웹 서비스 기반 윈도우 환경 클러스터의 설계

방인주° 김진석
서울시립대학교 컴퓨터통계학과
{ijbang°, jskim}@venus.uos.ac.kr

WebService based Window Environment Cluster Design

In Ju Bang° Jin Suk Kim
Dept. of Computer Science & Statics, University of Seoul

요 약

현재까지 대부분의 클러스터들은 웹 서비스를 지원하지 않고 있다. 하지만 웹 서비스를 사용하면 서비스 사이의 상호 의존성을 최소한으로 줄일 수 있다. 그리고 프로토콜에 얽매이지 않기 때문에 코드 재활용성 및 확장성이 높아진다는 장점이 있다. 또한 서비스 간에는 개발 언어에 상관없이 서비스 개발이 가능하며 이렇게 개발된 서비스간의 통합 역시 용이하기 때문에 클러스터를 구축하는 비용을 줄일 수 있다. 더욱이 클러스터를 윈도우 환경으로 구축할 경우에는 다양한 GUI 도구들과 많은 어플리케이션을 사용할 수 있어서 리눅스를 기반으로 클러스터를 구축할 때 보다 좀 더 쉽게 클러스터를 구축할 수 있다. 따라서 본 논문에서는 윈도우 환경의 컴퓨터들을 서로 연결하여 클러스터를 만들고 웹 서비스를 통하여 작업을 스케쥴링할 수 있는 클러스터 작업 스케쥴러를 설계하고자 한다.

본 논문에서 설계한 클러스터 작업 스케쥴러는 크게 SchedulingClient, ClusterJobScheduler와 ResourceManager 3 부분으로 나누어진다. SchedulingClient는 사용자의 작업 정보를 입력받는 사용자용 어플리케이션이고 ClusterJobScheduler는 사용자의 작업 정보를 큐에 저장하고, 스케쥴링하며, 작업의 결과를 사용자에게 전달하는 웹 서비스 어플리케이션이다. 그리고 *ResourceManager*는 클러스터를 구성하는 컴퓨터를 관리하는 기능을 한다.

1. 서 론

클러스터는 소프트웨어와 네트워크를 통하여 여러 대의 컴퓨터들을 서로 연결하여 마치 하나의 컴퓨터처럼 사용할 수 있도록 만든 것으로서 병렬 프로그래밍에 주로 사용되어 진다. 또한 클러스터는 일반적으로 저렴한 가격으로 구축이 가능하며 단일 컴퓨터보다 높은 활용성과 성능을 낼 수 있다. 지금까지 연구되어 온 클러스터 컴퓨팅 연구로는 Beowulf, Condor[1], MPI[2], NOW, NIMROD, PBS, PVM 등이 있다. 또한 현재 상업용 소프트웨어로는 Codine, LoadLeveler[3], LSF[4] 등이 있으며 이외에도 많은 기업에서 클러스터 컴퓨팅 제품을 출시하고 있다.

대부분의 클러스터들은 리눅스를 기반으로 구축되어 있다. 하지만 리눅스는 윈도우와 비교하여 사용이 어려우며 보급이 많이 되어 있지 않고 GUI 개발도구가 부족하다는 단점이 있다. 반면에 윈도우는 리눅스보다 많은 어플리케이션과 GUI 도구들을 가지고 있으므로 클러스터의 개발비용을 줄일 수 있으며 확장성을 높일 수 있다. 또한 웹 서비스[5]를 사용하면 서비스간의 상호의존성을 최소한으로 줄일 수 있고 프로토콜에 얽매이지 않기 때문에 코드의 재사용성을 높일 수 있다. 따라서 본 논문에서는 윈도우 환경의 컴퓨터들을 서로 연결하여 클러스터를 구축하고 웹 서비스를 통하여 여러 개의 작업을 스케쥴링할 수 있는 클러스터를 설계하고자 한다.

본 논문의 순서는 다음과 같다. 2장에서는 관련 연구를 살펴보고 3장에서는 본 논문에서 설계한 윈도우 환경 클러스터를 설명한다.

2. 관련 연구

관련 연구로서 LSF는 Workload를 조절하는 작업 스케쥴러이다. LSF는 다양한 스케쥴링 기법을 지원하고 이기종의 컴퓨터들로 클러스터를 구축할 수 있지만 리눅스를 기본 운영체제로 채택하고 있어서 LSF를 사용하기 위해서는 많은 양의 매뉴얼을 읽어 보아야 한다. Server Cluster는 Microsoft에서 개발 중인 Windows Server 2003 환경의 클러스터이다. Server Cluster는 윈도우 환경의 클러스터이긴 하지만 다양한 스케쥴링 기법을 지원하지 않는다는 단점이 있다. LoadLeveler는 IBM의 배치 작업 스케쥴링 어플리케이션이다. LoadLeveler는 IBM RS/600, Sun SPARCstations, Silicon Graphics IRIS 등 제한된 컴퓨터에서 운영된다는 단점이 있다.

