인터넷 기반의 병렬 컴퓨팅을 위한 사용자 라이브러리 설계 및 성능 분석

신 필 섭[†] · 정 준 목[††] · 맹 혜 선[†] · 홍 원 기[†] · 김 신 덕[†††]

요 약

본 연구에서는 인터넷 기반의 병렬 수행 플랫폼으로 자바 인터넷 컴퓨팅 환경이 설계 및 구현되었다. 자바 인터넷 컴퓨팅 환경은 다양한 시스템에 대한 지원을 위하여 자바를 기반으로 다중 스레딩과 원격 메소드 호출 메카니즘을 사용하여 구현되었다. 자바 인터넷 컴퓨팅 환경의 기본 구성 모델은 클라이언트, 워커, 브로커들의 컴포넌트들로 이루어지며 워커 시스템들간의 통신을 위해 공유 메모리 시스템을 제공한다. 사용자는 자바 인터넷 컴퓨팅 환경에서 제공되는 다양한 사용자 라이브러리를 사용하고, 매스터-슬레이브 프로그래밍 모델을 사용하여 프로그램을 병렬로 수행시킬 수 있다. 기본 구성 모델의 성능적 제약을 극복하기 위해 이를 다중 관리 시스템으로 확장하고 수식적 분석과 벤치마크 프로그램을 직접 수행시키는 실험을 통하여 자바 인터넷 컴퓨팅 환경의 성능이 조사되었다. 결론적으로 자바 인터넷 컴퓨팅 환경은 워커수가 증가함에 따라, 기존의 공유 메모리 다중 프로세서 시스템과 유사한 성능 향상이 나타남을 알 수 있었다. 또한 자원 사용 시간에 대한 분석을 통해, JICE의 프로그래밍 모델과 사용자 병렬화 라이브러리의 효율성을 살펴보았다.

Design and Analysis of User's Libraries for Parallel Computing based on the Internet

Pil-Sup Shin[†] · Chun-Mok Chung[††] · Hye-Seon Maeng[†]
Won-Kee Hong[†] · Shin-Dug Kim[†††]

ABSTRACT

As the Internet and Java technology have been growing up, parallel processing approach to utilize those idle resources connected to the Internet has become quite attractive. In this paper, JICE(Java Internet Computing Environment) was implemented as a parallel computing platform based on the Internet using multithreading and RMI mechanisms provided by Java. The basic model of JICE is constructed as three components, such as a client, a set of workers, and a broker. A worker communicates with other workers via a globally shared memory system. It provides users with master-slave programming model and a collection of library functions. The basic model of JICE is also extended as a multimanaging system. This multimanaging system is evaluated by analysis to show its effectiveness. According to numerical analysis and experiments with several benchmarks, it is shown that the performance of basic model depends on the shared memory reference ratio and user's library is a quite promising.

※ 본 논문은 한국과학재단 특정기초연구지원을 받아 수행된 결과의 일부임(KOSEF 97-0102-0201-3).
† 준 회 원 : 연세대학교 대학원 컴퓨터과학과
†† 정 회 원 : LG정보통신 정보응용기술실 연구원
††† 정 회 원 : 연세대학교 기전공학부 교수
논문접수 : 1999년 7월 27일, 심사완료 : 1999년 10월 11일

1. 서 론

지난 수년동안 인터넷은 전 세계의 컴퓨터들을 하나의 네트워크로 연결하는데 성공하였고, 인터넷은 정보를 제공하는 자원으로서의 기능뿐 아니라 컴퓨팅 자원으로서의 기능이 새로이 정립되고 있다. 이는 분산 병렬 컴퓨팅 환경을 지역 네트워크 환경에서 인터넷상의 수많은 유휴 자원을 활용하기 위한 방법으로써 관심이 높아지고 있다[1-4]. 이러한 시도는 국지적인 컴퓨팅 자원뿐만 아니라 인터넷상에 존재하는 수많은 컴퓨터들을 이용할 수 있다는 점에서 기존의 병렬 분산 컴퓨팅[5,6]이 제약으로 가졌던 컴퓨팅 자원의 확보를 용이하게 해준다. 이러한 연구들은 글로벌 환경에서의 분산 병렬 컴퓨팅 환경을 제공하기 위한 방법으로 대부분 웹 인터페이스, 즉 웹 브라우져와 자바(Java)언어를 사용하고 있다[12]. 이것은 인터넷에 연결되어 있는 거의 모든 시스템들이 웹 브라우져를 포함하고 있어 부가의 인터페이스의 개발 및 설치가 필요 없고 자바언어로 작성된 프로그램은 컴퓨터의 플랫폼에 관계없이 손쉽게 수행될 수 있기 때문이다. 현재, 자바는 그 수행성능 면에서 기존의 프로그래밍 언어보다 떨어진다는 단점을 가지고 있다. 그러나, 자바의 성능을 높이려는 연구들[7]과 자바 프로그램의 효율적 병렬화에 대한 연구들이 진행되고 있고, 또한 네트워크 고속화에 대한 다양한 연구들을 고려하면 다양한 시스템들의 자원을 활용한 인터넷상의 병렬수행 방법의 가능성은 점점 증대되고 있다.

본 연구에서는 인터넷 기반의 병렬 분산처리를 위한 수행 플랫폼으로 JICE(Java Internet Computing Environment)를 설계하고, 이를 보다 효율적으로 이용할 수 있도록 프로그래밍 모델과 사용자 라이브러리를 제공한다. JICE는 컴퓨팅 자원의 제공자나 사용자들 모두에게 쉽게 인터넷 컴퓨팅에 참여할 수 있도록 하기 위한 시스템이다. JICE의 기본 구동 모델은 자바를 기반으로 하는 다중 스레딩(multithreading)기법을 사용하여 구현되었다. JICE의 기본구성은 컴퓨팅 파워를 요구하는 어플리케이션을 인터넷에 연결된 수많은 자원들을 이용해서 수행시키고자 하는 클라이언트(client), 자원의 소유주가 인터넷 컴퓨팅에 참여함으로써 클라이언트가 요청한 어플리케이션을 수행하는 하나 이상의 워커(worker)들, 그리고 클라이언트와 워커간의 자원을 연결시켜주는 브로커(broker)로 이루어진다. JICE는 워커간의 상호 통신을 위해 공유 메모리 시스템[8-10]을 제공하며 시스템의 확장성을 제공하기 위해 다중 관리 시스템 구축 방법이 설계된다. 특히 JICE는 인터넷 컴퓨팅 환경에서 사용자가 인터넷 컴퓨팅 환경의 특성을 고려할 필요 없이 어플리케이션을 작성할 수 있도록 다양한 사용자 클래스 라이브러리를 제공한다. 따라서 사용자는 JICE에서 제공하는 라이브러리를 이용한 매스터-슬레이브 프로그래밍 모델을 사용한다면, 편리하고 간단히 어플리케이션을 작성해 JICE에서 병렬로 수행시킬 수 있다.

본 연구에서는 JICE환경에서 수행되는 어플리케이션의 수식적 분석과 JICE에서의 벤치마크 프로그램을 수행시키는 실험을 통하여 JICE의 성능이 조사되었다. 첫 번째로 JICE의 기본 구성 모델이 제시되고 벤치마크 프로그램을 이용 성능분석을 수행한다. 두 번째로 기본 구성 모델에 확장성을 제공하기 위해 다중 관리 시스템 모델로 확장하고 그 효용성을 분석적 방법에 의해 평가한다. 기본 구성 모델은 실험을 통해 워커수가 증가함에 따라, 기존의 공유 메모리 다중 프로세서 시스템과 유사한 성능 향상이 나타남을 알 수 있었다. 또한 수행 시간 분석을 통해 다중 관리 시스템을 지원하는 JICE는 전역 공유 메모리 쓰기 비율에 크게 영향을 받음을 알 수 있었다. 전역 공유 메모리 쓰기 비율에 따라 다중 관리 시스템을 지원하는 JICE는 기본 구성 모델에 비해 최고 2.5~3배의 성능 향상을 기대할 수 있다. 또한 자원 사용 시간에 대한 분석을 통해, JICE의 프로그래밍 모델과 사용자 병렬화 라이브러리의 구성이 효율적임을 보였다.

