그리드 환경에서 어플리케이션의 특성을 고려한 자원선택 시스템의 설계

정평재°, 이윤석
한국외국어대학교 전자정보공학부
{system9°, rheeys}@dslab.hufs.ac.kr

Design of a resource selection system considering the characteristics of applications in Grid environment

Pyoung-Jae Jung° Yunseok Rhee
School of Electronics & Information Eng., Hankuk Univ. of Foreign Studies

요 약

본 연구에서는 웹서비스 기반 그리드 환경에서 효과적인 자원선택을 할 수 있는 자원 선택 프레임웍을 설계 구현하였다. 이 프레임웍을 활용하여, 어플리케이션이 요구하는 자원 명세와 자원 공급자의 자원공급 정책을 기술할 수 있도록 지원하고, 이를 토대로 다양한 자원 선택알고리즘을 개발할 수 있을 것으로 기대된다. 또한 본 연구에서 구현한 프로토타입 시스템을 소규모의 그리드 환경에 적용해 얻은 실험 결과를 통해, 그리드 상에서 어플리케이션의 성질을 고려한 자원 선택이 충분한 효과를 나타냄을 확인할 수 있었다.

1. 서론

그리드(Grid)는 지리적으로 분산되어 있는 고성능 컴퓨팅 자원을 네트워크로 상호 연동하여 조직과 지역에 관계없이 사용 할 수 있는 환경을 말하며, 과학 계산, 경제예측, 온라인 게임 서버 등 많은 컴퓨팅 자원을 필요로 하는 시스템에서 그 발전 가능성을 인정받고 있다.

특히 그리드 서비스를 제공하는 미들웨어에서 자원 관리 시스템(resource management system)은 원격지의 자원 상태를 파악하고, 정보를 수집하며, 사용자의 요구에 가장 적합한 자원을 할당하는 기능을 제공하는데 [1,2], 수많은 자원 가운데 최적의 자원 집합을 선택하는 것이 작업의 처리 성능에 결정적으로 영향을 미칠 것이다.

그러나, 대부분의 자원선택 미들웨어는 자원선택 과정에서 최소 요구조건(resource constraint)만을 고려하였고, 이를 만족하는 집합에서 자원선택 시스템이 스스로 정한 임의의 기준에 따라 자원을 선택, 제공하였다. 그러나, Condor의 matchmaking[3], AppLes의 application-specific scheduling[4]과 같이 어플리케이션의 자원 요구 선호도와 자원 제공자의 제공 정책을 고려한 연구들이 활발히 진행되고 있으며 그리드 시스템의 규모가 증대될수록 그 영향은 더욱 커질 것으로 예상된다.

본 연구에서는 이러한 연구 중요성에 비추어 그리드 시스템을 지원하는 미들웨어중의 하나인 웹서비스 기반의 표준 그리드 미들웨어로 자리잡은 Globus Toolkit 3 OGSA(Open Grid Service Architecture) [5]기반에서 다양한 형태의 자원선택 알고리즘을 구현할 수 있는 프레임웍을 설계하고 프로토타입을 구현하였다.

2. 자원 선택 시스템 설계

그리드 시스템상의 수많은 자원에는 각 자원별 사양과 해당 자원 공급자의 정책들이 다양하게 분포한다. 이러한 자원을 요구하는 사용자 측에서도 마찬가지로 사용자별 자원 선호도와 그리드 시스템에서 실행할 어플리케이션이 선호하는 자원 사양이 다양하게 분포한다. 자원 선택 시스템에서는 이러한 다양한 요구를 정확하게 표현이 가능하도록 제공 해주어야 하며, 자원 공급자와 사용자의 선호도나 요구사항에 대한 내용을 계산하기 쉽게 수치적으로 표현이 가능해야 한다. 이렇게 수많은 자원 공급자의 자원 명세와 사용자의 자원 요구사항을 효과적으로 처리하기 위해서 자원 관리, 선택 시스템에서는 다음과 같은 기능을 지원해야 한다.

