Journal of Korea Multimedia Society Vol. 7, No. 5, May 2004(pp. 755-765)

PDP 시스템의 실시간 모니터링 및 시각화

김수자[†], 송은하[††], 박복자[†††], 정영식[††††]

요 약

최근에 많은 유휴 상태의 호스트 자원들을 이용한 인터넷 기반 분산/병렬 컴퓨팅은 대용량 작업처리와 여러 중요 논제들에 대해 그 유용성이 증명되고 있다. 대용량 작업이 수행되는 동안, 작업에 참여하는 호스트의 성능과 상태 변화에 대처하기 위한 실시간 모니터링 기능이 요구된다. 본 연구에서는 글로벌 컴퓨팅(global computing) 인트라스트럭처(intrastructure)로 구축된 인터넷 기반 분산/병렬 처리 프레임워크인 PDP(Parallel Distributed Processing)상의 실시간 모니터링 및 시각화에 대한 내용을 소개한다.

Realtime Monitoring and Visualization for PDP System

Su-ja Kim[†], Eun-Ha Song[††], Bock-Ja Park[†††], Young-Sik Jeong[††††]

ABSTRACT

Recently, the Internet-based distributed/parallel computing using many of idle hosts has been demonstrated its usefulness for processings of a large-scale task and involving several important issues. While executing a large-scale task, the realtime monitoring is required for adaptive strategy of the performance and state change of host. This paper provides the realtime monitoring and visualization on global computing infrastructure called PDP(Parallel Distributed Processing) which is a parallel computing framework implemented with Java for parallel computing on the Internet.

Key words: Distributed/Parallel System(분산/병렬 시스템), Monitoring(모니터링)

1. 서 론

인터넷에 연결된 많은 유휴 상태의 호스트 자원들을 이용하여 슈퍼컴퓨터로 처리해야 하는 대용량 작업을 처리하는 연구가 활발히 진행되고 있다[1,2,4,6,9,11,12,14]. 인터넷 기반 분산/병렬 처리 시스템은 호스트 자원을 공유하여 작업 수행 속도를 향상시키기 위해 다양한 연구가 진행되어 왔다. Parallel Virtual Machine(PVM)[15]과 Message Passing Interface(MPI)[16]는 원격지의 호스트에 존재하는 프로세스 간에 서로 메시지를 전달하는 기반을 마련해 주었다. 하지만, 프로그램 코드를 각 호스트에 배분하기 위해 호스트에 대한 계정을 필요로 하는 단점과 동적 호스트 관리에 대한 문제를 해결하기 어렵다. JavaParty[11]는 JavaRMI와 일반 소켓을 이용하여 호스트에 객체를 자동으로 분산시키는 원격 객체에 대한 메커니즘을 마련하여 공유 메모리 공간을 형성해 주나, 웹 환경에 적용하기 위한 해결책을 제시하지 못하였다. ParaWeb[14]은 자바 병렬 클래스 라이브러리의 구축을 통해 메시지 전달 방법을 사용하여 웹 컴퓨팅 자원을 활용 할 수 있는 해결책을 제시하였지만, 메시지 전달 방법에 대한 균형있는 작업 할

※ 교신저자(Corresponding Author) : 정영식, 주소 : 전북 익산시 신용동 344-2(570-749), 전화 : 063)850-6746, FAX : 063)856-8009, E-mail : ysjeong@wonkwang.ac.kr
접수일 : 2003년 10월 1일, 완료일 : 2003년 10월 31일
[†] 원광대학교 대학원 컴퓨터공학과
(E-mail : sjkim@wonkwang.ac.kr)
[††] 준회원, 원광대학교 대학원 컴퓨터공학과
(E-mail : ehsong@wonkwang.ac.kr)
[†††] 원광대학교 대학원 컴퓨터공학과
(E-mail : ppojja@wonkwang.ac.kr)
[††††] 정회원, 원광대학교 공과대학 컴퓨터 및 정보통신공학부 교수
※ 본 연구는 2003년 정보통신부에서 지원하는 기초기술연구지원사업(C1-2003-A1-2000-0013)으로 수행되었음.

당을 위한 해결 방법을 제시하지 못했다. Charlotte[1]는 순수 자바를 이용하여 인터넷에 연결된 어떤 호스트라도 웹 브라우저를 통해 자원을 제공할 수 있도록 설계된 병렬 플랫폼이며 결함허용과 균형적인 작업 할당 전략을 제공한다. 그러나, 가상 공유 메모리 관리에 따른 통신량 및 동기화에 따른 오버헤드가 크다는 단점을 가지고 있다. 그 외 ATLAS[6], JET[9], POPCORN[12] 및 Javelin[2]등이 개발되었으나, 이러한 분산/병렬 처리 연구는 호스트들의 성능 변화와 예측하기 힘든 호스트의 상태를 고려하지 않았기 때문에 호스트의 성능을 반영한 균형적인 작업할당을 수행하지 못하여 전체 작업시간이 길어지게 된다. 특히, 호스트의 이탈은 원격지에 있는 사용자의 행동에 따라 영향을 받고 시스템의 다운, 네트워크의 불안정성으로 인한 연결 해체 등 다양한 호스트의 결함 상태를 의미하지만 관리자는 호스트의 결함 원인을 알 수 없다. 그러므로 작업이 수행되는 동안 작업에 참여하는 호스트의 성능과 상태 변화에 대처하기 위한 실시간 모니터링 기능이 요구된다. 뿐만 아니라, 인터넷 기반 분산/병렬 처리 시 대용량 작업 수행을 위한 효율적인 호스트 자원 정책에 대한 신뢰성 향상을 위해 실시간 모니터링 기능이 필수적이다.

