실시간 네트워크 모니터링을 적용한 PDP 시스템의 성능 평가

송 은 하† · 정 재 홍† · 정 영 식††

요 약

인터넷 기반 분산/병렬 처리 시스템인 PDP(Parallel/Distributed Processing)는 인터넷의 유휴상태 호스트들을 이용하여 대용량 작업을 병렬로 처리해서 전체 수행 시간을 감소시킨다. 본 연구에서는 실시간 네트워크 모니터링을 활용하여 수시로 변화하는 네트워크 환경에 적응하여 병렬/분산 처리되는 방안을 제안한다. 실시간 네트워크 모니터링 정보를 PDP 주요 핵심 알고리즘들에 적용하여 네트워크 과부하 및 결함으로 발생하는 작업 지연 요소에 적응적으로 대처함으로써 전체 성능이 향상됨을 보인다.

Performance Evaluation of PDP System Using Realtime Network Monitoring

Eun-Ha Song† · Jae-Hong Jeong† · Young-Sik Jeong††

ABSTRACT

PDP(Parallel/Distributed Processing) is an internet-based parallel/distributed processing system that utilizes resources from hosts on the internet in idle state to perform large scale application through parallel processing, thus decreasing the total execution time. In this paper, we propose an adaptive method to be changed network environment at any time using realtime monitoring of host. It is found from experiments that parallel/distributed processing has better performance than its without monitoring as an adaptive strategy, which copy with task delay factor by overload and fault of network, be applicable to the cockpits of task allocation algorithm in PDP.

키워드 : 병렬/분산 시스템(Parallel/Distributed System), 분산 모니터링(Distributed Monitoring), 네트워크 모니터링(Network Monitoring), SNMP

1. 서 론

최근에 대용량 어플리케이션을 수행하기 위해 인터넷의 많은 유휴 컴퓨팅 자원을 사용하는 분산/병렬 처리에 대한 연구가 진행되고 있다[1-8]. 구체적으로 분산/병렬 시스템은 다수의 컴퓨팅 자원에 대한 효율적인 관리, 이질적인 컴퓨팅 자원의 사용, 컴퓨팅 자원의 오류 발생시 처리 방법, 웹 기술을 기반으로 어플리케이션의 병렬 처리를 위한 프로그래밍 모델 제공 등에 대한 연구가 진행되어 왔다. 그러나 부하 균등 및 작업 재할당 정책에서 네트워크 환경을 고려한 연구는 미흡하다. 현재의 인터넷은 VOD 서비스, P2P 서비스를 통한 대용량 멀티미디어 파일의 송·수신, 그밖에 고부가 서비스로 인해 많은 트래픽이 발생하고 있다. 이러한 네트워크에 발생하는 트래픽은 인터넷 기반 분산/병렬 시스템에서 처리 결과의 전송 지연으로 전체 작업 지연을 초래할 수 있다. 또한, 작업에 참여한 특정 호스트의 작업 지연으로 작업이 완료된 호스트들이 전체 작업 완료까지 대기함으로써 컴퓨팅 자원이 낭비되는 결과를 가져올 수 있다.

네트워크에 발생하는 트래픽으로 인해 작업 수행의 지연이나 컴퓨팅 자원의 낭비는 참여한 호스트들에게 할당될 작업량에 차이를 둠으로써 해결할 수 있다. 네트워크 트래픽으로 작업 수행 지연이 예상되는 호스트에게는 적은 작업 할당으로 작업 수행시간을 단축할 수 있다. 그렇지 않은 호스트는 더 많은 작업 할당으로 작업 수행시간을 증가시킨다. 결과적으로 네트워크 트래픽이 발생한 호스트의 작업량은 적게 할당하고 그렇지 않은 호스트의 작업량은 많이 할당함으로써 전체 작업 수행시간을 단축시킨다.

본 논문에서는 이러한 네트워크에 발생하는 트래픽이 분산/병렬 시스템에 어떠한 영향을 미치는지 알아보고 그에 따른 해결방안으로 네트워크 모니터링 기반 작업할당 전략

※ 본 연구는 정보통신부 지원 ITRC 프로그램의 지원을 받아 수행되었음.
† 준 회 원 : 원광대학교 대학원 컴퓨터공학과
†† 종신회원 : 원광대학교 컴퓨터 및 정보통신공학부 교수
논문접수 : 2003년 12월 5일, 심사완료 : 2004년 6월 11일

을 제안한다. 웹 기반 분산/병렬 시스템에 자원을 제공하는 호스트가 속해 있는 서브 네트워크의 트래픽을 모니터링할 수 있는 실시간 네트워크 모니터인 RNM-PDP(Realtime Network Monitor for Parallel/Distributed Processing) 구현을 통해서 사용 가능한 호스트의 네트워크 대역폭을 중심으로 작업을 할당한다.

논문의 구성은 제 2장에서는 기존의 네트워크 모니터링 방법과 관련된 기술들을 살펴보고 제 3장에서는 PDP 시스템에서 사용하는 작업할당 알고리즘과 호스트 관리 알고리즘에 대해서 알아본다. 제 4장에서는 네트워크 트래픽 측정을 위한 모니터링 요소 분석 및 RNM-PDP 설계, 구현에 대해서 설명한다. 제 5장에서는 테스트베드를 구축하여 본 논문에서 제시한 메커니즘에 대한 성능평가를 수행한다.

2. 관련 연구

본 절에서는 기존의 네트워크 모니터링 도구를 살펴보고 도구의 기능과 모니터링 방법들을 소개한다.

MRTG(Multi-Router Traffic Grapher)[9]는 네트워크 링크간의 트래픽 부하량을 측정하는 도구로써 HTML을 이용하여 네트워크 트래픽을 GIF 이미지로 생성하여 파악한다. MRTG는 Perl과 C언어를 사용하여 만들어졌으며 여러 다양한 유닉스 플랫폼과 윈도우에서 사용되고 있다.

