ISBN 978-89-6211-279-5

가시화 시스템 운영정책

(Visualization system management policy)

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

표 차례

그림 차례

소스 차례

1. 개요

여기서는 KISTI가 운영하고 있는 visualization 시스템인 Picasso와 Mondrian의 제반 운영 정책을 설명한다. 이 보고서에서 설명하는 모든 내용은 시스템 관리자가 반드시 숙지하고 있어야 하며, 필요한 경우에는 사용자에게도 전달해서 시스템의 운영에 차질이 없도록 해야 한다.

Visualization 시스템의 운영 정책은 그때그때의 요구에 따라서 그 내용에 변경에 있을 수 있다. 하지만 매번 그 내용을 보고서 형태로 발간하는 것은 낭비이므로 웹(http://sv.ksc.re.kr/svwiki/SystemManagementPolicy)을 통해서 수시로 그 내용을 확인할 것을 권장한다.

2. 시스템 소개

Picasso는 외부 사용자에게 서비스를 제공하는 것을 전제로 구축한 시스템이지만, 슈퍼컴퓨터 4호기 2차분의 도입 전까지는 선별된 KISTI 내/외 연구자만을 대상으로 서비스를 제공한다. 이유는 슈퍼컴퓨터 4호기에 대한 Picasso의 역할이 확실히 정해지지 않았고, 이 문제는 2차분의 도입이 완료될 때까지 충분한 시간을 갖고 논의하기로 했다. 그리고 외부 사용자를 대상으로 본격적인 서비스를 제공할 경우, 이에 따르는 관리/운영 인력의 부재로 인해 전체적인 서비스의 quality가 떨어질 수 있다. 이는 시스템오/상주인력/슈퍼컴퓨팅사업팀 사이의 역할 분담이 아직 정확하게 나눠지지 않았기 때문이다.

장비별 세부사양

VisualizationComputer : Picasso, Mondrian, Chagall의 세부사양

InputOutputDevice : 출력장치 세부사양

3. 사용자 관리

가. 시스템 사용자

Picasso는 대외 서비스를 전제로 하는만큼 슈퍼컴퓨팅센터의 슈퍼컴퓨터 사용자 계정 관리정책을 따른다.

나. Picasso에서의 사용자 등록/삭제

1) NIS

Picasso에서는 사용자 계정을 NIS로 관리하고 있다. 슈퍼컴퓨터 4호기 2차분이 도입되면 LDAP을 이용해서 통합관리가 될 수도 있지만 지금은 계산 시스템과 visualization 시스템 사이의 연계가 명확하지 않기 때문에 독자적으로 사용자 계정을 관리한다. 우선, NIS 서버와 모든 NIS 클라이언트 노드의 /etc/sysconfig/network에 다음과 같이 domainname을 설정했다.

```
NISDOMAIN=visualization
```

[소스 3-1] /etc/sysconfig/network에서의 NISDOMAIN 설정

Picasso의 NIS 서버는 mgmt01이 담당한다. 따라서 모든 계정의 삭제/추가는 mgmt01에서 시작한다. 여느 NIS 서버와 마찬가지로 Picasso에서는 portmap, ypserv, yppasswdd가 실행되고 있어야 한다.

Picasso의 NIS 클라이언트는 mgmt01, master01~03, display01~16, render01~90 등이 해당되는데, 이 모든 노드에서도 역시 portmap과 ypbind가 실행되고 있어야 한다. 위의 노드에서는 모두 /etc/nsswitch.conf를 다음과 같이 수정해서 사용하고 있다. 다시 말해서 NIS로 관리되는 항목은 사용자 계정과 패스워드, 그리고 그룹 정보라는 뜻이다.

```
passwd:     files nis
shadow:     files nis
group:      files nis
```

[소스 3-2] /etc/nsswitch.conf의 수정

2) 사용자 분류

국내외 상관없이 소속 기관(최상위 조직)별로 사용자 group을 만들어서 mgmt01의 /etc/group에 등록한다. 사용자의 default group은 users지만, 소속기관에 따라서 해당하는 적절한 그룹에 동시에 등록한다.

