

PDSWeb: Intranet에서 분산 병렬 처리 스킴의 성능평가

송은하, 정영식
원광대학교 컴퓨터 및 정보통신공학부

PDSWeb: Performance Evaluation of Distributed Parallel Processing Scheme on the Intranet

Eun-ha Song, Young-Sik Jeong

School of Computer and Communication Engineering, Wonkwang University
(ehsong, ysjeong)@wonkwang.ac.kr

요 약

Intranet 환경에 다수의 유휴 호스트를 이용하여 많은 계산량을 지닌 응용 문제를 분산시켜 병렬 수행함으로써 효율성의 향상이 기대된다. 하지만 유휴 호스트를 이용하는데 있어서 이질성과 가변성 및 자율성으로 인해 요청에 대한 신뢰성을 예측하기가 어렵다. 본 논문에서는 태스크 할당 및 호스트 관리에 있어서 부하 균등을 위한 동적 분산 병렬 스킴인 적응적 태스크 재할당 기법을 제시한다. 또한, 이미지 랜더링 생성과 프래탈 이미지 처리와 같은 많은 연산량을 지닌 응용 문제를 PDSWeb 시스템에 적용하여 제안 알고리즘의 성능을 분석하고 평가한다.

1. 서론

Intranet에서 분산 병렬 컴퓨팅에 대한 시도는 적은 비용으로 유휴 상태(idle state)에 있는 컴퓨팅 자원의 활용도(utilization)를 높이는 데 있다. 그러나, 이러한 컴퓨팅은 몇 가지 어려움이 있다[1,5]. 호스트들의 이질성(heterogeneity)으로 인한 각각의 플랫폼에 맞는 컴파일링 작업 추가와 호스트들의 지속적인 수행 능력을 예측할 수 없다. 또한, 각기 나름대로 수행 목적이 다르므로 갑작스런 환경 변화 및 과부하로 성능비가 극심하다. 뿐만 아니라, intranet 환경은 호스트들의 결함 및 추가에 따른 상태 변화와 자율성을 가진다. 더욱이 분산 병렬 처리를 위한 기존 연구들은 대부분 어플리케이션 객체와 병렬 처리 객체와의 상호 의존성이 깊다[3,4]. 새로운 어플리케이션을 구축되어 있는 분산 병렬 처리 시스템에 적용하기 위해서는 작업 분할 정책 이외에도 독립적인 어플리케이션 개발을 위한 객체 지향적 설계가 필요하게 된다[2].

따라서, 앞에서 언급한 intranet 컴퓨팅의 고려사항

에 대해 적응성을 추가하여 분산 병렬 처리를 위한 태스크 할당과 호스트 관리 기법이 요구된다[7].

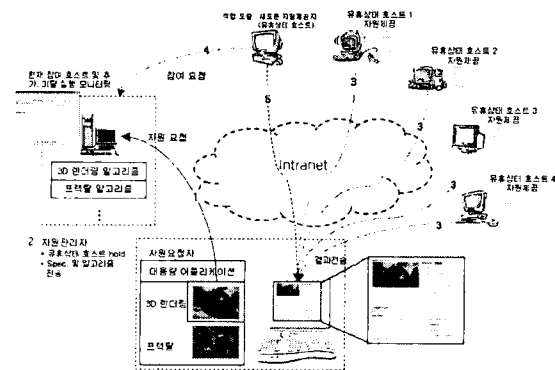


그림 1. PDSWeb 전체 구성도

그림 1은 본 논문에서 제시하는 시스템의 전체 구조이다. 특정 작업 수행에 자원을 요구하는 자원 요청자(Resource Requester, 이하 RR), 연산에 자원을 제공하는 자원 제공자(Resource Provider, 이하 RP), 자원 요청자와 자원 제공자 사이의 원활한 작업 전달과

본 연구는 한국과학재단 목적기초 연구과제(과제번호:2001-1-30300-011-2) 연구비에 의해 연구되었음

이들을 관리하는 자원 관리자(Resource Manager, 이하 RM)로 기본 구성한다. 추가로, 작업 처리율이 가변성 있는 자원 제공자를 관리하기 위한 가상 관리자(Virtual Manager, 이하 VM)와 특정 어플리케이션과 시스템과의 연결을 담당하는 어플리케이션 중계자(Application Proxy)를 둔다.