Ntop[10]은 Deri Luca가 오픈 프로젝트로 개발하고 있는 네트워크 트래픽 모니터링 및 분석 시스템이다. Ntop은 유닉스 시스템 자원 상태를 보여주는 top 명령어를 네트워크 모니터링 및 분석 시스템에 응용하였다. 웹 기반에서 호스트, 프로토콜, 어플리케이션 정보 등을 분석해 준다. 호스트 위주로 분석 정보를 제공하여 특정 호스트에서 나가고 들어오는 패킷의 양이라든가 특정 호스트에서 사용되는 프로토콜을 확인하였다.

Etherfind[11]는 소프트웨어 패킷 모니터링 도구로서 네트워크 인터페이스를 통해 해당 호스트로 오는 패킷은 물론 그렇지 않은 패킷까지 모두 읽어 들여 필요한 정보를 얻어서 파일로 저장하는 역할을 한다. 여기서 제공되는 정보에는 프로토콜 타입, 크기, 패킷의 발신지와 수신지 등이 있다. 그렇지만 이것은 텍스트 기반의 인터페이스로 작동하기 때문에 불편한 점이 있고 시스템에 관해 특정한 권한이 있는 사용자만이 이용할 수 있고, 분석 기능을 제공하지 않고 있어 트래픽 정보를 알아내기 위해서는 사용자가 다시 한번 데이터를 기반으로 정보를 조합해 내는 과정이 필요하다.

NFSwatch[12]는 NFS 파일 서버의 기능을 모니터링하기 위해 만든 것으로 현재 들어오는 모든 네트워크 트래픽을 여러 가지로 분류해서 보여준다. 기본적으로는 NFSwatch는 자기 자신을 향하는 트래픽 외에도 특정 호스트를 지정하면 해당 호스트로 향하는 트래픽 정보까지 보여줄 수 있는 기능을 가지고 있다는 것이 특징이다. NFSwatch는 어느 특정 운영체제에 종속적인 방법을 취하고 있어 운영체제에 대한 독립성이 없고, 텍스트 기반이고 분석 기능을 제공하지 않는다.

TCPdump[13]는 인터넷 게이트웨이 성능을 높이기 위한 측정 도구로 처음 개발되었는데 최근에 개발된 버전은 libpcap이라는 운영체제 독립적인 패킷 캡처링 라이브러리를 이용하여 다양한 운영체제에서 널리 사용되고 있다. TCPdump는 네트워크 상에서 전송되는 패킷의 헤더 정보를 분석하여 보여주는 역할을 하는데 사용자는 그 정보를 수동으로 분석하여 네트워크 상태를 파악한다. 따라서 장시간의 네트워크 상태를 파악하는 데는 적당하지 않고 텍스트 기반에 분석 기능을 제공하지 않는다.

Argus[14]는 IP 네트워크의 감사 도구로 지정한 네트워크 인터페이스 디바이스를 읽어 들여 해당 인터페이스로 들어오는 모든 패킷을 분석하고 패킷의 정보를 저장한다. 이렇게 저장된 정보는 다양한 형태로 가공되어 사용자에게 보여 지기 때문에 주로 네트워크 보안 및 설정에 문제가 있거나 해킹 시도 감지 등에 용도로 사용되고, 정보는 텍스트 기반으로 제공한다.

기존의 네트워크 모니터링 도구들은 네트워크 관리자를 위한 도구로서 네트워크 트래픽을 분석하여 어느 호스트나 어느 응용 프로그램이 많은 트래픽을 유발하고 있는지를 파악하여 한정된 네트워크 자원을 효율적으로 사용하는데 목적을 두고 있다.

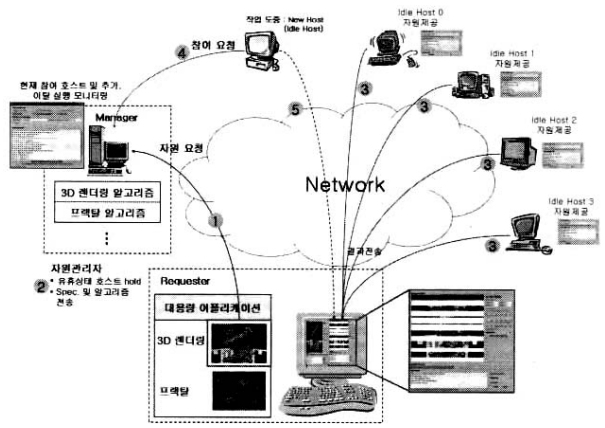
본 논문에서는 네트워크의 효율적인 사용보다는 인터넷 기반 병렬 분산 처리 프레임워크인 PDP(Parallel/Distributed Processing)에서 작업에 참여하는 호스트들의 네트워크 사용률을 실시간으로 모니터링하여 모니터링 결과를 각 호스트의 작업할당에 반영하여 보다 효율적으로 분산/병렬 처리가 되도록 전체 작업 수행시간을 줄이는데 목적이 있다.

3. PDP 시스템

PDP(Parallel Distributed Processing) 시스템은 네트워크 상에 존재하는 유휴 컴퓨팅 자원을 활용하여 많은 연산 수행을 필요로 하는 프랙탈이나 렌더링 같은 이미지 프로세싱 작업을 수행하는 분산/병렬 처리 시스템이다[16]. PDP 시스템은 전체 작업 알고리즘 결정 및 작업에 참여한 자원을 관리하는 관리자(Manager), 작업 수행에 자원을 제공하는 유휴 컴퓨팅 자원인 호스트(Host), 작업을 수행하기 위해서 자원을 요청하는 요청자(Requester)로 구성되어 있다. PDP의 전체 구성과 제어 흐름은 (그림 1)과 같다.