1. 자원 제공자의 자원 명세 및 제공 정책 기술
각 노드가 제공하는 CPU, 메모리, 디스크, 네트웍 등의 자원 정보를 기술하고 기관별, 사용자별에 따른 각각의 제공 정책을 기술함으로써 사용자와 제공자 간의 상호 선호도를 반영한 자원 선택이 가능해야 한다.

2. 그리드 자원 검색
그리드 시스템에서 제공하는 여러 자원 목록과 상태를 제공하는 정보제공 시스템이 필요하고, 이를 통해 그리드 시스템의 전체 가용한 자원의 양을 파악하거나, 세부적으로 들어가서 구체적인 위치의 자원 명세의 검색이 가능하다.

3. 사용자의 자원 요구사항 기술
사용자가 그리드 시스템에서 실행할 어플리케이션에 따라 요구되는 자원 필요 사항(예를 들어, CPU 성능, 운영체제, 메모리, 디스크 용량 등)을 기술할 수 있는 기능을 지원해야 한다.

4. 실행할 어플리케이션의 특성 표현
어플리케이션의 자원 소모 특성과 선호하는 자원의 형태를 기술할 수 있어야 한다. 본 연구에서는 그리드 자원과 어플리케이션과의 선호도를 수식 모델을 통해 표현할 수 있도록 지원하였다.

위의 요구조건을 만족하는 시스템을 설계하기 위해서 본 논문 에서는 그림 1과 같이 각 기능별로 Job Submission Client, Job Submission Service, Job Scheduling Service로 서비스를 구분하여 자원 선택 시스템을 설계하고 각 서비스 간에는 XML 명세를 통해 데이터를 주고 받도록 설계하였다.

3. 모듈별 기능 설명

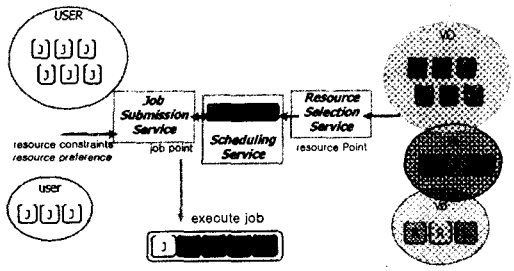

그림 1 시스템 구성도

3.1. Job Submission Client

사용자가 그리드 시스템으로 어플리케이션 실행을 요청할 수 있도록 한 GUI 방식의 인터페이스 프로그램이다. 그리드 시스템에서 실행할 작업의 어플리케이션을 선택, 어플리케이션이 요구하는 자원의 종류와 최소 요구조건, 수식으로 표현하는 자원 선호 정도를 설정할 수 있고 최종적으로 계산되어 전달받은 자원 목록과 해당 자원의 상세 명세, 그리고 실행 지시, 현재 실행을 하고 있는 작업의 상태를 모니터링 할 수 있는 기능들로 구성 되어 있다.

3.2. Job Submission Service

Job Submission Service는 Job Submission Client로부터 받은 어플리케이션 실행요청을 전달받는 서비스 모듈이다. Job Submissio n Client에서 입력한 어플리케이션 실행 조건을 Scheduling Service 에 전달하고, Scheduling Service로부터 전달받은 최적 선택자원들을 사용하여 어플리케이션을 실행하는 모듈이며, 실행중인 어플리케이션의 상태를 모니터링하는 역할도 수행한다. Job Submission Client와 직접적으로 연결되는 모듈이므로 사용자에게 전해지는 정보(실행중인 어플리케이션의 상태, 자원의 상세 명세 등)를 제공하는 인터페이스가 준비되어 있다. 이러한 기능을 위해서 Job Submission Client로부터 여러 가지 자원 선택을 위해 받는 정보가 있는데, 그 내용은 다음과 같다.