RR의 의하여 계산된 작업 결과는 자바가 지원하는 원격 메소드 호출(Remote Method Invocation, 이하 JavaRMI)에 의해 원격 객체 참조 값을 이용하여 콜백(callback) 처리한다. 이는 RM을 거치지 않고 즉시 RR에게 전달됨으로써 이중으로 들어가는 통신 시간을 줄인다.

본 논문에서는 PDSWeb(Parallel Distributed Scheme on Web)을 구현하여 랜더링 이미지 생성과 프랙탈 이미지 처리(Fractal Image Processing)를 적용으로 제안 기법의 성능을 평가한다.

2. 동적 분산 병렬 처리 스킴

Intranet 컴퓨팅에서는 호스트들이 여러 사용자에게 개방되어 있으므로 성능 예측이 불가능하다. 즉, 사용자의 이용 정도에 따라 호스트에 상주하는 내부 호스트 수의 증감으로 성능 변화의 원인이 될 수 있다[8]. 따라서, 단순히 작업을 시작하는 시점에서 호스트 성능만으로 시간의 흐름에 따라 변하는 수행 능력에 대해 능동적일 수 없다. 그러므로, 수시로 변하는 성능을 적응적 태스크 재할당(adaptive task reallocation) 기법으로 해결한다.

제안하는 적응적 태스크 재할당의 기본 메커니즘은 다음과 같다. 초기 벤치마크에 의해 성능비대로 정적 참여 호스트에게 작업을 분배한다[6]. 작업 도중에 임의의 RP에 의해 작업 완료 메시지를 받게 되면, 대기 상태에 놓인다. RM은 가장 낮은 작업 처리율에 있는 호스트를 찾아 연산을 정지시킨다. 해당 RP의 미수행된 작업량을 기준으로 재할당한다. 이는 작업 수행 상태에 따른 추가 연산으로 낮은 작업 처리율을 가진 RP의 태스크 일부를 대신함을 의미한다. 따라서, 일부 호스트의 과부하로 인한 연산 지연이 제거되며, 참여 호스트가 동일한 연산 시간을 가진다.

네트워크로 연결된 호스트들은 자율성을 지닌다. 연산 연결 오류나 호스트의 내부 결함으로 예고없이 더 이상 자원을 제공할 수 없어 전체 컴퓨팅의 100% 완성률을 보장할 수 없다. 이에 반해, 새로운 자원을 현재 진행중인 작업에 추가가 가능하다면, 추가 시점으로부터 보다 높은 컴퓨팅 파워로 전체 수행 시간이

감소한다. 적응적 태스크 재할당을 기반으로 호스트 수의 변화에 대해 동적으로 분산 병렬 처리한다.

그림 2는 이미 할당된 작업을 마쳐 대기 호스트가 존재하며, 작업 도중 호스트의 이탈이 발생한 경우를 가정한다.

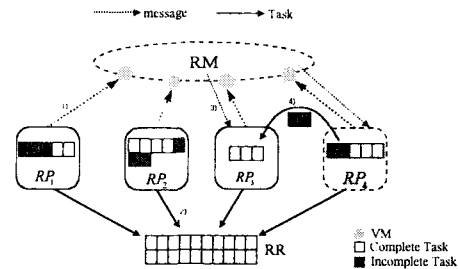


그림 2. 대기 호스트 존재시 이탈

RP4 자신의 의도에 의한 이탈로 완료 메시지와 연산 정보 반환하고 쓰레드를 종료한 후 시스템을 빠져 나온다. 이탈 응답을 받는 RM은 이탈 호스트 연산 정보로 미수행된 태스크를 추출하여 대기 호스트에 의해 추가 연산하는 결합내성을 지원한다.

만약, 해당 RP로부터 연산 정보를 반환하지 않고 완료 메시지만을 전송 받은 경우, 이탈한 호스트는 VM의 모니터링 정보로 미수행 태스크를 추출한다. 뿐만 아니라, 갑작스런 시스템과 네트워크의 오류로 인하여 사전 예고없는 호스트 이탈을 고려한다. 호스트의 연결 상태 및 연산 정보를 저장하는 VM를 시스템에 참여한다. RP의 참여 의사 전달 시점에 통신을 위한 세션과 쓰레드 형태로 생성한다. 통신 제어권을 VM에게 넘겨줘 RP와 RM간의 중계자 역할을 한다. 그러므로, 세션 상태를 모니터링하면서 호스트 결합 여부를 관측한다. 동시에 VM에게 처리된 작업 식별자를 전달로 결합 발생에 대해 미수행 작업 정보를 추출한다.