요청자는 대용량 어플리케이션을 수행하기 위해서 관리자에게 자원 요청을 하고 유휴 컴퓨팅 자원인 호스트들은

관리자에게 작업의 참여를 알린다. 요청자에게 자원의 요청을 받는 관리자는 작업 처리를 위한 알고리즘을 결정하고 작업을 분할해서 참여한 호스트에게 전달하고 호스트에게 작업 시작을 알린다. 작업 처리를 수행하는 호스트는 수행된 작업의 결과를 요청자에게 전달하고 요청자는 전달된 작업을 조합함으로써 작업이 수행된다.



(그림 1) PDP 구성과 제어 흐름

PDP는 크게 작업 할당 알고리즘과 호스트 관리 알고리즘을 가지고 있다. 작업할당 알고리즘은 균일 작업할당(Uniform Task Allocation : UTA), 성능 기반 작업할당(CPU Performance Task Allocation : CPTA), 적응적 작업할당(Static Adaptive Task Allocation : S_ATA)로 세분화되며, 호스트 관리 알고리즘은 정적 관리 알고리즘과 동적 관리 알고리즘이 있다.

〈표 1〉 PDP 알고리즘

호스트 관리 알고리즘	작업 할당 알고리즘	참여 호스트 수
정적 관리	균일 작업 할당(UTA)	고 정
	성능 기반 작업 할당(CPTA)	
	적응적 작업 할당(S_ATA)	
동적 관리	적응적 작업 할당(D_ATA)	가 변

UTA는 호스트의 성능에 상관없이 균등하게 작업을 할당하는 알고리즘이다. 작업도중 호스트의 상태를 고려하지 않으며 재할당이 이루어지지 않는다. CPTA는 호스트의 벤치마킹을 통한 성능을 고려하여 작업을 할당한다. UTA보다 빠른 수행시간을 보이지만 벤치마킹 때 기본적인 호스트의 성능을 작업이 수행하는 동안에 계속적으로 유지한다는 보장이 없다. S_ATA는 호스트의 성능 기반 할당으로 이루어지며 작업 중 작업을 완료한 호스트는 성능이 느린 호스트의 작업을 재할당하여 성능 기반 할당보다는 빠른 수행시간을 보인다. 재할당시 호스트 성능을 초기 벤치마킹 값으로 판단하기 때문에 시간이 지남에 따라 호스트의 성능 및 상태 변화에 대처하지 못한다. D_ATA는 작업도중 호스트의 참여, 이탈을 고려하여 작업 중 이탈한 호스트의 작업을 재할당하여 호스트 결함에 대응하는 알고리즘이다.

4. 실시간 네트워크 모니터

네트워크를 모니터링 하는 방법에는 크게 Libpcap[16] 라이브러리 등을 사용하여 패킷을 캡쳐하는 방법과 SNMP[15]를 사용하여 네트워크 모니터링 정보를 읽어 오는 방법이 있다. Libpcap 라이브러리 등의 패킷을 캡쳐하는 방법을 사용하면 응용 계층까지 분석할 수 있어 네트워크 트래픽의 총량 뿐만 아니라 호스트 정보, 프로토콜 정보, 서비스 정보 등을 확인 할 수 있다. 그러나 이러한 방법은 대용량 트래픽이 시스템에 들어오면 디바이스 드라이버는 패킷을 감지하면 인터럽트를 걸어 커널에 알리고 들어오는 모든 패킷을 커널의 패킷 메모리에 복사해야 하므로 CPU 과부하가 발생하고 커널의 패킷 메모리가 부족해지는 문제를 가지고 있다.

SNMP를 사용하는 네트워크 관리에는 관리국, SNMP 에이전트, MIB(Management Information Base), NMP(Network Management Protocol)로 구성되어 있다. 관리국은 네트워크 관리자로써 네트워크 정보를 요구한다. SNMP 에이전트는 관리국으로부터의 정보 요구나 특정 동작 요청에 응답하고, 예기치 않은 정보를 제공한다. MIB는 오브젝트 모임을 의미하며, 네트워크내 관리되는 대상 자원을 오브젝트로 표현한다. NMP는 관리국과 에이전트를 연결한다. SNMP를 이용해서 네트워크를 모니터링하는 방법은 링크 계층까지만 분석할 수 있다는 단점이 있다. 그러나 간단한 동작으로 트래픽의 총량이나 사용 가능한 대역폭에 관한 정보를 확인할 수가 있어, 본 논문에서 네트워크 모니터링 데이터 추출 방법으로 사용한다.

4.1 모니터링 요소 결정 및 추출

네트워크 성능을 모니터링할 수 있는 요소는 <표 2>와 같다. 여러 가지 요소들을 분석하여 네트워크의 성능을 평가할 수 있다. 본 논문에서 개발하고자 하는 네트워크 모니터는 호스트 네트워크의 사용 가능한 대역폭을 알아내어 호스트에게 적절하게 작업을 할당하는 것이다. 그러므로 네트워크 성능 모니터링 요소 중에서 *Utilization*과 *Bandwidth*를 네트워크 평가 요소로 사용한다. SNMP를 사용하여 호스트의 네트워크 *Utilization*과 *Bandwidth*를 계산할 수 있는 요소들을 추출한다. 본 논문에서는 네트워크 성능이 높다는 것은 네트워크 대역폭 이용률이 낮다는 것을 의미한다.