1. resource constraint
2. resource preference
3. resource arguments of interest

resource constraint는 어플리케이션이 요구하는 자원의 양을 말한다. 이는 그림3과 같이 구체적인 자원의 명세를 기술할 수 있는 GLUE schema[6]를 이용해서 기술한다. resource constraint의 의미는 어플리케이션이 작동할 수 있는 최소한의 자원 요구량을 의미하는 것이므로 Resource Selection Service(RSS)에서 자원을 검색할 때 필터링 기능을 한다.

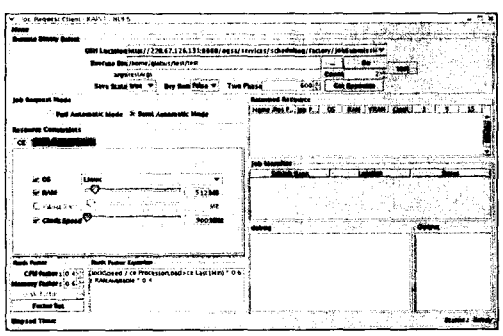

그림 2. Job Submission Client 화면

resource preference는 어플리케이션의 자원 선호도를 나타내는 수식 형태로 표현을 한다. 이 resource preference는 그림 4와 같이 어플리케이션의 자원 소모 특성을 반영한 수식(자원종류별 가중치를 가지는)을 이용해서 표현한다. Scheduling Service에서는 이 수식에서 계산 된 점수와 자원 제공자의 자원 제공 정책 점수를 이용, 어플리케이션의 자원별 선호도가 반영이 된 자원 우선순위를 산정하게 된다.

```
<constraints>
 <constraint type="glue">
  <element name="ce:OperatingSystem">
   <attribute name="ce:Name" value="Linux" condition="="/>
  </element>
 </constraint>
 <constraint type="glue">
  <element name="ce:Architecture">
   <attribute name="ce:PlatformType" value="i686" condition="="/>
  </element>
 </constraint>
 <constraint type="glue">
  <element name="ce:MainMemory">
   <attribute name="ce:RAMAvailable" value="256" condition=">="/>
  </element>
</constraint>
```

그림 3 Resource constraint의 XML 기술 예

```
<rank>
 <condition type="glue">
  <element name="ce:FileSystem">
   <attribute name="ce:Name" value="/tmp" condition="="/>
    <attribute name="ce:AvailableSpace" value="200" condition=">="/>
  </element>
 </condition>
 <point>
  <operator type="divide">
   <operand type="glue">
    <element name="ce:FileSystem">
     <attribute name="ce:AvailableSpace"/>
    </element>
   </operand>
   <operand type="integer">
   1000
   </operand>
  </operator>
 </point>
</rank>
```

그림 4 Resource preference의 XML 기술 예

3.3. Job Scheduling Service

Scheduling Service는 실행 요청을 받은 어플리케이션의 자원 요구 조건에 최적으로 어울리는 자원을 선택하고, 자원 할당 스케줄링을 수행하는 서비스 모듈이다. Job Submission Service로부터 받은 사용자의 자원 요구 명세는 앞에서 언급한 것처럼 resource constraints, resource preference로 구분되어 전달된다. 이 정보는 그대로 RSS로 넘겨져서 현재 그리드 시스템에서 해당 조건(resource constraints)을 만족하는 자원 후보 목록과 각각 자원에 대한 어플리케이션의 선호 점수(resource preference), 그리고 각 자원 제공자 정책에 따른 자원 제공자의 작업 선호 점수를 반환받게 된다. 본 논문에서는 어플리케이션의 자원 선호 점수를 job point라 명명하고 자원 제공자의 자원 제공 정책에 따른 작업 선호 점수를 resource point라로 명명한다.