본 연구에서는 글로벌 컴퓨팅(global computing) 인트라스트럭처(intrastructure)로 구축된 인터넷 기반 분산/병렬 처리 프레임워크 인 PDP(Parallel Distributed Processing)[4]상에서 실시간 모니터링 및 시각화에 대한 내용을 소개한다. 먼저, 분산 시스템에서의 모니터링에 대한 기존 연구들을 소개하고 RHM 특성과도 상호 비교한다. 제 3절에서는 실시간 모니터링을 위한 RHM 구성, 기능, 호스트 자원 요소 분석 및 구조 등에 대해 제안한다. 제 4절에서는 실시간 모니터링을 통해 얻은 호스트 성능 및 상태 정보를 시각화 시킨다. 마지막으로 제 5절에서는 PDP의 작업 수행 동안 실시간 모니터링 정보로부터 작업에 참여하는 호스트 성능 및 상태 변화가 되는 실제 상황에 대한 분석 결과를 제시한다.

2. 기존연구

모니터링은 1987년 Joyce[7]가 소프트웨어 처리나 객체들에 관한 정보를 동적으로 수집, 표현, 분석하는 행위라 정의하였고 Sloman[11]은 성능관리, 구성관리, 고장관리, 보완관리와 같은 내구적이고 연속적 특징에 대한 일반적인 관리 업무를 의미한다고 했다. 모니터링에 관한 연구는 모니터링 하는 대상의 성격에 따라 차이를 보이지만 일반적으로 분산 시스템 모니터링 모델의 공통적인 기능은 첫째, 모니터링 정보를 추출하여 원하는 형태로 생성(generation)하는 기능. 둘째, 생성된 정보는 조합, 연관성 등 필터링을 통해 각 정보를 원하는 형태에 맞게 처리(processing)하는 기능. 셋째, 모니터링 정보의 로그파일을 생성하여 관리자나 시스템 설계자에게 유포(dissemination)하거나, 모니터링 중 시스템의 결함 정보 등을 분산시스템에 알려주는 기능. 넷째, 수집되어 처리된 정보는 적당한 디스플레이 형태로 표현되어 시스템 설계자나 관리자에게 정보 분석을 쉽게 하도록 돕는 정보 표현(presentation) 기능으로 구분된다.

DRMonitor[3]는 네트워크내 개인 컴퓨터의 자원 사용률을 모니터링하는 시스템으로 모니터링된 자원 정보를 기록하지 않지만 장애를 가진 컴퓨터를 검출하고 부하 균형 정책을 도와주기 위한 성능 평가 결과 값을 일정시간 마다 갱신하며, 모니터링 정보는 그래프를 통해 제공된다.

TOPSYS[13]는 이질적인 기계에 상관없이 어플리케이션을 확장하기 위해 병렬시스템 프로그래밍과 사용법을 단순화하는 통합 도구 환경을 제공한다. TOPSYS 모니터링은 병렬 프로그램의 동적인 행동을 전반적으로 이해하여 부적절한 동기성과 통신 지연에 의한 성능 병목현상과 예기치 못한 행동을 검출한다. TOPSYS 시각화는 동기여부에 따라 데이터 지향성 시각화, 시스템 지향성 시각화, 문제 지향성 시각화 기능을 제공한다.

TMP[8]는 실시간 컴퓨팅 시스템에서 임의의 실

표 1. 기존의 모니터링 시스템과 비교

항목 \ 시스템	DRMonitor	TOPSYS	TMP	RHM
결함허용	×	○	×	○
실시간	○	×	○	○
실행환경	서버	병렬 시스템	실시간어플리케이션	웹 기반 병렬시스템
로그	×	×	○	○
스케줄링	×	×	○	○
시각화	○	○	×	○

행 시간을 가진 작업을 스케줄링하기 위한 시스템이다. TMP 모니터링 시스템은 호스트 성능에 최소한의 영향을 주기 위해 하드웨어 모니터와 혼합된 형태이다. 호스트 시스템이 모니터 되는 결과에 동적으로 반응하도록 시스템에 모니터 정보를 제공하여 좋은 성능 튜닝, 스케쥴링, 에러 핸들링이 가능하다.

표 1은 이들 모니터링 시스템과 본 연구에서 제안하는 RHM과의 특성을 비교한 것이다.

3. 실시간 호스트 모니터 시스템

3.1 PDP 및 작업 할당 알고리즘

PDP(Parallel Distributed Processing)는 인터넷의 유휴상태(idle state) 호스트 자원을 이용해 대용량 작업을 병렬로 처리하여 전체 작업 수행시간을 단축시키는 인터넷 기반 분산/병렬 처리 시스템이다. PDP는 작업에 참여하는 특성에 따라 자신의 자원을 제공하는 *host*, 호스트와 자원 요청자간의 작업 교류와 동적으로 변화하는 호스트를 관리하는 *manager*, 작업 수행을 요구하는 *requester*로 구성된다. PDP 시스템의 구성도 및 전체적인 제어 흐름은 그림 1과 같다.

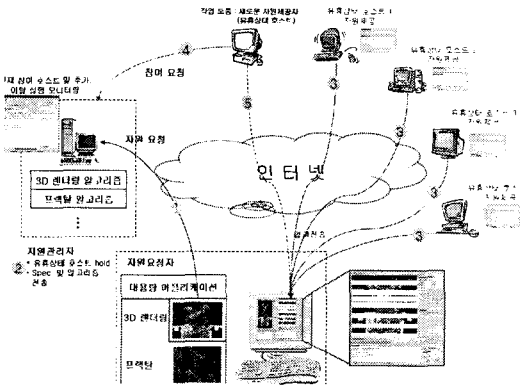


그림 1. PDP 구성도 및 제어 흐름

PDP의 작업 할당 알고리즘은 작업에 참여하는 호스트의 동적인 관리 여부에 따라 크게 정적 작업 할당(static task allocation)과 동적 작업 할당(dynamic task allocation)으로 분류된다.

