Grid에서의 자원 정보 저장 및 할당에 대한 설계 및 구현

유승훈[0] 이태동 진정훈 장재형 임중호 정창성
고려대학교 대학원 전자컴퓨터공학과
{friendyu[0], leetd, daein, jjh3368, jhlim}@snoopy.korea.ac.kr, csjeong@charlie.korea.ac.kr

Design and implementation about resource Information storing and allocation on Grid

Seung-Hun Yoo[0] ,Tae-Dong Lee, Jung-Hun Jin, Jae-Hyung Jang
Jung-Ho Lim, Chang-Sung Jeong
Dept. of Electronics & Computer Engineering Graduate School, Korea University

요 약

Grid란 네트워크로 연결된 가상의 슈퍼 컴퓨터를 말하는 것으로 Grid Middleware Toolkit인 Globus가 대표적인 구현물이다. Globus는 사용자에게 Grid의 상태 정보를 제공하는 방법으로 RSL에 의한 query와 MDS의 LDAP 프로토콜을 통한 query로 나뉘어진다. 그러나 이런 방법들은 다이나믹한 환경에서의 잦은 정보의 변화로 인해 query나 실시간적인 자료의 처리가 요구되는 시스템에서 긴 query response로 인해 효과적이지 않다. 따라서 본 논문에서는 MDS 서비스에서의 자료의 저장 방법 및 자원 할당방법에 대해 살펴보도록 하겠다.

1. Introduction

월드 와이드 웹(WWW)의 개발로 인해 전세계에 인터넷 상에 있는 정보들을 쉽게 검색하고 접근할 수 있었다. 그러나 이젠 단순한 자료 공유뿐만 아니라 다양한 리소스(CPU, HDD, MEMORY, NETWORK BANDWIDTH)등을 공유하여 고성능 컴퓨팅을 제공하기 위해 Grid Computing이 출현하였다. 다시 말하자면, Grid란 지리학적으로 분산되어 있는 고성능 컴퓨팅 자원을 네트워크로 상호 연동하여 조직과 지역에 관계없이 사용할 수 있는 환경이다. 현재 전 세계적으로 Grid에 관한 끊임없는 연구가 이루어지고 있으며, Grid 개발 과제에서 가장 많이 사용되는 Middleware는 바로 Globus Middleware Toolkit이다. Globus Toolkit이 널리 사용되는 이유로는 Grid에서 필요로 하는 다양한 서비스들을 독립적인 요소로 분리함에 따라 기존에 각 시스템 및 네트워크의 관리정책을 수용하면서 각 요소들과 결합하여 Grid를 구축해 나간다는 점이다.

Globus Toolkit이 제공하는 핵심 서비스는 크게 자원 정보 서비스, 자원 관리 서비스, 데이터 관리 서비스, 그리드 보안 서비스 등으로 나눌 수 있는데 우선 본 논문에서는 그리드 자원 정보 서비스를 수행하는 요소인 MDS가 이용하는 LDAP(Lightweight Directory Access Protocol)에 대해 알아보고 다음으로는 실질적으로 MDS 서비스를 제공하기 위한 GIIS 및 GRIS 환경 구성 및 설계 과정에 대하여 짚어보고 마지막으로는 MDS로부터 얻어진 각각의 정보들을 효율적으로 저장 및 Query 하기 위해 다양한 알고리즘 및 Database와 연동시켜 나타난 결과들에 대해 정리하고자 한다.

2.Components of Grid Information Services

2.1 LDAP(Lightweight Directory Access Protocol)

LDAP은 인터넷 상에서의 디렉토리 서비스를 위해 공개된 표준 프로토콜로써 네트워크 상에 있는 파일이나 장치들과 같은 자원 등의 위치를 찾을 수 있게 해주는 소프트웨어 프로토콜이다. LDAP은 디렉토리 서비스를 제공하는데 디렉토리는 네트워크상의 모든 리소스에 대한 내용을 계층적 구조로 리소스들을 저장하고, 모든 애플리케이션의 사용자 관리는 통합하여 이를 관리자로 하여금 중앙집중 식으로 관리할 수 있도록 하는 개념으로서 각 시스템을 통합, 조직화, 통합된 관리를 쉽게 할 수 있다는 이점을 가지고 있다. LDAP 디렉토리는 기본적으로 계층으로 구성되는 단순한 트리 구조를 갖고 있는데 이를 DIT(Directory Information Tree)라 일컫는다. LDAP 트리(Tree) 구조에서 각 노드들을 엔트리(Entry)라고 부르고 엔트리는 LDAP에서 하나의 데이터를 나타낸다. 또한 모든 엔트리는 그 자신의 위치와 고유성을 나타내는 DN(Distinguished Name)으로 구분된다. 기본적으로 LDAP은 디렉토리에 저장되어 있는 데이터를 어떻게 가지고 오는 방법을 프로토콜화 하여 Search, Modify, Add, Delete, Bind, Unbind 등과 같은 기본 명령어를 제공한다.

