웹 환경에서의 병렬/분산 처리를 위한 동적 호스트 관리 기법의 개발

(Development of the Dynamic Host Management Scheme for Parallel/Distributed Processing on the Web)

송 은 하 † 정 영 식 ††
(Eun-Ha Song) (Young-Sik Jeong)

요 약 웹에 존재하는 수많은 유휴상태 호스트들을 이용한 병렬/분산 처리는 대규모 응용문제에 대해 높은 가격 대 성능비를 가진다. 웹 환경에서 병렬/분산 처리를 위하여 호스트들의 이질성 및 가변성, 자율성, 지속적인 성능보장과 참여 호스트 수 변화 등 예측할 수 없는 상태에 대한 해결책을 제시하여야 한다. 본 논문은 지리적으로 떨어져 있는 참여 호스트들의 작업 처리를 성능에 기반하는 적응적 작업 재할당 전략을 제안한다. 또한, 대규모 응용문제의 병렬 처리 중에 호스트 수가 변하는 동적 환경에 대해 동적 호스트 관리 스킴을 제공한다. 본 논문에서는 PDSWeb (Parallel/Distributed Scheme on Web) 시스템을 구현하여, 많은 연산량을 지닌 랜더링 이미지 생성에 적용하여 평가한다. 그 결과 호스트의 가변성에 대해 적응적 작업 재할당은 최고 90%이상 향상하였으며, 호스트 추가와 삭제에 따른 성능 향상 정도를 보인다.

키워드 : 병렬/분산 처리, 병렬/분산 컴퓨팅 환경, 작업 할당, 고성능 컴퓨팅, 자바, 웹

Abstract The parallel/distributed processing with a lot of the idle hosts on the web has the high cost-performance ratio for large-scale applications. It's processing has to show the solutions for unpredictable status such as heterogeneity of hosts, variability of hosts, autonomy of hosts, the supporting performance continuously, and the number of hosts which are participated in computation and so on. In this paper, we propose the strategy of adaptive task reallocation based on performance the host job processing, spread out geographically. Also, It shows the scheme of dynamic host management with dynamic environment, which is changed by lots of hosts on the web during parallel processing for large-scale applications. This paper implements the PDSWeb (Parallel/Distributed Scheme on Web) system, evaluates and applies it to the generation of rendering image with highly intensive computation. The results are showed that the adaptive task reallocation with the variation of hosts has been increased up to maximum 90% and the improvement in performance according to add/delete of hosts.

Key words : Parallel/Distributed Processing, Parallel/Distributed Computing Environment, Task Allocation, High Performance Computing, Java, Web

1. 서 론

웹을 이용한 병렬/분산 컴퓨팅의 목적은 전 세계적으로 구축되어 있는 인터넷을 이용하여 적은 비용으로 유휴 상태(idle state)에 있는 컴퓨팅 자원의 활용도(utilization)를 높이는 데 있다[1]. 그러나, 웹 컴퓨팅은 다음과 같은 여러 가지 어려움이 있다[2]. 첫째, 웹에 연결된 호스트들의 이질성(heterogeneity)을 가진다. 대부분 컴퓨터 하드웨어 및 설치가 쉽지 않으므로 각각의 플랫폼에 맞는 구조나 운영 체제의 다양성으로 코드의 개발, 분산 컴파일링 작업이 추가로 요구된다. 또한, 제공자와 사용자간의 신뢰성(reliability)이 떨어지므로 기존연구들은 미리 설정되어 있는 호스트들만이 참여하는 제약사항이 있다. 둘째, 호스트들의 지속적인 수행 능력

· 본 연구는 한국과학재단 목적기초연구(과제번호 : R05-2001-000-01000-0) 지원으로 수행되었음.
† 학생회원 : 원광대학교 컴퓨터공학과
ehsong@wonkwang.ac.kr
†† 종신회원 : 원광대학교 컴퓨터및정보통신공학부 교수
ysjeong@wonkwang.ac.kr
논문접수 : 2001년 7월 21일
심사완료 : 2002년 3월 7일

을 예측할 수 없다. 각각 나름대로 수행 목적이 다르므로 이것이 병렬/분산 처리에 이용될 때 갑작스런 환경 변화 및 과부하로 인하여 성능비가 극심하다. 셋째, 웹은 참여 호스트들에 대한 상태 변화와 자율성을 가진다. 일단 연산에 참여한 호스트라 할지라도, 자신의 의사 및 결함과 같은 원인으로 연산의 지속성을 예상하기 힘들다. 또한, 일정한 호스트들로 구성되어 작업을 실행하는 도중에 새로운 컴퓨팅 자원의 참여도 고려해야 한다. 마지막으로 병렬처리에 대한 기존 연구들은 대부분 어플리케이션 객체와 병렬처리 객체와의 상호 의존성이 깊다. 새로운 어플리케이션을 구축되어 있는 병렬/분산 처리 시스템에 적용하기 위해서는 작업 분할 정책 이외에도 독립적인 어플리케이션을 개발하기 위한 객체지향적 설계가 필요하게 된다.

따라서, 본 논문에서는 앞에서 언급한 웹 컴퓨팅의 고려사항에 대해 대규모 작업을 병렬/분산 처리를 위한 동적 호스트 관리 기법을 제안한다.

본 논문에서 제시하는 시스템은 특정한 작업 수행에 자원을 요구하는 자원 요청자(Resource Requester, 이하 RR), 연산에 컴퓨팅 자원을 제공하는 자원 제공자(Resource Provider, 이하 RP), 자원 요청자와 자원 제공자 사이의 원활한 작업 할당과 이들을 관리하는 자원 관리자(Resource Manager, 이하 RM)로 구성된다. 추가로, 작업 처리율이 가변적인 자원 제공자를 관리하기 위한 가상 관리자(Virtual Manager), 특정 어플리케이션들과 시스템과의 연결을 담당하는 어플리케이션 중계자(Application Proxy)를 둔다.

특히, 자원 요청자의 의하여 계산된 작업 결과는 자바가 지원하는 원격 메소드 호출(Remote Method Invocation, 이하 JavaRMI)에 의한 원격객체 참조 값을 이용하여 콜백(call back) 처리한다. 따라서, 자원 관리자를 거치지 않고 즉시 자원 요청자에게 전달하여 이중으로 들어가는 통신 시간을 줄인다.

