제19회 한국정보처리학회 춘계학술발표대회 논문집 제10권 제1호 (2003.5)

클러스터를 위한 프로세스 모니터링 시스템 개발

김상완*, 박형우*, 이상산**
KISTI 슈퍼컴퓨팅센터** 그리드연구실*
e-mail: {sangwan, hwpark, sslee}@hpcnet.ne.kr

Development of Process Monitoring System for Clusters

Sangwan Kim, Hyungwoo Park, Sangsan Lee
KISTI Supercomputing Center

요 약

고성능 컴퓨팅 자원으로 널리 사용되고 있는 클러스터(cluster)는 독립된 시스템을 네트웍으로 연결해 놓은 것으로 단일 시스템 이미지를 제공하는 것이 중요한데, 그 중에서도 클러스터에서 수행되는 작업을 지속적으로 모니터링하는 것은 관리자와 사용자에게 모두 필요하다. 본 연구에서는 기존의 클러스터 모니터링 등을 목적으로 개발된 툴과 비교하여 기능 및 성능이 우수한 프로세스 모니터링 시스템을 설계하고 개발하였다.

1. 서론

클러스터는 고성능 컴퓨팅(High Performance Computing) 또는 고 가용성 컴퓨팅(High Availability Computing)을 제공하기 위해 널리 사용되고 있으며, 현재 슈퍼컴퓨터와 같은 값비싼 컴퓨팅자원을 빠른 속도로 대체해 나가고 있다. 클러스터는 PC(personal computer)와 같은 값싼 컴퓨팅 자원을 네트웍으로 묶어 병렬처리의 목적으로 사용하기 위한 것이다. 클러스터는 하드웨어와 운영체제가 분리되어 있는 독립적인 시스템들을 모아 놓은 것이므로, 이것이 하나의 커다란 컴퓨팅자원으로 보여지도록 하는 단일 시스템 이미지(single system image)기술이 중요하며, 이와 관련하여 현재까지 많은 소프트웨어들이 클러스터를 위해 설계되고 만들어지고 있다. 그 중에서 클러스터 모니터링을 위한 도구는 관리자가 클러스터의 상태를 빠르고 정확하게 파악하여, 필요한 조치를 취하기 위해 필수적인 부분이다.

본 연구에서는 클러스터의 각 노드에서 실행되고 있는 프로세스(process)들의 정보를 모니터링 할 수 있는 시스템을 개발하였다. 본 논문의 구성은 다음과 같다. 제2장에서는 클러스터 환경에 사용하기 위해 기존에 개발된 모니터링 도구의 설계 구조와 기능등에 대해서 설명한다. 제3장에서는 본 연구에서 개발된 클러스터 프로세스 모니터링 시스템에 대해서 설명하고, 향후 성능 및 기능을 확장하기 위한 방안을 소개한다. 마지막으로 제4장에서는 결론을 맺도록 한다.

2. 기존의 클러스터 모니터링 도구들

이 절에서는 클러스터 관리 및 모니터링을 위해 개발된 도구들 중에서 SCE와 Ganglia에 대한 설계 구조와 기능에 대해서 설명하고, 각각의 한계점에 대해서 설명한다.

2.1. SCMS(Scalable Cluster Management System)

SCMS는 타일랜드의 Kasetsart University의 Parallel Research Group에서 만든 클러스터 관리 소프트웨어로써 SCE (Scalable Cluster Environment)라는 통합 소프트웨어의 한 부분이다. 파이썬(python)언어로 작성되어 있으며, X윈도우 환경의 GUI도 함께 지원을 한다. UNIX의 rsh명령을 이용하여 원격 노드에서 명령을 실행하는 간단한 구조로 되어 있다. 기능은 클러스터 노드의 CPU, 메모리, 디스크 사용량, 프로세스 정보, RPM 패키지 정보 등을 모니터링 할 수 있다. 그림1은 SCMS에서 계산노드의 프로세스를 모니터링 한 화면을 보여준다. 이것은 리눅스의 ps 명령

을 실행한 결과와 같다. SCMS에는 rsh을 이용하여 여러 개의 원격호스트에서 명령을 수행하게 하는 p-command를 함께 제공한다. p-command를 이용하여 클러스터의 계산노드에 동일한 명령을 실행할 수 있다.

