자바를 기반으로 한 글로벌 인터넷 컴퓨팅 환경

김 희 철† · 신 필 섭†† · 박 영 진††† · 이 용 두†

요 약

본 연구에서는 한정된 자원만을 사용하는 기존의 워크스테이션 클러스터링 환경의 제한성을 극복하기 위하여 인터넷에 연결된 혼합 이기종 컴퓨터들을 병렬 컴퓨팅 플랫폼으로 활용하고자 하는 인터넷 컴퓨팅 환경의 구축 기법을 제안한다. 제안하는 글로벌 인터넷 컴퓨팅 환경(Global Internet Computing Environment)은 프로그램의 용이성, 이기종 지원의 효율성, 시스템의 확장성, 그리고 시스템 성능에 초점을 두고 자바를 프로그래밍 및 수행 환경으로 채택하여 인터넷 컴퓨팅 구축에 있어 필수적인 동적 자원 중계 및 관리, 효율적인 병렬 태스크 수행 기법을 제시한다. 본 논문에서는 제안하는 글로벌 인터넷 컴퓨팅 환경의 구성모델 및 동작모델, 그리고 시험시스템 구축 내용 및 벤치마킹을 통한 성능평가 결과를 제시하며 이를 바탕으로 인터넷 컴퓨팅 환경의 구축 개념, 복잡성, 성능의 문제에 대한 분석결과를 기술한다.

Global Internet Computing Environment based on Java

Hie-Cheol Kim† · Pil-Sup Shin†† · Young-Jin Park††† · Yong-Doo Lee†

ABSTRACT

Over the Internet, in order to utilize a collection of idle computers as a parallel computing platform, we propose a new scheme called GICE (Global Internet Computing Environment). GICE is motivated to obtain high programmability, efficient support for heterogeneous computing resources, system scalability, and finally high performance. The programming model of GICE is based on a single address space. GICE is featured with a Java based programming environment, a dynamic resource management scheme, and efficient parallel task scheduling and execution mechanisms. Based on a prototype implementation of GICE, we address the concept, feasibility, complexity and performance of Internet computing.

1. 서 론

과학 계산용 문제들이나, 3차원 그래픽 계산 등의 다양한 응용분야에서 대규모의 컴퓨팅 파워를 요구하는 어플리케이션의 병렬 처리를 위한 플랫폼으로서 네트워크에 연결되어 있는 컴퓨팅 시스템들을 하나의 논리적인 컴퓨팅 자원으로 사용하고자 하는 네트웍 기반의 병렬처리를 위한 노력이 1990년대 초반부터 매우 활발하게 진행되고 있다. 이러한 노력은 먼저 연구소나 대학 내의 네트워크 상의 워크스테이션들을 활용한 워크스테이션 클러스트링 환경의 구축에 초점이 두어졌다. 그 대표적인 연구로는 NOW[1], Linda[2], PVM[3], MPI[4], JPVM[6] 등이 있다. 버클리 대학의 NOW 프로젝트는 워크스테이션들을 이용하여 병렬 어플리케이션을 수행 할 수 있는 장점을 가지고 있지만, 자동적으로 시스템을 운용하기 위한 소프트웨어가 지원되지 않으므로 사용자 측면에서 프로그램 작성과 관리가 용이하지 않은 단점을 가지고 있다. 이를 해결하

※ 이 논문은 1997년 한국학술진흥재단 학술연구조성비에 의하여 지원되었음.
† 정 회 원 : 대구대학교 정보통신공학부 교수
†† 준 회 원 : 연세대학교 대학원 컴퓨터과학과
††† 정 회 원 : 대구대학교 대학원 정보통신공학부
논문접수 : 1999년 6월 12일, 심사완료 : 1999년 7월 28일

위한 시도로서 PVM과 MPI는 현존하는 프로세서를 감시해서 라우팅하고 메시지를 전송하는 일련의 일들을 처리하는 런타임 시스템을 제공한 후 그 상위의 API(Application Programming Interface)를 제공함으로써 프로그래머로 하여금 런타임 환경의 세부 내용에 대한 고려 없이도 병렬프로그램을 용이하게 작성할 수 있도록 하였다. 그러나 이러한 워크스테이션 클러스터링 환경 하에서는 병렬 컴퓨팅에 참여하는 모든 시스템 상에 계정이 필요하며 수행될 프로그램이 각 시스템 상에서 재컴파일되어 하드디스크에 저장되어야 하며 사용할 수 있는 컴퓨팅 자원이 지역 네트워크로 제한을 받는 단점을 갖는다.

한편 최근에는 인터넷의 급속한 발전과 광대역 통신망의 보급으로 인한 인터넷 통신 용량의 확장 등에 힘입어 인터넷에 연결된 다양한 컴퓨터들의 유휴 시간을 효과적으로 이용하고자 하는 인터넷 기반의 병렬 컴퓨팅 환경(본 논문의 나머지 부분에서는 *인터넷 컴퓨팅 환경*이라 칭함)의 구축에 관한 연구가 진행되고 있다[5, 7-12]. 인터넷 컴퓨팅 환경은 워크스테이션 클러스터링 환경과 마찬가지로 기존에 존재하는 컴퓨팅 자원을 활용하므로 비용 대 성능비가 매우 높다는 점과 인터넷 상의 컴퓨터들을 활용하게 됨으로 성능 증대에 필요한 컴퓨팅 자원을 매우 유연하게 확보할 수 있다는 장점을 갖는다.

인터넷 컴퓨팅 환경 구축과 관련된 대표적인 연구로는 Santa Barbara 대학의 Javelin[7], York 대학의 ParaWeb[8], New York 대학의 Charlotte[9] 등이 있으며, 모두 웹브라우저를 사용자 인터페이스로 이용하여 병렬 수행 환경을 제공하며 인터넷 환경에서의 자원의 효율적인 활용을 위해 플랫폼 독립적인 장점을 갖는 자바를 사용하고 있다. Javelin은 애플릿간의 직접 통신이 가능한 라이브러리를 구축하여 메시지 패싱 방식으로 병렬 태스크들간의 통신이 이루어지기 때문에 프로그래머가 병렬 프로그램을 작성하기가 쉽지 않다는 문제점을 가지고 있다. ParaWeb은 순수 자바를 사용하지 않고 자바 가상 머신의 수정을 통해 병렬 컴퓨팅 환경을 구현하였기 때문에 확장성에 문제를 가지며, Charlotte는 웹을 병렬 컴퓨팅 자원으로 활용하기 위하여 공유 메모리와 인터페이스를 제공하지만 eager scheduling 기법을 사용하기 때문에 병렬로 수행되는 각 작업간에는 공유메모리를 통한 통신을 지원하지 못하는 단점을 가진다. 현재 이러한 인터넷 컴퓨팅 환경의 구축을 위한 일부분은 각기 인터넷 컴퓨팅 환경의 구축에 필요한 특정 기술 부분들에 한정하여 수행되고 있으므로 포괄적인 인터넷 컴퓨팅 환경의 시험 시스템 구축 보다는 부분적으로 연구실 단위 정도의 소규모 컴퓨팅 환경에 몇 가지 모듈을 구현한 부분적 인터넷 컴퓨팅 환경의 구축이 진행되고 있는 정도이다.

