NPACI Rocks 4.0
클러스터 설치가이드

슈퍼컴퓨팅 연구실

NPACI Rocks 4.0
클러스터 설치가이드

목차

머리말

1. 사용자 추천

나는 NPACI Rocks 배포판의 품질에 매우 깊은 인상을 받았다. NPACI Rocks의 현 버전 덕분에 나의 Beowulf 클러스터를 셋업하는 데 들이는 노력과 시간을 상당히 줄일 수 있었다.
-- Martin Beaudoin, M.SC.A, ing. IREQ, Quebec, Canada

NPACI Rocks 클러스터 배포판은 우리가 곧바로 사용할 수 있는 완성형의 솔루션이다. 이것은 우리가 클러스터링을 시작할 때 굉장히 많은 시간을 줄일 수 있도록 도와주었으며, 과학자들과 학생들의 매우 어려운 작업을 수행하는 데 있어 안정성과 편리함을 증명해 왔다.
--Roy Dragseth, M.Sc., The Computer Center, Univ. of Tromso, Norway

Rocks는 우리 회사의 리눅스 고성능 클러스터를 수요자에게 제공하는 데 있어 그 근간이 되어 왔다. 쉬운 설치과정과 필요에 따라 특정한 환경을 설정할 수 있다는 점은 다양한 고객의 독특한 요구사항을 해결하는 데 큰 도움이 되었다. Rocks의 사용은 고객에게 견고하고 확장성 있는 클러스터 솔루션을 제공할 능력을 의미하며, Rocks에 의해 제공되는 쉬운 관리기능은 고객들이 그들의 클러스터를 관리하는데 시간을 보내는 대신에 과학 계산에 집중할 수 있도록 해준다.
--Laurence Liew, Scalable Systems Pte Ltd, Singapore

나는 NPACI Rocks 클러스터 배포판의 성능에 매우 만족한다. 나는 지난 5년간 많은 클러스터를 설치해 왔으며, 이를 위해 Scyld, OSCAR, 그리고 다양한 사용자 툴을 사용해 왔다. 경험상 Rocks 클러스터는 안정성과 관리성, 그리고 설치의 용이성 면에 최적화된 솔루션을 제공한다.
--Tim Carlson, PhD, PNNL, Richland, WA

우리의 301개의 노드의 고성능 컴퓨팅 클러스터의 관리 책임을 맡고 있는 유일한 시스템 관리자로서, NPACI Rocks와 같은 배포판은 우리에게 우리 스스로가 안정적인 컴퓨팅 인프라를 만들 수 있다는 확신을 준다. 또한 NPACI Rocks 팀이 나의 질문들에 대해 언제나 적절한 대답과 지원을 한다는 해준다는 사실은 마치 내가 관리자와 시스템 엔지니어의 그룹을 별도로 갖고 있는 것과 같은 착각을 준다. ------Steve Jones, Iceberg Cluster, Bio-X at Stanford University, Palo Alto, CA

2년 전에 우리의 기본 시스템으로 Rocks를 설치하기 전에 우리는 다양한 클러스터링 대안을 시도해왔었다. Rocks는 쉬운 설치, 설정, 확장, 그리고 사용법을 제공한다. Rocks는 매우 견고하며, 노스웨스턴 대학의 이론화학그룹의 핵심 계산 환경으로 자리를 잡았다. 또한 Rocks는 처음 한번 사용하는 것을 지켜보는 것만으로도, 사용자들이 사용하고 싶은 충동을 느끼게 하는 전염성이 있다.
--Frederick P. Arnold, Jr, NUIT, Northwestern University, Evanston IL

2. 소개

하드웨어 구성요소와 연산능력의 관점에서 볼 때, 상용클러스터는 가격대비 성능 면에서 컴퓨팅 엔진의 새로운 흐름이 되었다. 확장성 있는 클러스터의 관리 전략이 없다면, 클러스터의 저렴한 경제성도 장비를 관리하는 데 추가적인 인력비용이 요구된다는 점에서 반감될 수 있다. 클러스터 관리의 복잡성(예를 들면, 모든 노드들이 동일한 소프트웨어로 구성되어 있는가)은 평상시에 과학 어플리케이션을 연구하면서 파트타임으로 클러스터를 관리하는 과학자들에게 큰 부담이 될 수 있다. 만일 이러한 복잡한 상황이 발생하면, 장비의 상태는 극단의 상태에 빠질 수 있다: 설정문제 때문에 안정적이지 않을 수 있으며, 소프트웨어의 버전이 적절히 업데이트되지 않을 수도 있으며, 보안구멍이 발생하거나, 소프트웨어의 버그 패치가 제대로 되어 있지 않을 수도 있다.

이전의 클러스터링 toolkit 은 각 노드들의 설정을 비교하는 데 많은 노력을 해야 했지만, rocks 는 각 노드에 대한 완벽한 운영체계의 설치과정을 만들었다. 이 과정의 완벽한 자동화를 위한 노력 덕분에 설치 후 운영 시 전체노드 중 몇 개의 노드가 비정상적으로 동작하는지 파악하는데 걸리는 시간보다 모든 노드를 재설치 하는데 걸리는 시간이 더 짧아졌다.

일반사용자의 데스크탑과 달리, 클러스터의 노드는 급격하게 변하거나 update 될 수 있는 "soft state"로 고려된다. 이것은 일반적일 클러스터 환경에서 설정 관리 도구(Cfengine)의 철학인 “설치된 OS 에 대한 철저한 검사와 parity 체크”보다도 더 중요한 요소로 고려될 수 있다.

처음에는 설정변수의 수정이 필요할 때 OS 를 재 설치하는 것은 잘못된 방법처럼 보였다. 특히 하나의 노드 때문에 모든 노드에 대해 재 설치하는 것은 가혹한 방법으로 생각되었다. 하지만 이 접근 방법은 굉장히 확장성이 좋고, 또 적정크기의 클러스터에 대해서는 선호할 만한 방법이 될 수 있다. 즉 OS 가 짧은 시간 내에 scratch 로부터 설치되기 때문에, 다른(그리고 아마도 동시 사용이 불가능한) application 에 의존적인 설정이 쉽게 각 노드에 이루어질 수 있다(예: LAM/MPI<=> MPICH). 부가적으로 이러한 구조에서는 어떠한 종류의 업그레이드도 실행중인 job 을 간섭하지 않을 것이다.

rocks 의 가장 중요한 요소 중 하나는 특정노드에 사용하기 위한 (전체 소프트웨어 set 이 들어 있는) customized 된 배포판을 만들 수 있는 견고한 mechanism 을 제공한다는 것이다. 하나의 클러스터는 계산노드, frontend 노드, 파일 서비스 노드, 모니터링 노드 등 여러 종류의 노드를 필요로 한다. 이런 노드들은 특정한 소프트웨어 패키지를 필요로 한다.

하나의 배포판 내에서 이러한 노드타입은 Rocks Kickstart Graph 에서 만들어지는 노드타입 개개의 Red Hat Kickstart file 을 통해 정의 된다.

하나의 Kickstart 파일은 하나의 노드에 설치할 모든 소프트웨어 패키지와 그에 대한 설정을 정의하는 텍스트 파일이다. Rocks Kickstart Graph 는 RedHat Kickstart file 을 정의하기 위해 XML-based tree structure 를 사용한다. 이 Graph 를 사용하여 Rocks 는 공통적인 요소를 재 복사하지 않고 효율적으로 각 노드타입을 정의할 수 있다.

포유류가 가진 유전자중 80 %가 동일한 것처럼, Rocks 의 다른 종류의 타입의 노드도 소프트웨어 set 의 대부분은 공통적으로 같다. Rocks Kickstart Graph 는 이러한 공통점에 대한 명시를 하지 않고도 쉽게 각 노드타입의 차이를 쉽게 정의한다. (이러한 구조에 대한 디자인을 좀더 깊이 있게 설명하는 논문을 보길 원한다면, 이 매뉴얼의 끝부분에 나오는 참고문헌을 참조해라)

이런 설치과정을 사용함으로써, 우리는 많은 하드웨어의 차이를 추상화 할 수 있었고, Kickstart 프로세스가 load 해야 할 적절한 하드웨어 모듈을 자동으로 인식하도록 할 수 있었다. (예를 들면 디스크 서브시스템타입: SCSI, IDE; 내장 RAID adapter; Ethernet interfaces; 그리고 고속 network interfaces). 더 나아가, 우리는 상용리눅스 배포판이 제공하는 견고하고 풍부한 지원 혜택을 이용할 수 있었다.

2.1. Rocks 와 SQL

가능하면, Rocks 는 (각 서비스의) 설정간의 차이들을 규정하기 위해 자동화된 방법을 사용했다. 즉, 클러스터는 통합된 하나의 머신이므로, 머신 전체에 대한 정보를 필요로 하는 서비스는 많지 않다. 예를 들면, hosts 파일과 Queueing 시스템 정도가 모든 계산노드의 리스트를 필요로 한다. Rocks 는 SQL database 를 사용하여 이에 대한 전체적인 설정을 저장한다. 그리고 나서 특정서비스에 필요한 설정파일(예로, DHCP 설정파일, /etc/hosts, 그리고 PBS node 파일)을 만들기 위해 database 를 생성시킨다.

2.2. 목표

2000 년 5 월 이후, Rocks group 은 관리 가능한 클러스터의 설치가 얼마나 어려운가에 대해 언급해왔다. Rocks 는 “클러스터를 쉽게”라는 하나의 목표를 가지고 움직여 왔다 여기서 “쉽게” 라는 말의 의미는 클러스터의 설치, 관리, 업그레이드를 쉽게 하자는

의미이다. Rocks 는 다양한 분야의 과학관련 사용자가 클러스터의 계산능력을 이용할 수 있도록 하는 것을 돕기 위해 이러한 목표를 추구해 왔다. 다양한 과학자들이 이용할 수 있는 안정적이고 관리 가능한 병렬처리 플랫폼을 만드는 것이 병렬처리의 기술수준을 높이는 데 크게 공헌할 것이다.

3. Contact

토의내용을 follow-up 하거나 공지 사항을 알기 위해서 혹은 Rocks 관련 개발자가 되기 위해서 여러분의 다음의 메일링 리스트를 이용할 수 있다.

3.1. 메일링 리스트

npaci-rocks-discussion

이곳은 사용자가 클러스터에 관련된 개선사항, 추가사항, 그리고 관련 기술에 대해 사용자들이 토의 할 수 있는 메일링 리스트이다. 누구나 참여 할 수 있다.

npaci-rocks-devel

이것은 Rocks 관련 소프트웨어를 개발하는 사람들이 자세한 내용을 토의할 수 있는 개발자의 메일링리스트이다. 누구나 참여할 수 있다. 그러나 추가적인 내용들이 패스워드에 의해 보호받을 수 있다.

3.2. Email

distdev

질문이 있다면 SDSC 의 클러스터 개발자 그룹에 직접연락을 해도 좋다.(주석 3 참조)

4. 협력 관계

4.1. 연구 그룹

San Diego Supercomputer Center, UCSD

- Philip Papadopoulos
- Mason Katz
- Greg Bruno
- Nadya Williams
- Federico Sacerdoti (past member)

Scalable Systems Pte Ltd in Singapore

- Laurence Liew
- Najib Ninaba
- Eugene Tay
- Sivaram Shunmugam
- Tsai Li Ming

High Performance Computing Group, University of Tromso

- Roy Dragseth
- Svenn Hanssen
- Tor Johansen

The Open Scalable Cluster Environment, Kasetsart University, Thailand

OpenSCE
Open Scalable Cluster Environment

- Putchong Uthayopas
- Thadpong Pongthawornkamol
- Somsak Sriprayoonsakul
- Sugree Phatanapherom

Korea Institute of Science and Technology Information (KISTI)

KROCKS 슈퍼컴퓨팅센터

- Jysoo Lee
- Yuchan Park
- Jeongwoo Hong
- Hong Taeyoung
- Sungho Kim

Sun Microsystems

Sun microsystems

Sun Microsystems has supported Rocks through their gracious hardware donations, most notibly the 129-node cluster that was built in two hours on the vendor floor at SC 2003.

Advanced Micro Devices

AMD

AMD has loaned us several Opteron and Athlon boxes to make sure Rocks always supports their latest chip architectures. In addition, AMD co-sponsored Rocks-A-Palooza I, the first Rocks All Hands Meeting.

Dell

Dell has loaned us several x86, x86_64 and ia64 boxes to make sure Rocks always supports their server hardware. They have also provided extremely valuable bug reports, and feature requests. We thank Dell for helping make Rocks stronger.

SilverStorm Technologies

SilverStorm Technologies (formerly Infinicon Systems) donated 64 nodes worth of Infiniband gear in order to provide an appropriate development platform for the Rocks team to produce the first version of the IB Roll for SilverStorm fabrics.

Compaq Computer Corporation (Now HP)

Compaq has donated several x86 and ia64 servers to the Rocks group for development, and production clustering. We gratefully acknowledge the support of Compaq Computer Corporation, especially Sally Patchen, the California Educational Accounts Manager.

4.1. 개인

이 목록은 완전하지 않다. Rocks system 에 패치를 가한 사람들을 모두 포함하려고 한다. 만약 이름이 빠져있다면 우리에게 연락을 주기 바란다. 이름은 알파벳 순서로 나열하였다.

- Fred Arnold, NUIT, Northwestern University, Evanston IL
- Justin Boggs, Advanced Micro Devices, Sunnyvale CA
- Tim Carlson, PNNL, Richland WA
- Sandeep Chandra, GEON Group, SDSC, San Diego CA
- Steve Jones, Bio-X at Stanford University, Palo Alto, CA

- Robert Konecny, The Center for Theoretical and Biological Physics, UCSD, San Diego CA
- Matt Massie, Millennium Project, UC Berkeley, CA
- Doug Nordwall, PNNL, Richland WA
- Glen Otero, Callident, San Diego CA
- Vladmir Veytser, NEES Project, SDSC, San Diego CA
- Matt Wise, Advanced Micro Devices, Sunnyvale CA

1 장. Rocks 클러스터 설치하기

1.1. 시작하기

클러스터를 구축하고 관리하는데 Rocks 를 사용함으로써 얻는 이점은 명확하다. 클러스터의 구축과정은 상당히 단순한 작업이지만, 그것의 소프트웨어에 대한 관리는 상당히 복잡해질 수 있다. 이 복잡성은 클러스터의 설치 및 확장 시 가장 관리하기 어려운 부분이 되곤 한다. Rocks 는 이러한 클러스터의 설치 및 확장 과정에서의 복잡성을 제어하기 위한 방법과 성능 모니터링 도구를 제공한다. 이 장에서는 클러스터를 구축하고, 그것의 소프트웨어를 설치하는 과정을 설명한다.

1.1.1. 지원 하드웨어

Rocks 는 레드햇 리눅스 배포판을 기반으로 만들었기 때문에, 레드햇이 지원하는 모든 하드웨어를 지원한다. 단 x86, x86_64 및 IA-64 architecture 프로세서만을 지원한다.

Processors
 x86 (ia32, AMD Athlon 등.)
 IA-64 (Itanium, McKinley 등.)
 X86_64(AMD opteron 과 EM64T)
Networks
 이더넷 (인텔 Gigabit 이더넷 포함한 레드햇이 지원하는 모든 종류,)
 미리넷 (Lanai 9.x)

1.1.2 최소 하드웨어 요구 사항

Frontend 노드

- 디스크 용량: 16GB
- 메모리 용량: 512MB
- 이더넷: 2 개의 물리적인 포트(즉, eth0 와 eth1)

계산 노드

- 디스크 공간: 16GB
- 메모리 용량: 512MB
- 이더넷: 1 개의 물리적인 포트(즉, eth0)

1.1.3. 물리적 조립

관리해야 할 첫 번째는 클러스터의 물리적인 배치(deployment)이다. 클러스터를 물리적으로 어떻게 구축(construct)할 것인가에 대한 많은 연구가 있어 왔다. O'Reilly 사의 *Building Linux Clusters* 책은 클러스터의 물리적인 구축(setup) 및 마더보드 등의 장비 선정 방법 등에 대부분의 내용을 할애하고 있다. 또한 *How to Build a Beowulf* 책도 물리적인 구축에 대한 좋은 팁을 제공한다. Rocks 는 상대적으로 높은 신뢰성과 공간 효율성(density)을 제공하는 rack 장착 설비(이것은 좀 더 비싸다)를 선호한다. 하지만 mini-towers 형태로 만들어진 Rocks 클러스터도 존재한다. 무엇이 자신의 시스템에 적절한지 선택해야 한다. Rocks 클러스터의 물리적인 구축 시 다음의 노드타입 중 하나 이상을 포함해야 한다.

Frontend

이 타입의 노드들은 외부세계에 노출된다. 많은 서비스(NFS, NIS, DHCP, NTP, MySQL, HTTP 등)가 이 노드에서 실행된다. 일반적으로 이것은 유능한 시스템 관리자를 필요로 한다. Frontend 노드는 사용자가 로그인하여, Job 을 제출하고, 코드를 컴파일하는 등의 일을 하는 곳이다. 이 노드는 network address translation (NAT)을 사용하여 클러스터의 라우터 역할을 수행 할 수도 있다. 이 노드는 일반적으로 다음의 특징을 갖는다.

- 두 개의 이더넷 인터페이스 - 하나는 public 용, 하나의 private 용
- 파일을 저장하기 위한 충분한 디스크

Compute

이것은 일꾼(workhorse) 노드이다. 이 노드들은 임의성이 강하다(disposable). 즉 Rocks 의 관리 체계는 짧은 시간(대략 10 여분)내에 모든 노드에 OS 를 재 설치하는 방법을 제공한다. 이 노드들은 외부의 인터넷에는 보이지 않는다. 계산노드는 다음의 특징을 갖는다.

- 전원 케이블
- 관리 네트웍용 이더넷
- 기본 운영체계(OS 및 라이브러리)를 caching 하기 위한 디스크 드라이브
- 추가적인 고성능 네트웍(예를 들면 미리넷)

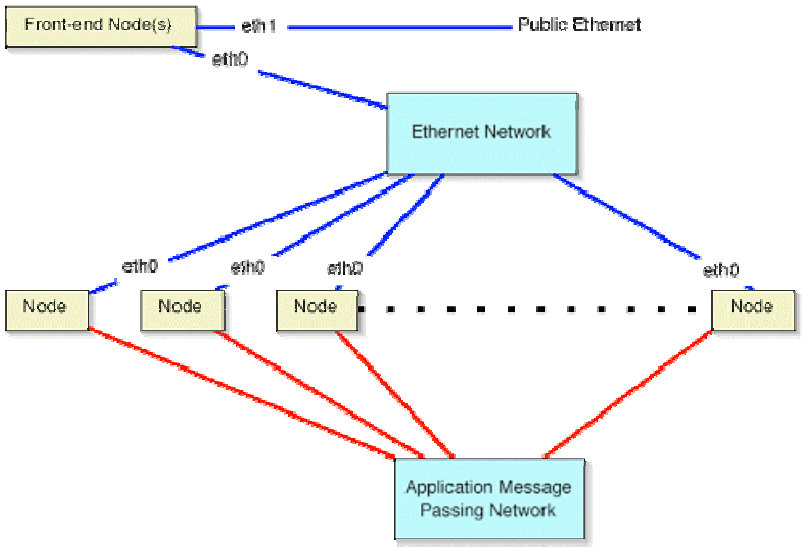
Ethernet Network

모든 계산노드는 이더넷으로 private 네트웍에 연결된다. 이 네트웍은 관리, 모니터링, 및 기본적인 파일 공유용으로 사용된다.

Application Message Passing Network

모든 노드는 Gigabit 레벨의 네트웍으로 연결될 수 있으며, 이 경우 스위치가 필요하다. 이 네트웍은 병렬프로그램을 위한 고성능 message passing 을 가능케 하는 low-latency, high-bandwidth 네트웍이다.

Rocks 클러스터 architecture 에서는 노드들이 다음과 같이 연결되어야 한다.



계산노드에서, 이더넷 인터페이스는 eth0 로 매핑 되어야 하며, 클러스터의 이더넷 스위치에 연결되어야 한다. 이 네트웍은 private 으로 간주한다. 즉 이 네트웍의 모든 네트웍 트래픽은 외부의 public 네트웍(예로 인터넷)과 물리적으로 분리되어야 한다.

Frontend 에서는 두 개의 이더넷 인터페이스가 필요하다. eth0 에 매핑된 인터페이스는 계산노드의 네트웍에 연결되어야 한다. eth1 에 매핑된 인터페이스는 외부의 네트웍(즉, 인터넷 혹은 기관내의 인트라넷)에 연결되어야 한다.

일단 물리적으로 클러스터의 조립이 완료되면, 각 노드는 키보드 없이 부팅되도록 설정되어야 한다. 이 과정에서 BIOS 값을 변경해야 하며, 불행히도 마더보드마다 이 값이 틀리다. 또 키보드 없이는 부팅이 불가능한 몇 가지 마더보드들이 있다는 것을 유의해야 한다.

1.2. Frontend 설치 및 설정

이 섹션에서는 Rocks Base CD 와 HPC Roll CD 를 사용하여 Rocks 클러스터의 Frontend 를 설치하는 방법을 설명한다.

> 프론트엔드로 정상적으로 동작하기 위해서는 최소한 Kernel Roll CD, Base Roll CD , HPC Roll CD 와 OS Roll CD 가 있어야 한다.
> Base+ HPC+ Kernel Meta Roll CD 는 각 BASE, Kernel, HPC Roll 대신에 사용할 수 있다.

1. Kernel Roll CD 나 BASE+ HPC+ Kernel Roll CD 를 frontend 노드에 넣고 fronted 노드를 재부팅 시킨다.

이 섹션 이후부터 우리는 bare-bone frontend 의 설치를 예로 들 것이다. 다시 말하면, Base+ HPC+ Kernel Roll 과 OS- Disk 1 Roll 과 OS – Disk2 Roll 를 이용할 것이다.

2.CD 를 넣고 frontend 를 부팅하면 다음과 같은 화면을 보게 될 것이다.

위와 같은 화면을 보게되면 다음과 같이 입력한다.

```
frontend
```

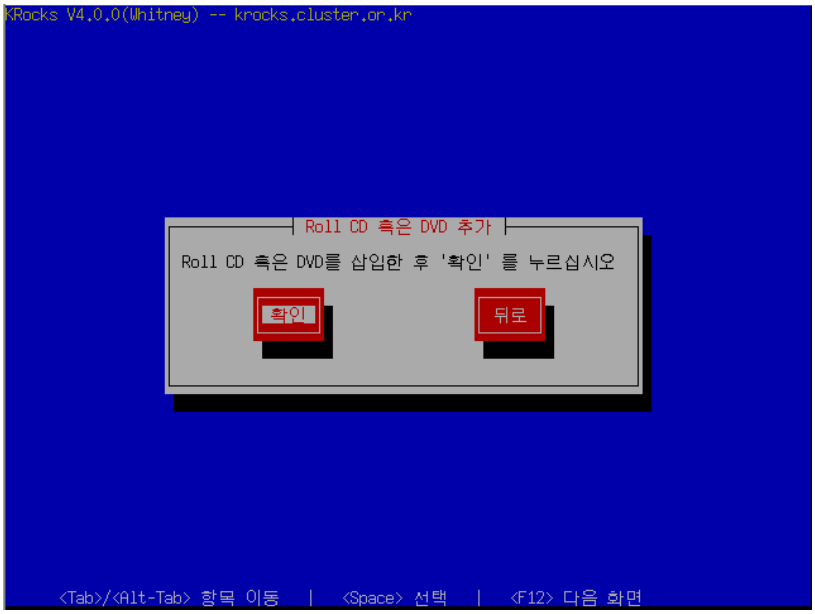
"boot:" 프롬프트가 보인 후 재빨리 화면이 바뀐다. 따라서 위의 화면을 놓치기가 쉽다. 아무런 입력이 없을 때, 기본적으로 계산 노드로 가정한다. 따라서 이 경우 frontend 설치는 되지 않을 것이며, 반드시 리부팅한 후 설치과정을 다시 시작해야 한다.

만약 설치를 실패하면 /tmp/ks.cfg kickstart 파일을 찾을 수 없다는 화면을 보게 될 것이다. 이 실패에 대한 좀더 많은 정보를 얻으려면 Alt-F3 과 Alt-F4 를 눌러 kickstart 와 시스템 로그를 확인하도록 한다.

3. frontend 를 타이핑하고 나면, 인스톨러가 동작할 것이다. 그러면 다음과 같은 화면을 볼 수 있다.

이 화면은 Base+HPC+Kernel Roll(Base, HPC 그리고 Kernel)로부터 필요한 roll 을 찾은 후 추가되었다는 것을 가리킨다.

4. Base+ HPC+ Kernel Roll 미디어를 CD/DVD 드라이브에서 꺼내면 다음과 같은 화면을 볼 수 있을 것이다.



OS-Disk1 Roll CD 를 드라이브에 넣은 후에 '확인'을 누른다. '확인'버튼은 스페이스바를 이용한다.

5. OS – Disk 1 Roll 을 찾게 되면 다음과 같은 메시지를 볼 수 있을 것이다.

6. OS – Disk 1 Roll 을 추가한 후의 화면을 보면 다음과 같을 것이다..

이 화면은 Base+ HPC+ Kernel Roll(Base, HPC 와 Kernel)과 OS- Disk 1 Roll 을 발견했고, 추가했음을 가리킨다.

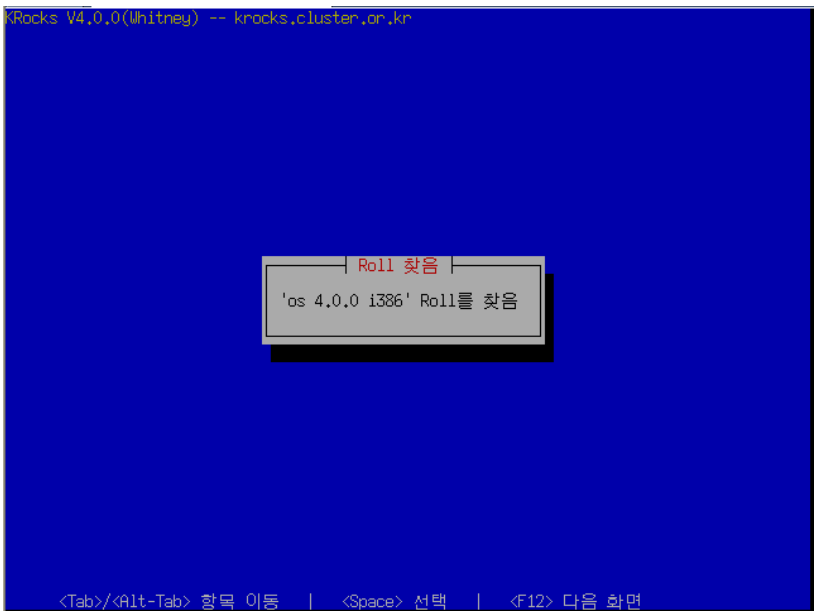
frontend 에는 OS-Disk2 Roll 가 설치되어야 한다. OS- Disk 2 Roll 를 설치하려면 '예'를 선택한 후 스페이스 바를 누르기 바란다

만약 frontend 를 Itanium 에서 설치한다면 OS-Disk 1 Roll 만 필요로 한다.

주의: i386 계열에서는 OS -Disk 2 Roll 를 꼭 설치해야 한다. 만약 그렇지 않으면 frontend 가 제대로 동작하지 않을 것이다.

7. 다시 아래와 같은 화면을 볼 수 있다.

8. OS – Disk 2 Roll 을 발견하면 아래와 같은 메시지를 볼 수 있을 것이다.



9. OS – Disk2 Roll 를 추가하고 나면 아래 화면과 같이 볼 수 있을 것이다.

위 화면은 Base+ HPC+ Kernel Roll(Base, HPC 와 Kernel), OS – Disk 1 Roll 그리고 OS – Disk 2 Roll 를 발견한 후 추가했음을 가리킨다.

