웹 환경에서 동적 작업 프로세서 관리를 위한 WebSM 개발

정 권호, 송 은하, 정 영식
원광대학교 컴퓨터 및 정보통신 공학부

Development the WebSM for Dynamic Task Processor Management on Web

Kwen-Ho Jung, Eun-Ha Song, Young-Sik Jeong
School of Computer · Communication Engineering, Wonkwang University

요 약

웹 환경의 많은 유휴상태 호스트를 이용하여 작업이 병렬 처리가 가능하다. 그러나, 이로 인해 발생하는 호스트들의 가용성 및 웹의 가변성을 예측하기는 힘들다. 본 논문에서 제시한 시스템 WebSM은 작업에 대한 효율적인 병렬 처리를 위한 적응적 작업 할당 기법을 제시한다. 또한 호스트들의 자유로운 연산 참여에 따른 작업 재할당 기법과 결함 및 삭제로 인해 미수행된 일부분의 작업 처리가 가능하도록 동적 작업 프로세서 관리 스킴을 제시한다.

1. 서론

웹을 하나의 거대한 가상 시스템으로 볼 때, 이들 안에 존재하는 많은 컴퓨팅 리소스들은 이질성을 지닌다. 하지만, 다량의 이질적인 컴퓨팅 리소스를 이용하여 병렬처리 함으로써 방대한 연산을 요구하는 애플리케이션 수행이 가능하다[6]. 뿐만 아니라, 임의의 애플리케이션에 대해 여러 개의 프로세서가 작업을 나눠서 수행함으로써 전체 수행시간을 줄일 수 있는 효과가 있다.

동일한 CPU 능력을 가진 P대의 프로세서로 처리한 실행시간을 $T(P)$라 정의했을 때, 스피드업(speedup)은 $S(P) = T(1)/T(P)$가 된다. 그러므로, 다수의 프로세서가 애플리케이션 연산에 참여함으로써 높은 효과를 얻는다[5].

웹 기술을 바탕으로 애플리케이션을 병렬처리 하기 위한 특별한 프로그래밍 모델을 제시한 기존 연구로는 ATLAS[1], POPCORN[2], JET[3], Javelin[4] 등이 있다. 하지만, 이들 대부분은 로컬 네트워크 작업 환경에서 수행되었으며, 단순한 작업 할당 알고리즘만 제시하였을 뿐 확장된 웹 환경이 미치는 가변적인 요인에 효과적으로 대처하기 위한 방안을 제공하지 못한다. 특히 웹에 참여한 이질적인 호스트들은 지리적으로 떨어져 있으며 사용자에게 마저 개방되어 있으므로 전체 수행에 많은 영향을 받는다. 이에 따른 원인으로, 연산에 참여 중인 프로세서 가용성(availability)에 따른 성능 변화가 발생한다. 더욱이, 연산도중에 유휴상태(idle time) 호스트들의 삽입, 호스트들의 결함(failure) 및 임의의 종료를 예측할 수가 없다. 따라서, 본 논문에서는 참여한 프로세서 수행능력의 다양성과 프로세서의 삽입, 결함의 유동성에 적응할 수 있는 동적 프로세서 관리 스킴을 개발한다.

또한 본 논문에서 개발된 시스템인 WebSM의 활용으로 많은 계산을 요구하는 애플리케이션을 활용함으로써 제시된 동적 프로세서 관리 스킴 알고리즘의 타당성을 제시한다.

2. 동적 작업 프로세서 관리 스킴

앞에서도 언급했듯이, 호스트들의 성능 차이로 수행능력이 다르다. 또한 가용성에 따라 상태변화를 가져오며, 프로세서 수에 따라 전체 수행시간 결정하는 스피드업에 차이가 있다. 따라서 본 논문에서는 이에 대한 대처방안으로 적응적 작업 재할당 기법을 제시하며 이를 기반으로 동적 작업자 관리를 수행한다.

2.1 적응적 작업 할당

초기의 프로세서는 자원 매니저(Resource Manager : RM)에 요청하여 간단한 Linpack 벤치마크에 의해 성능비에 따라 작업 양을 할당받는다. 할당받은 작업 처리 상태 정보를 등록과 함께 생성되는 가상 관리자(Virtual Manager)에 기록되며, 웹의 물리적 거리 및 지역 애플리케이션의 과다 연산 성능 변화가 극심한 경우에 재할당(reallocation) 연산을 수행한다.