본 논문에서는 구현한 시스템을 PDSWeb(Parallel Distributed Scheme on Web)이라고 명명하고, 렌더링 이미지 생성을 적용하여 제안한 작업 할당 및 호스트 관리 기법의 성능을 평가하였다.

2. 관련연구

웹 환경의 풍부한 유휴 자원을 이용하여 대규모 작업을 병렬/분산 수행시키는 관점에서 본 논문과 관련되는 시스템들은 다음과 같다.

ATLAS, JavaParty, Charlotte, ParaWeb, Javelin, POPCORN, JET 및 Java// 등이 있으며, 이들은 기본적으로 웹 기술을 바탕으로 어플리케이션을 처리하기 위한 특별한 프로그래밍 모델을 제공하고 있다.

웹 컴퓨팅의 인프라스트럭처인 ATLAS[1]는 자바와 Clik의 프로그래밍 기술을 접합하여 멀티쓰레드로 작성된 프로그램을 네트워크 기반의 자원들을 활용하여 병렬 처리할 수 있도록 설계되었다. 작업 취득 스케줄링(work-stealing scheduling) 알고리즘을 사용하여 작업 할당 전략을 지원하고, 확장성을 제공하기 위한 트리 구조의 프로그래밍 모델을 가진다. 그러나, 트리 구조에서 오는 적용 가능한 어플리케이션의 제한 및 플랫폼 종속적인 라이브러리의 사용, 일반 사용자의 접근이 어려운 단점이 있으며 결함내성에 대해 언급하지 않았다.

JavaParty[3]는 JavaRMI나 일반 소켓 프로그램을 이용하여 원격객체 구현시 발생하는 코드 오버헤드를 줄이기 위해 여러 호스트에 자동으로 분산시키는 메커니즘을 마련하여 공유 메모리 공간을 형성해 준다. 하지만, 웹에 적용하기 위한 해결책을 제시하지 못하였다.

Charlotte[4]은 인터넷에 연결된 어떤 컴퓨터라도 웹 브라우저만 있으면 자원 제공이 가능하도록 설계된 자바 기반 병렬 플랫폼이다. 열정적 스케줄링(eager scheduling)과 2단계 멱등 수행 전략을 사용하여 결함내성과 작업 할당 전략을 제공한다. 태스크들간의 통신을 위해 가상 공유 메모리 방법을 지원하였으나, 메모리 관리에 대한 오버헤드와 똑같은 태스크를 가진 여러 호스트들이 공유 메모리를 접근함에 있어서 통신량의 증가에 따른 과도한 자원 낭비가 발생하고 확장성에도 문제가 있다.

ParaWeb[5]은 JPCL(Java Parallel Class Library)의 구축과 메시지 전달 방법을 사용하여 웹 컴퓨팅의 자원을 활용할 수 있는 해결책을 제시하였다. 하지만, 메시지 전달 방법으로 균형적인 작업 할당을 얻지 못한다. 뿐만 아니라, 순수 자바를 사용하지 않고 기존의 자바 가상 머신 인터프리터를 수정하여 병렬 컴퓨팅 환경에 구현함으로써 확장성에 문제가 있다.

Javelin[6]은 기존의 JVM과의 호환성을 목적으로 실시간 라이브러리(Load-Balancing Runtime Library)를 구현하였다. 낙관적 작업 전파(optimistic load propagation)에 기반한 간단한 작업 할당 전략을 제시한다. 자바 애플릿간의 직접 통신 및 메시지 전달 방식으로 태스크간의 통신은 이루어지지만 결함내성을 제공하지 못하였다.

POPCORN[7]은 Market/Buyer/Seller의 구조를 가지며, computelet라고 불리는 태스크들이 ATLAS의 프로그래밍 모델과 유사한 트리 구조를 가진다. 호스트의 결함으로 수행되지 못한 태스크에 대해 다른 호스트에게 대신 작업을 요구하는 방법으로 결함내성을 제공한다. 또한

요구받은 computelet의 실행을 빨리 마친 Seller에게 다른 computelet을 맡기는 방법으로 작업 할당 전략을 사용하였다. 그러나 이러한 전략은 미흡하며, 하나의 Market 시스템이 시작에서 결과까지 관리하므로 병목 현상이 발생되기 쉽고 확장성이 부족한 단점을 가진다.

JET[8]는 Master/Worker의 구조로 웹 기반 병렬 라이브러리를 구축하였다. 웹 환경에서 동작하는 여러 시스템들의 결함을 고려하여 Master와 Worker의 상태를 체크포인팅과 로깅기법의 적용으로 결함내성을 제공하였다. 그러나, 작업 할당 전략에 관한 알고리즘은 제시하지 않았다.

Java//[9]는 지금까지 웹 컴퓨팅 기반의 객체 지향 병렬 프로그래밍을 지원하는 가장 적당한 프레임워크로 평가받고 있다. 객체 분배 문제 및 동시성 제어에 대한 문제를 이 프레임워크 하위 계층에서 해결하고 있어서 상위 레벨에서 병렬화가 가능한 문제들에 대한 모델링 및 알고리즘에 집중하도록 해준다. 하지만, 객체들간의 행동 방식과 프로그래머 입장에서 객체를 다루는 방법에 대해 치중하며, 동적으로 변화하는 호스트에 대한 해결책은 지원하지 않으므로 웹 컴퓨팅 자원들을 최대한 활용할 수 있는 아이디어가 부족하다.

본 논문에서 구현한 PDSWeb 시스템은 작업 관리자의 중재로 인한 통신 오버헤드를 줄이는 메커니즘을 적용하며, 자바 기술을 이용하여 이기종 시스템간의 호환성과 확장성을 유지하도록 하였다. 또한 웹 환경이 미치는 가변적 요인에 효과적인 적응적 작업 재할당 기법과 동적으로 변하는 호스트들을 위한 대처 방안을 제공한다.

3. 작업 할당 스킴

3.1 성능 기반 작업 할당

웹에 연결된 호스트들은 이질성을 가지므로 하드웨어 자원에 따라 모두 다른 성능을 보인다[1]. 병렬 수행의 전체 작업 시간은 호스트들이 연산 시작을 알리는 순간부터 참여 호스트들이 모든 연산을 마치고 작업 완료 응답을 보내는데 걸리는 시간이다. 그러므로, 각 호스트들의 성능을 고려하여 적절한 분배가 이루어지지 않는다면 성능이 낮은 호스트들의 늦은 응답으로 전체 비용이 크게 된다. 이로 인해 병렬 컴퓨팅의 효율성(efficiency)이 낮아진다. 즉, 임의의 호스트가 갖는 하드웨어적 특성이 전체 컴퓨팅 시간에 큰 영향을 미치는 요인이 될 수 있다. 각 호스트들이 하드웨어나 자원을 제공할 때 성능비대로 작업을 나누어 배분하여 동일한 시간에 작업을 마칠 수 있는 성능 기반 작업 할당(performance-based task allocation) 기법을 그림 1에서 나타낸다.