이 정도면 필요한 모든 Roll 은 추가되었다고 볼 수 있다. 그 외의 Roll 을 추가하려면 '예'를 누른 후 Roll 을 삽입하면 된다(위의 화면에서와 비슷한 내용을 보게 될 것이다) 만약 더 이상 추가할 Roll 이 없다면 탭 키(tab key)를 이용하여 '아니오' 버튼을 선택한 다음에 스페이스 바를 눌러 선택한다.

10. 다음으로 "클러스터 정보" 화면을 볼 수 있다.

이 화면에서 중요한 항목 중의 하나는 “전체 호스트 이름(Fully Qualified Hostname)”이다. 그 외의 항목은 선택사항이다.
호스트 이름은 매우 주의 깊게 선택해야 한다. 호스트 이름은 frontend 와 계산 노드등의 12 개 이상의 파일에 기록이 된다. 만약 frontend 를 설치한 후에 호스트 이름을 변경하면 여러 클러스터 서비스들은 frontend 시스템을 찾지 못할 것이다. 이러한 서비스의 예로는 SGE, NFS, AutoFS, Apache 등이 있다.

각 항목을 모두 채우고 나면 ‘확인’ 버튼을 누르도록 한다.

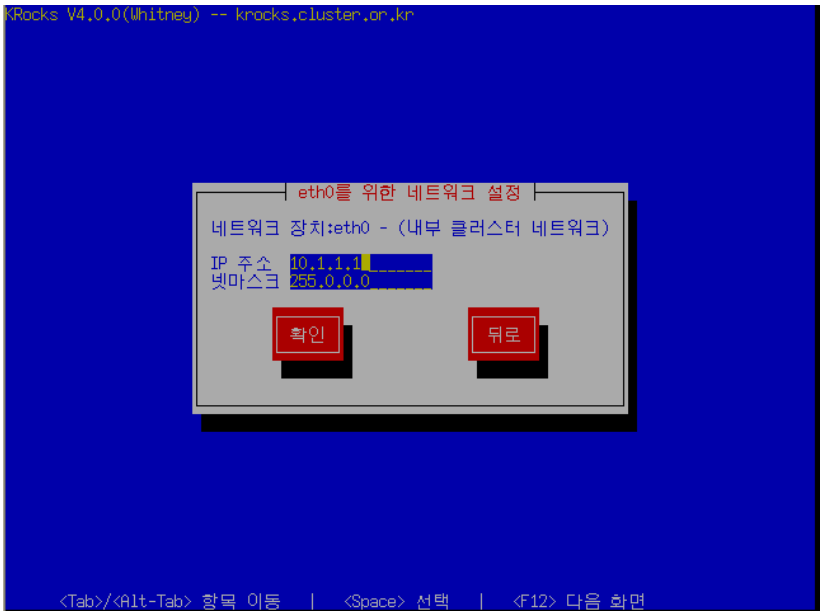
11. 디스크 파티셔닝 화면을 통해 ‘자동’ 혹은 ‘수동’으로 파티셔닝을 할 수 있다.

자동 파티셔닝을 선택하려면 ‘자동 파티션 설정’ 버튼을 누르도록 한다. 이를 선택하면 frontend 의 파티셔닝은 다음과 같이 설정된다.

Table 1-1. Frontend – root 디스크 파티션 기본 설정

Partition Name	Size
/	6GB
Swap	1GB
/export(/state/partition1로 심볼릭)	그외 모든 할당 공간

만약 수동으로 파티셔닝을 수행하려면 ‘Disk Druid’를 선택한다.



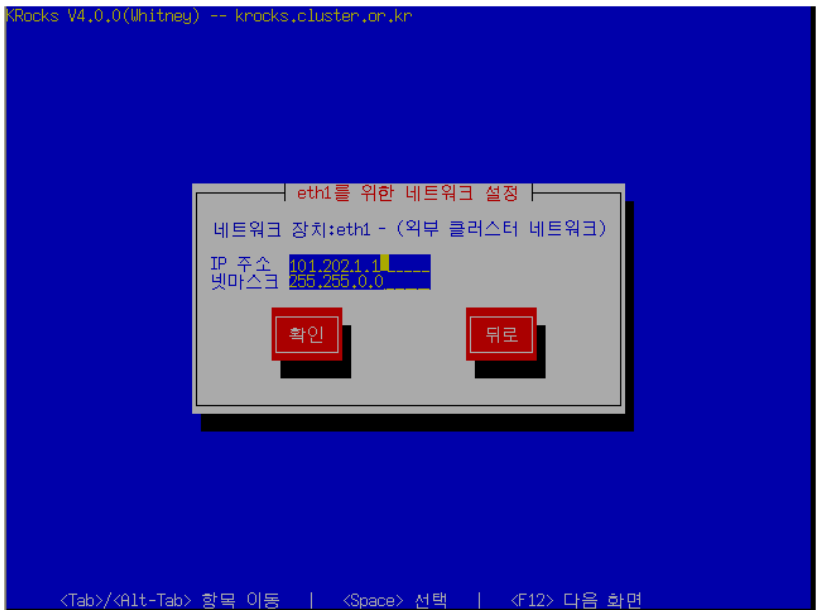
만약 수동으로 파티셔닝을 수행한다면 반드시 root 파티션은 최소한 6GB 이상으로 설정해야 하고, /export 파티션을 별도로 생성해야 한다.

자동 파티셔닝이 default 이며, 특별한 이유가 없는 한 자동 파티셔닝을 추천한다.

12. 내부(private) 클러스터 네트워크 설정 화면에서는 frontend 와 각 계산 노드간에 연결할 수 있는 이더넷 네트워크를 설정할 수 있다.

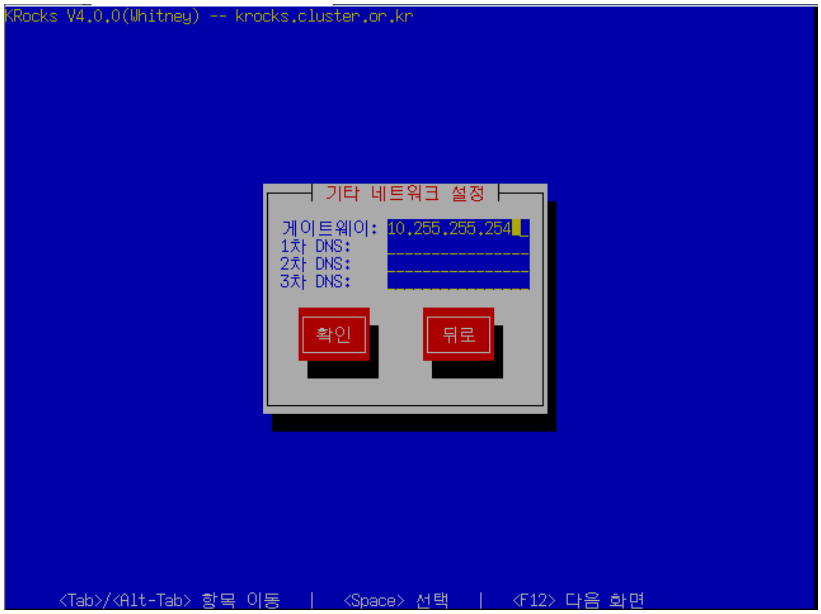
가능한 기본적으로 설정되어 있는 값을 이용하기 바란다. 각 항목을 이동하려면 탭(tab) 키를 이용하고, 확인 버튼이 활성화되면 엔터(enter) 키를 누르기 바란다.

13. 외부 공개(public) 클러스터 네트웍 설정 화면에서 frontend 와 외부 네트웍(예를 들면 인터넷)과의 연결을 위한 네트웍 변수를 setup 할 수 있다.



위의 화면에서는 frontend 시스템에 있는 외부 네트워크 설정의 예를 보여주고 있다.

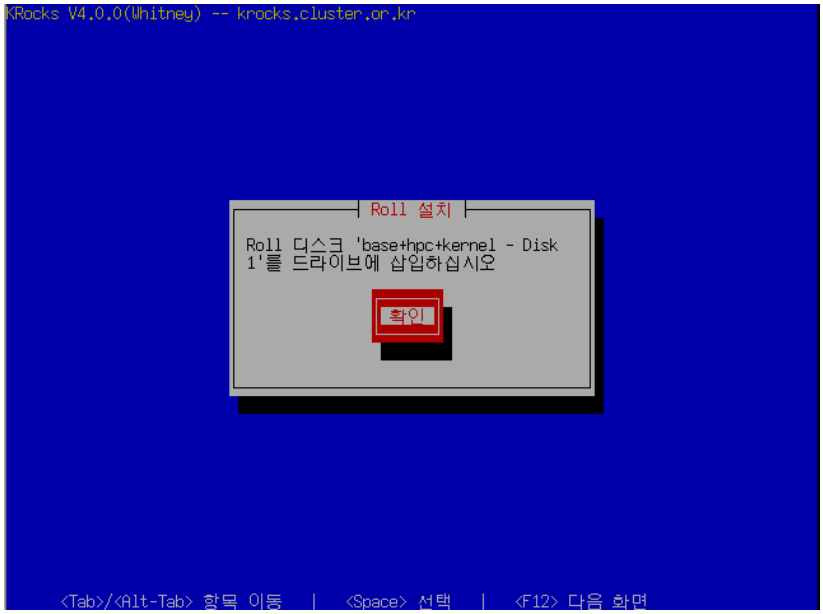
14. Gateway 와 DNS 를 설정한다.



15. 시간을 설정한다.

16. root 패스워드를 입력한다.

17. Frontend 에 파일시스템을 포맷한 후, frontend 설치를 시작할 때 추가한 roll CD 를 요구할 것이다.

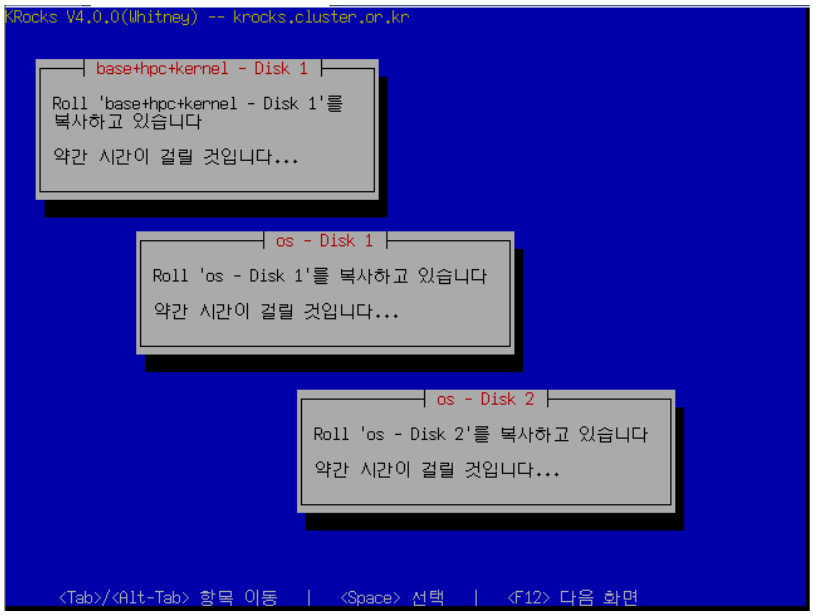


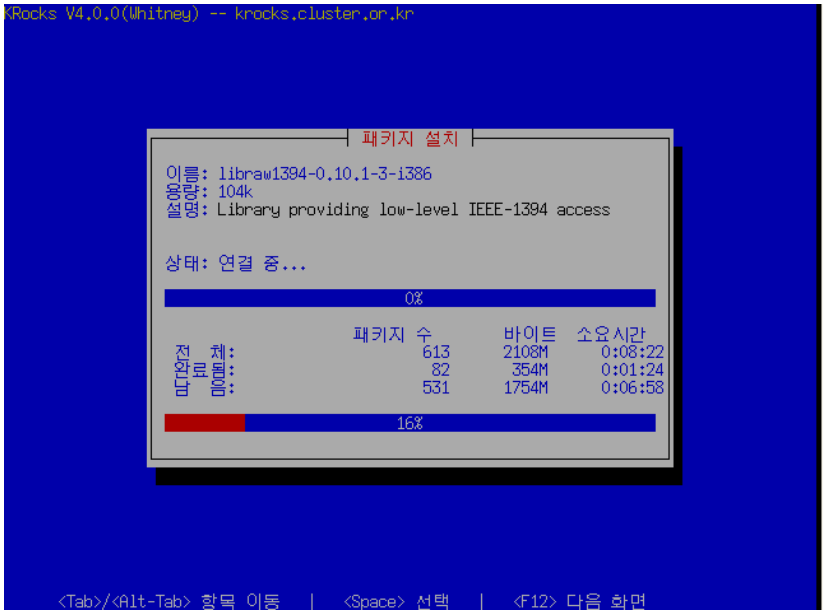
위의 화면은 Base+ HPC+ Kernel Roll 를 드라이브에 넣고 '확인'을 선택하는 화면이다. CD 의 내용은 frontend 의 하드 디스크에 복사될 것이다.


모든 Roll 이 복사된 다음에는 더 이상 사용자의 입력이 필요하지 않는다.

Installer 는 Rocks Base CD 의 내용을 frontend 의 local 하드 디스크에 복사할 것이다.

18. 마지막 roll CD 까지 복사하고 나면 패키지를 설치할 것이다.





19. 마지막으로 부트 로더가 설치되고 post 설정 스크립트가 백그라운드로 실행할 것이다. 이러한 백그라운드 작업이 끝나면 frontend 는 재부팅된다.

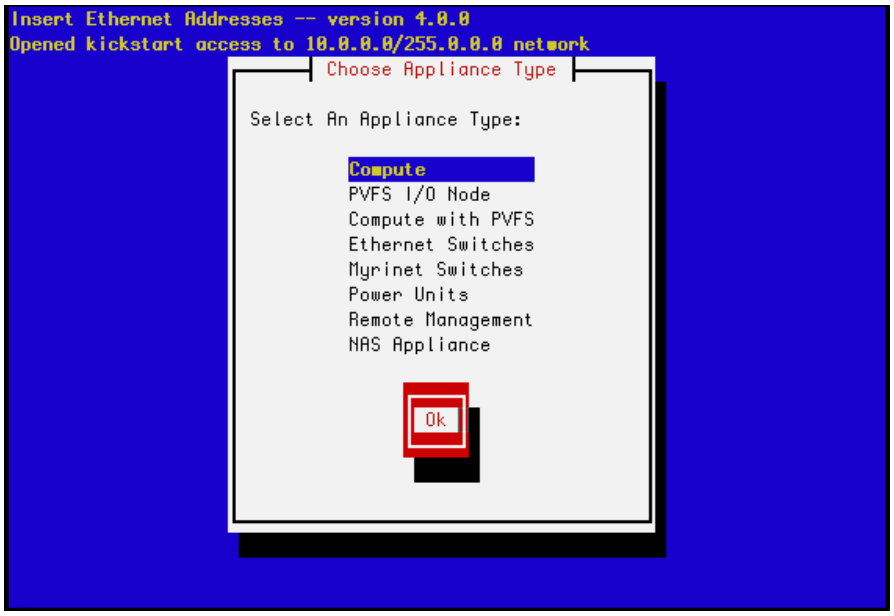
1.3. 계산노드 설치하기

1. frontend 노드에 root 로 로그인 하라.
2. 계산노드의 DHCP 요청을 capture 해서, Rocks MySQL database 에 입력하기 위해 다음을 실행하라

```
# insert-ethers
```

이것은 다음과 같은 화면을 출력할 것이다

만일 계산노드에 CD 드라이브를 갖고 있지 않고, 계산 노드에 장착된 네트워크 어댑터가 PXE 를 지원하지 않는다면 Floppy 를 이용한 PXE 부트 항목을 보기 바란다.



.

frontend 및 계산노드가 managed 이더넷 스위치로 연결되어 있을 경우, 위의 목록 중에서 'Ethernet Switches'를 선택하기를 원할 것이다. 이것은 많은 managed 스위치들이 사용자가 스위치를 설정하고 모니터할 수 있도록 스위치가 사용할 IP 주소에 대한 DHCP 요청을 하기 때문이다.

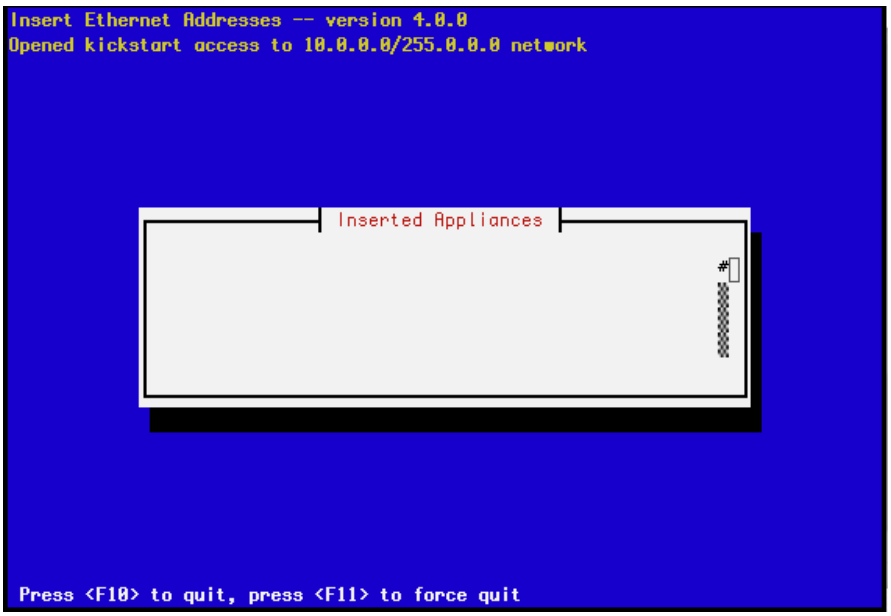
insert-ethers 명령은 managed 스위치를 위한 DHCP 요청을 capture 하여 이것을 Ethernet Switch 로 설정하며, 이 정보를 frontend 의 MySQL database 에 저장한다.

부가적으로 언급하면, 이더넷 스위치가 DHCP 요청을 broadcast 할 때까지 몇 분은 기다려야 한다. 대략 10 분 후, 혹은 insert-ethers 가 정확히 이더넷 스위치를 발견하여 그것을 설정한 후에, F1 키를 사용하여 insert-ethers 를 중지시켜야 한다.

그리고 나서 insert-ethers 를 다시 시작하여, 계산노드에 대한 설정을 계속해야 한다.

초기값, 즉 'Compute'를 선택하여 계산노드에 대한 설치를 시작하라.

3. 다음과 같은 화면이 나올 것이다:



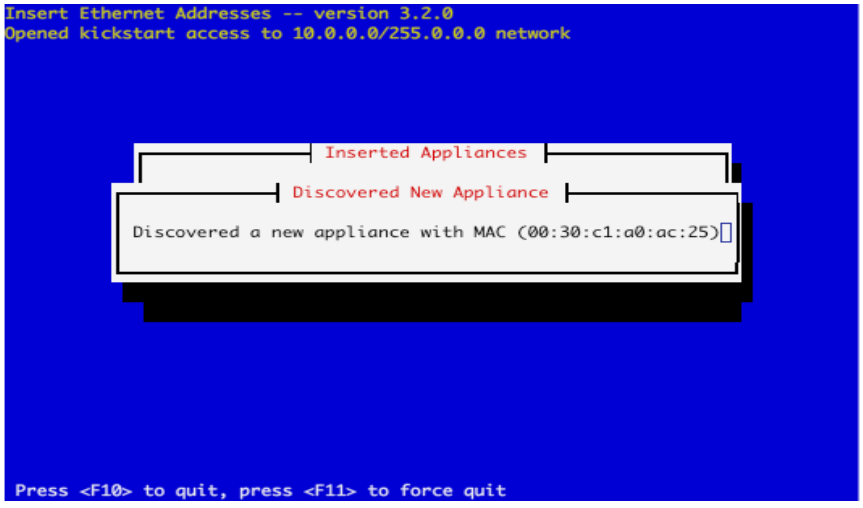
이것은 insert-ethers 가 새로운 계산노드를 기다리고 있음을 보여준다.

4. Rocks base CD 를 설치하고자 하는 첫 번째 계산노드에 넣어라. 클러스터의 계산노드의 IP 주소는 10.1.1.254 로부터 역순으로 주어진다.

만약 계산 노드에 CD 드라이브가 장착되어 있지 않다면 PXE(네트워크 부팅)을 이용할수 있다.

5. 설치하고자 하는 첫 번째 계산노드를 켜라

6. Frontend 노드가 DHCP 요청을 방금 전에 켠 계산 노드로부터 받는다면 다음과 같은 화면을 볼 것이다.



위의 화면은 insert-ethers 가 계산 노드에서 DHCP 를 요청받았고, 그 사항을 데이터베이스에 추가하고, 모든 설정 파일 즉, /etc/hosts, /etc/dhcpd.conf, DNS 와 배치 시스템 파일 등을 수정했음을 가리킨다.

위의 화면이 나타난지 몇초 후에는 다음과 같은 화면을 볼 수 있을 것이다.

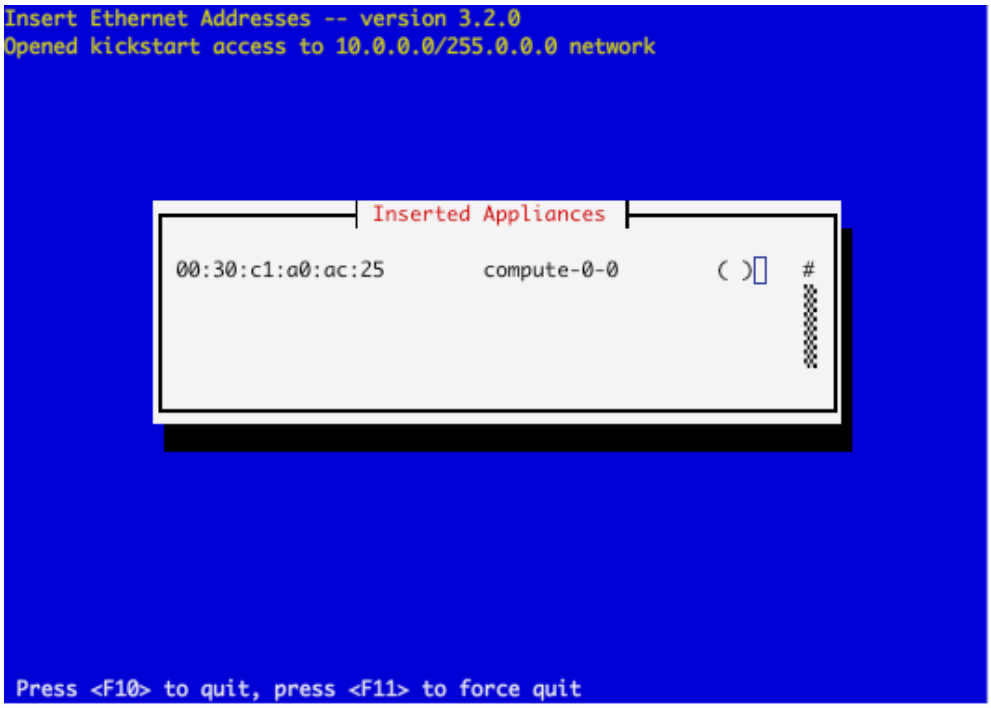


그림: insert-ethers 는 계산 노드를 발견했으며, compte-0-0 다음에 있는 "()"는 노드가 아직 kickstart 파일을 요청하지 않았음을 가리킨다.

이와 같은 종류의 output 을 insert-ethers 가 계산노드를 발견할 때 마다 볼 수 있다.

kickstart 파일은 411 키와 다른 주요 정보를 plain text 형태로 가지고 있기 때문에 네트워크로 전송할 때는 암호화한다. 부가적으로 오직 "인식된" 노드만이 이 kickstart 파일을 요청하는 것이 허용된다. Insert-ethers 는 새로운 노드를 추가하는 도구로 사용되기 때문에 주의해서 사용할 필요가 있다. 보안이 문제가 될 수 있는 상황이면, insert-ethers 상에 보이는 "미지의" MAC address 에 대해 의심해 볼 필요가 있다.

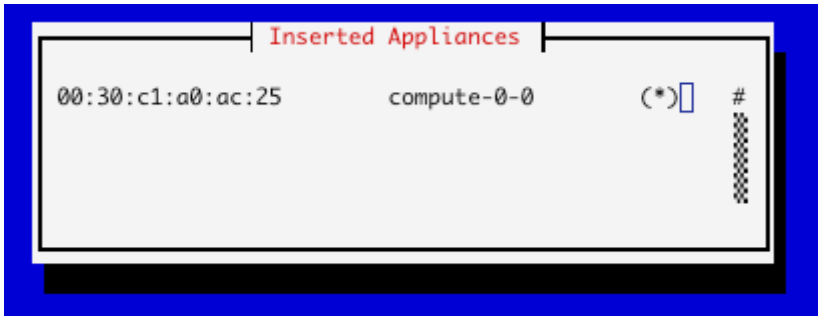


그림: 위의 그림은 계산노드가 성공적으로 kickstart 파일을 frontend 로부터 요청했음을 보여준다. 더 이상의 계산노드가 없다면, inset-ethers 를 중지시킬 수 있다. Kickstart 파일은 https 를 통해 보내지므로, 전송 중에 error 가 발생하면, HTTP 에러를 "(*)"대신 보게 될 것이다.

7. 이제 ssh 를 이용하여 설치 과정을 확인할 수 있다. Insert-ethers 에서 설치중인 계산 노드의 이름을 이용하여 다음 명령을 수행한다.(위의 예에서는 계산 노드의 이름을 compute-0-0 이라 하였다)

```
# ssh compute-0-0 -p 2200
```

8. 계산 노드의 설치 과정이 끝나면 CD 가 나올 것이다. CD 를 꺼내어 다음 계산 노드에 넣은 후 설치 과정을 계속하면 된다.

9. 첫 번째 cabinet(이것은 rack-mounted cluster setup 을 고려한 경우이다) 에 있는 모든 계산 노드의 설치가 끝나면 F10 키를 사용하여 insert-ethers 를 중지시킨다

10. 첫 번째 cabinet 에 있는 모든 계산노드에 대한 설치가 끝나면, 다음 cabinet 에 있는 계산노드에 대한 설치를 하고자 할 것이다. 단지 insert-ethers 명령을 다음의 옵션을 사용하여 다시 실행하면 된다.

```
# insert-ethers --cabinet=1
```

이렇게 하면, 설치될 새로운 계산노드의 이름은 e compute-1-0, compute-1-1, ... 과 같이 될 것이다.

1.4. Cross Kickstarting

Rocks 는 서로 다른 하드웨어 아키텍쳐를 가진 노드들로 구성된 heterogeneous(이기종) 클러스터를 cross-kickstarting 이라는 이름의 프로세스를 통해 지원한다. Frontend 와 다른 아키텍쳐를 가진 노드에 대한 설치를 지원하기 위해서, frontend 에 추가적인 패키지가 필요하다. 이 섹션은 어떻게 frontend 에서 다른 아키텍쳐의 노드의 설치를 할 수 있는가에 대해 설명한다.

Frontend 를 설치 혹은 업그레이드한 후에 다음의 명령을 따라 cross kickstarting 을 진행하라.
예로서, x86 architecture frontend 에서 x86_64 아키텍쳐를 위한 cross kickstarting 을 설명한다.

1. x86_64 에 필요한 Rocks roll 을 다운로드 한다(그 외의 roll 이 필요할 수도 있다)
 Rocks base CD 이미지를 /mnt/cdrom 으로 마운트한다. 이것은 CD 를 구울 필요 없이 진행할 수 있으며 다음 명령을 이용하면 된다.

```
# mount -o loop <roll-name>.iso /mnt/cdorm
```

그런 후 그 내용을 다음 명령을 이용하여 local 미러하도록 한다.

```
# rocks-dist -install copyroll
```

2. /mnt/cdrom 을 umount 하고 각각의 roll 에 대해 위와 같은 과정을 반복한다.
3. 새 아키텍쳐를 위한 배포판을 rebuild 한다.

```
# cd /home/install
# rocks-dist --arch=x86_64  --notouch dist
```

이것은 원래 아키텍처에 대해 빌드하도록 한다.

