그리드 환경에서 글로벌 작업 스케줄러의 설계

허대영[O*] 황선태* 정갑주**
*국민대학교 컴퓨터학부, **건국대학교 인터넷 미디어 공학부
{phinoccio[O*], sthwang*}@cs.kookmin.ac.kr {jeongk**}@konkuk.ac.kr

Design of Global Job Scheduler in Grid Environments

Daeyoung Heo[O*] Suntae Hwang* Karpjoo Jeong**
*Department of Computer Science, Kookmin University
**College of Information and Communication, Konkuk University

요 약

그리드 환경으로 구성된 다양한 컴퓨팅 자원을 효율적으로 이용하기 위해 글로벌 스케줄러의 필요성이 강조되고 있다. 하지만, 글로벌 스케줄러는 각 자원에 대한 영향력이 약해, 자원을 효율적으로 관리하는데, 작업 상태 파악, 자원 사이트의 균형 조절, 사용자의 자원 독점 등과 같은 문제점이 있다. 본 논문에서는 기존의 글로벌 스케줄러의 문제점을 기준으로 사용자의 수준의 정보를 기반으로 한 스케줄러의 설계를 통해 해결하고자 한다.

1. 서 론

컴퓨팅 능력을 향상시키기 위해 여러 대를 묶어 사용할 수 있는 클러스터에 대한 연구는 많이 진행되어 있다. 이러한 클러스터를 사용할 때, 대부분 작업 스케줄러를 사용하여 현재 사용 중이지 않은 유휴 컴퓨팅 자원에 작업을 분배한다. 최근 이의 개념을 넘어서 분산된 자원을 공유하여 사용하기 위한 그리드 컴퓨팅의 기술이 발전하고 있다. 그리드 역시 다양한 자원을 이용하기 때문에 컴퓨팅을 이용하고자 할 때, 적절히 유휴 컴퓨팅 자원에 작업을 분배할 수 있는 도구가 필요하다. 하지만, 그리드라는 특성 상, 컴퓨팅 자원의 구성요소들이 이기종이라고 보아야 하고, 컴퓨팅 자원마다 구축되어 있는 응용 환경이 다르다. 본 논문에서는 바이오/나노 연구자들이 사용할 수 있도록 만든 그리드 응용 플랫폼인 MGrid[1] 시스템에 적용할 글로벌 작업 스케줄러에 대해 제시한다.

2. 관련연구

작업 스케줄러는 다수의 사용자가 클러스터나 슈퍼컴퓨터와 같은 컴퓨팅 자원을 얻고자하고, 사용자에 비해 컴퓨팅 리소스가 적을 때, 즉 한꺼번에 사용할 수 없는 환경일 때, 필요하다. 기본적인 스케줄러의 형태는 일괄처리 형태이다. 이는 사용자의 작업을 들어오는 순서대로 큐에 넣고, 수행가능 할 때, 한꺼번에 수행하는 방법이다. 본 논문에서는 스케줄러의 용어를 구분할 것이다. 클러스터 환경의 작업 스케줄러를 지역 스케줄러라고 부르고, 그리드 환경에서 사용할 수 있는 스케줄러를 글로벌 스케줄러라고 하겠다.

2.1 Condor-G

Condor-G[2]는 지역 스케줄러인 Condor[2,3,4]의 기능을 확장한 것으로 Condor의 스케줄링의 알고리즘과 기법들을 그대로 사용하여 그리드 환경 중 글로버스 도구[2]에 맞게 확장된 형태의 작업 스케줄러이다. Condor-G는 Condor에서 사용한 작업 기술 방법을 확장해서 사용하며, Condor에서 확장될 수 있는 부가적인 기술들을 사용할 수 있다. 예를 들어 DAG를 처리할 수 있는 DAGMan을 Condor-G에서도 사용할 수 있게 되어 있다.

2.2 GRMS

Grid Resource Meta Scheduler[5]의 약자로써 GridLab에서 개발하는 GAT[6]의 일환이다. GRMS는 GAT에서 정의한 작업을 사용하여 자원 사이트 단위로 스케줄링하며 자원 사이트에서의 스케줄링은 지역 스케줄러가 담당하게 한다.

2.3 UNiCORE

UNiCORE[7,8]는 자원을 할당해주는 스케줄러가 없는 시스템으로 사용자가 모든 자원 노드에 대한 정보를 보고 이해 할 수 있도록 정보를 제공한다. 여기에 작업을 정의할 때, 자원에 대한 요구사항을 기술하여, 자원 선택을 할 때에 이를 활용할 수 있도록 설계, 구현되어 있다.