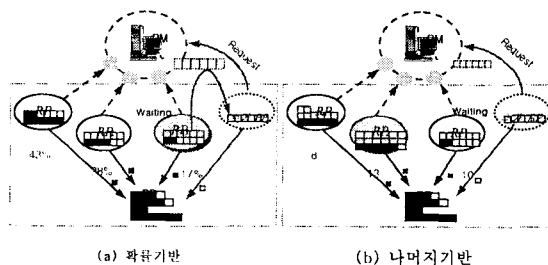


그림 3. 호스트 추가 기법

새로운 자원의 추가 메시지를 받았지만, 미작업 엔

트리 스택이 비어있는 경우로 모든 호스트가 연산 중에 RP의 추가 발생이다. VM에 의해 가장 낮은 성능으로 전체 연산 시간을 지연하는 RP를 추출한다. 추출 방법으로 그림 3과 같이 진행 상태의 작업률(task ratio)을 기준으로 하는 확률 기반(probability-based)과 아직 처리해야 할 작업수(task size)에 의한 나머지 기반(remain-based) 동적 재할당으로 구분한다.

그림 3의 (a)는 새로운 호스트 요청시 현재 RP들이 43%, 28%, 17%의 작업 처리율을 보인다. 이중 가장 낮은 처리율을 가진 RP₃를 재할당 대상으로 추출하여 미수행 태스크 일부를 추가 RP에게 이행한다. 하지만, 이 전략은 잘못된 호스트의 추가는 한쪽에 집중적인 재할당 연산으로 병목 현상(bottleneck)이 발생한다. 이러한 병목 현상의 단점을 고려하여 미수행 태스크 수로 호스트를 선정하는 나머지 기반 동적 재할당으로 해결한다. 3개의 RP는 각각 남은 태스크 수가 8, 13, 10으로 RP₂의 태스크 일부를 추가 RP에게 할당함으로써 한 호스트에 절대 영향을 미치는 부분이 제거된다.

즉, 본 논문에서는 수시로 변환 수 있는 호스트에 대해 동적 관리한다. RP를 관리하는 각각의 VM들을 임의의 기 값에 의한 벡터 형태로 관리하여 동적인 호스트 추가 및 삭제에 대한 RM의 과부하를 최대한 줄인다.

3. PDSWeb 시스템

3.1 PDSWeb 구조 및 통신 프로토콜

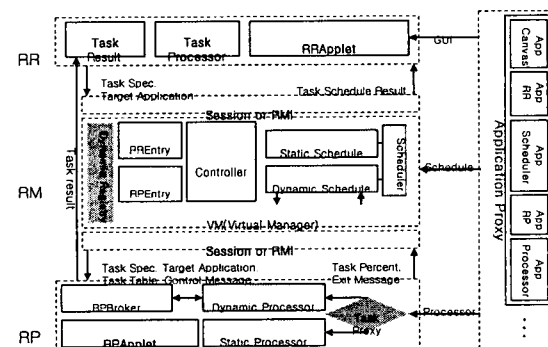


그림 4. PDSWeb 구조

본 논문에서 제안하는 시스템의 가상 상위 패키지는 PDSWeb이라 정의하며, 각 구성요소간의 상호 관계는 그림 4이다.

구성요소들의 패키지는 ResReq, Manager, ResPro, 그리고 어플리케이션을 위한 App로 나뉜다. 패키지들

간의 통신은 하부 객체인 세션과 JavaRMI를 통하여 메시지 전달 및 원격 메소드 호출을 사용한다.

PDSWeb 시스템에 초기 등록에서부터 모든 작업이 완료되기까지의 각 패키지간의 메시지 전달 프로토콜은 그림 5이다.

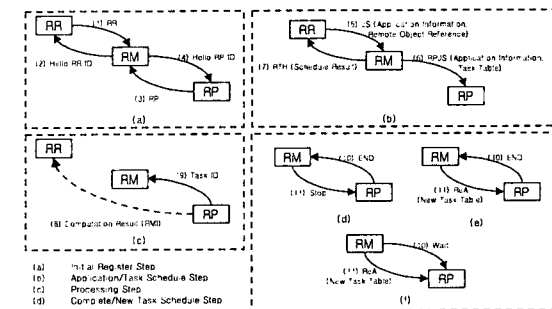


그림 5. PDSWeb 통신 프로토콜

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

3.2 PDSWeb 클래스

PDSWeb 시스템은 순수 자바코드를 사용하여 구현한다. RR과 RP는 애플릿 환경에서 사용하며 웹 브라우저를 사용한다. PDSWeb 시스템의 클래스는 모두 7개의 기본 패키지와 적용 어플리케이션의 수에 따라 하나의 패키지씩 증가한다. 다음은 각 패키지의 주요 클래스이다.