이제 frontend 는 cross-kickstart 를 통해 이기종(위의 예에서 x86_64)의 계산노드 및 클러스터 노드를 설치할 수 있다.

 rocks 은 PXE 를 이용한 cross-kickstart 를 지원하지 않는다. 따라서 커널 roll 을 포함하고 있는 native-architecture Rocks CD 로 non-native 계산 노드를 부팅해야 한다. 위의 예에서는 PXE 가 아닌 x86_64 부트 미디어를 이용하여 x86_64 계산 노드를 설치하는 것을 보여주고 있다.

1.5 기존의 Frontend 업그레이드 하기

만일 Rocks 클러스터를 이미 보유하고 있다면, 기존의 frontend 를 포함한 계산노드에 대해서 업그레이드를 할 수 있다. 아래에서 이 과정에 대해 설명한다

* 계산 노드가 PXE 설치를 할 수 있도록 다음과 같이 준비시킨다.
클러스터가 현재 네트워크로 연결된 상태이면, frontend 를 업그레이드하기 전에 계산 노드가 PXE 설치가 가능하도록 만들어 두어야 한다. 즉 계산노드가 PXE 설치를 지원한다면, 다음의 명령을 실행하여 계산노드가 재 부팅시 PXE 를 통하여 부팅하도록 해야 한다.

```
# ssh-agent $SHELL
# ssh-add
# cluster-fork 'touch /boot/grub/pxe-install'
# cluster-fork '/boot/kickstart/cluster-kickstart --start'
# cluster-fork '/sbin/chkconfig --del rocks-grub'
```

이제 계산노드를 shutdown 할 수 있다. frontend 의 업그레이드가 끝난 후에 “insert-ethers” 명령을 실행시킨 상태에서 각각의 계산노드를 하나씩 켜면 각 계산노드의 업그레이드를 진행할 수 있다. 이것은 rocks 를 처음 설치할 때와 비슷하지만 이때에는 각각의 계산 노드를 CD 를 이용하여 초기화하거나 각 계산 노드의 BISO 스크린에서 강제적으로 PXE 로 부팅할 필요는 없다. 이 과정은 클러스터를 새로 설치하는 것보다 훨씬 더 빠르게 업그레이드할 수 있다.

만일 각 계산노드가 PXE 부팅을 지원하지 않는다면, 위해 frontend 의 업그레이드가 끝난 후에 계산노드를 CD 를 사용하여 업그레이드를 진행하면 된다.

* Rocks 설치 CD 를 넣은 후, frontend 를 리부팅 시킨다.
* “boot:” 프롬프트를 보게 되면 다음과 같이 입력한다.

```
frontend upgrade
```

이제부터 설치 과정은 일반적인 frontend 를 설치하는 것과 동일하다. 따라서 단 한 가지를 제외하고는 매뉴얼의 “Frontend 설치하기” 섹션을 그대로 이용할 수 있다.

파티션을 수동으로 직접 설정해야 한다. 또 / partition 과 /boot partition (만일 이것이 존재하면) 다시 포맷해야 한다

1.5.1. Frontend upgrade 과정의 이해

우리는 frontend 업그레이드를 지원하기 위해 Red Hat installer 에 몇 가지 코드를 추가했다. 업그레이드 과정의 초기에 (root 파티션이 포맷되기 전에), 추가된 코드는 기존의 root 파티션("/ ")을 마운트하여 다음의 파일을 추출한다.

```
/etc/passwd
/etc/shadow
/etc/gshadow
/etc/group
/etc/auto.home
/etc/fstab
/etc/exports
```

모든 패키지의 설치가 끝나면, 추가된 코드는 이 추출된 파일의 내용을 root 파티션에 새로이 설치된 파일과 merge 시킨다. 새 파일과 추출된 이전의 파일이 공통적인 entry 를 가지고 있을 경우, 새 파일의 entry 가 우선하며, 이전의 entry 는 없어진다. root 파티션이 아닌 모든 파티션은 그대로 데이터를 유지하며, frontend 의 업그레이드가 완전히 끝나면, remount 된다. 예를 들면, 표준 Rocks frontend 는 root 파티션과 더불어 "/export"라는 이름의 파티션이 존재한다. 이 경우 root 파티션은 재 포맷되지만, /export 파티션은 그대로 기존의 정보를 유지하며 frontend 의 업그레이드가 끝나면 remount 된다.

1.6. 네트워크를 통해 Frontend 설치하기

이 섹션에서는 WAN(Wide Area Network)상의 중앙서버(central server)로부터 하나의 Rocks frontend 를 설치하는 방법을 설명한다. 이 과정은 "WAN kickstart"라고 불린다. Client frontend 는 Rocks 소프트웨어(base 와 Rolls) 그리고 설정 정보를 인터넷을 경유하여 받아 오며, 이것을 설치과정에 이용한다.

WAN kickstart 는 중앙서버와 frontend client 사이에 몇 가지 사용자의 설정이 필요하다. 이 단계는 서로를 인증해야 하는 양쪽 모두에서 필요하다. 하지만 일단 최초의 kickstart 가

한번 이루어지고 나면, 그 이후로는 client frontend 는 동일한 중앙서버로부터 재 설치시 인증을 위한 작업을 다시 할 필요가 없다.

1. client frontend 를 부팅하기 위한 아키텍처에 대한 Kernel Roll 를 포함하고 있는 Rocks Roll 를 사용하거나 간단한 부팅 디스켓(다운로드 페이지 참고)를 이용하도록 한다. Splash 화면이 나타나면 다음과 같이 입력한다.

```
frontend central=name.your.org
```

여기에서 “name.your.org”는 중앙서버의 FQDN 이어야 한다. http:// 혹은 다른 prefix 또는 suffix 를 사용해서는 안 된다. 만일 “Rocks”라는 이름을 사용하면 SDSC 에 있는 central-400.rocksclusters.org 를 중앙서버로 사용하여 설치를 시도하게 된다.

2. client frontend 가 위의 중앙서버로부터 처음으로 kickstart 를 통해 설치되는 것이면, 중앙서버는 IP 주소 혹은 DNS 를 통해 client frontend 를 인증해야 한다. 이를 통해 중앙서버의 보안에 관한 신뢰성을 조사할 기회도 될 것이다. 중앙서버의 insert-access 프로그램은 반드시 client frontend 에서 인자로 준 IP 주소, 네트워크, 도메인 이름과 일치해야 한다.

```
(중앙서버에서)
# insert-access .sdsc.edu
```

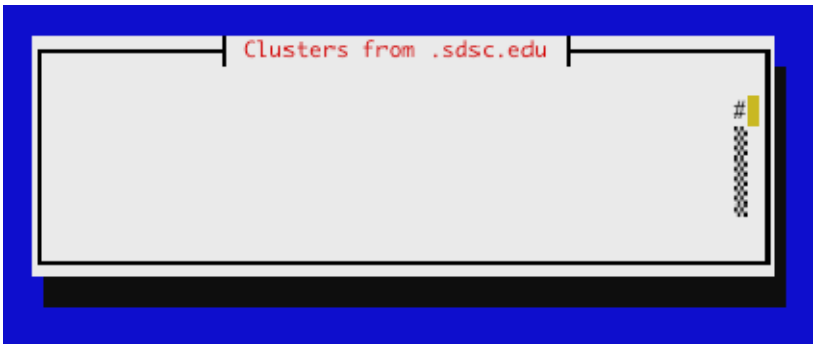


그림: 중앙서버의 insert-access 프로그램. 위의 설정을 한 경우, “.sdsc.edu” 도메인의 frontend 는 중앙서버를 통해 kickstart 를 실행할 수 있다; 모든 다른 도메인의 frontend 는 허용되지 않는다.

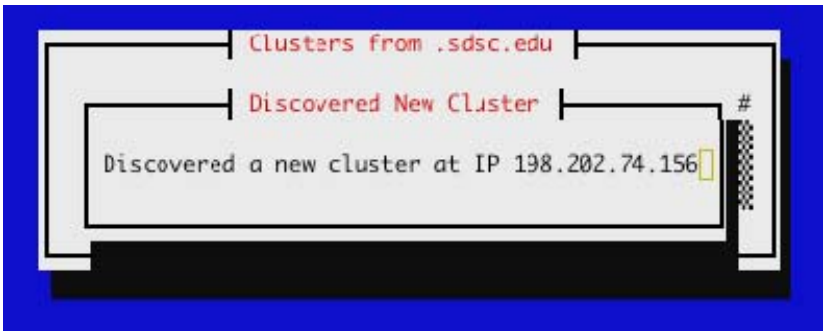


그림: 중앙서버의 insert-access 가 frontend 의 접근 시도가 감지되었음을 보여준다.

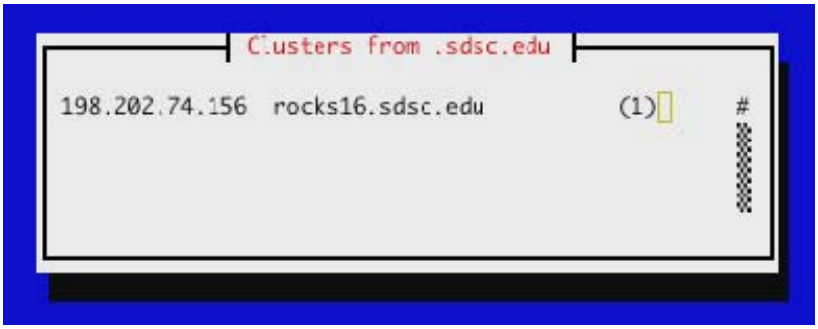


그림: 중앙서버의 insert-access 프로그램이 kickstart 를 하고자 하는 frontend 의 IP 주소, 이름, 및 kickstart 시도횟수를 보여준다.

3. 중앙서버를 통해 처음으로 설치를 하는 것이면, 보안 신뢰성(security credentials)에 대한 정보를 보여주는 아래의 화면을 볼 수 있을 것이다. 이 정보는 mod_ssi 를 이용한 apache 서버 인증의 내용으로 client frontend 에 보여진다. 중앙서버에서는 kickstart 를 시도하고 있는 client frontend 의 IP 주소를 보여준다.

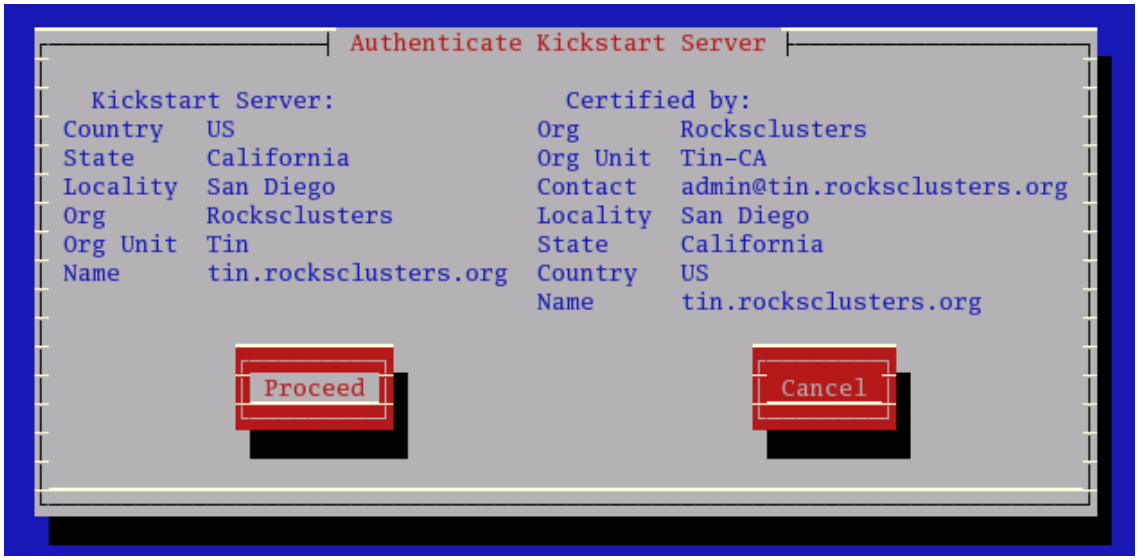
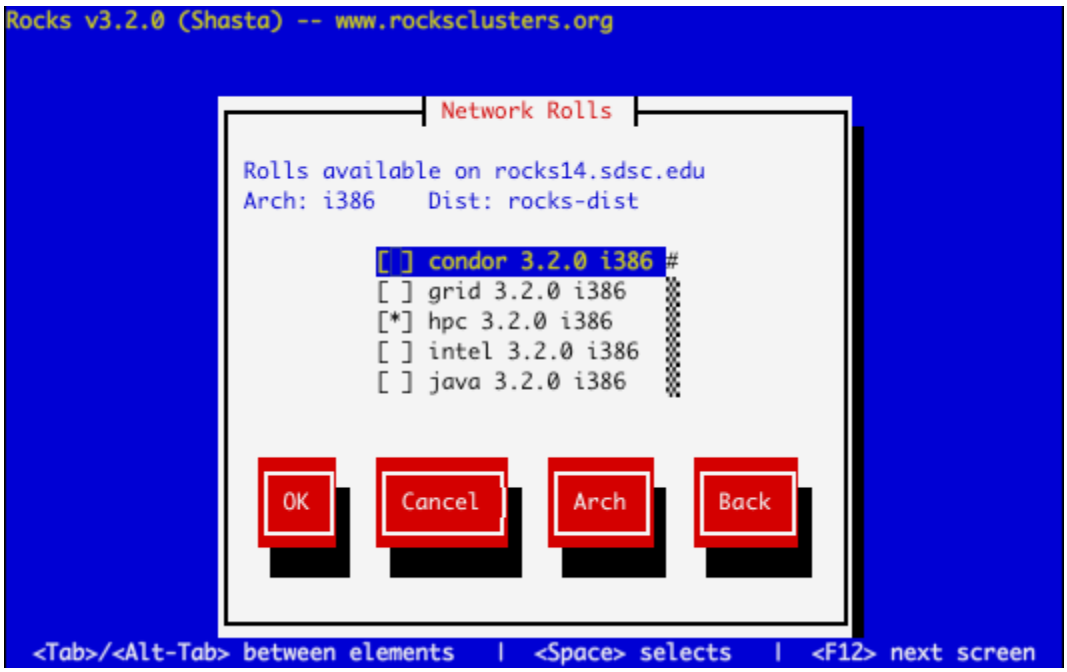


그림 설명: WAN kickstart frontend client 인증 스크린. 화면의 왼쪽은 중앙서버의 보안 신뢰성을 보여준다. 오른쪽은 중앙서버에 대한 인증서를 발행한 인증기관을 보여준다.

“Proceed” 를 선택하면 설치 과정은 계속될 것이나 “cancel”을 선택하면 설치과정이 중지될 것이다.

화면에 보여진 보안 신뢰성에 대한 정보가 정확하면, “Proceed”를 선택하여, 설치를 계속 진행하라. 만일 중앙서버가 insert-access 를 실행중이 아니면 “Could not get access to server” 에러 메시지를 볼 수 있을 것이다. 만일 예전에 이 중앙서버로부터 kickstart 설치를 진행한 적이 있다면, insert-access 프로그램을 중앙서버에서 실행할 필요가 없으며, 곧바로 사용자는 아래의 roll 선택 창을 보게 될 것이다.

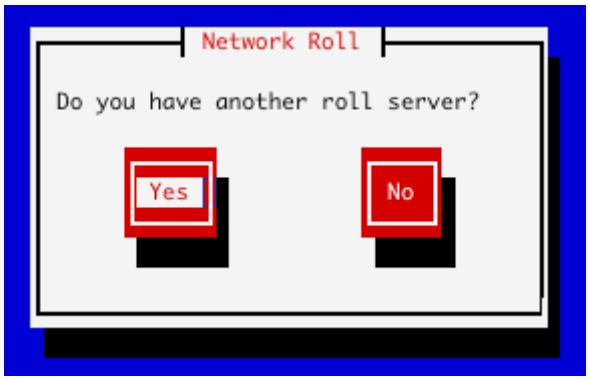
4. 이제 중앙서버를 통한 access 가 허용되면 client frontend 는 이제 HTTPS 를 통해 kickstart 파일을 다운로드한다. 그리고 다음의 roll 선택화면을 볼 것이다.



적절한 roll CD 를 선택한후 ‘OK’를 누른다.

 만일 이 기종의 아키텍쳐를 위한 roll 이 필요하다면, “arch”를 먼저 누른 후 다음의 화면에 보이는 바와 같이 다른 roll 서버를 선택하면 다른 아키텍쳐용의 roll 를 찾을 수 있다.

5. 다른 roll 를 얻기 위해 다른 중앙서버를 선택하기 위한 옵션은 다음과 같다.



새로운 network 을 통한 roll 의 추가가 끝나면, central server 의 질문에 “No”라고 답하면 된다.

6. 이제 하나의 Rocks base CD 와 더불어 여러 roll 을 추가할 수 있게 되었다. CD 기반의 설치과정은 앞서 “Frontend 설치하기”에 설명되어 있으며 이를 참조하면 된다.

7. roll 의 선택이 끝나면, 이제 익숙한 Rocks 의 설치 윈도우를 볼 수 있다. 설치과정은 선택한 roll 의 종류에 따라 다를 수 있다. 설치과정은 “Frontend 설치하기”에 설명되어 있는 내용을 참조하기 바란다.

8. Rocks base CD 의 내용이 이제 네트워크를 통해 frontend 에 옮겨질 것이다. 650MB 의 데이터가 표준 wget 을 사용하는 http 를 통해 전달될 것이다. 이 과정은 frontend 와 central 서버간의 네트워크의 상태 그리고 bandwidth 에 따라 걸리는 시간이 달라질 것이다.

설치프로그램은 선택한 network roll 을 다운로드한 후, 모든 rolls 포함하도록 배포판을 재빌드 할 것이다. 그리고 나서 패키지에 설치를 시작할 것이다. 그리고 마지막으로 post script 를 수행하고 일반적인 frontend 설치과정을 진행할 것이다.

Frontend 는 이제 설치가 되었으며, 계산노드를 설치할 준비가 되었다(“계산노드 설치하기”를 참조하라)


네트워크를 통하여 frontend 를 업그레이드할 수도 있다. 이때 boot 명령 즉, “frontend central=name” 뒤에 “upgrade” 키워드를 추가해야 한다.

고급 사용자의 경우 설치과정 동안 Rocks ekv 서비스를 활성화시키길 원할 수도 있을 것이다. ekv 는 사용자가 네트워크를 통해 설치과정을 interactive 하게 모니터링할 수 있게 한다. 이를 위해서는 키워드 “ekv”를 boot 명령에 함께 사용해야 한다. 일단 설치가

시작되면, 사용자는 ssh 를 통해 frontend 에 억세스할 수 있게 된다. ssh 가 연결이 되면 RedHat 및 Rocks 의 설치화면을 볼 수 있을 것이다. 불행히도 ekv 가 활성화되면, frontend 에 직접 물리적으로 연결된 콘솔의 화면은 알아 볼 수 없는 상태가 될 수 있다.


고급 사용자는 frontend 의 central 서버를 이용한 설치 시 다른 boot: 옵션이 존재한다는 사실을 알아야 한다. 만일 central 서버에 복수개의 배포판(즉, 하나는 영문판, 하나는 한글판)이 존재한다면, dist="이름"을 적절히 명시해주어야 한다. 즉, "boot frontend central=name.my.org dist=kr-dist"의 형태가 될 것이다. 여기에서 예로 든 "kr-dist"는 최상위의 배포판 디렉토리의 이름이어야 한다. 즉 central 서버는 이것을 "/home/install/external/it-dist"로 생각할 것이다.

1.6.1. insert-access

insert-access 명령은 중앙 서버에 접근하는 것을 제어하며, 중앙 서버에서 frontend 가 kickstart 를 실행하는 것을 제한한다. 접근 방식은 다음과 같이 두가지로 나누어진다.
(1) 누가 kickstart 파일에 접근하는가 (2)누가 RPM 패키지(Roll 접근)에 접근하가
Kickstart 파일은 암호키를 가지고 있기 때문에 앞의 두 경우에서 좀더 주의해서 다루어야 한다.

insert-acess 는 항상 중앙 서버에서 실행한다.

```
insert-access … [--all] [--rolls] [--remove] [--stop] [--permanent] … address
```

일반적으로 insert-access 는 insert-access 가 동작중일 경우에는 address 에 속하는 모든 clients 가 kickstart 에 접근하는 것을 허용한다. RPM 패키지에는 kickstart 파일을 보유하고 있는 클러이언트만 접근할 수 있다.

```
# insert-access .sdsc.edu
```

위 명령은.sdsc.edu 도메인을 가진 client 일 경우 kickstart 에 접근을 허용한다는 가리킨다. address 인수는 apache ACLs 에 있는 일반적인 형식을 따른다.

insert-access 에서 사용 가능한 옵션은 다음과 같다.

* --permanent

이 옵션은 address 에 속하는 client 가 kickstart 와 roll 에 대한 접근을 무제한 허용하는 것이다. Insert-access 에 이 플래그를 이용하면 사용자 입력을 요구하지 않는다.

* --remove, --stop

이 옵션은 address 에 속하는 clients 의 kickstart 와 roll 접근 속성을 제거한다. 만약 address 를 지정하지 않으면 '모든' client 의 접근 권한을 제거한다. 이는 --permanent 와는 반대 속성을 가지고 있다.

* --rolls

이 옵션은 address 에 속하는 client 가 RPM 패키지에 접근을 무제한 허용한다. Kickstart 에 대한 접근은 할 수 없다.

만약 여러 개의 중앙 서버에서 roll 를 얻고자 한다면 모든 중앙 서버에서 --rolls 에 대한 접근 권한을 가지고 있어야 한다. 우리는 종종 다음 명령어를 이용하여 우리 중앙서버에서 roll 를 얻고자 하는 모든 client 를 허용한다.

```
# insert-access  --rolls  --all
```

* --all

이 옵션은 "모든" address 를 나타내며 모든 client 를 지정한다. 만약 이 플래그를 사용하면 address 는 필요하지 않다.

1.7. Frontend 중앙 서버

중앙 서버는 다른 frontend 노드를 설치할 수 있으며 네트워크를 통해 roll 을 다른 frontend(client frontend)에 제공할 수 있는 하나의 frontend(중앙 서버)노드이다.

Rocks 3.3.0 부터 모든 Rocks 는 /home/install/rocks-dist 에 있는 표준 배포판에 WAN kickstart 를 위한 파일들을 갖추고 있다. 필요한 유일한 작업은 RPM 패키지를 웹 상에서 전송하기 위해, frontend 서버상에서 WWW 억세스를 허용하는 것이다. WWW 억세스 허용하기 부분을 참조하라.

중앙서버의 호스트 네임은 반드시 FQDN 이어야 한다. 특히 mysql 의 Rocks DB 의 app_globals 테이블에 있는 "PublicHostname"의 값이 정확한 값으로 수정되어야 하며, 외부 인터넷에서 이 이름으로 억세스가 가능해야 한다

1.7.1 중앙 서버에 서비스 할 Roll 추가하기

중앙 서버를 설치할 때 추가하지 않은 roll 를 서비스하고자 할 경우가 있다. 모든 frontend 는 client frontend 에 포함된 roll 를 서비스하지만 다른 roll 를 서비스하고자 할 경우도 있을 것이다. 그럴 경우 다음 과정을 이용할 수 있다

```
1. roll CD 를 CD 드라이브에 넣고, /mnt/cdrom 으로 마운트한다.
2. # rocks-dist copyroll
3. CD 를 unmount 시켜라
  (각 roll 에 대해 이 과정을 반복하라)
4. # cd /home/install
   # rocks-dist dist
```

만일 *.iso 이미지의 roll 를 가지고 있다면 “mount -o loop <name>.iso /mnt/cdrom” 명령을 통해 CD 를 구울 필요 없이 마운트할 수 있다.

2 장. 클러스터 계산 시작하기

2.1. Interactive Jobs 실행

2.1.1. mpirun 사용

Rocks 클러스터의 이더넷 디바이스를 위한 mpirun 은 이더넷 디바이스를 사용하는 MPICH jobs 을 시작할 수 있다.


HPL 을 root 가 아닌 일반유저로 실행하라
만일 클러스터에 사용자 계정을 갖고 있지 않다면, 다음과 같이 당신이 사용할 계정을 하나 만들어라
useradd *username*

예를 들어 노드당 하나의 프로세서를 가진 클러스터에서 두 개의 프로세서에 대해 benchmark 프로그램 High-Performance Linpack (HPL)을 시작하기 위한 방법은 다음과 같다.

1. 위에서 만든 계정의 홈디렉토리에 machines 이라는 이름의 파일을 만들고, 여기에 두 개의 노드의 이름을 적어라:

```
compute-0-0
compute-0-1
```

2. 두개의 프로세서를 위한 HPL 설정파일을 다운로드하여 이것의 이름을 HPL.dat 로 하여 당신의 홈디렉토리에 저장하라.

3. 그런 후에 다음의 명령을 frontend 에서 실행하라.

```
$ ssh-agent $SHELL
$ ssh-add
$ /opt/mpich/gnu/bin/mpirun -nolocal -np 2 -machinefile machines /opt/hpl/gnu/bin/xhpl
```

2.1.2. Cluster-Fork

Cluster-Fork 는 클러스터 계산 노드에 있는 명령을 실행한다.
종종 우리는 유닉스 표준 명령어를 사용하여 병렬 job 을 실행하기를 원한다. 여기에서 “병렬”은 클러스터내의 여러 개의 노드에 대해 동일한 명령을 실행하는 것을 의미한다.

우리는 파일을 옮기거나, 조그마한 테스트를 하고자 할 때, 그리고 관리자의 다양한 일을 할 때 이런 종류의 job 을 실행시키고자 할 것이다.

Rocks 는 이 목적을 위해 cluster-fork 라고 불리는 간단한 툴을 제공한다. 예를 들면 클러스터내의 모든 프로세스의 리스트를 보고자 할 때, 다음과 같이 명령을 내리면 된다.

```
$ cluster-fork ps -U$USER
```

cluster-fork 는 일련의 ssh 연결들을 사용하여 클러스터내의 모든 계산 노드에 대해 순차적으로(serially) 작업을 수행하게 한다. cluster-fork 는 죽어 있는 노드는 무시할 만큼 영리하다. 또한 작업은 보통 "blocking"된다: 즉 cluster-fork 는 다른 노드로 작업을 넘기기 전에 하나의 노드에서 작업이 시작될 때까지 기다리는 상태로 있게 된다. "—bg" 플래그를 사용하면 cluster-fork 를 background 에서 작업을 하도록 할 수 있다. 이것은 ssh 의 "-f" 옵션과 동일한 효과를 준다.

```
$ cluster-fork --bg hostname
```

종종 작업을 시작할 노드의 이름을 명시하기를 원할 때가 있을 것이다. 이것은 SQL 구문이나 약칭의 노드이름 명시를 통해 이루어 질 수 있다. 예를 들면, 클러스터의 첫 번째 rack 에 있는 계산 노드들에 대해서만 명령을 내리고자 하면, 다음을 실행하라.

```
$ cluster-fork --query="select name from nodes where name like 'compute-1-%'" [cmd]
```

--query 옵션은 노드 이름들의 column 을 리턴한다. 두 번째 방법은 사용자가 노드의 이름을 직접 명시해야 한다. 많은 노드에 대해 하나의 job 을 실행시키고자 할 때, 이것은 매우 귀찮은 일일 것이다. 때문에 우리는 이를 위해 노드이름에 대한 간단한 약칭 표현방법을 제공한다. 이 약칭 표현은 MPD job launcher 에서 빌려온 것이며, 사용자가 빠르고 간단하게, 대규모의 노드 범위를 지정할 수 있게 해준다.

노드 이름의 단축 형은 노드의 이름이 특정한 패턴을 따르도록 만들어졌을 때 사용될 수 있으며, " -nodes" 플래그를 사용할 수 있다.
즉, 노드 "compute-0-0 compute-0-1 compute-0-2"를 하나의 노드 범위로 지정하기 위한 약칭은 "compute-0-%d:0-2"이다. 이 방법은 노드의 이름들이 공통적인 접두사를 가지고, 이름들의 차이가 숫자로 주어질 때, 쉽게 사용할 수 있다. Rocks 계산 노드는 이러한 관례를 따르는 이름을 갖는다.