초기 프로세서의 성능비에 따라 할당받은 작업을 수행하는 도중에 임의의 작업 제공자(Resource Provider : RP)의 수행능력 저하로 모든 프로세서가 동일한 시간에 작업을 마치지 못한 경우이다. 이때, 이미 할당받은 작업을 마친 RP에 의해서 작업의 일부를 적응적 재할당 받아 연산이 이루어지는 과정을 보인다.

애플리케이션에 대한 작업 명세와 작업 시작 메시지를 전달받은 RP는 작업을 끝날 때까지 진행한다. 수행능력이 좋아 먼저 할당받은 연산을 마친 RP4는 RM에게 할당된 작업을 마침을 알리는 *Complete* 메시지를 전달하고 다음 작업이나 명령을 기다리게 된다. 성능기반으로 할당받은 작업량 중 미수행된 작업량이 큰 수행능력이 낮은 RP는 RM로부터 현재 진행되고 있는 연산을 일시 중지하고 재할당을 기다리기 위한 *Wait* 메시지를 받는다.

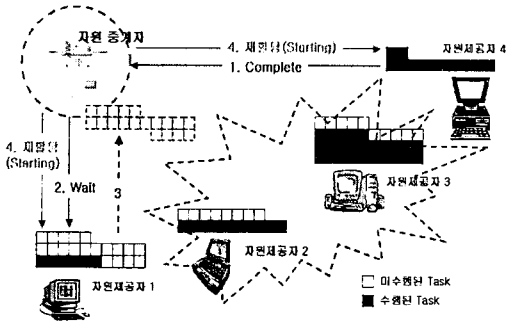

그림 1. 적응적 작업 할당 기법

그림 1은 *Complete* 메시지를 보낸 RP4는 *Wait* 메시지를 받은 RP1은 현재 수행능력에 따라 적응적으로 재할당 연산한다. 이때, 재할당 대상에 속하지 않는 RP2와 RP3는 RM의 연산과는 상관없이 자신의 작업을 계속해서 진행하게 됨으로써 재할당 연산으로 지연되는 수행시간을 최소한으로 줄이며, 전체 RP는 동일한 시간에 작업을 마친다.

2.2 동적 작업 프로세서 처리 스킴

연산 참여의 자율성을 가진 웹의 장점에서 단순히 고정된(static) 프로세서 즉, 초기 RM의 등록을 거친 프로세서에 대해 할당받은 작업의 처리시간에 일정한 한계를 가진다. 그러므로, 하나의 애플리케이션을 수행 도중에 프로세서 추가와 네트워크 및 시스템 내부 결함으로 프로세서 삭제와 임의의 종료를 대처하기 위한 방법이 요구된다.

동적 프로세서 처리 스킴으로 작업 재할당 방법에 대한 두 가지 관리 전략을 제시한다.

먼저 확률기반 동적 재할당 과정이다. 그림2와 같이 수행도중에 새로운 프로세서가 RM에게 연산 참여를 요청하거나, 할당받은 작업을 미쳐 수행하지 못하고 마친다고 가정하자. 일단, 물리적인 결함과 임의로 자신의 연산을 종료할 경우에는 아직 미수행된 작업은 삽입된 프로세서에 의해서 우선 처리한다. 특히, 이 스킴은 미수행된 작업이 발견되지 않은 경우에 삽입된 프로세서는 할당받은 작업 처리율(Processing Ratio)이 가장 낮은 프로세서가 가진 일부분의 작업을 재할당 받아 수행하는 방법이다. 하지만 이 방법은 동적 프로세서의 변화가 극심한 경우 어떤 하나의 프로세서를 작업에 대해 연속적인 재할당으로 병목현상이 발생하여 전체 수행시간이 증가된다. 또한 잦은 프로세서 추가시 어느 한쪽에만 집중적으로 재할당 연산이 일어난다.

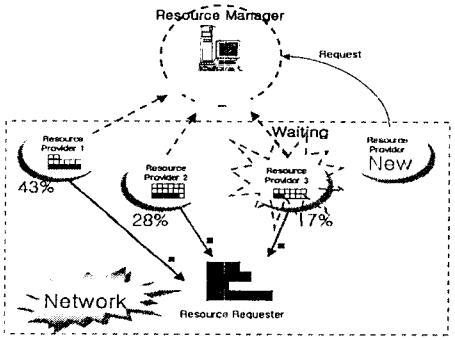

그림 2. 확률 기반 동적 재할당

두 번째 전략으로 나머지 기반 동적 재할당이다. 확률 기반 동적 재할당의 단점인 처리율이 낮은 프로세서의 작업에 대해 과다한 재할당으로 병목현상을 고려한다. 할당받은 작업 처리량(Processing Size)에 우선순위(priority)를 두어 가장 많은 양의 미수행된 프로세서의 작업을 삽입된 프로세서가 재할당 받아 연

산한다. 그림3의 3개의 프로세서는 각각 8, 13, 10개의 처리해야할 작업을 가지고 있다. 가장 미수행된 작업이 많은 RP2의 작업을 재할당 한다.