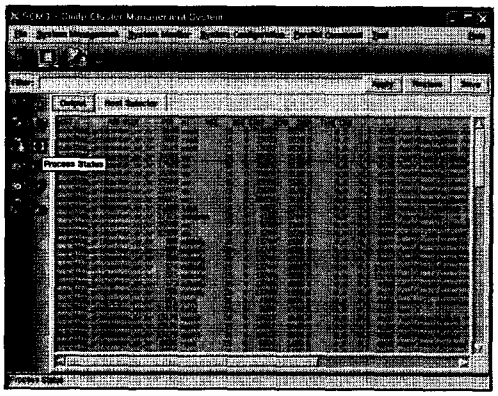

그림 1. SCMS processes status

2.2. Ganglia

Ganglia는 생명체의 신경절(節)을 의미하는 영어단어로, 클러스터의 사용 상태를 모니터링해 주는 도구이다. UC Berkerley의 Millennium Project (http://www.millennium.berkeley.edu/) 에서 개발되었다. Ganglia는 다음과 같은 세 개의 서로 독립적인 모듈로 구성이 된다.

- Ganglia Monitor Daemon (gmond)

모니터링하기 원하는 모든 호스트에 설치되며, 다른 호스트의 gmond 데몬에 멀티캐스트 메시지를 보내어 자신의 상태를 알리고, 다른 호스트의 정보를 수집하여 자신과 다른 노드의 시스템 상태를 XML(eXtensible Makeup Language)형식의 문서로 만들어 전달한다.

- Ganglia Meta Daemon (gmetad)

gmetad 데몬은 여러 gmond 데몬으로 부터 수집된 호스트 정보를 RRD(round-robin database)에 저장한다. gmetad는 Ganglia Web Interface가 설치될 웹서버서 동작한다.

- Ganglia Web Interface

웹 인터페이스 모듈은 gmetad에 의해 수집되어 RRD 데이터베이스에 저장된 시스템 정보를 웹을 통해 보여주는 역할을 한다.

Ganglia를 통해서 모니터링 할 수 있는 정보는 클러스터 각 호스트의 CPU, 메모리, 디스크 사용량과 네트웍 입출력 사용량 등이다. gmond로부터 수집된 데이터는 RRD 데이터 베이스에 축적되어 저장되며, 시간, 일, 주, 월, 년 단위로 그래프로 출력이 가능하다. 그림2는 Ganglia를 이용하여 클러스터를 모니터링 하고 있는 화면이다.

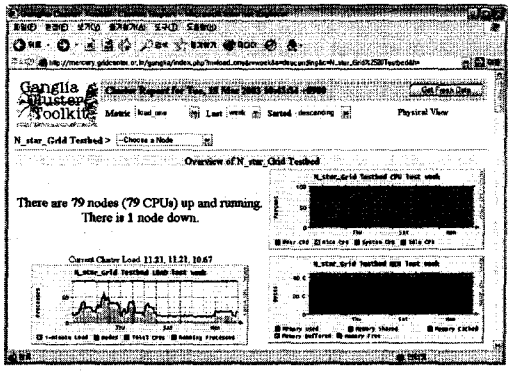

그림 2. Ganglia monitoring page

2.3. 기존에 개발된 도구들의 문제점

SCMS는 클러스터 상에서 계산노드의 프로세스의 상태 정보를 모니터링 할 수 있다는 장점이 있지만, 단순히 ps명령의 결과만을 보여줌으로써, 일정 시간동안 지속적 또는 주기적으로 모니터링 하는 것이 불가능하다. 또 웹 환경이 아닌 터미널 환경을 지원함으로써 사용이 불편하다는 단점도 가지고 있다. Ganglia는 계산노드의 전체적인 상태를 모니터링 할 수 있고, 모니터링 된 데이터가 데몬에 의해 데이터베이스에 쌓이게 되므로, 지속적인 모니터링이 가능하다는 장점이 있지만, 노드의 프로세스를 하나씩 모니터링 할 수 없다는 단점이 있다.

3. 클러스터 프로세스 모니터링 시스템

3.1. 시스템 구조

본 연구에서 개발된 클러스터 프로세스 모니터링 시스템의 구조는 그림 3과 같다. 모니터링 하려고 하는 노드들에는 pmond라고 불리는 모니터링 데몬이 동작한다. pmond 데몬의 역할은 네트웍크의 특정 포트를 통해서 모니터링 요청(request)이 들어오면 그 순간 호스트에서 실행되고 있는 프로세스의 정보를 수집하여 네트웍으로 전달하는 역할을 한다. 프로세스 정보는 리눅스의 경우 /proc 파일 시스템을 이용하여 수집이 가능하며, ps 명령의 출력을 파싱하여도 같은 정보를 얻을 수 있다. 수집된 프로세스 정보는 XML 형태의 문서로 만들어져 보내진다. 모니터링 요청의 옵션에 따라 네트웍으로 전송되기 전에 압축하여 전송이 가능하다.