다른 MPD 방식의 약칭 이름의 예는 다음과 같다.

불연속적인 범위지정: "compute-0-0 compute-0-2 compute-0-3" -> "compute-0-%d:0,2-3"
여러 개의 범위지정 "compute-0-0 compute-0-1 compute-1-0 compute-1-1" -> "compute-0-%d:0-1
compute-1-%d:0-1"
곱하기를 이용한 반복 표현: "compute-0-0 compute-0-0 compute-0-1 compute-0-1 compute-0-2" ->
"2*compute-0-%d:0-1 compute-0-%d:2-2"

```
$ cluster-fork --nodes="compute-2-%d:0-32 compute-3-%d:0-32" ps -U$USER
```

앞선 예는 두 번째 그리고 세 번째 랙에 있는 64 개의 노드에서 실행중인 process 의 리스트를 보고자 할 때 사용될 수 있다.

2.2. Grid Engine 을 이용한 Batch Job 실행방법

이 섹션은 Rocks 클러스터에서 Grid Engine 을 사용하여 batch job 을 실행시키기 위한 간단한 명령 및 스크립트를 설명한다. job 은 스크립트를 통해 Grid Engine 에 제출된다. 여기에 Grid Engine 스크립트의 간단한 예로 테스트하기 위한 sge-qsub-test.sh 파일이 있다. 이것은 Grid Engine 이 두 개의 프로세서(다섯 번째 줄: #$ -pe mpi 2)에서 job 을 실행할 것을 요청한다. 그리고 프로그램(이 예에서 xhpl)을 시작시키기 위해 mpirun 이 사용할 임시 ssh key 를 만든다.

다음과 같이 PBS 에 job 을 제출할 수 있다:

```
qsub sge-qsub-test.sh
```

job 이 일단 시작되면, 다음의 명령을 통해 큐(queue)의 상태를 query 할 수 있다:

```
qstat -f
```

Grid Engine 은 job 의 결과를 네 개의 파일에 저장한다. 이중에서 다음의 두 개의 파일이 작업에 관한 보통 유용한 정보를 가진다.

```
$HOME/sge-qsub-test.sh.o<job id>
```

(stdout 메시지)

```
$HOME/sge-qsub-test.sh.e<job id>
```

(stderr 메시지).

나머지 두 개의 파일은 Grid Engine 의 상태에 대한 정보를 가지며 이름은 다음과 같다.

```
$HOME/sge-qsub-test.sh.po<job id>
```

(stdout 메시지)

```
$HOME/sge-qsub-test.sh.pe<job id>
```

(stderr 메시지).

2.2.1. Cluster-Fork 와 SGE

Rocks 클러스터의 기본 Batch 시스템인 SGE 는 사용자가 작업을 실행할 수 있는 일단의 노드들을 할당할 것이다. 그러나 사용자 대신 작업을 시작시키지는 않을 것이다. 대신에 SGE 는 $PE_HOSTFILE 이라는 이름의 파일에 할당된 일단의 노드들의 이름들을 명시할 것이다. MPI 병렬환경에서는 특정한 시작 스크립트가 이 파일의 내용을 읽어와서, mpirun launcher 를 시작시킬 것이다. 그러나 SGE 를 통해 비 MPI 작업을 실행하기 위해서는 cluster-fork 가 도움이 될 것이다(cluster-fork 에 대한 자세한 내용은 2.1.2 를 참조하라)

Cluster-fork 는 SGE 에 의해 주어진 PE_HOSTFILE 을 읽을 수 있으며, --pe-hostfile 옵션은 이 목적을 위해 사용될 수 있다. 예를 들면 SGE 가 할당한 모든 노드에서 hostname 명령을 사용하기 위해서는 다음과 같이 할 수 있다.

```
/opt/rocks/bin/cluster-fork --bg --pe-hostfile $PE_HOSTFILE hostname
```

2.3. 고성능 MPD Job Launcher 사용

MPD 는 새로운 high-performance job launcher 로서 MPICH 의 개발자에 의해 만들어졌다. 이것은 병렬 job 을 실행하는 데 mpirun 의 간단한 대체용으로 사용될 수 있다. 또 MPD 는 MPI 혹은 MPI 가 아닌 병렬 applications 을 실행하는 데 사용될 수 있다.

MPD 의 장점:

- 빠른 실행: 일 초 이내의 짧은 시간 내에 100 노드에서 job 를 실행(launch)시킬 수 있다.
- Jobs 이 끝난 후의 청소(cleanup): MPD 는 Ctrl-C 및 Ctrl-Z (즉 SIGTERM 및 SIGINT) signal 을 정확하게 전파한다. 따라서 frontend 에서 하나의 명령으로 runaway job 을 중지시킬 수 있다.
- Fault Tolerance: MPD 는 node failure 가 발생하여도 job 을 시작시키고, signal 을 보낼 수 있다.

MPD 의 단점:

- 호환성: MPI 어플리케이션은 MPD job launcher 를 사용하기 위해 반드시 재컴파일해야 한다. 그러나 일반적인 어플리케이션(즉, MPI 라이브러리를 이용하지 않는 어플리케이션)은 그대로 MPD 에서 사용될 수 있다.
- 보안성: MPD 는 job 을 시작하거나 신호를 전달하기 위해 ssh 를 사용하지 않으며 명령어를 암호화하지 않는다. 이 정도 수준의 보안성은 외부로부터 보호된 내부 네트웍에서만 적절하다. Rocks 클러스터는 이러한 기준을 잘 따르고 있으므로 MPD 를 상대적으로 안전하게 사용할 수 있다.
- 복잡성: MPD 는 클러스터 노드 사이에서 반드시 만들어 진후에 지속되어야 하는 대몬들의 "ring"에 의존한다.
- 속도: MPD 는 이전의 launcher 보다 더 frontend 의 NFS file server 에 부담을 줄 것이다. 병렬 job 을 실행중인 모든 노드는 거의 동시에 어플리케이션 실행파일의 복사본을 요청할 것이다. 이것은 NFS 서버를 고통스럽게 할 것이다.(모든 홈 디렉토리는 NFS 를 통해 서비스된다는 것을 기억해라). 큰 사이즈의 job 은 작은 job 보다 NFS 서버를 위태롭게 할 수 있다. MPD 의 차후 버전은 실행 파일의 분배를 위해 “ring”의 효율적인 communication pipe 를 이용할 것이다.

2.3.1. MPI 어플리케이션에 MPD 사용하기

interactive 하게 혹은 batch 작업으로 MPI 어플리케이션을 실행하기 위해서, 사용자는 사용자의 어플리케이션을 MPICH 의 MPD 버전으로 재컴파일해야 한다. 이 라이브러리는 일반적인 MPICHG 와 동일하며, 동일한 인터페이스를 지원한다. 이것의 경로는 다음과 같다.

```
/opt/mpich-mpd/gnu/lib
```

일단 실행파일이 MPD 라이브러리를 통해 컴파일이 되면, mpirun 명령을 사용할 수 있다.

```
/opt/mpich-mpd/gnu/bin/mpirun
```

이 MPD 버전의 mpirun 은 예전의 버전과 비슷하게 작동한다. 자세한 내용은 맨페이지 혹은 “—help” 옵션을 사용하라. 사용자의 작업이 실행중인 모든 노드에 대해 해당 사용자는 access 하여 control 할 수 있는 권한을 가진다. 여기에는 실행중인 병렬 작업에 대한 signal 을 보낼 수 있는 권한도 포함된다.

2.3. Linpack 실행하기

2.3.1. Interactive 모드 실행하기

이 섹션은 Rocks 클러스터에서 HPL job 을 늘려가는(scale up) 방법을 설명한다. 시작하기 위해 먼저 이더넷 위에서의 mpirun 명령(앞 섹션 참조)을 사용하여 두 개의 프로세서에 대한 HPL 을 실행하는 절차를 먼저 해볼 것을 권고한다. 그런 후에, 프로세서의 수를 늘리기 위해서는, *machines* 파일에 엔트리를 추가하면 된다. 예를 들어 4-프로세서 job 을 계산 노드에서 실행하기 위해서는 다음의 내용이 *machines* 파일에 들어 있으면 된다.

```
compute-0-0
compute-0-0
compute-0-1
compute-0-1
```

그리고 나서 HPL.dat 파일의 프로세서의 수를 수정해야 한다. :

변경 전:

```
1 Ps
2 Qs
```

변경 후:

```
2 Ps
2 Qs
```

주의: HPL 이 사용하는 프로세서의 수는 P 와 Q 를 곱한 값이다. 즉 16-프로세서 job 을 위해서는 다음과 같이 지정해야 한다.

4 Ps
4 Qs

그리고 마지막으로 mpirun 명령의 np 옵션을 수정해야 한다.

```
$ /opt/mpich/gnu/bin/mpirun -nolocal -np 4 -machinefile machines /opt/hpl/gnu/bin/xhpl
```

job 을 좀더 오래 실행하기 위해서는 problem size 를 증가시켜야 한다. 즉 HPL.dat 에서 Ns 변수를 증가시키면 된다.

변경 전:

```
1000 Ns
```

변경 후:

```
2000 Ns
```

 주의해라, Ns 변수를 2 배 증가시키면, 일의 양은 4 배가 된다

.  HPL.dat 에 있는 변수에 대한 자세한 설명은, HPL Tuning 을 참조하라.

2.4.2. Batch 모드 실행하기

이 섹션에서는 Grid Engine 을 통해 제출된 HPL job 을 확장하는(scale up) 방법을 설명한다. 시작하기 위해, interactive 모드에서 HPL job 을 확장하는(scale up)하는 방법에 대한 설명을 참고할 수 있다. job 이 사용하는 프로세서의 수를 늘려가기 위해, HPL.dat 를 적절히 수정해야 한다(이것은 앞의 interactive 모드에서 설명했다) 그리고 나서 sge-qsub-test.sh(이것은 batch job 시작하기에서 설명했다)를 다음과 같이 수정해야 한다.

```
#$ -pe mpi 2
```

예로 이 세팅을 4-프로세서에 대한 job 으로 만들고 싶으면, 다음과 같이 수정하면 된다.

```
#$ -pe mpi 4
```

이제 당신의 (새로운) job 을 Grid Engine 에 제출하라

3 장. 클러스터 모니터링

3.1. 클러스터 모니터링 하기

Rocks 클러스터는 클러스터의 상태와 설정을 모니터링하기 위한 일단의 웹페이지를 제공한다. 클러스터의 frontend 노드는 내장된 Apache 웹 서버를 이용하여 이 웹 페이지를 서비스한다. 이 섹션에서는 모든 rocks 클러스터 노드에 대해 원격에서 모니터링 할 수 있는 웹 기반의 모니터링 툴을 설명한다. 보안상의 이유 때문에, 웹 접근은 오직 클러스터 내부 네트웍에만 기본적으로 허용된다. 또 모니터링 웹 페이지를 보기 위해서는 약간의 노력이 필요할 수 있다. 클러스터 모니터링 웹 페이지를 보기 위한 가장 쉬운 방법은 클러스터의 frontend 노드에 모니터, 키보드, 마우스를 부착한 후, 그것의 X window 를 적절히 설정하는 것이다. 다음의 명령어를 이용할 수 있다.

```
# system-config-display
# startx
```

일단 위 명령이 제대로 실행되면 RedHat 데스크탑 환경이 나타날 것이다. 클러스터 사이트를 보려면 웹 브라우저에서 http://localhost 를 입력하라.

3.1.1. SSH Tunneling 을 이용한 클러스터 웹사이트의 접근

웹 페이지를 보는 첫 번째 방법은 웹 브라우저 스크린을 안전하게 암호화된 SSH 채널을 통해 보내도록 하는 것이다. 이렇게 하기 위해 다음의 단계를 따르면 된다.

1. 클러스터의 frontend 에 로그인하여 패스워드를 입력하라.

$ ssh mycluster

2. X server 가 local machine 에서 실행 중 인지 확인하라. 브라우저를 다음의 명령으로 실행하라. ssh 프로세스가 브라우저 윈도우를 위한 암호화된 채널을 만들 것이다

$ firefox --no-remote &

3. local machine 에서 웹 브라우저가 나타날 때까지 기다려라. URL 이 http://localhost/인 웹페이지가 당신의 클러스터의 홈페이지로 나타나야 한다.

3.1.2. 웹사이트 접근에 대한 제어

영구적으로 public 네트웍 위에 있는 특정한 machine 에서의 cluster 에 대한 웹 접근을 허용하기 위해서 다음의 단계를 따르면 된다. Apache 의 access 제어 지시어는 클러스터 웹

사이트의 가장 민감한 부분에 대한 보호를 제공할 것이다. 하지만, 이것을 효율적으로 이용하기 위해서는 약간의 노력이 필요할 것이다.

HTTP (web access protocol)는 clear-text 채널이다. Apache 웹 서버는 매우 발전된 프로그램이며, 많은 테스트를 거쳤지만, PHP 엔진의 보안구멍(security holes)들이 발견되고 있으며, 이를 악의적으로 이용한 경우도 있었다. 따라서 다음의 과정을 따라, 외부에 웹 억세스를 허용한다면, 당신의 클러스터는 악의적인 공격이나 침입(breakins)을 받기 쉬운 상태가 될 수 있다.

1. /etc/sysconfig/iptables 을 편집하라. 이 파일내의 다음의 특정한 줄에 대한 주석을 제거하라

```
...
 # 클러스터에 대한 웹 access 를 허용하기 위해 아래의 줄에 대한 주석을 제거하라(즉 # 기호를 없애라)
#-A INPUT -m state --state NEW -p tcp --dport https -j ACCEPT
#-A INPUT -m state --state NEW -p tcp --dport www -j ACCEPT
... 다른 방화벽 관련 지시어 ...
```

2. iptables 서비스를 다시 시작하라. root 로 실행해야 한다.

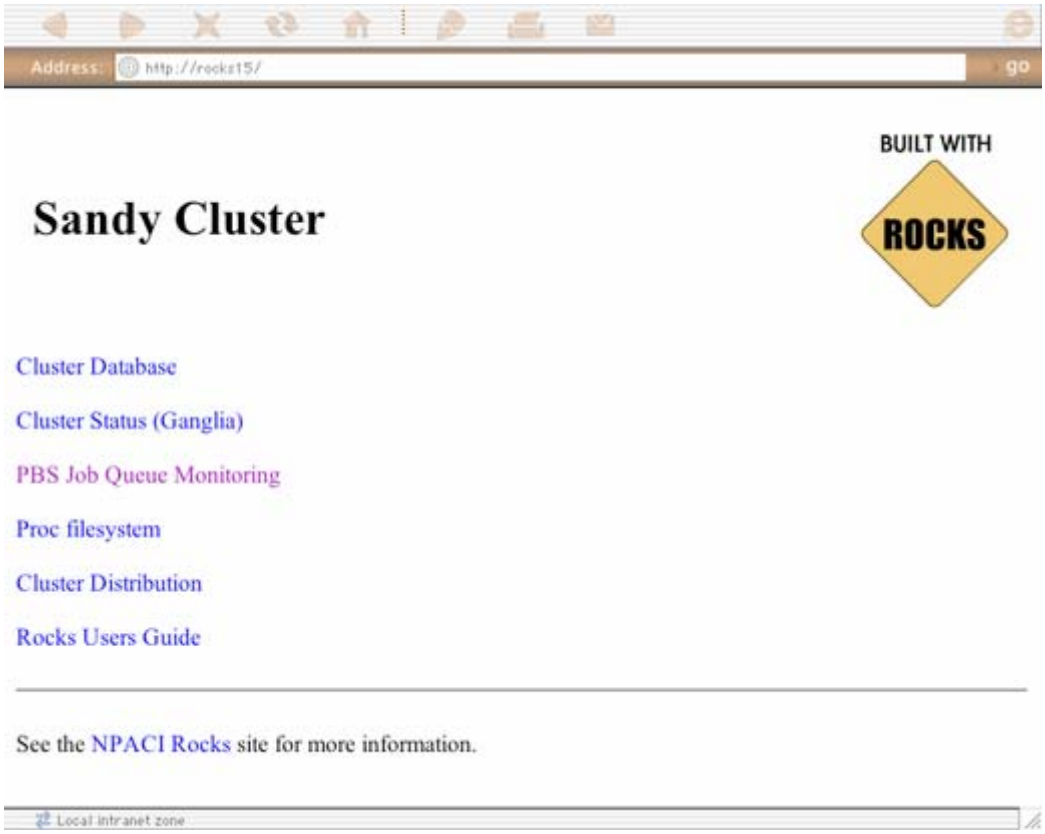
 # service iptables restart

3 외부에서 http://my.cluster.org/ 주소를 사용하여 웹 브라우저를 열고 위에서 바꾼 세팅을 테스트하라. 여기에서 웹 주소 my.cluster.org 는 DNS 에 등록되어 있는 frontend 의 이름이어야 한다.

만일 이 주소로 연결할 수 없다면, 대부분의 경우 문제는 당신의 웹 브라우저와 클러스터간의 네트웍 연결이 잘못된 경우일 것이다. “ping"을 사용하여 frontend 와 당신이 웹 브라우저를 실행하고 있는 machine 간의 연결을 체크하라.

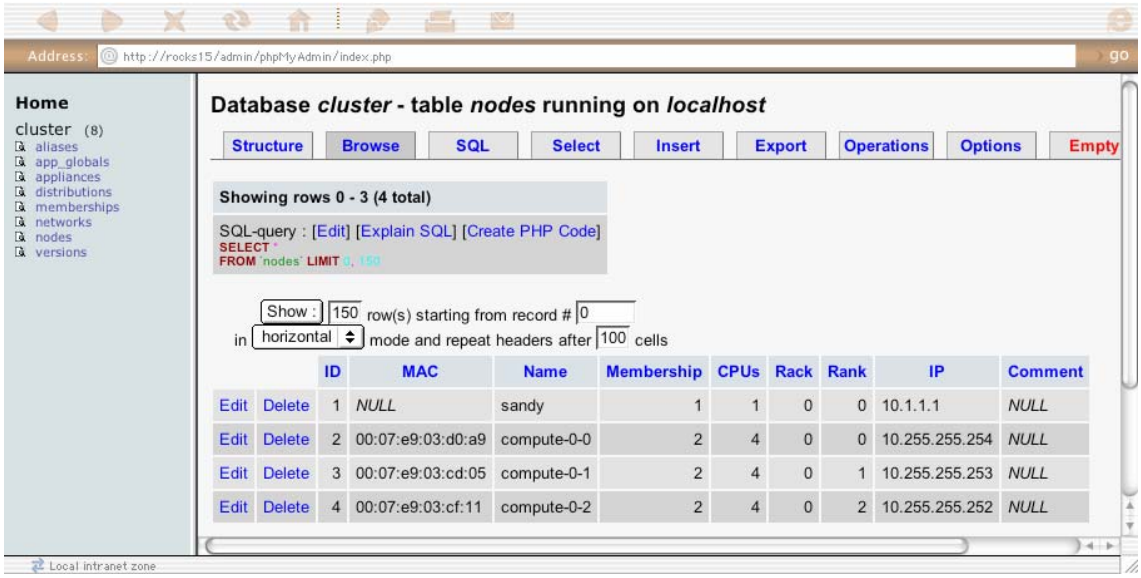
3.1.3. Contents Page

만일 성공적으로 클러스터의 웹 서버에 연결할 수 있다면, Rock 의 *Table of Contents* 로 시작하는 화면을 볼 수 있을 것이다. 이 단순한 페이지는 클러스터에 대한 이용 가능한 모니터링서비스와 링크되어 있다.



3.2. 클러스터 데이타베이스

이 웹 어플리케이션은 현재의 Rocks SQL 데이터베이스를 볼 수 있게 하며, 수정도 할 수 있도록 해준다. Rocks 는 이 데이터베이스를 클러스터의 설정과 클러스터의 노드에 대한 정보를 저장하는데 사용한다. 이 데이터베이스의 structure 와 semantics 에 대해 보다 자세히 알고 싶다면, 이 문서의 뒷부분에 나오는 Rocks 클러스터 Schema 페이지를 참고하기 바란다.



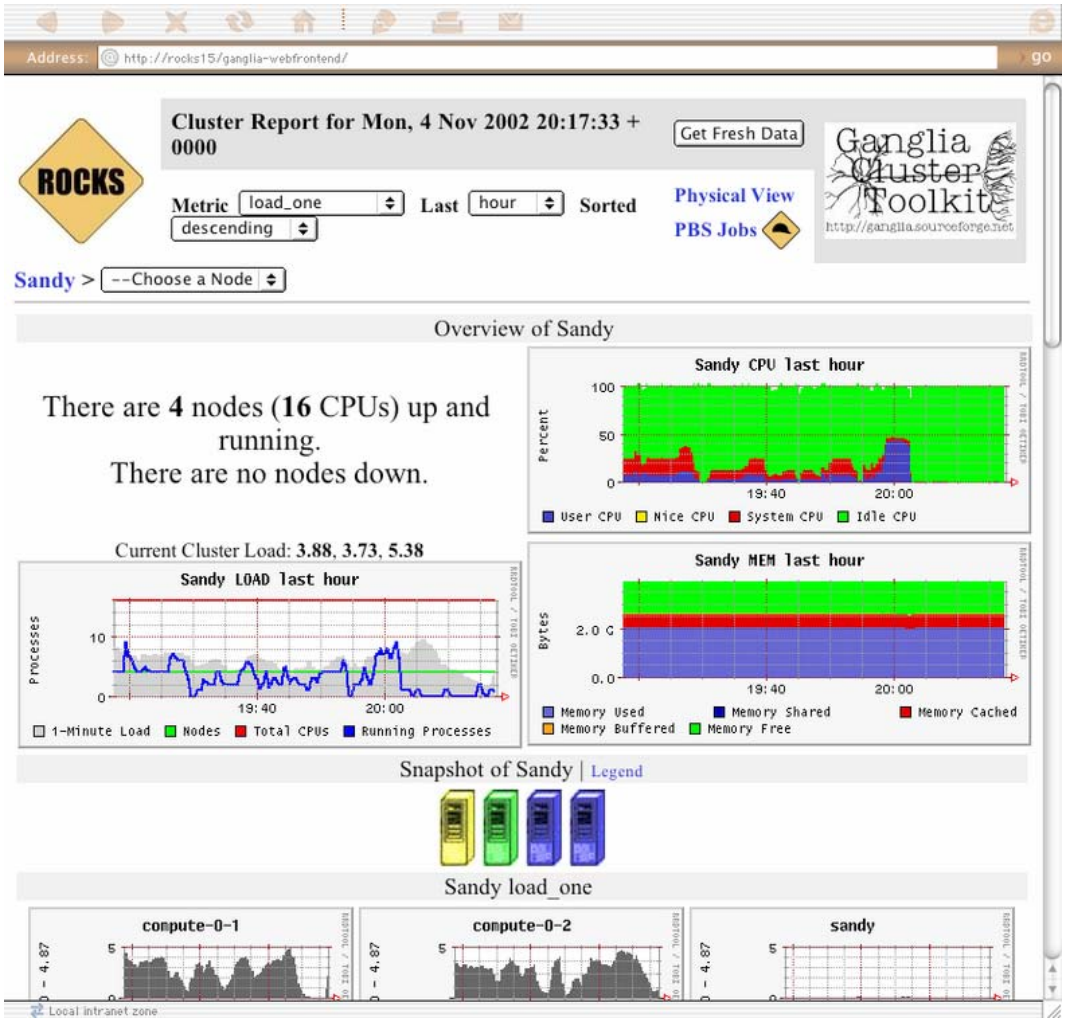
이 웹 데이터베이스 어플리케이션은 현재의 데이터베이스에 대해 Queries, Inserts, Updates, and Deletes 등의 SQL 명령을 허용한다. 이 웹 어플리케이션을 통한 데이터베이스의 변경내용은 이 데이터베이스를 이용하는 서비스에서 즉각적으로 이용될 수 있다. 이러한 기능 때문에 우리는 이 페이지에 대한 access 를 내부 네트웍에 있는 hosts 로만 제한한 것이다. 이 데이터베이스 어플리케이션에 대한 access 를 확대하고자 한다면, /etc/httpd/conf/rocks.conf 파일을 다음과 같이 수정하라.

```
<Directory "/var/www/html/admin/phpMyAdmin">
        Options FollowSymLinks Indexes ExecCGI
        AllowOverride None
        Order deny,allow
        Allow from 127.0.0.1
        Deny from all
</Directory>
```

위의 "Allow" 지시어 뒤에 추가적으로 이 웹 데이터베이스 어플리케이션에 대한 억세스를 허용하고자 하는 host 의 IP 주소를 추가하면 된다. Allow 지시어에 대한 format 은 Apache 메뉴얼에서 확인할 수 있다.

3.3. 클러스터 상태(ganglia)

이 링크를 통해 이용할 수 있는 웹 페이지는 각 클러스터 노드에서 실행중인 Ganglia 모니터 프로그램(gmond 및 gmetad)을 통해 얻어진 현재의(live) 클러스터 정보에 대한 graphical interface 를 제공한다. Ganglia 모니터 프로그램(gmond)은 CPU load, free Memory, disk usage, network I/O, operating system version 등 다양한 metric 에 대한 현재 값(value)을 수집한다 (local machine 에 대한 metric value 를 /proc 으로부터 직접 구한 후, 이것을 multicast channel 을 통해 Ganglia 모니터 프로그램(gmond)이 실행중인 모든 노드에 전파한다. 따라서 각 노드의 Ganglia 모니터 프로그램(gmond)은 클러스터의 모든 노드에 대한 정보를 가지고 있게 된다) 이 metric values 들은 private 네트웍을 통해 (tcp/ip 방식으로 frontend 의 Ganglia 의 또 다른 핵심프로그램인 gmetad 에) 전달되며, 이 페이지에서 볼 수 있는 historical graph 를 만들기 위해 frontend 에서는 RRDTool 을 사용한다.



The Rocks Cluster Group 은 Rocks 를 사용하여 만들어진 많은 클러스터로부터 받고 있는 Ganglia 정보를 웹에 게시하는 “Meta"라는 이름의 웹 페이지를 운영하고 있다. 이것은 Ganglia 가 가진 힘과 확장성을 볼 수 있는 기회를 제공할 것이다. 이 메타 페이지는 http://meta.rocksclusters.org/에서 볼 수 있다.

Ganglia 는 Berkeley 의 Matt Massie (massie@cs.berkeley.edu)에 의해 2000 년도에 개발되었으며, 현재도 계속 Berkeley, SDSC 등과의 협력을 통해 발전되고 있다. 또한 GPL 소프트웨어 라이센스 하에서 Sourceforge.net 을 통해 배포된다.

3.4. 클러스터 Top

이 페이지는 UNIX 표준 명령인 “top”의 클러스터용 버전이다. 이 페이지는 클러스터의 각 노드로부터 받은 각 노드의 프로세스 정보를 제공한다. 이 페이지는 각 노드들의 보다 정밀한 작동상태(activity)를 알고자 할 때 유용하다.

클러스터의 “top”은 여러 가지 면에서 표준의 “top” 명령과 다르다. 첫 번째로 각각의 줄은 하나의 “HOST” 이름과 data 의 나이(즉 데이터가 frontend 에 도착한 후 현재까지 지난 시간)를 나타내는 “TN”이라는 속성을 가지고 있다. 프로세스를 측정하는 것 자체가 어느 정도 자원을 소비하기 때문에, 계산노드에 대한 프로세스 측정 및 보고는 평균 60 초에 한번씩 이루어진다. 프로세스 항목에 TN=30 이라는 내용은 그 프로세스에 대한 정보가 30 초 전에 보고되었음을 말하는 것이다.