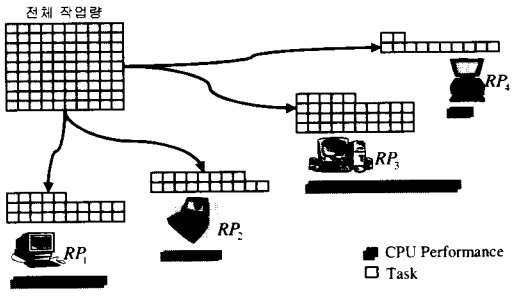


그림 1 성능 기반 작업 할당

각 서브작업간의 종속 관계를 고려하지 않으며, 연산을 수행하는 RP들이 독립적으로 작업을 할당받는다. RM은 주어진 작업을 일정한 크기를 지닌 서브작업으로 나누어 성능에 맞도록 분배한다.

RP들은 RM에 연결하여 초기 인터페이스를 위한 애플릿을 다운로드한다. 최초로 자신의 성능을 측정하여 RM에게 전달함으로써 작업 참여를 알린다. 성능평가 방법으로는 행렬을 이용한 선형 방정식을 푸는 문제이며, 방정식의 해는 부분 피봇팅에 의한 가우스 소거법을 이용하는 Linpack 벤치마크[10]를 사용하였다. 이러한 계산은 단위 시간당 부동 소수점 덧셈과 곱셈 연산의 수행 횟수를 측정하여 MFLOPS 단위 값을 돌려준다.

RM은 성능평가 정보를 각각 RP의 엔트리 클래스(entry class)에 저장한다. 서브작업이 RP의 성능으로 결정되므로 동일한 시간 내에 작업을 마치게 된다.

3.2 적응적 작업 재할당

웹 컴퓨팅에서는 호스트들의 위치가 지리적으로 멀리 떨어져 있을 수 있다. 또한, 여러 사용자들에게 마저 개방되어 있으므로 성능 예측이 불가능하다. 뿐만 아니라, 사용자의 이용 정도에 따라 호스트에 상주하는 내부 호스트 수가 증감되어 성능 변화의 원인이 된다. 따라서, 단순히 작업이 시작하는 순간에 RP의 성능만으로 시간이 흐름에 따라 변하는 수행 능력에 능동적일 수 없다. 그러므로, 수시로 변화하는 성능을 적응적 작업 재할당(adaptive task reallocation)으로 해결한다.

그림 2는 4대의 호스트에 의해 작업을 적응적으로 하는 과정이다.

앞서 제시한 작업 할당 기법을 기반으로 성능비대로 4대의 호스트에게 작업을 분배한다. 작업 수행 도중 RP_1는 초기 할당받은 작업을 모두 마쳐 유휴 상태에 있고, 남은 3대의 호스트는 불규칙한 성능 변화가 발생

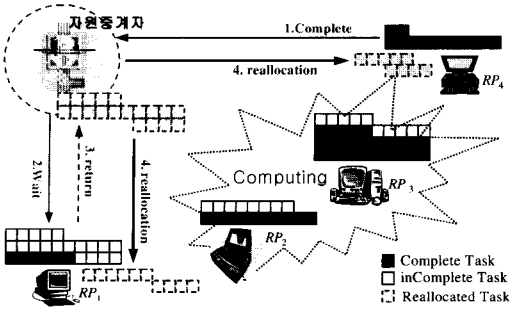


그림 2 적응적 작업 재할당

한다. 특히, RP_1이 심각한 시스템 과부하가 발생했다고 가정하자. 작업 도중에 임의의 RP_i(RP_4)에 의해 완료(Complete) 메시지를 받게 되면, 대기 상태로 다음 작업이나 명령을 기다리게 된다. RM은 가장 낮은 작업 처리율을 가진 호스트를 찾아 대기(Wait) 메시지를 전달해 작업을 일시 정지한다. 해당하는 호스트인 RP_1의 잔여 작업량을 재할당한다. 즉, 작업 수행 상태에 따른 추가 연산으로 RP_1의 태스크 일부를 RP_4가 대신한다. 더욱이 재할당에 참여하지 않는 $RP_{2,3}$은 해당 연산을 계속하여 재할당으로 인한 지연시간을 줄인다. 성능 기반 작업 할당에 적응력을 추가하여 잔여 태스크를 나누어 수행함으로써 전체 작업 시간의 영향을 줄인다.

4. 동적 호스트 관리 스킴

웹은 많은 자율성을 내포하고 있으며, 웹 컴퓨팅에서는 방대한 양의 연산에 자원을 제공할 메시지를 전달하였음에도 불구하고 작업 수행도중 이탈이 일어날 수 있다. 또한, 네트워크의 오류나 호스트의 내부 결함으로 예고없이 더 이상 자원 제공을 못할 수도 있다. 이와 같은 점을 고려하지 않는다면, 초기 설정된 정적 자원에 의한 연산은 지역적인 환경에서만 적용하는 시스템으로 국한되며, 전체 컴퓨팅 작업의 100% 완성률을 보장할 수 없다. 이와 반면에, 작업 수행 도중에 새로운 호스트의 추가이다. 즉, 자원을 현재 진행중인 작업에 추가가 가능하다면, 추가시점으로부터 보다 높은 컴퓨팅 파워로 전체 수행 시간은 감소한다.

본 논문에서는 호스트의 추가와 이탈을 고려한 동적 호스트 관리 기법을 제시한다.

4.1 동적 RP 이탈 메커니즘

그림 3은 이미 할당받은 작업을 마친 대기 호스트가 존재할 때, 작업 도중에 호스트의 이탈이 일어난 경우의 연산 과정이다.