〈표 2〉 네트워크 성능 모니터링 요소

구 분	모니터링 요소	설 명
Throughput	Utilization	현재 사용하고 있는 대역폭
	Bandwidth	네트워크의 최대 대역폭
	Capacity	송수신 가능한 옥텟 단위의 가장 큰 IP 데이터 그램
Delay	One way delay	근원지로부터 목적지까지의 도달 시간
	RTT delay	근원지로부터 목적지를 경유하여 다시 돌아오는 시간
	Jitter	선행 지연과의 변이율(최대/최소/평균)
Loss	RTT loss	근원지로부터 목적지를 경유하여 다시 돌아오는 도중에 패킷의 손실
	RT loss	근원지로부터 목적지까지 도달하는 동안의 패킷 손실

사용 가능한 호스트 네트워크 대역폭을 계산하기 위해서는 호스트 네트워크의 대역폭과 시간당 입·출력 데이터의 크기를 추출해야 한다. <표 3>은 이러한 값들을 얻기 위해 사용하는 MIB 오브젝트이다. 관리국은 일정시간 간격으로 SNMP 폴링을 통해서 에이전트의 MIB에서 가용 가능한 호스트 네트워크 대역폭 계산에 필요한 값들을 추출한다. 이때 SNMP 폴링에서 발생하는 데이터의 양은 극히 적으므로 호스트의 네트워크 트래픽에 영향을 주지 않는 것으로 간주한다. 얻어진 값들은 누적된 값들이므로 n번째 얻은 값에서 n-1번째 얻은 값의 차를 구함으로써 사용 가능한 호스트 네트워크 대역폭 계산에 필요한 값들을 얻는다.

〈표 3〉 대역폭 분석을 위한 MIB-II 오브젝트

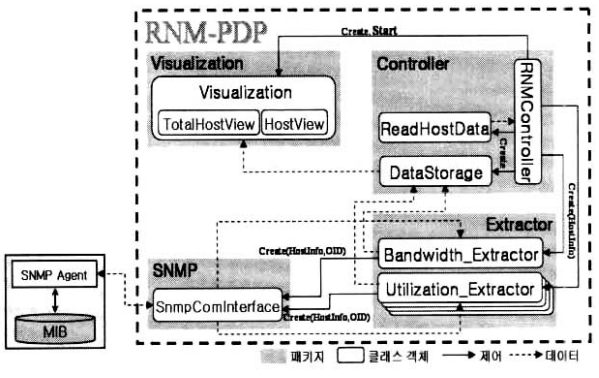
MIB-II 오브젝트(OID)	설 명
system.sysUpTime (1.3.6.1.2.1.1.3)	시스템 가동 후 경과된 시간 (1/1000초)
interfaces.ifTable.ifEntry.ifSpeed (1.3.6.1.2.1.2.2.1.5)	시스템 인터페이스의 대역폭
interfaces.ifTable.ifEntry.ifInOctets (1.3.6.1.2.1.2.2.1.10)	시스템 가동 후 인터페이스에 도착한 전체 옥텟 수
interfaces.ifTable.ifEntry.ifOutOctets (1.3.6.1.2.1.2.2.1.16)	시스템 가동 후 인터페이스로 전송된 전체 옥텟 수

5. RNM-PDP 시스템

5.1 RNM-PDP 구조

RNM-PDP는 PDP와 호환성을 위해서 Java로 구현하였고 전체적인 시스템 구조는 (그림 2)와 같이 구성된다.

RNM-PDP의 구조는 Visualization, Controller, Extractor, SNMP 패키지로 구성이 된다. Visualization 패키지는 관리자 인터페이스 부분으로 두 가지의 뷰를 제공하기 위한 TotalHostView와 HostView 클래스를 포함하고 있다. TotalHostView 클래스는 모니터링 되는 전체 호스트의 네트워크 사용률을 비교할 수 있는 뷰를 제공하며, HostView 클래스는 선택된 호스트의 네트워크 사용률, 송·수신되는 옥텟의 수 및 현재 모니터링되고 있는 호스트의 대역폭, IP 주소 등의 정보를 제공한다. RNMController는 RNM-PDP의 전체 동작을 제어 하는 부분으로 사용자 인터페이스 역할을 하는 Visualization, ReadHostData와 DataStorage 클래스 객체를 생성한다. 또한, PDP로부터 호스트 정보를 받아서 Extractor 패키지의 Bandwidth_Extractor와 Utilization_Extractor 클래스 객체를 생성하고 Visualization 클래스에 start 메시지를 호출하는 역할을 수행한다. Bandwidth_Extractor 클래스는 호스트의 네트워크 대역폭 정보추출 기능을 수행하는 것으로 호스트가 추가될 때마다 한번만 호출을 하지만 Utilization_Extractor 클래스는 쓰레드를 사용하여 호스트 대역폭 사용률 계산에 필요한 요소를 추출하여 사용률을 계산하며, 호스트가 추가될 때마다 생성된다. ReadHostData는 객체 생성과 함께 작업에 참여할 호스트들의 네트워크 구조 정보를 미리 작성된 HostInfo 파일에서 읽어와 RNMController에 반환한다. HostInfo 파일에는 작업에 참여할 호스트의 IP 주소, 호스트가 속한 네트워크가 연결된 스위치 IP 주소, 스위치의 인터페이스 번호가 기록되어 있다. DataStorage 클래스는 데이터를 저장하기 위한 데이터 타입들을 정의하는 클래스로서 static 객체로 생성이 된다. Bandwidth_Extractor와 Utilization_Extractor 클래스는 각각 SNMP 패키지에 ComInterface 클래스 객체를 생성하여 호스트의 네트워크 모니터링 정보를 추출한다. ComInterface 클래스는 SNMP 에이전트와 소켓을 연결하고 모니터링 정보를 추출하기 위해 SNMP 메시지를 생성하며, 에이전트로부터 수신된 메시지를 분석하여 메시지에서 해당 MIB 오브젝트의 값을 추출하여 반환하는 기능을 수행한다.

