WWW 기반 자바 병렬 처리 시스템의 설계 및 구현과 성능 향상 기법

한연희*, 박찬열*, 정영식**, 황종선*
*고려대학교 컴퓨터학과, **원광대학교 컴퓨터공학과

A Design and Implementation of a Java Parallel Processing System based on the WWW and Its Performance Improvement Schemes

YounHee Han*, Chan Yeol Park*, YoungSik Jung**, Chong-Sun Hwang*
*Dept. of Computer Science and Engineering, Korea University,
**Dept. of Computer Science and Engineering, WonKwang University

요 약

인터넷이 급속도로 발전하여 이러한 환경에서 네트워크로 연결된 여러 호스트들의 자원을 이용하는 시도가 활발하게 이루어지고 있다 본 논문은 이러한 환경에서 의뢰인-병렬처리서버-작업자 구성을 이용하여, 작업자 애플릿을 임의의 호스트에 분산시키고, 대량의 연산수행을 지닌 작업을 배분하여 수행시킨 뒤, 그 결과를 의뢰인에게 보여주는 WWW 기반 자바 병렬 시스템의 설계 및 구현에 관하여 기술한다 성능 향상을 위해서 자바의 원격 메소드 호출(Remote Method Invocation)을 이용한 애플릿간 통신 메커니즘을 구현하고, 작업자의 결과를 의뢰인에게 서버를 거치지 않고 곧바로 보낸도록 한다. 또한, 각 작업자마다의 성능비를 분석하여 태스크들을 할당하는 방법을 통해 작업 시간을 단축시킨다. 이 시스템에 연산 수행량이 많은 프랙탈 이미지 처리 작업을 배분하여 수행시키고, 작업 태스크의 크기에 따른 수행성능과 작업 배분방법에 따른 수행성능을 측정하여 그 결과를 비교, 제시한다.

1. 서 론

WWW는 수 백만 대의 컴퓨터를 연결한 가장 큰 가상 시스템이 되고 있다. 게다가, 플랫폼 독립적이며, 네트워크 관련 API 및 쓰레드 처리가 용이한 자바 때문에 클라이언트-서버 스타일 인터넷 프로그램 개발이 용이해 졌다[1]. 특히, 브라우저로 다운로드 되어 안전하게 수행되는 이동 바이트 코드인 애플릿으로 인하여, 즉시 수행 가능한 코드를 분산시키는 일이 쉬워졌다. 분산된 애플릿은 내재된 네트워크 API를 통하여 여러 연산을 수행한 후 그 결과를 즉시 다른 서버나 애플릿으로 전달하여 정보의 교환을 통한 효율성을 높일 수 있다[2].

병렬 컴퓨팅의 환경 및 도구로 웹과 자바를 활용할 때 얻을 수 있는 몇 가지 장점들이 있다. 첫 번째, 웹과 브라우저의 사용으로 시스템 확장성(Scalability)이 무한정으로 높아진다. 두 번째, 자바의 플랫폼 독립적인 특성은 여러 호스트들의 이질성(heterogeneity)을 쉽게 극복하게 해준다. 세 번째, 호스트 CPU 활용도를 극대화 할 수 있다.

[3]은 자바를 활용하여 인터넷 환경에서 병렬 시스템을 구축하며, 웹 기반 병렬시스템은 [4, 5]에서 구축한다. 이 논문들에서 구현한 한 Javelin은 웹 기반 병렬 시스템의 원형을 제시하며, 구축된 시스템에 광선추적법(RayTracing) 및 머르센(Mersenne) 소수의 탐색 프로그램을 탑재하여 그 수행 결과를 제시한다. Javelin처럼, 만약 시스템이 TCP 및 UDP를 이용하는 소켓만을 이용한다면, 애플릿은 단지 자기가 로드되어진 호스트로만 네트워크 연결을 행할 수 있는 원천지 호스트(host-of-origin) 연결 보안 정책을 따르게 된다. 그러므로, 작업자의 결과는 반드시 서버로 보내어 다시 의뢰인에게 보내는 수 밖에 없다.