2장에서는 본 연구와 관련된 연구들에 대하여 살펴보고, 3장에서는 JICE에 대하여 자세히 살펴본다. 4장에서는 JICE환경에서의 병렬처리를 위한 프로그래밍 모델과 사용자 라이브러리에 대하여 기술한다. 5장에서는 JICE의 성능 및 효율성을 시뮬레이션과 실험을 통해 평가하며, 끝으로 6장에서 결론을 맺는다.

2. 관련 연구

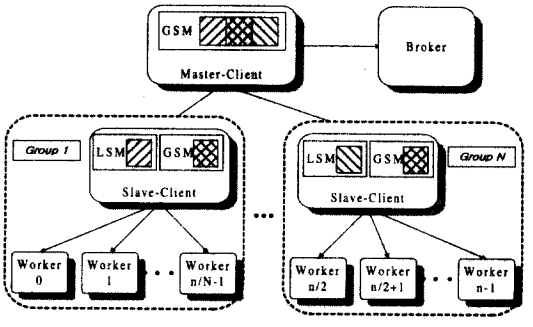
인터넷 환경에서의 자원을 활용한 병렬 수행 환경을 제공한다는 면에서 본 연구와 유사성을 가진 연구들로는 ParaWeb[1], Charlotte[2], ATLAS[3], SuperWeb[4] 등이 있다.

ParaWeb은 자바의 다중 스레딩을 병렬로 사용하여,

자바 프로그래밍 환경과 자바 수행 시스템을 인터넷 기반으로 확장한다. Charlotte은 웹(Web)을 병렬 컴퓨팅의 자원으로 활용하기 위해 자바로만 구현된 분산 공유 메모리 시스템을 제공한다. 그러나, 웹 인터페이스를 사용하기 때문에 Charlotte에서 수행되는 프로그램은 Charlotte 홈페이지의 접속을 통해 자발적으로 자원을 제공하는 도움자들의 수에 따라 그 수행 성능이 가변적이게 된다.

ATLAS는 자바와 Cilk 기술을 접합하여, 다중 스레드로 작성된 프로그램을 네트워크 기반의 자원들을 활용하여 병렬로 수행할 수 있도록 설계되었다. 다중 스레딩은 Cilk에 기반을 두며, 자바에 적합하도록 수정되었다. URL-기반의 파일 시스템과 지역 캐쉬를 사용하여 노드간의 일관성을 유지하며, 작업 취득(work stealing) 방법론을 통해 노드간의 작업량 균형을 지원한다. SuperWeb은 인터넷의 연결된 자원을 활용하기 위해 제안되었으며 안전성을 위한 암호화된 컴퓨팅 방법을 제안한다.

공유메모리 시스템에서 순차적 일관성 모델은 단일 프로세서 시스템에서와 같이 프로그램내의 순서들을 명시할 수 있는 모델을 의미하며, 구현이 용이한 장점을 가진다[8]. JICE의 공유 메모리 시스템은 이러한 순차적 일관성 모델을 지원한다. TreadMarks[11]는 공유 메모리 시스템의 성능 향상을 위한 방법으로 지연 해제 일관성(lazy release consistency)과 다중 쓰기 프로토콜 (protocol)을 사용하며, 사용되는 노드간의 통신만을 수행함으로써 무효화에 소요되는 시간을 줄일 수 있고, 네트워크 환경에서의 통신량과 통신시간의 감소를 통한 성능 향상을 추구한 접근이다.



(그림 1) JICE 구성

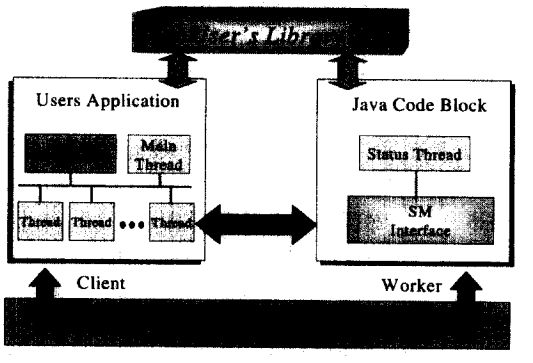
이들 대부분의 연구는 구현에 따른 결과만을 보여주며, 수치적 분석에 의한 성능 향상기법들을 고려하지 않고 있다. 하지만 본 논문에서 제안하는 JICE는 기존의 웹 인터페이스를 기반으로 구동되는 제약을 해결하고, 성능 향상과 확장성 있는 환경을 구성하기 위해 변형된 공유 메모리 시스템을 지원한다. 또한 수행 결과의 수치적 분석을 통해 성능 향상 기법들을 고려하고 있다.

3. 자바 인터넷 컴퓨팅 환경

본 절에서는 자바 인터넷 컴퓨팅 환경(JICE)을 구성하고 있는 컴포넌트와 각 컴포넌트의 기본구성 모델을 제시하고 각각의 기능과 역할을 정의한다.

3.1 자바 인터넷 컴퓨팅 환경의 구성

JICE는 현존하는 자바의 핵심 기술을 이용함으로써 JICE 프로그래머에게 이기종성에 대한 고려 없이 프로그램을 작성 할 수 있게 해주며, 본 환경을 보다 쉽게 사용하게 해주기 위한 프로그래밍 모델과 라이브러리를 제시한다. JICE는 원거리에서 자바 프로그램을 병렬로 수행시킬 수 있는 기본 메카니즘을 제공하고, 본 환경에 참여한 워커들간의 통신을 위해 공유 메모리를 제공함으로써 인터넷상에서 효율적인 분산 병렬처리를 수행할 수 있다. 또한 자바를 사용하기 때문에 프로그래머는 자바 가상 머신을 탑재하고 있는 어떤 시스템에서라도 자신의 프로그램을 작성하고, 디버그하고, 컴파일 할 수 있다. 결과적으로 이기종 병렬 분산 컴퓨팅에서 전형적인 문제였던 운영체제의 호환성 여부와 데이터 표현, 데이터 정렬과 같은 문제들을 프로그래머가 걱정할 필요가 없게 되었다.



(그림 2) 클라이언트와 워커 시스템의 구조

본 연구에서 제시하는 JICE는 기존의 웹 모델을 확장한 인터넷 컴퓨팅 모델이다. 기존의 웹 인터페이스를 이용하는 웹 컴퓨팅 방식에서는 참여하는 시스템이 웹 컴퓨팅을 운영하는 웹 서버에 접속하고 있을 경우에만 그 시스템을 웹 컴퓨팅 자원으로 사용할 수 있었지만, JICE는 한번의 등록만으로 그 자원이 사용될 수 있게 된다. (그림 1)에서 각각의 그룹은 JICE의 기본 구성 모델이며, 시스템의 확장성과 효율적인 통신 메카니즘을 고려해 다중 관리 시스템 모델이 설계된다. JICE의 구성 요소는 인터넷에 존재하는 컴퓨팅 자원을 총괄하는 매스터 클라이언트 시스템과 각 서브 그룹 내에서 그룹 내의 자원을 관리하는 슬래이브 클라이언트, 인터넷 컴퓨팅에 참여함으로써 병렬로 수행될 자바 어플리케이션을 다운로드 받고 자신의 자원을 클라이언트 시스템에게 제공해주는 워커 시스템, 그리고 JICE에 참여한 컴퓨팅 자원을 관리하는 브로커 시스템으로 구성되어 있다. JICE는 단일 그룹의 기본 구성 모델에서 문제가 되고 있는 통신시간을 감소시키는 효율적인 방법으로 (그림 1)에서 보여주는 것처럼 클라이언트 시스템을 매스터 클라이언트와 슬래이브 클라이언트로 분리하였다. 매스터 클라이언트는 사용자 시스템으로서 모든 워커들이 공유할 수 있는 전역 공유 메모리(GSM)만을 제공하고, 슬래이브 클라이언트는 그룹내의 효율적 통신을 위한 지역 공유 매모리(LSM)를 제공한다.

