커널 기반 그리드 응용 모니터링 시스템의 개발

김 태 경[†] · 김 동 수[†] · 변 옥 환[††] · 정 태 명[†††]

요 약

그리드 응용이 수행되는데 필요한 네트워크 자원을 실시간으로 측정하고, 그에 대한 통계 데이터를 제공함으로 그리드 응용별 시스템 및 네트워크의 사용 자원의 양을 분석하기 위해서, 본 논문에서는 효율적인 모니터링 방법에 대한 연구를 수행하여, 커널을 기반으로 하는 모니터링 방법을 제안하였다. 이 방식을 이용한 성능 모니터링 측정의 장점은 tcpdump 등 기존의 패킷 캡처 방법에 비해서 시스템 자원을 적게 사용하면서 정확하면서도 적은 지연시간으로 측정이 가능하다는 것이다. 또한 이 모니터링 방법을 이용하여 실제 그리드 어플리케이션의 시스템 및 네트워크의 사용 정보량을 측정하는 시스템을 구현하였다. 이러한 연구는 그리드 응용 개발 및 그리드 네트워크 상의 자원할당 및 분배가 효율적으로 이루어 질 수 있도록 스케쥴러에게 필요한 정보를 제공할 수 있으며, 네트워크 관리자에게 그리드 네트워크 구축에 대한 근거 자료를 제시할 수 있다.

The Development of Kernel-based Monitoring System for Grid Application

Tae-Kyung Kim[†] · Dong-Su Kim[†] · Ok-Hwan Byeon[††] · Tai M. Chung[†††]

ABSTRACT

To analyze the usage information of system and network resources to the each grid application by measuring the real time traffic and calculating the statistic information, we suggested the kernel-based monitoring methods by researching the efficient monitoring method. This method use small system resourcesand measure the monitoring information accurately with less delay than the usual packet capture methods such as tcpdump. Also we implemented the monitoring systems which can monitor the used resources of system and network for grid application using the suggested kernel-based monitoring method. This research can give the useful information to the development of grid application and to grid network scheduler which can assign the proper resources to the grid application to perform efficiently. Network administrator can decide whether the expansion of network is required or not using the monitoring information.

키워드 : 그리드 응용(Grid Application), 성능측정(Measuring Performance), 커널(Kernel)

1. 서 론

그리드는 지리학적으로 분산되어 있는 고성능 컴퓨터 자원을 네트워크로 상호 연동하여 조직과 지역에 관계없이 사용할 수 있는 환경을 말한다. 그리드는 네트워크, 통신, 연산, 정보자원을 통합하여 동일한 방식으로 연산과 데이터 관리를 위한 가상의 플랫폼을 제공하는 것으로, 인터넷에서 자원을 통합하여 정보를 위한 가상의 플랫폼을 구성한다. 즉, 그리드 기반구조는 능동적으로 자원을 결합하여 대규모의 많은 자원을 필요로 하는 분산 응용 프로그램을 수행할 수 있는 능력을 제공한다[1]. 그리드 서비스의 관리는 서비스를 수행할 수 있는 가용성을 유지하는 것에 중점을 두는 것에서 수용 가능한 품질을 유지하는 것으로 옮겨가고 있다. 그리드 컴퓨팅은 작업의 처리를 "best effort" 방식으로 수행을 하는데, 이 "best effort" 방식은 더 이상 그리드 서비스에 있어서 효율적인 방식이 아니며, 그리드 네트워크에서 서비스 품질(Quality of Service)을 만족시키는 방안을 제시하는 것이 필요하다. 그러므로, 이러한 품질을 만족시키기 위해서는 우선적으로 그리드 네트워크의 성능을 측정하는 시스템이 필요하다.

그리드 네트워크의 성능을 측정하는 시스템은 그리드 네트워크 자체의 성능을 측정하는 시스템과 그리드 네트워크 상에서 수행되는 그리드 응용의 성능을 측정하는 시스템으로 나누어 볼 수 있으며, 본 연구는 그리드 응용의 트래픽을 측정하고, 분석함으로 그리드 응용별 필요 자원에 대한 분석

† 준 회 원 : 성균관대학교 대학원 정보통신공학부
†† 종신회원 : KISTI 슈퍼컴퓨팅센터 책임연구원, 초고속연구망실장, 슈퍼컴퓨팅인프라 개발실장
††† 종신회원 : 성균관대학교 정보통신공학부 교수

논문접수 : 2004년 4월 24일, 심사완료 : 2004년 9월 8일

및 자원 제공을 위한 스케쥴링 작업에 필요한 정보를 제공하기 위해 본 연구를 수행하였다. 이러한 시스템을 설계함에 있어서 고려해야 할 사항은 성능측정을 시스템의 부하를 최소화 하면서 정확하게 측정할 수 있는 도구를 구현함으로, 실제 그리드 네트워크 상에서 효율적으로 사용될 수 있는 시스템을 개발해야 한다는 것이다.

Brian L. Tierney의 연구[2]에 의하면 분산 응용의 성능 문제와 관련하여 45%는 네트워크와 관련하여 발생하고, 45%는 응용 프로그램의 디자인 문제나 버그에 의해서 발생하며, 10%는 호스트와 디스크의 문제로 발생한다고 조사되었다. 이러한 네트워크와 응용 프로그램의 성능 문제를 해결하기 위해서 그리드 응용 트래픽의 모니터링이 필요하며, 이러한 측정을 통하여 각 응용별로 사용되는 자원의 특성을 분석하고, 분석된 자료를 바탕으로 네트워크의 자원 할당 시에 각 응용별 특성에 맞게 자원을 할당함으로 시스템의 효율성을 높일 수 있으며, 그리드 응용 개발자에게는 사용자원에 대한 정보를 제공함으로, 성능을 안정적으로 관리할 수 있으며, 또한 그리드 네트워크를 구축하는 기반자료로 사용할 수 있다.

논문의 구성을 살펴보면, 2장에서는 관련연구에 대해서 살펴보고, 3장에서는 그리드 응용 성능측정을 위한 시스템의 설계 및 세부적인 구현에 대해서 제시 하였으며, 4장에서는 결론 및 향후 연구방향에 대하여 언급하였다.