본 연구에서는 앞에서 지적한 기존 방법들의 문제점들을 해결할 뿐만 아니라 혼합 이기종성과 확장성 그리고 프로그램 작성 면에서 효율적인 인터넷 컴퓨팅 환경의 구축 기법을 제안한다. 본 연구의 접근 방법으로 먼저 인터넷상의 혼합 이기종 컴퓨팅 환경을 효율적으로 지원하기 위하여 기존의 인터넷 컴퓨팅 방식들이 채택하고 있는 웹 브라우저(web browser)와 애플릿(applet) 기반 대신에 이식성이 좋고 머신에 독립적인 자바 언어[13]를 기반으로 하는 프로그래밍 환경을 채택하였다. 그리고 병렬 프로그램 작성이 용이하도록 단일 어드레스 공간을 갖는 공유 메모리[14] 기반의 프로그래밍 모델을 채택하였다.

이러한 모델에 대하여 병렬컴퓨팅 표준환경의 설계에 요구되는 시스템의 구성 메카니즘, 작업 관리와 운영에 요구되는 환경설계와 시험 시스템을 구축하였다. 이러한 시험 시스템 상에서 데이터의 병렬성을 갖는 응용 프로그램을 선정하여 성능 분석을 수행하였다. 주요 실험 결과로서 대규모 데이터의 병렬성과 병렬로 분할 수행되는 각 태스크들간에 낮은 상관성을 보이는 응용 프로그램들에 대해 작업 수행 컴퓨터의 수가 증가할수록 거의 선형적인 성능 향상을 얻을 수 있음을 볼 수 있었다. 그러나 병렬로 분할되어 수행되는 태스크들간에 통신의 양이 증가할수록 통신에 소요되는 오버헤드의 급격한 증가로 시스템의 성능과 효율이 저하되는 점을 볼 수 있었다.

본 논문의 구성은 다음과 같다. 2장에서는 본 연구에서 제안하는 인터넷 컴퓨팅 환경의 기본 구성, 운영 방법, 그리고 동작 모델을 제시하며 3장에서는 그 구현에 관해 설명한다. 4장에서는 제시된 시스템 구성에 대한 평가를 위해 병렬 수행 특성을 반영하는 실제 벤치마크를 사용한 실험을 통한 성능 평가 결과를 기술한다.

2. GICE : 제안하는 인터넷 컴퓨팅 환경

인터넷 컴퓨팅 환경은 네트워크 기반의 컴퓨팅이라

는 점에서 워크스테이션 클러스터링 환경과 공통점을 갖는다. 그러므로 인터넷 컴퓨팅 환경의 구축을 위해서는 지역 네트워크 기반의 워크스테이션 클러스터링 환경에서 고려해야 하는 프로그래밍의 용이성, 스케줄링, 보안 등과 같은 문제점들을 해결하여야 한다.

그러나 인터넷은 공유된 파일 시스템이 없다는 점과, 모든 사용 머신에 대한 계정을 가지지 못한다는 점, 그리고 공통의 아키텍처가 없다는 점에서 지역 네트워크 환경과는 구별되는 성격을 갖는다. 또한 다양한 기종의 컴퓨터 시스템들이 연결된 동적 네트워크의 특성을 갖는 인터넷 상에서는 각 컴퓨터 시스템들의 사용 가능 여부와 네트워크의 지연시간 등을 미리 예측할 수가 없다. 그러므로 인터넷 컴퓨팅을 위해서는 먼저 워크스테이션 클러스터링 환경을 위해 해결하여야 하는 여러 문제점들을 동적인 컴퓨팅 자원 관리, 이기종성 및 이식성들을 고려하여 보다 포괄적으로 해결할 수 있어야 한다. 나아가 동적 컴퓨팅 자원을 최적으로 활용할 수 있도록 해주는 자원 배분 기능이나, 병렬 처리 작업을 현재 사용 가능한 머신들 중에서 가장 효율적으로 수행될 수 있는 머신들을 선택할 수 있는 스케줄링 기능과 프로그램 수행 중 발생할 수 있는 시스템 오류를 감지하여 동적으로 시스템을 재구성할 수 있는 기능의 제공이 필수적이다.

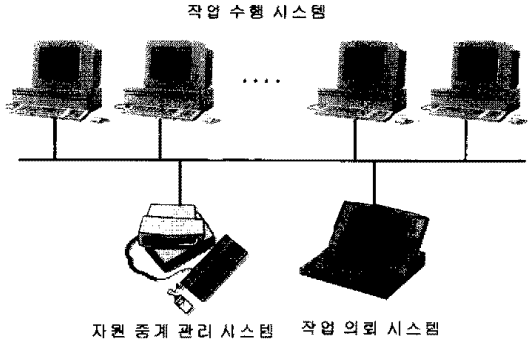
본 연구는 초고속 통신망으로 연결된 인터넷 상의 혼합 이기종 컴퓨팅 환경에서 응용프로그램의 개발 및 분산·병렬 실행을 효율적으로 지원하기 위한 표준환경의 구성 및 분석을 최종 목적으로 한다. 이러한 목표를 달성하기 위하여 본 연구에서는 인터넷 컴퓨팅을 위한 논리적 핵심 요소인, 이기종 컴퓨팅 환경 지원, 프로그래밍 모델, 프로그래밍 용이성 측면에서 우수한 글로벌 인터넷 컴퓨팅 환경(GICE : Global Internet Computing Environment)을 제안한다. 본 절에서는 GICE의 병렬 프로그래밍 모델, 시스템 구성 및 동작 모델, 그리고 그 구현 내용에 관하여 설명한다.

2.1 GICE의 병렬 프로그래밍 모델

인터넷 환경에서 수행될 병렬 프로그램을 작성할 때, 수행환경 즉, 참여하는 시스템들의 수, 위치, 시스템 플랫폼 및 사양, 그리고 네트워크의 상태 등을 사전에 알 수가 없다. GICE 사용자가 인터넷 환경 하에서 보다 쉽게 프로그램을 작성할 수 있도록 가상머신 기반의 수행환경을 갖는 자바를 프로그래밍 개발 및 수행 환경으로 채택하고 있다. 한편 GICE에서는 단일 어드레스 공간을 갖는 공유메모리 모델을 병렬 프로그래밍 모델로서 채택하고 있다. 공유메모리 모델은 메시지 통신 기반의 분산메모리 구조보다 병렬프로그램의 작성이 매우 용이한 장점을 제공하며 자바 개발 환경은 순수 자바 언어로 작성된 프로그램의 작성 및 수행 환경을 제공하기 때문에 GICE는 참여한 시스템들의 이기종성을 효율적으로 지원하여 준다. GICE에서의 공유메모리 구현에 관한 내용은 본 절의 나머지 부분에서 자세히 설명한다.

2.2 GICE 시스템 구성

GICE는 인터넷에 연결된 독립된 컴퓨터들을 이용하여 주어진 응용프로그램을 병렬로 수행하기 위한 환경이다. GICE는 인터넷에 연결된 다양한 시스템들로 구성된 가상 멀티컴퓨터 환경을 제공하며 각 시스템은 멀티컴퓨터에서 하나의 노드에 해당한다. GICE에서 주어진 노드는 그 역할에 따라 작업 의뢰 시스템(JRS : Job Requesting System), 작업 수행 시스템(JPS : Job Processing System), 또는 자원 중계 관리 시스템(RMS : Resource Management System)으로 구분된다(그림 1). JRS는 사용자가 병렬로 수행하고자 하는 자바 응용 프로그램을 작성해서 RMS로부터 GICE에 참여한 JPS의 정보를 이용하여 각 JPS에게 컴퓨팅을 요청하는 시스템이고, JPS는 JRS의 요청에 따라 작업을 수행하는 시스템이다. 그리고 RMS는 JRS에게 등록된 JPS에 대한 정보 제공의 기능을 담당하는 시스템이다.