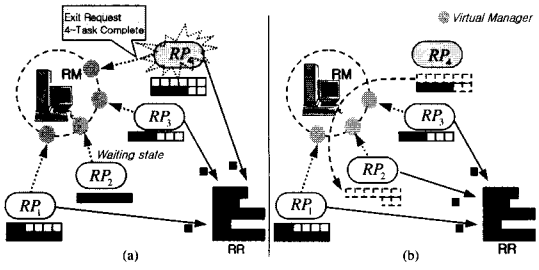


그림 3 대기 호스트 존재시 이탈

그림 3의 (a)는 RP_4가 자신의 의도로 시스템을 이탈한 경우이다. RP_4는 종료(Exit) 메시지와 지금까지 처리한 연산 정보를 반환하고 쓰레드 종료 후 현재 시스템에서 빠져 나온다. 이탈 응답을 받는 RM은 이탈 호스트로부터 얻은 연산 정보에 의해 잔여 태스크를 추출한다.

그림 3의 (b)는 이탈된 RP가 정보를 반환하지 않았으며, RM은 종료 메시지만을 전송받은 경우이다. RM은 이탈 호스트의 가상 관리자(Virtual Manager, 이하 VM)에서 모니터링한 데이터를 기준으로 잔여 태스크를 추출한다. 대기 상태의 RP_2에게 RP_4의 잔여 태스크를 넘긴다. 이탈 호스트의 VM는 종료되고, 3개의 호스트에 의해 적응적 작업 재할당한다.

어떤 호스트의 이탈 시점에 대기 호스트가 존재하지 않으며, 잔여 태스크 정보를 미작업 엔트리 스택에 푸시한다. 차후 대기 호스트에 의한 재할당 연산으로 전체 연산을 마친다.

또한, 갑작스런 시스템과 네트워크의 오류로 사전 예고없이 호스트가 이탈하는 경우를 고려한다. 호스트의 연결 상태 및 작업 정보를 저장하는 VM를 시스템에 참여한다. RP의 참여 의사 전달 시점에 통신을 위한 세션과 쓰레드 형태로 VM를 생성한다. 통신 제어권을 VM에게 넘겨줘 RP와 RM간의 중계자(communication broker) 역할을 한다. 그러므로, 세션 상태를 모니터링하면서 호스트의 결함 여부를 관측한다. 동시에 VM에게 처리된 작업 식별자를 전달한다. 따라서, RP의 결함 발생에 대해 잔여 태스크 정보를 추출한다.

4.2 동적 RP 추가 메커니즘

4.2.1 미작업 엔트리 스택에 태스크 존재시 RP 추가

호스트 결함 및 이탈로 시스템 내부의 미작업 엔트리 스택에 처리해야할 작업이 남았을 때 새로운 호스트를 추가할 경우이다. 추가된 RP가 초기 할당받을 태스크는 미작업 엔트리 스택의 태스크로 설정한다. 추가 호스트의 참여 정보를 얻은 RM은 대응하는 VM를 동적으로

생성한다. 접속된 세션과 미작업 엔트리 스택의 상위 계층에 있는 미작업 그룹을 추가 호스트에게 할당한다. 추가 메커니즘이 끝난 후에는 정상적인 적응적 작업 재할당 기법을 수행한다.

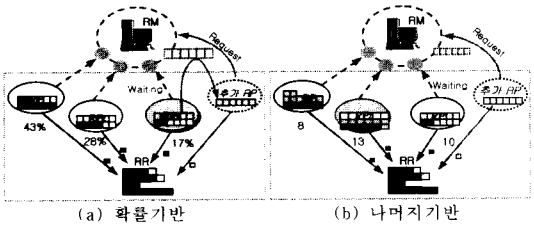
4.2.2 모든 호스트가 연산 중에 RP 추가

새로운 자원의 참여에 관한 메시지를 받았지만 모든 RP가 할당받은 연산을 수행하는 상태에서 미작업 엔트리 스택이 비어있는 상태의 호스트 증가이다. 각각 RP를 모니터링하는 VM가 가진 연산 정보에 의해 가장 낮은 성능으로 전체 연산에 영향을 주는 RP를 추출한다.

가장 낮은 성능의 RP를 추출하는 방법으로, 전체 작업량의 진행 상태를 작업률(task ratio)을 기준으로 하는 확률 기반 동적 재할당(probability-based dynamic task reallocation)과 처리해야 할 작업수(task size)에 의한 나머지 기반 동적 재할당(remain-based dynamic task reallocation)으로 구분한다.

그림 4의 (a)는 3대의 호스트가 각각 43%, 28%, 17%의 작업률로 가장 낮은 처리율을 가진 RP_3을 재할당 대상으로 추출한다. 이때, 추출된 RP_3의 미수행된 태스크의 일부를 새로운 RP에게 이행한다. 하지만, 작업률에 의존하는 추출은 잦은 호스트 추가가 일어날 경우에는 한쪽에 집중적인 재할당 연산으로 병목현상(bottleneck)이 발생한다.

이러한 병목현상의 요인을 제거하기 위하여 새로운 호스트 추가시 가장 많은 잔여 작업수에 의한 재할당 호스트를 추출하는 기법으로 그림 4의 (b)와 같은 나머지 기반 동적 재할당을 제안한다. 3개의 RP는 각각 남은 태스크 수가 8, 13, 10으로 잔여 작업수가 가장 많은 RP_2의 태스크 일부를 추가한 호스트에게 할당함으로써 한 호스트에 절대적인 영향을 미치는 부분이 제거된다.



(a) 확률기반 (b) 나머지기반

그림 4 호스트 추가 기법

또한, 모든 호스트가 할당된 작업을 100%에 근사하게 끝마쳤을 때 새로운 호스트 참여를 고려한다. 재할당 연산 시간을 고려하여 현재 진행하고 있는 호스트에 의해 연산을 지속한다. 추가 RP는 작업에 참여하지는 않지만, 진행 호스트의 결함 및 이탈을 감안하여 대기한다.

앞에서 언급한 바와 같이 연산이 진행되는 동안 RP 수는 수시로 변한다. 본 논문에서는 RP를 관리하는 각각의 VM들을 임의의 키 값에 의한 벡터 형태로 관리하여 동적인 호스트 추가와 이탈에 대한 RM의 과부하를 최대한 줄인다.

5. PDSWeb 시스템 설계 및 구현

5.1 PDSWeb 전체 구조 및 프로토콜

본 논문에서 제안하는 시스템의 가상 상위 패키지를 PDSWeb이라고 정의하며, 그림 5는 구성요소간의 상호관계를 나타낸다. 구성요소들의 패키지는 ResReq, Manager, ResPro, 그리고 어플리케이션을 위한 App로 나뉜다. 패키지들간의 통신은 하부 객체인 세션이나 JavaRMI를 통하여 메시지 전달과 원격 메소드 호출을 사용한다.