2. 관련 연구

네트워크의 측정 방식은 크게 액티브(Active) 방식과 패시브(Passive) 방식이 있다[13]. 모든 네트워크 측정도구는 위의 두 방식으로 구분 할 수 있는데, 액티브 측정 방식은 도구가 스스로 조사 패킷(probing packet)을 생산하여 네트워크에 보내는 것을 의미하며, 패시브 측정 방식은 특별한 조사 패킷을 생성하지 않고 네트워크 상의 트래픽을 모니터링 함으로서 네트워크 특성을 측정하는 것을 말한다. 대표적인 액티브 네트워크 측정 도구로는 Iperf와 Gloperf가 있다.

2.1 Iperf의 기능

Iperf는 TCP와 UDP 대역폭 성능을 측정하기 위한 도구이다. Iperf를 사용하여 사용자는 자신의 네트워크에 대한 최대 TCP 대역폭을 측정할 수 있고, 다양한 매개변수 및 UDP 특성들을 튜닝 할 수 있다. Iperf는 네트워크의 단대단 대역폭 및 지연, 지터, 데이터그램 손실률 등을 사용자에게 알려주는 기능을 가지고 있다[3, 12].

Iperf에서 대역폭의 측정은 단대단에서 획득할 수 있는 TCP와 UDP의 최대 대역폭을 측정한다. 이를 위해 Iperf는 고정된 시간 동안 일정양의 TCP 스트림이나 일정 비율을 갖는 UDP 데이터그램을 생성하고, 측정하고자 하는 특정 경로의 양 종단에, 한 쪽에는 Iperf 서버 데몬을 사용하여 패킷 수신을 기다리고, 한 쪽에는 Iperf 클라이언트를 사용하여 생성한 스트림이나 데이터그램을 송신한다. 클라이언트가 보낸 데이터의 양과 서버가 받은 데이터의 양, 측정 시간, 측정시 발견된 손실률, 지연, 지터 등이 고려되어 서버는 사용자에게 TCP 대역폭의 측정을 위해 일정한 시간(10초 정도)에 걸쳐 패킷을 보내어 측정된 누적 대역폭 값을 사용한다. 또한, 패킷의 손실 및 지터의 값을 측정하기 위해서 Iperf 서버는 데이터그램의 ID 번호를 통해 UDP 데이터그램 손실률을 알아낼 수 있다. 일반적으로 UDP 데이터그램은 몇 개의 IP 패킷이 된다. 데이터그램 손실률 대신에 패킷 손실률을 측정하기 위해, Iperf에서는 -i 옵션을 사용하여 UDP 데이터그램을 단일 패킷에 알맞도록 충분히 작게 만든다. 디폴트 패킷 크기는 이더넷에서 동작하는 1470byte이다. UDP 테스트는 경로에 대한 패킷 손실률을 테스트 하는데 사용되며, 지터는 서버에 의해서 지속적으로 계산된다. 이것은 [4]에 명시된RTP(A transport protocol for real-time 어플리케이션)에 의해서 가능하다. 클라이언트는 패킷 내에 64bit second/microsecond 타임스탬프를 기록한다. 서버는 전송 시간을 '서버의 받은 시간 클라이언트의 전송 시간'의 방법으로 계산한다. 지터는 연속적인 전송 시간(transit time)들 사이의 차이에 대한 평균값으로 산출된다. 마지막으로, Iperf 퀵모드는 SLAC의 IEPM-BW(Internet End-to-end Performance Measurement-Bandwidth in the World) 프로젝트의 하나로 진행된 Iperf TCP 대역폭 측정 메커니즘의 확장 버전이다. Iperf 퀵모드 [5]는 기존 TCP의 문제점을 해결함으로 TCP를 사용하는 단대단의 경로 상에서 실제로 획득할 수 있는 최대 네트워크 대역폭을 좀 더 빠르고 정확하며, 간단하게 측정하는 것이 목적이다.

이를 정리하면, TCP를 통해서는 대역폭의 크기를 측정하고 읽는 크기와 MTU(Maximum Transfer Unit) 크기를 조사하여 사용자에게 알려준다. 만약 pthreads나 Win32 threads가 이용가능하다면 다중 쓰레드가 지원되어, 클라이언트와 서버는 동시에 여러 개의 연결을 가질 수 있다. 그리고 UDP를 통해서는 패킷의 손실의 크기 및 지연, 지터를 측정하며, 멀티캐스트 기능을 지원한다. Iperf의 주요 목적 중 하나는 특정 경로 상에서 TCP 연결에 대한 튜닝을 돕는 것이다. 가장 기본적인 TCP 튜닝의 이슈는 TCP 윈도우 사이즈를 결정하는 것이며, TCP 윈도우 사이즈는 특정 시점에서 네트워크 상에 존재할 수 있는 데이터의 양을 의미한다. TCP의 두 번째 튜닝 이슈는 MTU(maximum transmission unit)이다. 최대의 효율을 내기 위해서, 양단의 호스트들은 Path MTU Discovery를 지원한다.

2.2 Gloperf의 기능

Gloperf는 글로버스 그리드 컴퓨팅 툴킷의 한 부분으로서 개발되었다. 일반적인 성능 모니터링 시스템은 4가지 기능적인 요소로 구성된다. ① 센서(sensors), ② 데이터 대조

(collation of data), ③ 유도 데이터의 생산(production of derived data), ④ 데이터의 접근과 사용(access and use of data) 이다. Gloperf는 센서로서 동작하는gloperfd 데몬들의 집합으로 구현된다. Gloperf 데이터는 LDAP(Lightweight Directory Access Protocol) 서버인 글로버스 MDS(Meta-computing Directory Service)에서 대조된다. MDS는 스키마에 이름을 부여하는 구조화된 정보를 구현함으로써 특정 MDS 클라이언트가 Gloperf 데이터에 접근하는 것을 가능하게 한다. Gloperf는 각각의 gloperfd가 측정정책을 따르도록 하는 정보에 대해서MDS에 의존한다. 각각의 데몬은 MDS 정보를 통해 개체(peer)와 그룹을 발견하고, 이러한 발견을 통해서 측정을 수행하기 위한 개체를 선정한다. gloperfd는 외부 시그널에 의해 제거될 때까지 루프를 도는데, 매 루프마다 gloperfd는 다른 모든 gloperfd가 개체나 그룹 발견을 하도록 MDS에게 질의한다. 그룹 멤버쉽은 데몬의 MDS 객체에 표시되어 있는 그룹 이름에 의해 표시된다. gloperfd는 오직 동일한 그룹 내에 있는 peer에게만 테스트를 실시하며, Gloperf는 netperf 라이브러리를 기반으로 단대단의 대역폭과 지연시간 측정을 수행한다[6].

