버 설치

위에서 1을 선택하면 다음의 화면이 나오는데, 이 중 0, 1, 6, b를 설치하도록 선택한다. 특히 IPOIB와 SDP를 꼭 선택해야 한다.(MPI는 별도로 설치한다)

```
Please Select Install Action:

0) IB Network Stack  [   Install    ] [Available]
1) IB Development    [   Install    ] [Available]
2) IB Boot          [Do Not Install] [Not Avail]
3) Fast Fabric       [Do Not Install] [Not Avail]
4) Virtual HBA (SRP) [Do Not Install] [Available]
5) Virtual NIC       [Do Not Install] [Available]
6) IP over IB        [   Install    ] [Available]
7) MPI Runtime       [Do Not Install] [Available]
8) MPI Development   [Do Not Install] [Available]
9) MPI Source        [Do Not Install] [Available]
a) uDAPL            [Do Not Install] [Available]
b) SDP              [   Install    ] [Available]
c) RDS              [Do Not Install] [Available]
```

[소스 3-9] Infinicon 드라이버 설치

모든 선택이 끝나면 P를 눌러서 나머지 과정을 전부 진행한다. 대부분의 물음에는 default로 대답하면 된다.

라. Torque 스케줄러 설치

Torque는 가장 단순한 스케줄링 알고리즘을 구현한 스케줄러를 사용한다(pbs_sched). 물론 필요에 따라서는 MAUI나 MOAB과 같은, 더 좋은 스케줄러를 사용할 수도 있겠으나 Picasso가 한 번에 여러 명의 작업을 실행하는 경우는 그리 많이 않을 것이라 생각해서 pbs_sched를 그대로 사용하기로 했다.

1) Building & installing torque

여기서 설명하는 모든 과정은 scheduler 노드에서 실행해야 한다. Torque는 우선 tarball을 갖고 와서 RPM으로 만든 후에 설치한다. Picasso에서 torque 2.3.3을 설치하려고 하면 RPM을 만드는 과정에 에러가 있기 때문에 2.3.2를 설치했다.

Torque를 가져와서 압축을 푼 후, configure 스크립트는 다음과 같이 실행한다. 한 가지 주의해야 할 사항은 --with-default-server를 별도로 지정하고, 그 값을 scheduler-ib로 지정한다는 점이다. MVAPICH2와의 연계를 생각해보면 torque에서 만들어내는 hostfile도 infiniband로 연결되는 호스트이름이어야 할 것이라고 생각했고, 이를 위해서는 스케줄러에서도 IB와 연결되는 호스트로 인식해야 할 것으로 판단했다.

```
$ ./configure --prefix=/r --with-rcp=scp --libdir=/usr/lib64 --wi
th-default-server=scheduler-ib
```

[소스 3-10] Torque 스케줄러의 configure 스크립트 실행

Configure의 실행이 끝난 후 RPM을 바로 만든다.

```
$ make PROMPTS="--target x86_64" rpm
```

[소스 3-11] Torque 스케줄러의 RPM 만들기

위의 과정이 아무 문제없이 끝나면 /usr/src/redhat/RPMS/x86_64에 다음의 파일들이 만들어진다.

```
$ ls -l /usr/src/redhat/RPMS/x86_64
total 1168
-rw-r--r-- 1 root root 106378 Aug 29 16:08 torque-2.3.2-1cri.x86_
64.rpm
-rw-r--r-- 1 root root 128842 Aug 29 16:08 torque-client-2.3.2-1c
ri.x86_64.rpm
-rw-r--r-- 1 root root 123529 Aug 29 16:08 torque-devel-2.3.2-1cr
i.x86_64.rpm
-rw-r--r-- 1 root root 475972 Aug 29 16:08 torque-docs-2.3.2-1cr
i.x86_64.rpm
-rw-r--r-- 1 root root 142694 Aug 29 16:08 torque-mom-2.3.2-1cri.
x86_64.rpm
-rw-r--r-- 1 root root  43273 Aug 29 16:08 torque-scheduler-2.3.2
-1cri.x86_64.rpm
-rw-r--r-- 1 root root 138653 Aug 29 16:08 torque-server-2.3.2-1c
ri.x86_64.rpm
```