- KISTI 슈퍼컴퓨팅센터의 A 사용자를 등록할 경우

```
$ cat /etc/group

...

kisti:x:gid:A  <- 부서가 아닌, 연구소를 등록한다.
```

[소스 3-3] /etc/group의 등록방법 1

- X 대학교 Y 연구실의 Z 사용자를 등록할 경우

```
$ cat /etc/group

...

X:x:gid:Z <- 학교를 등록한다.
```

[소스 3-4] /etc/group의 등록방법 2

3) 사용자 등록/삭제

시스템 관리자는 사용자 등록/삭제는 반드시 mgmt01에서 수행하고, NIS가 운용되고 있다는 사실을 잊어서는 안 된다.

가) 사용자 추가 절차

1. 사용자 ID 추가 : useradd / uid는 useradd가 자동으로 정해주는 숫자를 그대로 사용하면 된다
2. 패스워드 변경 : passwd / chfn (chfn으로 full name과 Office(소속기관)을 반드시 적어줘야 한다)
3. /etc/group 수정
   - 새로 추가하는 모든 계정은 users 그룹에 반드시 포함돼야 하고, /etc/passwd의 default group을 users로 설정한다.
   - 새로운 사용자의 소속기관이 기존 /etc/group에 존재하지 않으면 소속기관을 추가한다 (man groupadd)
   - 새로운 계정이 소속 기관 그룹에도 속해야 한다.
4. NIS DB 업데이트 : /usr/lib64/yp/ypinit -m
   - DB를 업데이트했다고 해서 NIS 서버에서 ypserv, NIS 클라이언트에서 ypbind를 재시작할 필요는 없다.
5. 다른 노드에 들어가서 제대로 입력이 됐는지 확인한다(finger)

나) 사용자 삭제 절차

1. 사용자 ID 추가 : userdel
2. /etc/group 수정 : users 그룹과 소속기관 그룹을 모두 확인
3. NIS DB 업데이트 : /usr/lib64/yp/ypinit -m
   - DB를 업데이트했다고 해서 NIS 서버에서 ypserv, NIS 클라이언트에서 ypbind를 재시작할 필요는 없다.
4. 다른 노드에 들어가서 제대로 삭제됐는지 확인(man finger)

4) Password

사용자의 패스워드는 http://www.goodpassword.com에서 만들어진 것을 사용한다.

- 화면의 왼쪽에 'Random Password' 섹션을 이용하고, Size=8로 지정한다.
- 0-9, a-z, A-Z, 특수문자들 모두 사용하도록 체크한 후에 'Generate Password'
- 그 결과를 해당 사용자의 패스워드로 지정하고, 나중에 사용자에게 e-mail로 통보한다.

4. Job scheduling

가. Picasso에서의 Torque 스케줄러 설치방법

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 --with-default-server=scheduler-ib
```

[소스 4-1] Torque 스케줄러의 configure 스크립트 실행

Configure의 실행이 끝난 후 RPM을 바로 만든다.

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

[소스 4-2] 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 4-3] 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 4-1] 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 4-4] /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 4-5] 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 4-6] pbs_server와 pbs_sched의 서비스 등록

render 노드에서는 pbs_mom이 추가돼야 한다.