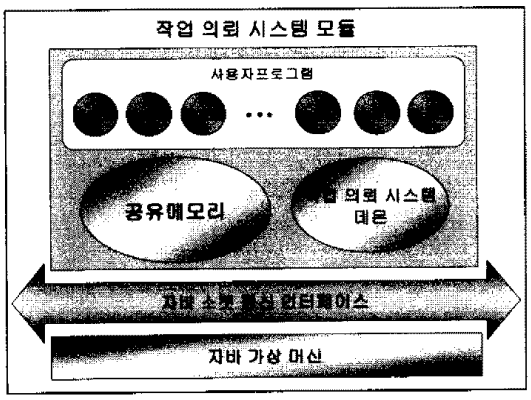
(그림 1) GICE 구성

임의의 시점에 GICE에 참여하는 시스템들은 하나의 가상머신을 구성하여 주어진 병렬프로그램을 수행하게 된다. 참여 시스템들의 자원들과 주소들을 관리하는

RMS를 제외한 GICE에 참여하는 각 시스템은 동등한 기능과 권한을 가지고 서로의 자원을 공유하며, 수행 과정에 따라 JRS 또는 JPS로서의 기능을 담당한다.

2.1.1 작업 의뢰 시스템(JRS)

JRS는 GICE에서 작업 수행을 요청하는 사용자 시스템이며 동시에 작업을 수행할 때는 병렬 수행 프로그램이 수행되는 시스템이다. JRS의 프로그래밍 환경은 사용자가 이미 구축한 자바 프로그래밍 환경으로 용이하게 병렬 프로그램을 구동할 수 있는 구조를 갖는다. (그림 2)와 같이 JRS는 작업 의뢰 시스템 데몬, 사용자 프로그램, 공유메모리, 통신 인터페이스 부분으로 구성되며 그 주요 부분인 사용자 프로그램과 작업 의뢰 시스템 데몬의 내용은 아래와 같다.

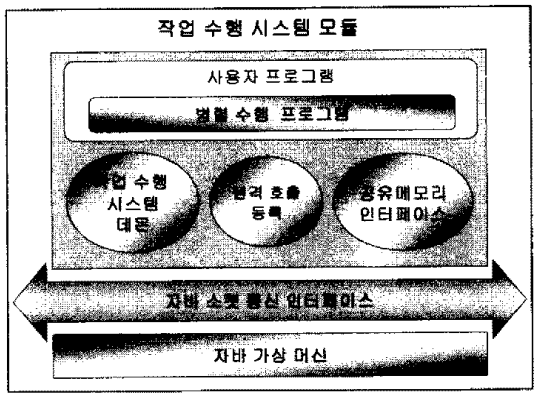


(그림 2) 작업 의뢰 시스템의 구성

- *사용자 프로그램* : 사용자 프로그램은 사용자가 GICE 환경에서 병렬로 수행할 목적으로 작성한 프로그램으로서 자바에서 지원하는 다중스레딩을 통해서 현재 참여하고 있는 JPS의 수나 어플리케이션의 병렬성을 고려하여 설계된 스레드들로 구성된다. 프로그램의 수행시 생성되는 스레드들은 각각 JPS로 매핑되며 각 스레드는 원격 메소드 호출을 통해 사용자 프로그램 내에서 JPS 사용시 프로그램을 호출함으로써 병렬 수행이 이루어진다.
- *작업 의뢰 시스템 데몬* : RMS와 통신을 통해 현재 사용 가능한 작업 시스템에 대한 정보를 수집하는 기능을 수행하며, 이는 가변적인 네트워크 상에서 JPS의 상태 변화에 따른 수행의 안정성을 보장하는 역할을 담당한다. 또한 원격 메소드 호출 메카니즘을 이용해서 공유 메모리의 영역을 할당하고 등록하는 역할의 공유메모리에 JPS들이 접근할 수 있는 인터페이스를 제공해준다. 그리고 작업 의뢰 시스템 데몬은 내부 통신 인터페이스를 통해 JPS 들과의 동기화를 맞추어 주거나 JPS내의 사용자 프로그램(즉, 병렬 수행 코드 블록)을 전송하는 역할을 한다.

2.1.2 작업 수행 시스템(JPS)

JPS는 JRS에게 자신의 자원을 사용할 수 있도록 제공하는 시스템이다. JPS는 자바 가상 머신 위에서 작동하며 (그림 3)과 같이 사용자 프로그램, 작업 수행 시스템 데몬, 원격 호출 등록부, 공유메모리 인터페이스, 통신인터페이스 부분으로 구성되며 그 주요 내용은 아래와 같다.



(그림 3) 작업 수행 시스템의 구성

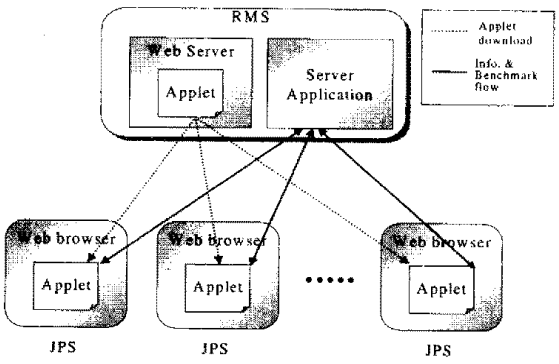
- *작업 수행 시스템 데몬* : JPS 상에 실행상태로 상주하며 JPS의 상태에 대한 정보를 유지한다. 이러한 정보를 기반으로 하여 JPS는 JRS와의 통신을 통하여 자원 사용허가를 결정한다. 자원 사용허가를 받은 JRS의 요구에 따라 코드를 전송 받아 수행시키며 다른 JPS와의 통신을 통하여 동기화도 유지시킨다.
- *원격 호출 등록부* : 원격 메소드 호출을 수행하기 위해 JPS에서 생성된 원격 객체(remote object)를 등록시키는 부분으로, 원격 호출 등록(RMI-registry)을 수행함으로서 JRS가 수행을 요구하는 사용자 프로그램을 등록시키는 일을 담당한다.
- *사용자 프로그램* : JRS로부터 수행을 의뢰 받은 병렬 수행 자바 코드로서 JRS의 사용자 프로그램이 시작할 때 등록된 JPS에게 전송된다. JRS가 병렬

수행으로 명시된 프로그램의 한 부분을 수행하면 JPS의 사용자 프로그램 코드블록이 시작되게 된다. 작업 수행에 필요한 데이터는 JRS가 제공하는 공유 메모리 인터페이스를 이용하여 공유메모리에 접근을 통하여 얻으며 연산의 결과도 공유 메모리에 기록하게 된다.

2.1.3 자원 중계 관리 시스템(RMS)

RMS는 인터넷 상의 물리적으로 분산되어 있는 다양한 JPS들을 모아서 하나의 병렬 플랫폼으로 사용할 수 있도록 하기 위한 참여시스템의 등록 기능 및 등록된 참여 시스템들 간의 물리적 거리 및 네트워크 부하, 이기종성 그리고 수시로 변하는 작업량 등을 고려한 참여시스템 관리 기능을 갖는다.