3. 웹 서비스 기반 윈도우 환경 클러스터

3.1 병렬 프로그래밍용 라이브러리

웹 서비스를 기반으로 한 윈도우 환경의 클러스터를 구축하기 위해서는 병렬 프로그래밍용 라이브러리가 필요하다. 본 논문에서는 윈도우용 MPICH를 사용하였다. MPICH는 공개 소프트웨어로 윈도우, 유닉스, Globus용

3가지 버전이 있으며 윈도우용 MPICH를 사용하면 Visual C++ 6 과 Visual Fortran 6를 사용해서 MPI 어플리케이션을 컴파일 할 수 있다.

3.2 웹 서비스 기반 클러스터의 구조

웹 서비스를 기반으로 한 윈도우 환경 클러스터의 전체적인 구조를 살펴보면 <그림 1>과 같다.

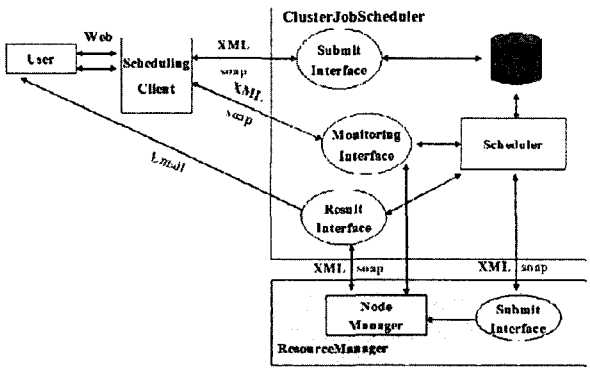

그림 1. 웹 서비스를 기반으로 하는 윈도우 환경 클러스터의 전체적인 구조

사용자는 SchedulingClient를 통하여 작업 정보를 입력하고 작업 정보는 XML형태로 변환되어 SOAP 프로토콜을 통하여 ClusterJobScheduler에 전달된다. ClusterJobScheduler는 작업을 큐에 저장하고 스케쥴링하며 자원에 작업을 할당한다. 또한 ResourceManager는 작업을 실행시키고 작업이 완료되면 작업의 결과를 사용자의 Email로 전달한다.

3.3 SchedulingClient

SchedulingClient는 사용자들에게 윈도우 환경의 클러스터를 쉽고 편리하게 사용할 수 있는 인터페이스를 제공한다. SchedulingClient는 사용자의 컴퓨터나 마스터 노드에서 운영되며 사용자들에게 익숙한 웹 어플리케이션 또는 윈도우 어플리케이션 프로그램이다.

SchedulingClient에 사용자가 입력해야하는 기본 파라미터는 다음과 같다. 사용자 인증에 필요한 사용자 이름과 암호, 그리고 스케쥴링에 필요한 필요 노드 수, 예상 실행시간, 결과를 받기위한 사용자의 메일 주소, 마지막으로 실제 병렬처리 작업파일인 실행파일이다.

<그림 2>는 SchedulingClient에서 ClusterJobScheduler에게 SOAP 프로토콜을 사용해서 서비스를 요청하는 예를 보여준다. 사용자가 입력한 파라미터는 XML 형태의 문서로 전달된다.

```
POST /ClusterJobScheduler/Service1.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/SubmitInterface"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <SubmitInterface xmlns="http://tempuri.org/">
      <name>string</name>
      <password>string</password>
      <nodecount>int</nodecount>
      <runtimeestimation>int</runtimeestimation>
      <email>string</email>
      <finename>string</finename>
    </SubmitInterface>
  </scap:Body>
</soap:Envelope>
```

그림 2. SOAP 프로토콜을 사용한 서비스 요청 예제

3.4 ClusterJobScheduler

ClusterJobScheduler는 웹 서비스를 통하여 SchedulingClient로부터 작업을 받아서 스케쥴링 및 작업 할당, 모니터링, 큐잉 기능을 수행한다. ClusterJobScheduler는 SubmitInterface, MonitoringInterface, ResultInterface, Scheduler, Queue, Administrator로 구성되며 각 기능은 <표 1>과 같다.

표 1. ClusterJobScheduler의 구성 요소 및 기능

구성 요소	기 능
SubmitInterface	-사용자 인증 -큐에 사용자의 작업 저장
MonitoringInterface	-사용자의 작업 진행 상태 확인
ResultInterface	-사용자에게 작업 결과를 Email로 전달
Scheduler	-큐에 저장된 사용자의 작업 스케쥴링
Queue	-사용자의 작업 정보 저장 -스케쥴링 정보(log) 저장
Administrator	-클러스터의 전체 노드 설정 및 관리 -사용자 등록, 수정, 삭제 -사용자의 작업 상태 확인

<그림 3>은 ClusterJobScheduler에서 제공하는 웹 서비스 정의형식의 일부분을 나타낸다. 클러스터 사용을 원하는 사용자는 웹 서비스 정의형식에 따라 서비스 요청을 하여야한다.