3.2 자바 인터넷 컴퓨팅 플랫폼 구조

JICE의 기본 구성 모델은 인터넷 컴퓨팅에 참여하는 역할에 따라 클라이언트와 워커, 브로커의 단일 그룹으로 구성되어 진다. 브로커를 제외한 클라이언트와 워커 시스템은 상호 어떤 역할이든 할 수 있는 시스템이다. 즉, 한 시스템이 클라이언트의 역할을 함과 동시에 워커의 역할을 할 수 있다. 클라이언트 시스템과 워커 시스템의 기본 구조를 살펴보면 (그림 2)와 같다. 클라이언트 시스템은 JICE환경이 제공하는 사용자 라이브러리로 구현된 사용자 어플리케이션으로 구현된다. 이 사용자 어플리케이션은 (그림 2)에서 보여지는 것처럼 워커들간의 통신과 어플리케이션 데이터를 제공하기 위한 공유 메모리와 JICE에 참여한 워커들을 수행시키고 관리하기 위한 다중 스레드(multithread)로 구성되어진다. 공유 메모리는 효율적인 통신을 위해 지역 공유 메모리(LSM)와 전역 공유 메모리(GSM)로 구분되어 진다. 슬래이브 클라이언트가 제공하는 LSM의 데이터는 매스터 클라이언트가 병렬 수행을 위한 그룹 생성 시, 그룹내의 노드들이 참조하는 GSM 영역의 데이터를 복사하고 각 그룹별로 독립된 영역으로 유지된다. 슬래이브 클라이언트는 그룹 내의 노드와 다른 그룹 노드간의 통신을 위해 LSM과는 별도로 GSM을 지원한다. 각 어플리케이션의 병렬성 및 참여 워커 시스템의 수에 따라 생성된 스레드는 참여 시스템과 일대일로 매핑이 이루어진다. 이는 각 스레드가 하나의 워커의 수행을 담당하게 되고, 이 스레드는 각각 독립적으로 수행이 이루어지기 때문에 병렬 수행이 가능하다.

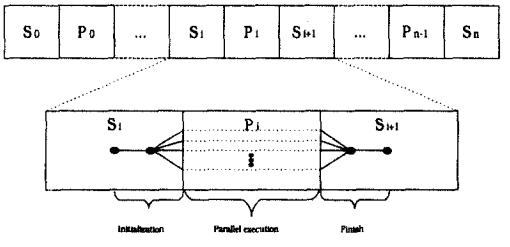
워커 시스템은 병렬 수행 코드 블록과 공유메모리 인터페이스에 의해 구동되어진다. 병렬 수행 코드 블록은 워커가 기본적으로 클라이언트로부터 제공받은 코드블록으로써 클라이언트의 사용자가 작성한 프로그램 중에서 병렬 수행으로 명시된 프로그램의 한 부분으로, JICE에서 제공하는 라이브러리로 작성됨에 따라 클라이언트가 원격에서 수행시킬 수 있다. 상태 스레드는 항상 워커에 수행중인 상태로 대기하여 워커 시스템의 상태를 클라이언트에게 제공한다. 이는 워커 시스템의 상태 변화에 따른 수행의 안정성을 보장하고, JICE환경이 언제나 그 워커를 활용할 수 있다는 측면에서 JICE는 참여한 자원을 보다 적극적으로 활용할 수 있는 환경이 된다. 공유 메모리 인터페이스는 클라이언트가 제공하는 두 가지 종류의 공유메모리를 워커가 접근하기 위한 인터페이스로서 통신인터페이스를 통해서 클라이언트가 제공하는 공유메모리와 통신을 한다.

3.3 자바 인터넷 컴퓨팅 환경을 위한 공유 메모리

공유메모리 환경에서 임의의 두 노드간의 통신은 동일한 영역을 참조하는 변수를 사용한다. 즉 일정한 변수를 공유하여, 하나의 노드가 변수를 갱신하고, 다른 노드가 갱신된 영역을 읽는 방법을 사용한다. 공유메모리 환경에서는 데이터 일관성을 유지하기 위해서 임의의 두 노드가 통신을 수행하는 과정동안에는 공유변수를 갱신 가능한 권한으로 사용하는 모든 노드가 공유 변수의 참조를 중지하여야 한다. 그러므로, 공유메모리 환경에서는 통신의 비율이 높을수록 수행 성능이 낮아지게 되며, 이런 비율은 연결된 노드가 많을수록 통신을 위한 오버헤드가 많이 발생한다. 따라서 노드간의 통신 시간이 오래 걸리는 네트워크 환경의 분

산 병렬 처리 환경에서는 그룹화를 통한 통신 시간의 감소가 필요하게된다. 이를 위해 JICE는 다중 관리 시스템 환경을 지원하고 GSM뿐만 아니라 LSM도 지원을 한다. 이는 그룹 내 노드간의 통신 시간이 그룹 외 노드와의 통신시간보다 작기 때문에 LSM의 참조시간은 GSM보다 작고, 노드들간의 통신은 동일한 병렬 부분을 수행하는 노드간의 통신 비율이 높기 때문에 LSM의 지원은 GSM을 통한 통신 시간보다 더 빠른 통신을 지원할 수 있다. 또한 LSM의 통신에는 그룹 내 노드들만 데이터 일관성을 유지하면 되므로, 그룹들은 데이터 일관성을 위배하지 않고, 병렬로 통신을 수행할 수 있다.

JICE에서 읽기 전용 참조인 경우는 LSM과 GSM의 참조에서 모두 동일하게 수행된다. 현재 쓰기 참조중인 노드가 없는 경우, 모든 노드가 동시에 읽기 전용 참조가 가능함을 의미한다. 쓰기 참조 시에는 LSM과 GSM의 수행 과정이 달라진다. LSM이 갱신되는 동안에는 LSM 영역을 공유하는 그룹 내 노드들의 연산이 중지되지만, LSM 영역을 공유하지 않는 다른 그룹의 노드들은 관계없이 작업을 계속 수행할 수 있다. GSM이 갱신되는 도중에는 GSM을 참조하는 모든 노드들의 수행이 중지된다. LSM은 그룹 내 노드들만이 참조하므로, 하나의 노드가 LSM을 참조할 경우, 그룹내의 노드들만이 대기하게 된다. GSM을 참조하는 경우는 모든 노드가 대기하게 되며, 모든 그룹에서 GSM간의 일관성을 유지하기 위한 작업이 함께 수행된다.



(그림 3) 사용자 프로그램의 구조

4. JICE에서의 병렬 처리

본 절에서는 사용자 프로그램을 JICE에서 병렬로 수행하기 위한 프로그래밍 모델을 정의하며, 사용자 프로그램을 병렬로 수행하기 위해 제공되는 JICE의 사용자 라이브러리에 대하여 기술하고 라이브러리 활용예를 통해 JICE에서 수행되는 사용자 프로그램의 구성 및 수행 모델을 설명한다.

4.1. 병렬성과 프로그래밍 모델

병렬성은 데이터 병렬성(data parallelism)과 수행기능 병렬성(function parallelism)으로 나눌 수 있다. 데이터 병렬성은 SIMD(Single Instruction Multiple Data)계열 컴퓨터나 비공유 메모리형 MIMD(Multiple Instruction Multiple Data)계열 컴퓨터에서 최소의 오버헤드만을 발생시키며 효과적으로 실행시킬 수 있다. 그러나 수행 기능 병렬성과 다중 처리는 기존의 프로그래밍 모델을 지원하는 공유 메모리형 MIMD계열 컴퓨터에서 효과적으로 수행시킬 수 있다. 인터넷을 기반으로 하는 JICE환경은 각 태스크간에 발생하는 통신요구가 적은 경우에 효과적으로 운영될 수 있다. JICE의 구성은 데이터 병렬성은 물론 기능 병렬성에서 효율적인 성능을 제공하는 공유 메모리형 MIMD와 유사한 구조를 제공한다. 이는 무엇보다도 JICE환경이 저속의 네트워크 기반으로 구성되기 때문에, 병렬 수행 부분간의 통신으로 인한 오버헤드가 상당하기 때문이다. 그러므로, 하나의 노드에서 수행에 소요되는 시간이 통신으로 인한 오버헤드보다 작을 때에는 병렬성으로 인한 성능 향상을 기대할 수 없게 된다.