[소스 3-12] Torque RPM 파일의 확인

만들어진 RPM을 각 노드의 역할에 따라서 선택적으로 설치한다. 각 노드에 따라서 설치하는 RPM은 다음과 같다.

	scheduler	master	render
torque-scheduler-2.3.2-1cri	○		
torque-2.3.2-1cri	○	○	○
torque-server-2.3.2-1cri	○		
torque-client-2.3.2-1cri	○	○	○
torque-docs-2.3.2-1cri	○	○	○
torque-devel-2.3.2-1cri		○	○
torque-mom-2.3.2-1cri			○

[표 3-1] 노드별로 필요한 Torque RPM 파일

위에서 제시한 노드 외의 노드(display 포함)에는 아무것도 설치할 필요가 없다.

2) Post setup

가) /var/spool/torque/server_priv/nodes

Torque가 관리하는 노드를 저장하는 파일로 scheduler 서버에 있어야 한다.

```
$ cat /var/spool/torque/server_priv/nodes
render01-ib np=2
render02-ib np=2
render03-ib np=2
render04-ib np=2
render05-ib np=2
render06-ib np=2
render07-ib np=2
render08-ib np=2
render09-ib np=2
render10-ib np=2
render11-ib np=2
...
```

[소스 3-13] /var/spool/torque/server_priv/nodse

원래 정책에 따르면 np=6이어야 하지만 2008년 10월 현재 외부 사용자 지원을 위해 임시로 2로 맞춰놓은 상태다.

나) pbs_server의 설정

가장 주의해서 봐야 할 부분이 queue 설정의 resources_available.nodect와 resources_assigned.nodect로, 이것은 노드 수 x processor per node로 정해야 한다. 원래의 정책대로 한다면 540개로 맞춰져야 하겠으나 2008년 10월 현재 외부업체의 지원 때문에 ppn을 2로 맞춰서 사용하고 있다(원래는 540으로 맞추는 것이 맞다)

마찬가지로 server 설정의 resources_assigned.nodect도 적절한 값을 갖도록 바꿔야 한다. 나머지 설정은 아래의 내용대로 고치면 된다.

```
$ qmgr
Max open servers: 4
Qmgr: list server
Server scheduler
       server_state = Active
       scheduling = True
       total_jobs = 498
       state_count = Transit:0 Queued:318 Held:0 Waiting:0 Runnin
g:179 Exiting:1
       acl_hosts = scheduler,scheduler-ib
       default_queue = batch
       log_events = 511
       mail_from = adm
       resources_assigned.nodect = 180
       scheduler_iteration = 60
       node_check_rate = 120
       tcp_timeout = 6
       pbs_version = 2.3.2
       submit_hosts = master01,master02,master03,master01-ib,mas
ter02-ib,
                    master03-ib
       allow_node_submit = False
       next_job_number = 1326
       net_counter = 80 33 38

Qmgr: list queue batch
Queue batch
       queue_type = Execution
       total_jobs = 498
       state_count = Transit:0 Queued:318 Held:0 Waiting:0 Runnin
g:174 Exiting:6
       resources_max.nodect = 90
       resources_min.nodect = 1
       mtime = Mon Sep  1 12:39:24 2008
       resources_available.nodect = 180
       resources_assigned.nodect = 180
       enabled = True
       started = True

Qmgr:
```

[소스 3-14] qmgr의 설정

다) /etc/init.d/pbs_*

노드의 특성에 따라서 시스템 부트 과정에 시작하는 서비스의 종류가 다르다. 예를 들어서 master에서는 별다른 서비스를 실행할 필요가 없지만 scheduler 노드에서는 pbs_server와 pbs_sched가 추가돼야 한다.

```
$ chkconfig pbs_server on
$ chkconfig pbs_sched on
```

[소스 3-15] pbs_server와 pbs_sched의 서비스 등록

render 노드에서는 pbs_mom이 추가돼야 한다.

```
$ chkconfig pbs_mom on
```

[소스 3-16] pbs_mom의 서비스 등록

4. 응용 프로그램 설치 및 설정

가. 프로그램 설치에 관한 일반사항

1) RPM 패키지

RPM 개수는 다음과 같이 노드를 그룹지어서, 동일한 그룹에 속하는 노드들은 항상 같은 버전/같은 수의 RPM이 설치되어 있어야 한다(다시 말해서 7개의 노드 그룹군이 존재한다는 뜻이다).