본 논문에서 제시하는 시스템은 기존의 의뢰인-병렬처리서버-작업자 구성을 이용한다 웹 서버가 동작하는 호스트에 작업자, 의뢰인 관리와 작업 태스크의 배분을 위한 병렬 처리 서버가 동작한다. 의뢰인(client)들은 브라우저를 통하여 서비스 요청 애플릿을 웹서버로부터 받아 서비스를 개시하며, 작업자(worker)들도 작업자 애플릿을 받아 수행하여, 의뢰인으로부터 개시된 서비스 요청을 서버로부터 간접적으로 나누어 받아 일을 해주고 그 결과를 의뢰인에게 직접 보내준다. 본 논문의 시스템은 작업자가 작업한 결과를 자바의 RMI(Remote Method Invocation) 기능을 이용해서, 서버를 거치지 않고 곧바로 의뢰인에게 전달한다. 이렇게 함으로써 서버의 부담을 크게 덜어 주면서도 총 수행 시간도 단축할 수 있다[6].

또한, 본 논문은 작업자들에게 균등하게 배분하는 방법과 그 작업자들의 성능비에 따라 다르게 배분하는 방법을 각각 사용하며 각 방법을 비교, 분석한다. 결국, 성능비에 따라 다르게 배분하는 방법이 더 효율적임을 보이고, 본 시스템은 이 방법을 이용하여 구축한다.

본 논문은 구현된 시스템에서 프랙탈 이미지를 처리하는 작업을 수행시키고 성능을 평가한다. 두 가지 태스크 할당 방법에 따른 성능 비교 및 태스크의 크기 따른 비교를 성능 평가의 주요 사항으로 정한다.

2. 시스템 구조 및 설계

2.1 기본적 요소

본 시스템은 다음과 같은 여러 요소들로 구성된다.

- 의뢰인(Client) : 작업을 서버에게 의뢰하고 결과를 받는다.

† 이 연구는 정보통신부의 대학기초연구 지원사업에 의하여 수행 되었음.

• 작업자(Worker) : 작업 수행 알고리즘 코드를 지니며, 서버로부터 작업에 대한 명세를 받자마자 그 명세를 분석하여 알고리즘을 수행한다. 수행이 끝나는 즉시 의뢰인에게 그 결과를 보낸다.

• 병렬 처리 서버(Parallel Processing Server) : 의뢰인과 작업자의 등록 및 관리를 수행하며, 의뢰인으로부터 작업이 개시되면, 태스크 분할후 각 작업자에 해당 태스크들을 할당한다.

• 웹 서버(Web Server) : 의뢰인과 작업자는 자바 애플릿이기 때문에, 애플릿을 포함하는 HTML 문서를 공급해줄 수 있는 웹서버가 필요하다. 이 웹서버는 병렬 처리 서버와 같은 호스트에 존재한다.

2.2 요소들간의 통신 프로토콜

의뢰인, 작업자와 서버와의 통신 프로토콜을 그림 1에서 보여준다. .의뢰인은 CR(Client Ready) 메시지를 서버에게 알리고 의뢰인 ID를 받는다. 한편, 작업자는 WR(Worker Ready) 메시지와 자신의 CPU능력을 Mflops/s 단위로 함께 전달한 후에 작업자 ID를 받는다. 이후, 의뢰인은 JS(Job SPec) 헤더와 함께 작업 관련 명세 및 자신의 원격 참조값(Remote Reference)를 서버에게 보낸다. 이 때, 서버는 각 작업자들의 성능을 고려하여 작업자들로의 태스크 할당 작업을 개시하며, 의뢰인이 보내준 작업 명세와 함께, 태스크 할당 정보를 각 작업자들에게 전달한다. 한편 작업자에게 의뢰인 원격 참조(Remote Reference)를 그 명세와 함께 보내는데, 이것은 작업자가 작업을 수행한 후 그 결과를 곧바로 의뢰인에게 보내기 위해 필요한 정보이다. 이러한 RMI 및 원격 참조를 이용하여 의뢰인의 메소드를 호출하는 콜백(callback) 방법은 애플릿간 통신을 자유롭게 할 수 있는 주요 메커니즘이다.