RMS는 (그림 4)와 같이 애플릿을 전송할 수 있는 웹 서버와 GICE에 참여하고자 하는 JPS 및 JRS와의 통신이 가능한 서버 어플리케이션으로 구성된다. 주어진 시스템을 GICE에 참여시키기를 원하는 사용자는 RMS의 홈페이지에 가서 자신의 IP(Internet Protocol) 주소와 참여 시스템의 기종을 등록하고 JPS 데몬만을 설치하면 된다. 이때 RMS는 각 등록 시스템의 현재 작업량 뿐만 아니라, 네트워크의 상황 및 거리를 고려해서 JRS에게 정보를 제공하기 위하여 간단한 마이크로 벤치마크를 이용하여 참여하는 JPS의 작업량과 네트워크의 상황 및 거리를 측정한다. 즉, JPS가 자신의 자원을 RMS에 등록할 때 마이크로 벤치마크를 내장한 자바 애플릿을 다운로드 받은 후 이를 이용해서 간단한 네트워크의 상황과 작업량 및 거리를 체크해 볼 수 있다.

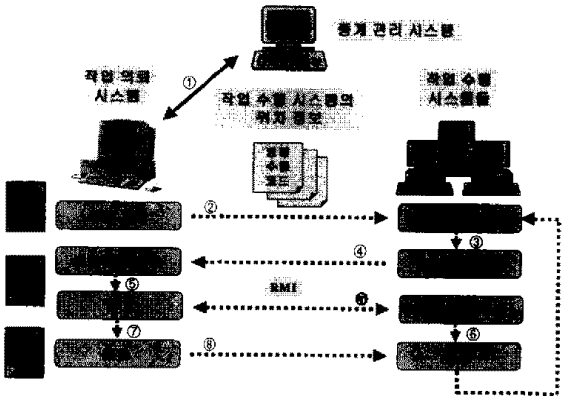


(그림 4) 자원 중계 관리 시스템의 구성

그러므로 혼합 이기종 시스템들로 구성된 인터넷에서의 분산 병렬 처리를 효율적으로 지원하기 위하여 RMS는 JPS들의 성능과 부하균형을 위한 마이크로 벤치마크의 결과, 즉 IP주소, 기종, 네트워크의 상황, 현재 작업량의 정보를 테이블로 구성한 후 대상 JPS들을 사용의 최적성을 기준으로 정렬하여 갖고 있게 된다. 현재 등록되어 있는 JPS의 할당은 정적 스케줄링 기법을 사용한다. 즉 임의의 JRS가 JPS를 요청 할 경우에 RMS는 테이블의 최상위에 있는 JPS의 시스템 정보를 그 JRS에게 제공하게 된다. 그러므로 GICE에서는 JRS들이 직접 컴퓨팅 자원을 찾지 않고 단지 인터넷 상에서 유휴시간을 보내고 있는 시스템들의 주소를 관리하고 있는 RMS을 이용함으로 인해서 체계적이고 확장성이 높은 시스템을 구축할 수가 있게 된다.

2.2 GICE 동작 모델

GICE 상에서의 프로그램 동작은 JRS와 JPS들 간의 상호작용에 의해 각각의 시스템의 상태 변화로 나타난다. GICE의 동작은 크게 각 시스템별로 준비 단계, 수행 단계, 종료 단계로 구분된다. GICE 상에서의 프로그램 동작 모델의 상태 변화와 동작 순서는 (그림 5)에 보여지며 그 내용은 아래와 같다.



(그림 5) GICE의 수행 모델

주어진 응용 프로그램의 수행을 위해 GICE에 등록된 사용자 JPS는 JRS로서 역할을 하게 된다. JRS는 RMS에게 현재 사용 가능한 JPS의 위치에 대한 정보를 요청한다. RMS는 JRS에게 GICE에 등록된 JPS들 중에서 요청한 개수의 JPS들의 주소와 상태를 알려준다. JRS는 전송 받은 JPS들과 연결한다(①). JRS는 각각의 JPS에게 자원 사용에 대한 허가를 신청한다. JPS는 주어진 설정에 준하여 JRS에게 사용을 허가한다.

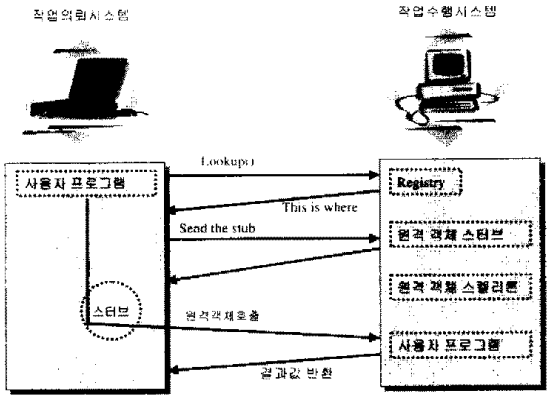
JRS는 사용허가를 얻은 각 JPS들에게 수행을 요청할 사용자 프로그램 코드블록들을 분할하여 전송한다(②). 각 JPS는 전송 받은 사용자 프로그램 코드블록으로부터 원격 호출을 위한 등록과정을 마친 후, JRS로부터 원격 호출을 기다리는 대기 상태가 된다(③). 각 JPS로부터 대기 상태 응답을 받은 후, JRS는 사용자 프로그램을 시작하여(④) 수행 상태가 된다. JRS는 수행 중 병렬코드 부분에 대해서는 작업수행 시스템들에게 원격 호출을 요청하고 결과를 기다리는 대기 상태가 된다(⑤). JPS들은 JRS로부터의 수행을 위한 원격 호출 요청에 따라 전송된 사용자 프로그램 코드블록을 각각 수행하는 상태가 된다(●). 원격 호출에 의한 작업 수행을 끝낸 JPS는 수행 결과를 JRS로 돌려준 후, 원격 수행이 끝났음을 알리는 종료 상태가 된다(⑥). JRS는 수행 중인 모든 JPS으로부터 결과가 돌아오면, 이를 취합해 최종 결과를 산출한다(⑦). 끝으로, 모든 수행이 완료되면 JRS는 각 JPS에게 사용이 끝났음을 알려주고, JPS들은 사용허가 상태에서 준비 상태로 전환하고, JRS 자신도 JRS 상태에서 JPS 준비 상태로 전환한다(⑧).

3. GICE의 구현

3.1 병렬 프로그램 수행 메카니즘

GICE에서는 자바의 다중스레딩 기능을 기반으로 하여 각각의 스레드가 서로 다른 JPS에 할당되어 수행될 수 있도록 한다. 그 구현은 원격에 있는 객체를 지역 객체처럼 호출할 수 있는 원격 메소드 호출 메카니즘을 이용하고 있어 사용자는 전체 프로그램에서 병렬로 수행 가능한 부분을 다중스레딩과 원격 메소드 호출 메카니즘을 사용하여 간편하게 작성할 수 있게 된다.