- Config 패키지
ConfigEnv : PDSWeb 시스템의 환경 정보를 저장하고, 모든 클래스들이 공통으로 참조
- Manager 패키지
PDSWebMonitor : 연산 전략 설정과 컴퓨팅 정보

를 알려주는 인터페이스

PDSWebManager : RM의 초기 구동 및 RR, RP의 등록

SchedulerRP : 특정 어플리케이션의 할당 알고리즘을 정의

DynamicScheduler : 적응적 태스크 재할당과 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 태스크 재할당과 호스트 관리 전략 연산

BenchMarker : RP의 성능 평가

RPBroker : 동적 연산을 수행시 RM과의 메시지 전달에 대한 중재자 역할

RPApplet : RP 인터페이스

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

4. 성능분석

제한한 태스크 할당 및 동적 분산 병렬 처리 스킴에 대한 성능 평가 결과이다. 적용된 어플리케이션으로 렌더링 이미지 생성과 프랙탈 이미지를 처리한다. 이 시스템에 참여 호스트의 구성은 플랫폼의 이질성을 고려하여 표 1과 같다.

4.1 호스트 수의 증가 및 성능변화에 따른 성능 평가

그림 6은 외부의 영향이 거의 없다고 가정하고, 동일 호스트와 표 1과 같은 이질 호스트에 의한 이미지 렌더링 생성 결과이다. 동일 CPU의 경우 호스트 수

표 1. 성능 평가에 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

에 비해 스피드업이 비교적 선형적이다. 이질 호스트에 대해 동일하게 태스크를 할당한 단순 할당 (simple), 간단한 평가에 의해 성능비대로 할당하는 성능 기반 할당(static), 가변적인 환경에 재할당을 추가한 적응적 할당(adaptive)을 비교한다. 최악의 호스트(RP6)를 추가하면 단순 할당의 경우 스피드업이 2를 넘지 못하는 매우 낮은 효율성을 보이는 반면에 제안한 분산 병렬 처리 스킴은 스피드업이 8.2에 해당하는 매우 높은 효율성을 보인다.

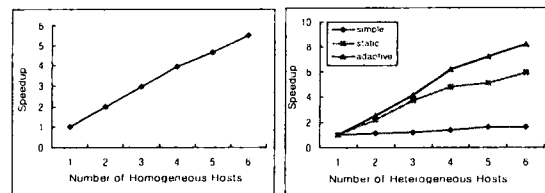


그림 6. 호스트 수 증가에 따른 성능 향상

그림 7은 최악의 호스트를 제외하고 호스트 수의 증가에 따라 프랙탈 이미지 처리와 이미지 렌더링 결과이다.

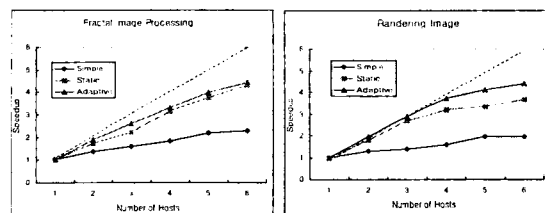


그림 7. 특정 어플리케이션의 호스트 수 증가에 따른 성능 향상

각 호스트간의 성능 비율에 맞게 정적으로 태스크를 할당하여 동일한 시간동안 일을 할 것을 기대하지만, 호스트 수행 목적에 따른 가변성을 예측하기 어렵다. 그림 8은 4대의 호스트로 고정하고 성능이 시간에 따라 변한다는 특성을 반영한 결과이다. 성능변화에

적응력을 추가한 처리 스킴이 성능기반 할당에 비해 재할당 연산에 소요되는 시간을 고려할지라도 높은 효율성을 보인다.