2.4 MGrid

MyJOB[9]은 바이오/나노 이론과학의 전산 모사를 그리드에서 수행할 수 있도록 설계된 MGrid[1] 시스템에서 사용하는 작업을 기술하는 것으로, 매우 다양한 표현이 가능하다. 단일 작업인 Task, DAG와 같이 순서가 있는 작업, 순서는 없지만 관계가 있는 작업의 그룹인 Job이 있고, 각 요소를 파라미터화하여 파리미터만을 변경하여 많은 전산 모사를 수행할 수 있도록 지원하고 있다.

3. 글로벌 작업 스케줄러에 대한 요구 사항

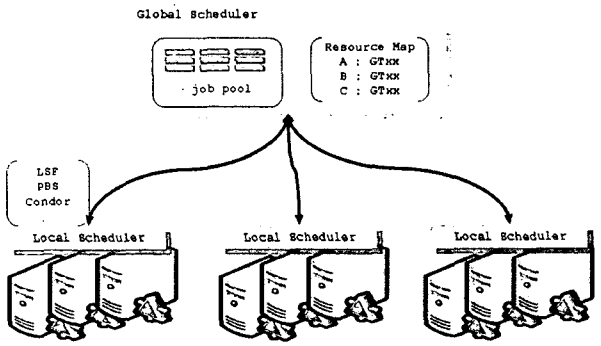
그리드 환경은 클러스터와 슈퍼컴퓨터보다 컴퓨팅 리소스가 훨씬 풍부하다. 하지만, 사용자가 이용할 수 있는 리소스가 많은 것이긴 하지만, 그 만큼의 사용자가 증가하고, 요구되는 컴퓨팅 능력도 더 증가한다. 즉, 그리드 환경에서도 사용자의 작업을 처리하기 위한 스케줄러가 필요하다.

하지만, 대부분의 동질성을 가지는 클러스터 환경과는 다르게 그리드 환경에서 각각의 컴퓨팅 자원은 매우 다양한 형태와 환경을 가진다. 그리드 환경에서는 클러스터들과 수퍼 컴퓨터가 동시에 존재할 수 있으며, 클러스터에 설치된 플랫폼이 제각각일 것이다.

기존에 사용하던 지역 스케줄러는 이러한 이질성이 높은 환경에서 사용하기에는 부적합한 점이 많다. 플랫폼의 차이가 가장 크며, 모든 컴퓨팅 리소스가 동일한 소프트웨어, 즉 같은 컴퓨팅 환경을 제공하지 않는다.

따라서, 글로벌 스케줄러는 그리드에 환경에 조성된 가상 그룹(Virtual Organization)의 컴퓨팅 자원의 환경을 알고 있어야 한다.

4. 기존 스케줄러의 문제점



[그림 1] 통상적인 글로벌 스케줄러

그리드 상에서 만들어져 제공되고 있는 스케줄러는 그렇게 많지 않다. Condor-G, GridLab에서 만든 GRMS(Grid Resource Meta Scheduler)가 있다. 이 스케줄러는 클러스터 환경에서 사용하던 Scheduler를 약간의 수정을 통해 그리드 상에서 실행할 수 있도록 수정한 것이다. 이러한 스케줄러는 모두 자원의 사용 유무가 기준이 되어 스케줄링을 수행한다.

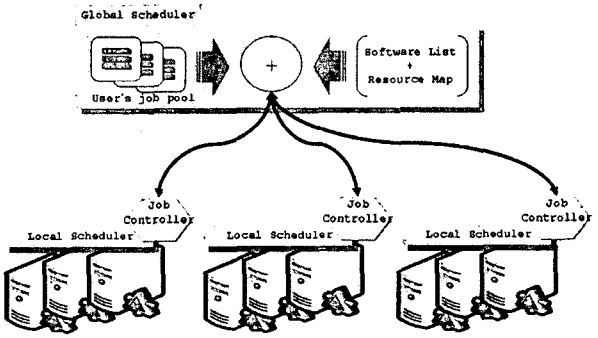
이들 스케줄러의 문제점은 크게 세 가지로 볼 수 있다. 첫째, 작업 상태 파악의 어려움이다. 실제 컴퓨팅 자원 사이트에서의 스케줄링은 지역 스케줄러에게 위임하는 방법을 사용한다. 이는 글로벌 스케줄러의 통제력, 영향력이 자원 사이트의 수준에서 약화되어 있음을 말한다.