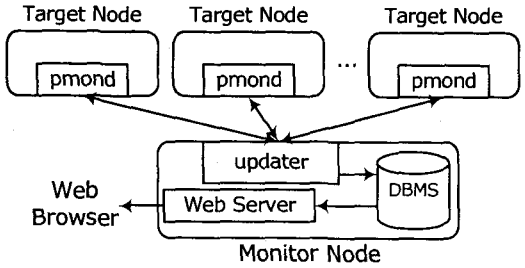

그림 3. Cluster monitoring system architecture

pmond는 외부의 요청에 따라 수동적으로 동작하도록 되어 있는 반면, 모니터 노드에서 동작하는 updater 데몬은 주기적으로 pmond 데몬들을 통해 노드의 프로세스 정보를 수집하는 역할을 한다. 수집된 데이터는 데이터베이스에 저장되고, 데이터베이스에 저장된 모니터링 정보는 웹서버와 사용자 인터페이스를 통해서 웹브라우저로 서비스된다.

3.2. 동작 및 기능 설명

그림 4는 웹브라우저를 통해 제공되는 사용자 인터페이스를 이용하여 클러스터의 프로세스를 모니터링하고 있는 화면이다. 프로세스 정보는 데이터베이스에 저장되므로 프로세스가 종료된 후에도 마지막으로 모니터링 된 프로세스 정보는 데이터베이스에 남아 있게 된다. 따라서, 연속적인 시스템의 모니터링이 가능하며 주어진 조건에 맞는 프로세스만 검색하여 모니터링 하는 것도 가능하다. 예를 들어 각 호스트에서 CPU를 몇%이상 점유하고 있는 프로세스만 선택할 수도 있고, 특정 사용자가 실행중인 프로세스만 선택할 수도 있다. 명령어에 특정 문자열이 포함되어 있는 것만 선택할 수도 있다.

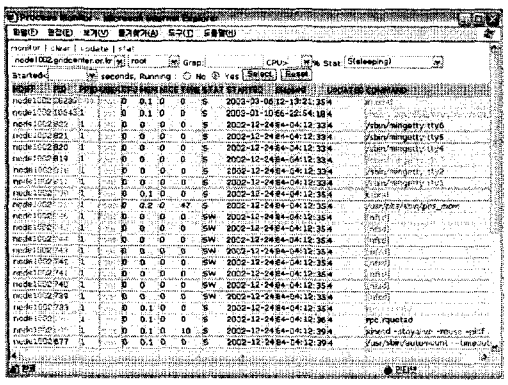

그림 4. Process monitoring user interface

그림 5는 updater 데몬에 의해서 호스트의 프로세스 정보가 수집되고 있는 상태를 보여 준다. 모니터링 할 호스트에 대한 정보도 DB에 저장되고, 각 호스트마다 업데이트 주기를 다르게 설정할 수 있다.

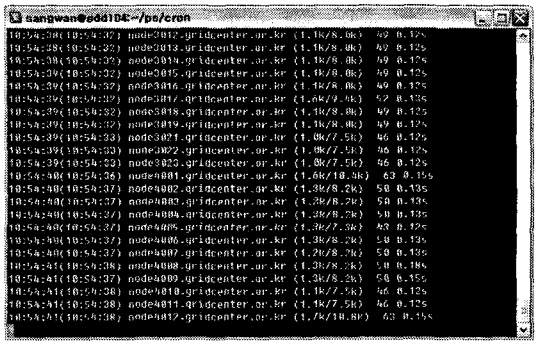

그림 5. Updating node information periodically

```
<?xml version="1.0"?>
<pg date="2003-03-16" time="04:22:07">
<p user="root" pid="1" ppid="0" cpu="0.0" mem="0.0"
stat="S" nice="0" etime="72-11:48:52" time="00:00:07"
ucomm="init" command="init [3]"/>
<p user="root" pid="2" ppid="1" cpu="0.0" mem="0.0"
stat="SW" nice="0" etime="72-11:48:52" time="00:00:00"
ucomm="keventd" command="[keventd]"/>
<p user="root" pid="3" ppid="1" cpu="0.0" mem="0.0"
stat="SW" nice="0" etime="72-11:48:52" time="00:00:00"
ucomm="kapmd" command="[kapmd]"/>
...
</pg>
```

그림 6. Monitoring information sent from pmond

그림6은 pmond로부터 가져온 프로세스 모니터링 정보이다. XML형식의 구조로 되어 있으며, 프로세스의 PID와 실행시간 등의 정보를 담고 있다.

3.3. 구현 환경 및 성능

본 연구에서는 모니터링 호스트로 Intel Pentium 4 1.7GHz에 RedHat Linux 8.0와 MySQL 3.23.49를 사용하였으며, updator를 PHP언어로 작성하였다. XML 문서의 파싱은 PHP의 DOM(Document Object Model) 관련 함수를 이용하여 구현하였다.