자바 원격 메소드 호출 메카니즘은 네트워크을 통해 객체를 전송할 수 있을 뿐만 아니라, 다른 머신 상에서 메소드를 호출하기 위해 객체들을 옮길 필요없이 다른 컴퓨터상의 객체들에 대한 메소드를 호출할 수 있다는 장점을 제공한다. GICE에서 병렬 응용프로그램을 작성하는 프로그래머는 실제 작업 수행 시스템에서 수행되는 객체와 관련하여 필요한 모든 정보를 JRS 측의 인터페이스 부분과 JPS 측의 구현 부분으로 분리해야 한다. GICE에서 사용된 원격 메소드 호출 메카니즘을 살펴보면 (그림 6)과 같다. JRS 사용자는 사용자 프로그램에서 작업 수행 시스템 내에 할당된 병렬 수행 코드에 대한 메소드를 호출함으로써 동작하게 되는데, JRS 코드가 JPS상의 원격 메소드를 호출하기를 원할 때 JRS에 존재하는 스터브(Stub)라 불리는 대표 객체 내에 캡슐화되어 있는 일반 자바 메소드를 호출하게 된다. 스터브는 JRS 상에 존재하며, 원격 메소드 내에서 사용되는 전달인자를 취하고 이 전달인자를 네트워크을 통해 전송하기 적합한 형태로 인코딩하는 과정 즉, 마샬링(marshalling)을 거친 후, JPS에게 전달한다. 패킷 내에 포함되어 있는 정보를 인지하고 이 원격 메소드를 실행하는 실제 객체에게 전달하는 객체인 JPS 내에 존재하는 골격(skeleton)객체는 JRS로부터 전달된 전달인자를 언마샬링(unmarshalling)하고 서버에 존재하는 실제 객체에서 원하는 메소드를 호출하고 다시 결과 값을 마샬링해서 JRS의 스터브에게 돌려보낸다. 실제 원격 메소드 호출은 이러한 복잡한 과정을 거쳐서 수행되지만 자바는 원격 메소드 호출과정을 사용자에게 투명하게 제공하고 있다.



(그림 6) 원격 메소드 호출 메카니즘

3.2 공유메모리 메카니즘

공유메모리는 모든 JPS들이 공유하여 접근할 수 있는 가상 어드레스 공간이다. GICE에서 제공하고 있는 공유메모리는 사용자 프로그램이 위치하는 JRS에 할당되며 데이터 분할 구동시 단일 공유 메모리를 각 JPS에 할당할 수 있게 된다. 모든 JPS들은 필요한 데이터를 얻기 위해 동시에 JRS가 제공하는 공유메모리 영역에 접근할 수 있지만, 계산 결과를 공유 메모리에 쓰기 위해서는 동기화를 맞추어야만 한다. 제안된 공유메모리는 원격 메소드 호출 메카니즘을 이용해서 구현되었기 때문에 JRS 데몬은 병렬 수행을 시작하기 전에 공유메모리를 등록하는 작업을 수행하고, JPS에

서 계산할 부분 데이터들을 공유 메모리에 초기화한다. 작업 수행시 각 JPS는 JRS내의 공유 메모리를 접근하여 필요한 데이터를 얻어 연산을 수행한 후에 결과 값을 공유 메모리에 쓰게 된다.

3.3 GICE에서의 병렬 프로그램의 예

GICE환경 하에서 사용자는 프로그램을 병렬로 수행할 부분과 순차적으로 수행할 부분으로 구분하여 프로그램을 작성한다. 데이터의 병렬성을 이용하는 일반적인 예로서 복합 행렬 곱셈의 예는 여러 개의 $n \times n$ 행렬 곱셈을 요구하는 응용 프로그램으로 3차원 그래픽스의 응용에서 사용되는 예이다. 행렬 곱셈부분을 병렬로 작성한 프로그램의 예로서 <표 1>과 <표 2>는 복합 행렬 곱셈 어플리케이션의 일반 병렬 컴퓨터의 호스트 프로그램에 해당하는 JRS 사용자 프로그램과 일반 병렬 컴퓨터의 노드 프로그램에 해당하는 JPS 사용자 프로그램 코드블록을 보여준다. 즉, <표 1>은 참여한 JPS의 수 만큼 스레드를 생성시켜 각 JPS에게 매핑시키는 작업을 수행하는 코드이고 <표 2>는 각 JPS에서 병렬로 수행되는 행렬 곱셈 코드의 일부를 나타낸다. 다음은 복합 행렬 곱셈 프로그램의 실제 동작 과정을 살펴본다.

- **준비 단계** : 사용자는 자신의 시스템을 JRS로 사용하여 프로그램 수행을 시작한다. JRS는 먼저 작업 선행 처리를 수행하여 RMS로부터 GICE에 등록된 JPS의 위치와 사양에 관한 정보를 넘겨받고, 공유 메모리를 위한 원격 호출 등록부를 수행한 후, 등록된 JPS들로 두 개의 행렬을 곱하는 부분의 코드들을 전송한 후, 작업 준비 상태가 된다. JPS는 원격 메소드 호출을 위한 원격 호출 등록부를 수행한 후 작업 수행을 위한 작업 준비 상태가 되었음을 JRS에 알려준다.
- **수행 단계** : JPS들로부터 응답을 받은 후 JRS는 프로그램을 수행하며 JRS는 프로그램 내에 명시된 개수만큼의 스레드를 생성한다. 생성된 스레드들은 병렬적으로 JPS내의 행렬 곱셈 부분들의 수행을 의뢰한 후, JPS들로부터의 결과를 기다리는 결과 대기 상태가 된다. JPS는 요구된 두 행렬의 곱셈을 수행한 후, 결과 행렬을 JPS의 공유 메모리에 저장하게 된다. 이후, JPS는 다시 작업 준비 생태가 된다. 결과를 돌려 받은 JRS는 처리할 데이터가 있으면 다시 작업을 수행을 하고, 처리할 데이터가 더 이상 없으면, 작업이 끝났음을 JPS들에게 알려준다.
- **종료 단계** : JRS의 신호에 따라 JPS는 원격 호출 등록부를 제거하고, 작업 대기 상태가 된다. JRS는 모든 JPS들로부터의 결과를 모아 최종 결과를 사용자에게 제시한 후, 작업 대기 상태의 작업 준비 시스템이 된다.

〈표 1〉 작업 의뢰 시스템 사용자 프로그램

```
for( int Index=0; Index<nServer; Index++ ) {
        mmth[Index].setMat( Index, Size );
        mmth[Index].start();
}
while(true) {
        int count=0;
        for( int i=0;i<nServer;i++ ){
        if( mmth[i].isAlive() == true )
                        break;
        else
                        count++;
        }
        if(count==nServer)
                        break;
}
```

〈표 2〉 작업 수행 시스템 사용자 프로그램

```
Size = size * size;
x=sm.SM_GetPartMem( x_Start, Size );
y=sm.SM_GetPartMem( y_Start, Size );
z=new int [Size];
for( int i=0; i<size; i++ ) {
      for( int j=0; j<size; j++ ) {
            int temp = 0;
            for(int k=0; k<size; k++ )
                temp+=x[i*size+k]*y[k*size+j];
            z[i*size+j] = temp;
    }
}
sm.SM_SetPartMem(z, z_Start, Size);
```

4. 성능 평가

본 절에서는 GICE의 개념, 구현성, 그리고 성능을 분석하기 위한 시험시스템 환경의 구축 내용과 시험시스템 상에서의 벤치마크를 사용한 성능 분석 결과를 기술한다.