기존에 수행된 관련연구들은 그리드 상에서 수행되는 응용 프로그램에 대한 정보를 수집하기 보다는 응용 프로그램이 수행될 당시의 네트워크의 정보를 수집하는 형태를 가지고 있다. 그럼으로 수행되고 있는 각각의 응용 프로그램이 수행되기 위해서 필요한 네트워크 자원을 실시간으로 측정하고, 통계 데이터를 제공하여, 그리드 응용별 시스템 및 네트워크의 사용 자원의 양을 분석하는 것이 어렵다.

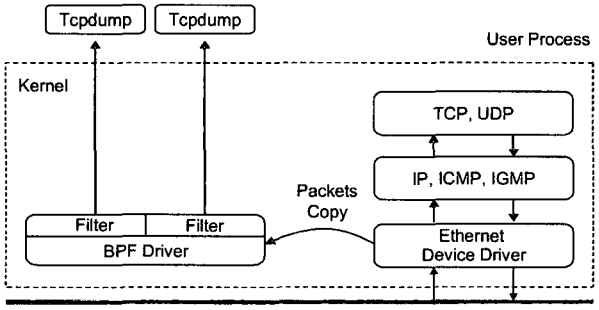
이에 본 연구에서는 각 그리드 응용 프로그램별 시스템 자원 사용량을 측정함으로, 그리드 네트워크 구축에 대한 근거자료의 제시와 그리드 네트워크 상의 자원할당 및 분배가 효율적으로 이루어 질 수 있도록 하는 데에 필요한 정보를 제공할 수 있다.

3. 그리드 응용 트래픽 모니터링 방법론

3.1 응용 계층에서의 성능 측정 방법

응용 계층에서의 네트워크 성능을 측정하는 방법으로는 패킷을 캡쳐하여 분석하는 방법이 있다. 패킷 캡쳐란 네트워크 상에 돌아다니는 패킷을 들여다보는 것을 의미한다. 만약 사용하는 호스트가 포함된 네트워크를 관리하는 라우터가 스위칭 라우터가 아닌 일반 라우터라면, 내부로 향하는 모든 패킷은 브로드캐스팅 된다. 이는 스위칭 라우터가 아닌 한은 모든 로컬 네트워크의 패킷을 캡쳐할 수 있음을 의미한다. 운영체제는 자신에게 도착된 패킷 중 목적지가 자신인 패킷만을 처리하여 응용 계층까지 올려 보낸다. 응용 계층의 측정도구로는 Libpcap이 있다. Libpcap은 "Portable Packet Capturing Library"의 약자이며, 네트워크 패킷을 간편하게 캡쳐하기 위한 API를 제공하는 라이브러리이다. 이와 유사한 패킷 캡쳐 관련 도구로는SOCK_PACKET, LSF, SNOOP, SNIT 등이 있지만, 이들은 대부분 운영체제에 종속적이어서 운영체제 별로 프로그램을 작성해야 하는 불편함이 있다. Libpcap은 이와 달리 운영체제에 상관없이 범용적으로 사용 가능한 API를 제공해 줌으로써 운영체제에 상관없이 운용 가능한 프로그램과 라이브러리의 제작이 가능하게 해준다. 또한 간단하게 사용 가능한 사용자 계층의 라이브러리라는 장점이 있다. Libpcap을 개발한 Van Jacobson, Craig Leres, Steven McCanne(Lawrence Berkeley National Laboratory, University of California, Berkeley, CA)에서는 Libpcap을 활용한 패킷 캡쳐 인터페이스 프로그램인 TCPDUMP를 제공한다. 이외에 ETHEREAL, SNORT, NTOP, NMAP, NGREP 등 Libpcap을 이용한 응용 프로그램들이 많이 존재하며, 상용 IDS 제품의상당수가 패킷 분석을 위하여 Libpcap을 사용한다. Libpcap을 이용하여 응용의 네트워크 트래픽 사용량을 측정할 때 추출 가능한 인자의 도메인으로는 TCP 헤더, IP 헤더, UDP 헤더가 있다[7].

일반적으로 Berkeley-derived 시스템에서 사용하는 BPF (Berkeley Packet Filter)는 이더넷 디바이스 드라이버를 promiscuous 모드로 변환시키고 사용자에 특화된 필터(user-specified filter)에 의해서 사용자 프로세스가 관심이 있다고 판단되는 패킷들만 사용자 프로세스로 전달된다. BPF의 작동은 커널에서 동작되고 사용자 프로세스가 관심이 있는 패킷만이 사용자에 특화된 필터에 의해서 어플리케이션(TCP DUMP)으로 전달된다. 즉, 커널 영역에서 필터링을 하고 버퍼링이 뒤따르기 때문에 버퍼 오버플로우로 인한 패킷 손실을 줄일 수 있다. 이에 대한 구조는 다음과 같다.



(그림 1) BPF(Berkeley Packet Filter)의 구성도

3.2 커널 계층에서의 성능 측정 방법

커널 계층에서의 성능 측정 방법은 LibKVM(Kernel Virtual Memory access library)을 이용한방법과 /proc 파일시스템 모니터링을 통한 측정 그리고 커널의 네트워크 관련 모듈을 통한 측정방법이 있다.