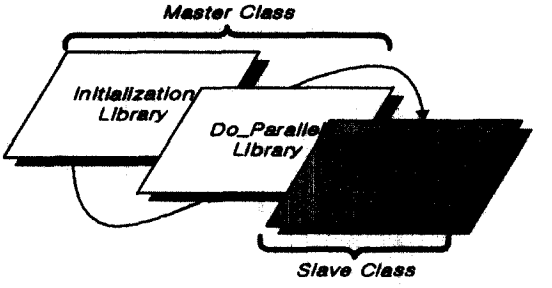
JICE는 그 구조에서 공유 메모리 MIMD와 유사한 구조를 지니는 것만큼, 프로그래밍 모델도 동일한 모델을 제공한다. 프로그램의 병렬성 증가에 따라 수행 시간은 감소하게 된다. 그러나, 병렬성의 증가는 준비시간이나 통신시간, 정리시간의 증가를 초래한다. 여기서, 준비시간과 정리시간은 프로그램의 크기와 상관없이 병렬성 증가에 따라 일정한 시간이 소요되며, 이는 전체 수행 시간의 관점에서는 무시될만하다. 그러나, 통신 시간은 병렬성 증가에 따라 증가되며, 통신시간 동안은 데이터 일관성을 위하여 병렬로 수행하는 각 노드들이 공유 영역에 대한 참조를 중지하고 통신이 완료 될 때까지 기다려야 하기 때문에, 전체 수행 성능과 밀접한 연관을 가진다.

JICE는 인터넷에 연결된 독립된 컴퓨터를 사용하여 프로그램을 병렬로 수행하기 위한 환경이며 프로그램을 분할하여 병렬로 수행하기 위한 구조를 지원해야한다. JICE의 장점 중 하나는 JICE에서의 프로그래밍 환경이 작업을 의뢰하는 시스템에서 제공되는 자바 환경이란 점이다. 즉 JICE는 이기종간의 효율적인 협력 작

업을 위해 자바 언어로 작성된 프로그램을 수행한다. 자바는 시스템들의 이기종성을 지원해줄 뿐 아니라, 자바의 다중 스레딩은 하나의 프로그램을 여러 개의 스레드로 나누어 수행할 수 있도록 지원해준다. JICE에서는 이러한 자바의 다중 스레딩을 여러 개의 워커에 각각 분할하여 수행할 수 있도록 한다. 사용자는 전체 프로그램에서 병렬로 수행 가능한 부분을 다중 스레딩을 사용하여 작성하는 것만으로 JICE에서 효율적으로 수행할 수 있다.

JICE에서 수행되는 사용자 프로그램의 구조는 (그림 3)에서 보는바와 같이 여러 개의 자바 코드 블록들의 연결로 볼 수 있다. 각 블록은 순차 코드 블록과 병렬 코드 블록으로 구분된다. 순차 코드 블록과 병렬 코드 블록은 서로가 교차되어 연속적인 구조를 가진다. 순차 코드 블록은 이전 병렬 수행 결과를 취합하는 처리 부분, 다음 병렬 수행을 위한 준비 작업 부분, 병렬성이 없어 순차적으로 처리해야 하는 부분으로 구성되며, 병렬 코드 블록은 병렬로 수행될 수 있는 부분을 의미한다. $S_iP_iS_{i+1}$로 이루어진 코드 블록 집합을 기본 병렬 모듈이라 정의하면, S_i은 병렬 수행을 위한 준비 작업 과정을 수행하며, P_i는 병렬 수행 과정을, S_{i+1}는 병렬 수행 결과의 취합 과정을 의미한다.

JICE에서 순차 코드 부분은 클라이언트에서 직접 수행이 되며, 병렬 코드 블록은 JICE에 참여한 각 워커들에 할당되어 병렬로 수행된다. 이러한 순차 코드 블록과 병렬 코드 블록의 구분은 사용자가 프로그래밍 시에 명시적으로 지정하여야 하며, 이를 위해 JICE에서는 매스터-슬래이브(Master-slave) 프로그래밍 모델을 지원한다. 사용자는 순차 코드 부분은 매스터 모듈로, 병렬 코드 부분은 슬래이브 모듈로 각각 나누어 프로그래밍하고, 이를 JICE에 환경 변수로 설정하여 주면, JICE는 구동 시에 슬래이브 모듈을 각각의 워커들에서 병렬적으로 수행하는 과정을 거쳐, 최종 결과를 사용자에게 제공한다.



(그림 4) 사용자 라이브러리

4.2 병렬 처리를 위한 사용자 라이브러리

JICE의 구현에는 효율적 병렬 프로그램 작성 및 수행을 위한 사용자 라이브러리를 포함한다. JICE에서 제공하는 사용자 라이브러리는 (그림 4)에서처럼 클래스 라이브러리 형태를 가지며, 매스터 모듈 클래스와 슬래이브 모듈 클래스로 구성되어 있다. 매스터 모듈 클래스는 JICE를 초기화 시켜주는 메소드와 주로 매스터 프로그램, 즉 순차 프로그램을 구동시키고 각 데몬과의 통신을 통해 병렬로 수행될 프로그램을 구동시키는 메소드들로 구성되어있다. 슬래이브 모듈 클래스는 각 워커에서 수행되는 병렬 프로그램 상에서 클라이언

〈표 1〉 플랫폼 초기화 라이브러리

메 소 드	설 명
GetWorkerNum()	브로커와의 통신을 통해 현재 등록되어 사용 가능한 워커들의 수를 알려준다.
GetWorkerAddress()	JICE에서 사용 가능한 워커들의 위치정보(IP 어드레스)를 제공한다.
SetWorker(String Add[], int workers)	JICE 사용자 프로그램을 수행하기 위해 필요한 워커의 수와 어드레스를 데몬에게 제공한다.
SetUsrPrg (String Usr_Prg[])	JICE에 참여한 워커들에게 필요한 사용자 프로그램을 전송하기 위해서 클라이언트 데몬에게 사용자 프로그램을 변수로 알려줌으로써 데몬에게 전송 수행 명령이 전달되면 워커에게 사용자 프로그램이 전송된다.
InitWorker()	요청한 워커들을 점유한 후, 초기화 작업을 수행한다.
InitSM()	JICE에서 사용되는 공유 메모리 시스템을 초기화한다.
Start()	JICE를 처음 구동시키는 메소드로서, 클라이언트가 제공하는 공유 메모리를 사용 가능하게 등록해주고 등록된 워커에게 필요한 사용자 프로그램을 전송시킨다.
Run()	JICE에 참여한 워커의 사용자 프로그램을 수행시킴으로써 병렬로 수행시킨다.
Finish()	JICE 프로그램을 종료하고 요청된 워커들을 브로커에게 반납한다.

〈표 2〉 사용자 프로그램 병렬화 라이브러리

메 소 드	설 명
SetParallel(int Parallelism, String[] Worker_Add)	사용자 프로그램이 최대한 사용할 수 있는 병렬성을 명시함으로써 클라이언트 내부에 참여한 워커의 개수만큼 스레드를 생성한 후, 각 워커를 생성된 스레드에 할당한다.
DoParallel()	SetParallel() 메소드에 의해 클라이언트에서 생성된 스레드를 수행시킨다. 즉, 병렬로 명시된 사용자 프로그램 코드 부분을 병렬로 수행한다.

트가 제공하는 공유 메모리의 접근과 워커간의 상호 통신을 하기 위한 통신 메소드가 제공된다. 사용자는 이들 클래스 라이브러리를 직접 사용하거나, 계승한 라이브러리를 작성하여 사용함으로써 병렬로 수행 가능한 사용자 프로그램을 작성할 수 있으며, JICE에서 효율적으로 수행할 수 있다. JICE에서 제공하는 사용자 라이브러리는 크게 플랫폼 초기화 라이브러리, 병렬 수행을 위한 라이브러리, 공유 메모리 및 통신 라이브러리로 구분된다.

4.2.1. 플랫폼 초기화 라이브러리

플랫폼 초기화 라이브러리는 JICE환경에서 사용자가 프로그램을 수행시키기 위한 라이브러리로, 클라이언트 시스템을 초기화하고, 모든 워커 시스템들을 JICE환경에서 수행될 수 있도록 준비해주고, 전체 JICE를 구동시켜주고 관리하는 라이브러리이다. 이를 위해 플랫폼 초기화 라이브러리는 클라이언트가 브로커로부터 JICE를 활용할 수 있는 유효 자원에 대한 정보를 제공받기 위해 사용 가능한 메소드들과 워커 시스템에서 수행할 프로그램을 전송해주는 메소드와 같이 JICE와 사용자간의 통신 인터페이스의 역할을 담당하는 메소드를 제공한다.