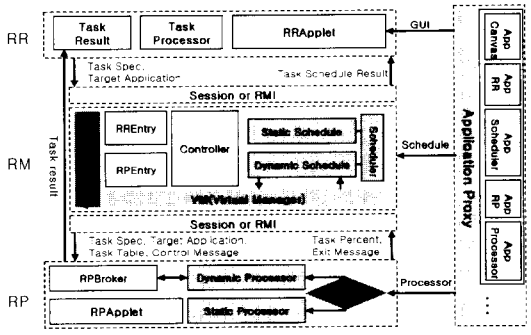


그림 5 PDSWeb 구조

특히, 본 논문에서 제안한 호스트의 동적 관리에 해당하는 DynamicProcessor 객체를 이용할 경우 수시로 작업 중지와 재시작을 한다. 이로 인해, 자바에서 제공하는 쓰레드 중지와 시작에서 발생하는 시스템의 과부하를 최대한 줄이기 위하여 그림 6과 같이 인터럽트(interrupt)를 이용하여 쓰레드를 관리한다.

작업 수행은 그림 1에서 정의된 태스크 단위로 수행되고, is_Continue에 의해 연속적인 작업 여부를 확인한다. RP가 할당받은 작업을 모두 마치고 RM의 일시 중지 명령을 얻으면, is_Continue를 변환하여 쓰레드는 대기 상태가 되고 RM의 추가 명령을 기다린다. 추가 작업이 전송되면, RPBroker는 DynamicProcessor 클래스의 New_Task_Assign() 메소드 호출로 is_Continue를

변환하여 대기 상태의 쓰레드가 해제되어 일시중지 및 재시작이 동적으로 이루어지게 한다. RM에 의해 작업 종료 메시지를 전송 받으면, 해당 쓰레드의 isExit 값을 변환하여 쓰레드가 종료된다.

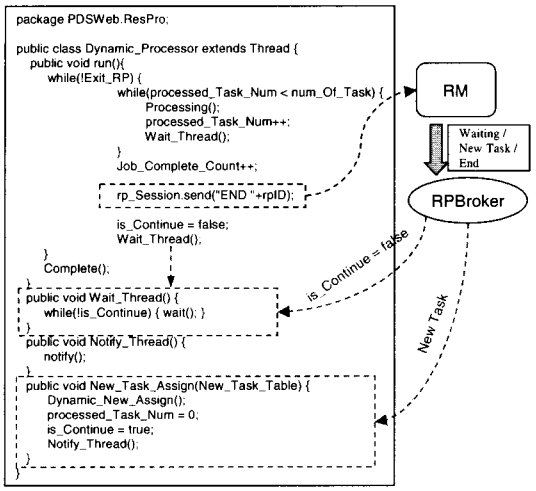


그림 6 동적 쓰레드 관리

그림 7은 PDSWeb 시스템의 초기 등록에서부터 모든 작업이 완료되기까지 각 패키지간의 메시지 전달 프로토콜이다.

RR와 RP는 RM로부터 다운받은 애플릿을 통하여 자신을 등록한다. 요청한 작업의 어플리케이션 이름과 작업 명세를 RM에게 보낸다. 이때, 원격 객체 참조값을 RM의 인터페이스를 통해 등록한다. RR에 의해 JS(job spec.)을 얻은 RM은 작업을 분배한다. 할당받은 RP는 연산을 수행하고, 연산 결과는 RM으로부터 가져온 RR의 원격 객체 참조값을 이용하여 RR에게 즉시 전송한다. 동시에 RM에게는 정상적으로 연산을 수행하고 있다는 정보로 식별자인 TaskID를 전달한다. 만약, 더 이상 재할당을 할 필요가 없을 경우에는 중지(Stop) 메시지를 RP에게 보내어 작업을 마친다. 재할당 연산이 이루어진 경우에는 ReA 메시지와 함께 새로운 작업 정보를 전달하여 (c)의 단계로 돌아간다. 이외에 도중에 일시 중지해야 할 경우에는 (f)와 같이 Wait 메시지를 전달하여 일시 중지 명령을 전달한 후 ReA 메시지를 이용하여 추가 작업을 전달한다.

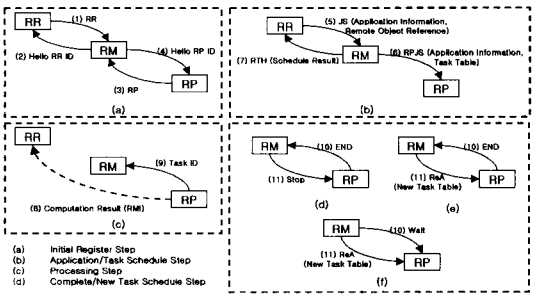


그림 7 PDSWeb 통신 프로토콜

5.2 PDSWeb 시스템 구현

PDSWeb 시스템은 순수 자바코드를 사용하여 구현한다. RR과 RP는 애플릿 환경에서 사용되어지며 인터넷 익스플로러(Internet Explorer)와 넷스케이프(Netscape)의 내장된 JVM(Java Virtual Machine)을 사용한다.

PDSWeb 시스템은 모두 7개의 기본 패키지와 적용 어플리케이션의 수에 따라 하나씩 증가하는 패키지로 구성된다. 다음은 각 패키지의 주요 클래스이다.

- Config 패키지
  ConfigEnv : PDSWeb 시스템의 환경 정보를 저장하고, 모든 클래스가 공통으로 참조
- Manager 패키지
  PDSWebMonitor : 시스템의 전략 설정 및 컴퓨팅 정보를 알려주는 인터페이스
  VirtualManager : RP 관리
  RPEntry : RP 정보 저장
  RREntry : RR 정보 저장
  PDSWebManager : RM의 초기 구동과 RR, RP의 등록
  SchedulerRP : 특정 어플리케이션의 할당 알고리즘을 정의
  DynamicScheduler : 적응적 작업 재할당 및 동적 호스트 관리 전략을 수행
- SharedInterface 패키지
  RemoteTaskResult : 연산 결과를 RR에게 직접 전달하는 메소드 정의
- ResReq 패키지
  TaskProcessor : 연산을 위한 메소드 정의
  RRApplet : RR 인터페이스
- ResPro 패키지
  DynamicProcessor : 적응적 작업 재할당 및 동적 호스트 관리 전략 연산
  BenchMarker : RP의 성능 평가
  RPBroker : 동적 연산을 수행시 RM과의 메시지 전달에 대한 중계자 역할
  RPApplet : RP 인터페이스

