리눅스 클러스터 모니터링 시스템 설계

심 형 용[0] · 선 동 국 · 김 성 조
중앙대학교 컴퓨터공학과

The Design of Linux Cluster Monitoring System.

Hyung Yong Shim[0], Dong Guk Sun, Sung Jo Kim
Dept. of Computer Science & Engineering, Chung-Ang University

요 약

인터넷의 보편화와 컴퓨팅 기술의 다양화에 따라 점차 고가용성 서버에 대한 관심이 높아지면서 낮은 비용과 높은 성능을 만족시켜줄 수 있는 리눅스를 기반으로 한 클러스터 서버가 인기를 끌고 있다. 하지만 클러스터 서버의 높은 성능에 비해, 여러 노드를 통합하여 효율적으로 관리 및 모니터링 할 수 있는 시스템은 상대적으로 부족한 상태이다. 하지만 이 논문에서는 기존의 모니터링 시스템의 문제점을 분석하고 이를 해결할 수 있는 새로운 모니터링 시스템을 설계하고자 한다.

1. 서론

네트워크 기술이 빠르게 성장하고, 그 이용분야가 넓어짐에 따라 네트워크 트래픽은 매년 평균 성장률이 100%에 이르고 있다[1]. 이에 따라 서버의 처리능력도 빠르게 증가해야 하지만 현실적으로 처리능력의 개선에는 한계가 있다.

처리능력의 개선을 위한 첫 번째 방법은 더 높은 성능을 가진 서버로 교체하는 것이다. 하지만 처리량이 증가할때마다 계속 서버를 교체하기에는 비용이 너무 높다는 단점이 있다. 두 번째 방법으로는 기존에 사용하던 다수의 서버를 이용하여 많은 태스크(Task)를 동시에 분산 처리하는 방법이 있다. 이 방법은 적은 비용으로 많은 성능향상을 가져올 수 있으므로 가장 현실성 있는 솔루션이라고 할 수 있다. 특히 최근에는 저가형 서버시장에서 새롭게 부각되고 있는 리눅스(Linux) 운영체제를 기반으로 한 클러스터 서버들이 속속 등장하고 있다[1].

클러스터 서버는 다수의 노드를 이용하여 태스크를 처리하기 때문에 각 노드의 상태 변화 및 성능을 모니터링하는 것이 중요한 이슈로 부각되고 있다. 하지만 현재의 모니터링 기법은 클러스터 환경에 유연하고 효율적으로 적용하기 힘들다. 이에 본 논문에서는 보다 유연하면서도 오버헤드를 줄일 수 있는 효율적인 모니터링 기법을 제안하고자 한다. 2장에서는 현재의 리눅스 기반의 클러스터 프로젝트 및 현재의 모니터링 소프트웨어의 특징, 동작원리에 및 문제점에 대해서 기술하며 3장에서는 클러스터 시스템에서 모니터링이 되어야 하는 성능요소에 대해서 기술한다. 그리고 4장에서는 2장에서 제시한 문제점 해결을 위한 새로운 모니터링 시스템에 대해서 기술한다. 그리고 5장에서 결론을 맺는다.

2. 기반 연구

리눅스를 이용하여 클러스터를 구성하는 방법은 SMP(Symmetric Multiprocessing), MPI(Message Passing Interface), PVM(Parallel Virtual Machine)등 여러 가지가 있다. 이러한 알고리즘을 기반으로 하여 현재까지 진행되었던 리눅스 기반의 클러스터 프로젝트들은 다음과 같다.

• 베오울프(Beowulf)[2][3]는 병렬 계산에 사용될 수 있도록 이더넷을 통해 서로 연결된 하나의 서버 노드와 하나 이상의 클라이언트 노드로 구성되어진 시스템을 구성하는 프로젝트이다. 베오울프는 리눅스를 작동 시킬 수 있는 일반적인 PC, 표준 이더넷 어댑터들, 그리고 스위치들로 이루어지며, PVM 과 MPI와 같은 소프트웨어를 사용한다. 서버 노드는 모든 클러스터들을 조절하고 클라이언트 노드에게 작업을 분배하며, 클러스터의 콘솔과 외부로 통하는 게이트웨이를 제어한다.

• 아발론(Avalon)[4]은 140개의 프로세서로 구성된 알파 베오울프 클러스터로서 일반 개인용 PC 기술만을 사용하여 만들었다. 사용된 운영체제는 레드햇5.0(RedHat5.0)을 사용하였고 노드간의 메시지 전달을 위하여 TCP 소켓 위에 작성된 기본적인 MPI 루틴들을 사용하고 있다. 아발론은 원래 천체 물리학에서부터 분자역학에까지의 사용을 목적으로하여 제작된 클러스터로써 로스 앨러모스 국립 연구소 비선형학 센터와 이론부서의 공동 벤처에 의해 제작되었다.