모든 RP가 할당받은 작업을 100%에 가깝게 끝마쳤을 때 새로운 프로세서가 추가 될 수도 있다. 이 경우는 재할당 연산시간을 감안하여 삽입된 프로세서를 연산에 추가시키기보다는 현재 진행상태인 프로세서의 의해 작업을 마치도록 한다.

마지막으로, 동적 프로세서 관리로 인해 연산이 진행되는 동안 RP의 수는 수시로 변한다. 따라서 본 논문에서는 RP를 관리하기 위한 각각의 가상 관리자들을 임의의 키 값에 의한 벡터 형태로 관리함으로써 동적 프로세서 추가 및 삭제에 대해 RM의 과부하를 최소화한다.

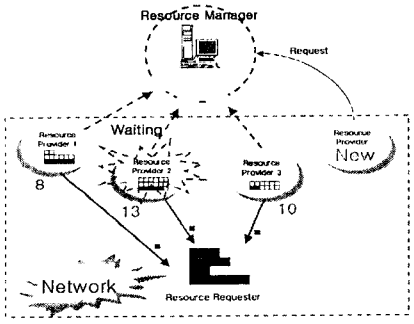

그림 3. 나머지 기반 동적 재할당

3. WebSM 개발

3.1 WebSM 설계

웹에 연결된 자원을 동적으로 이용하여 병렬 컴퓨팅이 가능한 WebSM 시스템은 특정한 작업을 수행하고자 많은 자원을 요구하는 자원 요청자(Resource Requester : RR), 자신의 자원을 제공하여 작업 수행에 도움을 제공하는 자원 제공자(Resource Provider : RP), RP와 RP 사이의 원활한 작업과 동적으로 변화하는 RP들에게 대치하는 자원 관리자(Resource Manager : RM), 연산도중 새로운 RP의 추가나 RP의 결함 및 임의의 종료 등 동적으로 변화하는 프로세서를 관리하는 RP 가상 관리자(RP Virtual Manager : RPVM), 애플리케이션들과 WebSM 시스템과의 연결을 담당하는 애플리케이션 중계자(Application Proxy : AP)로 구성된다. 그림4는 구성요소들의 관계이다.

WebSM 시스템 클래스 구조는 그림5이며, 구성요소에 따라 설정된 클래스들의 기능은 다음과 같다.

RR는 원하는 작업에 대한 GUI(Graphic User

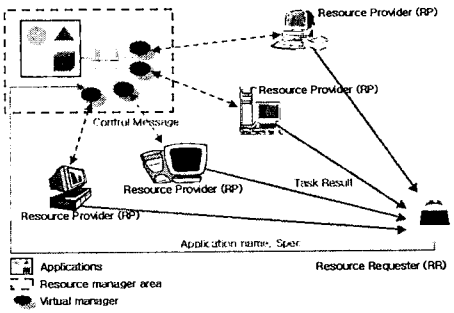

그림 4. WSM 시스템 구성도

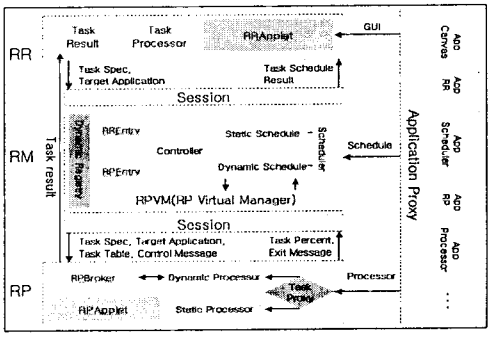

그림 5. WebSM 클래스 구조

Interface)를 애플리케이션으로부터 받아와 작업을 요청하기 위한 환경을 구축한 후, 다양한 작업 명세에서 연산을 원하는 애플리케이션과 작업 스케줄링 기법을 설정하고 RM에게 전달한다. 특히, 수행 결과를 RP로부터 직접 받기 위하여 자신의 원격 객체 참조값을 RM에게 전달한다.