UTA(Uniform Task Allocation)는 호스트의 성능에 상관없이 균등하게 작업을 할당하고, CPTA

표 2. PDP의 작업 할당 알고리즘

분류	작업 할당 알고리즘	호스트의 수
정적	단일 작업 할당(UTA)	고정적
	성능기반 작업 할당(CPTA)	
	정적 적응 작업 할당(S_ATA)	
동적	동적 적응 작업 할당(D_ATA)	가변적

(CPU Performance Task Allocation)는 호스트의 성능을 고려하여 작업을 할당한다. 또한, S_ATA (Static Adaptive Task Allocation)는 작업을 완료한 호스트에 성능이 느린 호스트의 작업을 재할당하고, D_ATA(Dynamic Adaptive Task Allocation)는 작업 중 이탈하거나 참여한 호스트에 작업을 할당하여 호스트 결함에 대해 대응한다. 하지만, 관리자는 어떤 호스트가 결함을 가지는지 결함여부를 정확히 알 수 없다. 단지 호스트 이탈로 처리 할 뿐이며 호스트의 성능 및 상태 변화에 대한 실시간 정보를 알 수 없다.

PDP에서 작업 할당 시 호스트 성능의 부정확한 벤치마킹 값과 동적 호스트 관리 시 새로운 호스트의 참여, 이탈로 인한 작업 재할당을 위해 작업에 참여중인 호스트들 중 작업 처리율이 가장 낮은 호스트를 추출하기 때문에 호스트의 상태 변화를 정확히 파악하기 위해 실시간 모니터링 기능이 필수적으로 요구된다.

PDP 시스템은 각 호스트의 성능을 알기 위해 초당 몇백만번의 부동소수점 연산을 수행하는지 MFLOPS 단위 값을 사용한다. LINPACK[5] 벤치마킹을 통해 얻어진 값은 호스트의 성능을 나타내기에는 작업 시작 시점에서 정적으로는 어느 정도 정확하다. 그러나 시간이 지남에 따라 호스트의 성능은 실시간으로 변하게 되고 벤치마킹 값은 초기 호스트가 접속했을 때의 값을 계속 유지하고 있는 문제점이 있다.

PDP에서는 나머지 기반 동적 재할당 방식을 제공하고 있다. 나머지 기반 동적 재할당은 아직 처리해야 할 작업이 가장 많이 남아있는 호스트를 추출하는 방식이다. 실제 호스트의 성능은 고려하지 않고 현재까지의 작업 진행률만을 고려하여 호스트를 추출한다. 작업 중에 남은 양은 적지만 호스트상의 시스템 병목현상으로 지연됨을 인식하지 못하고 다른 더 많이 남아있는 호스트의 작업을 할당하게 된다.

3.2 호스트 자원 요소 분류

호스트를 모니터링 하기 위한 정보는 PDP의 *host*에서 직접 추출한 하드웨어 정보와 PDP의 *manager*에서 전송 받는 모니터링 정보 요소로 구분된다. PDP의 *host*측면에서 살펴보면, *host*의 하드웨어 정보는 가변적인 여부에 따라 정적(static)자원 요소와 동적(dynamic)자원 요소로 분류된다. 정적 자원 요소는 시간 흐름에 따라 고정적인 호스트 정보로 메모리 정보와 CPU 정보이다. 동적 자원 요소는 시간 흐름에 따라 가변적인 정보로 자원 사용률, 시스템 객체 (system object), 메모리 정보로 구성된다.

PDP의 *manager*측은 *host*에서 전송받은 자원 요소와 JobObject, HostState로 구성된다. JobObject는 HostID, IPAddress, JobSize등을 가지며, HostState정보는 호스트의 상태 정보로 표 3과 같다.

표 3. 호스트 상태 정보

HostState	설 명
Running	작업 수행 중
Wait	RHM 접속, PDP에 로그인하지 않은 상태
Complete	작업을 완료한 상태
Die	비 연결
Idle	로그인하고 아직 작업을 수행하지 않은 상태 동적 참여시 complete host를 기다리는 상태
Error	host에서 Process Exit 수행으로 이탈한 경우
Login	PDP에 로그인한 상태

각 호스트 상태는 모니터링 정보 분석의 기준이 되는데, 작업 전(Login 상태) 호스트의 성능변화는 작업 중(Running 상태) 호스트 성능변화와 비교하여 연 관성을 발견함으로써 결함 여부를 알 수 있는 척도가 된다. 이때 분석 데이터의 샘플링 기준으로 각 호스트의 상태 정보를 사용하고 각 상태 전이도는 그림 2와 같다.

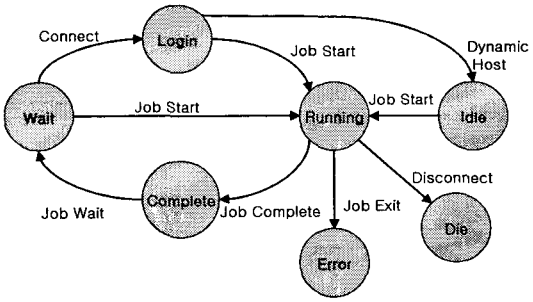


그림 2. 상태 전이도

RHM에서 사용하는 전체 모니터링 요소는 표 4와 같다.

표 4. 모니터링 시각화에 사용하는 자원 요소

분류	요소	설명
JobObject	HostID	호스트 로그인시 부여받은 ID
	HostTime	호스트 자원 추출 시간 호스트 들간의 동기화
	HostName	호스트 네트워크이름
	HostIP	IP
	HostScaleOfJob	작업크기
	HostColor	호스트 색
	HostJobRate	호스트 작업참여율
	HostSizeOfTask	호스트에 부여받은 작업량
	HostJobProgress	호스트이 작업 처리율
	HostPerformance	호스트 성능값
STATE	HostState	호스트 상태
DYNAMIC	JobProcess CPUusage	PDP의 작업을 받아 처리하는 프로세서의 CPU사용률
	CPUusage	호스트 CPU사용률
	Memusage	메모리 사용률
	ProcessNum	프로세스 수
	ThreadNum	쓰레드수
	ReadyThreadNum	대기열 쓰레드 수
	EventNum	이벤트 수
	MutexesNum	뮤텍스 수
	SectionsNum	섹션수
	SemaphoresNum	세마포어수
	AvailRAM	사용가능한 물리적 메모리량
	LimitRealMemory	부과 메모리 한도
	PagedKernel	페이지된 커널 메모리
	NonPagedKernel	페이지 안된 커널 메모리
STATIC	TotalRAM	전체 물리적 메모리량
	TotalRealMemory	전체 부과 메모리
	TotalKernel	전체 커널 메모리
	CPUspeed	속도
	CPUnum	수
	CPUtype	종류
	CPUlevel	레벨