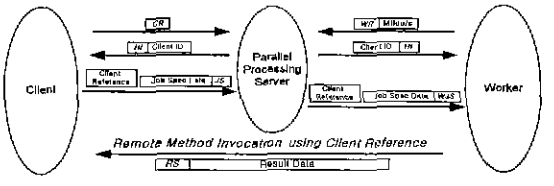

그림 1 시스템 프로토콜

2.3 시스템 설계

본 시스템은 [그림 2]에 제시된 대로 총 3개의 층으로 구성된다. 가장 하위층에는 세션층이 존재하며, 중간층에는 병렬처리층, 마지막으로 가장 상위층에는 응용층이 존재한다. 세션층은 자바에서 제공하는 소켓(Socket)과 서버소켓(ServerSocket) 및 이들로부터 얻어내는 입력스트림(InputStream)과 출력스트림(OutputStream)을 함께 하나의 클래스인 Session의 구성원으로 묶어서 상위층으로 하여금 일관된 관점을 준다.

병렬처리층은 의뢰인 및 작업자의 관리를 수행하며 특히, 의뢰인으로부터 요청이 들어 온 총 작업량을 현재 등록된 여러 작업자들에게 배분하는 작업을 수행한다. 이미 구현이 완성된 일반 클래스도 존재하지만, 상위 응용층의 클래스에서 상속을 하고, 추상 메소드의 구현을 완성하여 이용하는 추상클래스 및 인터페이스도 존재한다. 병렬처리층은 의뢰인부분, 작업자부분 및 서버부분으로 나뉜다. 특히, 작업자부분에서는 자신의 벤치마크를 수행하는 클래스 및 작업마다의 알고리즘 연산에 필요한 여러 메소드를 선언 및 구현한 클래스도 존재한다.

응용층에서는 병렬 처리층을 이용하여 성능향상을 얻으려는 문제들마다 다르게 구현되며, 그림 2에서 보이는 바와 같이 확장(extend) 및 구현(implement) 방침을 그대로 따라야 한다

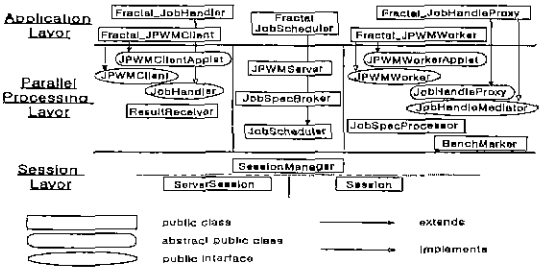

그림 2 시스템 설계 및 클래스 구성도

3. 시스템 구현

3.1 원격 참조값과 RMI

본 시스템은 RMI를 이용하여 의뢰인이 자신의 원격 참조값을 서버에 등록한다. 한편, 작업자는 등록된 의뢰인 참조값을 서버로부터 가져간다. 그러므로, 서버측의 클래스는 Remote인터페이스를 상속하는 ClientRefManager 인터페이스를 구현해야 하며, ClientRefManager에는 원격 참조값을 등록하고, 수취할 수 있는 메소드가 선언된다.

한편, Remote를 확장하는 또 다른 RemoteResultReceiver 인터페이스에 작업자가 결과를 보낼 때 사용하는 메소드가 선언되며, 의뢰인의 클래스중 ResultReceiver가 이 메소드들을 구현한다.

3.2 성능 기반 작업 분배 스케줄링

인터넷상의 호스트는 모두 다른 성능을 보여준다. 그러므로, 그러한 성능을 고려하여 작업을 분배하여, 의뢰인에게 빠른 응답시간을 보여주어야 한다

본 시스템은 작업자들이 각 호스트로 로드되어진 후, 서버에게 자신을 등록할 때, 스스로 벤치마크 테스트를 수행하여 그 결과를 서버에게 보내준다. 각 작업자에게 수행시키는 문제는 행렬을 이용한 선형방정식(Ax = b) 이다. 이와 같은 방법은 수년동안 컴퓨터의 연산능력을 평가했던 고전적 방법이다[7]. 이러한 계산에 대한 결과로 초당 몇백만 번의 부동소수점 연산을 수행하는 지에 대한 Mflops/s 단위값을 얻어내어 서버에게 전한다

서버는 각 작업자들이 보내 온 결과를 유지하다가, 태스크를 분배할 때에, 현재 태스크의 총 수를 각 작업자가 돌려준 성능비에 따라 나누어 보낸다. 이렇게 하여, 각 작업자들의 능력만큼 태스크를 분배하게 되므로, 이론적으로 각 작업자에게 보내준 태스크들에 대한 결과가 의뢰인에게 동시에 보내지게 된다. 이 방법은 의뢰인에게 빠른 응답시간을 보여줄 수 있다.