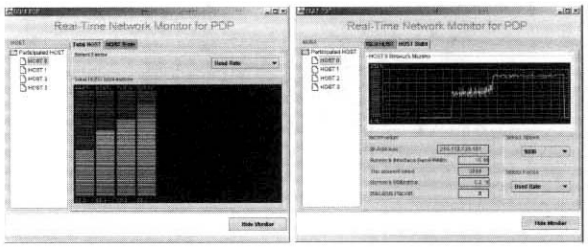


(그림 2) RNM-PDP 구조

5.2 모니터링 정보 시각화

RNM-PDP에서 관리자 인터페이스는 작업에 참여한 호스트들을 보여주는 호스트 리스트 트리와 모니터링된 호스트 네트워크의 상태를 시각화한 두 가지 뷰를 포함하는 시각화 패널로 구성된다. 시각화 패널은 참여한 전체 호스트에 네트워크 사용률을 비교해서 볼 수 있는 Total Host와

호스트 리스트 트리에서 선택된 호스트의 네트워크 사용률을 보여주는 Host State 패널로 구성되어 있다.



(a) (b)

(그림 3) RNM-PDP 모니터링 정보의 시각화

Total Host 패널은 (그림 3)(a)와 같이 참여한 호스트들의 네트워크 사용률을 비교해서 볼 수 있도록 막대 그래프 형태의 시각화를 제공하는 뷰를 포함한다. Host state 패널은 (그림 3)(b)와 같이 호스트 리스트 트리에서 선택된 호스트의 시간에 따른 네트워크 사용률의 변화를 볼 수 있도록하는 꺾은 선형 그래프 형태의 뷰와 텍스트 형태로 호스트 IP 주소, 네트워크 대역폭과 호스트 네트워크 사용률에 대한 정보를 제공한다.

5.3 네트워크 모니터링 기반 PDP 시스템의 작업 할당 알고리즘

모니터링된 호스트의 네트워크 대역폭을 이용하여 PDP 시스템의 작업에 참여한 호스트에게 할당되는 작업량 결정식은 다음과 같다.

$$Host_i Job = \frac{B_i + N}{\sum_{j=0}^{NmHost-1} (B_j + N)} \cdot TotalJob$$

여기서, $Host_i Job$: $Host_i$가 할당받을 작업의 양

$TotalJob$: 전체 수행해야할 작업의 양

B_j : $Host_i$의 사용 가능한 대역폭

N : 작업할당 상수

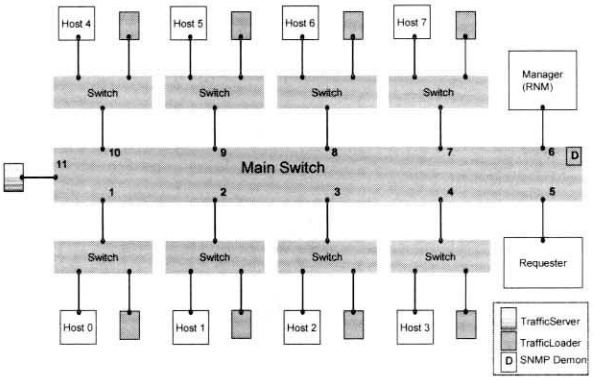
$NmHost$: 작업에 참여한 전체 호스트의 수

작업할당 상수는 호스트의 사용 가능한 네트워크 대역폭을 작업할당에 적절하게 적용하기 위한 역할을 한다. 할당 상수의 값은 안정 상태 성능평가를 통해서 얻었으며 할당 상수의 값이 너무 적게 되면 사용 가능한 네트워크 대역폭이 적은 호스트에게 지나치게 적은 양의 작업을 할당하고 사용 가능한 네트워크 대역폭이 큰 호스트에게는 많은 작업을 할당해서 사용 가능한 네트워크 대역폭이 적은 호스트의 작업이 너무 일찍 끝나게 된다. 반면에 할당 상수의 값이 너무 크게 되면 반대의 경우가 발생하여 이용 가능한 네트워크 대역폭에 따른 작업할당 의미가 퇴색하게 한다.

6. 성능평가

6.1 테스트베드

외부 네트워크 환경에 영향을 받지 않고 성능평가를 수행하기 위해서 (그림 4)와 같이 테스트베드를 구축하였다.