3.3 RHM 구조

RHM은 기능적으로 RHMMonitor와 RHMWindow로 분리된다. RHMMonitor는 호스트 정보 수집, 저장, 가공을 담당하므로 PDP의 *host*와 *manager*에 삽입되며, RHMWindow는 시각화를 담당하므로 PDP의 *manager*에 삽입된다. PDP *host*의 RHMMonitor는 자원정보를 추출하기위한 DLL과 중개자 역할을 하는 ResBroker, 갱신된 정보를 저장하는 ResRegister, 정보를 *manager*에게 전송하는 ResTransmission으로 구성되고 그 구조는 다음 그

림 3과 같다. ResBroker는 C로 작성된 DLL에 접근하기 위해 JNI를 사용하여 하드웨어 정보를 얻는다. 정적 자원요소는 초기 접속 시 한번만 추출하고 동적 자원요소는 초당 자원정보를 추출하여 RHMManager에 전송한다.

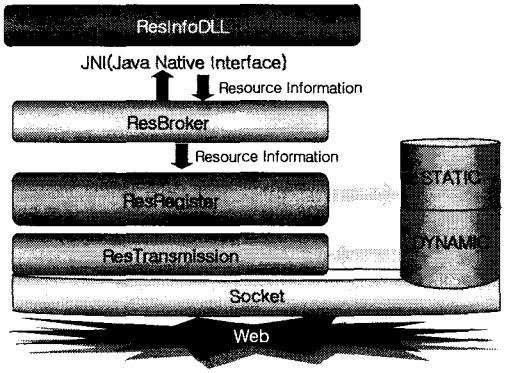


그림 3. PDP *host*의 RHMMonitor구조

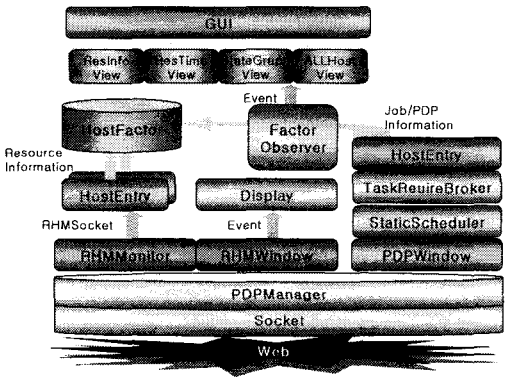


그림 4. PDP *Manager*측의 RHM 구조

PDP *manager*의 RHMMonitor는 PDP와의 피드백을 위해 서로 호환이 되어야 한다. RHMMonitor는 *host*의 연결이 확립되면 PDP와 별개의 소켓으로 *host* 정보를 받기 위해 호스트별로 HostEntry를 생성한다. HostEntry는 수신받은 정보를 토큰별로 분리하여 각 자원 요소에 저장하고 로깅 파일 생성 및 기록을 담당한다. HostFactor는 가장 최근의 *host* 자원 정보를 담고 있으며 서로 연관되어 있는 요소와의 조합으로 새로운 자원요소를 산출한다. HostFactor는 각 *host*의 작업정보를 실시간으로 갱신된다. HostFactor안의 정보는 동일 시간 내의 호스트 정보를 가진다. *host*에서 자원 정보를 추출 했을 때의 물리적인 시간은 같을 수 없다. 통신 지연으로 인해 이들 간의 동기화가 필요하므로 HostFactor에는 타임스탬프를 둔다.

RHMWindow는 모니터의 GUI에 해당하는 부분으로 호스트 선택 이벤트를 입력받는다. 이벤트에 따라 시각화에 필요한 자원요소를 관찰(observable)하여 정보의 변화가 있을 때 각 화면도 함께 갱신하여 실시간 모니터링이 가능하다.

4. 호스트 상태 및 성능 변화 정보의 시각화

4.1 시각화 분류

분석은 호스트 모니터링 정보를 호스트의 특정한 상태변화 값이나 평균값을 결정하는 일이다. 또한 호스트의 성능 변화 경향을 분석하여 고장을 예측하는데 매우 중요하며 결함 진단은 검출과 관련된 이벤트 리포트를 필요로 한다. 이러한 호스트의 정보 분석은 그래프와 히스토그램과 같은 정보 표현의 한 부분으로 인식된다. 시각화는 관리자와 시스템 설계자에게 보다 정확한 분석을 위해 모니터링 부분에서 중요하다.

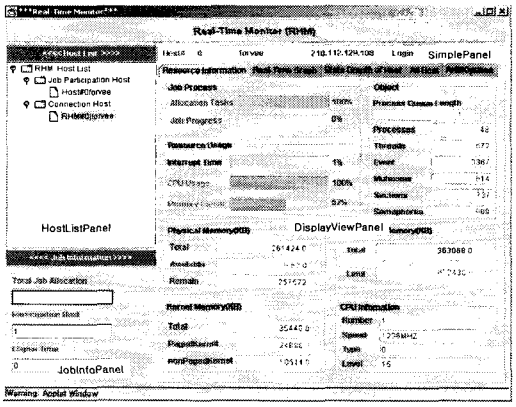


그림 5. RHM의 전체적인 GUI 구성도

RHM의 RHMWindow이 시각화 기능을 담당하고 제공하는 기본 GUI는 그림 5와 같으며, HostListPanel, SimplePanel, JobInfoPanel, DisplayViewPanel로 구성된다. HostListPanel은 호스트의 참여, 이탈, 접속 상태에 따른 전체 호스트의 리스트를 제공하는 패널로 이곳에서 *control flow event*가 발생한다. SimplePanel은 선택한 호스트의

간단한 정보인 호스트 ID, HostName, HostIP, HostState정보를 제공하는 패널이다. JobInfoPanel은 전체 PDP의 작업 정보를 다루는 패널로 전체 작업의 할당 비율과 참여한 호스트의 수, 모니터링 경과시간 정보 뷰를 나타내는 패널로 ResInfo, Real-Time Graph, State Graph of Host, All Host와 RHMOption으로 이루어져 있다.