4. 성능 평가

그림 3 은 본 시스템에 프랙탈 알고리즘 처리를 수행시켜 의뢰인의 애플릿에서 결과를 작업자들로부터 받는 도중 및 최종 결과를 받은 그림을 보여준다. 프랙탈 알고리즘은 간단한 복소변환($z = z^2 + c$)의 규칙이 복잡한 구조를 만들어 내는 멘델브로트(Mandelbrot) 방법을 사용한다.

총 256 라인에 각 라인 당 256 픽셀을 할당하며, 임의의 연속된 여러 라인 묶음을 한 태스크로 정하며, 몇 라인을 하나의 태스크로 묶는

지에 따라 태스크의 크기가 결정된다. 그림 3에서는 2라인을 하나의 태스크의 크기로 정했을 때의 모습이다.

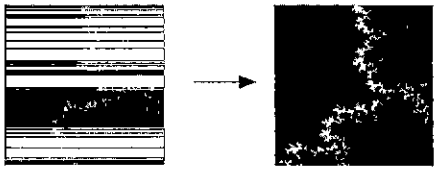

그림 3. 프랙탈 작업 수행 모습

첫 번째 성능 평가는 작업에 참여하는 작업자의 수에 따른 성능 향상을 실험한다. 이 결과는 [4, 5]에서 수행한 결과와 비슷하다. 즉, 작업자의 수가 늘어날수록 늘어난 비율과 비슷하게 성능이 향상되다가, 어느 정도 작업자의 수가 많아지면, 그 수가 늘어난 비율보다는 작은 비율로 성능이 향상된다. 성능 평가 그래프는 [4, 5]에서 제시된 것과 비슷하다.

두 번째 성능 평가는 작업자의 수는 4대로 고정하고, 태스크의 크기에 따른 성능 향상 비교를 행한다. 태스크의 크기에 따라 총 6가지 경우를 나누며, 수행 연산량이 많은 순서대로 작업 명세 1부터 5까지 (Spec1 ~ Spec5) 설정한다. WWW상의 컴퓨터는 그 수행 성능이 모두 다르므로, 작업자를 위한 호스트 4대의 성능은 모두 다르게 설정한다. 균등 배분법에서는 모든 태스크가 동일한 수로 나뉘어 배분되며, 성능 기반 배분법에서는 총 태스크의 수를 각 작업자의 성능 비로 나뉘어 배분된다. 배분된 태스크는 태스크마다 독립적으로 쓰레드로 생성되어 작업을 수행하며, 그 결과는 만들어진 즉시 의뢰인에게 전달된다. 표 1에 각 작업자에게 할당되는 태스크 수를 보여준다.

표 1 각 작업자에게 할당되는 태스크 수

작업자 ID	성능 비	태스크 수(균등|성능 기반)					
		2 Line	4 Line	8 Line	16 Line	32 Line	64 Line
1	1.00	32|18	16|9	8|5	4|2	2|1	1|0
2	1.76	32|31	16|15	8|7	4|4	2|2	1|1
3	1.80	32|32	16|16	8|8	4|4	2|2	1|1
4	2.67	32|47	16|24	8|12	4|6	2|3	1|2

성능 평가를 위하여 의뢰인과 작업자들은 Pentium 200과 Ultra SPARC 10 에서 커뮤니케이터 4.05이상을 이용하여 수행시키며, 서버는 독립적인 Ultra SPARC 10을 사용한다.