• LVS(Linux Virtual Server)[1]는 리눅스서버들을 클러스터로 구성하여 뛰어난 확장성과 가용성을 얻기위해 진행된 프로젝트이다. LVS에서 사용자는 투명하게 오직 하나의 가상 서버만을 볼 수 있지만 실제로는 다수의 서버들이 고속의 LAN이나 지역적으로 분산된 WAN에 연결되어 있다. 실제 서버의 프론트엔드(Front-End)에 부하분산서버가 위치하고 있는데 들어오는 요청들을 클라이언트 들에게 스케쥴링하며, 클러스터로 구성된 병렬의 서버를 단일 IP의 가상 서버로 인식하도록 한다. LVS는 클러스터 노드를 사용자에게 투명하게 추가 및 삭제가 가능하여 확장성을 높일 수 있으며, 노드나 데몬에 문제가 생기면 이를 바로 인식하여 이 즉시 시스템을 재구성하므로 높은 가용성을 얻을 수 있는 장점을 가지고 있다.

이렇듯 리눅스를 이용하여 클러스터 서버를 구성하기 위한 많은 프로젝트들이 진행되고 있는 가운데, 클러스터 서버의 성능 뿐만 아니라 클러스터의 상태변화에 따른 이상 작동 유무를 확인할 수 있는 관리 및 모니터링 분야가 중요한 요소로 부각되고 있다. 클러스터 모니터링 소프트웨어는 클러스터 패키지를 제공하고 있는 회사마다 각각 자사의 클러스터 기술에 최적화되어 클러스터 패키지에 함께 제공되고 있다. 하지만 기본 작동원리 및 개념은 비슷하다. 현재 일반적으로 사용되는 모니터링 도구들을 정리하면 다음과 같다.

• Mon[5]은 클러스터의 각 노드의 가용성 및 상태를 모니터링 하여 이상이 발생한 경우 경고를 보내주는 소프트웨어이다. Mon 소프트웨어는 SNMP 및 특별한 몇가지의 트랩(Trap)을 발생시켜 모니터링 및 경고 메소드를 실행할 수 있는 기능을 가지고 있다.

Mon은 두가지의 태스크를 수행함으로서 시스템 자원에 대한 모니터링 결과를 보여주는데, 첫 번째 태스크는 해당 기능에 대한 상태를 시험하고, 두 번째 태스크는 해당 기능의 이상여부에 따라 적절한 동작을 취하게 한다. 즉 Mon은 각 노드의 상태를 감지할 수 있는 모듈을 실행할 수 있는 스케쥴러라고도 할 수 있다.

• Ldirectord[6]는 LVS나 Heartbeat[1]와 같은 고가용성 클러스터 패키지와 같이 사용된다. 이 소프트웨어의 주된 역할은 주기적으로 해당 노드의 URL에 메세지를 보내서 각 노드들의 상태를 모니터링하게 된다. 만약 어느 한 노드가 다운되고 나면 다운된 서버는 서버들의 풀(Pool)에서 제외되고, 정상으로 돌아오면 다시 풀에 포함된다.

• Nanny[7]는 LVS를 위한 모니터링 소프트웨어이다. 하지만 오래된 모니터링 소프트웨어이므로 현재는 거의 사용되지 않는다. 이 소프트웨어는 각 호스트들에게 차례대로 쿼리문을 보내서 돌아오는 응답을 확인하여 각각의 노드상태 및 처리상태 등의 정보를 얻어온다.

이러한 모니터링 도구들은 단순히 각각의 서비스에 대해서 스크립트를 이용하여 실행된 명령에 대한 응답을 로그(Log)파일로 만들어서 상태를 모니터링하는 것이 대부분이기 때문에 다수의 노드와 다양한 서비스를 동시에 모니터링할때 발생할 수 있는 과도한 I/O 오버헤드가 문제점이 될 수 있다. 또한, 동적으로 노드를 추가하거나 제거할 때 마다 상황에 맞게 스크립트로 작성되어 있는 환경변수를 변경해 주어야 하는 단점을 가지고 있다.

3. 모니터링 요소

클러스터는 여러대의 서버를 유기적으로 조합하여 요구된 작업을 분산 처리하므로 각 노드의 상태를 관찰할 수 있는 모니터링 도구가 필요하다. 이러한 모니터링 도구는 여러 가지 클러스터 성능 요소들을 관찰하여야 하는데 그 항목들을 정리하면 <표 1>과 같다.