```
$ chkconfig pbs_mom on
```

[소스 4-7] pbs_mom의 서비스 등록

3) 다른 스케줄러와의 연계

지금까지 파악한 바로는 rendering farm 전용 스케줄러들은 Torque나 LoadLeveler와 같은 계산용 스케줄러와는 다른 형태로 작동하고, 이들 스케줄러와의 연계도 잘 되지 않는 것 같다. 게다가 rendering farm의 rendering software의 경우, 자체적으로 multi-thread를 이용해서 렌더링 속도를 올리려고 시도하는 경우가 있는데, Torque는 이 기능을 전혀 탐지할 수 없다는 문제점도 있다. 따라서 Picasso를 rendering farm으로 사용하면 Torque의 운영을 아예 중단시키거나 rendering farm의 rendering software가 사용하는 CPU의 수와 Torque가 사용하는 CPU의 수를 강제로 배분시키는 방법밖에 없으니 이 점을 주의해야 한다.

나. Picasso의 Torque 스케줄러 운영정책

Visualization은 전산(컴퓨터 그래픽스)관련 전공자도 많이 다루는 만큼 계산과학자들의 천편일률적인 MPI+FORTRAN(또는 C++)가 아

닌, 다양한 프로그래밍 언어와 테크닉을 볼 수 있다는 차이가 있다. 여기에는 multi-thread, MPI가 아닌 메시지(데이터) 전달(IceT) 등도 포함된다.

Picasso는 실시간 데이터 분석(real-time rendering with user interaction), 일반적인 batch job, rendering farm 등 서로 다른 실행 특성을 갖는 어플리케이션을 모두 지원해야 하기 때문에 이를 반영한 최적의 스케줄링 정책을 정하는 것은 어렵지만, 어떻게든 서비스 가능한 수준으로 정해보고자 했다.

1) 작업 구분

Picasso에서 실행하는 모든 작업(job)은 다음과 같이 구분한다.

- **Interactive job** : 주로 master01과 display01~display16을 이용하는 real-time rendering, video streaming 등의 작업이 여기에 해당한다.
- **GPU batch job** : User interaction이 없고 주 계산을 GPU로 수행하는 프로그램들은 모두 이 범주에 들어간다. 특성상 CPU는 GPU 계산을 위한 전처리/후처리 역할만을 주로 수행하기 때문에 많은 CPU를 사용하지 않는 경향이 있다.
- **CPU batch job** : User interaction이 없고 주 계산을 CPU로 수행하는 프로그램들은 모두 이 범주에 들어간다. 일반적으로 우리가 생각하는 MPI 작업이 여기에 해당된다.
- **Rendering farm job** : Picasso를 rendering farm으로 사용할 때에만 나타나는 작업으로, Torque로 통제가 불가능한 형태의 작업이다. (아래 참고)

2) 스케줄러 정책 일반

모든 렌더링 노드는 8개의 CPU(코어)를 갖고 있다. 이 중 1개는 OS 및 백업용으로 고정하고, 나머지 7개를 각 형태의 어플리케이션에 분배하는 것을 원칙으로 한다. 특별한 경우(Picasso의 모든 컴퓨팅/그래

픽 능력을 필요로 하는)가 아니면 master01~master03과 display01~display16은 Torque 스케줄러의 관리 대상에 포함하지 않는다. 그 대신 master01~master03에서는 job submission이 가능하도록 설정한다. 그리고 테스트/OS 재설치/노드 고장 등 어떤 이유에서건 일반 서비스를 제공할 수 없는 노드는 Torque 스케줄러 관리 대상에서 제외시킨다. 마지막으로 멀티쓰레드 작업은 Torque가 전혀 관리를 못하므로 기본적으로 모든 어플리케이션은 single-thread로 작성한다.

3) 작업 사이의 CPU 배분

- Picasso를 rendering farm으로 사용하지 않을 경우, 단일 rendering 노드에서 interactive job, GPU batch job, CPU batch job 사이의 CPU 분배를 1:2:4로 한다. (나머지 한 개는 OS/여분으로 생각한다)
- Picasso를 rendering farm으로 사용할 경우에는 rendering farm을 위한 전용 스케줄러가 별도로 가동되는데, 문제는 이 스케줄러가 Torque와는 전혀 연계가 안 된다는 점이다. 그렇기 때문에 Picasso를 rendering farm으로 사용할 때 다른 작업을 위한 CPU 여분을 두기는 상당히 까다로운 편이다. 하지만 최소한의 지원을 위해서 각 노드별로 rendering farm에는 6개의 CPU, GPU job과 interactive job에 각각 하나씩의 CPU를 할당한다.

5. 네트워크 설정

가. 일반사항

Picasso는 GLORIAD와 슈퍼컴퓨터 4호기 통합 네트워크에 동시에 연결되어 있다. 따라서 방화벽 운영정책을 신중하게 세워야 한다.

System	Picasso	Mondrian / Chagall
4호기 통합 네트워크	연결	차단
GLORIAD	연결	연결
SSH 허용 여부	제한적 허용	제한적 허용
TELNET/FTP	차단	차단
VNC	전용 노드를 통해 허용	차단
개별 어플리케이션	제한적 허용	제한적 허용
KISTI 수행 프로젝트 지원	제한적 허용	제한적 허용

[표 5-1] Visualization 시스템의 방화벽 정책 일반

나. 방화벽

Picasso는 슈퍼컴퓨터 4호기(통합 네트워크)를 위한 방화벽뿐만 아니라 Picasso의 각 노드에 대해 netfilter를 별도로 설정해서 2중의 방화벽을 운영한다. 각 노드별로 방화벽을 설정하는 방법은 대외로 유출하지 않는 것이 바람직하므로 여기에 공개하지는 않겠다.

Mondrian/Chagall은 GLORIAD에만 연결돼있지만 GLORIAD 차원에서 별도의 방화벽을 운영하지 않기 때문에 Mondrian의 개별 노드에 대해 netfilter만 설정해서 운영하고 있다.

6. NTP

가. 현재 설정

1) Server

Picasso의 NTP 서버는 mgmt01이 담당한다. 현재 슈퍼컴퓨터 4호기와의 연계정책이 정해지지 않았으므로 일단 Picasso 내에서 노드 사이의 시간을 맞추는 목적으로만 사용한다.

2) Client

mgmt01 이외의 모든 노드는 mgmt01에 대한 NTP 클라이언트로 작동한다. 비록 클라이언트라고는 해도 각 노드가 ntpd를 띄울 필요는 없다. 그 대신 /etc/cron.hourly에 다음과 같은 스크립트를 넣어서 매 시간마다 mgmt01과 시간을 맞추도록 하면 충분하다.