3.2.1 LibKVM을 이용한 방법

LibKVM을 이용한 방법은 /dev/kmem 장치를 직접적으로 경유하여 커널 자료에 접근하기 위해 kvm_open, kvm_nlist, kvm_read와 같은 루틴들을 사용한다. 일반적으로 이러한 프로그램들은 /dev/kmem을 열어, 커널의 심볼 테이블(symbol table)을 읽고, 이 테이블을 가지고 움직이는 커널안에 자료

를 위치시킨다. 그리고 커널 주소 공간 안에서 해당되는 주소들을 읽는다. 많은 시스템들이 LibKVM와 같은 라이브러리를 사용하여 커널의 정보를 읽어 들이기 위해 사용된다. 그러나 KVM 관련 루틴들은 리눅스의 라이브러리에 포함되어있지 않으므로 이식성이 높지 않다는 단점이 있다.

3.2.2 /proc filesystem 모니터링을 통한 측정

기본적으로 proc 파일 시스템은 커널이 가지고 있는 여러 가지 데이터 구조체를 시스템 사용자에게 쉽게 전달하기 위해서 사용하는 목적으로 만들어졌다. Proc 파일시스템을 이용하게 됨으로써 보다 쉽게 시스템 정보를 얻어올 수 있으며, 여러 가지 커널 옵션을 특별한 프로그래밍 과정 없이 파일의 정보 변경만을 통해서 쉽게 변경할 수 있도록 해준다. 실제로 proc 파일시스템을 이용하지 않고 커널 데이터 구조체에서 직접 원하는 시스템 정보를 가져올 수 있기는 하지만, 별도의 프로그래밍 과정을 거쳐야 하고, 복잡한 요건을 만족 시켜야 하는 불편함이 있다. 특별히 성능을 중요시 여기지 않는 경우 proc 파일시스템을 이용하여 정보를 가져오는 것으로도 기본적인 모니터링을 수행할 수 있으며, 수집된 정보는 시스템 관리와 시스템 최적화를 위한 시스템 성능 분석 등에 유용하게 사용될 수 있다.

리눅스의 /proc 디렉토리에는 프로세스 정보를 가지는 디렉토리가 존재한다. 이들 프로세스 정보 디렉토리는 각 프로세스의 PID를 이름으로 하며, 디렉토리 안에는 프로세스의 상태 등 프로세스와 관련된 다양한 정보들을 제공한다. 리눅스의/proc 파일시스템은 프로세스의 정보뿐만 아니라 커널이 실행되면서 작성된 각종 정보를 파일로 관리한다. 그리고 /proc/net 디렉토리에는 네트워크와 관련된 네트워크 통계 정보, TCP 소켓 등 네트워크와 관련된 정보들을 가지고 있는 파일들이 있다.

3.2.3 커널의 네트워크 관련 모듈을 통한 측정

커널의 네트워크 관련 모듈로서 넷필터 계층(Netfilter Layer)을 이용하는 방법이 있다[8]. 넷필터는 표준 버클리 소켓 인터페이스의 외부에 존재하는 패킷을 분해하는 프레임워크로서, 크게 세 부분으로 구성되어 있다. 먼저 각각의 프로토콜은 "hooks"라는 것을 정의하며, 이는 패킷 프로토콜 스택의 packet's traversal에 있는 포인터를 의미한다. 이러한 포인터에서, 각각의 프로토콜은 패킷과 훅넘버(hook number)를 이용하여 넷필터 프레임워크를 호출하게 된다. 두 번째로, 커널의 일부분은 각 프로토콜에 대하여 다른 훅을 감시하도록 등록할 수 있다. 따라서 패킷이 넷필터 프레임워크를 통과할 때, 누가 그 프로토콜과 훅을 등록했는지 확인하게 된다. 이러한 것이 등록되어 있다면, 등록된 순서대로 패킷을 검사하고, 패킷을 무시하거나(NF_DROP), 통과시키고(NF_ACCEPT), 또는 패킷에 대한 것을 잊어버리도록 넷필터에게 지시하거나(NF_STOLEN), 사용자 공간에 패킷을 대기시키도록(queuing) 넷필터에게 요청한다(NF_QUEUE). 세 번째 부분은 대기된 패킷을 사용자 공간으로 보내기 위해 제어하는 것으로 이러한 패킷은 비동기방식으로 처리된다. 이러한 저수준 프레임워크와 더불어 다양한 모듈이 작성되어 있으며, 이는 이전 버전의 커널에 대하여 유사한 기능, 확장 가능한 NAT시스템 그리고 확장 가능한 패킷 필터링 시스템을 제공한다.

이러한 넷필터를 이용하여 IP 단위로 연결 추적을 구현하면 각 연결 단위의 자세한 모니터링이 가능해진다. 현재 넷필터는 IP 단위의 연결 추적 커널 모듈인 ip_conntrack을 자체 제공하고 있으며, 이를 통해 IP 단위의 연결에 대한 추적을 가능하게 한다. 그리고 넷필터 확장을 통해 새로운 프로토콜에 대한 지원 및 사용자 함수 등을 추가시킴으로써 보다 세분화된 모니터링을 구현할 수 있다.

본 연구에서는 넷필터 계층의 수정을 통하여 그리드 응용 트래픽의 시스템 자원 사용량에 대한 측정을 수행하였다. 그 이유는 패킷을 캡쳐해서 트래픽을 분석하는 방법은 그 분석과정 자체에서 시스템의 자원을 사용해야 하며, 네트워크 트래픽의 정도에 따라 네트워크 모니터링 작업이 발생시키는 시스템 자원의 부하가 시스템의 사용에 지장을 초래하는 경우도 발생할 수 있다. 이에 비하여 커널 계층의 성능 측정 방법은 커널에서 기본적인 네트워크 정보를 제공할 뿐만 아니라, 운영체제 내의 단계적으로 나뉘어진 네트워크 레이어를 거치며 수집된 정보들을 간단한 처리만으로 사용할 수 있다. 이는 패킷을 캡쳐하여 분석하는 과정에서 발생하는 시스템 자원의 사용을 방지할 수 있다. 성능 비교를 위해 수행한 LibPCAP을 이용한 패킷의 캡쳐와 커널을 이용한 네트워크 자원의 사용량을 측정한 결과치에 의하면 LibPCAP을 사용하는 방식이 CPU를 31.8%를 사용하였으나, 커널을 사용한 측정방식에서는 0.2%로 CPU를 거의 사용하지 않았다. 커널은 기본적으로 해당 시스템의 네트워크 값들을 실시간으로 업데이트 하며 커널의 구조체 혹은 변수에 저장하고 있다. 일반적인 사용자 계층의 모니터링 툴들은 이러한 값들을 주기적인 시스템 콜을 통해 얻어 온다. 사용자 계층에서 수행되는 이러한 수집 과정을 커널 내에서 이루어지게 한다면 이러한 시스템 콜로 인한 시스템 콜 인터럽트를 줄일 수 있다는 장점이 있다.