- master01~03, display01~16, render01~90
- mgmt01~02
- gw01~09
- input
- license
- scheduler
- web

2) non-RPM 패키지

RPM 형태로 제공되지 않는 응용 프로그램은 /usr/local에 설치한다. 상용 S/W는 이름을 /usr/local 밑에 대문자로 된 디렉터리를 만들고, 그 밑에 설치한다. 보통 autoconf/automake 기반의 어플리케이션은 prefix를 /usr/local/프로그램이름(대문자)로 지정하면 된다. 프로그램 이름에는 버전도 포함한다. (예: Software-x.y.z)

대표명칭으로 버전 정보가 들어가지 않는 심볼릭 링크를 만들고(예: Software를 Software-x.y.z으로의 심볼릭 링크로 만든다) 어플리케이션의 모든 configuration에는 대표명칭으로 사용된 심볼릭 링크가 들어가도록 한다.

- 옳은 예: 설치 디렉토리=/usr/local/Software

- 틀린 예: 설치 디렉토리=/usr/local/Software-x.y.z

위와 같이 하면 버전만 다른 같은 소프트웨어가 있어도 상황에 따라 심볼릭 링크만 바꿔가면서 운영할 수 있게 된다.

```
lrwxrwxrwx  1 root root    22 Feb  1 17:28 CAVE -> /usr/local/CAV
E.32-bit
drwxr-xr-x  7 root root  4096 Jan  8 14:42 CAVE.32-bit
drwxr-xr-x 12 root root  4096 Jan 27 00:03 CAVE.64-bit

...

lrwxrwxrwx  1 root root   11 Feb  4 23:00 MPICH -> MPICH-1.2.5
drwxr-xr-x 13 root root  4096 Feb  1 01:58 MPICH-1.2.5
lrwxrwxrwx  1 root root   12 Feb  4 23:01 MPICH2 -> MPICH2-1.0.2
drwxr-xr-x 14 root root  4096 Feb  4 22:46 MPICH2-1.0.2
drwxr-xr-x 14 root root  4096 Feb 12 13:33 MPICH2-Intel
drwxr-xr-x 13 root root  4096 Feb  8 23:42 MPICH-Intel

...

lrwxrwxrwx  1 root root    9 Feb  1 02:22 VTK -> VTK-5.0.4
drwxr-xr-x  6 root root  4096 Feb  1 02:20 VTK-5.0.4
```

[소스 3-17] 프로그램 설치 디렉토리

나. Chromium

1) 컴파일

Chromium의 소스 중 util/sdp.c를 보면 아래와 같은 라인이 존재한다.

```
#ifndef AF_INET_SDP
#define AF_INET_SDP 26
#endif
```

[소스 3-18] Chromium의 util/sdp.c 확인

SDP를 사용하는 것 자체는 일반 socket 프로그래밍에서 크게 달라지는 것이 없다. 다만 socket 시스템 콜을 호출할 때 첫 번째 파라메터로 들어가는 domain이 위의 값으로 대치되는데, 이것이 SDP 모듈이 등록하는 네트워크 프로토콜 번호와 일치해야 한다. 예를 들어 QLogic(과거 Infinicon)의 인피니밴드 드라이버를 사용할 경우 ics_sdp 모듈을 로드하면 아래와 같은 메시지를 볼 수 있는데, 여기서 protocol family가 27번으로 등록됨을 알 수 있을 것이다.

```
NET: Registered protocol family 27
```

[소스 3-19] /var/log/messages의 protocol family 등록 확인

따라서 위의 소스에서 AF_INET_SDP를 27로 변경해야 Chromium이 QLogic의 SDP를 사용할 수 있게 된다.

```
#ifndef AF_INET_SDP
#define AF_INET_SDP 27
#endif
```

[소스 3-20] Chromium의 util/sdp.c 수정

하지만 Picasso를 새로 설치하면서 SDP의 protocol family가 30으로

바뀌었다. 따라서 위의 27을 30으로 바꿔야 제대로 SDP를 사용할 수 있었다.