4.1 GICE의 구현 환경

GICE 시험시스템 구축을 위하여 인터넷에 연결된

본 연구실내의 컴퓨터 시스템들을 사용하여 구성되었다. 각 시스템들은 10Mbps 이더넷 어댑터를 장착하고 있으며 PC들은 모두 MS윈도우즈98 운영체제를 탑재하고 있으며 자바 프로그램 작성 및 수행은 JDK1.1.7을 사용하였다.

실험을 위하여 <표 3>과 <표 4>와 같이 이기종 시스템 환경과 동일 기종 두가지 GICE 시험시스템 환경을 구축하였다. 각 GICE 시험시스템은 9개의 컴퓨터로 구성되어 M1~M8은 JPS로 사용되며 M9는 JRS의 기능을 담당한다. 즉, 벤치마크에서 순차부분의 수행과 병렬로 수행되는 부분으로 나누어 참여한 JPS에게 배분해주는 작업은 JRS M9에서 수행되며, 병렬로 수행되는 부분들은 JPS M1~M8로 나뉘어서 수행된다. 물론 M9도 JRS뿐만 아니라 동시에 JPS의 기능도 수행할 수 있으나, 본 실험에서는 JPS들 간의 작업 수행의 불균형을 초래할 수 있으므로, M9는 JRS 전용으로 사용한다.

병렬 자바 벤치마크들에 대하여 GICE 환경에서 직접 수행하였다. 실험은 혼합 이기종 시스템 환경과 동일 기종 시스템 환경에서 제안된 GICE가 데이터의 병렬성을 이용하는 응용프로그램들에 대해 분산·병렬 컴퓨팅에 있어 선형적인 성능향상을 얻을 수 있는가를 알아보기 위해서 수행되었다. 모든 실험은 각 시스템의 일반적인 부하를 가진 환경 하에서 수행되었으며 각 실험에서 JPS의 수를 변경하며 벤치마크를 수행하였다. 즉, <표 3>과 <표 4>의 순서대로 M1에서부터 M8을 추가하며, JPS의 수가 1대, 2대, 4대, 8대인 경우에 대하여 동일한 벤치마크를 수행하였다.

4.2 병렬 벤치마크

GICE의 성능 평가 실험은 여섯 가지 벤치마크에 대하여 수행하였다. 실험을 위해 사용된 병렬 벤치마크로는 크게 복합 행렬 곱셈(Complex Matrix Multiplication)과 소수 계산(Prime Number Detection), 머지 소트(Merge Insertion Sort)를 사용하였으며, 이중 복합 행렬 곱셈 벤치마크는 행렬의 크기에 따라 각각 MatMult_A(100×100), MatMult_B(200×200), MatMult_C(400×400), MatMult_D(800×800)로 나누어서 실험을 수행하였다.

복합 행렬 곱셈 벤치마크는 3차원 그래픽 응용에서 사용되는 것으로 입면체를 이동하거나 회전시키기 위해 다각형(polygon) 단위들로 나누어 다각형을 이루는 점 변환(vertex transformation) 연산에 사용된다. 이는 입면체를 구성하는 많은 수의 점들에 대해 동일한 연산을 해 주는 것으로, JRS로부터 연산할 데이터를 읽거나, 연산이 끝난 데이터를 리턴하는 통신비용이 많이 들지만 데이터 병렬적 수행이 가능한 특징을 가지고 있다.

소수 찾기 벤치마크는 주어진 커다란 수가 소수인지 아닌지를 판별해내는 연산으로, 커다란 소수는 암호화 및 복호화 키의 중요 요소로 작용한다. 소수 계산은 범위를 조각으로 나누어서 각각의 범위 조각에 대해 소수 판정 알고리즘을 병렬로 수행함으로써 가능하다. 소수 계산은 연산 집중적인 병렬 수행이므로 전체 수행을 위한 통신 시간의 비율이 낮아서 GICE와 같은 네트워크 기반의 환경에 알맞은 장점을 가진다.

머지 소트 벤치마크는 병렬 컴퓨팅에서 가장 일반적이고 중요한 어플리케이션으로, N개의 데이터 시퀀스

〈표 3〉 이기종 환경 구성 시스템

	작업 수행 시스템								작업 의뢰 시스템
시스템	M1	M2	M3	M4	M5	M6	M7	M8	M9
CPU(MHz)	Ultra Sparc II	P2-233	Ultra Sparc I	P2-266	P2-233	Pp-200	Axil 220	K6-233	P2-300
RAM(MB)	256	64	64	64	64	64	64	64	64

〈표 4〉 동일기종 환경 시스템 사양

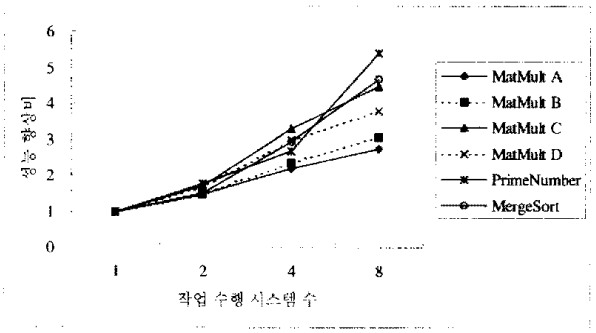
	작업 수행 시스템								작업 의뢰 시스템
시스템	M1	M2	M3	M4	M5	M6	M7	M8	M9
CPU(MHz)	P2-266	Pp-200	P2-233	P2-300	P2-266	Pp-200	K6-200	K6-230	P2-300
RAM(MB)	64	64	64	128	64	64	64	64	64

가 주어졌을 때 동일한 데이터를 포함하는 하나의 정렬된 시퀀스를 생성하는 어플리케이션이다. 실험에서는 잘 알려진 병렬 머지 소트 알고리즘을 사용하여 분산된 데이터를 정렬한다. 머지 소트는 복합 행렬 곱셈보다 컴퓨팅 시간에 비해 통신에 소요되는 시간이 적기 때문에 통신이 성능에 미치는 영향을 알아보기 위하여 선택한 벤치마크이다.

4.3 실험 결과 및 분석

본 실험에서 사용하는 여섯 가지 벤치마크 중에서 공유메모리 참조를 필요로 하는 복합 행렬 곱셈과 머지 소트 벤치마크는 이기종 환경에서 수행되었고, 공유메모리 참조가 필요하지 않은 소수 찾기 벤치마크는 동일기종 환경에서 수행되었다. GICE에서 각 벤치마크에 대하여 JPS의 수에 따른 전체 수행 시간과 JRS의 요청에 따른 JPS에서의 수행 시간 그리고 GICE에서의 시간 중 JRS와 JPS간의 파라미터와 결과 값 전송 및 공유 메모리 참조를 위한 통신 시간은 수행 트레이스를 사용하여 각각 산출하였다.