4.2.2. 사용자 프로그램 병렬화 라이브러리

JICE에서 수행될 사용자 프로그램은 명시적으로 병렬화 되어야 한다. 즉, 프로그램과 데이터가 병렬로 수행될 수 있는 구조를 가져야 한다. 이를 위해 JICE에서는 프로그램과 데이터의 병렬성을 명시하기 위한 라이브러리를 제공한다. 사용자는 이들 라이브러리를 사용하여, 프로그램의 병렬성을 명시하고, 병렬로 구동시킬 수 있다. 현재의 JICE 모델에서는 데이터 병렬 프로그램을 수행하는 SPMD(Single Program Multiple Data)모델을 지원한다.

4.2.3. 공유 메모리 및 통신 라이브러리

JICE에서는 사용자 프로그램의 병렬 수행 중 워커간에 공유되는 데이터를 위한 통신과 동기화를 위해 글로벌 공유 메모리를 제공한다. 글로벌 공유 메모리는 클라이언트 시스템에서 자바가 제공하는 원격 메소드 호출 메카니즘을 사용하여 작성된다. 공유 메모리를 원격 객체로 둠으로써 원거리에 있는 워커에 의해 이 객체를 전송할 수 있거나, 이 객체를 지역 객체인 듯이 호출해서 사용할 수 있기 때문에 글로벌 공유 메모리의 운용을 간편하게 하는 장점을 제공한다. JICE는 사용자 일관성을 위하여 일반 메모리 시스템과 같은 인터페이스를 제공한다. 현재의 JICE의 공유 메모리 시스템은 정수형 데이터와 부동 소수 등 기본 데이터 타입에 대한 공유 메모리를 지원한다. 또한 분산 병렬 컴퓨팅 환경의 특성상 읽기가 필요한 데이터에 대해서는 여러 차례에 작은 데이터를 읽어오는 것보다는 한번에 많은 데이터를 읽어 들이는 것이 효율적이기 때문에, 블록 읽기와 블록 쓰기가 가능하도록 SetBlockMem()과, GetBlockMem()과 같은 메소드를 제공하고 있다.

4.3. 라이브러리 활용

JICE를 사용하기 위해 사용자는 JICE에서 제공하는 라이브러리를 이용, 프로그램을 병렬로 수행할 부분(슬래이브 모듈)과 순차적으로 수행할 부분(매스터 모듈)으로 구분하여 프로그램을 작성한다. (그림 5)와 (그림 6)은 JICE 프로그래밍 시 사용자가 매스터 모듈과 슬래이브 모듈에서 작성해야하는 JICE 프로그래밍 절차를 보여주는 코드부분이다. 매스터 모듈의 첫 번째 단계는 워커와의 통신을 담당하는 데몬 클래스를 선언하고, 제공되는 메소드를 이용해서 클라이언트가 JICE를 구동시키기 위한 초기작업으로 브로커로부터 현재 사용 가능한 워커의 정보를 받아오는 메소드를 호출한다. 다음으로 각 워커의 정보를 JICE에서 수행시키기 위해 각 워커와 통신을 담당하는 데몬에게 전달하고, 병렬 수행 코드 블록(슬래이브 모듈)을 참여한 워커에게 전송한다. 다음단계는 JICE에 참여하는 워커를 초기화하고 수행시키고, 마지막으로 워커로부터 수행결과를 수집하고 자원을 반납함으로써 매스터 프로그

〈표 3〉 공유 메모리 및 통신 라이브러리

메 소 드	설 명
SM_NewMem_Int(int Size)	사용자는 클라이언트 시스템 상에 파라메터로 제공된 크기만큼의 정수형 메모리 영역을 공유 메모리로 할당한다.
SM_NewMem_Float(int Size)	사용자는 클라이언트 시스템 상에 파라메터로 제공된 크기만큼의 부동소수점 메모리 영역을 공유 메모리로 할당한다.
SM_SetInt(int Value)	정수형 공유 메모리에 파라메터로 전송된 4 바이트의 정수형 데이터를 저장한다.
SM_SetBlockMem_Int (int Value[])	정수형 공유 메모리에 파마메터로 전송된 한 블록의 정수형 데이터를 저장한다.
SM_GetInt()	정수형 공유 메모리에서 4바이트의 정수형 데이터를 읽는다.
SM_GetBlockMem_Int (int Start, int End)	정수형 공유 메모리에서 파라메타로 전송된 공유메모리의 시작과 끝 위치에 있는 데이터를 전송한다.
SM_SetFloat(float Value)	부동 소수형 공유 메모리에 파라메터로 전송된 4 바이트의 부동소수형 데이터를 저장한다.
SM_SetBlockMem_Float(float Value[])	부동 소수형 공유 메모리에 파라메터로 전송된 한 블록의 부동 소수형 데이터를 저장한다.
SM_GetFloat()	부동 소수형 공유 메모리에서 4바이트의 부동 소수형 데이터를 읽는다.
SM_GetBlockMem_Float (int Start, int End)	부동 소수형 공유 메모리에서 파라메타로 전송된 공유메모리의 시작과 끝 위치에 있는 데이터를 전송한다.

```
WorkerDaemon Workerd = new WorkerDaemon();
worker_add = Workerd.GetWokerAddress();
Workerd.SetWorker( worker_add, count);
Workerd.SetWorkerUserProg( user_worker );
Workerd.Start();
Workerd.Run();
Master mmc = new Master();
mmc.setParallel( N_parallel, jps_add );
mmc.doParallel();
Workerd.Finish();
```

(그림 5) 매스터 모듈 라이브러리 사용 예

```
SharedMemory sm
float[] x;
float[] y;
x = sm.SM_GetPartMem_float(start, size)
                .
          (연산 수행 코드)
                .
y = sm.SM_SetPartMem_float(start, size)
```

(그림 6) 슬래이브 모듈 라이브러리 사용 예

램 작성을 마치게 된다. 각 워커에 할당되는 병렬 수행 코드 블록인 슬래이브 모듈은 (그림 6)에서처럼 초기 데이터를 얻기 위한 공유 메모리 억세스 작업과 병렬 수행코드 그리고 결과를 리턴하기 위한 공유 메모리 억세스 코드로 구성된다.

4.4 JICE 환경의 수식적 분석

(그림 3)에서 T_{S_i}는 i번째 순차적 코드 부분(S_i)을, T_{P_i}는 i번째 병렬 코드 부분(P_i)을 수행하는데 소요되는 시간이라고 하면, 전체 수행 시간(T)은 다음과 같은 수식 (1)로 표현된다.

$$T = \sum_{i=1}^{n} T_{S_i} + \sum_{i=1}^{n-1} T_{P_i} \qquad (1)$$

순차 코드 부분의 수행 시간을 좀더 자세히 알아보면, T_{S_i}는 이전 병렬 코드 부분(P_{i-1})의 결과를 수집하는데 소요되는 시간인 $T^{G}_{S_i}$, 다음 병렬 코드 부분(P_i)을 수행하기 위한 코드 및 데이터 할당하는데 소요되는 시간인 $T^{D}_{S_i}$, 순수 순차 처리를 위해 소요되는 시간인 $T^{A}_{S_i}$의 세 부분으로 나뉘어진다. 그러나, 첫 번째 순차 코드 부분(S_0)은 이전 병렬 코드 부분의 결과 수집 시간이 없으며, 마지막 순차 코드 부분(S_n)은 다음 병렬 코드 부분을 위한 할당 시간을 필요로 하지 않는다.

$$T_{S_i} = \begin{cases} T^{A}_{S_0} + T^{D}_{S_0}, & \text{if } i=0 \\ T^{G}_{S_i} + T^{A}_{S_i} + T^{D}_{S_i}, & \text{if } 0<i<n \\ T^{G}_{S_n} + T^{A}_{S_n}, & \text{if } i=n \end{cases} \qquad (2)$$

$$T_{P_i} = T^{M}_{P_i} + T^{E}_{P_i}.$$

병렬 코드 부분의 수행 시간인 T_{P_i}는 공유 메모리 참조를 기준으로 두 가지 부분으로 구분되어진다. $T^{M}_{P_i}$

은 공유 메모리 참조에 소요되는 시간이고, $T^{E}_{P_i}$는 공유 메모리 참조 이외의 산술 연산에 소요되는 시간을 나타낸다.