2) 설치 후 설정

가) CR_SDP_SUFFIX

많은 경우 클러스터는 기가빗 이더넷, 인피니밴드 등 서로 다른 종류의 네트워크 카드를 동시에 사용하고, 그에 따라서 한 노드에 대해 두 개 이상의 hostname을 지정하기 때문에 할 것이다. 이 경우 Chromium은 인피니밴드에 대응하는 hostname을 사용하도록 설정해야 하는데, 이 때 사용하는 환경변수가 바로 CR_SDP_SUFFIX다.

```
# System configuration
SHELL     = "/usr/bin/ssh"
ENV       = "CRMOTHERSHIP=master01 CR_SDP_SUFFIX='-ib' LD_LIBRARY
_PATH=/usr/local/CR/lib/Linux:/usr/java/default/jre/lib/amd64/se
rver/:/usr/java/default/jre/lib/amd64/xawt/"
CRSERVER   = "/usr/local/CR/bin/Linux/crserver -mothership master
01"
CRAPPFAKER = "/usr/local/CR/bin/Linux/crappfaker -mothership mas
ter01"
```

[소스 3-21] CR_SDP_SUFFIX의 설정

나) MTU

mothership 설정 파일에서 MTU를 충분히 크게 잡아주지 않으면 buffer 크기가 작다는 경고 메시지와 함께 tilesort SPU의 성능도 많이 떨어지는 것을 볼 수 있다. Picasso에서는 MTU를 최소한 2MB 이상으로 잡아줘서 경고 메시지를 제거할 수 있었고, 실제 운용을 위해서는 8388608(8MB)로 설정해서 사용하고 있다.

다) Mothership 설정

거의 모든 mothership configuration에는 다음과 같은 라인이 존재할

것이다.

```
tilesortspu.AddServer( servernode, protocol='tcpip', port=7000 +
n )
```

[소스 3-22] Mothership 설정 파일 내에서의 서버 추가

위의 tcpip를 sdp로 바꿔주면 된다. 단, SDP 모듈이 실행되고 있음을 반드시 확인해야 한다.

```
tilesortspu.AddServer( servernode, protocol='sdp', port=7000 + n
)
```

[소스 3-23] Mothership 설정 파일의 서버 추가부부분 수정

다. VTK

1) 설치 조건

Picasso에 VTK를 설치할 때에는 다음의 조건을 충족시키도록 했다.

- VTK가 지원하는 모든 programming language interface (TCL, C++, JAVA) 지원
- JAVA는 SUN JAVA 사용
- MPI를 이용한 parallel rendering 지원

2) 세부 설정

가) C 컴파일러 지정

CMAKE에서 사용하는 default C compiler를 mpicc로 지정한다(C++ 컴파일러는 기본값을 그대로 사용하면 된다). ccmake를 실행하기 전에 default compiler를 바꾸기 위한 환경 변수를 다음과 같이 정한다.

```
# export CC=/usr/local/MPI/bin/mpicc
```

[소스 3-24] VTK 컴파일을 위한 C 컴파일러 설정

나) MPI

다음과 같이 설정한다. 여기서 VTK_MPI_POSTFLAGS는 충분한 테스트를 거치지 않았으므로 추가의 테스트가 필요한 상황이다.

- VTK_MPI_MAX_NUMPROCS : 512
- VTK_MPIRUN_EXE : /usr/local/MPI/bin/mpirun_rsh
- VTK_MPI_POSTFLAGS : -hostfile /etc/mpi/hostfile

다) JAVA

SUN JAVA를 사용할 때 LD_LIBRARY_PATH를 다음과 같이 정해준다.

```
#  export  LD_LIBRARY_PATH=/usr/java/default/jre/lib/amd64/server
/:/usr/java/default/jre/lib/amd64/xawt/
```

[소스 3-25] VTK 컴파일을 위한 LD_LIBRARY_PATH 설정

CCMAKE에서는 다음과 같이 설정한다.