```
$ cat /etc/cron.hourly/time-sync-client.ntp
#!/bin/bash
/usr/sbin/ntpdate -s -u 172.16.3.1
```

[소스 6-1] NTP 클라이언트 노드에서의 cron 설정

3) 네트워크

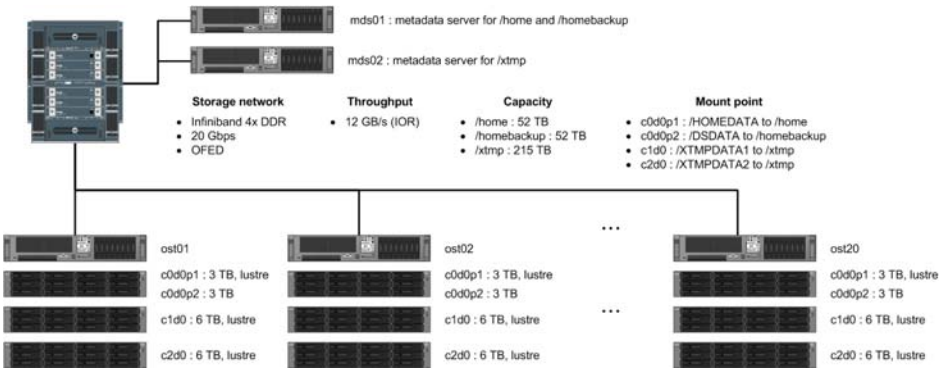
NTP로 시간을 맞추기 위해 사용하는 네트워크는 관리 네트워크로 충분하다(Gigabit ethernet, 172.16.0.0)

나. 향후 계획

슈퍼컴퓨터 4호기와의 연계방침이 결정되면 mgmt01은 외부 타임 서버에 대한 클라이언트로 작동하면서, 동시에 Picasso 내부 노드에 대한 타임 서버의 역할을 하도록 수정할 계획이다.

7. 스토리지

가. 스토리지 네트워크의 구성



[그림 7-1] 스토리지 네트워크의 구성

나. 주요 디렉토리 정보 및 운영지침

1) /home

/home에는 사용자의 기본 데이터를 저장한다(실험에 의해서 임시로 만들어지는 데이터는 모두 /xtmp에 저장한다). /home을 마운트(mount)할 수 있는 호스트 그룹에는 어플리케이션 서버, 컴퓨팅/렌더링, 관리 서버 그룹이 있다. 그 외의 노드(예: 게이트웨이)는 굳이 /home을 마운트할 필요가 없다.

2) /homebackup

/homebackup은 /home과 같은 데이터를 갖고 있어서 /home에 문제가 발생할 경우, 이를 대신하기 위해 운영하고 있다. /homebackup은 /home과 동일한 용량을 갖고 있지만 OasisFS라는, Lustre와는 다른 파일시스템이 설치돼있다. 따라서 커널 패치까지 필요로 하는, 그렇기

때문에 조금만 문제가 생겨도 전체 파일시스템을 사용할 수 없을 수도 있는 Lustre에 문제가 발생한다고 해도 OasisFS는 전혀 영향을 받지 않고 운영이 가능하다.

하지만 일반적인 상황에서라면 /homebackup을 모든 노드가 볼 필요는 없다. 따라서 평상시에는 관리서버 그룹만 /homebackup을 볼 수 있도록 설정해놓는다.

3) /xtmp

/xtmp는 실험을 위해 대량의 데이터를 일시적으로 저장하기 위해 사용한다. 이 디렉토리는 가용용량이 215TB로, /home의 52TB의 4배에 이른다. 하지만 /homebackup과 같이 백업을 위한 별도의 파티션은 운영하고 있지 않기 때문에 데이터 손실 문제 등은 사용자가 직접 관리하도록 한다.

다. 백업

/home은 같은 용량을 갖는 /homebackup에 rsync를 이용해서 미러링(mirroring)을 하기 때문에 최대 하루 정도의 시간차 내에서 전체 데이터에 대한 백업이 진행되고 있다.

8. Software 관리

가. 소프트웨어 관리 정책 일반

일반 사용자들에게 잘 알려진 프로그램은 많은 경우 RPM으로 찾을 수 있기 때문에 가능하면 RPM을 받아와서 설치하는 것을 원칙으로 한다. 어지간한 RPM은 http://rpm.pbone.net에서 찾을 수 있다. 하지만 RPM이 존재하지 않을 경우, src.rpm을 가져와서 rpmbuild로 RPM package로 만든 후에 설치하는 것을 차선책으로 한다. 마지막으로 RPM을 찾을 수 없어서 tarball을 가져와서 설치한다고 해도, 자체적으로 RPM을 만들어주는 기능을 제공할 수도 있다(예: Torque) 이때에도 역시 RPM으로 만들어서 설치한다.

하지만 NVIDIA 비디오 카드 드라이버, 10G 이더넷 드라이버, OFED 등 RPM으로는 아예 설치가 불가능한 것은 예외로 처리한다.

나. 소프트웨어 설치 디렉토리

리눅스 배포판에 포함돼있는 소프트웨어는 모두 기본 디렉토리에 설치되지만 그렇지 않은 소프트웨어는 특수한 경우를 제외하고는 거의 대부분 /usr/local에 설치한다.

/usr/local의 모든 소프트웨어는 실제로는 '/usr/local/소프트웨어-버전' 형태의 디렉토리에 설치하지만, 이 디렉토리를 가리키는 '/usr/local/소프트웨어'라는 심볼릭 링크(symbolic link)를 같이 만들어준다. 따라서 사용자는 '/usr/local/소프트웨어' 디렉토리에 소프트웨가 존재하는 것으로 가정하고 환경을 설정하면 나중에 해당 소프트웨어의 업그레이드가 진행된다고 해도 사용자는 별도의 환경 설정을 바꾸지 않고도 최신 버전을 사용할 수 있게 된다.