모니터링되는 요소는 노드 정보와 서비스 상태 그리고 클러스터 정보 이렇게 크게 세가지 그룹으로 나눌 수 있다. 노드 정보에는 현재 동작중인 각 노드별의 CPU, 메모리, 네트워크 등 각각의 자원에 대한 정보들이 포함되어 있다. 이 정보들을 이용하여 개별적인 노드의 상태를 모니터링 할 수 있다. 서비스 상태에는 각 노드에서 제공되는 HTTP, TELNET, FTP, DNS등의 다양한 인터넷 서비스의 동작 및 이상 유무를 알아낼 수 있는 정보가 포함되어 있다. 마지막으로 클러스터 정보 그룹이 있는데 이 정보는 클러스터 전체가 적절히 유지되고 있는지를 감시하기 때문에 이는 어떻게 보면 클러스터 모니터링 요소들 중에서 가장 중요한 요소들이라고 할 수 있다. 이 클러스터 정보에는 노드 전체에서의 분산 처리 정도를 알 수 있는 로드(Load)분산률과 노드의 생존유무 그리고 노드 전체 자원의 소모율에 대한 통계자료들이 포함되어 있다.

이러한 클러스터 모니터링 통계 정보들을 이용하여 클러스터 서버 관리자는 클러스터의 이상 유무를 빠르게 파악하여 유지 보수 할 수 있다.

그룹	성능 메트릭
노드 정보	CPU 사용량
	메모리 사용량
	네트워크 사용량
	현재의 세션(사용자)수
	프로세스 수/크기
	스레드 수
서비스 상태	HTTP
	SMTP
	TELNET
	FTP
	DNS
	NNTP
	POP3
클러스터정보	각 노드의 로드 분산률
	각 노드의 생존 유무
	각 노드의 자원 사용률

<표 1> 클러스터 성능 요소

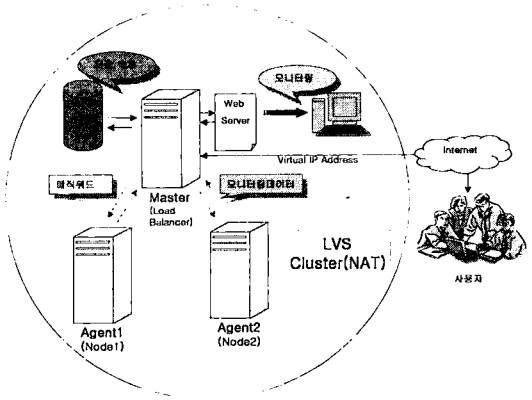
4. 시스템 설계

4.1 시스템 구성

본 논문에서 제안하는 모니터링 시스템은 마스터(Master)/에이전트(Agent) 모델을 사용하는데, 기본구조는 각 노드에서 에이전트가 작동하여 마스터로 모니터링 정보를 전송해주는 구조로 설계되었다. 이 구조의 특징은 마스터가 에이전트가 설치된 노드들의 상태에 따라 유연하게 모니터링할 수 있다는 점이다. 즉 새로운 노드가 클러스터에 참가하여 동작할 때와 노드의 이상으로 인해 더 이상 노드로써의 역할을 하지 못할 때를 구분하여 모니터링 하는 것이 가능하다.

마스터 프로세스는 클러스터의 상태를 각 노드에서 동작하고 있는 에이전트 프로세스로부터 정보를 수집하여 데이터베이스에 저장한다. 이는 정보를 단순한 로그파일로 저장하여 발생하는 I/O오버헤드를 최소한으로 줄이기 위함이다. 이렇게 설계된 클러스터의 전체 구조를 살펴보면 [그림 1]과 같다.

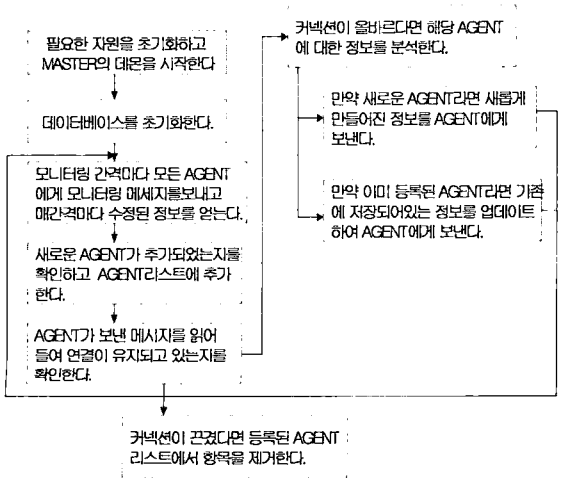
본 시스템은 LVS를 이용하여 구성된 클러스터를 기반으로 설계 되었다. 클러스터 서버의 기본구조는 외부에서 가상IP로 접근해온 요청을 로드 밸런서(Load Balancer)에 의해 Node1이나 Node2에 분산시켜 처리하고 그 결과를 다시 로드 밸런서를 통하여 사용자에게 응답한다. 마스터는 로드 밸런서 서버 또는 별도의 서버에서 동작하여 각 노드들에서 동작중인 에이전트들로부터 모니터링 정보를 받아 데이터베이스에 저장한다. 관리자는 이렇게 저장된 모니터링정보의 통계자료를 웹을 통해서 확인할 수 있다.