- JAVACOMMAND : /usr/java/default/bin/java
- JAVA_ARCHIVE : /usr/java/default/bin/jar
- JAVA_AWT_INCLUDE_PATH : /usr/java/default/include/linux
- JAVA_AWT_LIBRARY : /usr/java/default/jre/lib/amd64/libjawt.so
- JAVA_COMPILE : /usr/java/default/bin/javac
- JAVA_INCLUDE_PATH : /usr/java/default/include
- JAVA_INCLUDE_PATH2 :
- JAVA_JVM_LIBRARY : /usr/java/default/jre/lib/amd64/server/libjvm
- JAVA_RUNTIME : /usr/java/default/bin/java

라) MVAPICH2와 SEEK_SET, SEEK_CUR, SEEK_END

/usr/local/MPI/include/mpicxx.h를 보면 다음의 메시지를 확인할 수 있다.

```
// There is a name conflict between stdio.h and the MPI C++ bindin
g
// with respect to the names SEEK_SET, SEEK_CUR, and SEEK_END.  MP
I
// wants these in the MPI namespace, but stdio.h will #define thes
e
// to integer values.  #undef'ing these can cause obscure problem
s
// with other include files (such as iostream), so we instead use
// #error to indicate a fatal error.  Users can either #undef
// the names before including mpi.h or include mpi.h *before* std
io.h
// or iostream.
```

[소스 3-26] Picasso에 설치된 MVAPICH2의 mpicxx.h

이것 때문에 VTK가 MVAPICH2를 사용하도록 컴파일할 때에도 에러가 발생하고, 에러 메시지는 대체로 다음과 같은 형태로 나타난다.

```
/usr/local/MPI/include/mpicxx.h:37:2: error: #error "SEEK_SET is
#defined but must not be for the C++ binding of MPI"
/usr/local/MPI/include/mpicxx.h:41:2: error: #error "SEEK_CUR is
#defined but must not be for the C++ binding of MPI"
/usr/local/MPI/include/mpicxx.h:46:2: error: #error "SEEK_END is
#defined but must not be for the C++ binding of MPI"
make[2]: *** [Parallel/CMakeFiles/vtkParallel.dir/vtkCompositer.
o] Error 1
make[1]: *** [Parallel/CMakeFiles/vtkParallel.dir/all] Error 2
make: *** [all] Error 2
```

[소스 3-27] VTK 컴파일 과정에서 발생하는 에러 메시지

에러가 발생하는 파일은 다음과 같다.

- VTK/Parallel/vtkCompositer.cxx:25:
- VTK/Parallel/vtkTreeCompositer.cxx:53:
- VTK/Examples/ParallelProcessing/Generic/Cxx/ParallelIso.cxx:46:
- VTK/Parallel/Testing/Cxx/GenericCommunicator.cxx:38:
- VTK/Parallel/Testing/Cxx/ParallelIso.cxx:46:

- VTK/Wrapping/Python/vtkPythonAppInit.cxx:21,
- VTK/Parallel/vtkMPI.h:19,
- VTK/Parallel/vtkMPICommunicator.cxx:24:
- VTK/Wrapping/Python/vtkParaPythonAppInit.cxx:16:

해결방법은 다음과 같다.

```
- Before modification -
#include <mpi.h>

- After modification -
#undef SEEK_CUR
#undef SEEK_END
#undef SEEK_SET
#include <mpi.h>
```

[소스 3-28] VTK 소스의 수정

하지만 헤더파일이 여러 개 붙어있고, 그 다음에 #include <mpi.h> 문장이 나올 경우, 단순히 undef만 추가하는 것만으로는 안 되며, 이 문장의 위치를 바꿔야 한다.

```
- Before modification -
#include "vtk ..."
#include "vtk ..."

#include <mpi.h>

- After modification -
#undef SEEK_CUR
#undef SEEK_END
#undef SEEK_SET
#include <mpi.h>

#include "vtk ..."
#include "vtk ..."
```