두 번째 문제는 자원 사이트간의 부하 균형을 조절할 수 없다. 위에서 소개한 글로벌 스케줄러는 지역 사이트에서 수행될 수 있는 소프트웨어의 환경을 알고 있지 않다. 즉 각각의 자원사이트에서 어떠한 작업을 수행할 수 있는지 스케줄러는 알 지 못한다. 따라서 사용자가 글로벌 스케줄러를 사용하기 전에 자원사이트에서 수행할 수 있는 작업의 종류를 알아야 하며, 글로벌 스케줄러에 작업을 보낼 때에는 수행될 자원사이트를 지정해야만 한다. 예를 들어 같은 소프트웨어의 환경을 갖추고 있는 클러스터 군이 2개 존재하고, 한 쪽의 성능이 더 낮다고 가정할 때, 대다수의 사용자는 높은 성능을 내는 클러스터 군을 선호할 것이다. 이것은 높은 성능의 클러스터 군으로 사용자가 집중되게 만드는 데, 글로벌 스케줄러는 그 것을 알 수 있는 방법이 없다.

마지막으로 사용자의 자원 사이트를 독점하는 것을 막을 수 없다. 그리드 환경에서는 컴퓨팅 자원이 대폭 늘어났고, 그를 위한 가편한 사용 환경도 구축되어 질 때, 사용자가 컴퓨팅 자원을 점유하는 속도는 과거에 비해 매우 빨라진다. 기존의 글로벌 스케줄러는 이를 제어하지 않고 있다.

5. 사용자 정보 기반의 스케줄러 설계

본 논문은 위의 문제를 해결하기 위하 보다 사용자 수준으로 올려서 스케줄러를 설계하고자 한다. 위의 각 문제를 해결하기 위해, 기존의 그리드 환경에 다음의 컴포넌트를 추가하였다.



[그림 2] 새로운 스케줄러

첫 번째, 자원 사이트의 지역 스케줄러를 실시간으로 통제, 감시할 수 있는 컴포넌트이다. 이 컴포넌트는 글로벌 스케줄러로 부터의 작업 수행 명령을 가로채어 지역 스케줄러에게 보내는 데, 이 때, 감시를 위한 장치를 둘러싸서 보내게 된다. 이를 작업자(Worker)라고 부르고, 작업자는 사용자 작업 프로세스의 상태가 변할 때마다, 통제하는 컴포넌트에 보고하며, 이 컴포넌트는 그러한 메

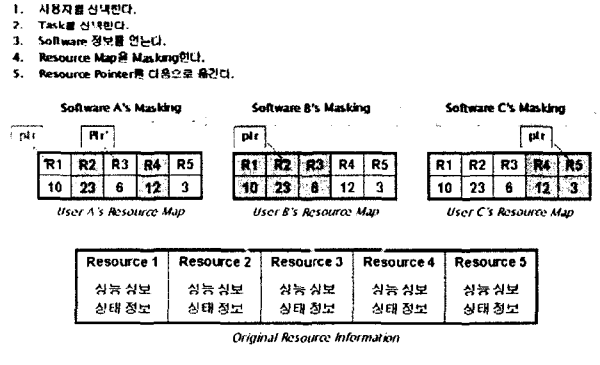
시지를 스케줄러에 전달하게 된다.

두 번째, 소프트웨어 환경을 알기 위한 컴포넌트로 정보 서비스라는 것을 추가하였다. 이 서비스는 항상 자원 사이트들의 성능과, 자원에서 수행 가능한 소프트웨어 환경에 대해 알고 있다. 소프트웨어 환경을 알기 위해서는 다음과 같은 방법을 사용한다. 앞서 말한 첫 번째 컴포넌트는 각 자원에서 수행 가능한 소프트웨어 목록을 감시한다. 감시를 통해 얻어진 것을 긴 주기를 가지고 정보 서비스로 보낸다.

세 번째, 사용자의 인증을 취급하여, 사용자 별로 따로 작업 풀을 가지도록하고, 사용자의 작업 풀을 경쟁시키도록 한다. 또한, 한 사용자가 각 자원 사이트에 보낼 수 있는 작업의 양을 제한하여, 독점을 막는다.