4. 그리드 응용 성능 측정시스템의 설계 및 구현

리눅스 커널의 수정은 편집기로 직접 소스 코드를 수정하는 방법도 가능하지만, 이 방법은 실수로 인한 시스템의 정지 및 커널 버전 별로 커널 패치를 만들어야 하는 어려움이 있다. 이러한 어려움을 피하기 위해서 본 논문에서는 네트워크 모니터링 기능을 커널 모듈 형식으로 구현함으로써 커널 패치의 위험성을 피할 수 있도록 구현을 하였다. 또한 BSD 소켓 계층 상위에 네트워크 모니터링을 위한 인터페이스를 구현하여 커널 모듈은 해당 인터페이스를 통해 수집한 정보를 사용자 모드에서 사용할 수 있도록 하였다. 그러나, 커널 내에서 네트워크에 대한 분석을 수행하기 위해서는 커널의 수정 혹은 추가 모듈이 필요하다는 단점이 있다.

4.1 그리드 응용 성능 측정시스템의 설계

그리드 응용 성능 측정시스템을 개발하기 위한 개발환경은 다음과 같다.

- 하드웨어

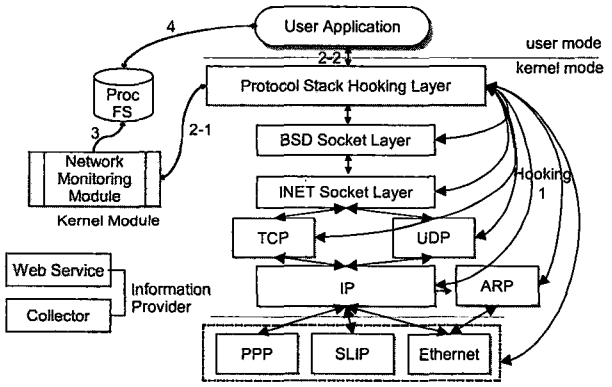
CPU : Intel Pentium III 600MHz
MEMORY : 192MB
DISK : 6.1GB, 3.1GB

- 운영체제

Red hat Linux 7.3
Kernel 2.4.19

- 실행환경
  - Unix C
    Kernel module
    Information collector daemon
  - JAVA(Jakarta Tomcat, Web Service Developer Pack)
    Information Provider Web Service

(그림 2)는 커널 기반 정보 수집자의 그림을 나타낸 것이다.



(그림 2) 커널 기반 정보 수집자

리눅스 커널 기반의 정보 수집자(Information provider)는 기존의 리눅스 커널 내의 자동으로 축적되는 네트워크 관련 정보에 대해서 커널 계층에서 미리 가공된 정보를 제공할 수 있도록 네트워크 정보를 수집, 저장, 가공하는 역할을 한다. 각 기능별 설명은 다음과 같다.

1 : Hooking

기존에 커널에서 수행되던 네트워크 정보 수집과 관련된 로직을 사용자가 원하는 형태의 정보를 수집하도록 구성한 로직으로 대체 수행하는 역할을 한다.

2-1 : Information gathering(kernel module)

각 프로토콜 스택에서 수집하는 정보를 사용자가 사용할 수 있도록 수집하여 저장하는 역할을 한다. Protocol stack hooking layer는 각 프로토콜 스택에서 수집된 네트워크 정보를 커널 모듈이 수집할 수 있도록 제공한다.

2-2 : Information gathering(kernel virtual memory interface)

리눅스에서는 제공되지 않는 방법으로, 일반 유닉스 환경에서는 사용자가 커널이 가지고 있는 정보를 사용할 수 있도록 사용자가 커널에 접근할 수 있도록 "커널 가상 메모리 인터페이스"를 제공한다.

3 : Data accumulating

커널 모듈은 사용자가 정보를 사용할 수 있도록 사용자 레벨에서 접근 가능한 파일 시스템에 저장한다. Proc 파일 시스템은 가상 파일시스템을 사용함으로써 실제 파일시스템을 사용함으로써 발생하는 부하를 줄일 수 있다.

4 : Information processing

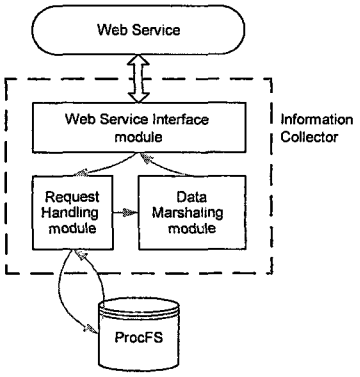
Proc 파일 시스템에 공개된 네트워크 모니터링 값에 대하여 사용자 어플리케이션이 읽어 들여 필요한 인자로 가공하여 사용한다.

정보 수집 시스템의 구성은 Protocol stack hooking layer를 이용하였다. 이는 리눅스 커널 2.4부터 지원하는 넷필터 레이어를 이용한다. 넷필터 레이어는 사용자 함수를 통하여 원하는 프로토콜 스택에 후킹을 할 수 있도록 지원한다. 프로토콜 스택의 코드를 변경하는 것이 아니기 때문에 커널이 사용하는 원본 데이터를 변경시키지 않으면서 정보를 가공하는 것이 가능하다. 정보 제공자에서 사용하는 커널 모듈은 넷필터에서 제공하는 ip_conntrack 모듈을 수정하여 구성하였다. 다음의 그림은 linux-2.4.19/include/linux/ netfilter_ipv4/ip_conntrack.h을 수정하여 들어오고 나가는 패킷에 대한 정보를 수집하기 위한 배열을 추가하는 예를 나타낸 것이다.

```
+#ifdef CONFIG_IP_NF_STATS
+     /* Statistics collected in ip_conntrack_in */
+     u_int64_t pkt_count[IP_CT_DIR_MAX];
+     u_int64_t byte_count[IP_CT_DIR_MAX];
+#endif
```

(그림 3) 커널 모듈 수정의 예

아래의 그림은 정보 수집 데몬을 나타낸 것이다.