수식 (2)에 따라, 전체 수행 시간은 다시 수식 (3)과 같이 클라이언트에서의 순차 연산 시간, 워커에서의 병렬 연산 시간, 통신 시간으로 나뉘어진다. 통신 시간은 병렬 수행을 위한 병렬 수행 어플리케이션의 분배 시간($T^{D}_{S_i}$), 각 워커로부터의 병렬 수행 결과 수집 시간($T^{G}_{S_{i+1}}$), 병렬 수행 중 공유 메모리 참조 시간($T^{M}_{P_i}$)을 포함한다.

$$T = \sum_{i=0}^{n} T^{A}_{S_i} + \sum_{i=0}^{n-1} T^{E}_{P_i} + \sum_{i=0}^{n-1} T^{C}_{i},$$
$$T^{C}_{i} = \left(T^{D}_{S_i} + T^{M}_{P_i} + T^{G}_{S_{i+1}} \right). \quad (3)$$

병렬 코드 부분을 수행하기 위한 작업 분배 시간은 클라이언트가 비 공유 데이터와 코드를 각각의 워커들로 순차적으로 전송하는 시간으로 나타내며, 병렬 수행 결과 수집 시간은 작업 분배와는 반대로, 모든 워커의 수행 결과들이 클라이언트에 수집되는 시간이다. 일반적으로 인터넷에서는 데이터 전송이 패킷단위로 수행되기 때문에, 작업 분배와 결과 수집은 패킷크기에 맞는 블록단위로 수행된다. $T^{D}_{S_i}$와 $T^{G}_{S_i}$ 은 <표 1>에 정의된 기호들을 사용해 다음과 같은 수식으로 표현된다.

$$T^{D}_{S_i} = \left\lceil \frac{S_{PD} + S_{SD}}{S_P \cdot N} \right\rceil \cdot T_U,$$
$$T^{G}_{S_i} = \left\lceil \frac{S_{SD}}{S_P \cdot N} \right\rceil \cdot T_U. \quad (4)$$

수식의 단순화를 위하여, 임의의 병렬 코드 부분에서 작업량이 각 워커에 균등하게 분배되어, 공유 메모리를 참조할 비율이 모든 워커에서 동일하다고 가정하면, 워커에서의 연산 과정은 모든 워커에서 동시에 수행될 수 있으나, 클라이언트의 공유 메모리를 참조해야 하는 부분은 데이터의 일관성을 유지하기 위해 순차적으로 수행되어야 한다. 그러므로, $T^{E}_{P_i}$와 $T^{M}_{P_i}$는 다음과 같이 표현된다.

$$T^{E}_{P_i} = (1 - R_M) \cdot \left\{ \frac{T_{P_i}}{N} \right\},$$
$$T^{M}_{P_i} = R_M \cdot T_{P_i} \left(\frac{1 - R_{MW}}{N} + R_{MW} \right) \cdot T_U. \quad (5)$$

〈표 1〉 용어정의

기호	의 미
S_{PD}	i 번째 병렬 코드 부분의 private 데이터 크기
S_{SD}	i 번째 병렬 코드 부분의 공유 데이터 크기
S_P	단위 패킷 사이즈
R_M	공유 메모리 참조의 비율
R_{MW}	공유 메모리 참조의 write-update의 비율
R_{MR}	공유 메모리 참조의 read의 비율
R_{MWL}	지역공유메모리의 write-update 비율
R_{MWG}	전역공유메모리의 write-update 비율
T_{Pi}	i 번째 병렬 코드 부분 순차 수행 시간
T_U	단위 패킷 데이터 전송 시간
T_{UL}	슬레이브 클라이언트에서의 단위 패킷 데이터 전송 시간
T_{UG}	매스터 클라이언트에서의 단위 패킷 데이터 전송 시간
N	워커의 수
G	슬레이브 클라이언트의 수
A	멀티캐스팅 계수

그러나 다중 관리로 인해 JICE가 제공하는 공유 메모리는 LSM과 GSM의 두가지 종류로 구분된다. GSM은 모든 워커가 순차적으로 참조하게 되지만, LSM은 그룹별로 병렬로 참조가 가능하다. 따라서 모든 워커에서 LSM의 참조 비율과 GSM의 참조 비율이 동일하다고 가정하면, $T^{M}_{P_i}$은 다음과 같이 변화된다.

$$T^{M}_{P_i} = R_M \cdot T_{P_i} \cdot \left\{ \left(\frac{1 - R_{MW}}{N} + \frac{R_{MWL}}{G} \right) \cdot T_{UL} + R_{MWG} \cdot (T_{UG} + A \times T_{UL}) \right\}. \quad (6)$$

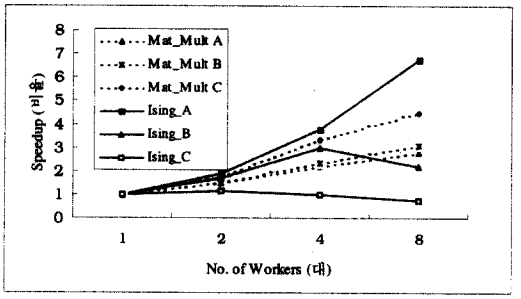
5. 성능 평가

본 장에서는 수식적 분석에 대한 시뮬레이션 결과와 벤치마크를 통해서 JICE가 효율적인 분산 병렬 컴퓨팅 환경임을 보이고 참여한 워커의 효율성을 살펴봄으로써 JICE가 제공하는 프로그래밍 모델과 라이브러리가 효율적임을 보여준다.

5.1. JICE 기본 구성 모델의 성능 평가

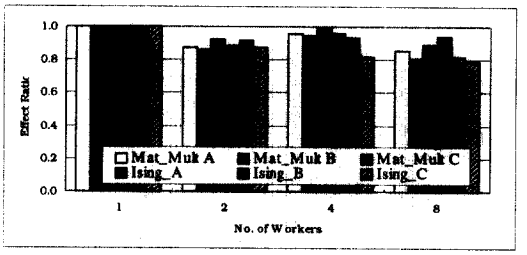
JICE 기본 구성 모델의 성능 평가 실험은 10 Base-T 이더넷으로 연결되어 있는 다양한 성능을 가진 8대의 컴퓨터에서 수행되었다. 모든 시스템은 JDK1.1.7과 자바 호환 웹 브라우저를 탑재하고 있다. 성능 평가는 병렬 분산 컴퓨팅에서 벤치마크로 많이 이용되는 행렬 곱셈(Matrix Multiplication) 어플리케이션과 에너지 입자의 움직임을 시뮬레이션 하는 이징(Ising) 어플리케이션을 사용하였다. 이징 어플리케이션은 2차원 격자가 할당

된 에너지 입자를 시뮬레이션 하는 어플리케이션으로써, 하나의 입자가 네 방향중의 한쪽으로 움직일 가능성은 모두 동일하다고 가정하고, 에너지 입자의 움직임을 어떤 주어진 상태에 따라 시뮬레이션 하게된다. 실험에서 각 입자의 초기 상태는 랜덤하게 주고 40000개 입자의 움직임을 2차원 격자에서 랜덤하게 조사하였다. 성능향상의 요인을 알아보기 위해 각 어플리케이션마다 약간씩 통신비와 연산비를 조정하여 실험하였다. 행렬 곱셈 벤치마크는 행렬의 크기를 달리한 정수 행렬에 대하여 수행되었고(Mat_Mult A, Mat_Mult B, Mat_Mult C), 이징 어플리케이션은 연산과 통신의 비를 각각 달리하는 변형 이징 어플리케이션을 수행하였다(Ising_A, Ising_B, Ising_C). 행렬 곱셈은 블록 데이터전송을 통해 통신하기 때문에 공유메모리 경쟁이 이징보다 적다. Ising_A는 800번의 반복을 가지는 이징 프로그램으로 모든 워커에게 입자가 동일하게 분포되어있고, 모든 워커는 연산도중에 통신을 필요로 하지 않는다. Ising_B는 800번의 반복동안에 매 10회 반복마다 한번의 통신을 갖도록 변형하였고, Ising_C는 통신의 비와 연산의 비가 1:1인 경우에 대해서 실험하였다.