그림 4는 균등 배분 방법을 사용했을 때, 그림 5는 성능 기반 배분 방법을 사용했을 때의 각 작업 명세당 수행 시간을 그래프이다. 그림 4, 5에서, 작업자 4대의 성능 비가 약 0.6의 표준편차를 보일 때, 균등 배분 방법보다 성능 기반 배분 방법이 25.3% 성능 향상을 보임을 알 수 있다. 또한 계산량이 상대적으로 많은 작업 명세 5에서는 40.5%의 높은 성능 향상을 보여준다.

한편, 태스크의 크기(라인 수)가 증가될수록 더 좋은 성능을 보인다 그러므로, 태스크의 크기를 작게 하여 각 작업자마다 많은 수의 태스크를 할당하는 정책보다 적은 수의 큰 크기의 태스크를 할당하는 정책이 더 좋은 성능을 보임을 알 수 있다

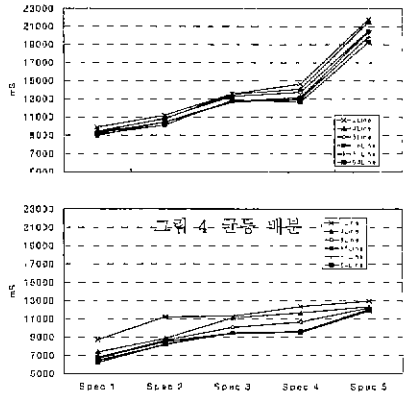

그림 5 성능 기반 배분

5. 결론 및 향후과제

본 논문에서는 WWW에서의 자바 병렬처리 머신에 대한 설계와 구현방법을 제시하며, 성능 향상을 위해 자바의 RMI을 이용한 애플릿간 직접 통신 메커니즘을 구현하여 작업자의 결과를 의뢰인에게 직접 보내는 방법을 제안한다. 또한, 병렬 처리 서버의 작업 태스크를 각 작업자들의 성능비에 따라 다르게 할당하는 방법을 제시하여, 실제로 웹 환경에서 프랙탈 이미지 처리 작업을 적용했을 때, 균등 분배법보다 성능 기반 배분법이 더 좋은 성능을 나타냄을 보였다.

작업자들의 성능이 계속 변하는 환경에 효과적으로 대처하는 동적 작업 분배방법 및 작업자의 결함(fault)으로 인한 여러 상황에 대처할 수 있는 방법이 연구중이다. 또한, 의뢰인과 작업자들 사이에 교환되는 정보가 보안상 중요한 내용이 있을 때, 그 정보를 보호할 수 있는 메커니즘도 연구대상이다.

[참고 문헌]

[1] K. M Chandy, B Dimitrov, H Le, J Mandleson, M Richardson, A Rifkin, P A G. Sivilotti, W. Tanaka, and L. Weisman, "A World-Wide Distributed System Using Java and the Internet," In the *Proceedings of the Fifth IEEE International Symposium on High Performance Distributed Computing*, Syracuse, NY, August 1996

[2] A. Baratloo, M. Karaul, H. Karl and Z M Kedem, "An Infrastructure for Network Computing with Java Applets," *the ACM Workshop on Java for High-Performance Network Computing*, 1998

[3] Tim Brecht, Harjinder Sandhu, Meijuan Shan, and Jimmy Talbot, "ParaWeb Towards World-Wide Supercomputing," In *the Proceedings of the Seventh ACM SIGOPS European Workshop*, pp 181-188, September, 1996

[4] Bernd O. Christiansen, Peter C Japello, Mithai F. Ionescu Michael O. Neary, Klaus E Shauser, and Dameal Wu. "Javelin. Internet-based parallel computing using java," *the ACM Workshop on Java for Science and Engineering Computation*, 1997.

[5] A Alexandrov, M Ibel, K. E. Schauser, and C. Scheiman, "SuperWeb Research Issues in Java-Based Global Computing," *the Concurrency' Practice and Experience*, June 1997

[6] Sun Microsystems, Inc., "Java Remote Method Invocation Specification," October 1997, Revision 1.42.

[7] J Dongarra, Performance of Varios Computers Using Standard Linear Equations Software, http://www.netlib.org/benchmark/performance.ps, August 1998