RHMWindow는 뷰(view)방식, 디스플레이(display)방식, 분석(analyze) 측면으로 분류되며 표 5와 같다.

표 5. 시각화의 분류

관점	시각화	기능분류
뷰 (view)	자원정보 뷰	GUI
	실시간 그래프 뷰	
	상태 그래프 뷰	
	모든 호스트 뷰	
디스플레이 (diplay)	그래프	컴포넌트
	진행바	
	텍스트	
	테이블	
분석 (analysis)	일변량 분석	분석
	다변량 분석	

뷰 방식은 사용자에게 ResInfoView, ResTimeView, StateGraphView, AllHostView로 구성된다. ResInfoView는 호스트의 자원 요소를 *Data flow Event*에 따라 갱신된 정보를 제공한다. 전체적인 자원 요소를 진행 바와 텍스처로 시각화한다. ResTimeView는 시간에 따라 변화하는 동적 정보를 히스토그램, 꺽은선 그래프로 시각화하여 샘플링된 시간동안 한 자원 정보의 변화 추이를 모니터할 수 있고 다변량 분석이 가능하다. StateGraphView는 PDP 시스템에 접속하여 작업에 참여하고 있는 호스트나 작업이 완료된 호스트, 새롭게 추가된 호스트, 이탈한 호스트를 쉽게 모니터하기 위해 이미지 형식의 시각화를 제공한다. AllHostView는 전체 호스트의 성능 변화를 동시에 모니터하기 위해 테이블과 그래프로 시각화되고 다변량 분석이 가능하다.

디스플레이 방식은 그래프, 진행 바, 텍스트, 테이블 형태로 구분되며, 그래프는 작업 중에 호스트 CPU 사용률, 메모리 사용률, 인터럽트 처리율, Idle CPU 등의 정보를 시각화할 때 사용된다. 또한 참여한 전체 호스트의 CPU 사용률을 동시에 표현할 때 사용한다. 진행 바는 일반적인 자원 정보를 시간 변화에 대한 정량적인 값을 표현하고자 할 때 사용한다. 텍스트 형태는 메모리 크기, CPU 종류, CPU 속도, 할당받은 작업 수, 호스트 상태 정보 등을 표현할 때 사용한다.

분석 측면은 사용자에게 일변량 분석(univariate analysis)과 다변량 분석(multivariate analysis) 형태로 제공되는 관점이다. 일변량 분석은 하나의 자원 요소에 관한 자료를 통계적으로 처리하는 분석 기법으로 작업 중의 CPU 사용률 변화에 대한 분석 시 사용된다. 다변량 분석은 동시에 분석해야 할 자원 요소가 2개 이상인 분석 방식으로 이들 자원 요소들이 얼마나 연관성이 있는지를 알 수 있다. 제공되는 다변량 분석은 호스트의 Idle CPU, Memory usage, Interrupt rate에 대한 다변량 분석 그래프, 전체 참여한 호스트의 CPU usage에 관한 다변량 분석 그래프, 분석 자원 요소가 변경 가능하여 호스트에 대한 여러 자원 요소간의 다변량 분석을 위한 테이블이 제공된다.

4.2 RHMWindow 설계

실시간 모니터링 시각화는 호스트 자원 상태 정보가 변화하면 동시에 변화되어야 한다. RHM GUI는 두개의 이벤트에 의해 뷰가 변화하도록 사용자의 액션을 받아드린다. 사용자의 액션은 HostListPanel로부터 모니터링 하고자 하는 호스트를 선택하는 액션과 보고자 하는 시각화 방식을 선택하는 액션이다. RHMWindow는 HostListPanel로부터 선택한 호스트의 HostID를 추출하기 위해 HostObserver를 두어 별도의 쓰레드가 없어도 선택한 호스트의 ID가 변경되면 자동으로 갱신된다. 각 뷰는 *Display Interface*를 상속받아 개별 쓰레드로 동작한다. 선택된 호스트의 ID에 해당하는 자원 정보를 원하는 뷰 형태에 알맞게 처리하여 디스플레이한다.

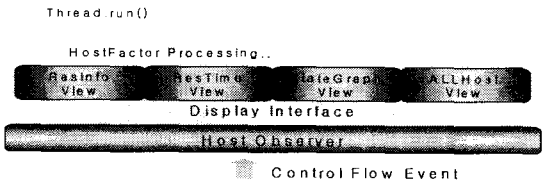


그림 6. RHMWindow의 디스플레이 모델

RHM 모니터링 이벤트는 자원 요소값의 변화를 나타내는 *data flow event*와 사용자의 *control flow event*가 있다. *data flow event*는 *FactorObserver*를 통해 상태를 받아드리고 각 표현 정보를 갱신한다. *control flow event*는 *HostObserver*를 통해 입력받고 사용자의 뷰 선택 및 호스트 선택 이벤트로 표시할 호스트의 자원정보를 수정하고 뷰 선택에 따라 해당하는 뷰에 맞는 자원을 가공한다. 이와 같은 이벤트 흐름은 그림 7과 같다.

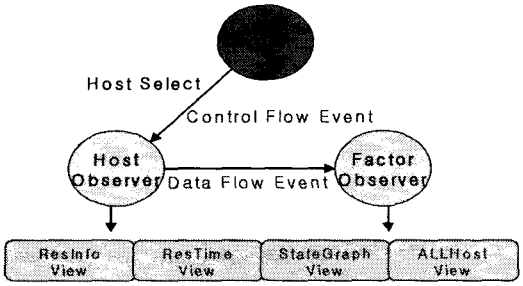


그림 7. 이벤트 흐름(event flow)

5. 모니터링 실례

PDP의 작업 할당 알고리즘들이 실행되는 동안, 실시간 모니터링 정보로부터 작업에 참여하는 호스트 성능 변화에 따른 분석 결과를 제시한다. 또한 작업 수행 시간 동안의 각 호스트 정보를 기록한 로그 파일을 분석한다. 참여하는 호스트의 기본 하드웨어 정보는 다음 표 6과 같다.