(그림 4) 테스트베드

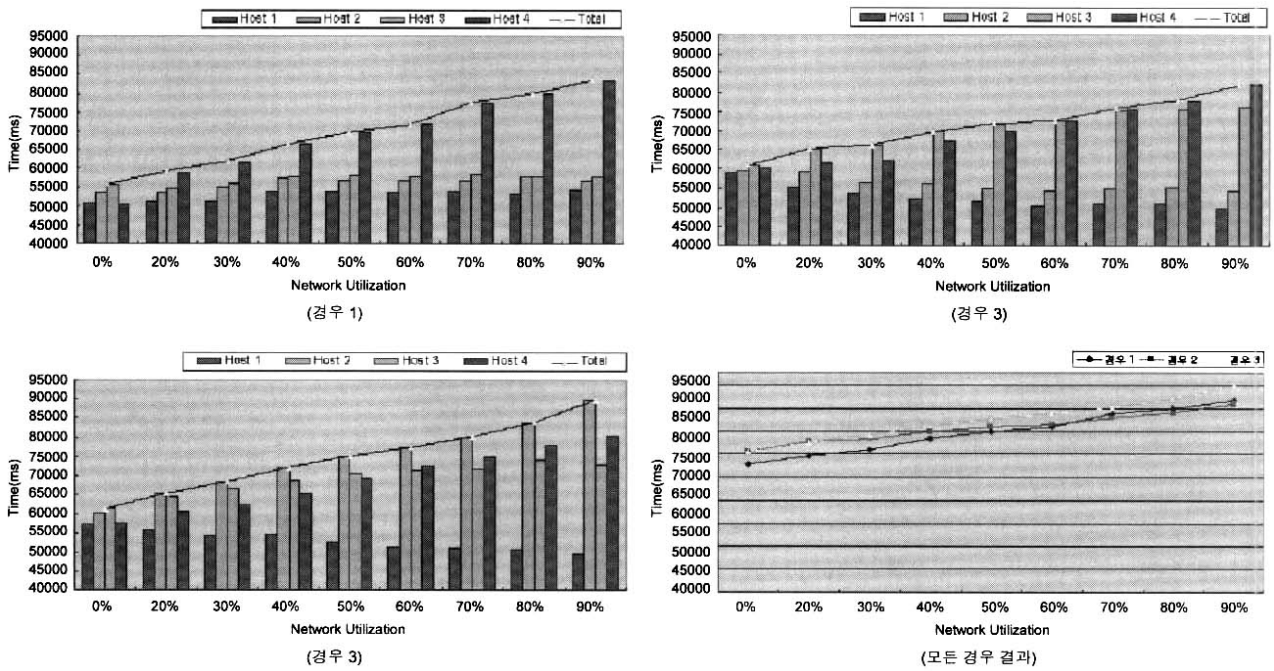
성능평가를 위해 구축된 테스트베드에 각 호스트들은 시스템 성능에 따라 발생하는 작업 수행시간의 차이를 최소화하기 위해 Pentium 4 1.3GHz와 256MB 메모리 및 모든 하드웨어를 동일하게 사용하였다. 따라서, 네트워크 사용률에 따른 수행시간의 차이를 정확히 확인하여 보다 정확한 성능평가를 수행한다. 모든 호스트는 호스트와 트래픽 로더를 스위치에 연결해서 실험을 위한 서브 네트워크를 구성하였다. 이러한 8개의 서브 네트워크를 SNMP 데몬이 수행되는 메인 스위치에 연결하였다. 구축된 서브 네트워크의 스위치와 메인 스위치는 10Mbps 대역폭에 반이중 방식으로 한다. 요청자는 작업을 수행하는 호스트로부터 많은 양의 데이터를 수신하고 트래픽 서버는 트래픽 로더가 생성한 패킷을 수신하기 때문에 트래픽의 병목 현상을 방지하기 위해서 100Mbps 대역폭에 전이중 방식을 사용하였다. 트래픽 로더는 서브 네트워크를 구성하는 스위치와 메인 스위치를 연결하는 노드에 대역폭을 20~90%까지 사용하는 트래픽을 생성하는 역할을 하게 된다.

6.2 균일 작업 할당의 성능 평가

PDP 시스템의 균일 작업 할당에 대한 성능평가는 <표 4>와 같이 세 가지 경우로 실험하였다. 트래픽의 생성은 트래픽 로더를 이용하여 20~90%까지 발생시키며 각각의 결과는 실험값들의 평균으로 사용하였다.

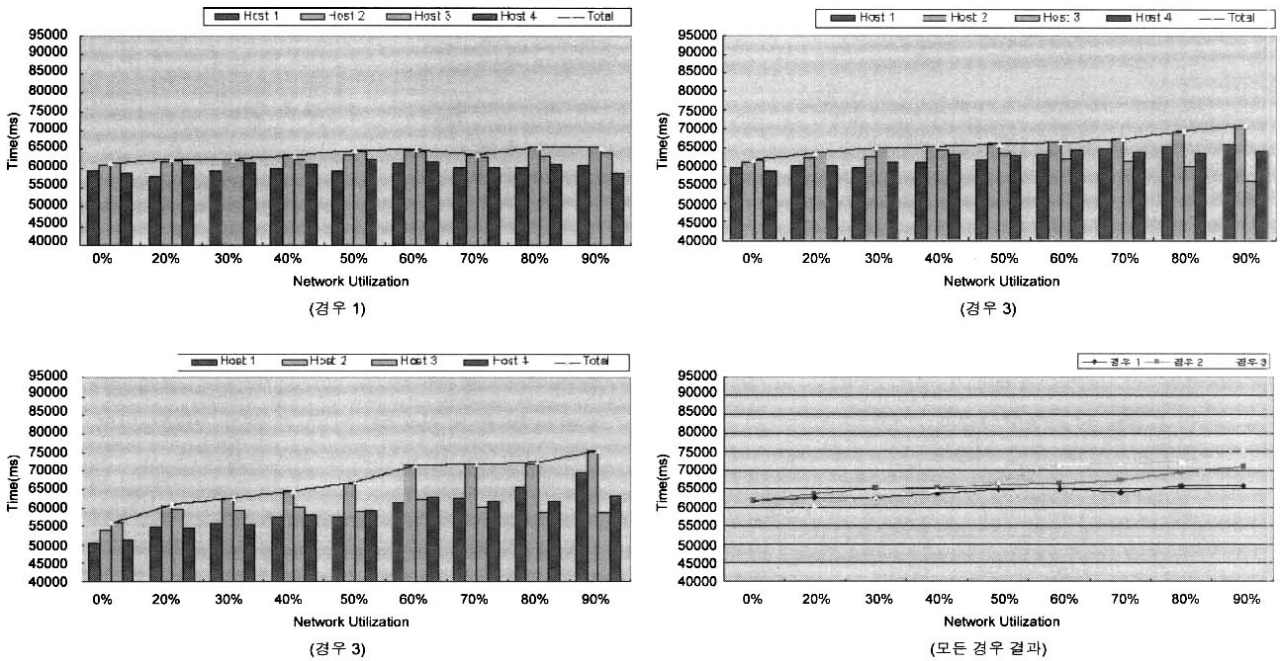
〈표 4〉 호스트 네트워크의 트래픽 생성 유·무(○ : 유, × : 무)