(그림 7) 성능 향상 비

(그림 7)은 워커 수가 1대에서부터 8대까지 변화할 때 벤치마크의 성능향상 비로써, 성능향상 비는 각 벤치마크의 소요 시간을 하나의 워커로 수행하였을 때 소요된 시간으로 나누어줌으로서 얻을 수 있다. 행렬 곱셈의 경우는 블록 전송을 하기 때문에 통신이 빈번하게 발생하는 Ising_B나 Ising_C에 비해 성능향상이 더 좋게 나타난다. Ising_A는 연산도중에 통신이 발생하지 않기 때문에 거의 선형적인 성능향상을 보이며, Ising_B와 Ising_C는 통신의 비율이 25%, 50%로 상당히 높고, 블록 데이터 전송이 아니기 때문에 통신으로 인한 오버헤드가 4대 이상일 경우에는 병렬 수행의 이득보다 크기 때문에 성능이 오히려 감소하고 있음을 보여준다. 따라서 네트워크 오버헤드를 가지는 JICE환경에서는 통신의 비율이 적고 블록 데이터전송을 하는 어플리케이션이 효과적으로 수행 가능한 응용 프로그램임을 알 수 있다. 이러한 성능 향상 폭은 암달(Amdahl)의 법칙으로 쉽게 설명된다. 병렬로 수행될 수 있는 부분이 많은 Ising_A는 워커 수에 따른 성능 향상 폭이 크게 되며, 순차 처리 부분이 많은 Ising_B나 Ising_C 순서로 성능 향상이 나타난다. JICE에서의 워커 수 증가에 의한 성능 향상은 공유 메모리를 가지는 다중 프로세서 시스템과 비교하였을 때, 유사한 성능 향상 결과를 보여준다. 이를 통해 JICE의 프로그래밍 모델과 병렬화 라이브러리가 JICE의 특성에 맞도록 충분히 효율적으로 구성되어 있음을 보여준다.



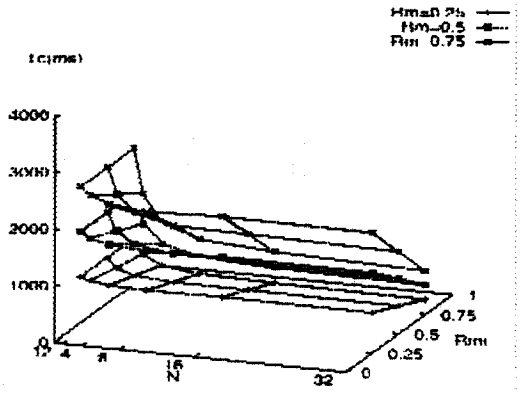
(그림 8) 워커 수에 따른 효율성

(그림 8)은 JICE의 효율성을 나타낸 그래프이다. 사용된 워커의 수를 고려한 소요시간, 즉 할당된 자원의 사용시간은 작업에 소요되는 비용을 의미한다. 하나의 벤치마크를 수행하는 동안 참여한 워커들은 시간을 할애하기 때문에 이는 JICE환경의 효율성을 나타내는 기준으로 사용될 수 있다. 따라서 (그림 8)은 하나의 워커를 사용하였을 때의 효율성을 1로 했을 때, 워커수의 증가에 따른 JICE의 효율성을 나타낸다. JICE의 효율성은 보다 많은 워커를 사용함에 따라 낮아지는 결과를 보여준다. 이는 워커의 수가 증가함에 따라 병렬 수행을 위한 오버헤드가 증가하게 되고, 공유 메모리를 참조할 때 네트워크의 충돌(conflict)에 의한 평균 참조 시간이 길어지는데 기인한다. 그림에서는 전체적으로 JICE의 효율성은 80%이상을 유지함을 보이며, 빈번한 통신으로 인한 Ising_C는 가장 효율성이 낮은

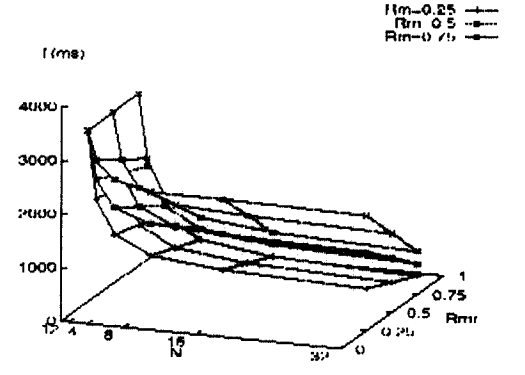
결과를 보여주고 통신량이 적은 행렬 곱셈의 효율성이 높음을 알 수 있다. 자바 RMI가 내부적으로 가지는 통신 메카니즘의 오버헤드와 순수한 자바로 작성된 JICE의 공유 메모리 시스템이 메시지 전송에 의해 처리됨을 고려하면, JICE의 프로그래밍 모델이나 병렬화 라이브러리는 상당히 효율적인 성능을 제공함을 알 수 있다.

5.2 JICE 다중 관리 시스템 분석

(그림 9)와 (그림 10)은 수식 (4), (5)를 근거로 클라이언트가 하나일 경우의 JICE 성능을 시뮬레이션한 결과이다. (그림 9)는 JICE에 참여하고 있는 워커의 수가 늘어남에 따른 공유 메모리 참조 시간의 변화를 보여준다. 노드가 증가할수록, 클라이언트로의 통신집중으로 인해, 메모리 참조 시간의 감소율이 점점 줄어들게



(그림 9) 통신시간(T_i^c)

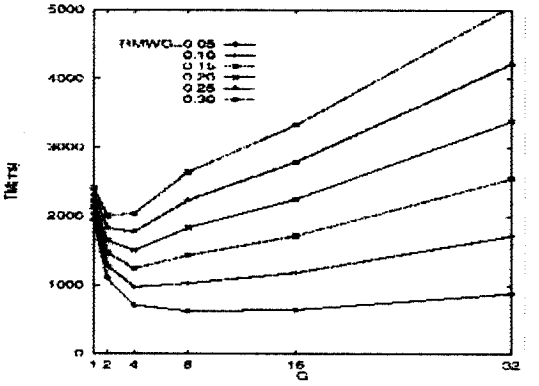


(그림 10) 전체 수행 시간(T_{Pi})

되어 대략 8개 이상의 노드에서는 동일한 메모리 참조 시간을 보임을 알 수 있다. (그림 10)은 전체 수행 시간의 변화를 나타낸다. 메모리 참조 비율이 높은 프로그램에서 노드 수 증가에 따른 성능 향상 비율이 낮아진다. 이들 프로그램은 많은 부분이 메모리 참조 시간에 영향을 받으며, 메모리 참조 시간이 8개 이상의 노드에서는 비슷한 성능을 보여주기 때문이다. 이는 메모리 참조 비율이 높은 프로그램에서의 효율적인 병렬성이 감소함을 암시한다.

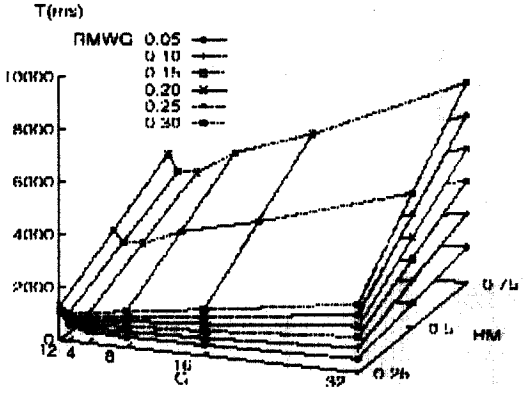
(그림 11)과 (그림 12)는 32개 노드를 다양한 수의 관리 시스템 그룹으로 구분하였을 때, 공유메모리 참조시간 T_M과 전체 수행시간 T의 변화를 보여준다. 그림은 수식 (6)을 바탕으로 작성되었으며, 그룹화에 의한 영향을 명확히 하기 위해, $R_{MR}=0$을 가정하였다. 그림 11은 $R_M=0.5$일 때의 T_M의 변화를 보여준다. 그림에서 최소의 T_M값을 보여줄 때의 그룹수가 최적의 그룹수, G_{OPT}를 의미한다. 그림에서 R_{MWG} 값이 변화함에 따라 G_{OPT}가 변화함을 알 수 있다. $R_{MWG}=0.05$의 비율에서는 $G_{OPT}=N$을 나타내며, R_{MWG}이 커질수록 G_{OPT}는 감소한다. 이는 R_{MWG}이 커질수록, 데이터 일관성을 위한 오버헤드의 비율이 커져서, 그룹의 수에 따른 증가폭이 커지기 때문이다. 그림 12는 R_M에 따른 전체 수행시간 T의 변화를 보여준다. T는 R_M의 변화에 따라 변화한다. R_M이 증가함에 따라, 그룹화에 의한 영향이 커지게 되며, 수행 시간의 차이폭도 커지게 된다.