(그림 7)은 여섯 가지 벤치마크에 대하여 JRS 1대와 JPS 1대로 구성된 GICE 환경에서 각 벤치마크의 성능에 대하여 JPS 개수의 증가에 따른 성능향상비(speedup)를 보여준다(<표 5> 참조). 일반적으로 JPS의 수가 증가함에 따라 병렬로 수행되는 코드부분은



(그림 7) Speedup 비교

증가하게 되므로 JPS의 개수가 증가함에 따라 원거리 수행을 위한 오버헤드의 증가에도 불구하고 JPS 개수의 증가는 전체 성능의 향상을 가져오며 이는 일반적인 병렬컴퓨터의 성능향상비와 비슷함을 알 수 있다. 하지만 4대일 경우와 8대일 경우 성능의 증가량이 이상적인 병렬 분산 컴퓨팅의 결과 (즉, 4대일 경우 4배, 8대일 경우 8배)와는 차이를 보이는데, 이는 전체 수행 시간은 컴퓨팅 시간이 가장 오래 걸리는 JPS의 수행 시간에 의하여 결정되며 특히 참여하는 JPS의 성능이 서로 다르고 수행 당시의 작업량이 다르기 때문이라고 분석된다. 여섯 가지 벤치마크 중에 소수 찾기, 머지 소트, 그리고 복합 행렬 곱셈 순으로 더 좋은 성능 향상비를 보여주고 있는데 이는 전체 수행 시간 중에서 통신 시간이 차지하는 비율이 작은 벤치마크일수록 더 좋은 성능향상을 얻을 수 있다는 것을 보여준다.

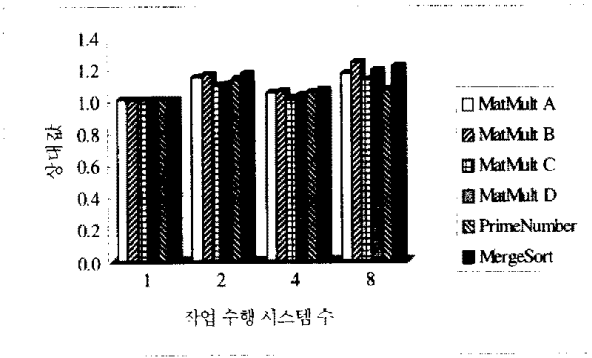
(그림 8)은 GICE환경 하에서 프로그램 수행에 소요된 전체 시간 중에서 각 JPS가 벤치마크의 병렬 수행 부분의 연산에 소요된 시간 즉 순수하게 연산에 소요된 절대시간 만을 모두 누적한 값을 나타낸 것이다. 이 시간들은 GICE에 참여한 각각의 JPS들이 수행에 참여한 전체 시간 중에서 연산 즉, JPS의 수가 4인 경우 4대의 JPS가 각각 연산을 수행하기 위해 소요된 시간을 모두 더한 값이다. 모든 경우에 대해서 각각의 벤치마크는 동일한 작업량을 가지기 때문에 작업 수행 시간의 누적은 JPS의 수가 1대일 경우에 소요된 시간과 거의 유사한 값을 가져야 한다. 그러나 (그림 8)에서 보이는 결과는 JPS의 수가 증가함에 따라 약간의 차이가 생기는데 이는 각 JPS의 현재 부하량과 각각의 성능차이에 기인하는 것으로 분석된다.

(그림 9)는 공유 메모리 참조를 많이 하는 복합 행렬 곱셈벤치마크와 머지 소트 벤치마크에 대하여 연산 시간을 제외한 통신 등에 소요되는 통신시간의 증가 비율을 보여준다. 통신시간은 각 벤치마크들에 대하여 JRS와 JPS간의 원거리 수행을 위한 파라미터와 리턴

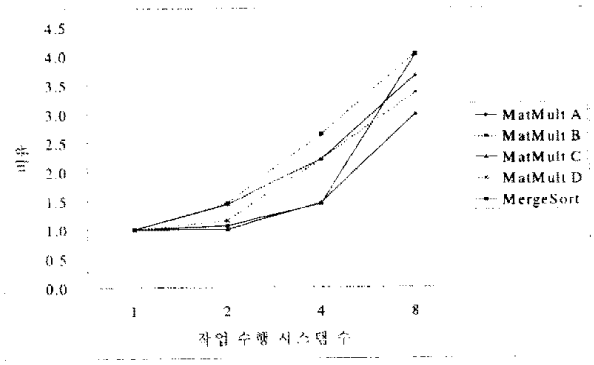
〈표 5〉 GICE 상에서의 전체수행시간(초)

벤치마크 \ 작업수행시스템 수	1	2	4	8
복합행렬곱셈A(100)	5.16(1.00)	3.46(0.67)	2.36(0.46)	1.87(0.36)
복합행렬곱셈B(200)	29.44(1.00)	19.89(0.68)	12.53(0.43)	9.55(0.32)
복합행렬곱셈C(400)	243.92(1.00)	142.09(0.58)	73.33(0.30)	54.43(0.22)
복합행렬곱셈D(800)	2127.21(1.00)	1271.14(0.60)	715.46(0.34)	561.07(0.26)
소수 찾기	1120.37(1.00)	637.41(0.57)	414.36(0.37)	206.96(0.18)
머지소트	170.49(1.00)	112.54(0.66)	57.94(0.34)	36.25(0.21)

값 전송에 걸리는 시간 및 원격리 수행을 위한 연결과정에 소요된 시간, 그리고 공유 메모리 참조시간과 같이 통신에 소요되는 시간 등으로 구성되며 각 JPS의 수행 트레이스를 통하여 산출하였다. 일반적으로 JPS가 1대인 경우와 비교하여 JPS의 수가 증가함에 따라 통신시간은 증가함을 볼 수 있다. 이러한 통신시간의 증가는 주로 공유메모리로의 통신 집중이 가장 큰 요인으로 작용하고 있음을 분석결과 알수 있었다.

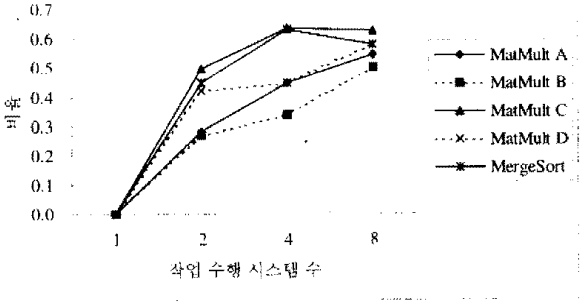


(그림 8) 수행 시간의 누적



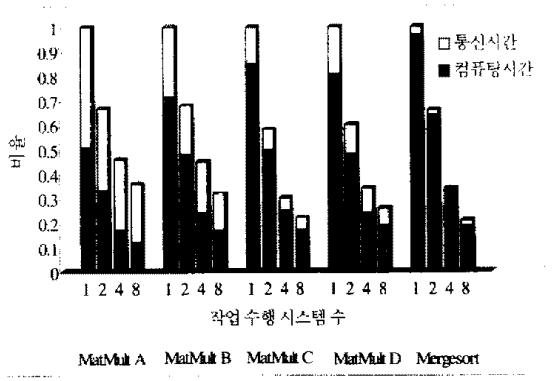
(그림 9) 통신 오버헤드 비율

전체 통신시간은 JPS의 수가 증가함에 따라 JPS마다 원격 호출을 위한 연결과정이 필요하게 되므로 증가하게 된다. 그러나, 여러 JPS들이 동시에 수행되는 상황에서는 주어진 JPS의 통신은다른 JPS의 연산과 중첩될 수 있다. (그림 10)은 전체 통신시간에서 다른 JPS의 연산시간과 중첩이되는 통신시간의 비율을 보여주고 있다. 일반적으로 그 비율은 JPS의 수가 증가함에 따라서 함께 증가함을 볼 수 있다. 그러나 MatMult C와 머지 소트는 JPS의 수가 4대에서 8대로 증가했을 경우에 통신시간이 연산시간에 숨겨지는 비율이 낮아지는 것을 볼 수 있는데 (그림 10)에서 볼 수 있듯이 통신 오버헤드가 급격히 증가되면서 통신에 소요되는 시간이 증가하기 때문이다.