(그림 4) 정보 수집 데몬

정보 수집 데몬은 Proc 파일 시스템에 공개되는 정보를 그리드 어플리케이션의네트워크 사용과 관련된 정보로 가공하여 수집하는 데몬이다.

1 : Web Service Interface module

웹 서비스[9]로부터 데이터 요청을 받고, 데이터를 전달해 주는 역할을 한다. 웹 서비스와 같은 호스트 상에 존재하며, 플랫폼에 독립적인 방식으로 웹 서비스와 통신 하며, 웹 서비스와의 통신 방식으로는 데이터그램 전달 방식과 메시지 패싱 방식을 지원한다.

2 : Request Handling module

웹 서비스 인터페이스를 통해 전달 받은 요청을 Proc 파일시스템에서 읽어 들여 Data Marshaling module로 전달하는 역할을 한다.

3 : Data Marshaling module

Request handling module로부터 받은 데이터를 웹 서비스의 프로토콜에 맞추어 XML로 인코딩하는 역할을 한다. Data Marshaling module에서 정의되는 XML 스키마는 그리드 응용 트래픽 모니터링 시스템에서 제공하는 측정 인자들에 해당한다.

수집된 정보들은 웹 서비스를 이용하여 정보를 제공하는데, 수집하는 정보로는 호스트의 기본 정보인 IP 주소 값과 호스트의 운영체제와 커널 버전, 호스트의 네트워크 인터페이스의 속도에 대한 정보를 제공하며, 시스템 정보로는 CPU 정보(User, Kernel 유휴 CPU 점유율), Memory 정보(전체 메모리, 시스템에 사용된 메모리, 사용 가능한 유휴 메모리, 공유 메모리 버퍼 메모리의 양) 등을 제공하며, 이 외에 시스템의 전체 대역폭 및 사용하고 있는 어플리케이션 연결 별 네트워크 대역폭에 대한 정보를 제공한다.

웹 서비스에서 사용하는 프로토콜의 정보는 다음과 같다.

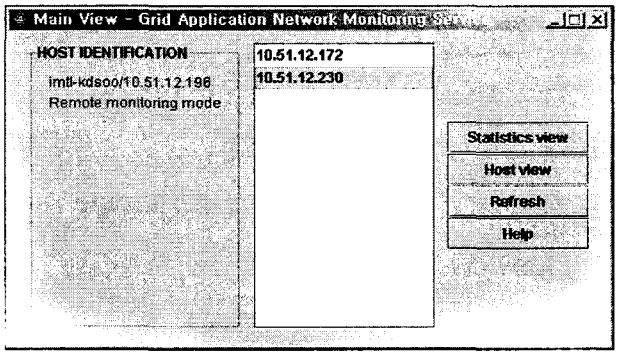
```
<?xml version = "1, 0" encoding = "UTF-8"?>
<Message_root>
    <SystemParam>
        <CPU>
            <user> user </user>
            <sys> system </sys>
            <nice> nice </nice>
            <idle> idle </ idle>
        </CPU>
        <MEM>
            <av> average </av>
            <used> used </used>
            <free> free </free>
            <shrd> shared </shrd>
            <buff> buffer </buff>
        </MEM>
</SystemParam>

<NetworkParam>
        <Traffic>
            <in> incoming bytes/sec </ in>
            <out> outgoing bytes/sec </out>
        </Traffic>
        <Bandwidth>
            <prot> protocol </prot>
            <src> source address </src>
            <dst> destination address </dst>
            <sport> source port </sport>
            <dport> destination port </dport>
            <inpkt> inpacket </inpkt>
            <inbyte> inbyte </inbyte>
            <inband> incoming bandwidth </inband>
            <outpkt> outpacket </outpkt>
```

(그림 5) 메시지 프로토콜 구조

모든 그리드 노드들은 정보 제공자와 모니터링 UI(User Interface)를 갖고 있으며, 이들 중 사용자가 모니터링 UI를 실행하게 되면, UI는 기본적으로 글로버스의 GRAM[10] 서비스에 있는 그리드 노드들의 IP 리스트를 얻어 오게 된다. 이 IP 목록을 기준으로 모니터링 UI를 통해 각 그리드 노드들을 모니터링 하고, 실시간 및 통계 항목들을 열람 할 수 있다. 모니터링 방식은 JAXM(Java API for XML Messaging) runtime을 통해 모니터링 호스트에서 각 그리드 노드들에 대해 SOAP(Simple Object Access Protocol)[11]을 통해 요청하게 되며, 각 그리드 노드들은 요청에 대한 측정값을 JAXM runtime을 통해 모니터링 호스트로 SOAP을 통해 전송한다. 모니터링 UI는 수신한 SOAP을 해석하여 모니터링 결과를 UI에 나타낸다.

4.2 그리드 응용 성능 측정시스템의 구현

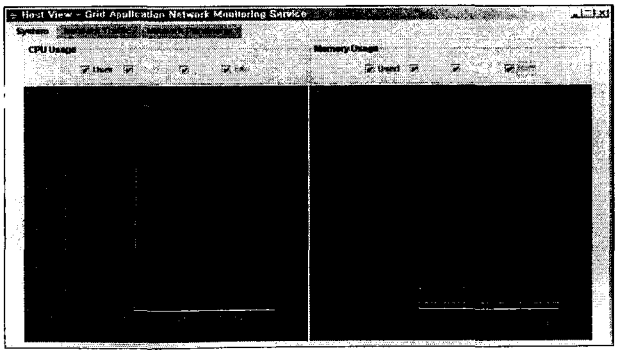
그리드 응용 성능 측정시스템을 구현한 화면은 다음과 같다.