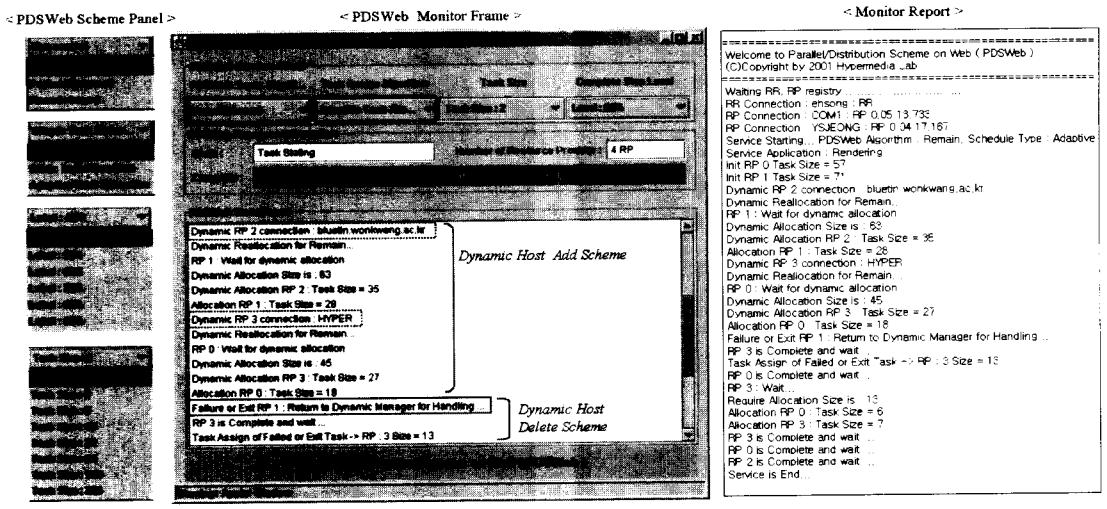


그림 8 RM 실행화면

TaskProxy : 특정 어플리케이션의 연산 알고리즘을 위한 메소드 정의

- App 패키지

[App]Processor : 어플리케이션 연산에 대한 정보 제공

[App]ProcessProxy : 어플리케이션 연산에 필요한 알고리즘 수행

[App]Scheduler : 어플리케이션을 스케줄링 하는데 필요한 정보 제공

- Session 패키지

SessionBroker : 구성요소의 세션을 넘김

5.3 PDSWeb 시스템 동작

그림 8은 본 논문에서 구현한 시스템의 RM 실행화면이다. 초기에 클래스 로딩으로 JavaRMI 기능을 하기 위한 레지스트리 데모와 관련된 설정을 시작하여 모니터 프레임을 부른다. RM은 3개의 인터페이스로 구성된다. 제안한 전략을 설정하는 가장 상위 인터페이스(PDSWeb Scheme Panel), 시스템의 현재 상태 및 참여 RP수와 작업 진행률을 나타내는 인터페이스(PDSWeb Resource Information)와 연산에 대한 모니터링 인터페이스(PDSWeb State Panel)이다.

초기 2개의 RP로 연산을 수행하는 도중에 새로운 RP_2(bluetin.wonwang.ac.kr)와 RP_3(HYPER)가 추가되어 기존에 연산중이던 호스트인 $RP_{3,1}$과 재할당한다. 또한, 연산중이던 RP_1(YSJEONG)이 갑작스런 오류 등으로 현재 시스템을 이탈하여 할당받았지만 미수행된 태스크를 RP_3에 의해 처리되는 과정을 보인다. 이는 본 논문에서 제안한 동적 호스트 관리 기법에 의해 호스트의 추가 및 이탈이 발생하였을지라도 작업이 정상적으로 처리됨을 보여준다.

그림 9는 웹 상에서 수행되는 RP의 수행화면이다. 하나의 컴퓨터에 하나의 RP가 할당받은 태스크를 수행한다고 가정한다. 참여 RP의 시스템 성능과 등록 정보, 그리고 연산 정보와 참여 여부로 구성된다. 그림 9의 작업 처리 과정은 초기에 성능비대로 할당받은 연산을 마친 후, 또 한번의 재할당이 발생하였으며 수행 시간이 42661m/s이다.

RR은 어플리케이션과 다양한 작업 스펙, 그리고 이에 의해 생성되는 GUI로 구성된다. 그림 10은 연산중에 호

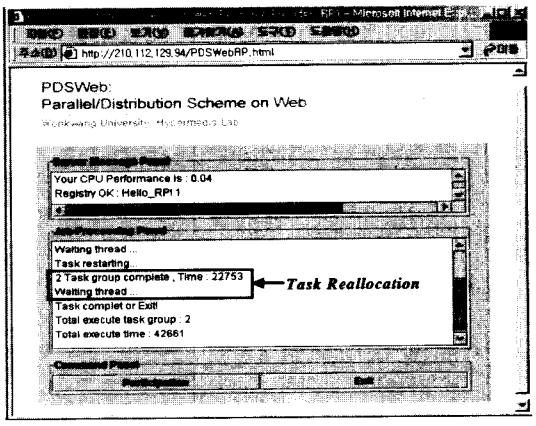


그림 9 RP 실행화면

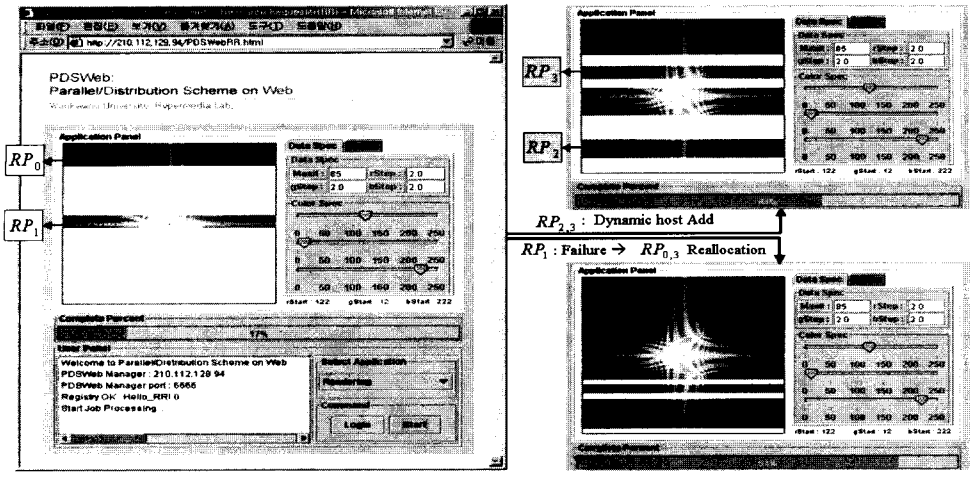


그림 10 동적 호스트 추가 및 이탈하는 RR의 실행화면

스트의 추가와 이탈이 발생하여 동적 호스트 관리 기법에 의한 연산 상태을 나타낸다. 초기에 $RP_{0,1}$로 시작하여 실행 중에 새로운 $RP_{2,3}$가 추가된다. 또한 추가 호스트를 포함해 4개의 호스트에 의한 연산중에 RP_1의 이탈이 발생한다. 이것은 그림 8의 작업 명세서에 준한다.