	Host 1	Host 2	Host 3	Host 4
경우 1	×	×	×	○
경우 2	×	×	○	○
경우 3	×	○	○	○



(그림 5) UTA 성능 평가

(그림 5)는 <표 4>에 따른 작업 수행 후 각 호스트의 작업 수행시간과 전체 작업 완료시간을 나타낸다. 그래프는 네트워크에 트래픽이 생성된 호스트의 작업 수행시간이 트래픽 양에 따라 증가하였다. 참여 호스트에게 동일한 양의 작업을 할당하는 균일 작업할당에서는 네트워크의 트래픽에 영향을 받은 호스트는 작업 결과가 지연되고 작업 수행시간이 증가하였다. 즉, 네트워크의 트래픽이 발생한 호스트의 작업 수행시간은 네트워크 트래픽 증가와 함께 작업 수행시간도 증가하고 전체 작업 수행시간이 증가함을 알 수 있다. 반면 네트워크 트래픽이 발생하지 않은 호스트는 네트워크에 트래픽이 발생한 호스트의 작업 수행시간이 길어지면 그만큼의 시간을 대기하므로 자원을 낭비하게 된다. 또한, 네트워크 트래픽이 생성된 호스트의 개수가 전체 작업 수행시간의 변화에 거의 영향을 미치지 않음을 알 수가 있다. 전체 작업 수행시간은 어느 한 호스트가 작업을 마치는 순간이 아니고 모든 호스트의 작업이 완료된 순간이므



(그림 6) NPTA 성능 평가

로 네트워크 트래픽에 의해서 작업 수행시간이 증가한 호스트의 영향으로 전체 수행시간이 증가하였다.

6.3 호스트의 네트워크 기반 작업할당

호스트의 네트워크 사용률에 따른 작업할당은 네트워크 트래픽이 발생한 호스트와 그렇지 않은 호스트의 작업 할당량에 차이를 두는 방법으로 네트워크의 트래픽이 발생한 호스트에게 작업을 적게 할당한다. 또한 네트워크 트래픽이 발생하지 않은 호스트는 많은 작업을 할당하여 호스트의 대기시간을 줄이고 네트워크 트래픽이 발생한 호스트의 작업은 적게 할당하여 작업 수행시간을 줄여 전체 작업 수행시간을 단축한다.

(그림 5)에서는 네트워크 트래픽이 발생한 호스트의 작업 수행시간이 증가하고 이 호스트의 작업 수행시간이 곧 전체 작업 수행시간과 유사함을 보인 것이다. 그러나 (그림 6)의 그래프들은 모든 호스트들의 작업 수행시간 증가가 (그림 5)의 그래프에 비해서 상당히 적음을 알 수 있다. 네트워크 트래픽이 발생한 호스트의 작업을 적게 할당하고 그렇지 않은 호스트들이 많은 작업을 할당함으로써 모든 호스트들의 대기시간을 최소화하여 전체 작업 수행시간이 줄었음을 보여주는 것이다.

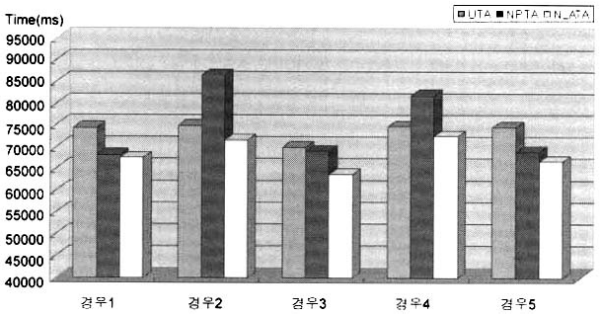
6.4 적응적 작업할당

네트워크 환경은 앞의 실험과 같이 항상 같은 양의 트래픽을 유지하지는 않는다. 작업 시작 전에 측정한 트래픽 양보다 더 증가할 수도 있고 그렇지 않을 수도 있다. 이런 네트워크 환경 변화에 대처하기 위한 방법이 적응적 작업할당이다. <표 5>에서 정의된 네트워크 사용률에 변화를 주면서 적응적으로 작업이 수행되는 실험결과가 (그림 7)이다.

경우 1은 네트워크 사용률에 변화를 주지 않은 것으로 균일 작업할당에 비해서 네트워크 사용률 기반 작업할당과 적응적 작업할당이 작업 수행시간이 단축되었다. 그러나 네트워크 기반 작업할당과 적응적 작업할당은 많은 차이를 보이지 않았는데 이는 네트워크 기반 작업할당을 통해서 전체 작업 수행시간의 균형을 맞추어 놓았기 때문에 작업의 재할당 횟수가 적어서 차이를 보이지 않은 것이다. 그러나 경우 2와 4는 균일 작업할당보다 네트워크 사용률 기반 작업할당이 작업 수행시간이 더 큼을 보이고 있다. 이 두 경우는 네트워크 사용률을 증가시켰기 때문이다. 경우 2는 Host 1이 네트워크 사용률이 없으므로 다른 호스트보다 많은 양의 작업을 할당받았는데 작업 수행 중 네트워크 사용률의 증가로 작업 수행시간이 증가되었기 때문이다. 경우 4도 Host 2의 네트워크 사용률의 증가로 인해서 작업 수행시간이 증가했다. 적응적 작업할당은 이 두 경우가 모두 작업 수행시간이 줄었는데 먼저 작업 수행이 완료된 호스트가 모든 작업이 완료될 때까지 대기하지 않고 많은 양의 작업이 남은 호스트의 작업을 재할당 받아 처리했기 때문이다. 경우 3과 5는 작업 도중 네트워크 사용률이 감소한 경우이다. 이 경우는 균일 작업할당과 네트워크 사용률 기반 작업할당 수행시간의 차는 경우 1과 유사하나 적응적 작업할당의 수행시간이 더 낮아졌다. 이것은 네트워크 사용률의 감소로 적은 작업을 할당받은 호스트의 작업이 빨리 끝나게 되고 작업이 끝난 호스트가 다른 호스트의 작업을 재할당 받아서 처리함으로써 이와 같은 결과를 나타낸다.