표 6. 호스트 하드웨어 성능 모니터링

호스트	Host Name	CPU 속도	메모리 크기	MLOPS
Host0	jaehong	Pentium 4 1296MHz	255MB	68.667
Host1	ehsong	Pentium 4 2019MHz	512MB	68.667
Host2	hyper	Pentium 4 1993MHz	1024MB	45.778
Host3	forvee	Pentium 4 1296MHz	255MB	68.667

5.1 로그 정보 생성

모니터링 정보의 재사용성을 위해 호스트 자원 정보는 별도의 정보로 저장된다. 이 로그 파일은 작업이 완료된 후 분석을 위한 기본 자료를 제공하고 작업 중 호스트가 할당받은 작업의 양, CPU 사용률 등 동적 자원 요소에 대한 타임스탬프 간격으로 저장된다. 그림 8은 모니터링 자원 정보 기록 파일 샘플을 나타낸다.

```
##### Resource Provider Infomation #####
Write Date: Tue Mar 18 11:28:48 GMT+09:00 2003
RP#1Name: forvee IP: 210.112.129.108
STATIC CPUINFO: 1296 1 0 15
STATIC MEMINFO: 261424 2097024 632032 99
DYNAMIC USAGE: 100 61
DYNAMIC MEM: 240612 2464 1918688
DYNAMIC ETC: 38 0 440
```

그림 8. 모니터링 자원 정보 기록 파일 샘플

5.2 UTA 경우 모니터링 분석

UTA 경우 호스트의 성능에 상관없이 작업을 균등하게 할당한다. RHM 모니터를 통해 호스트가 작업을 잘 수행하는지 성능차이를 얼마나 보이는지를 제공한다. 동일한 작업량을 할당하므로 작업을 처리하는데 걸린 총 시간이 호스트 성능을 나타낸다. 작업에 참여한 호스트의 성능은 ResInfoView를 통해 일변량 분석이 가능하며, 모니터링 화면은 그림 9와 같고 현재 작업에 참여한 호스트의 성능변화 값을 보여준다.

UTA 경우 성능에 상관없이 작업을 균등하게 할당받으므로 기본적인 성능을 고려하면 Host 2가 작업 수행률이 좋다고 예측이 가능하다. 그러나 RHM 모니터링을 통해 호스트의 CPU 사용률을 그림 10의 a)에서 실시간으로 모니터링한 결과를 보면, Host0, Host1은 CPU 사용률이 매우 낮고 변화추이가 약한 반면, Host2는 평소 상태에 높은 CPU 사용률을 보이며 변화추이가 발생한다. 모니터링 그래프를 분석하

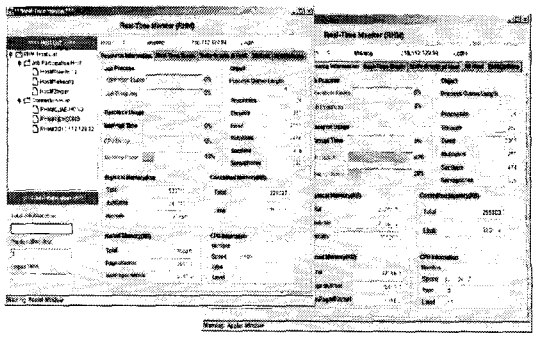
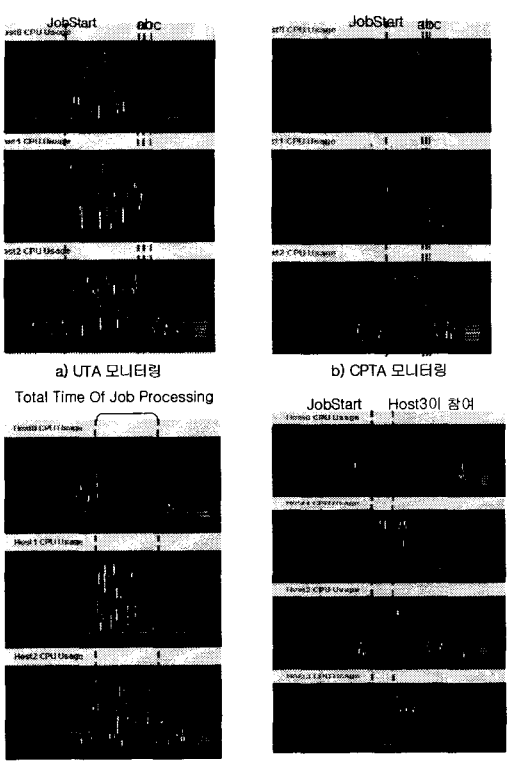


그림 9. 작업에 참여한 호스트의 성능변화 모니터링

면, Host0이 가장먼저 a시점에서 작업을 완료하고 b시점에서 Host1이 완료한다. 가장 수행속도가 좋을 꺼라 예측했던 Host2가 가장 늦게 완료됨을 알 수 있다. 즉 호스트의 CPU속도나 하드웨어 성능이 좋다고 하여 호스트의 작업 수행능력이 좋다는 예측은 어렵다. 호스트의 자원 사용률이 성능에 영향을 준다는 사실을 알 수 있다.



a) UTA 모니터링 b) CPTA 모니터링

c) S_ATA 모니터링 d) D_ATA 모니터링

그림 10. 각 알고리즘별 CPU 모니터링

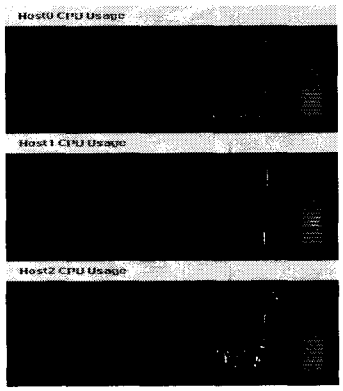


그림 11. 작업 CPU 사용률

5.3 CPTA 경우 모니터링 분석

호스트의 성능을 고려하여 작업을 할당하는 알고리즘이 수행하는 경우로서 호스트의 작업 완료 시점이 거의 비슷하게 끝나도록 각 호스트에게 작업을 할당한다. 작업 중인 호스트의 CPU 사용률은 그림 11과 같이 거의 90~100%의 CPU 사용률을 보인다. 그림 10의 b)에서 호스트들의 CPU 사용률 그래프를 분석하면 성능에 따라 작업을 할당하였고 전체 수행시간이 UTA보다 빨라졌음을 알 수 있다. UTA에서 Host2의 총 수행시간이 Host1, Host0보다 차이가 컸던 반면 CPTA를 통해 Host2의 총 수행시간이 빨라졌음을 모니터할 수 있다.