LDAP의 특징으로는 우선 검색이나 읽기에 있어서 탁월한 성능을 가지고 있고 계층적 트리 구조로 표현이 되기 때문에 데이터베이스의 관계형 모델보다 관리나 뷰가 편리하고 다중값을 넣을 수 있는 Attribute를 제공한다. 또한 SQL보다 디렉토리의 Query가 더 간단하고 네임스페이스를 통한 검색이 가능 하다. 예를 들어, SQL 문법을 알 필요 없이 (,), >=, <=, *, &, |, ! 등과 같은 연산자를 이용한 유연성 있는 Query를 제공한다. 다음과 같은 특징들을 가지고 실제 LDAP을 이용하여 네트워크 리소스에 대한 접근 및 검색을 할 것이다.

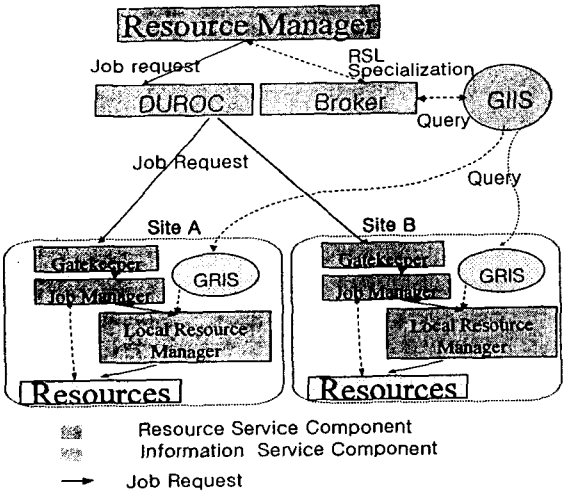
2.2 MDS(Metacomputing Directory Service)

Grid의 computing environment는 기본적으로 물리적으로 나누어진 개인이나 조직이 가상 공간에서 자신들이 소유하고 있는 자원을 공유할 수 있는 VO(Virtual Organization)을 형성한다. 이러한 VO을 구성하기 위하여 먼저 필요한 것은 여러 곳에 분산된 자원들에 대한 metadata의 효율적 유지와 관리, 그리고 접근 서비스를 제공하는 것인데 Globus Toolkit 내에서는 MDS가 그 역할을 대신하고 있다. MDS는 앞서 언급한 LDAP을 기반으로 자원의 발견, 검색, 모니터링, 서비스에 대한 availability 등을 제공하고 있다.

MDS는 기본적으로 두 가지 서비스, GRIS(Grid Resource Information Service)와 GIIS(Grid Index Information Service)로 나뉜다. GRIS는 각각의 로컬 자원에 대해 주기적으로 자원에 대해 메타데이터를 유지하고 GIIS 서버에 접속하여 자신의 존재를 알림과 동시에 자신의 정보를 GIIS에 등록시킨다. 또한 GIIS는 GRIS를 트리 형태로 구성하여 이루어지고, GRIS를 통해 제공되는 정보들을 모아 가상의 통합된 정보 서비스를 제공하고 사용자는 GIIS로부터 분산된 자원의 상태를 종합하여 볼 수 있는 한편 특정한 하드웨어 사양을 갖고 있는 호스트에 대한 리스트 정보를 얻어올 수도 있다.

3. Design of Resource Management

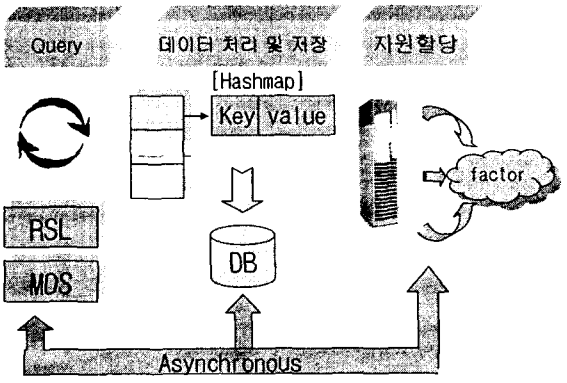
3.1 Composition of Fundamental Environment



[그림 1]