RR측 상태는 호스트의 추가와 이탈 및 재할당 여부를 알 수 없다. 단지 가시적 이미지 처리 상태를 추측하여 PDSWeb 시스템의 전략에 대한 RR측의 지역 투명성(location transparency)을 유지한다.

6. 성능 평가

구현한 PDSWeb 시스템을 통해 제안한 작업 할당 및 동적 호스트 관리 전략에 대한 성능 평가 결과이다. 성능 평가를 위해 적용된 어플리케이션으로 랜더링 이미지를 생성한다.

표 1 성능 평가 표시에 따른 전략

정의	전략
Simple	단순 작업 할당 기법
Static	성능 기반 작업 할당 기법
Adaptive	적응적 작업 재할당 기법
Probability	확률 기반 동적 호스트 관리
Remain	나머지 기반 동적 호스트 관리

6.1 호스트 수의 증가에 따른 성능평가

먼저 동질 호스트에서의 성능과 웹을 고려한 이질 호스트에서 호스트 증가에 따른 성능을 비교 분석한다.

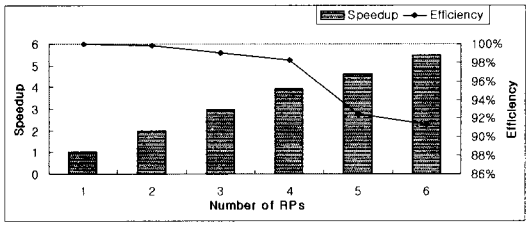


그림 11 동질 호스트의 스피드업 및 효율성

그림 11은 동일한 성능(CPU(PII266MHz))을 가진 호스트 수의 증가에 따른 성능 향상이다. 호스트 수가 증가하면 그에 따라 스피드업도 증가한다. 하지만, 스피드업의 증가도는 서서히 줄어든다. 이는 병렬 통신과 동기화에 일어나는 손실이 호스트를 추가함으로써 일어나는 이익만큼 커지기 때문이다. 즉, 추가 호스트당 효율성은 떨어짐을 의미한다.

표 2 이질 호스트 성능평가에 참여한 컴퓨팅 자원

역할	CPU	OS
RM	PIII 500	Windows 2000 Server
RR	PIII 500	Windows me
RP1	**PII 300**	**Linux 6.1**
RP2	**PII 266**	**Windows 98**
RP3	**PII 500**	**Windows 2000 Server**
RP4	**PIII Xeon**	**Windows 2000 Server**
RP5	Sun SPARC station20	Sun Solaris 2.7
RP6	PII 400	Windows 2000 Professional

웹에 상주하는 호스트의 이질성을 고려하여 표 2와 같은 컴퓨팅 환경을 이용한다. 작업 할당 전략으로 참여 RP의 수에 비례하여 균등하게 할당하는 단순 할당, 초기 간단한 평가에 의해 성능비대로 할당하는 성능 기반 할당, 가변적인 환경에 재할당을 추가한 적응적 할당을 비교한다(그림 12). 단순 할당의 경우 스피드업이 2를 넘지 못하며 매우 낮은 효율성을 보인다. 하지만, 작업 개시 Mflop/s 결과에 의한 성능 기반 할당은 연산 도중 성능 변화에 무관하게 5.9를 넘는 비교적 효율성이 높다.

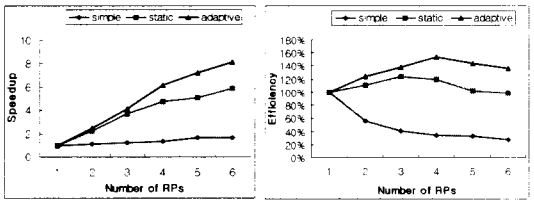


그림 12 이질 호스트의 스피드업 및 효율성

6.2 성능 변화에 따른 성능평가

웹의 또 다른 성격으로 참여한 호스트들의 성능변화를 예측하기 어렵다. 그림 13은 연산 도중 성능이 수시로 변하는 경우와 변화가 거의 없는 경우의 응답 시간이다. 성능평가 환경은 표 2에서 정의한 RP중 4대를 이용한다. 성능변화가 거의 없는 경우 적응적 재할당 기법이 24.12%의 성능 향상을 보인다. 이것은 웹의 가변성으로 RP의 초기 CPU 성능에 부응하는 연산의 지속성을 기대하기 어렵다. 즉, 벤치마크 결과를 100% 신뢰할 수 없다.

RP_2가 매우 심하게 성능변화를 일으킨다고 가정하자. 불규칙적인 성능 변화에 대해 적응력을 추가할 때, 재할당 수행 유무를 결정하는 레벨을 80%로 설정하고 적응적 작업 재할당 기법을 적용한 평가이다. 성능 기반 할당에 비해 최고 94.28% 향상한다. 작업 실행 도중 RP의 가변성에 적절한 재할당이 이루어져 높은 성능 향상을 얻었음을 알 수 있다.

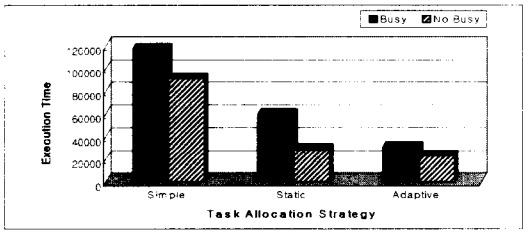


그림 13 수행능력 변화에 따른 성능 향상

6.3 동적 호스트 추가 및 이탈에 따른 성능평가

그림 14는 추가 호스트에 따른 확률 기반과 나머지 기반 동적 재할당의 성능 비교이다. 새로운 RP의 증가에 따라 나머지 기반 동적 재할당의 성능이 향상한다. 이것은 확률 기반 전략에서 나타나는 병목 현상으로 새로운 호스트가 추가될 때, 적절한 재할당 대상이 되는 RP를 추출하지 못하였기 때문에 발생한 현상이다.