5.4 S_ATA, D_ATA 경우 모니터링 분석

그림 10의 c)에서처럼 Host0의 CPU 사용률 변화가 심한 경우, 모니터링 결과 그림 10의 b)처럼 성능 기반으로 작업을 할당하나 Host0이 매우 수행능력이 떨어지고 Host2가 수행 능력이 좋으므로 Host0의 작업을 재할당하여 수행하게 되어 총 수행시간이 Host0과 동일하게 이루어진다. 전체적인 작업 수행시간이 호스트의 성능에 영향을 받고 있음을 알 수 있다. 호스트의 결함이 발생하면 작업 재할당이 그림 12를 통해 이루어지고 있다는 사실을 알 수 있다. Host2가 작업 중 연결을 해제하여 더 이상 자원 요소의 변화가 나타나지 않는다. Host1은 Host2의 작업을 할당받아 작업을 완료한다.

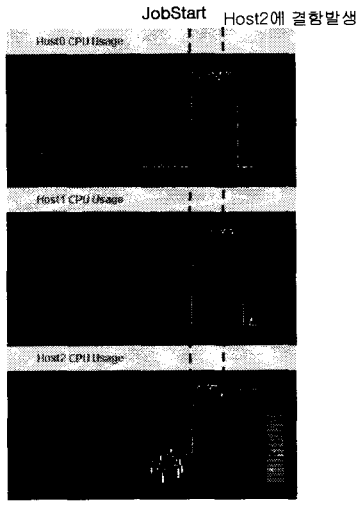


그림 12. 호스트결함 발생

호스트가 새로 추가될 경우, 가장 성능이 낮은 호스트의 자원을 재할당하여 작업을 수행한다. 그림 13에서 추가된 Host3이 작업을 수행함을 알 수 있다. 그림 13의 작업 수행시간 보다 Host3이 추가하여 전체적인 수행 시간이 짧아졌음을 알 수 있다.

작업의 전체 수행시간은 참여한 호스트의 수와 호스트 성능에 영향을 받는다. 그러나 너무 많은 호스트의 참여가 재할당 연산이 많아져 Wait상태가 빈번하면 전체 수행 시간이 오히려 떨어진다.

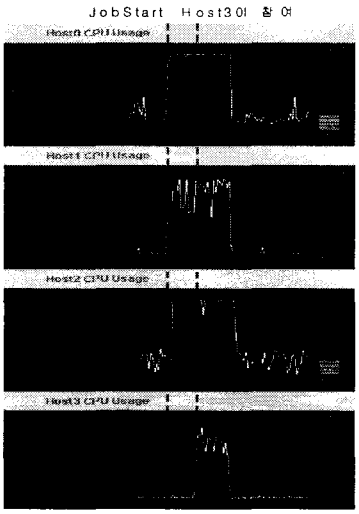


그림 13. 새로운 호스트 참여

5.5 호스트 자원의 다변량 분석

다변량 분석은 동시에 분석해야 할 자원 요소가 2개 이상인 경우에 제공되는 것으로 표 7의 Host0의 Idle CPU, Memory usage, Interrupt rate에 대한 다변량 분석 그래프는 그림 14와 같다.

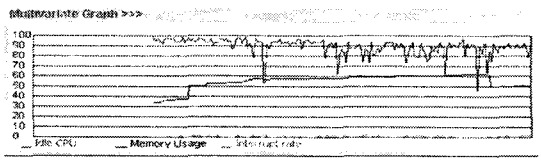


그림 14. Host0의 다변량 분석 그래프

그림 14의 빨간색은 idle CPU, 파란색은 memory usage, 분홍색은 interrupt rate를 나타낸다. idle CPU가 안정된 상태를 나타내면 memory usage이 감소하는 연관성을 알 수 있다. Host0의 System Object간의 초기 값에 대한 증가율을 보여주는 그래프는 그림 15과 같다. 표 7의 Host0, Host1, Host2가 참여한 호스트의 CPU 사용률에 관한 다변량 분석 그래프는 그림 16과 같이 모니터링이 가능하다. 그림에서 hyper 호스트가 가장 사용률 변화 추이가 빈번함을 알 수 있다.

또한, 참여한 전체 호스트간의 다변량 분석을 위해 분석할 자원 요소가 변경 가능한 테이블이 제공된다. 그림 17의 테이블 행은 변경 가능하며 system object간의 변화를 보고 싶을 경우 그림 18의 b)처럼 나타나고 그림 17의 테이블을 호스트 별로 정렬하여 그림 18의 a)처럼 한 호스트에 대한 분석이 가능하다.