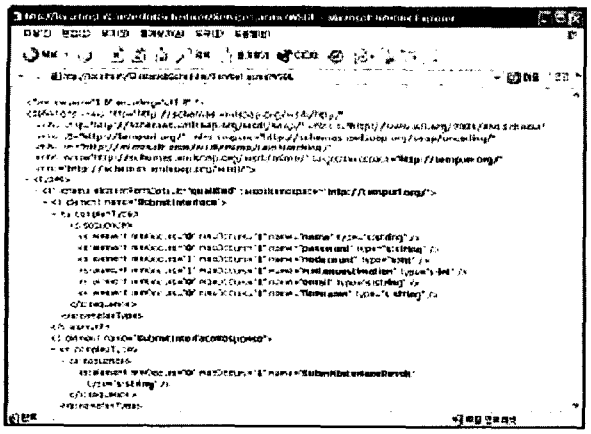

그림 3. ClusterJobScheduler에서 제공하는 웹 서비스 정의형식

3.5 ResourceManager

ResourceManager는 각각의 모든 계산 노드에서 운영되며 SubmitInterface와 NodeManager로 구성된다. SubmitInterface는 사용자의 작업을 받아 계산 노드에서 실행되도록 하는 기능을 수행하고, NodeManager는 계산 노드를 관리하면서 ClusterJobScheduler에게 자원의 정보(CPU, Memory, WorkLoad)와 작업의 결과를 되돌려주는 역할을 한다.

3.6. Scheduling Algorithm

스케쥴링 알고리즘은 사용자의 작업 정보를 이용하여 작업의 실행 순서와 작업이 수행될 노드를 결정하는 것이다. 클러스터는 스케쥴링을 어떻게 하는가에 따라서 그 사용 효율이 크게 달라지기 때문에 클러스터에서는 매우 중요한 역할을 하고 있으며, MCT, MET, MECT, SA, Backfilling, gang 스케쥴링 등 많은 연구들이 이루어져 오고 있다[6, 7, 8, 9].

스케쥴링 알고리즘을 수행 시간에 따라 분류하면 온라인 방식과 배치 방식으로 나누어진다. 온라인 방식은 작업을 전달 받는 즉시 스케쥴링을 하는 방식이고 배치 방식은 작업을 큐에 저장한 다음 한꺼번에 스케쥴링을 하는 방식이다. 온라인 방식과 배치 방식에서 스케쥴링을 정확히 하기 위해서는 작업의 정확한 준비시간 실행시간, 완료시간 등을 알아야하지만 실제 환경에서는 이 값을 정확히 알기 어렵다. 따라서 본 논문에서는 사용자가 예측한 실행시간을 사용한 Backfilling 스케쥴링 알고리즘을 사용한다.

4. 결 론

본 논문에서는 웹 서비스를 기반으로 한 윈도우 환경의 클러스터를 설계하였다. 웹 서비스를 기반으로 한 윈도우 환경 클러스터는 리눅스 환경의 클러스터보다 개발이 용이하고 확장성이 높다. 또한 웹 서비스를 사용함으로써 프로토콜에 얽매이지 않기 때문에 코드의 재사용이 가능하며, 서비스 간에는 개발 언어에 상관없이 개발이 가능하다는 장점을 갖는다.

참 고 문 헌

[1] M. J. Litzkow, M. Livny, and M. W. Mutka, "Condor-A hunter of idle workstations," Proc. of the 8th International Conference of Distributed Computing Systems, pp. 104-111, 1988.

[2] MPI, http://www-unix.mcs.anl.gov/mpi

[3] S. Kannan, P. Mayes, M. Roberts, D. Brelsford, and J. F Skovira, "Workload Management with LoadLeveler," IBM Redbooks, 2001.

[4] S. Zhou, J. Wang, X. Zheng, and P. Delisle, "Utopia: A load sharing facility for large, heterogeneous distributed computing systems," Technical Report CSRI-257, Computer Systems Research Institute, University of Toronto, 1992.

[5] Web Services, http://www.w3.org/2002/ws

[6] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, "Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneous Computing Systems," Proc. of the 8th Heterogeneous Computing Workshop, pp. 30-44, 1999.

[7] S. Chiang, A. Arpaci-Dusseau, and M. K. Vernon, "The Impact of More Accurate Requested Runtimes on Production Job Scheduling Performance," LNCS, pp. 103-127, 2002.

[8] E. Shmueli and D. G. Feitelson, "Backfilling with Lookahead to Optimize the Performance of Parallel Job Scheduling," LNCS, pp. 228-251, 2003.

[9] D. Feitelson, L. Rudolph, and U. Schwiegelshohn, "Parallel Job Scheduling - A Status Report," 10th Workshop on Job Scheduling Strategies for Parallel Processing, 2004.