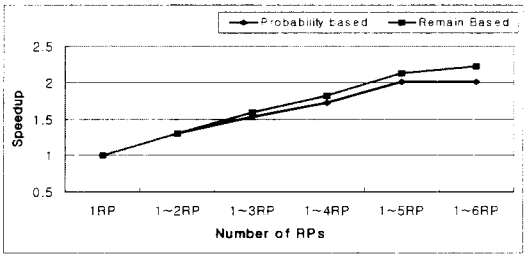


그림 14 동적 호스트 추가에 따른 성능향상

그림 15는 작업 도중 호스트의 이탈 시기에 따른 성능 비교이다. RP 동적 추가와는 반대로 이탈 시기가 늦을수록 컴퓨팅에 참여하는 시간이 길어져 빠른 연산시간을 보인다.

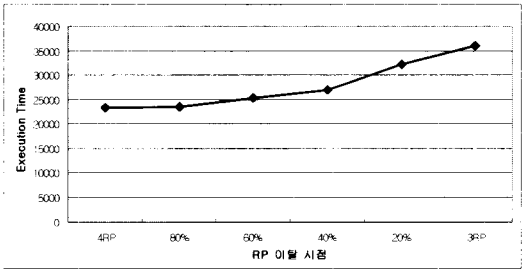


그림 15 동적 RP 이탈 시점에 따른 성능평가

본 논문은 결함 및 이탈로 참여 호스트가 감소할지라고 재할당으로 전체 연산이 정상 종료되는 결함내성을 보이며, 이탈된 호스트를 제외한 남은 호스트에 의해 적응적 작업 재할당을 수행함으로써 전체 연산 시간을 줄인다.

7. 결 론

본 논문은 LAN이라는 지역적으로 한정된 곳에서 이루어지고 있는 현재, 병렬 컴퓨팅의 제한된 범위를 벗어나 웹이라는 컴퓨팅 자원을 이용한다. 방대한 연산을 요구하는 어플리케이션을 병렬 수행하기에 필요한 요소들

과 해결해야 할 문제들에 대하여 알아보았고, 이에 대한 대처 방안으로 성능기반 작업 할당, 적응적 작업 재할당 및 동적 호스트 관리 스킴을 제안하였다. 또한 이를 기반하에 PDSWeb을 구현하여 성능 평가 및 검증하였다.

웹에 연결되어져 있는 서로 다른 성능의 컴퓨팅 자원을 벤치마크하여 각 자원 제공자들의 성능에 맞게 작업을 할당한다. 뿐만 아니라, 급격한 환경 변화에 대처하기 위한 적응적 작업 재할당 기법으로 수많은 환경 변화에 노출되어 있는 웹의 특성을 고려한 병렬 컴퓨팅 전략을 설계하였다.

작업을 수행하고 있는 도중에 새로운 자원을 추가 및 삭제가 가능하게 하였다. 추가함으로써 컴퓨팅 파워가 향상되어 전체 수행시간이 줄었음을 평가하였다. 또한 연산 중에 컴퓨팅 자원의 감소로 전체 시스템에 미치는 영향을 대비하는 동적 호스트 관리 기법을 제안하여 평가하였다.

마지막으로 다양한 어플리케이션을 적용하여 어플리케이션이 제시한 전략의 타당성을 보였다. 특히, 다양한 어플리케이션 적용을 위해 의존적인 부분을 병렬처리 계층과 분리하여 어플리케이션에 관련된 객체들은 필요시에 동적으로 생성하는 동적 객체 생성 메커니즘을 이용하여 독립적인 어플리케이션 개발을 효율적으로 지원하였다.

향후 연구 과제로서는 고성능 통신망으로 대량 데이터를 처리하기 위해 단일 수퍼컴퓨터화하는 그리드컴퓨팅(grid computing)이 가능하여, 자원을 수평 공유하고자 하는 향상된 그리드 기반요소의 개발이다. 본 논문에서 지원한 어플리케이션에 독립된 특징을 계속해서 이어 다양한 어플리케이션의 개발하여 성능을 비교 분석하여 어플리케이션 특성에 의존하지 않는 시스템 개발을 위한 전략적 연구가 필요하다. 그리고, 본 논문에서는 고려하지 않았던 서브 작업간에 상호 연관성에 대해 동적 호스트 관리 기법에 메시지 패싱 기법을 적용이 필요하다.

참 고 문 헌

[1] 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.

[2] A. Vahdat, P. Eastham, C. Yoshikawa, E. Bealani, T. Anderson, D. Culler, and M. Dahiln, "WebOS : Operating System Services For Wide Area Application," Technical Report CSD-97-938, UC Berkeley, 1997.

[3] P. Michael, and Z. Matthies, "JavaParty : Transparent Remote Objects in Java," In the ACM Workshop on Java for Science and Engineering Computation, 1997

[4] A. Baratloo, M. Karaul, Z. M. Kedem, and P. Wychoff, "Charlotte : Metacomputing on the Web," The 9th International Conference on Parallel and Distributed Computing Systems, Dijon, 1996.

[5] 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, 1996.

[6] 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.

[7] 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.

[8] 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.

[9] D. Caromel, W. Kauser, and J. Vayssiere, "Java// : Towards Seamless Computing end Metacomputing in Java," Concurrency Practice and Experience, p.1043-1061, 1998

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

송 은 하

1997년 원광대학교 통계학과 이학사. 2000년 원광대학교 컴퓨터공학과 공학석사. 2001년 ~ 현재 원광대학교 컴퓨터공학과 박사과정. 관심분야는 병렬분산시스템, 병렬처리, 그리드컴퓨팅.

정 영 식

1987년 고려대학교 수학과 이학사. 1989년 고려대학교 전산과학과 이학석사. 1993년 고려대학교 전산과학과 이학박사. 1998년 미시간주립대학교 교환교수. 1993년 ~ 현재 원광대학교 컴퓨터 및 정보통신 공학부 부교수. 관심분야는 병렬분산시스템, 컴퓨터시뮬레이션, 그리드 컴퓨팅, 이동컴퓨팅.