(그림 13)은 효율적 그룹화를 이루었을 때의 수행 시간을 그룹화 이전 모델과 비교한 결과이다. 그룹화의 성능은 전역 메모리 쓰기 비율에 크게 영향을 받음을

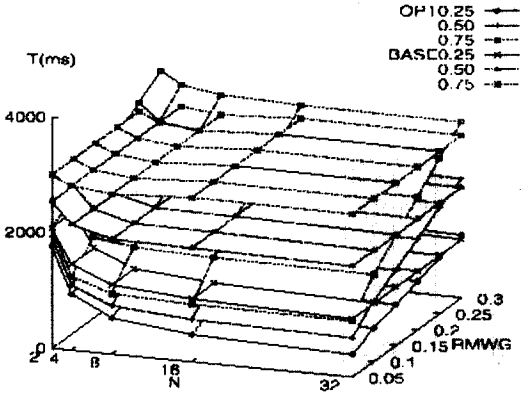


(그림 11) 공유메모리 참조 시간 (R_M=0.5)

알 수 있다. R_{MWG}이 적을수록 성능 향상이 많다. 이는 지역 메모리 참조 비율이 높아서, 많은 부분을 병렬로 수행할 수 있기 때문이다. R_{MWG}에 따라, 기본 모델에 비해 최고 2.5~3배의 성능 향상을 기대할 수 있다. 그룹화의 성능은 R_{MWG}이 높아질수록, 기본 모델과 비슷한 성능을 보여주며, 0.3이상의 쓰기 비율을 가지는 경우에는 기본 모델보다 성능이 나빠져서, 기본 모델의 대략 0.9의 성능이 기대된다. 이는 데이터 일관성을 위한 오버헤드가 성능 향상폭을 능가하기 때문이다.



(그림 12) 전체 수행 시간



(그림 13) 다중 관리에 의한 성능 향상

6. 결 론

인터넷이 발전함에 따라 네트워크에 연결된 유효 자원을 효과적으로 사용하는 방법으로써 인터넷 기반의 병렬 수행 플랫폼인 JICE가 설계 및 구현되었다. JICE의 기본 구성 모델은 클라이언트, 워커, 브로커들의 컴포넌트들로 이루어지며 워커간의 통신을 위해 공유 메모리 시스템을 제공한다. 사용자는 JICE에서 제공되는 다양한 사용자 라이브러리를 사용하고, 매스터-슬래이브 프로그래밍 모델을 사용하여, 프로그램을 JICE에서 병렬로 수행할 수 있다.

본 연구에서는 JICE 환경에서 수행되는 어플리케이션의 수식적 분석과 JICE에서의 벤치마크 프로그램을 수행시키는 실험을 통하여 JICE의 성능이 조사되었다. 실험을 통해 JICE환경은 워커수가 증가함에 따라, 기존의 공유 메모리 다중 프로세서 시스템과 유사한 성능 향상이 나타남을 알 수 있었고, 수행 시간을 분석을 통해 다중 관리를 지원하는 JICE는 전역 공유 메모리 쓰기 비율에 크게 영향을 받음을 알 수 있었다. 전역 공유 메모리 쓰기 비율에 따라 다중 관리를 지원하는 JICE는 기본 모델에 비해 최고 2.5~3배의 성능 향상을 기대할 수 있다. 또한 자원 사용 시간에 대한 분석을 통해, JICE의 프로그래밍 모델과 사용자 병렬화 라이브러리의 구성이 효율적임을 보였다. 자바 프로그램의 병렬화 연구와 네트워크 고속화에 대한 연구들에 의해 다양한 시스템들의 자원을 활용한 인터넷상의 병렬처리 응용 기술이 요구된다.

참 고 문 헌

[1] Tim Brecht, Harjinder Sandhu, Meijuan Shan, and Jimmy Talbot, "ParaWeb:Towards World-Wide Supercomputing," *7th ACM SIGOPS European Workshop*, pp.181-188, Sep. 1996.

[2] Arash Baratloo, Mehmet Karaul, Zvi Kedem, and Peter Wyckoff, "Charlotte : Metacomputing on the Web," *9th International Conference on Parallel and Distributed Computing Systems*, pp.151-159, Sep. 1996.

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

[4] Albert Alexandrov, Maximilian Ibel, Klaus E.

Schauser, and Chris Scheiman, "SuperWeb : Towards a Global Web-Based Parallel Computing Infrastructure," *11th International Parallel Processing Symposium*, pp.100-106, Apr. 1997.

[5] Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, and Vaidy Sunderam, "PVM : Parallel Virtual Machine - A Users' Guide and Tutorial for Networked Parallel Computing," The MIT Press, Cambridge, Massachusetts, 1994.

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

[7] Chun-Mok Chung and Shin-Dug Kim, "A Dual-threaded Java Processor for Java Multithreading," *International Conference on Parallel and Distributed Systems*, pp.693-700, Dec. 1998.

[8] Sarita V. Adve and Kourosh Gharachoroloo, "Shared Memory Consistency Models : A Tutorial," *IEEE 0Computer*, Vol.29, No.12, pp.66-76, Dec. 1996.

[9] Donald Yeung, John Kubiatowicz, and Anant Agarwal, "MGS : A Multigrain Shared Memory System," *23rd Annual International Symposium on Computer Architecture*, pp.44-55, May 1996.

[10] Richard P. Matin, Amin M. Vahdat, David E. Culler, and Ehomas E. Anderson, "Effect of Shared memory reference Latency, Overhead, and Bandwidth in a Cluster Architecture," *24th Annual International Symposium on Computer Architecture*, pp.85-97, June 1997.

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

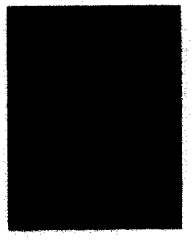
[12] 박지민, 맹혜선, 한탁돈, 김신덕, "네트워크로 연결된 자바스테이션 상에서의 웹 컴퓨팅," 1998년 한국정보과학회 봄 학술대회, pp.392-395, April 1998.

신 필 섭

email : ambush@kurene.yonsei.ac.kr
1997년 광운대학교 컴퓨터공학과 졸업(학사)
1998년~현재 연세대학교 대학원 컴퓨터과학과 석사과정
관심분야 : 분산병렬컴퓨팅, 인터넷 컴퓨팅, 자바

정 준 목

email : chunmok@lgic.co.kr
1997년 광운대학교 컴퓨터공학과 졸업(학사)
1999년 연세대학교 대학원 컴퓨터과학과 졸업(석사)
1999년~현재 LG정보통신 정보응용기술실 연구원
관심분야 : 분산병렬컴퓨팅, 인터넷 컴퓨팅, 자바

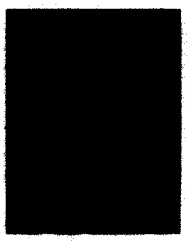


맹 혜 선

email : carchi@kurene.yonsei.ac.kr
1994년 연세대학교 전산과학과 졸업(학사)
1996년 연세대학교 대학원 컴퓨터과학과 졸업(이학석사)
1996년~현재 연세대학교 대학원 컴퓨터과학과 박사과정
관심분야 : 분산병렬컴퓨팅, 인터넷 컴퓨팅, 자바

홍 원 기

email : wkhong@kurene.yonsei.ac.kr
1995년 연세대학교 컴퓨터과학과 (학사)
1997년 연세대학교 컴퓨터과학과 (석사)
1997년~현재 연세대학교 컴퓨터과학과(박사과정)
관심분야 : VLIW, 프로세서-메모리 집적 구조, 병렬처리, 3D 그래픽 가속기



김 신 덕

email : sdkim@kurene.yonsei.ac.kr

1982년 연세대학교 공과대학 전자공학과 졸업(학사)

1987년 University of Oklahoma 전기공학(석사)

1991년 Purdue University 전기공학(박사)

1993년~1995년 광운대학교 컴퓨터공학과 조교수

1995년~현재 연세대학교 공과대학 기전공학부 정보산업전공 부교수

관심분야 : 병렬처리 시스템, 컴퓨터 구조, 인터넷 컴퓨팅