호스트에서 50개 정도의 프로세서가 실행되고 있을 때 pmond에서 만들어진 XML문서의 크기는 약 8Kbytes 정도가 되며, 이것을 압축하면 원래 크기의 15%정도로 줄어든다. updater가 XML문서를 파싱하고, 데이터베이스를 업데이트 하는 데 걸리는 시간까지 합하여 하나의 호스트에서 실행되는 모든 프로세스를 모니터링 하는데 걸리는 시간은 대략 0.13초 가량이 소요되었다. 80노드 클러스터에서 4000개 정도의 프로세스를 한번 모니터링 하는데 소요된 시간은 총 10초를 약간 넘었다.

3.4. 확장 방안

1, 2, 3절에서는 본 연구에서 개발된 모니터링 시스템의 구조와 기능 등을 설명하였다. 이 절에서는 이상에서 설명한 시스템을 보다 큰 규모의 클러스터로 확장할 수 있는 방안 등에 대해서 설명하고자 한다. 모니터 노드는 미리 지정된 호스트에 대해서 주기적으로 반복적인 데이터 수집을 실시한다. 이때 하나의 모니터 노드가 모니터링 해야하는 노드의 수가 많아지게 되면 각 호스트에 대해서 모니터링 주기가 길어지게 되고, 따라서 수집된 데이터의 정확도가 떨어지게 된다. 이러한 단점을 보완하기 위해서는 한 대의 모니터 노드가 담당하는 노드의 수를 제한하여 클러스터 내에 두 개 이상의 모니터 노드를 설치하여야 한다. 이때 데이터베이스도 노드수의 증가에 따라 분리시킬 필요가 있다. 분리된 데이터베이스를 하나로 보이도록 하기 위해서 모니터 쿼리(query) 노드를 추가하여 분산된 데이터베이스를 동시에 검색하도록 하면 된다. 쿼리 노드는 특정한 기준에 맞는 프로세스 정보를 모니터 노드로부터 가져와 사용자에게 제공하는 기능을 한다.

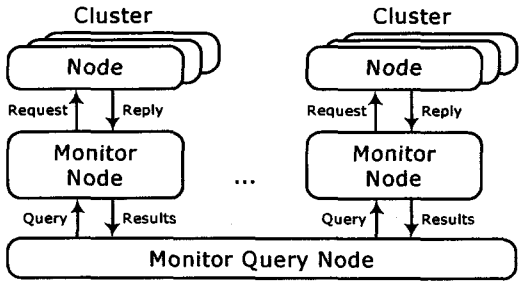

그림 5. *Extended monitoring system architecture*

pmond 데몬과 updater 데몬 사이는 요청(request)과 응답(reply) 프로토콜을 이용하고 있기 때문에, 모니터 노드의 설정에 따라 필요한 정보만 모니터링 하는 것이 가능하다. 요청 메시지에 모니터링 하려고 하는 정보가 어떤 것인지를 명시해 주고 pmond에 의해 명시된 정보만 모니터 노드로 보내는 방법이다. 예를 들어 특정 PID의 프로세스를 모니터링 하거나, 특정 사용자의 프로세서만 가져오도록 pmond에게 요청을 보냄으로써, 모니터링 정보의 양을 줄여, 빠른 응답속도와 정확도를 유지할 수 있다.

또한 모니터링 데몬과 모니터 노드에 필요한 모듈을 추가함으로써 프로세스 정보뿐만 아니라 기타 클러스터의 관리와 사용에 관련된 다른 정보들을 모니터링하도록 쉽게 수정이 가능하다.

4. 결론

본 연구에서는 클러스터에서 프로세스 정보를 모니터링 할 수 있는 시스템의 구조를 설계하고, 구현하였다. 구현된 시스템은 기존의 모니터링 툴과 비교하여 기능적인 면에서 우수할 뿐만 아니라 유사한 확장을 통하여 클러스터의 다른 정보에 대해서 모니터링을 할 경우에도 적용이 쉽다.

클러스터는 슈퍼컴퓨터와 같은 고성능 컴퓨팅자원을 대체하여 널리 사용되고 있으며, 그리드(Grid)와 같은 분산 협업환경의 플랫폼으로도 널리 사용될 전망이다. 이러한 점에서 클러스터와 관련된 관리도구와 모니터링 도구의 개발은 매우 중요하다고 할 수 있다.

참고문헌

[1] High Performance Cluster Computing, vol. 1&2, Rajkumar Buyya, 1999, Prentice Hall

[2] OpenSCE
http://smile.cpe.ku.ac.th/research/sce/

[3] Ganglia
http://ganglia.sourceforge.net/