자원 목록은 그림 6과 같은 형태로 RSS로부터 Scheduling Servic

```
<resourceList>

 <resource name="eclipse2.hufs.ac.kr" id="eclise2.hufs.ac.kr">
   <rank jobPoint="12" resPoint="10"/>
 </resource>
 <resource name="eclipse3.hufs.ac.kr" id="eclipse3.hufs.ac.kr">
   <rank jobPoint="1" resPoint="6"/>
 </resource>

</resourceList>
```

그림 6 RSS로부터 전달받은 자원 목록의 예

e로 전달되며 각 자원의 위치 정보와 job point, resource point가 함께 전달된다. Scheduling Service에 설치되어 있는 자원 할당 스케줄링 알고리즘 엔진에 의해 RSS로부터 전달받은 job point, resource point와 자원의 상세 명세를 가지고 최적자원을 계산할 수 있다. 본 논문에서는 다양한 자원 선택 알고리즘을 개발할 수 있는 프레임웍을 작성하는 것에 초점을 맞추어서 다양한 자원 할당 스케줄링 알고리즘 엔진을 Job Scheduling Service에 설치하여 실제로 자원이 효과적으로 선택되는지 실험할 수 있도록 하였다. 이렇게 크게 3개 모듈로 구분이 되는 시스템은 그림 7과 같이 동작하며 각 모듈간 통신은 XML을 이용한다.

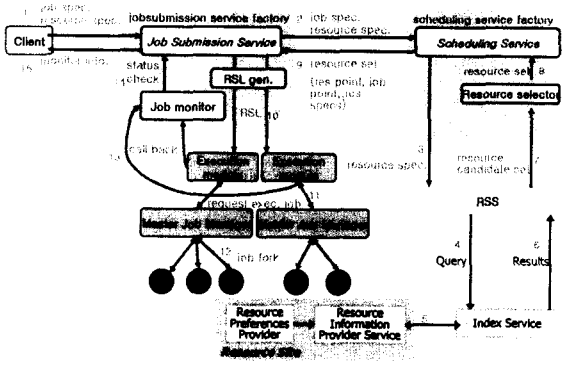

그림 7 시스템 구성 및 동작 순서도

4. 동작 과정 및 실험 결과

본 논문에서 설계 구현한 자원선택 시스템의 동작과 다양한 사양의 자원이 혼재한 환경에서 최적자원을 선택하는 실험을 통해 자원 선택 시스템의 필요성과 그 효과를 알아보고자 한다.

이를 위해 그림 8과 같이 그리드 시스템에서 실행할 어플리케이션의 설정과 필요 자원의 사양을 설정하고, count값은 요구할 자원의 개수, resource constraints는 어플리케이션의 필요 자원 사양을 각 항목별로 설정 가능하도록 구현하였다. 그리고 작업의 자원 선호도를 표현하기 위해 실행할 어플리케이션의 자원 선호 정도를 CPU, Memory, Disk 등의 성능 척도를 수식으로 표현하여 입력한다.

그림 9에서는 Resource Scheduling Service에 의해 반환된 추천 자원 목록과 각 자원의 항목별 상세 명세와 현재 작업상태가 표시되고, resource constraint에서 설정한 자원 사양에 만족하는 자원이 선택된 것을 확인할 수 있다.

먼저 어플리케이션에 적합한 자원을 기술하고 선택하는 과정이 성능에 미치는 영향을 수행시간 측면에서 알아보았다. 실험에 사용된 테스트 프로그램은 sqrt, sin, cos, tan 등의 네 가지 math 라이브러리 함수를 2억5천만번 호출하는 프로그램으로 소모되는 메모리의 양은 거의 없고, CPU자원을 주로 요구하는 특성을 보인다. 펜티엄4 2.0MHz 사양의 컴퓨터에서 약 2분이 소요되는 프로그램이다.