6. 스케줄러의 구현

본 논문에서 설계한 스케줄러는 [그림 3]에서 보는 것과 같이 사용자의 작업 풀과, 정보 서비스, 작업 통제 컴포넌트를 통해 수행한다. 사용자의 작업이 어떠한 소프트웨어를 사용하는지 파악하기 위해 본 논문에서는 MGrid의 MyJOB으로 기술할 것이다.



[그림 3] 스케줄러의 동작

사용자는 자신의 작업 풀에 작업을 전달한다. 스케줄러는 사용자의 과거에 사용했던 시간을 고려하여, 사용자의 작업 풀을 선택하고, 해당 작업 풀에서 수행해야 하는 작업을 전달받는다. 그리고 전달 받은 작업이 어떠한 소프트웨어를 사용하는지 파악하여 우선 작업 수행의 가부를 결정하게 된다. 가능 할 경우에, 스케줄러는 사용 가능한 자원 사이트의 목록을 작성한다. 만약 작업의 기술에 자원 사이트가 결정되었는지 확인하고, 결정되었을 경우는 해당 자원이 자원 사이트 목록에 있는지 검색하여 수행의 가부를 결정한다. 이와는 다르게 자원 사이트가 결정되지 않았다면, 자원 사이트의 목록과 함께 자원 할당 스케줄러에게 위임한다. 자원 할당 스케줄러는 받은 목록을 다양한 알고리즘으로 부하 균형을 고려할 수 있을 것이다. 본 논문은 간단하게 라운드 로빈 방식을 사용하여 구현하였다.

자원의 부하 균형은 각 자원 사이트에 설치된 컴포넌트를 적극 활용하였다. 위에서 자원이 선택되었을 때, 바로 자원 사이트로 작업을 보내지 않고, 자원에 우선 질의부터 하게 된다. 자원에 질의를 하여, 작업 수행이 가능하다는 응답이 오게 되면, 작업을 전달하며, 그렇지 않은 경우에는 대체 자원을 다시 검색한다. 모든 시도를 하여 자원이 없다고 판단하였을 때에는 큐에 두며, 자원이 사용 가능해질 때에 다시 작업을 전달하는 방식으로 하였다.

7. 결론 및 향후 연구 과제

본 논문에서 제안한 방법으로 실제 서비스로 구현하여 검증 중에 있다. 현재는 Least-Recently 방식으로 사용자의 작업 풀의 선택하고, 라운드 로빈 방식으로 자원을 선택하고 있다. 이러한 환경에서 실제 사용자의 독점 문제와, 자원 부하 균형의 상황을 파악하고, 이를 개선 하고자한다.

본 논문에서 적용한 서비스는 과학 계산에 사용되는 소프트웨어를 대상으로 하였기 때문에 작업의 수행 시간을 사전에 예측할 수 있을 것이다. 따라서 향후, 이 항목을 추가하고, 사용자의 자원 요구사항을 반영하여, 스케줄링을 하도록 기능을 향상시킬 것이다.

8. 참고 문헌

[1] http://www.mgrod.or.kr

[2] http://www.cs.wisc.edu/condor/

[3] M. J. Litzkow and M. Livny "Experience With The Condor Distributed Batch System" Proceedings of the IEEE Workshop on Experimental Distributed Systems, pp.97-101, October 1990

[4] M. C. Ferris and T. S. Munson "Modeling languages and Condor: metacomputing for optimization" Mathematical Programming, 88(3), pp. 487-505, 2000.

[5] Jiandong Huang, Y. Wang, N.R. Vaidyanathan and Feng Cao "GRMS: A Global Resource Management System for Distributed QoS and Criticality Support" ICMCS, pp. 424-432, 1997.

[6] Ed Seidel, Gabrielle, AndréMerzky and Jar다 Nabrzyski "GridLab-a grid application toolkit and testbed" Future Generation Computer Systems, 18(8), pp. 1143-1153, October 2002.

[7] Mathide Romberg "The UNICORE Grid infrastructure" Scientific Programming, 10(2), pp. 149-157, 2002.

[8] Dietmar W. Erwin "UNICORE - a Grid computing environment"

[9] 심규호, 황선태, 정갑주, 박형우 "응용 그리드를 위한 워크플로우 시스템 설계" 한국정보과학회 봄 학술발표대회 논문집(B), 제 31권, 제 1 호, pp.388-390