(그림 10) 통신시간 중첩률

통신시간이 성능에 미치는 영향을 나타내기 위하여 (그림 11)은 전체 수행시간에서 통신시간과 연산시간의 구성 비율을 보여준다. 여기서 통신이 빈번한 복합행렬 곱셈의 경우에는 통신시간이 성능 향상에 미치는 요인이 크다는 것을 볼 수 있다.



(그림 11) 통신시간이 전체 성능에 미치는 영향

5. 결 론

인터넷 컴퓨팅 환경은 인터넷에 연결되어 있는 PC, 워크스테이션, 기타 시스템들을 통합하여 하나의 컴퓨팅 자원으로 활용함으로써 고성능을 필요로 하는 어플리케이션의 수행을 위한 비용 면에서 저렴하고 성능 면에서 우수한 컴퓨팅 플랫폼으로 사용될 수 있는 가능성을 제시한다. 본 논문에서 제안하는 GICE는 효율적인 인터넷 컴퓨팅 환경의 구축 기법을 제시한 후 인터넷 환경 구축의 개연성, 복잡성, 그리고 성능에 대한 평가 결과를 제시하여 인터넷 컴퓨팅 환경의 가능성을

분재식으로 실증하고 있다. GICE는 스레드 및 원격객체 호출 그리고 소켓 통신 기능을 갖는 자바를 기반으로 한 프로그래밍 개발 및 수행 환경을 제공하며 공유메모리 기반의 병렬 프로그래밍 모델을 지원한다. 성능 면에서 대규모 데이터의 병렬성과 분할 수행되는 각 태스크들 간에 낮은 상관성이 제공되는 응용 프로그램들에 대하여 JPS의 수가 증가할수록 거의 선형적인 성능 향상을 얻을 수 있었다. 향후 보다 향상된 성능을 위하여 현재의 공유메모리 구조를 캐쉬일관성을 지원하는 분산 공유메모리로의 확장, 보다 향상된 신뢰성을 위하여 시스템을 구성하는 각 태스크의 최적 프로세서로의 분배 기법과 프로그램 수행 중 부하 불균형과 시스템 오류를 동적으로 감지하여 태스크를 이주 및 재 수행시킬 수 있는 관리기법에 대한 연구를 수행할 예정이다.

참 고 문 헌

[1] Thomas E. Anderson, David E. Culler, David A. Patterson, and the Now team, "A Case for NOW," IEEE Micro, Vol.15, No.1, pp.54-64, Feb. 1995.

[2] N. Carri and D.Gelernter, "Linda in context," Communication of the ACM, Vol.32, No.4, pp.444-458, April 1989.

[3] V.S.Sunderam, "PVM : A Framework for Parallel Distributed Computing," Technical Report ORNL/TM-11375, Dept. of Math and Computer Science, Emory University, Atlanta, GA, USA, February 1990.

[4] William Gropp, Ewing Lusk, and Anthony Skjellum, "Using MPI : Portable Parallel Programming with the Message-Passing Interface," The MIT Press, Cambridge, Massachusetts, 1994.

[5] A. S. Grimshaw, Wm. A. Wulf, and the Legion team, "The Legion Vision of a Worldwide Virtual Computer," Communications of the ACM, 40(1), January 1997.

[6] A. Ferrari, "JPVM : Network Parallel Computing in Java," in Proc. ACM 1998 Workshop on Java for High-Performance Network Computing, Palo Alto, 1998.

[7] B. Christiansen, P. Cappello, M. Ionescu, "Javelin : Internet-Based Parallel Computing Using Java," 1997 ACM Workshop on Java for Science and Engineering Computation, pp.30-40, June 1997.

[8] T. Brecht, H. Sandhu, M. Shan, and J. Talbot, "ParaWeb : Towards World-Wide Supercomputing," 7th ACM SIGOPS European Workshop, pp.181-188, Sep. 1996.

[9] A. Baratloo, M. Karaul, Z. Kedem, and P. Wychkoff, "Charlotte : Metacomputing on the Web," 9th International Conference on Parallel and Distributed Computing Systems, pp.151-159, Sep. 1996.

[10] J. Eric Baldeschwieler, Robert D. Glumofe, and Eric A. Brewer, "ATLAS : An Infrastructure for Global Computing," 7th ACM SIGOPS European Workshop, pp.160-167, Sep. 1996.

[11] Albert Alexandrov, Maximilian Ibel, Klaus E. Schauser, and Chris Scheoman, "SuperWeb : Towards a Global Web-Based Parallel Computing Infrastructure," 11th International Parallel Processing Symposium, pp.100-106, Apr. 1997.

[12] K. M. Chandy, B. Dimitrov, and H. Le, "A World-Wide Distributed System Using Java and the Internet," In Proceedings of the Fifth IEEE International Symposium on High Performance Distributed Computing, Syacuse, NY, Aug. 1996.

[13] J. Golsing and H. McGilton, "The Java Language Environment - A Whitepaper," Technical report, Sun Microsystems, October 1995.

[14] C. Amza, A. L. Cox, S. Dwarkadas, "TreadMarks : Shared Memory Computing on Networks of Workstations," IEEE computers, pp.18-28, Vol.29, No.2, Feb. 1996.

김 희 철

e-mail : hckim@biho.taegu.ac.kr

1983년 연세대학교 전자공학과 졸업(공학사)

1983년~1988년 (주)삼성전자 주임연구원

1991년 Univ. of Southern California(Computer Eng. M.S.)

1996년 Univ. of Southern California(Computer Eng. Ph.D.)

1996년~1997년 (주)삼성SDS 수석연구원

1997년~현재 대구대학교 정보통신학부 조교수

관심분야 : 병렬처리, 컴퓨터구조, 컴파일러

신 필 섭

e-mail : ambush@kurene.yonsei.ac.kr

1997년 광운대학교 컴퓨터공학과 졸업(공학사)

1998년~현재 연세대학교 컴퓨터과학과 석사과정

관심분야 : 웹컴퓨팅, 컴퓨터구조, 병렬처리

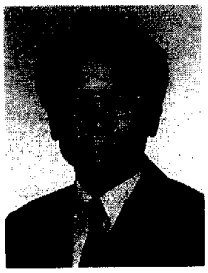
박 영 진

e-mail : yjpark@kachi.com

1997년 대구대학교 정보통신공학부 졸업(공학사)

1998년~현재 대구대학교 정보통신공학과 석사과정

관심분야 : 컴퓨터구조, 병렬처리, 인터넷



이 용 두

e-mail : ydlee@biho.taegu.ac.kr

1975년 한국항공대학교 통신학과 졸업(공학사)

1981년~1982년 (일)동경대학 전자공학과 객원교수

1982년 영남대학교 대학원 전자공학과(공학석사)

1991년~1993년 Univ. of Southern California 교환교수

1995년 한국항공대학교 대학원 전자공학과(공학박사)

1995년~현재 대구대학교 정보통신학부 교수

관심분야 : 컴퓨터구조, 컴퓨터통신, Internet 응용 기술