표 1은 실험에 사용된 자원의 사양을 나타내며, 실험 방법은 총 5개의 자원 중에서 3개를 요청하였다. 테스트에 사용된 프로그램의 특성은 CPU의 클럭 속도에 큰 영향을 받으므로, 자원 선택 시스템이 활성화 되었을 경우와 비활성화 되었을 경우를 비교해서 올바른 자원이 선택이 되었을 경우와 그렇지 않을 경우의 성능을 비교해보았다. 자원선택 시스템을 비활성화시킬 경우 표 2와 같이 기본적으로 먼저등록이 되어 있는 자원이 선택이 되어 저사양의 시스템(여기서는 crystal과 orbit) 둘이 선택되는 것을 확인할 수 있었다.

노드명	CPU	메모리
crystal	Pentium III 933MHz	384 MB
orbit	Pentium III 733MHz	512 MB
eclipse6	Pentium IV 2GHz	1536 MB
eclipse7	Pentium IV 2GHz	1536 MB
eclipse8	Pentium IV 2GHz	1536 MB

표 1 실험에 사용된 자원 사양

실험 결과 자원선택 시스템을 이용해서 보다 작업이 원활히 돌아갈 수 있는 자원을 선택한 결과, 그렇지 않았을 경우에 비해 30.1%의 성능 향상을 보였다. 물론 이 수치는 본 실험의 경우에 나타난 한 결과값에 불과하지만, 작업에 보다 적합한 자원을 선택하는 것이 성능 향상에 상당한 영향을 줄 수 있다는 것을 확인할 수 있다.

순번	자원선택 시스템 활성	자원선택 시스템 비활성
1	143.820 sec	208.378 sec
2	142.535 sec	203.107 sec
3	144.630 sec	202.814 sec
4	140.898 sec	202.368 sec
5	139.676 sec	201.525 sec
6	142.226 sec	202.437 sec
평균	142.203 sec	203.438 sec

표 2. 자원선택 시스템 활성, 비활성 시의 성능 비교

5. 결론

본 논문에서는 그리드 환경에서 자원선택의 중요성을 인식하고 그리드 시스템을 지원하는 미들웨어의 표준인 Globus Toolkit 상에서 다양한 자원 선택 알고리즘을 실험할 수 있는 프레임웍을 설계, 구현하였다. 그리드 시스템 상에서 작업실행 요청을 처리하는 Job Submission Service, 사용자가 원하는 자원 사양과 자원 제공자의 정책을 반영한 자원의 조합을 계산해서 최적의 자원 할당 스케줄링을 선택해주는 Scheduling Service, 그리고 사용자가 편하게 그리드 시스템에 작업 실행을 요청할 수 있게 하는 클라이언트인 Job Submission Client의 모듈을 설계하였고 어플리케이션의 자원 소모, 자원 요구 특성을 반영할 수 있는 resource preference를 처리할 수 있도록 설계하였다.

이를 활용한 간단한 자원선택 실험을 통해 적절한 자원 선택이 성능에 미치는 영향을 확인할 수 있었으며, 향후 보다 다양한 자원 선택 알고리즘을 개발하는데 기여할 것으로 기대한다.

또한 어플리케이션의 프로파일링(profiling) 기능을 추가해서 사용자에게 보다 정확한 어플리케이션의 자원 소모 정보를 제공해서 반복적으로 수행하는 어플리케이션에 대해 정확하고 효율적인 자원 선택이 이루어지도록 지원하는 것이 향후 과제이다.

[1] I. Foster, C. Kesselman, S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International J. of Supercomputer Applications, 15(3), 2001.

[2] I. foster, C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit" J. of Supercomputer Application, 1997.

[3] Rajesh Raman, Miron Livny, and Marvin Solomon, "Policy Driven Heterogeneous Resource Co-Allocation with Gangmatching", In Proc. of HPDC, 2003

[4] F. Berman et al., Shao. Application level scheduling on distributed heterogeneous networks. In Proc. of Supercomputing 1996.

[5] http://www.globus.org/ogsa

[6] http://www.hicb.org/glue/glue.htm