간결성과 성능에 주는 영향을 최소한으로 줄이기 위해, 각 노드는 자신이 갖고 있는 CPU 의 수 만큼에 해당하는 프로세스의 수를 보고하도록 되어 있다.(즉 CPU 가 하나인 노드는 1 개의 최 상위 프로세스만 보고하며, 이에 반해 2 개의 CPU 를 갖고 있는 2-way SMP 노드는 2 개의 최 상위 프로세스에 대해 보고를 한다) 즉, 여기에서 보이는 프로세스는 측정 시 각 노드에서 가장 높은 “%CPU”를 갖는 프로세스이다. 불행히도 현재, 각 노드 당 보고할 수 있는 프로세스의 수를 관리자가 직접 “조정”할 수 없다. 이러한 제한은 Ganglia 모니터링 시스템의 구조에 기인한다. 즉 Ganglia 는 정보를 전달하는 기능만을 가지고 있으며, 프로그램이 실행 중에 새로운 변수(parameter)를 받아서 이를 적용하는 능력은 가지고 있지 않다. 하지만 가장 많이 CPU 를 이용하는 프로세스(the most CPU intensive process)를 보는 것만으로도 현재 CPU 가 어떻게 활용되고 있는지에 대해 많은 것을 알 수 있을 것이다.

프로세스 데이터는 각 노드에서 /proc 파일시스템에 대해 raw processing 을 거쳐 얻어진다. 각 프로세스에 대한 메모리 통계 정보는 UNIX 표준명령인 “ps” 출력을 통해 얻은 것과 일반적으로 약간 다르며, /proc/[pid]/statm 가상파일을 통해 계산된다.

Process 열의 변수에 대한 설명

TN

이 행에 있는 정보의 나이(초)

HOST

이 프로세스가 실행중인 노드의 이름

PID

프로세스 ID, 각 노드에서 이 ID 는 유일한 값을 가진다.

USER

프로세스의 사용자

CMD

이 프로세스를 실행한 명령어 이름

%CPU

CPU 사이클 중 프로세스가 차지하고 있는 퍼센트

%MEM

물리적인 메모리 중 이 프로세스가 차지하고 있는 퍼센트

SIZE

이 프로세스의 "text" memory 의 크기(KB), BSS 의 크기에 따라 다르지만 일반적으로 이 값은 실행 파일의 크기와 비슷하다.

DATA

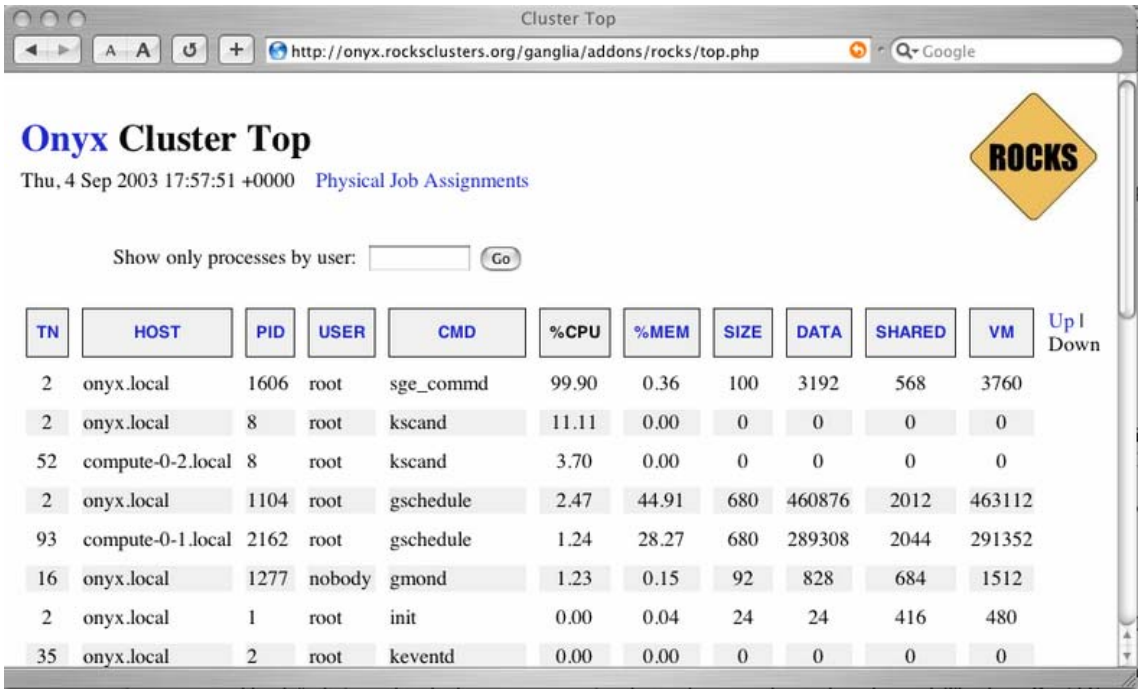
프로세스에 의해 동적으로 할당된 메모리에 대한 근사 값(KB), 이 값은 프로세스가 자식 프로세스를 가지고 있지 않으면 Heap 및 Stack 을 포함한다.

SHARED

이 프로세스가 사용하는 shared memory 의 크기(KB) 이 값은 표준 libc 및 loader 와 같은 shared 라이브러리를 포함한다.

VM

이 프로세스에서 사용하는 전체 virtual 메모리의 크기(KB)



3.5. 그 외의 다른 클러스터 모니터링 기능

3.5.1. /proc 파일시스템

이 링크에서 Linux 의 /proc 파일시스템을 표준 Apache 파일시스템을 통해 볼 수 있다. 이 파일과 디렉토리는 실제 디스크에 있는 것들이 아니며, 단지 요청이 있을시(on request) 리눅스 커널이 동적으로 생성한(dynamically generated) 것이다. 이들은 자원의 사용현황(usage)과 실행중인 프로세스에 대해 동적인 정보를 제공한다. /proc 파일시스템이 갖고 있는 위와 같은 특성 때문에, /proc 에서 제공하는 정보는 가장 최신의 것이며, 사실상 /proc 의 파일을 요청한 시점의 OS 의 상태를 나타낸다.

하지만, 이 파일들에 포함되어 있는 정보들은 해커 혹은 악의적인 목적을 가진 사람들에게 매우 유용한 정보를 가지고 있다. 즉 사용자 이름, 프로그램 변수와 더불어 네트웍 인터페이스 및 방화벽에 대한 정보를 포함하고 있다. 따라서 기본적으로 이 링크는 데이터베이스 웹 인터페이스와 더불어 "private" 네트웍에서만 사용할 수 있도록 제한되어 있다.

3.5.2. 클러스터 배포판

이 링크는 frontend 노드의 /home/install 디렉토리 트리에 있는 파일시스템을 보여준다. 이 디렉토리는 클러스터의 노드들을 설치하는 데 사용할 RPM 패키지와 다양한 노드타입을 정의하기 위해 사용되는 XML kickstart 그래프의 저장소(repositories) 역할을 한다. 클러스터를 설치하는 데 사용할 배포판을 여기에서 검토할 수 있다.

클러스터에 현재 설치된 소프트웨어의 버전에 대한 지식의 제공은 해커에게 이용할 수 있는 보안구멍(security hole)에 지식을 주는 것과 다름없으므로, 기본적으로 이 링크에 대한 접근(access)은 private 네트웍으로 제한된다.

3.5.3. 킥스타트 그래프

이 링크는 장비 타입(appliance type 즉, 노드 타입)에 따라 소프트웨어를 선택하는데 사용하는 현재 kickstart 그래프의 이미지를 보여준다. Rocks 는 이 그래프에 그려진 노드(여기서 노드는 그림내의 타원들을 의미)와 선(edges)에 따라서 kickstart 파일을 만든다. (Rolls 는) 이 그래프에 새로운 추가를 하거나 수정을 할 수 있으며, 새로운 장비 타입을 정의할 수도 있다. 현재 Rocks 에서는 각 장비타입이 이 그래프의 시작 노드에 의해 달라진다. 그러나 보다 복잡하게 정의를 할 수도 있다.

이 링크에 의해 보여지는 GIF 이미지는 이 링크를 클릭하면서 동시에 만들어진 것이며, 따라서 현재의 구조를 보여준다. 이 그림은 GraphVIZ 프로젝트의 일부인 “dot” 어플리케이션을 이용하여 만들어 진다.

3.5.4. 클러스터의 라벨 만들기

"Make Labels" 링크는 클러스터의 각 노드를 구별하는 데 사용할 라벨(여기에는 클러스터의 이름과 각 노드의 이름)들이 들어 있는 PDF 문서를 만든다. 이 문서를 저장한 후, “Avery 5260 address stock”에 출력하여 쉽게 클러스터의 각 노드의 전면에 부착할 수 있다.

3.5.5. Rocks 사용자 가이드

이 마지막 링크는 Rocks 사용자 가이드를 보여준다. 이것은 단지 공식 Rocks 웹사이트 http://www.rocksclusters.org/에 있는 문서의 local 복사본이다. 이 가이드를 통해 클러스터에 새로운 노드를 추가하는 방법과 문제 해결시 유용하게 쓰일 수 있는 FAQ 정보

등을 제공한다. 당신이 이미 많은 양을 읽고 있을지도 모르는 이 문서의 내용이 이 링크를 통해 제공된다.

이 링크는 표준 Rocks 클러스터에서 이용 가능한 모니터링 툴을 보여준다. 또 데이터베이스 편집, 클러스터 자원 상태 및 병렬 작업 상태 파악, 소프트웨어 패키지 저장소 점검 등을 여기에서 할 수 있다. 이러한 툴은 현재 활발히 개발 중이며, 차후 추가적인 웹 페이지를 가질 수도 있다.

3.6. Ganglia 를 이용한 다중 클러스터 모니터링 하기

Ganglia 는 여러 개의 클러스터의 모니터링 데이터를 관찰하여 이를 보고하는 기능을 가지고 있다. 모니터링 되고 있는 일단의 클러스터를 Ganglia 에서는 Grid 라는 이름으로 부르고 있다. 이 섹션은 여러 클러스터로 구성된 Grid 를 모니터링하기 위한 셋업 과정을 설명한다.

기본 아이디어는 하나의 클러스터의 frontend 에 있는 gmetad 대몬이 자신의 클러스터 이외에 다른 클러스터에 대해서도 관찰하도록 셋업하는 것이다. 이 과정은 하나의 위치로부터 대 규모의 클러스터 집합을 모니터하기 위해 반복될 수 있다.
이 내용에 대해 설명하기 위해 우선 “A”와 “B”라는 이름을 가진 두개의 클러스터가 있다고 하자. 클러스터 “A” 에 있는 frontend 를 최상위 레벨의 모니터로 선택한다.

1. “A” frontend 에서 /etc/gmetad.conf 에 다음의 내용을 추가하라.

```
data_source "Cluster B" B.frontend.domain.name
```

그리고 나서 “A” frontend 에서 gmetad 를 재시작 하라.

2. “B” frontend 에서 /etc/gmond.conf 에서 다음의 줄을 추가하라.

```
trusted_hosts A.frontend.domain.name
```

그리고 나서 “B” frontend 에서 gmetad 서버를 재시작 하라

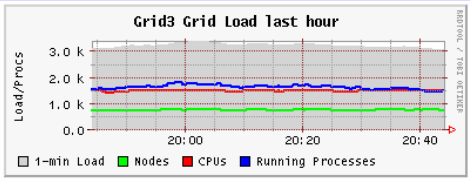
3. “A”에서 Ganglia 웹 페이지를 열어서 확인하라. 이제 “B”의 정보와 양 클러스터의 summary 정보를 볼 수 있어야 한다.

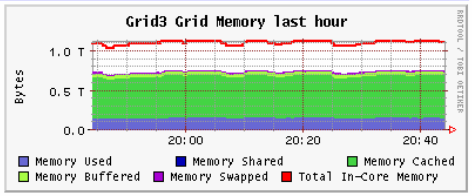
아래의 screenshot 예는 iVDGL Physics Grid3 project 를 보여준다. 이와 유사한 방법으로 대규모의 Grid 를 Ganglia 를 통해 모니터링 할 수 있다.

Grid3 Grid (23 sources) (tree view)

CPUs Total: **1493**
Hosts up: **740**
Hosts down: **214**

Avg Load (15, 5, 1m):
213%, 209%, 207%
Localtime:
2003-11-21 20:44

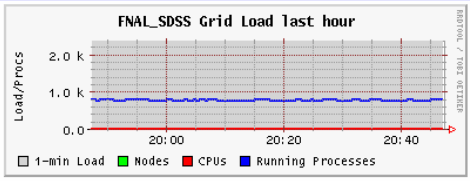


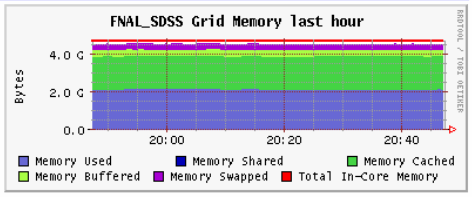


FNAL_SDSS Grid (tree view)

CPUs Total: **10**
Hosts up: **5**
Hosts down: **0**

Avg Load (15, 5, 1m):
23711%, 23713%, 23716%
Localtime:
2003-11-21 20:47





4 장. 클러스터 서비스

4.1. 클러스터 서비스

이장에서는 Rocks 클러스터의 다양한 다른 서비스를 설명한다.

4.2. 411 Secure Information Service

411 보안정보서비스는 NIS 와 비슷한 기능을 Rocks 클러스터에 제공한다. 이것의 이름은 긴급 전화번호인 411 의 이름을 따서 붙여졌다. 우리는 411 을 passwd 파일, user, group 데이터베이스 파일 등을 분배하는 데 사용한다.
Public Key 암호화를 사용하여 411 은 파일의 내용을 보호하며, 이것은 NIS 처럼 줄 단위로 작동하는 것이 아니라 파일 단위로 이루어진다. 411 은 RPC 에 의존하지 않으며, 대신에 파일 자체를 분배한다. 이것의 주된 역할은 파일 단위 분배 데이터베이스를 일관성 있는 문법을 통해 안전하게 유지하는 것이다. 411 의 디자인의 목적 중에는 변화가 발생할 때 확장성을 가져야 한다는 것과 짧은 대기 시간을 가져야 한다는 것, 그리고 문제 발생시 이에 대한 복원력을 가져야 한다는 것 등이 있다.


Rocks 3.1.0 Matterhorn 버전부터, 411 은 기존의 NIS 대신에 /etc/passwd 와 다른 로그인 관련 파일을 분배하는 기본 방법으로 사용된다. Rocks 는 더 이상 NIS 를 지원하지 않는다.

4.2.1. 411. 서비스 사용하기

411 서비스는 의도적으로 시스템 관리자를 위해 NIS 를 모방한 인터페이스를 만들었다. 물론 NIS 에 없는 요소들이 411 에는 있다. 즉, RSA public 및 private 암호화 키 등이 그 예이다. 그러나 Rocks 는 411 를 가능한 한 NIS 경험자가 쉽게 사용할 수 있도록 만들었다.

/var/411/Files.mk 에 열거된 파일은 자동적으로 411 에 의해 서비스 된다. 즉 여기에서 열거된 파일들은 클러스터내의 모든 계산노드의 411 agent 가 최신 버전으로 항상 유지한다는 것을 의미한다. 이것은 NIS 와 유사한 /var/411/Makefile 이라는 이름의 makefile 을 통해 이루어질 수 있다. 411 시스템이 모든 변화를 즉각 반영하도록 하기 위해서는 다음의 명령을 실행하면 된다.

```
# make -C /var/411
```

위 명령은 매 시간 cron 에 의해 frontend 에서 실행되며, passwd 의 변화 등등의 것들을 각 계산노드에 전파한다. 필요 시 Files.mk 에 새로운 파일을 추가하여 이 411 를 서비스를 이용할 수 있다.

모든 411 파일을 강제적으로 다시 암호화한 후 모든 노드에 변경 사항을 알리고자 한다면 frontend 에서 다음과 같은 명령을 실행한다.

```
# make -C /var/411 force
```

411 서비스는 최적의 성능을 내기 위해 private 네트워크에서 IP broadcast 메시지를 이용한다.
모든 계산 노드에서 frontend 에서 최신 파일을 강제적으로 추출하려면 다음 명령을 실행한다.

#cluster-fork 411get --all

4.2.2. 구조

4.2.2.1. Listener

client 노드는 master 노드의 "411alert" 메시지를 받기 위해 Ganglia multicast 채널을 열어 놓고 있다. 따라서 master 서버는 "411put"을 실행하는 동안 411 파일을 암호화하자마자 바로 411alert 메시지를 전송할 것이다. 이 alert 메시지는 client 노드에게 어떤 파일에 변화 생겼으며, 따라서 이 파일을 가져와야 한다는 것을 알려주는 단서가 된다. 이런 방법으로 411 시스템은 일반적으로 "변화"에 대해 빠른 응답 시간을 유지하게 된다. 411 alerts 는 수신받지 못한 메시지가 있을 경우를 대비하여 주기적으로 재전송한다.

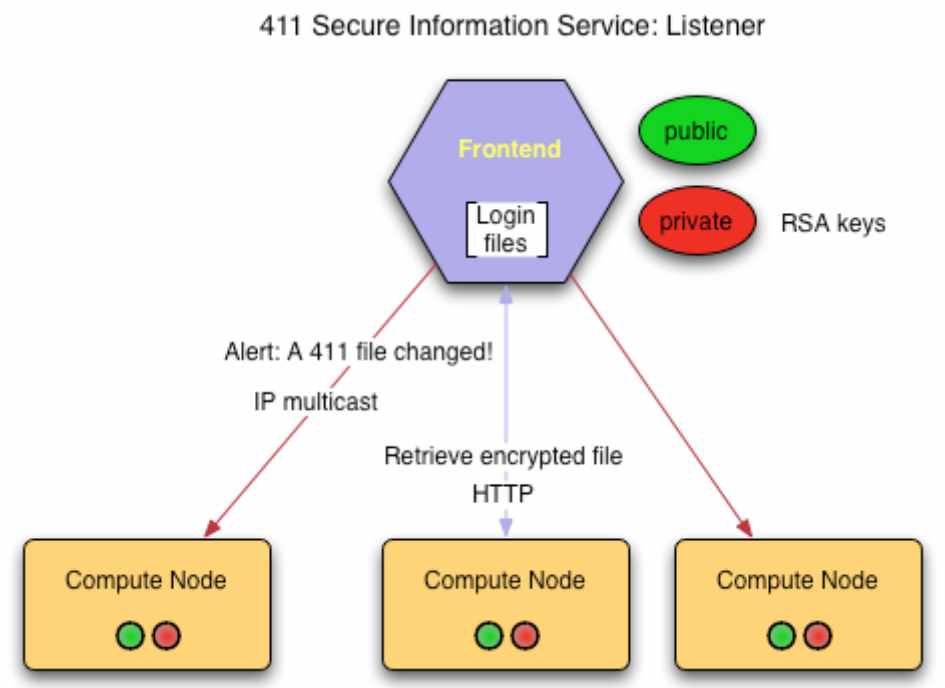


그림 설명: 411 listener 구조. Frontend 가 login 파일을 변경하면 411 makefile 은 이 변경사항을 각 계산 노드에 알리기 위해 broadcast 로 전송한다. 이것을 수령한 노드들은 HTTP 를 통해 frontend 노드로부터 해당 파일을 가져 온다

master 서버에 대한 request "flooding"을 막기 위해 listener 는 클러스터의 크기에 대한 추정치를 가지고 있어야 하며, 이를 사용하여 차후에 request 를 요청할 임의의 숫자를 세팅 한다. 이렇게 함으로서, client 는 변화된 파일에 대해 즉각적인 request 를 하지 않고 그것을 요청하기까지 일정한 시간을 기다리게 된다.

411 는 분산데이터베이스와 비슷하지만, NIS 처럼 중앙 집중 서비스가 아니다. 확장성으로 인해 411 은 새로운 파일을 즉시 제공하지 않는다. 클러스터 크기에 따라 411 makefile 과 모든 노드에서 변경된 파일을 전송받을때까지 대기시기간이 있다. 많은 노드를 가진 클러스터에서 매우 큰 password 파일은 모든 노드를 동기화시키는데 일분 정도까지 걸릴 수 있다.

4.2.2.2. Poller

각 노드에서 411 listener agent 외에도, 파일 변경 유무에 상관없이 일정 시간마다 파 frontend 에서 전송된 모든 메시지를 받는 또다른 agent 가 있다.
각 노드에 있는 polling 간격은 기본적으로 5 시간으로 설정된다. 이를 변경하려면 각 노드에 있는 411 설정 파일의 "interval" 옵션을 설정해야 한다.

```
/etc/411.conf:
...
<interval sec="300">
...
```

네트워크에서 대규모 데이터 송수신을 피하기 위해서 polling 간격을 자동적으로 무작위로 이용한다.

poller 는 greceptor 의 이벤트 모듈에 의해 구현되었으며, 이 greceptor 데몬의 작동에 의존한다. 411 poller 는 자신의 local 디스크에 있는 설정파일을 읽고, master 서버가 누구인지 파악한다. XML 로 쓰여진 이 설정 파일은 411 listener 에 의해 자동적으로 만들어진다.


411 poller 는 client node 에서 root 로 동작하기 때문에 master 서버에 있는 411 http 디렉토리는 다른 권한 부여를 피하기 위해 root 만이 기록할 수 있는 권한이 있다. Rocks 에서는 이를 기본 값으로 이용한다.

4.2.3. 보안

411 보안 관련 동작 방식에 대한 세부 사항은 부록에 있는 411 논문을 참조하라.

4.2.4. 411 Groups

Rocks 3.3.0 부터 411 은 클러스터의 일부 노드그룹(subset)에 대해 메시지를 전달할 기능을 갖게 되었다. 411 groups 라는 이름으로 불리는 이 기능은 사용자가 노드 타입에 따라 다른 파일들을 분배할 수 있도록 해준다. 이 group mechanism 은 각 계산노드의 local 411 설정 파일에 정의 되어 있는 group 이름에 의존한다.

411 에는 개별 그룹키는 있질 않다. Group 매카니즘은 단지 편리함을 위해 존재하는 것이며, 강제적으로 강화된 보안을 적용되지 않았다. 특히, 노드는 그 그룹 멤버가 아닌 외부 그룹의 메시지를 확인할 수 있다.

Group 이름은 multi-level 을 따르며, "파일의 경로"와 비슷한 구조를 가진다. 기본적으로, 모든 노드는 "/" 그룹(일반적인 411 그룹)의 멤버이며, 또한 "/Membership" 그룹의 멤버이다. 여기에서 membership 은 "Compute" 혹은 "NAS"와 같은 frontend 의 database 에 저장된 node 가 기본적으로 속한 membership 을 의미한다.

/var/411/Group.mk 라 불리는 특별한 Makefile 은 411 group 관리와 설정하는데 유용하다. 이 파일을 편집한 후에 특정 플래그를 입력한 다음에는 다음과 같은 명령을 실행한다.

```
#make -C /var/411 groups
#make -C /var/411
```

활성화하려면 그 411 그룹에 makefile 를 수행한다.

기본적으로 노드는 노드의 membership 의 이름과 동일한 그룹의 멤버가 된다. 예를 들어 계산 노드들은 자동으로 "Compute" 그룹의 멤버가 된다. 아래의 411.conf 예제 파일은 여러 개의 그룹에 속하도록 설정된 예이다.

```
<!-- Configuration file for the 411 Information Service -->
<config>
<master url="http://10.1.1.1/411.d/" score="0"/>
<group>/blue</group>
<group>Compute/light</group>
</config>
```

Multi-element 그룹 이름은 간단한 상속 모델을 가진다: 즉 특정 그룹은 보다 일반적인 그룹을 내포한다. 예를 들면, 만일 한 노드가 /compute/light 의 멤버이면, 이 노드는 자동적으로 /compute 그리고 /compute/light 그룹 앞으로 오는 메시지를 수신하게 된다. 반면에 /compute/heavy 그룹 앞으로 오는 메시지는 무시한다. 이 경우에 /compute/light 는 특정 그룹을 의미하며, /compute 는 보다 일반적인 그룹을 의미한다.

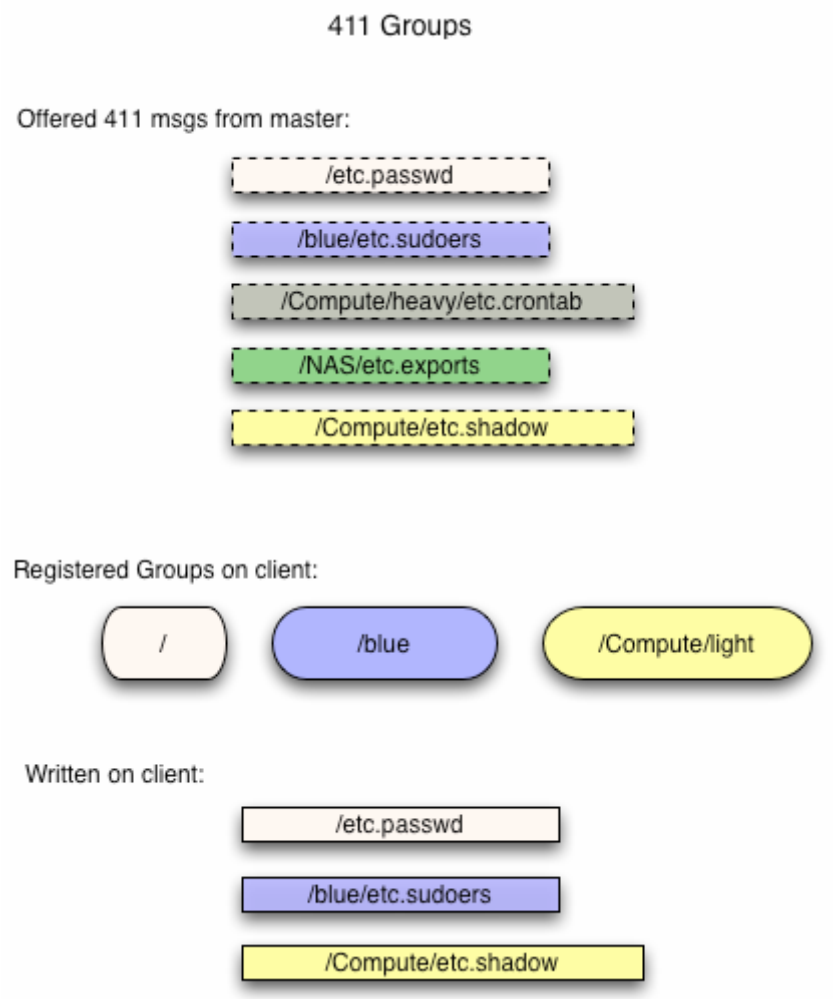


그림 설명: 411 그룹. Master 로부터 보내진 메시지를 필터링하기 위해 Client 는 자신의 local 설정 파일에 등록된 group 이름을 사용한다. 점선으로 둘레가 그려진 메시지는 master 노드에서 새로이 변경된 411 파일을 의미한다. 또한 그림 아래의 일직선으로 둘레가 그려진 메시지는 client 에 의해 수신된 메시지를 나타낸다. /compute/light 는 /compute 그룹을 내포함을 주의할 필요가 있다.

4.2.5. 명령어

4.2.5.1. 411get

```
411get [--all] [--master=url] [--conf] [--pub] [--shared] [--local] [file]
```

암호화된 411 메시지를 가져와서 해독한다. 이 결과 파일을 표준출력으로 보낸다. 특정 파일을 명시하지 않을 경우, 411get 은 이용 가능한 411 메시지 리스트를 출력한다..
다음의 옵션을 사용할 수 있다.

- *--all* 이용 가능한 모든 메시지를 가장 우선순위가 높은 master 서버로부터 가져와서 파일에 업데이트한다. stdout 에 출력하지 않으며, 파일에 덮어쓰기 할 때도 확인 질의를 하지 않는다.
- *--master* 이용할 411 master 서버의 url . 기본은"http://10.1.1.1/411.d/" 또는 "/etc/411.conf"에 주어진 서버 이름. 여기에서 master 의 이름을 명시하면, 설정파일에 등록되어 있는 이름을 override 한다.
- *--file, --local* 는 파일이 현재 디렉토리와 같이 local 에 있음을 말한다. 파일을 가져오기 위해 http 를 사용해서는 안된다. 파일 내용을 역암화화 후 출력한다.
- *--conf* 사용할 설정파일. 기본은 "/etc/411.conf"
- *--pub* 클러스터 public RSA 키의 위치. 기본은 "/etc/security/cluster-public-key.pem".
- *--shared* 클러스터 shared key 의 위치. 기본은 "/etc/411-security/shared.key"

master 서버는 자신의 품질 점수(quality score)는 각 계산노드의 "/etc/411.conf" 파일에 저장된다.