[그림 1] LVS 환경에서의 모니터링 기법 적용 모델

4.2 마스터

마스터 프로세스는 로드 밸런서 또는 별도의 서버에서 동작하며 각 노드의 상태를 체크하여 그 정보를 데이터베이스에 유지, 관리하는 역할을 한다. 마스터의 기본동작 과정은 [그림 2]와 같다. 마스터는 각 노드의 에이전트와 주기적으로 통신하며 각 노드의 상태정보를 받는다. 만약 새롭게 노드가 클러스터에 참가하고 로드 밸런싱이 이루어지기 시작한다면 에이전트로부터 메시지를 전송받고 데이터베이스의 노드리스트에 새로운 노드를 추가한다. 추가작업이 끝나면 마스터는 그 에이전트에게 해당 매직워드를 전송하는데 이 매직워드를 사용하여 각 노드에서 마스터를 구별한다. 그리고 설정된 시간간격마다 에이전트에게 정보를 요청하여 그 결과값을 데이터베이스에 저장한다. 이렇게 데이터베이스에 저장된 각 노드로부터의 모니터링 정보들을 분석하여 클러스터 전체의 로드 분산률 및 각 노드의 생존 유무 그리고 할당된 자원의 사용률에 대한 통계자료를 만들어서 다시 데이터베이스에 저장한다. 각각의 클러스터 노드들의 상태정보는 웹 인터페이스 즉 웹 브라우저를 통해서 확인할 수 있다.



[그림 2] 마스터(Master) 프로세스 동작과정

4.3 에이전트

에이전트 프로세스는 실제로 작업이 이루어지는 각각의 노드에서 동작하는데, 기본적인 역할은 에이전트가 위치한 노드의 각종 데이터를 수집하여 마스터로 정보를 보내는 일을 반복적으로 수행하는 것이다. 즉, 노드의 CPU 사용량, 메모리 사용량, 프로세스 및 스레드의 수를 얻고, 각 노드에서 제공되고 있는 SMTP, POP3, DNS, NNTP등과 같은 웹 서비스의 상태를 모니터링하여 그 결과를 마스터에 넘겨준다. 이러한 과정은 에이전트가 위치한 노드가 셧 다운되거나 노드에 이상이 발생할 때까지 일정 시간마다 반복하게 된다. 셧 다운이 되어 클러스터에서 노드가 더 이상 작동을 하지 않게 되면 커넥션 종료 메시지를 마스터에게 전송하여 마스터의 데이터베이스의 노드 리스트에서 해당 노드의 레코드를 삭제하게 된다.

5. 결론

본 논문에서는 기존의 클러스터 모니터링 소프트웨어에서의 새롭게 클러스터에 참여하게 된 노드의 활성화 및 셧 다운/오작동에 따른 불필요한 최적화 비용을 줄이고, 로그에 대한 I/O 오버헤드를 최소화하기위해 파일 입출력이 아닌 데이터베이스를 사용하도록 설계하였다. 이런 설계를 통해서 모니터링의 자동화와 작성된 데이터베이스에 기초한 여러 가지 통계 자료 및 각종 시스템 분석을 위한 어플리케이션으로의 확장이 가능하다.

향후 과제로는 데이터베이스의 다양한 분석에 따른 고장 노드의 처리 알고리즘이나 분석된 자료를 사용자에게 좀더 효율적으로 보여줄 수 있는 다양한 인터페이스의 개발이 필요하다. 또한 분산처리를 제어하는 로드 밸런서로 하여금 이상발생시 자동으로 클러스터 시스템을 복구하도록 하는 모듈이 개발된다면 클러스터 서버관리의 자동화를 이룰 수 있을 것으로 기대된다.

6. 참고문헌

[1] Wensong Zhang, Shiyao Jin, Quanyuan Wu "Creating Linux Virtual Servers"
LinuxExpo 1999.

[2] Jacek Radajewski, Douglas Eadline "Beowulf HOWTO", November 1998.

[3] Thomas Sterling, Donald J. Becker, Daniel Savarese, Michael R. Berry, and Chance Res. "Achieving a Balanced Low-Cost Architecture for Mass Storage Management through Multiple Fast Ethernet Channels on the Beowulf Parallel Workstation." Proceedings, International Parallel Processing Symposium, 1996.

[4] AVALON HomePage. http://cnls.lanl.gov/avalon

[5] The Linux Kernel Archives Homepage. http://www.kernel.org/software/mon

[6] UltraMonkey Project HomePage. http://www.ultramonkey.org

[7] RedHat HomePage. http://www.redhat.com