(그림 6) 주 화면의 UI

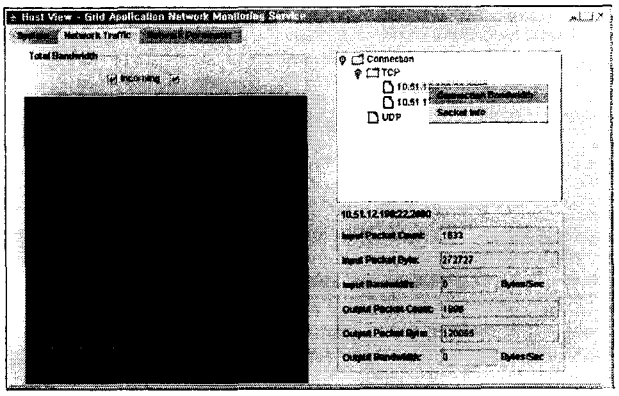
주 화면에서는 글로버스의 GRAM 서비스로부터 얻어 온 그리드 노드들의 IP 주소 목록을볼 수 있으며, 기능적으로는 statistics view (통계 화면)와 host view(개별 호스트 화면)을 실행할 수 있는 버튼이 있다. 또한 그리드 노드들의 IP 주소 목록은 refresh 버튼을 통해 항상 최신으로 유지 할 수 있다.

아래의 (그림 7)은 개별 호스트 모니터링 화면을 나타낸 것이다.



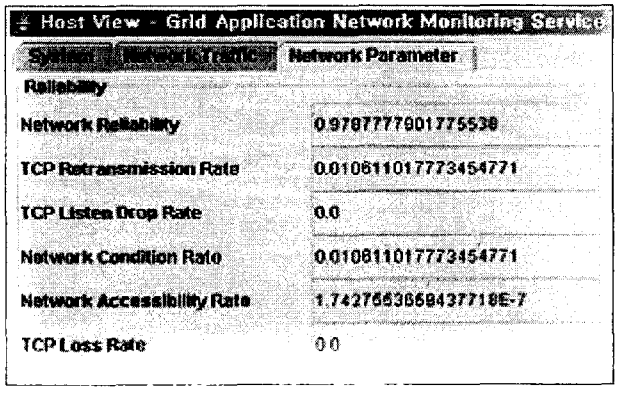
(그림 7) 호스트 화면의 UI

개별 호스트 모니터링 화면에서는 주 화면에서 선택한 IP 주소의 호스트에 대한 모니터링을 수행할 수 있다. 호스트 모니터링 화면은 총 세 개의 모니터링 탭을 가지고 있다. 첫 번째System 탭은 (그림 7)에서 보여지는 것과 같이 개별 호스트의 CPU 사용량을 user, nice, system, idle의 분류로 나누어 보여주며, 메모리 사용량을 사용된 메모리, 유휴 메모리, 버퍼 메모리의 양을 모니터링 할 수 있다. CPU 는 각 모드 별 CPU 점유 시간을 퍼센트(%) 비율로 나타내며, 메모리 사용량은 시스템의 전체 물리 메모리의 양을 퍼센트(%) 비율로 환산하여 보여준다. 이 데이터들은 1초에 한번 갱신되는 실시간 데이터이다. 두 번째로 Network Traffic 탭은 개별 호스트의 네트워크 사용량을 모니터링 할 수 있는 화면을 아래의 (그림 8)과 같이 제공한다.



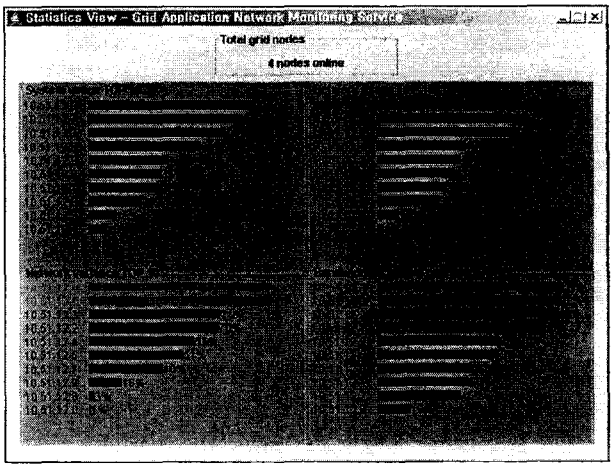
(그림 8) 네트워크 트래픽 모니터링 UI

세 번째로Network Parameter 탭이 제공된다. 해당 탭에서는 아래의 (그림 9)와 같이 그리드 네트워크 상에서 어플리케이션의 네트워크 사용 품질을 모니터링 하는 것이 가능하다.



(그림 9) 네트워크 신뢰성 측정 UI

이외에도, 주 화면의 UI의 Statistics view가 통계 정보를 제공한다. 이 화면에서 첫 번째 항목은 System usage Top 10으로, 그리드 도메인 상의 시스템들 중 CPU 와 메모리의 사용률이 가장 높은 순으로 나열을 한다. 이를 통해 사용률이 비정상적으로 높은 호스트에 해단 발견을 할 수 있다. 두 번째 항목은 Network usage Top 10이며, 그리드 도메인 상의 시스템들 중 네트워크 사용률이 가장 높은 순으로 나열을 한다. 네트워크 사용률이 가장 높은 호스트란 모니터링 중에 가장 많은 양의 네트워크 트래픽을 발생시키고 있는 호스트를 말한다. 세 번째로 Network variance Top 10이고, 그리드 도메인 상의 시스템들 중 네트워크 전송양의 변화가 가장 심한 순서로 나열한다. 이는 그리드 어플리케이션의 특성상 비정상적인 네트워크 사용을 하는 어플리케이션을 발견해내는데 도움을 준다. 마지막으로, Bandwith usage percentage Top 10 항목이며, 그리드 도메인 상의 시스템들 중 자신에게 허용된 네트워크 대역폭을 가장 많이 사용하는 호스트 순서로 나열한다. 이는 특정 노드의 대역폭 점유율이 지속적으로 높은 값을 유지할 때 해당 호스트의 대역폭을 증설 해줄 필요가 있음을 의미한다.