그림 15. Host 0의 System Object 증가율

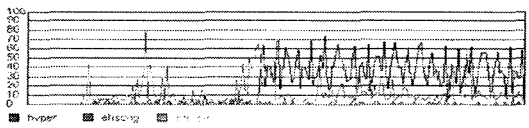


그림 16. 전체 호스트의 CPU 사용률

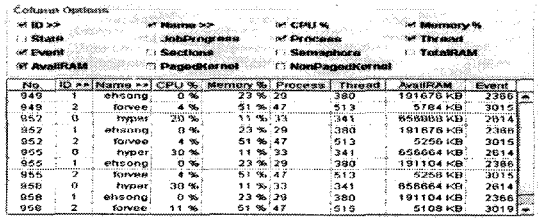


그림 17. 전체 호스트의 다변량 분석을 위한 테이블

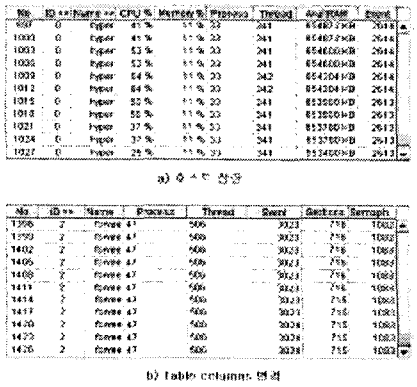


그림 18. 테이블 기능

6. 결 론

본 논문은 인터넷 기반 분산/병렬 처리 시스템인 PDP의 호스트 모니터링을 위해 RHM(Real-Time Monitor)에 대해 제안하였다. RHM은 호스트의 성능과 상태 정보를 수집, 처리, 유포, 시각화를 하였다. RHM은 DLL를 작성하여 쉽게 하드웨어 정보에 쉽게 접근할 수 있도록 제공하고 다양한 뷰를 제공하여 자원 정보를 시각화 하였다. 또한 RHM은 모니터링한 정보를 PDP의 작업 할당 알고리즘에 호스트의 성능 값 측정, 예측 등을 위한 값으로 사용한다. RHM을 통해 시스템 설계자나 관리자는 호스트의 참여, 이탈, 결함이 발생한 호스트를 판별하는데 도움을 줄 수 있으며, PDP의 작업 할당 알고리즘 유효성 검증에도 사용됨을 보였다.

추후 본 연구는 RHM 모니터 정보를 이용한 작업 스케줄링 알고리즘이 수행되어야 한다. RHM 스케줄링 알고리즘은 RHM의 호스트 성능 정보를 이용하여 기존 벤치마킹의 불변성 문제를 해결하여야 한다. 호스트 성능 값을 측정하기 위해 하드웨어가 호스트 성능에 미치는 영향력을 실험하고 작업 처리 능력이 증가하는지 보여야한다. 또한 성능이 가장 느린 호스트를 추출 할 경우 기존의 작업량을 가지고 수행하지만 호스트의 성능을 고려하여 추출할 수 있다.

참 고 문 헌

[1] A. Baratloo, M. Karaul, Z. M. Kedem, and P. Wychoff, "Charlotte: Meta-computing on the Weg," *The 9th International Conference on Parallel and Distributed Computing Systems*, Dijon, France, http://www.cs.nyu.edu/milan/charlotte/index.html, September 1996.

[2] B. O. Christiansen, P. Cappello, M. F. Ionescu, M. O. Neary, and K. E. Schauser, "Javelin: Internet-Based Parallel Computing Using Java," *ACM Workshop on Java for Science and Engineering Computation*, 1997.

[3] Domingues, P. Silva, L. Silva, J.G., "*DRMonitor-A Distributed Resource Monitoring System,*" Parallel, Distributed and Network-Based Processing, 2003. Proceedings. Eleventh Euromicro Conference on 127-133.

[4] Eun-Ha Song, Young-Sik Jeong, "Development of Dynamic Host Management Scheme for Parallel/Distributed Processing on the Web," *KISS* Vol. 8 No 3. 2002.

[5] J. Dongarra, J. Bunch, D. Moler, and G. W. Stewart, "*LINPACK User's Guide*" SIAM Philadephia PA 1979.

[6] J. E. Baldeschwieler, R. D. Blumofe, and E. A. Brewer, "ATLAS:An Infrastructure for Global Computing," In Proc. of the 7th ACM SIGOPS European Workshop : System Support for Worldwide Application, 1996.

[7] J. Joyce, et al., "*Monitoring Distributed Systems,*" ACM Trans. Comput. Syst. vol 5 no 2 May 1987. 121-50.

[8] Haban, D. Shin, K. G, "*Application of real-time monitoring to scheduling tasks with random execution times,*" IEEE Transactions on Software Engineering, vol 16 Dec. 1990 pp 1374-1389.

[9] 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, 1997.

[10] Michael Philippsen and Matthies Zenger, "*JavaParty-Transparent Remote Objects in Java,*" In the ACM Workshop on Java for Science and Engineering computation, 1997.

[11] M. Sloman, "*Distrubuted Systems Management,*" Imperial College Research Report DOC 87/6 April 1, 1987.

[12] N. Camiel, S. London, N. Nisan, and O. Regev. "The POPCORN Project:Distributed computing over the Internet in Java," In Proc. 6th International World Wide Web Conference, 1997.

[13] T. Bemmerl, R. Lindhof, and T. Treml, "*The Distributed Monitor System of TOPSYS,*" Proceeding CONPAR 1990 - VAPP IV Sep. 1990 Springer-Verlang, pp. 756-65.

[14] 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.

[15] V. Sunderam, G. Geist, J. Dongarra, and R. Manchek, "*The PVM concurrent system: Eolution, experiences, and trends,*" Parallel Computing, 1994.

[16] W. Gropp, E. Lusk, and A. SKjellum. "*Using MPI: Portable Parallel Programming with the Message-Passing-Interface,*" MIT Press, 1994.

김 수 자

2002년 원광대학교 컴퓨터 및 정보통신공학부(공학사)
2002년~현재 원광대학교 컴퓨터공학과 석사과정

관심분야 : 분산병렬시스템, 모니터링

송 은 하

1997년 원광대학교 통계학과(이학사)
2000년 원광대학교 컴퓨터공학과(공학석사)
2001년~현재 원광대학교 컴퓨터공학과 박사과정

관심분야 : 분산병렬시스템, 그리드컴퓨팅, LBS

박 복 자

1998년 조선대학교 전자계산학과(이학사)
2001년~현재 원광대학교 교육대학원 정보·컴퓨터전공 석사과정

관심분야 : WBI, SCORM, e-learning system

정 영 식

1987년 고려대학교 수학과(이학사)
1989년 고려대학교 전산과학과(이학석사)
1993년 고려대학교 전산과학과(이학박사)
1993년~현재 원광대학교 컴퓨터 및 정보통신공학부 교수

관심분야 : 분산병렬시스템, 그리드컴퓨팅, LBS