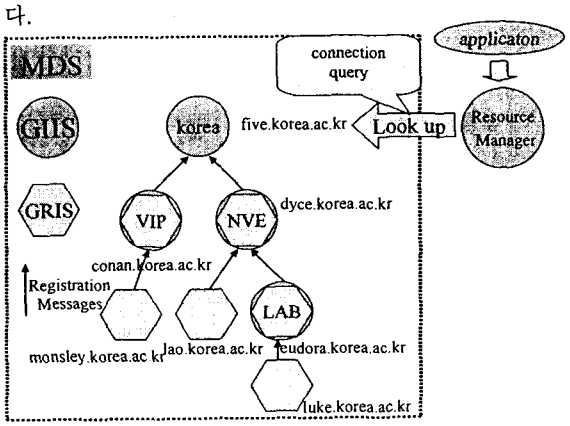
[그림 1]은 Resource Manager의 Information을 제공하는 방법에 대한 개괄적인 그림을 잘 나타내 주고 있다. Resource Manager는 Broker에게 RSL script를 넘겨주고 Broker는 이 RSL 스크립트를 받아 MDS를 통해 발견한 자원 정보를 토대로 여러 개의 세부 작업으로 분할하여 DUROC에게 넘기면 DUROC은 세부 작업들을 나누어 각각의 자원들로 할당해 동시에 실행시킨다.

[그림 2]에서는 본 논문에서 실제로 다루는 부분을 나타낸다. 전체적으로 Client가 RSL이나 MDS를 이용하여 자원 정보를 query하고 그 정보들을 적절한 알고리즘 및 DB를 이용하여 저장하였고 또한 자원의 각 요소에 대해



[그림 2]

factor값을 부가함으로써 실질적인 자원 할당이 가능하게 구현하였다. 이 세 가지 동작은 dynamic한 환경의 특성으로 인해 asnchronous한 쓰레드 방식을 이용하여 구현해 보았다. 먼저 MDS의 설정은 다음과 같이 구성하였다.



[그림 3]

[그림 3]에서 가장 크게는 3부분으로 나누어지게 되는데 먼저 사용자 혹은 application과 각각의 GRIS의 정보를 통합하고 있는 GIIS 서버, 그리고 사용자가 GIIS로부터 쿼리한 결과를 저장 및 관리하는 Resoure Manager(RM)로 구성된다. 또한 [그림 3]은 실질적인 테스트로 기반으로써 앞에서 언급한 MDS를 이용하기 위하여 다음과 같이 7대의 host로 GIIS 및 GRIS 환경을 구축한 것을 보여주고 있다. 이것은 기본적으로 Tree 형태로 나타나고 있으며 GIIS 서버에 의해 모든 host의 리소스 정보들은 관리, 제공되고 있다.

[그림 3]에서 VIP, NVE와 LAB host는 GRIS로써 로컬 리소스에 대해 메타데이터 및 GIIS 서버로써 하위 노드 host의 정보 또한 가지고 있다. 사용자는 우선 GIIS 서버에 접속하여 모든 노드의 host 정보를 가져오거나 혹은 특정한 하드웨어 사양을 갖고 있는 host들의 정보만을 가져올 수 있다. 이후에 사용자는 그 정보를 바탕으로 job을 할당하거나 수행시킬 수 있다. 기본적으로 사용자는 특정 자원의 타입, 노드 개수, 메모리 등의 리소스에 대한 정보를 RSL로 표기하여 보내게 되며, RSL(Resource Specification Language)은 Globus의 컴포넌트 사이에 주고 받는 메시지의 기본 형식을 나타낸다. RSL은 <Attribute, Value> 쌍의 리스트로써 정보를 표현하며

간략한 예로는 다음과 같다.

```
&(Mds-Os-name=Linux)(Mds-Cpu-
model=Pentium *)(Mds-Cpu-speedMHz>=500)
```

이 문장은 OS를 리눅스로 사용하고 CPU의 속도는 500MHz 이상인 펜티엄 CPU를 가진 host의 리스트를 보여주게 된다.