RM은 RR측에서 설정한 애플리케이션이 AP에 의해 해당 작업 명세를 전달받아 이미 등록된 RP에게 성능 비에 따라 각각 작업을 분배하여 전달한다. 이때, 보유된 RP 수만큼 RPVM를 생성하여 수행 상태를 관리한다. RPVM는 연산도중 프로세스의 결함 및 종료시 재할당 연산에 필요한 정보를 제공한다. 또한 새로운 프로세서를 위한 동적 레지스트리(Dynamic Registry)를 두어 성능에 비례하여 도중 작업 참여가 가능하도록 한다.

단순한 정적(static) 프로세서와 동적(dynamic) 프로세서 스케줄링 기법으로 구분된 RP측은 RR의 작업 명세에 따라 호출되어 수행된다.

3.2 WebSM 구현

그림6은 성능에 차이가 있는 4대의 프로세서(P1 :

PentinumII 150MHz · 32MB, P2 : PentinumIII 500MHz · 128MB, P3 : PentinumII 233MHz · 64MB, P4 : PentinumII 233MHz · 32MB)에 의해 RM이 요청한 애플리케이션인 프랙탈 이미지를 처리한다. 이는에 연산을 참여하기 위해 등록하여 초기의 벤치마킹에 의해 성능비에 따라 할당받은 작업을 연산하는 실행화면이다.

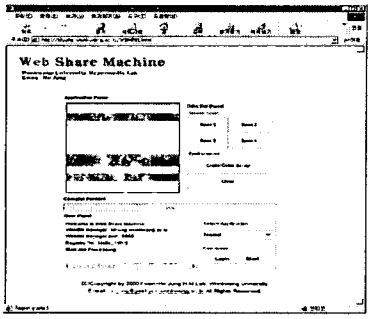

그림 6. 이질적인 4-프로세서 실행화면

연산도중 새로운 프로세서의 추가로 그림7은 5-프로세서에 의해 수행된다. 웹의 가변성에 의해 P2가 자신이 수행능력을 제대로 이행하지 못하고 있으며, 이때 추가된 새로운 프로세서에 의해 일부분의 작업이 재할당되어 수행되는 화면이다.

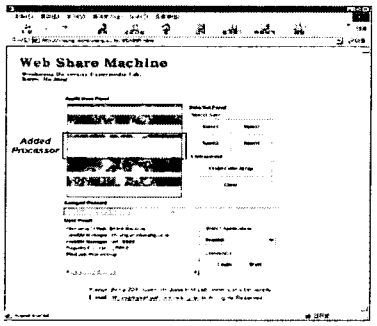

그림 7. 1-프로세서 추가로 재할당 수행화면

4. 결론

웹 환경의 많은 유휴상태의 시스템을 이용하여 거대 문제를 병렬 처리하는데 있어서 예측할 수 없는 무한한 가능성중 호스트의 가용성 및 삽입과 결함으로 인한 삭제 등 상태 변화의 대응책에 대해서 제시하였다.

구현된 WebSM은 성능기반으로 작업을 할당하여 참여 프로세서의 이질성을 극복하여 하였으며, 재할당 연산에 기반한 동적 작업 프로세서 관리 기법을 제안하여 연산도중 프로세서의 참여 가변성에 대해 적응성을 가진다. 또한 프로세서의 동적 참여 증가로 과다한 재할당이 이루어짐으로써 발생 가능한 병목현상을 방지하기 위한 프로세서 관리 기법을 제시한다.

[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, September 1996.

[2] N. Camiel, S. London, N. Nisan, O. Regev. "The POPCORN Project : Distributed computing over the Internet in Java," In Proc. 6th International World Wide Web Conference, April 1997.

[3]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, June 1997.

[4]B. O. Christiansen, P. Cappello, M. F. Ionescu, M. O. Neary, K. E. Schauser, D. Wu, "Javelin : Internet-Based Parallel Computing Using Java," ACM Workshop on Java for Science and Engineering Computation, June 1997.

[5]송은하, 정권호, 정영식, "WebImg 시스템 설계 및 구현," 한국정보기술학회 춘계학술발표논문집, pp. 511-525, 2000.

[6]정권호, 송은하, 정영식, "웹 환경에서 유연성 있는 작업 할당을 위한 가상 병렬 처리 시스템 개발," 한국멀티미디어학회논문지, 제3권 3호, pp.320-332, 2000.