4.2.5.. 411put

```
411put [--411dir=dir] [--urldir=dir] [--see] [--noalert] [--alert=channel]
[--411name] [--pub] [--priv] [--comment=char] [--chroot=dir]
[--chroot-here] [--group=group] file1 file2 ...
```

411 서비스를 통해, 파일을 암호화 하고 publish 한다. 기본적으로 클라이언트에 multicast 메시지를 보내고, 수정된 파일에 대해 alert 메시지를 보낼 것이다.

다음은 이용 가능한 옵션들이다.

- *--chroot=dir* 특정“dir”을 411 파일의 root 디렉토리로 사용한다. 이것은 파일들이 master 노드와 client 노드에서 각각 다른 경로에 위치할 수 있도록 한다.

  ```
  Example:
  411put --chroot=/var/411/groups/compute /var/411/groups/compute/etc/passwd
  ```

  위의 옵션은 계산 노드에 있는 “/var/411/groups/compute/etc/passwd” 파일을 “/etc/passwd”로 이용한다는 의미이다.

- *--chroot-here* 이 옵션은 –chroot=$PWD 와 동일한 의미이며 단순히 편리성을 위해 생성된 옵션이다.

- *--group=name* 이 파일에 대한 411 그룹을 말한다. Client 는 그 그룹에 속하지 않으면 그 그룹에 있는 411 메시지를 무시한다. 클러스터의 하위 부분에 411 파일을 배포할 수 있도록 한다. 이름은 다음과 같이 경로처럼 되어 있다 “Compute/green” 혹은 “/Compute/green”, “a space/yello”와 같이 스페이스가 들어가도 유효한 그룹 이름으로 사용할 수 있다.
- *--comment* 해당 411 파일에서 사용할 주석 처리 기호. 일반적인 설명을 위해 사용한다. 대부분은 “#”을 이용하지만 기본 설정은 없다.
- *--411dir* 암호화된 411 메시지가 위치한 로컬 디렉토리. 기본은 "/etc/411.d/". 해당 디렉토리에 대한 권한 설정을 확인하라.
- *--urldir* 411 메시지를 이용할 수 있는 웹 디렉토리. 기본은 "/411.d/"이다.
- *--see* stdout 으로 암호화된 파일 내용을 보여준다.
- *--noalert* alert 메시지를 생략한다.
- *--alert* UDP multicast, 혹은 unicast 로 이용될 alert 채널을 명시한다. 기본은 ganglia 채널 (239.2.11.71)을 사용한다.
- *--411name* 해당 파일의 411 메시지 이름을 출력한다.
- *--pub* 클러스터 public RSA 키의 위치. 기본은 a 1024 비트 키이며 "/etc/411-security/master.pub"에 위치한다. 이 파일은 0444 퍼미션을 가져야 한다(즉 모든 사용자가 읽기 권한을 가져야 한다).
- *--priv* 클러스터 private RSA 키의 위치. 기본은 1024 비트 키이며, "/etc/411-security/master.key". 에 위치한다. 이 파일은 root 가 소유권을 가져야 하며, 0400 퍼미션을 가져야 한다 (즉 root 만이 읽기 권한을 가져야 한다)
- *--make-shared-key* 새로운 무작위 shared key 를 생성한다. 키는 base64 로 암호화되어 있으며 256 길이이다.

4.3. 도메인 네임 서비스 (DNS)

Lhotse 버전부터, rocks 클러스터는 완전한 기능을 갖춘 DNS 서버를 frontend 에 포함하고 있다. 이 네임 서버는 클러스터내의 각 노드에 대한 네임-> ip 주소 매핑을 관리 제공한다. rocks 의 이전 버전에서는 호스트네임(hostname)은 /etc/hosts 파일의 NIS map 파일을 통해 resolve 되었다

전 기능을 갖춘 name 서비스로의 변환은 우리의 naming 정책에 좀 더 엄격한 기준을 요구한다. 우리는 내부 클러스터의 도메인 이름으로 “local”을 기본적으로 선택했으며, 내부의 이름과 외부의 이름을 엄격히 분리했다.

표준 UNIX naming(그리고 특히 리눅스)의 한 가지 문제는 하나의 머신은 오직 한 개의 이름만을 가져야 한다는 것이다. 이것은 두 개의 네트웍 인터페이스(한 개는 내부 네트웍용, 다른 한 개는 외부 네트웍용)를 갖는 frontend 와 같은 머신의 경우 문제가 된다. Globus 와 같은 외부의 서비스는 frontend 의 실제 이름이 public 이름(즉, FQDN)이어야 한다고 요구하는 반면에, queuing system(PBS 등)과 같은 내부 시스템은 frontend 가 내부의 local name(즉, private name)을 갖기를 선호한다.

Rocks 에서 모든 ".local" 도메인 네임은 private 클러스터 네트웍에 있는 인터페이스에 사용하기로 결정했다. 이것은 모든 노드와 frontend 의 eth0 인터페이스, 그리고 일반적으로 10.x.x.x 범위의 private ip 주소를 사용하는 모든 네임을 포함한다.

Globus 와의 호환성을 위해 frontend 노드는 public 네임을 갖는다. 이것은 frontend 에서 "hostname" 명령을 내리면, ".local"로 끝나는 네임이 아니라, public 네임이 보여짐을 의미한다. 어떤 내부 시스템은 이러한 선택에 의해 좀 더 복잡해질 수도 있다. 그러나 libc 내의 표준 resolver 라이브러리를 사용하는 것들은 문제를 일으키지 않을 것이다.

"insert-ethers"에 의해 추가된 새로운 노드는 자동적으로 local DNS 네임에 추가될 것이다. 노드 네임의 복잡한 리스트를 보기 위해서 다음의 명령을 내리면 된다.

```
$ host -l local
```

4.3.1. DNS 확장하기

Rocks 는 클러스터의 DNS 의 제어 하에서 외부 호스트네임을 추가하는 방법을 제공한다. 외부 호스트들은 일반적으로 site 의 DNS 서버에 의해 서비스를 받을 것이다. 그러나 이용 가능한 외부의 DNS 서버가 존재하지 않을 경우, 관리자는 클러스터 노드가 아닌 일반 서버에 대한 name->IP 매핑을 처리하기 위해 frontend 의 DNS 를 사용하기를 원할 것이다.

DNS 설정은 자동적으로 rocks 의 dbreport 에 의해 생성되며, 관리자가 /var/named 에 있는 표준 zone 파일에 static 설정을 추가할 수는 없다. 대신에, 로컬 name mapping 정보를 다음의 파일에 넣어라.
/var/named/rocks.domain.local
그리고, reverse mapping(IP->name)정보를 다음의 파일에 추가하라.
/var/named/reverse.rocks.doamin.local
이 파일들은 자동적으로 Rocks 의 DNS dbreport 에 의해 포함되며, 다음의 명령을 통해, 업데이트될 수 있다.

```
# insert-ethers --update
```

Rocks dbreport 에 의해 만들어진 표준 rocks.domain 및 reverse.rocks.domain 파일들은 BIND 설정 포맷을 따른다.


외부 호스트는 .local 클러스터 도메인에 이름을 갖게 된다.


로컬 DNS 파일에 에러가 있으면, 전체 클러스터 도메인 naming 이 작동하지 않을 수 있다. 주의해서 진행할 필요가 있다

4.4. 메일

Shasta 버전부터, Rocks 는 클러스터전체의 email 서비스를 위해 Postfix 메일서버를 사용한다.
Rocks 에서는 frontend 에서 모든 클러스터 노드에 대한 메일 릴레이 서비스를 담당한다.
이 의미는 계산 노드는 모두 frontend 로 메일을 전송하며, frontend 가 이를 외부로 전달(forward)한다.


frontend 의 /var/log/mail 파일에 있는 메일의 로그정보를 볼 수 있다. Postfix 는 이 파일에 들어오거나(incoming) 나간(outgoing) 메일의 메시지정보를 기록한다.

5 장. Rocks 설치과정 수정하기(customizing)

5.1. 계산 노드에 패키지 추가하기

추가하고자 하는 패키지를 다음의 디렉토리에 넣어라:

```
/home/install/contrib/4.0.0/arch/RPMS
```

여기에서 *arch* 는 architecture("i386", "ia64", etc)을 의미한다.

현재의 compute.xml 설정파일을 확장하기 위해 새로운 XML 설정파일을 만들어라.

```
# /home/install/site-profiles/3.3.0/nodes
# cp skeleton.xml extend-compute.xml
```

extend-compute.xml 파일 내에 다음의 섹션을 수정하여 패키지 이름을 추가하라.

before:

```
<package> <!-- insert your package name here --> </package>
```

after:

```
<package> your package  </package>
```

extend-compute.xml 파일에 패키지의 full name 이 아니라 base name 만을 입력하는 것이 중요하다. 예를 들면, 추가하고자 하는 패키지의 이름이 XFree86-100dpi-fonts-4.2.0-6.47.i386.rpm 이면 XFree86-100dpi-fonts 만을 입력하면 된다.

```
<package>XFree86-100dpi-fonts</package>
```

만일 추가하고자 하는 여러 개의 패키지가 있다면, 각각의 패키지에 <package> tag 를 사용하면 된다. 예를 들어 100 dpi 폰트와 75 dpi 폰트 패키지를 추가하고자 한다면 다음의 줄을 extend-compute.html 에 추가하면 된다.

```
<package>XFree86-100dpi-fonts</package>
<package>XFree86-75dpi-fonts</package>
```

이제 새로운 rocks 배포판을 만들어야 한다. 아래의 명령은 새로이 추가한 패키지를 /home/install/rocks-dist/에 있는 Redhat 호환 배포판에 결합시킬 것이다.

```
# cd /home/install
# rocks-dist dist
```

이제 계산노드를 재설치 한다.

5.2. 계산노드의 설정 수정하기(customizing)

현재의 compute.xml 설정파일을 확장하기 위해 새로운 XML 설정파일을 만들어라.

```
# cd /home/install/site-profiles/4.0.0/nodes/
```

```
# cp skeleton.xml extend-compute.xml
```

필요하다면, extend-compute.xml 파일 내부에 설정 스크립트를 추가하라. 이 설정 스크립트는 계산노드를 위한 kickstart 설정 파일(이를테면 ks.cfg)의 "post" 섹션에 추가되어 Red Hat installer(Anaconda)에 의해 “post” configuration 단계에서 실행될 것이다. 자세한 내용은 Kickstart howto 를 참조하기 바란다

<post> 와 </post> tag 사이에 bash 스크립트를 입력하라.

```
<post>
        < !-- insert your scripts here -->
</post>
```

이 customized 된 설정 스크립트를 적용하기 위해서는 배포판을 재 빌드한 후 계산노드를 재 설치해야 한다.

```
# cd /home/install
# rocks-dist dist
```

5.3. 추가 이더넷 카드 설정하기

Rocks 는 관리(예를 들면, 재설치), 모니터링(예를 들면, Ganglia), Message Passing(예를 들면, MPICH over ethernet)등을 위해 계산노드, 및 frontend 노드의 첫 번째 이더넷 인터페이스(eth0)를 사용한다. 종종 계산노드는 한 개 이상의 이더넷 인터페이스를 가지고 있을 수 있다. 이 과정은 이들(추가의 네트웍 인터페이스)에 대한 설정 방법을 설명한다. 추가의 이더넷 인터페이스는 frontend 에서 명령 행 유틸리티인 add-extra-nic 을 통해 설정할 수 있다. 이 유틸리티는 한 노드의 여분의 인터페이스에 대한 정보를 추가하기 위해 frontend 의 네트웍 table(네트웍 table 에 대한 자세한 정보는 Rocks Cluster Schema 참조)수정한다. 이 프로그램은 단지 데이터베이스만 수정을 하며, 현재 구동 중인 계산 노드의 실제 설정에 대해서는 아무것도 하지 않는다.

일단 네트웍 table 에서 정보를 갖게 되면, 매 번 재 설치할 때 마다, 추가적인 NIC 에 대한 설정이 이루어진다. 이러한 구조는 노드당 여러 개의 추가적인 이더넷 인터페이스를 지원한다.

● 추가적인 이더넷 인터페이스를 갖고 있는 각 노드에 대해 다음을 실행하라.

```
# add-extra-nic --if=<interface> --ip=<ip address> --netmask=<netmask> --gateway=<gateway> --name=<host name> <compute node>
```

여기에서 각 옵션은 다음을 의미한다.

interface: 이더넷 인터페이스의 이름(e.g., eth1).

ip address: 인터페이스를 위한 IP address (e.g., 192.168.1.1).

netmask: 인터페이스를 위한 network mask (e.g., 255.255.255.0).
gateway: 이 인터페이스의 gateway(eg., 192.168.1.254)
host name: 인터페이스를 위한 Host name (e.g., fast-0-0).
compute node: 이 설정을 적용할 계산 노드의 이름 (e.g., compute-0-0).

예를 들면, 계산 노드 compute-0-0 의 추가적인 이더넷 인터페이스 eth1 에 IP 주소 192.168.1.1, netmask of 255.255.255.0, gateway 는 192.168.1.254, 그리고 새 인터페이스를 위한 hostname 을 fast-0-0 을 주고자 하면,

```
# add-extra-nic --if=eth1 --ip=192.168.1.1 --netmask=255.255.255.0 --gateway=192.168.1.254 --name=fast-0-0 compute-0-0
```

와 같이 명령을 내리면 된다.

- 변경사항을 적용하려면 추가적인 인터페이스를 정의한 노드를 다시 설치한다(shoot-node 이용)

5.4. 계산 노드 디스크 분할하기

5.4.1. 기본적인 디스크 분할

기본적으로 root 파티션에 6 GB, swap 파티션에 1 GB 가 할당되며, root 디스크의 나머지 부분에 파티션 /state/partition1 이 할당된다.

표 5-1. 계산 노드 - 기본적인 Root 디스크 파티션

Partition Name	Size
/	6 GB
swap	1 GB
/state/partition1	루트 디스크의 나머지

이외의 모든 나머지 디스크 드라이브는 각각 하나의 파티션만을 갖게 되며, 그 이름은 /state/partition2, /state/partition3, ... 으로 주어진다.
예를 들면 다음의 표는 3 SCSI 디스크 드라이브를 가진 하나의 계산노드에 대한 기본적인 파티션 방법을 보여준다.

표 5-2. 3 개의 SCSI Drive 를 가진 계산 노드

Device Name	Mountpoint	Size
/dev/sda1	/	6 GB
/dev/sda2	swap	1 GB
/dev/sda3	/state/partition1	루트 디스크의 나머지
/dev/sdb1	/state/partition2	*size of disk*
/dev/sdc1	/state/partition3	*size of disk*


초기 설치 후에도 /state/partitionX 에 있는 모든 데이터는 재설치 시에도 지워지지 않는다

5.4.2. 계산노드의 root 및 swap 디스크 크기의 수정

이 섹션에서는 계산 노드의 root 와 swap 파티션의 크기를 조정하는 간단한 방법에 대해서 설명한다. 만약 여러 계산 파티션을 수정해야 한다면 계산 노드 디스크 파티션 수정 항목을 보기 바란다.

먼저 다음과 같이 extend-auto-partition.xml 파일을 생성한다..

```
# cd /home/install/site-profiles/4.0.0/nodes/
# cp skeleton.xml extend-auto-partition.xml
```

<main> 항목 위에 다음 두 줄을 삽입한다.

```
<var name="Kickstart_PartsizeRoot" val="10000"/>
<var name="Kickstart_PartsizeSwap" val="2000"/>
```

이 설정은 6GB 에 root 파티션을 10GB 로 증가시킬 것이다. 또 Swap 파티션을, 1GB 에서 2GB 로 증가시킬 것이다.

위에서 지정한 내용으로 계산 노드 compute-0-0 을 다시 포맷하려면 먼저 데이터 베이스에서 compute-0-0 에 대한 파티션 정보를 제거해야 한다.

그리고 이 설정을 배포판에 적용하기 위해 다음을 실행하라.

```
#  rocks-partition --list --delete --nodename compute-0-0
```

그런 후 각 계산 노드의 각각의 파티션에 있는 .rocks-release 파일을 제거해야 한다.

여기에 간단한 스크립트를 참조하기 바란다.

```
for i in `df | awk '{print $6}'`
do
          if [ -f $i/.rocks-release ]
          then
                    rm -f $i/.rocks-release
          fi
done
```

위의 스크립트를 /home/install/sbin/nukeit.sh 로 저장하고 다음을 실행하도록 한다.

```
#  ssh compute-0-0 'sh /home/install/sbin/nukeit.sh'
```

그런 후 각 노드를 다시 설치한다.

```
#  ssh compute-0-0 '/boot/kickstart/cluster-kickstart'
```

5.4.3. 계산노드의 디스크 파티션 수정

새 XML 설정 파일은 현재의 auto-partition.xml 설정을 대신한다.

```
# # cd /home/install/site-profiles/4.0.0/nodes/
# cp skeleton.xml replace-auto-partition.xml
```

replace-auto-partion.xml 내부에 다음을 추가하라

```
<main>
        <part> / --size 8000 --ondisk hda </part>
        <part> swap --size 1000 --ondisk hda </part>
        <part> /mydata --size 1 --grow --ondisk hda </part>
</main>
```

이것은 루트 파티션으로 8 GB, 스왑 파티션으로 1 GB, 그리고 이 드라이브의 나머지 부분에 /mydata 을 할당한다. 계산 노드의 나머지 디스크 드라이브에 대해서는 --ondisk 뒤에 적절한 변수를 주어 공간을 할당할 수 있다. 위의 예에서 <part>와 </part> tag 을 제외한 나머지 문법은 모두 Redhat 의 kickstart 의 문법을 따른다. 좀 더 자세한 내용은 RedHat Linux 4:Official Red Hat Linux Customization Guide 의 "part"" 키워드에 대한 설명을 참조하기 바란다.

 사용자가 임의로 정한 mountpoint 의 이름(예를 들면 /mydata)은 문자수가 15 개를 넘으면 안 된다.

만일 software-RAID 를 계산노드에서 이용하고자 한다면, replace-auto-partion.xml 을 다음과 같이 수정하라.

```
<main>
        <part> / --size 8000 --ondisk hda </part>
        <part> swap --size 1000 --ondisk hda </part>

        <part> raid.00 --size=10000 --ondisk hda </part>
        <part> raid.01 --size=10000 --ondisk hdb </part>

        <raid> /mydata --level=1 --device=md0 raid.00 raid.01 </raid>
</main>
```

만약 사용자가 지정한 파티션 scheme 가 설치중인 계산 노드의 현재 설정이 아니라면 계산 노드에 있는 모든 파티션은 삭제될 것이고, 사용자가 지정한 파티션 scheme 가 그 노드에 강제적으로 적용될 것이다.

만약 사용자가 지정한 파티션 scheme 가 현재 설치중인 계산 노드의 설정으로 되어 있다면 그 노드에 있는 모든 계산 노드는 그대로 남아있을 것이며, 단지 root 파티션만 재포맷될 것이다.

만일 파티션 scheme 을 변경하면, 모든 파티션은 제거되고, 다시 포맷될 것이다. .

그리고 이 설정을 배포판에 적용하기 위해 다음을 실행하라.

```
# cd /home/install
# rocks-dist dist
```

위에서 지정한 파티션으로 계산 노드 compute-0-0 를 다시 포맷하려면 먼저 데이터베이스에 있는 compute-0-0 의 파티션 정보를 먼저 제거해야 한다.

```
# rocks-partition --list --delete --nodename compute-0-0
```

그런 후 계산 노드에 있는 각 디스크의 첫번째 파티션에 있는 .rocks-release 파일을 제거해야 한다. 예제 스크립트는 다음과 같다.

```
 for i in `df | awk '{print $6}'`
do
          if [ -f $i/.rocks-release ]
          then
                    rm -f $i/.rocks-release
          fi
done
```

위 파일을 /home/install/sbin/nukeit.sh 로 저장한후 다음을 실행한다.

```
# ssh compute-0-0 'sh /home/install/sbin/nukeit.sh'
```

그리고 node 를 다시 설치한다.

```
# ssh compute-0-0 '/boot/kickstart/cluster-kickstart'
```

5.4.4. 계산노드의 파티셔닝을 기본 설정으로 되돌리기

이 과정은 현재 계산노드의 디스크 드라이브의 상태에 관계없이 계산 노드의 파티셔닝을 원래의 Rocks 파티셔닝으로 되돌리는 방법을 다룬다. Rocks 계산노드의 기본적인 파티셔닝에 대한 정보를 알기 위해서는 “기본적인 디스크 분할”을 참조하기 바란다.

현재의 auto-partition.xml 을 대체할 새로운 XML 설정 파일을 만든다.

```
# cd /home/install/site-profiles/4.0.0/nodes/
# cp skeleton.xml replace-auto-partition.xml
```

replace-auto-partiton.xml 에 다음을 추가한다.

```
<<main>
          <part> force-default </part>
</main>
```

그런 후에 다음을 실행하여 배포판에 설정 내용을 적용한다.

```
# cd /home/install
# rocks-dist dist
```

위에서 지정한 내용으로 계산 노드 compute-0-0 을 다시 포맷하기 위해서는 먼저 데이터베이스에 있는 compute-0-0 에 대한 파티션 정보를 제거해야 한다.

```
#  rocks-partition --list --delete --nodename compute-0-0
```

그런 후 계산 노드에 있는 각 디스크의 첫번째 파티션에 있는 .rocks-release 파일을 제거해야 한다. 예제 스크립트는 다음과 같다.

모든 계산노드를 원래의 기본적인 파티셔닝으로 바꾸기 위해서는 다음을 실행하라.

```
 for i in `df | awk '{print $6}'`
do
          if [ -f $i/.rocks-release ]
          then
                    rm -f $i/.rocks-release
          fi
done
```

위의 스크립트를 /home/install/sbin/nukeit.sh 로 저장한후 실행하도록 한다.

```
#
```

모드 계산노드를 원래의 기본적인 파티셔닝으로 되돌린 후에, 각 계산노드에서 root 파티션 이외의 파티션에 속한 데이터의 내용이 지워지지 않게 하기 위해서는 다음을 실행하라.

```
# ssh compute-0-0 'sh /home/install/sbin/nukeit.sh'
```

그런 후에 노드를 다시 설치한다.

```
# ssh compute-0-0 '/boot/kickstart/cluster-kickstart'
```

모든 계산 노드를 기본 파티션 설정으로 되돌린 후에는 Rocks 가 기존의 root 파티션 외의 데이터를 보존할 수 있도록 replace-auto-partition.xml 을 제거하도록 한다.

```
# rm /home/install/site-profiles/4.0.0/nodes/replace-auto-partition.xml
```

모든 배포판에 업데이트를 적용하려면 다음을 실행한다.

```
# cd /home/install
# rocks-dist dist
```

5.5. 맞춤(custom) 커널 RPM 만들기

5.5.1. kernel.org 로부터 다운로드 한 커널 소스로 커널 RPM

- Frontend 에 대해서는 Rocks 소스 코드를 확인하라. 읽기 전용으로 CVS 접근을 참고하라.
- 디렉토리를 변경한다..

```
# cd rocks/src/roll/kernel/src/kernel.org
```

- kernel.org 로부터 tarball 형태의 커널 소스를 다운로드 하라. 예를 들면 다음과 같다.

```
# wget http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.9.tar.gz
```

- kernel 의 “config” 파일을 생성하고 이를 config-<version>에 넣도록 한다.

<version> 번호는 반드시 kernel 소스와 일치해야 한다. 예를 들자면 만약 linux-2.6.9.tar.gz 을 다운로드 하였다면 config 이름은 반드시 config-2.6.9 라 해야 한다.

- Version.mk 를 갱신한다.

version.mk 파일은 다음과 같은 내용을 가지고 있다.

```
<NAME           = kernel
RELEASE         = 1
VERSION         = 2.6.9
SMP             = 1
```

VERSION 값은 반드시 다운로드한 커널 버전(예를 들어, 2.6.9)와 동일해야 한다.

만약 단일 프로세서 커널을 생성하고자 한다면 다음과 같이 설정한다.

```
SMP             = 0
```

혹은 다중-프로세서 커널(예. SMP)를 생성하고자 한다면, 다음과 같이 설정한다.

```
SMP             = 1
```

- 커널을 빌딩한다.

```
#make rpm
```

- 생성된 RPMs 를 현재 배포판으로 복사한다.

```
# cp ../../RPMS/<arch>/kernel*rpm /home/install/contrib/4.0.0/<arch>/RPMS/
```

- 여기에서 <arch>는 i386,x86_64 또는 ia64 가 된다.
- 다음 단계는 Rocks 4.0.0 에만 적용된다.

 Rocks 버전이 4.0.0 이라면 다음 단계가 필요하다.

```
#cd /home/install/site-profiles/4.0.0/nodes
# wget http://www.rocksclusters.org/ftp-site/beta/4.1.0/kernel.org/extend-compute.xml
```

- 배포판을 다시 생성한다.

```
# cd /home/install
# rocks-dist dist
```

- 계산 노드를 다시 설치하여 새로운 커널을 시험한다.

```
# shoot-node compute-0-0
```

- 만약 커널이 안정적으로 동작하면 새로운 커널을 적용하고자 하는 모든 계산 노드를 다시 설치한다.

5.6. 계산노드에서 RSH 사용하기

기본적으로 Rocks 는 rsh 명령이나 계산 노드에 직접적인 로그인을 허용하지 않는다. 대신에 Rocks 는 ssh 를 이용한다. 하지만 ssh 가 rsh 과 정확히 똑같은 역할을 할 수 없는 경우가 있을 수 있으며, 또한 사용자가 자신의 응용프로그램을 rsh 에서 ssh 으로 변경할 수 없는 경우도 있을 수 있다. 만일 당신이 이런 경우 중 하나에 해당된다면, 클러스터에 rsh 을 사용하기를 원할 것이다.

 클러스터에 rsh 를 사용하는 것은 보안에 중대한 영향을 줄 수 있다. Rsh 를 내부 네트워크에서만 사용한다고 하더라도 이게 ssh 만큼 보안성이 보장되는 것은 아니다.

rsh 을 가능하게 하는 위해 default 로 주어진 kickstart 그래프를 수정하면 된다. 먼저 아래와 같이 rsh.xml 을 customization 디렉토리에 복사한다.

```
#  cp /home/install/rocks-dist/lan/arch/build/graphs/default/base-rsh.xml W
          /home/install/site-profiles/4.0.0/graphs/default/
```

여기에서 *arch* 는 사용자의 시스템의 architecture 이다("i386", "x86_64" 혹은 "ia64" 등)

이제 /home/install/site-profiles/4.0.0/graphs/default/base-rsh.xml 를 다음과 같이 변경하라.

```
<!-- Uncomment to enable RSH on your cluster

        <edge from="slave-node">
                <to>xinetd</to>
                <to>rsh</to>
        </edge>

        -->
```

이 블록의 주석을 제거하면 된다. 이것은 "slave 노드 클래스"를 사용하는 모든 장비 타입(계산 노드, NAS 노드 등)에서 (private 네트웍의 모든 호스트를 "trust"하는) rsh 서비스를 이용할 수 있도록 해준다. 주석을 제거하면 다음과 같을 것이다.

```
<edge from="slave-node">
                <to>xinetd</to>
                <to>rsh</to>
        </edge>
```

다음 단계는 이 변화를 적용하기 위해서 배포판을 재 빌드하는 것이다.

```
# cd /home/install
# rocks-dist dist
```

이제 계산 노드를 다시 설치한다.

5.7. Ganglia 모니터 수정하기

5.7.1. Ganglia 대몬 최적화 하기

최상의 성능과 확장성을 갖게 하기 위해서는 각 계산노드의 ganglia gmond 대몬은 “deaf”모드로 실행되어야 한다 .즉 이것은 각 계산노드가 frontend 에 자신의 ganglia 데이터를 보고하지만, 다른 계산노드로부터 오는 ganglia 정보는 무시하도록 한다(multicast 를 사용하는 gmond 는 각 계산노드가 다른 노드의 ganglia 정보도 수집할 수 있도록 만들어져 있다). 이와 같은 설정은 각 계산노드의 자원을 절약시키는 효과를 준다.