(그림 10) 통계정보 UI

5. 결론 및 향후 계획

그리드 네트워크는 지리학적으로 분산되어 있는 고성능 컴퓨팅 자원을 네트워크로 상호 연동하여 조직과 지역에 관계없이 사용할 수 있는 네트워크 환경을 말한다. 이러한 환경에서 프로그램의 수행을 요청한 사용자에게 수행되고 있는 응용 프로그램에 대한 자원 사용량에 관해서 정보를 제공하는 것은 성능 관리 면에서 필수 불가결한 요소이다. 그러나 이러한 정보를 제공하기 위해서는 그리드 네트워크 사이에 주고 받는 패킷에 대한 캡처를 통해서 정보를 분석해야 하는데, 이를 위해서는 tcpdump 등의 프로그램을 사용하여야 한다. 그러나 이러한 응용계층에서의 패킷을 캡처하는 방식은 3.2에서 제시하였듯이 자체적으로 많은 시스템 자원을 사용하게 되므로, 모니터링 시스템을 실제적으로 구현하는 데에는 많은 어려움이 있다.

이에 본 연구에서는 효율적으로 분산된 네트워크 환경에서의 모니터링 방법에 대한 연구를 수행하여, 커널 기반의 넷필터 계층의 ip_conntrack 모듈을 수정하여 네트워크 및 시스템 자원의 사용량에 대한 측정을 수행하였다. 이 방식의 장점으로는 커널이 가지고 있는 네트워크 정보를 이용함으로써 패킷을 캡처하고 분석하는데서 발생하는 부하가 없고, 패킷 캡쳐만으로는 어려웠던 프로토콜별 상세 정보에 대한 모니터링을 가능하게 한다는 것이다.

향후 연구계획으로는 그리드 네트워크에서 수행되는 어플리케이션에 대해서 측정한 정보를 바탕으로 SLA(Service Level Agreement)의 제공방안 및 효율적인 자원할당 알고리즘에 대해서 연구를 수행할 계획이다.

참 고 문 헌

[1] F. Berman, G. Fox, T. Hey, The Grid : past, present, future, Grid Computing - Making the Global Infrastructure a Reality, Wiley and Sons.

[2] Brain L. Tierney, "End-to-End Application Monitoring using the Distributed Monitoring Framework," Lawrence Berkeley National Laboratory, 2002.

[3] Mark Gates, Ajay Tirumala, Jon Dugan, Kevin Gibbs, "Iperf User Docs," NLANR application support, March, 2003, http://dast.nlanr.net/Projects/Iperf/iperfdocs_1.7.0. html.

[4] www.faqs.org/rfcs/rfc1889.html.

[5] Ajay Triumala, Les Cottrell, "Iperf QUICK mode," IEPM, July, 2002, http://www-iepm.slac.stanford.edu/bw/iperf_res.html.

[6] Craig A. Lee, James Stepanek, Carl Kesselman, Rich Wolski, Ian Foster, "A Network Performance Tool Grid Environments," Proc. of Supercomputing '99, Nov., 1999.

[7] http://ee.lbl.gov/.

[8] Netfilter project : http://www.netfilter.org.

[9] Haas, H., Orchard, D. : Web Services Architecture Usage Scenarios, W3C Working Draft 30 July, 2002.

[10] K. Czajkowski, S. Fitzgerald, I. Foster and C. Kesselman, "Grid Information Services for Distributed Resource Sharing," IEEE International Symposium on High Performance Distributed Computing, (2001), IEEE Press.

[11] Nicholas Chase, "Send and receive SOAP messages with SAAJ," IBM DeveloperWorks, July, 2003.

[12] Ajay Tirumala, Les Cottrell, Tom Dunigan, "Measuring end-to-end with Iperf using Web100," PAM2003, April, 2003.

[13] R. L. Cottrell, C. Logg, "Experiences and Results form a New High Performance Network and Application Monitoring Toolkit, SLAC, Nov., 2002.

김 태 경

e-mail : tkkim@rtlab.skku.ac.kr

1997년 단국대학교 수학교육(학사)

2001년 성균관대학교 정보통신공학 (석사)

1996년~1997년 기아정보시스템사원

1997년~2001년 서울신학대학교 종합전산실 주임대리

현재 성균관대학교 정보통신공학부 박사과정 수료

관심분야 : 그리드 네트워크, 네트워크 보안, 모바일 에이전트, 모바일 그리드

김 동 수

e-mail : tkkim@rtlab.skku.ac.kr

2003년 성균관대학교 전기전자컴퓨터 공학부(학사)

2003년~현재 성균관대학교 컴퓨터공학과 석사과정

관심분야 : 그리드 네트워크, CMS, 호스트 보안, OS Kernel

변 옥 환

e-mail : ohbyeon@kisti.re.kr

1979년 한국항공대학교 통신정보공학과 (학사)

1993년 경희대학교 전자공학과 공학박사

1978년~1995년 KIST 시스템공학연구소 책임연구원, 연구전산망개발실장, 슈퍼컴퓨팅연구실장

1995년~1999년 ETRI 슈퍼컴퓨팅센터 책임연구원, 고성능망연구실장, 슈퍼컴퓨팅연구실장

1999년~현재 KISTI 슈퍼컴퓨팅센터 책임연구원, 초고속연구망부장, 슈퍼컴퓨팅인프라개발실장

관심분야 : 고성능망 관리 및 보안, 그리드 네트워킹 및 협업액세스그리드, 유비쿼터스 네트워킹 및 모바일 그리드

정 태 명

e-mail : tmchung@ece.skku.ac.kr

1981년 연세대학교 전기공학(학사)

1984년 University of Illinois Chicago, 전자계산학과 (학사)

1987년 University of Illinois Chicago, 컴퓨터공학과 (석사)

1995년 Purdue University, 컴퓨터공학 박사

1985년~1987년 Waldner and Co., System Engineer

1987년~1990년 Bolt Bernek and Newman Labs., Staff Scientist

현재 성균관대학교 정보통신공학부 부교수

관심분야 : 실시간시스템, 네트워크 관리, 시스템 보안, 네트워크 보안, 전자상거래