```
lrwxrwxrwx  1 root root    13 Apr  7 13:17 PortAudio -> PortAudio-
v19/
drwxr-xr-x  4 root root  4096 Apr  7 13:16 PortAudio-v19/
lrwxrwxrwx  1 root root    10 Feb  4 21:09 POVRay -> POVRay-3.6/
drwxr-xr-x  6 root root  4096 Feb  4 21:07 POVRay-3.6/
lrwxrwxrwx  1 root root     8 Feb  3 20:04 Qt -> Qt-4.2.3/
drwxr-xr-x 12 root root  4096 Feb  3 20:01 Qt-4.2.3/
drwxr-xr-x 12 root root  4096 Feb  3 18:22 Qt-4.3.3/
lrwxrwxrwx  1 root root    10 Feb  4 23:11 QUANTA -> QUANTA-1.0/
drwxr-xr-x  8 root root  4096 Feb  4 23:19 QUANTA-1.0/
lrwxrwxrwx  1 root root     8 Apr  7 13:40 SAGE -> SAGE-3.0/
```

[소스 8-1] 소프트웨어 설치 디렉토리(/usr/local)의 내용

다. YUM repository (Picasso only)

Picasso는 /xtmp/Software/LINUX 밑에 local yum respository를 운영하고, Picasso의 모든 노드는 /etc/yum.repos.d/에 OS에 대응하는 설정파일이 있다.

9. Visualization 시스템의 정기점검 지침

가. 정기점검일

- 3월, 6월, 9월, 12월 마지막 금요일
- 관리자는 정기 점검일 2주일 전과 1주일 전 2회에 걸쳐서 사용자 메일링 리스트를 통해 공고한다.

나. 정기점검일의 제약

정기 점검을 진행할 때 외부로부터의 시스템 접속, 내부 사용자의 사용, 투어 등이 일체 불가능하다.

다. 정기점검 내용

- 하드웨어 이상 유무 확인
- HDD
- RAID 컨트롤러
- 10G NIC
- Infiniband HCA
- 메인 메모리
- CPU
- 각 노드의 디렉토리 checksum 확인 : 해킹여부를 확인하기 위해 디렉토리 checksum을 새로 계산하고 이에 대한 확인