“deaf” 모드로 각 계산노드의 gmond 를 실행하는 것은 각 계산노드로부터 클러스터 전체의 ganglia 정보를 얻을 수 없음을 의미한다. 이것은 성능 분석을 위해 ganglia 데이터를 사용하거나, 고장 감내(fault tolerant)기능을 ganglia 가 같도록 할 경우 문제가 될 수 있다. 만일 계산노드에서 ganglia 의 모든 기능을 사용하도록 하고자 한다면 다음의 설명을 참조하라.

Ganglia 대몬은 Matterhorn Rocks 3.1.0 버전부터 “deaf” 모드를 기본적으로 사용하도록 변경되었다.

- Replace-ganglia-client.xml 이라는 이름의 새로운 xml 노드 파일을 추가하라.

이 파일에 다음의 내용을 추가하라.

```
<?xml version="1.0" standalone="no"?>
<kickstart>
  <description>
  UCB's Ganglia Monitor system for client nodes in the
  cluster.
  </description>
<post>
/sbin/chkconfig --del gmetad
</post>
</kickstart>
```

- 각 계산노드를 재 설치하라. 이제 ganglia 의 모든 모니터링 기능을 사용할 수 있다. 위 과정을 통해 각 계산노드는 frontend 와 동일한 수준의 모니터링 기능을 갖게 된다.

5.8. 새로운 Appliance Type 을 클러스터에 추가하기

이 과정은 새로운 장비 타입(appliance type)을 클러스터에 추가하기 위한 과정을 설명한다. 이것은 일단의 계산노드들의 그룹이 나머지 다른 계산노드들과 다른 방식(behavior)으로 작동하기를 원할 때 유용하게 사용될 수 있다. 예를 들면, cabinet 1 의 모든 계산노드가 다른 cabinet 의 계산노드들과 다르게 설정되게 하고자 할 때 쓰일 수 있다.

시작하기 전에 먼저 설정 그래프(configuration graph)를 만드는 Rocks XML 프레임워크(framework)에 대해 알아두는 것이 편할 수 있다. 이에 대한 자세한 내용은 Reference guide 를 참조하기 바란다.

먼저 새로운 XML node 파일을 만들어야 한다. 이 파일은 설정 스크립트 그리고/또는 각 장비 타입에 적용될 패키지의 리스트에 대한 정보를 포함한다. 이 노드파일을 my-compute.xml 이라고 하자. 이 파일은 /home/install/site-profiles/4.0.0/ 디렉토리에 만들어져야 한다. 아래는 이 파일의 내용이다.

```
<?xml version="1.0" standalone="no"?>

<kickstart>

<description>
My specialized compute node
</description>



<changelog>
</changelog>

<post>

<file name="/etc/motd" mode="append">
My Compute Appliance
</file>

</post>

</kickstart>
```

이제 위의 파일을 기존의 XML 설정 그래프에 링크시켜야 한다. 여기에서는 간단히 위의 XML 노드가 기존의 compute.xml 노드를 가리키도록 할 것이다. 객체 지향의 관점에서 위의 my-compute.xml 은 기존의 compute.xml 의 모든 기능을 상속하며, 이의 기능을 확장한다.

my-compute.xml 을 compute.xml 롤 링크시키기 위해서/home/install/site-profiles/4.0.0/graphs/default 디렉토리에 my-appliance.xml 파일을 만들어라 이것의 내용은 다음과 같다.

```
<?xml version="1.0" standalone="no"?>

<graph>

<description>
</description>

<changelog>
</changelog>

<edge from="my-compute">
        <to>compute</to>
</edge>

<order gen="kgen" head="TAIL">
        <tail>my-compute</tail>
</order>

</graph>
```

위 변경 내용을 적용하기 위해서는 다음의 명령을 실행하라.

```
# cd /home/install
# rocks-dist dist
```

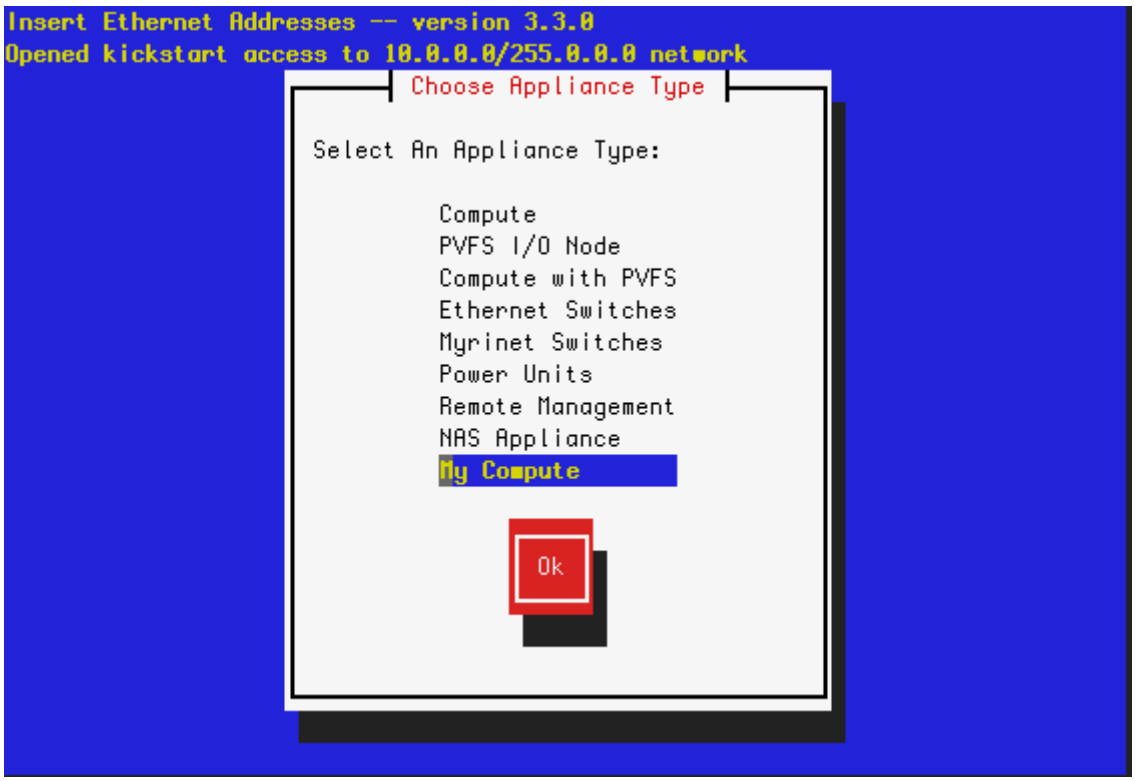
이제 이렇게 만들어진 장비 타입을 Rocks MySQL 의 데이터베이스에 추가해야 한다. 이것은 다음 명령어를 사용하면 된다.

```
# add-new-appliance --appliance-name "My Compute" --xml-config-file-name my-compute
```

이제 기존의 계산노드의 타입을 변경할 수 있다. 여기서는 insert-ethers 를 사용한다. 먼저 insert-ethers 를 compute-0-0 으로 대체하도록 한다.

```
# insert-ethers --replace compute-0-0
```

이것은 아래의 스크린을 보여줄 것이다.



“My Compute”를 선택한 후 “OK”를 클릭하라. 이것은 Rocks 데이터베이스에서 compute-0-0 의 정보를 지우고, DHCP 요청을 하는 노드가 있을 경우, 이것을 my-compute-0-0 로 설치할 준비를 하게 한다. 따라서 아래와 같이 compute-0-0 를 재설치 모드로 들어가게 하면 된다.

```
# ssh-add
# shoot-node compute-0-0
```

이제 insert-ethers 는 my-compute-0-0 를 발견할 경우 이를 보고할 것이다. 그리고 이 노드에 대한 설치가 끝나면 my-appliance 타입으로 설정이 될 것이다. 해당 노드에 로그인하여 이를 확인할 수 있다.

```
# ssh my-compute-0-0
```

이와 같이 사용자가 정의한 appliance 에 대해 insert-ethers 명령어를 사용하여 임의의 노드에 적용할 수 있다.

6 장. 다운로드

6.1. ISO 이미지 및 RPMS

Rocks 소프트웨어는 http://www.rocksclusters.org/Rocks/motd-archives/000064.html 에서 다운 받을 수 있으며 K*Rocks는 http://rocks.cluster.or.kr 에서 다운 받을 수 있다.

6.2. CVS 를 통한 Rocks 소스 트리 억세스 하기

Anonymous read-only access 가 다음 두 가지 형태로 제공된다:

- ViewCVS 를 이용한 Browsing
- CVS pserver 사용하기.

6.2.1. Read-Only 모드로 CVS 사용하기

소스를 다운 받기 위해서는 먼저 다음과 같이 로그인을 하여야 한다.

```
$ cvs -d:pserver:anonymous@cvs.rocksclusters.org:/home/cvs/CVSROOT login
```

여기서 비밀번호를 요구할 것이다. 하지만 비밀번호를 필요치 않으며, 단지 엔터만을 치도록 한다.

익명으로 로그인 한 후에는 다음과 같이 소스 트리를 확인하도록 한다.

```
$ cvs -d:pserver:anonymous@cvs.rocksclusters.org:/home/cvs/CVSROOT checkout rocks
```

초기화 확인(checkout) 후에는 이 rocks 디렉토리로 들어간 후 cvs 명령일 실행하되 -d 옵션을 빼도록 한다. 예를 들면 다음과 같다.

```
$ cvs update
```

6.2.2. Read-Write 권한으로 CVS 사용하기

ssh 버전 1 의 public 키의 복사본(예를 들면, ~/.ssh/identity.pub)과 왜 쓰기 권한이 필요한가에 대한 간략한 설명을 담은 메일을 cvs@rocksclusters.org 로 보내라. 만일 당신의 컴퓨터가 ssh 키를 갖고 있지 않다면, 다음의 명령을 실행하여 ssh 버전 1 의 키를 만들어라

ssh-keygen -t rsa1

일단 rocks 팀은 당신의 public 키를 받으면, 당신을 위한 계정을 만들고, 어떻게 access 해야 하는가에 대해 간략한 설명을 제공할 것이다.

7 장. 자주 받는 질문(FAQ)

7.1. 설치과정

7.1.1. Insert-ethers 에서 새로운 계산 노드가 보이질 않는다. 물론 frontend 에서 계산 노드의 DHCP 메시지 또한 볼수가 없다. 무엇이 문제인가?

계산 노드와 frontend 에 연결한 네트워크 스위치를 bypass 시도를 해보도록 한다. 스위치는 알려져있지 않는 IP 주소의 broadcast 메시지를 무시하도록 설정되어 있을 수 있다. 따라서 노드의 DHCP 메시지를 무시는 것이다. 스위치의 문제라면 다음과 같이 확인하도록 한다.

1. 크로스오버 케이블(또는 기가빗 이더넷을 사용하면 일반 케이블)을 단일 계산 노드와 frontend 의 'eth0' 인터페이스와 연결한다.
2. 계산 노드를 일반적으로 설치한다(계산 노드 설치 항목 참고). frontend 에서 노드에서 전송하는 DHCP 메시지를 확인할 수 있어야 한다.

7.1.2 계산 노드를 설치할 때, Rocks CD 로부터 부팅을 시켰다. 하지만 계산노드에 모니터를 연결하면, 다음과 같은 에러메시지가 나타난다. 'Error opening kickstart file /tmp/ks.cfg. No such file or directory' 무엇이 잘못되었는가?

계산 노드의 kickstart 가 제대로 작동하기 위해서는 다음의 서비스가 frontend 에서 실행 중이어야 한다.

1. dhcpd
2. httpd
3. mysqld
4. autofs

httpd 와 mysqld 가 실행 중인지 다음과 같이 확인한다.

```
# ps auwx | grep httpd
# ps auwx | grep mysqld
```

만일 하나라도 실행 중이 아니면, 다음과 같이 다시 시작시켜라.

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

그리고/또는

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

autofs 서비스는 'automount'라고도 불린다. 실행 중인지 확인한다.

```
# ps auwx | grep automount
```

실행 중이 아니면 재 시작시킨다.

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

마지막으로, 다음과 같이 Rocks 설치 관련 프로그램이 작동하는지 확인한다.

```
# cd /home/install
# ./kickstart.cgi --client="compute-0-0"
```

이것은 하나의 kickstart 파일을 output 으로 보여준다.
kickstart.cgi 에 어떤 에러가 있는지 다음과 같은 명령으로 확인한다

```
# ./kickstart.cgi --client="compute-0-0" > /dev/null
```

7.1.3. Rolls 를 성공적으로 설치했다. 그러나 머신이 reboot 후 마지막 stage 가 진행되는 동안 system 이 “error: *GRUB* Loading Stage2....” 메시지를 출력하면서 hang 에 걸려 버린다. 무엇이 잘못되었는가?

이것은 테스트 환경에서 간헐적으로 나타났던 문제이다. Grub 설치 프로그램이 알 수 없는 이유로 정상적으로 실행되지 않았다는 것을 빼면 설치는 제대로 되었다,

이 경우 해결방법은 다음과 같다.

- Rocks Base CD 를 frontend 에 넣고 부팅시켜라
- Boot prompt 에서 다음을 입력하라.

```
frontend rescue
```

- 화면이 나타날 것이다. 화면에서 “continue(계속)”을 선택하라
- 쉘 프롬프트가 나타나면, 다음의 명령을 실행하라

```
# chroot /mnt/sysimage
```

● 다음과 같이 grub 설치 프로그램을 실행하라.

```
# /sbin/grub-install `awk -F= '/^#boot/ { print $2 }' /boot/grub/grub.conf`
```

이것의 다음과 같은 내용을 출력할 것이다.

```
Installation finished. No error reported.
This is the contents of the device map /boot/grub/device.map.
Check if this is correct or not. If any of the lines is incorrect,
fi x it and re-run the script `grub-install'.

# this device map was generated by anaconda
(fd0)     /dev/fd0
(hd0)     /dev/hda
```

● chroot 환경을 빠져 나온다.

```
# exit
```

● frontend 를 리부팅한다.

● CD 드라이브에서 CD 를 제거하면 frontend 는 정상적으로 부팅되어야 한다.

7.1.4. 계산노드를 설치하고자 할 때, 계산노드에서 "Can't mount /tmp. Please press OK to restart"와 같은 에러 메시지가 보인다. 어떻게 해야 하는가?

대부분의 경우, 이 상황은 계산노드의 디스크 드라이브의 크기 때문에 발생한다. Rocks 의 설치과정은 이전에 Rocks 가 계산노드에 설치된 적이 없으면, 계산노드의 디스크를 포맷한다. 위의 에러를 없애기 위해서는 Rocks 가 디스크 드라이브의 파티션을 나누는 방법에 수정을 가해야 한다. 앞의 “파티션 나누기”에 대한 설명을 참조하라.

7.1.5. 계산노드는 CD 드라이브를 갖고 있지 않으며, 계산노드의 네트워크 카드는 PXE boot 도 지원하지 않는다. 하지만 floppy 드라이브는 가지고 있다. 이 경우 어떻게 계산노드에 설치를 할 수 있는가?

먼저 PXE 를 emulate 하는 boot floppy 를 만들어야 한다. 이 방법은 다음의 웹페이지에 자세히 나와있다.

ROM-o-matic.net

이 웹사이트에서 가장 최신의 안정버전을 선택하라. (이 글이 쓰여지는 시점에서는 5.4.0 이 가장 최신의 버전이다)

항목 1 에서 장치 드라이버를 선택하라. 항목 2 는 기본 사항을 그대로 이용하도록 한다(Floppy bootable ROM image). 그런 후 항목 4 에 있는 "Get ROM"를 클릭한다.

dd 명령을 이용하여 다운 로드한 플로피 이미지를 플로피 디스켓에 복사한다. 예를 들면 다음과 같다

```
# dd if=eb-5.4.0-pcnet32.zdsk of=/dev/fd0
```

이제 insert-ethers 명령을 frontend 에서 실행하고, floppy 를 이용해서 계산노드를 부팅시켜라.

7.2. 설정

7.2.1. 클러스터에서 하나의 계산노드를 어떻게 제거하는가?

아래의 명령은 직관적이지 않지만 작동은 한다.

frontend 에서 다음의 명령을 실행한다:

```
# insert-ethers --remove="[your compute node name]"
```

만일 계산노드의 이름이 *compute-0-1* 이면 다음과 같이 실행한다.

```
# insert-ethers --remove="compute-0-1"
```

7.2.2. frontend 에서 왜 startx 가 작동하지 않는가?

startx 를 시작하기 위해서는 비디오 카드에 대한 XFree86 설정을 해주어야 한다. 이것은 'system-config-display' 프로그램을 이용하여 할 수 있다. 만약 비디오카드에 대해 아는 바가 전혀 없으면, 비디오 램으로 "4MB" 그리고, "16 bit color 800x600 을 선택하라. 이 비디오 모드는 어떤 종류의 최신 VGA 카드에 대해서도 작동한다.

7.2.3. Dell 의 Powerconnect 5224 스위치를 사용하고 있는데, 계산노드를 설치할 수가 없다. 어떻게 해야 하는가?

Dell powerconnect 5224 스위치를 설정하는 방법은 다음과 같다.

모든 포트에 대해 다음과 같이 “edge port” flag 을 세팅해야 한다(어떤 Dell 스위치는 fast link 라는 이름을 쓴다).
먼저 스위치의 IP 주소를 설정해야 한다.

- 스위치에 직렬 케이블을 연결한다.
- 직렬 케이블로 연결된 스위치에 접속한다.
  username/password 는 admin/admin 이다.
- 스위치에 IP 주소를 설정한다.

```
# config
# interface vlan 1
# ip address 10.1.2.3 255.0.0.0
```

- 이제 이더넷을 통해 스위치에 억세스할 수 있어야 한다.
- 스위치와 노트북을 이더넷 케이블로 연결한다.
- 노트북의 IP 주소를 다음과 같이 설정한다.

```
IP: 10.20.30.40
netmask: 255.0.0.0
```

- 노트북에서 웹 브라우저를 열고, 10.1.2.3 주소로 들어간다.
  Username/password 는 admin/admin 이다.
- 모든 포트에 대해서 “edge port”를 설정한다. 이 설정은 메뉴 아이템에서 System->Spanning Tree->Port Setting 을 따라 가면 찾을 수 있다.
  설정을 저장한다.
- 즉 System->Switch->Configuration 메뉴에서 ‘Copy running config to File’에서 파일 이름을 ‘rocks.cfg’로 저장하고, ‘Start-Up Configuration File’에서 이 ‘rocks.cfg’ 파일이 보이도록 해야 한다.

7.2.4. 미리넷 네트웍이 제대로 작동하지 않는 것처럼 보인다. 어떻게 debug 해야 하는 가?

우리는 미리넷을 debug 하기 위해, TOP500 수퍼컴퓨터 리스트에서 순위를 정할 때 사용하는 High-Performance Linpack (HPL)을 사용한다. HPL 은 기본적으로 모든 계산노드에 설치되어 있다.

계산노드에서 HPL 을 실행하기 위해서 Interactive Mode 를 참조하라
먼저 compute-0-0 와 compute-0-1 에 대해서 테스트를 진행하라. 정상적으로 결과값이 얻어지면, compute-0-2 and compute-0-3 까지 추가하여 HPL 테스트를 진행하라. 이와 같은 방법으로 노드 수를 늘려가면서 테스트를 진행하라.

정상적으로 작동하지 않는 것처럼 보이는 계산노드가 발견되면, 해야 할 첫 번째 일은 Myrinet map 을 확인하는 것이다(이것은 해당 계산노드로부터 미리넷으로 연결된 모든 계산노드로 가는 routes 정보를 가지고 있다.

해당 계산노드에 로그인하여 다음을 실행하여 map 을 조사하라.

```
$ /usr/sbin/gm_board_info
```

이것은 다음과 같은 정보를 보여줄 것이다

```
GM build ID is "1.5_Linux @compute-0-1 Fri Apr 5 21:08:29 GMT 2002."
Board number 0:
lanai_clockval = 0x082082a0
lanai_cpu_version = 0x0900 (LANai9.0)
lanai_board_id = 00:60:dd:7f:9b:1d
lanai_sram_size = 0x00200000 (2048K bytes)
max_lanai_speed = 134 MHz
product_code = 88
serial_number = 66692
(should be labeled: "M3S-PCI64B-2-66692")
LANai time is 0x1de6ae70147 ticks, or about 15309 minutes since reset.
This is node 86 (compute-0-1) node_type=0
Board has room for 8 ports, 3000 nodes/routes, 32768 cache entries
Port token cnt: send=29, recv=248
Port:    Status   PID
0:       BUSY   12160 (this process [gm_board_info])
2:       BUSY   12552
4:       BUSY   12552
5:       BUSY   12552
6:       BUSY   12552
7:       BUSY   12552
Route table for this node follows:
The mapper 48-bit ID was: 00:60:dd:7f:96:1b
gmID   MAC Address                    gmName                Route
---- ----------------- ------------------------------ --------------------
1       00:60:dd:7f:9a:d4                compute-0-10           b7 b9 89
```

```
2       00:60:dd:7f:9a:d1               compute-1-15            b7 bf 86
3       00:60:dd:7f:9b:15               compute-0-16            b7 81 84
4       00:60:dd:7f:80:ea               compute-1-16            b7 b5 88
5       00:60:dd:7f:9a:ec               compute-0-9             b7 b9 84
6       00:60:dd:7f:96:79               compute-2-13            b7 b8 83
8       00:60:dd:7f:80:d4               compute-1-1             b7 be 83
9       00:60:dd:7f:9b:0c               compute-1-0             b7 be 84
```

이제, 정상적으로 작동하는 것으로 보이는 계산노드에 로그인하여 같은 명령 즉, /usr/sbin/gm_board_info 을 실행하라. e*gmID*'s 와 *gmName*'s 가 양 노드에서 똑같이 보이지 않으면, 문제가 있는 미리넷 구성요소가 있다고 생각할 수 있다.
문제가 해결하기 위해 미리넷 구성요소의 교체를 해보라. 즉 아래에 보이는 과정을 시도하라.
1. cable 을 교체해 보라.
2. cable 을 미리넷 스위치의 다른 포트에 연결해 보라.
3. 계산 노드의 미리넷 카드를 교체해 보라.
4. Myricom(주석 3)에 문의해 보라

위의 각 과정이 끝나면, 해당 계산노드에서 mapper 를 재시작하고, 만들어진 map 을 살펴보라(/usr/sbin/gm_board_info). mapper 를 다시 실행하기 위해서는 다음을 실행하라.

```
# /etc/rc.d/init.d/gm-mapper start
```

이 mapper 는 몇 초간 진행된 후, 종료될 것이다. 따라서 gm_board_info 를 실행하기 전에 mapper 의 실행이 종료되기를 기다려라. (즉 ps auwx | grep mapper 을 실행하여 mapper 가 완전히 종료되었는지 확인하라)

7.2.5. 계산 노드의 BIOS 부트 순서가 어떻게 설정되어야 하는가?

이것은 네트웍 부팅(PXE)을 지원하는 머신에만 해당되는 문제이다. 이 경우에 부팅 순서는 cdrom, floppy, hard disk, network 순이어야 한다. 즉 OS 가 하드 드라이브에 설치되어 있지 않고, CD 와 플로피 드라이브가 없는 하드웨어(bare hardware)에서만 첫 번째 boot 가 네트웍 부팅이 될 것이다. 이 PXE 부팅은 마치 Rocks CD 로부터 설치하는 것처럼 계산 노드에 Red Hat 설치 커널을 load 하고 Rocks 를 설치할 것이다.
만일 부팅순서에서 하드 디스크보다 PXE 부트를 앞쪽에 놓으면, 해당 노드는 계속해서 재설치를 반복할 것이다.

7.2.6. frontend 의 새로운 디렉토리를 계산노드에서 /home 의 하위 디렉토리로 억세스 할 수 있게 하기 위해서, 어떻게 export 를 설정해야 하는가?

다음 과정을 진행하라.

- /etc/exports 파일에 export 하고자 하는 디렉토리이름을 추가하라. 예를 들면, 만일 /export/disk1 을 export 하고자 한다면 다음을 /etc/exports 에 추가하라:

```
/export/disk1 10.0.0.0/255.0.0.0(rw)
```

이것은 내부 네트웍내에 있는 노드들에게만 이 디렉토리를 export 할 것이다. (앞의 예에서 내부 네트웍은 10.0.0.0 의 A 클래스 네트웍으로 설정되어 있다)

- NFS 를 재 시작하라:

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

- /etc/auto.home 에 엔트리를 추가하라.

예를 들면, frontend(hostname 은 frontend-0)의 /export/disk1 을 /home/scratch 라는 이름으로 마운트 하고자 하면 다음의 엔트리를 /etc/auto.home 에 추가하면 된다.

```
scratch frontend-0:/export/disk1
```

- 다음의 명령을 통해 411 정보를 수정하라

```
# make -C /var/411
```

이제 어떤 계산노드에건 로그인해서 /home/scratch 디렉토리로 들어가고자 하면(예를 들면 cd /home/scratch), 이 디렉토리는 자동으로 마운트된다.

7.2.7. 계산 노드를 hard reboot(컴퓨터 앞의 power 혹은 reboot 버튼을 이용한 강제 reboot)시키면, 계산노드가 재 설치되는데, 이것을 어떻게 중지시키는가?

계산 노드가 hard reboot 되면, 이 해당 노드는 root 파일시스템을 재 포맷하고 OS 를 재 설치할 것이다. 이와 같은 Rocks 의 특징을 중지시키고자 하면,

- 먼저 frontend 에 로그인하라.
- 그리고 기본값을 override 하기 위해 다음의 파일을 생성하라.

```
# cd /home/install
# cp rocks-dist/lan/arch/build/nodes/auto-kickstart.xml W
site-profiles/4.0.0/nodes/replace-auto-kickstart.xml
```

arch 는 i386, "x86_64" 혹은 ia64 이다.

- site-profiles/4.0.0/nodes/replace-auto-kickstart.xml 파일을 편집한다.
- 다음 줄을 제거한다.

```
<package>rocks-boot-auto<package>
```

- 편집이 끝나면, 배포판을 재 빌드하라

```
# cd /home/install
# rocks-dist dist
```

- 모든 계산노드를 재 설치하라.

만일 계산노드 전체를 재 설치하는 대신 다른 방법을 사용하고 싶다면, 모든 계산노드에 로그인하여 다음의 명령을 실행하면 된다.

```
# /etc/rc.d/init.d/rocks-grub stop
# /sbin/chkconfig --del rocks-grub
```

7.3. 시스템 관리

7.3.1 Ganglia 그래프에 호스트이름이 아닌 IP 주소로 나타난다. 왜 그런가?

클러스터의 DNS 는 종종 Ganglia 가 IP 주소를 이용해 정보를 저장하도록 한다. 이런 상황을 없애기 위해서는 "gmond"와 "gmetad"를 frontend 에서 재 시작 하라. 이것은 또한 ganglia 의 output 에서 죽은 노드의 정보를 제거할 때도 유용하게 쓰일 수 있다.

```
# service gmond restart
# service gmetad restart
```

이 방법은 클러스터내의 노드의 이름을 바꾸거나, 노드를 교체할 때에도 유용하게 쓰일 수 있다.

7.3.4. Ganglia page에서 graph를 볼 수가 없다. 단지 다음과 같은 에러 메시지가 보인다. "There was an error collecting ganglia data (127.0.0.1:8652): XML error: not well-formed (invalid token) at xxx"

이것은 Ganglia 의 gmond 의 XML output 에서 parse 에러가 발생했음을 가리킨다. 이것은 일반적으로 비 XML 문자(특히, &)가 클러스터 이름 혹은 클러스터 소유자의 이름에 들어가 있을 때 발생한다. 노드이름을 포함하여 ganglia 의 XML field 에 이와 같은 비 XML 문자가 들어가 있을 때에도 이와 같은 문제가 발생한다.