〈표 5〉 적응적 작업할당 실험 환경

	Host 1	Host 2	Host 3	Host 4	비 고
사용률	0%	30%	60%	90%	
경우 1					변화 없음
경우 2	90%				증 가
경우 3				0%	감 소
경우 4		90%			증 가
경우 5			0%		감 소



(그림 7) N_ATA 성능 평가

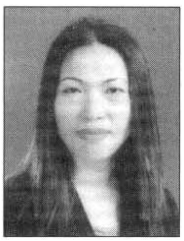
7. 결 론

본 논문은 인터넷 분산/병렬 시스템인 PDP에 호스트의 네트워크 환경을 모니터링하여 작업 할당 알고리즘에 적용하는 RNM-PDP(Realtime Network Monitor for PDP)를 제안하였다. 성능평가를 통해서 호스트 네트워크에 발생한 트래픽이 전체 작업 수행시간에 영향을 미치며, 호스트의 네트워크 환경을 실시간 모니터링하여 호스트 네트워크 사용률에 따라 호스트의 작업 할당량을 조절하여 트래픽으로 인한 작업 수행시간의 지연을 최소화시켰다. 또한 적응적 작업할당을 통해서 호스트의 네트워크 환경 변화에 유연하게 대처하여 작업 수행 효율성을 높였다.

참 고 문 헌

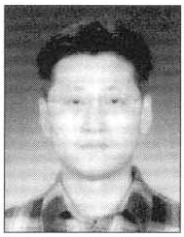
[1] J. E. Baldeschwieler, R. D. Blumofe and E. A. Brewer, "ATLAS : An Infra- structure for Global Computing," In Proc. of the 7th ACM SIGOPS European Workshop : Sys-

tem Support for Worldwide Application, September, 1996.
[2] Michael Philippsen and Matthies Zenger, "JavaParty-Transparent Remote Objects in Java," In the ACM Workshop on Java for Science and Engineering Computation, 1997.
[3] A. Baratloo, M. Karaul, Z. M. Kedem and P. Wychoff, "Charlotte : Meta-computing on the Web," The 9th International Conference on Parallel and Distributed Computing Systems, Dijon, France, http://www.cs.nyu.edu/milan/charlotte, September 1996.
[4] T. Brecht, H. sandhu, M. Shan and J. Talbot, "ParaWeb : Towards World- Wide Supercomputing," In Proc. of the 7th ACM SIGOPS European workshop : System Support for Worldwide Application, Connemara, Ireland, September 1996.
[5] Hernani Pedroso, Luis M. Silva, Victor Batista, Paulo Martins, Guilherme Soares and Telmo Menezes, "Web-based Metacomputing with JET," In Proc. of Concurrency : Practice and Experience, June, 1997.
[6] N. Camiel, S. London, N. Nisan, O. Regev. "The POPCORN Project : Distributed computing over the Internet in Java," In Proc. 6th International World Wide Web Conference, April, 1997.
[7] B. O. Christiansen, P. Cappello, M. F. Ionescu, M. O. Neary, K. E. Schauser, D. Wu, "Javelin : Internet-Based Parallel Computing Using Java," ACM Workshop on Java for Science and Engineering Computation, June, 1997.
[8] D. Caromel, W. Kauser, J. Vayssiere, "Java// Towards Seamless Computing end Metacomputing in Java," Concurrency Practice and Experience, pp. 1043-1061, September, 1998.
[9] Tobias Oetiker and Dave Rand, "MRTG : Multi Router Traffic Grapher," http://ee-staff.ethz.ch/~oetiker/webtools/mrtg/mrtg.html.
[10] L. Deri and R. Carbon, "Monitoring Networks Using Ntop," Released paper in http://luca.ntop.org, Jan 29th 2001.
[11] Craig Hunt, 'TCP/IP Network Administration', O'Reilly and Associates, Inc., 1992.
[12] Dave Curry and JeffMogul, "nfswatch-4.3," http://ftp.lip6fr/pub2/networking/nfs/.
[13] Lawrence Berkley National Laboratory, "tcpdump 3.4a6," ftp://ftp.ee.lbl.gov.
[14] Carter Bullard, "argus-1.7.beta.lb," ftp://ftp.sei.cmu.edu/pub/argus.
[16] Lawrence Berkley National Laboratory, "libpcap 0.6.2," http://www.tcpdump.org.
[15] SNMP Research International, Inc., http://www.snmp.org.
[16] 송은하, 정영식, "웹 환경에서 병렬/분산 처리를 위한 동적 호스트 관리 기법의 개발", 정보과학회논문지, 제8권 제3호, pp. 251-260, 2002.



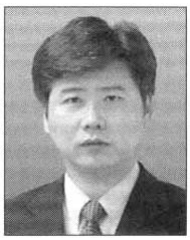
송 은 하

e-mail : ehsong@wonkwang.ac.kr
1997년 원광대학교 통계학과(이학사)
2000년 원광대학교 컴퓨터공학과(공학석사)
2001년~현재 원광대학교 컴퓨터공학과 박사과정
관심분야 : 분산병렬시스템, 그리드컴퓨팅, LBS



정 재 홍

e-mail : jhjeong@wonkwang.ac.kr
2002년 원광대학교 컴퓨터 및 정보통신공학부(공학사)
2004년 원광대학교 컴퓨터공학과(공학석사)
관심분야 : 분산병렬시스템, 그리드컴퓨팅



정 영 식

e-mail : ysjeong@wonkwang.ac.kr
1987년 고려대학교 수학과(이학사)
1989년 고려대학교 전산과학과(이학석사)
1993년 고려대학교 전산과학과(이학박사)
1993년~현재 원광대학교 컴퓨터 및 정보통신공학부 교수
관심분야 : 분산병렬시스템, 그리드컴퓨팅, LBS