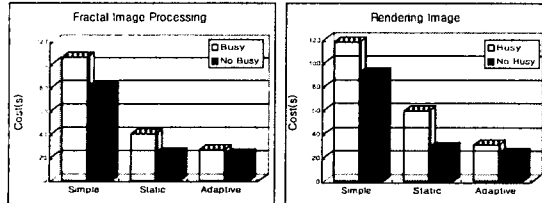


그림 8. 가변성을 고려한 성능 평가

4.2 Intranet의 자율성을 고려한 성능 평가

그림 9는 네트워크로 연결된 호스트들의 동적 작업 참여를 고려하여 연산도중 추가와 이탈을 고려한 성능 분석이다. 1대의 호스트에 의한 연산도중 3대의 호스트가 추가되어 컴퓨팅 파워가 증가한다. 미수행 태스크를 나누어 수행하여 전체 반환 시간이 감소한다. 특히 나머지 기반이 확률 기반 동적 재할당이 가지는 병목 현상을 제거하여 연산 시간이 줄었다. 이탈과 결합에 의한 호스트 감소로 전체 수행시간은 증가할지라도 작업이 정상 종료하는 신뢰성과 결합내성을 지원한다.

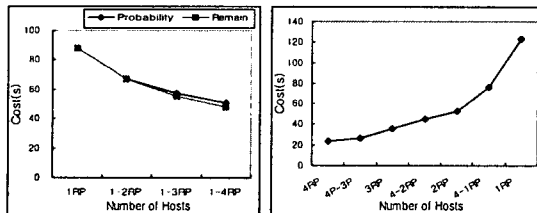


그림 9. 호스트 추가 및 삭제에 따른 성능 평가

5. 결과

대형 응용 문제를 분산 병렬 수행하기에 필요한 요소들과 해결해야 할 문제들에 대하여 알아보았다. 이에 대한 대처 방안으로 동적 분산 병렬 처리 스킴을 설계하여 성능 평가 및 검증하였다.

Intranet에 연결되어 환경 변화에 노출되어 있는 유향 자원을 병렬 수행에 이용하기 위해 환경의 적응성을 고려한 태스크 할당 기법을 제시하였다. 또한, 자원의 동적 추가 및 삭제를 고려한 호스트 관리로 전체 수행시간의 효율성을 증대 및 결합내성을 지원한다. 또한 다양한 어플리케이션을 적용하여 제시한 전

략의 타당성을 보였다. 특히 다양한 어플리케이션 적용을 위해 의존적인 부분을 병렬 처리 계층과 분리하여 어플리케이션에 관련된 객체들은 필요시에 생성하는 객체 생성 메커니즘을 이용하여 독립적인 어플리케이션 개발을 지원한다.

향후 연구 과제로서는 본 논문에서 지원한 어플리케이션에 독립된 특징을 계속해서 이어 이를 기반으로 한 다양한 어플리케이션의 개발함으로써 어플리케이션의 서로 다른 특성에 따른 성능을 평가 비교 분석하여 어플리케이션 특성에 의존하지 않는 시스템 개발을 위한 전략적 연구가 필요하다.

그리고, 본 논문에서는 고려하지 않았던 서버 태스크간에 상호 연관성이 있는 연산에 대하여 제공자간에 메시지 패싱(message passing)기법을 적용함으로 해결해 나아갈 수 있다.

[참고문헌]

- [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] P. Michael, and Z. Matthies, "JavaParty : Transparent Remote Objects in Java," In the ACM Workshop on Java for Science and Engineering Computation, 1997
- [3] B. O. Christiansen, P. Cappello, M. F. Ionescu, M. O. Neary, and K. E. Schauer, "Javelin : Internet-Based Parallel Computing Using Java," ACM Workshop on Java for Science and Engineering Computation, 1997.
- [4] 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.
- [5] D. Caromel, W. Kauser, and J. Vayssiere, "Java// : Towards Seamless Computing end Metacomputing in Java", Concurrency Practice and Experience, pp 1043-1061, 1998
- [6] J. Dongarra, J. Bunch, D. Moler, and G. W. Stewart, "LINPACK User's Guide," SIAM, Philadelphia, 1979.
- [7] T.s Hsu, J.C.IEE, D.R Lopez, "Task Allocation on a Network of Processor", IEEE Trans. on

Computer, Vol. 49, No. 12, 2000.

- [8] 신필섭, 김신덕, "자바 인터넷 컴퓨팅 환경을 위한 효율적인 자원 관리 시스템의 설계 및 분석," 한국정보처리학회 추계 학술발표논문집, 제6권 제2호, pp 155-162, 1999.