[소스 3-29] VTK 소스의 수정

라. GoogleEarth

Google Earth를 스크린으로 출력하려면 Chromium을 이용하면 된다. 한 가지 문제는 리눅스용 Google Earth가 32bit 버전만 존재하기 때문에 Picasso에서는 32bit용 Chromium을 별도로 준비해야 했다. 32-bit Chromium은 특별한 경우 외에는 거의 쓰일 일이 없을 것이라 생각했기 때문에 별도의 symbolic link를 만들지는 않았다.

```
lrwxrwxrwx  1 root   root      6 Jan 29 18:03 CR -> CR-1.9
drwxr-xr-x 30 root   root    4096 Feb  5 10:05 CR-1.9
drwxr-xr-x 30   2001 users   4096 Mar 28 11:54 CR-1.9-32bit
```

[소스 3-30] Picasso의 /usr/local 내 디렉토리 구조

Master 노드에서만 CRMOTHERSHIP을 정해주면 된다.

```
$ export CRMOTHERSHIP=master01
$ cd /usr/local/CR-1.9-32bit/mothership/configs/
$ python kisti-simple.conf /usr/local/bin/googleearth
```

[소스 3-31] Chromium으로 GoogleEarth를 실행하기 위한 CRMOTHERSHIP의 설정

위의 kisti-simple.conf의 내용은 다음과 같다. Tilesort SPU를 등록하기 전에 array SPU를 등록하는 것이 중요하다.

```
 생략

# List of nodes
TILES = [('display01',    0, 1036), ('display02', 2048, 1036),
        ('display03',    0,    0), ('display04', 2048,    0),
        ('display09', 3212, 1036), ('display10', 5260, 1036),
        ('display11', 3212,    0), ('display12', 5260,    0)]

# Creation of tilesort SPU
tileSortSPU = SPU('tilesort')
tileSortSPU.Conf('bucket_mode', 'Test All Tiles')
tileSortSPU.Conf('display_string', 'localhost:0.0')

# Client node : where application begins
clientNode = CRApplicationNode('master01')
clientNode.SPUDir(CRLIBDIR)
clientNode.SetApplication(sys.argv[1])
clientNode.AddSPU(SPU('array'))
clientNode.AddSPU(tileSortSPU)
clientNode.AutoStart([SHELL, \
                   "master01", \
                   "/bin/sh -c '" + ENV + " " + CRAPPFAKER + "'"])

for (hostname, x, y) in TILES:
    # Creation of a new node (display)
    node = CRNetworkNode(hostname)
    node.AddTile(x, y, TILE_WIDTH, TILE_HEIGHT)
    node.Conf('optimize_bucket', 0)

    # Create a render SPU & register it to the new node
    renderSPU = SPU('render')
    renderSPU.Conf('fullscreen', '1')
    renderSPU.Conf('display_string', 'localhost:0.0')
    renderSPU.Conf('borderless', '1')

#    if hostname == 'display01':
#        renderSPU.Conf('is_swap_master', '1')
#        renderSPU.Conf('num_swap_clients', '15')
    node.AddSPU(renderSPU)
    node.SPUDir('/usr/local/CR-1.9-32bit/lib/Linux')
    node.AutoStart([SHELL, \
                  hostname, \
                  "/bin/sh -c '" + ENV + " " + CRSERVER + "'"])

    # Add node to mothership configuration
    cr.AddNode(node)

 후략
```

[소스 3-32] Chromium으로 GoogleEarth를 실행하기 위한 mothership 설정 파일

5. 결론

지금까지 Picasso의 소프트웨어 설치 현황과 관리자의 관점에서 신규 소프트웨어의 설치 방법 그리고 주요 어플리케이션/라이브러리를 Picasso에 설치할 때 주의해야 할 사항들에 대해 설명했다.

Picasso는 계산 시스템이 아닌 만큼, 사용자가 증가하면서 다양한 종류의 소프트웨어에 대한 수요가 만들어질 것으로 예상한다. 따라서 관리자는 Picasso의 성능을 충분히 활용할 수 있도록 해당 소프트웨어를 수정하거나 설정을 최적화 할 필요가 있다.

소프트웨어의 신규 설치/업그레이드는 Picasso를 운영하는 기간 동안 지속적으로 이루어질 것이므로 본 보고서의 내용도 계속 보완될 것이다. 하지만 매번 그 변경 내용을 반영해서 문서로 배포하는 것은 그 한계가 있으므로 최신 내용을 http://sv.ksc.re.kr/AdministratorTip에서 확인할 수 있도록 했다.