3.2 Implementation

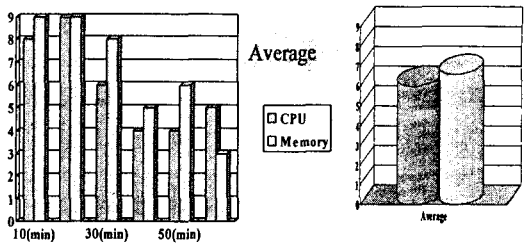
내부적으로 수행되는 실질적인 과정을 열거하면, 사용자는 GIIS 서버, 즉 LDAP 서버에 hostname과 port Number, base-dn(distinguish name) 정보를 주어 접속하게 된다. 성공적으로 접속이 이루어지면, 사용자는 RSL을 이용하여 filter정보를 입력하게 되고 실제 LDAP API의 Search를 이용하여 filter 정보를 만족하는 각 host의 모든 정보들이 하나의 entry 형태로 주어지게 된다. 여기에서 리턴된 entry 값을 내부적으로 각각의 호스트와 Attribute 별로 값을 매칭시키고 각각의 host에 대한 리소스들을 hashmap 자료구조를 이용하여 저장하였다. 일반적으로 map은 일반 정수를 인덱스로 사용하는 대신에 Key type을 갖고 또한 그 키와 연관되는 데이터 타입을 쌍으로 가진다. 일반적인 맵 구조에 hash function을 첨가한 hashmap은 Key type의 키 값을 이용하여 연관된 데이타를 다른 자료구조에 비해 신속하게 찾아낼 수 있다. 여기에서 키 값으로는 호스트 명을 사용하여 키 값의 중복을 배제시키고 hostname에 의한 검색을 할 때 보다 빠르게 Host에 대한 리소스들을 사용자에게 보여줄 수 있도록 하였다. 또한 이렇게 미리 저장된 Attribute들에 대해서 각각에 대해 사용자의 요구사항에 따라 factor 값을 입력 받도록 하였다.

```
    0  factor  1
    factor1 + factor2 + … + factorN= 1
ex) Mds-Cpu-speedMHz × 0.7 + Mds-Memory-
Ram-Total-freeMB × 0.3 = Value
```

위의 표에서 보는 바와 같이 입력받은 factor 각각의 값들은 0과 1사이이고, factor들의 합은 1을 만족해야 한다. 위의 예제는 CPU의 speed와 가용 메모리간의 비중을 7:3으로 할당한 것을 보이고 있고 Memory의 공간보다는 CPU의 speed가 우선시 됨을 알 수 있다. 사용자가 앞으로 job을 할당함에 있어 우선시 되는 Attribute들에 대해 위에서 보는 바와 같이 factor값을 입력하게 하고 factor와 Attribute의 곱을 더한 최종 값을 기준으로 정렬시켜 하나의 테이블로 생성하여 사용자에게 보다 쉽게 정보들을 보여 줄 수 있도록 하였다.

앞에서 잠깐 언급한 바와 같이, 네트워크 상의 리소스들은 dynamic하게 변하고 있다. 즉, 한 번 얻어진 리소스들의 정보는 계속 이어지는 것이 아니라 실시간으로 바뀌게 된다. 어떤 job에 의해 Memory가 full이 되었더라도 job이 끝나게 되는 순간 여유 Memory가 생겨날 것이다. 그렇게 된다면 랜덤하게 바뀌는 리소스들에 대해서 factor값은 큰 의미를 잃게 된다. DBMS를 이용하여 이러한 문제점을 다소 해결하려고 하였다.

[그림 4]의 보는 표와 같이 먼저 일정 시간 간격을 두고 네트워크 상의 리소스들을 주기적으로 저장하였다. 그리고 마지막에는 저장된 정보들에 대한 평균값에 대해 factor를 계산함으로써 dynamic한 환경에 따른 특징을 다소 나마 반영하였다.



[그림 4]

4. 결론 및 향후 과제

본 논문에서는 Grid상에서 자원 정보를 요구하는 query request에 대한 response 시간이 길다라는 문제점으로 주기적인 query가 힘들기 때문에, 이를 해결하는 방법으로 query를 담당하는 부분과 query에 대한 자료를 처리하고 저장하는 부분, 그리고 자원을 할당하는 부분의 세 부분의 Asynchronous한 방법으로 처리하도록 만들었다. 우리는 위와 같은 시스템을 가지고 차후에 Network Virtual Environment (NVE)의 시스템에 embed되도록 할 것이다. 지금까지의 구현에서는 자료의 처리만 있고 이 자료를 활용하는 면이 빠져있다. 차후에 이를 위해 데이터의 관리 및 처리를 위한 Data Manager를 만들 예정이다.

참고 문헌

[1] Globus Project, http://www.globus.org
[2] KISTI Grid Testbed, http://gridtest.gridcenter.or.kr
[3] Gregor von Laszewski and Ian Foster, "Usage of LDAP in Globus"
[4] Ian Foster, " A Resource Management Architecture for Metacomputing Systems"
[5] Ian Foster, " Grid Information Services for Distributed Resource Sharing"
[6] Ian Foster, " A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation"
[7] Junwei Cao, " Agent-Based Grid Load Balancing Using Performance-Driven Task Scheduling"
[8] S. Fitzgerald," Directory Service for Configuring High-Performance Distributed Computations"