Ganglia 의 차후 버전은 이와 같은 문제가 해결되기를 바란다. 만일 클러스터의 이름 등이 부적절한 이름을 가지고 있다면, frontend 의 /etc/gmond.conf 에서 적절히 수정한 후, gmond 를 재 시작해야 한다.

7.3.3 클러스터에서 외부 NIS 서버에 있는 사용자 계정을 어떻게 이용하는가?

여기에 적합한 특별한 방법은 없지만 만약 필요하다면 frontend 에서 외부 NIS 에 있는 사용자 계정을 주기적으로 가지고 올 수 있도록 "ypcat"를 이용하기를 추천한다. 그리고 클러스터 내부에서 이 정보를 배포할 수 있도록 411 시스템에 기본적으로 설정하도록 한다. 다음 cron 스크립트는 frontend 에서 외부 네트워크에 있는 NIS 정보를 추출하는데 이용할 수 있다. 여기에서 생성된 login 파일은 411 을 이용하여 자동으로 각 클러스터 노드에 배포될 것이다. 이 코드는 Minnesota 대학에 있는 Chris Dwan 이 작성하여 주었다.

```
(in /etc/cron.hourly/get-NIS on frontend)

#!/bin/sh
ypcat -k auto.master > /etc/auto.master
ypcat -k auto.home   > /etc/auto.home
ypcat -k auto.net    > /etc/auto.net
ypcat -k auto.web    > /etc/auto.web

ypcat passwd      > /etc/passwd.nis
cat   /etc/passwd.local /etc/passwd.nis > /etc/passwd.combined
cp    /etc/passwd.combined /etc/passwd
```

```
ypcat group        > /etc/group.nis
cat    /etc/group.local /etc/group.nis > /etc/group.combined
cp     /etc/group.combined /etc/group
```


NIS 에서 가지고 온 UID 나 GID 와 기존의 클러스터에 있는 정보가 충돌하는 것을 방지하는 방법은 없다. 따라서 이러한 충돌이 일어나 예상치 못한 동작을 하지 않도록 매우 주의를 기울어야 한다.

7.3.4 계산 노드에 대한 DNS 서버를 별도로 사용할 수 있는가?
Rocks 는 계산 노드에 대한 DNS 서버를 frontend 에서 사용하도록 되어 있지만, 계산 노드가 특정 DNS 서버를 사용할 수 있도록 설정할 수 있다.

상위(forwarder) 서버로 Frontend DNS 서버를 사용하도록 구성하도록 한다. 이렇게 하면 특정 DNS 서버에서 있는 이름을 포함하도록 rocks 서버를 확장하는 것과 동일하다(Matt Wise 가 확인)

1. frontend 에 있는 "/etc/named.conf" 파일을 편집하라
   "options" 에 다음과 같은 줄을 추가한다.

```
        forward first;
        forwarders { (local named server); };
```

2. 예를 들어 상위(forwarders) 도메인 서버가 198.202.75.26 이라면 파일 내용은 다음과 같게 된다.

```
//
// named.conf for Red Hat caching-nameserver
//

options {
        directory "/var/named";
        dump-file "/var/named/data/cache_dump.db";
      statistics-file "/var/named/data/named_stats.txt";
        /*
         * If there is a firewall between you and nameservers you want
         * to talk to, you might need to uncomment the query-source
         * directive below.  Previous versions of BIND always asked
         * questions using port 53, but BIND 8.1 uses an unprivileged
         * port by default.
```

```
        */
        // query-source address * port 53;
        forward first;
        forwarders { 198.202.75.26; };
};
```

3. named 서비스를 재시작한다.

```
service named restart
```

이와 같이 하면 지정한 nameserver 를 이용하여 계산 노드에 name 서비스를 수행할 수 있다.

7.3.5 NAS(Network Attatched Storage)에 사용자 계정을 어떻게 추가하는가?

현재에는 이게 쉽지는 않지만 다음과 같이 하도록 한다. NAS 노드가 "bigbird"라고 가정하자. bigbird 에서는 사용자 계정을 /export/home/에 있다고 가정하고, 새로운 사용자인 "dave"를 추가한다고 하자.

1. 먼저 frontend 에서 dave 에 대한 계정을 생성하기 위해 "useradd" 명령을 사용한다.

```
# useradd dave
```

2. dave 의 홈디렉토리를 nas 로 이동한다.

```
# scp -r /export/home/dave bigbird.local:/export/home
```

3. bigbird 에 있는 홈디렉토리의 퍼미션을 변경한다.

```
# chown -R dave.dave /export/home/dave
```

4. frontend 에 있는 /etc/auto.home 를 편집한다. Bigbird 에 있는 dave 라인을 다음과 같이 변경한다.

```
install         gold.local:/export/home/&
dave            bigbird.local:/export/home/dave
```

5. 411 에 있는 auto.home 를 변경하고, autofs 를 재시작한다.

```
# make -C /var/411
# service autofs restart
```

이와 같이 하면 dave 의 홈디렉토리는 NAS 서버인 bigbird 에서 사용할 수 있으며 전체 클러스터에서 볼 수 있을 것이다.

7.4. Architecture

1. 왜 계산노드에 apache 대몬이 실행되어야 하는가?

계산 노드를 위한 기본적인 설정은 Apache 서비스를 시작하는 것이다. 이것은 차후의 모니터링 툴에서 Linux 의 /proc 파일시스템을 HTTP 를 통해 서비스할 수 있도록 하기 위함이다. Ganglia 는 Rocks 의 기본 모니터링 툴로 남을 것이다 하지만, 매우 자세한 노드 정보를 얻기 위해서는 전체의 /proc 파일시스템만 있으면 충분하다. apache 를 중지시키길 원하면 **배포판의 설정 그래프 파일**에서 다음의 줄을 제거하라.

```
<edge from="slave-node" to="apache"/>
```

8 장. 자원

8.1 토론 목록 아카이브

npaci-rocks-discussion 목록의 최근 아카이브 [1]

mailman[3] 으로 변경하기 전의 아카이브 [2]

노트 1. https://lists.sdsc.edu/pipermail/npaci-rocks-discussion/

2. http://www.rocksclusters.org/mail-archive/threads.html

3. http://www.gnu.org/software/mailman/index.html

참고 문헌

Papers

411 on Scalable Password Service , Federico D Sacerdoti, Mason J. Katz, and Philip M. Papadopoulos, July 2005, IEEE High Performance Distributed Computing Conference, North Carolina. , (PDF) .

Rolls: Modifying a Standard System Installer to Support User-Customizable Cluster Frontend Appliances , Greg Bruno, Mason J. Katz, Federico D. Sacerdoti, and Philip M. Papadopoulos, September 2004, IEEE International Conference on Cluster Computing, San Diego , (PDF) .

Grid Systems Deployment & Management Using Rocks , Federico D. Sacerdoti, Sandeep Chandra, and Karan Bhatia, September 2004, IEEE International Conference on Cluster Computing, San Diego , (PDF) .

Wide Area Cluster Monitoring with Ganglia , Federico D. Sacerdoti, Mason J. Katz, Matthew L. Massie, and David E. Culler, December 2003, IEEE International Conference on Cluster Computing, Hong Kong , (PDF) .

Configuring Large High-Performance Clusters at Lightspeed: A Case Study , Philip M. Papadopoulos, Caroline A. Papadopoulos, Mason J. Katz, William J. Link, and Greg Bruno, December 2002 , Clusters and Computational Grids for Scientific Computing 2002 , (PDF) .

Leveraging Standard Core Technologies to Programmatically Build Linux Cluster Appliances , Mason J. Katz, Philip M. Papadopoulos, and Greg Bruno, April 2002 , CLUSTER 2002: IEEE International Conference on Cluster Computing , (PDF) .

NPACI Rocks: Tools and Techniques for Easily Deploying Manageable Linux Clusters , Philip M. Papadopoulos, Mason J. Katz, and Greg Bruno, Submitted: June 2002 , Concurrency and Computation: Practice and Experience Special Issue: Cluster 2001 , (PDF) (PostScript) .

NPACI Rocks: Tools and Techniques for Easily Deploying Manageable Linux Clusters , Philip M. Papadopoulos, Mason J. Katz, and Greg Bruno, October 2001 , Cluster 2001 , (PDF) (PostScript) .

Talks

Rocks Basics, SUN HPC Consortium. , November 2004 , (Powerpoint) .

High Performance Linux Clusters, Guru Session, Usenix. , June 2004 , (Powerpoint) .

NPACI All Hands Meeting, Rocks v2.3.2 Tutorial Session , March 2003 , (PDF) (Powerpoint) .

Managing Configuration of Computing Clusters with Kickstart and XML , March 2002 , (Powerpoint) .

NPACI All Hands Meeting, Rocks v2.2 Tutorial Session , March 2002 , (PDF) (Powerpoint) .

NPACI Rocks: Tools and Techniques for Easily Deploying Manageable Linux Clusters , IEEE Cluster 2001, Newport Beach, CA. , October 2001 , (PDF) .

Introduction to the NPACI Rocks Clustering Toolkit: Building Manageable COTS Clusters , NPACI All Hands Meeting, Rocks v2.0 Tutorial Session. , February 2001 , (Powerpoint) .

Press

Itanium Gets Supercomputing Software , 10 April 2003, C|Net , (HTML) .

HP Unveils Industry's First Multi-processor Blade Server Architecture for the Enterprise : Enables Customers to Achieve Adaptive Infrastructures , 26 August 2002, Hewlett-Packard Company , (HTML) .

Linux, Intel and Dell - Supercomputing at a Major University : Intel e-Business Center Case Study , August 2002, Intel Corporation , (PDF) .

NPACI Rocks Simplifies Deployment of Intel Itanium Clusters , 12 June 2002, NPACI & SDSC Online, (HTML) .

The Beowulf State of Mind, 01 May 2002, Glen Otero, Linux Journal, (HTML) .

Linux on Big Iron, 25 March 2002, eWeek, (HTML) .

Keck-funded SDSC Satellite Site Proves a Boon to Campus Scientists , 6 March 2002, NPACI & SDSC Online, (HTML) .

Cal-(IT)2, IBM, SDSC & Scripps Inst. of Oceanography Announce COMPAS, December 2001, Supercomputing Online, (HTML) .

Texas Advanced Computing Center Adds Two New Clusters, December 2001, IT @ UT, (HTML) .

Cluster-Management Software Developers Preparing for the TeraGrid, July-September, 2001, NPACI Envision, (HTML) .

SDSC AND COMPAQ ANNOUNCE ALLIANCE TO OFFER HIGH-PERFORMANCE COMPUTING CLUSTERS USING NPACI ROCKS CLUSTERING TOOLKIT AS WEB FREEWARE, 9 July 2001, SDSC Press Room, (HTML) .

High Performace Computing for Proliant, 22 June 2001, Compaq Corporation Web Site, (HTML) .

NPACI Rocks Open-source Toolkit Improves Speed and Ease of Use in Cluster Configuration, 21 March 2001, NPACI & SDSC Online, (HTML) .

NPACI Releases Rocks Open-source Toolkit for Installing and Managing High-performance Clusters, 1 November 2000, NPACI & SDSC Online, (HTML) .

Research

A Cache-Friendly Liquid Load Balancer , Federico D. Sacerdoti, Masters Thesis: June 2002 , UCSD Technical Report , (PDF) .

A. 부록

A.1 3.3.0 릴리즈 - 3.2.0과 변경사항

새로운 기능- 노코나(nocona) 지원. Rocks는 기존 x86_64모드에서 인텔 ia32e 아키텍처를 지원함.

새로운 Roll - kernel Roll. 사용자가 커널을 쉽게 추가할 수 있도록 Rocks base에서 분리함.

새로운 Roll - Vizualization Roll. tile wall을 쉽고 효과적으로 생성할 수 있음. OpenGL 확장 추가함. 4000^2 픽셀로 Doom을 실행할 수 있음.

새로운 기능: Kickstart 보안. Kickstart 파일을 https로 암호화하여 전송함. X509 키를 이용하여 생성하고 클러스터 멤버쉽을 강화함. LAN과 WAN에 접근하는 kickstart에 대한 두 가지 인증을 설치함.

새로운 기능 - Meta Roll. 여러 개의 Roll을 단일 CD/DVD 미디어로 만들 수 있음.

향상된 기능 - Wan kickstart. CD를 이용하지 않고 네트워크로 이용하여 frontend 보안을 설정할 수 있음. network/CD 조합을 지원하고 복수 개의 네트워크 Roll 소스를 지원함

새로운 기능- 411 그룹. 411 파일이 노드의 하위 그룹(subset)에도 적용될 수 있음. 새로운 'chroot' 옵션을 이용하여 클라이언트에서 사용자가 원하는 위치에 411파일을 위치시킬 수 있음.

새로운 기능- kickstart 부하 제어. 이 기능은 frontend에 동시에 요청할 수 있는 kickstart 수를 조절합니다. 이 기능을 통해 대규모 클러스터를 동시에 재설치할 수 있으며, 일정한 주기로 계속해서(repeated backoff) kickstart 요청을 함으로써, 모든 노드가 재설치될 수 있도록 보장해줍니다.

새로운 기능 - 클러스터 shepherd. frontend가 업그레이트되거나 재설치된 후에 모든 계산 노드를 자동으로 collect하도록 함.

향상된 기능 - AMD Roll. x86_64 아키텍처를 지원하는 32비트 라이브러리의 수가 증가함. 물론 RPM 기반

향상된 기능 - MPICH2 0.971

버그 수정 - Ganglia 2.5.7. gmetad는 대규모 클러스터에서 정적 메모리 footprint를 가짐

향상된 기능 - SGE 5.3p6

향상된 기능 - LAM/MPI 7.1.1

A.2. 3.2.0 릴리즈 – 3.1.0과 변경사항

새로운 기능 – Condor Roll이 추가됨. 이 Roll은 Condor 프로젝트에서 온 것으로 고성능 분산 데이터 처리 기능을 Rocks 클러스터에 준다.

배포된 향상된 기능

새로운 기능 – Area51 Roll 추가. 이 Roll은 보안툴을 포함하고 있으며 파일과 클러스터에서 동작중인 운영체제를 통합 점검할 수 있는 서비스를 제공함.

새로운 기능 – Ganglia RSS뉴스 이벤트 서비스

향상된 기능 – 계산 노드의 네트워크를 처리를 향상시킴. 클러스터 내부 네트워크에서 어떠한 인터페이스라도 이용할 수 있도록 함. 하지만 'eth0' 는 아님.

향상된 기능 – x86과 x86_64를 포함하는 크로스 아키텍처 클러스터를 좀더 잘 지원하도록 함.

향상된 기능 – GM 장치 드라이버를 사용자 커널을 가지는 계산 노드에서 빌드하고 로드할 수 있도록 함(커널은 kernel.org에서 이용할 수 있는 커널)

향상된 기능 – 계산 노드에서 파티셔닝을 소프트웨어 RAID로 사용할 수 있도록 지원함

향상된 기능 – 루트와 스왑 파티션에 대한 변수 추가. 만약 root 나 swap에 대한 크기를 변경하고자 할 경우 단지 두 개의 XML 변수만 변경하면 되도록 함.

향상된 기능 – root 파티션 크기를 6GB까지 증가하도록 함(기존에는 4GB 까지임)

향상된 기능 – SGE ganglia 모니터를 추가함. 모든 SGE job에 대한 상태를 frontend의 웹 페이지에서 추적할 수 있음.

향상된 기능 – PXE뿐만 아니라 플로피 기반 Etherboot와 ia64를 확장 지원하도록 함

향상된 기능 – EKV를 보안 강화를 위해 telnet 대신 ssh를 이용하도록 함

향상된 기능 – 새로운 Myrinet MPICH 버전. 1.2.5..12.

향상된 기능 – 자바 Roll. JDK를 1.4.2_04로 업데이트 함

향상된 기능 – RHEL 소스 rpm을 이용하여 세가지 아키텍처에 대해 최근 소프트웨어를 재컴파일하여 업데이트 함

향상된 기능 – 자동으로 MySQL cluster 데이터 베이스를 백업하도록 함

향상된 기능 – “Cluster Labels" 출력에서 각각의 노드에 MAC 어드레스가 포함되도록 함

향상된 기능 – Rocks Base CD에서 Frontend rescue 모드가 지원되도록 함. 부트 프롬프트에서 "frontend rescue"를 입력하는 것으로 frontend의 상태를 시험할 수 있는 쉘 프롬프트가 나타나도록 함.

버그 수정 – 411이 강화됨. 변경 파일에 대한 좀더 신뢰성 있는 통보가 가능해짐. Frontend의 첫 번째 부팅시 로그인 파일을 암호화하도록 Makefile를 수정함

버그 수정 – frontend에서 여러 CD 드라이브를 지원하도록 함. 만약 하나 이상의 드라이브가 frontend에 있으면 인스톨러는 사용하는 CD를 올바르게 구분할 수 있도록 함.

버그 수정 – ganglia matrics를 frontend가 리부트할 때 저장함. 리부트한 후 모든 ganglia

내역은 이전 부팅에서 저장한 내역을 다시 읽어들임.
버그 수정 - PVFS 를 옵테론에서 -mcmodel=kernel 로 컴파일함
버그 수정 - XML 탈출 문자(escape characters)(예를 들어, &, <, >)등을 설치 프로그램에서 지원함(예를 들어, 클러스터 정보 화면에서 루트(root) 암호 화면 등)
버그 수정 - Intel Roll -모든 Intel 컴파일러 라이브러리를 계산 노드로 복사하도록 함

A.3 3.1.0 릴리즈 - 3.3.0과 변경사항

현재 이용할 수 있는 RedHat Enterprise Linux 3 소스(Advanced Workstation)를 이용하여 컴파일된 리눅스 패키지를 기반으로 함.
Sun Grid Engine 5.3을 기본 배치 스케쥴 시스템으로 변경함
추가 Roll: NMI/Globus Release4, Java, Condor, Inter 컴파일러 Roll를 이용 가능함.
새로운 아키텍처: Opteron(x86_64) revceives first-class functionality.
향상된 기능 - 새로운 MPICH 1.2.5.2. 좀더 효과적인 MPD 병렬 job launcher를 처리할 수 있도록 함. MPICH2를 기본적으로 포함함.
향상된 기능 - 최근 Myrinet mpich-gm 2.0.8를 모든 아키텍처에서 사용하도록 함.
향상된 기능 - SSH 버전 3.7.1로 업데이트함.
향상된 기능 - 411 보안 정보 서비스를 기본적으로 사용함. NIS 대체.
향상된 기능 - Greceptor 를 Gschedule로 대체하여 mpdring, 411, cluster-top와 그외등에서 지원할 수 있도록 함. Gschedule보다 크게 향상된 성능을 위해서 archieves함.

부록 B. Kickstart Node Reference

B.1. Rocks Base Nodes

B.1.1. 411

이 패키지와 411 보안 정보 서비스의 다른 공통 요소

상위 노드:

- base

B.1.2. 411-client

클라이언트를 위한 411 보안 정보 서비스 설치. 411 서비스는 파일이 배포되면 자동적으로 설정될 것이다. 또한 frontend 에 있는 모든 현재 411 파일을 kickstart 파일에 입력하기 때문에 단일 411 에 대한 고장 감내를 수행하지 못한다. 411 은 단일 시간 내에 모두 성공적으로 수행되리라는 보장은 없다..

상위 노드:

- client

B.1.3. 411-server

Master 노드에 411 보안 정보 서비스를 설정한다. 클러스터에 대한 RSA 공개키와 private 키를 생성하여 411 에 대해 Apache 를 설정한다.

상위노드:

- server

B.1.4. apache

아파치 HTTP 서버

상위 노드:

- base
- cluster-db

B.1.5. autofs

NFS 를 이용하여 홈디렉토리를 자동 마운트하거나 loopback 장치를 마운트하기 위한 AutoFS

상위 노드:

- autofs-client
- autofs-server

B.1.6. autofs-client

AutoFS Client

상위 노드:

- client

하위 노드:

- autofs

B.1.7. autofs-server

AutoFS 서버

상위 노드:

- server

하위 노드:

- autofs

B.1.8. base

모든 Rocsk 노드에 대한 기본 클래스. 여기에는 계산 노드, frontend 노드, 단일 랩탑 노드, 컴퓨터 랩, 그래픽 노드, nfs 서버 등이 포함되어 있다.

Base class for all Rocks nodes. This should include compute nodes, frontend nodes, standalone laptops, computer labs, graphics nodes, nfs servers To achieve this level of flexibility this base class should have edges only to those classes that implement the core of Rocks.

상위 노드:

- client
- server

하위 노드:

- 411
- apache
- c-development
- disk-stamp
- elilo
- fstab
- grub
- installclass
- ip-diag
- keyboard
- logrotate
- node
- node-thin
- rpc
- scripting
- ssh
- ssl

B.1.9. c-development

C 개발을 위한 최소한의 내용. 여기에는 커널을 컴파일하기 위한 모든것들이 포함되어 있다.

상위 노드:

- base

B.1.10. cdr

CDR 도구 (burnings, iso, ripping, mp3 enconding)

상위 노드:

- devel

B.1.11. central

Rocks 클러스터 중앙 서버. 네트워크를 통해 다른 서버에 대해 kickstart 할 수 있다.

상위 노드:

- server

B.1.12. client

Graph 에 포함되어 있는 '클라이언트 노드'. 이 파일은 다른 XML 설정 노드에 대한 connect point 로 사용된다.

하위 노드:

- 411-client
- autofs-client
- base
- installclass-client
- ntp-client
- ssh-client
- syslog-client

B.1.13. cluster-db

NPACI Rocks 클러스터 데이터 베이스

상위 노드:

- server

하위 노드:

- apache

B.1.14. cluster-db-data

초기 데이터가 포함된 대중적인 클러스터 데이터 베이스

상위 노드:

- server

B.1.15. cluster-db-structure

클러스터 데이터베이스 SQL table 구조. 이는 데이터베이스를 dump 할 때 생성된다. 직접 편집할 것이다.

상위 노드:

- server

B.1.16. devel

Graph 에 포함된 'devel node'. 이 파일은 다른 XML 설정 노드에 대한 connection point 로 사용된다.

상위 노드:

- server

하위 노드:

- cdr
- docbook
- emacs
- fortran-development

B.1.17. dhcp-server

클러스터에 대한 DHCP 서버 설정

상위 노드:

- server

B.1.18. disk-stamp

Root 파티션을 얻고 생성한다. (stamp 가 있다면) 파티션을 보존하거나 (stamp 가 없다면) 모든 디스크를 삭제할 것인지를 판단하거나 설치하는데 이용하는 핵심 키이다.

상위 노드:

- base

B.1.19. dns-server

Frontend 에 있는 DNS nameserver 를 설정. Forward 와 reversed zone 모두 데이터베이스를 이용한다.

상위 노드:

- server

B.1.20. docbook

DOC Book 지원(roll 를 빌드할 때 필요함)

상위 노드:

- devel

B.1.21. elilo

IA-64 부트로더 지원

상위 노드:

- base

B.1.22. emacs

Emacs OS

상위 노드:

- devel

B.1.23. fortran-development

Fortran

상위 노드:

- devel

B.1.24. fstab

설치하고 있는 박스의 디스크를 검사하고 존재한다면 root 파티션이 아니라면 계속 보존한다.

상위 노드:

- base

B.1.25. grub

IA-32 부트 로더

상위 노드:

- base

B.1.26. install

계산노드를 kickstart 하기 위해 필요한 모든 사항. 혹은 이 시스템에서 임의의 노드를 kickstart 하는데 필요한 모든 사항들.

상위 노드:

- server

B.1.27. installclass

기본 installclass 파일. 이 graph node 는 다른 installclasss graph node 보다 선행되어야 한다.

상위 노드:

- base

B.1.28. installclass-client

클라이언트 installclass 파일

상위 노드:

- client

B.1.29. installclass-server

서버 installclass 파일

상위 노드:

- server

B.1.30. ip-diag

TCP/IP 네트워크 진단 툴.

상위 노드:

- base

B.1.31. keyboard

Ia64 를 위한 USB 키보드 지원

상위 노드:

- base

B.1.32. logrotate

/var/log 에 로그를 기록하기 위한 logrotate 에 rule 를 추가

상위 노드:

- base

B.1.33. media-server

CD/DVD 에 있는 kickstart 파일에 대한 ROOT

하위 노드:

- server

B.1.34. node

Node 는 클러스터에 포함된 기계를 말한다. Node 는 private 네트워크에 있으며 frontend 로부터 DHCP/NIS 상태를 얻어온다.

상위 노드:

- base

B.1.35. node-thin

이 패키지를 사용하지 않으면 제대로 동작하지 않을 것이라 생각된다. 이 항목은 CD 에서 매우 많은 부분을 차지하고 있다. DVD 기반 시스템일 경우에는 그다지 크게 생각하지 않아도 된다. 이렇게 해야 하는 이유는 단지 rocks-enabled solution 을 하나의 단일 CD 에 맞추기 위해서이고, 아래에 있는 패키지로 병렬 프로그램을 실행하는 사람들에게 직접 영향을 주지 않기 위해서이다.

상위 노드:

- base

B.1.36. ntp

네트워크 시간 프로토콜

상위 노드:

- ntp-client
- ntp-server

B.1.37. ntp-client

네트워크 시간 프로토콜

상위 노드:

- client

하위 노드:

- ntp

B.1.38. ntp-server

네트워크 시간 프로토콜

상위 노드:

- server

하위 노드:

- ntp

B.1.39. perl-development

Perl 지원

상위 노드:

- scripting

B.1.40. python-development

Python 지원

상위 노드:

- scripting

B.1.41. rocks-dist

rocks-dist 를 이용하여 배포판 생성

상위 노드:

- server

B.1.42. rpc

RPC 지원 t

상위 노드:

- base

B.1.43. scripting

상위 노드:

- base

하위 노드:

- perl-development
- python-development
- tcl-development

B.1.44. server

Graph 에 있는 'server node'. 이 파일은 다른 XML 설정 노드에 대한 connection point 로 사용된다.

상위 노드:

- media-server

- server-wan

하위 노드:

- 411-server
- autofs-server
- base
- central
- cluster-db
- cluster-db-data
- cluster-db-structure
- devel
- dhcp-server
- dns-server
- install
- installclass-server
- ntp-server
- rocks-dist
- syslog-server
- x11-thin

B.1.45. server-wan

WAN 을 통해 kickstart 를 수행한 Rocks 클러스터 시스템. 중앙서버가 이를 이용하여 최소한의 kickstart 파일을 생성한다.

하위 노드:

- server

B.1.46. ssh

SSH 사용

상위 노드:

- base

B.1.47. ssh-client

SSH Config 는 계산 노드나 frontend 외의 시스템에서 사용함. Rocks 에서는 클러스터에 있는 모든 SSH 서버간에는 하나의 키만을 이용하고 있다. 이는 Man-in-the-Middle 공격에 대해서 문제가 생길 수도 있다는 것을 의미한다. 이러한 공격을 막기 위해 다양한 릴리즈(broadcastSSH)등을 사용하고 있다. 이러한 로직은 ssh.xml 에 없기 때문에 frontend 는 자체 키를 생성할 것이다.

상위 노드:

- client

B.1.48. ssl

Open SSL 지원

상위 노드:

- base

B.1.49. syslog

Syslog 설정

상위 노드:

- syslog-client
- syslog-server

B.1.50. syslog-client

전달된 메시지를 클라이언트에서 받아들이기 위한 syslog 설정

상위 노드:

- client

하위 노드:

- syslog

B.1.51. syslog-server

전달된 메시지를 서버에서 받아들이기 위한 syslog 설정

상위 노드:

- server

하위 노드:

- syslog

B.1.52. tcl-development

Tcl 지원

상위 노드:

- scripting

B.1.53. x11

X11 데스크탑을 위한 프로그램

상위 노드:

- x11-thin

B.1.54. x11-thin

여러가지 GUI 를 사용할 필요가 없을 경우 X11 를 좀더 가볍게 만든 버전. 단지 netscape 정도을 동작시키고자 할 때 필요함

상위 노드:

- server

하위 노드:

- x11