그리드 미들웨어 : 자원 관리 및 원격 데이터 접근 기술 동향

서울시립대학교 김학두 · 김진석

1. 서 론

컴퓨터 자원들이 계속적으로 대용량화, 고속화되고 있는 추세이며 기상기후예측, 생명공학, 유전공학, 유체역학 등 여러 과학 분야에서는 단일자원으로서는 제공하기 힘든 계산 및 저장자원을 요구하고 있다. 미국 NASA의 IPG(Information Power Grid), 유럽의 EURO GRID 등 세계 각 국에서는 이러한 요구를 충족시키기 위하여 지역적으로 분산되어 있는 자원을 연결하여 단일 자원처럼 사용하기 위한 연구가 활발하게 진행되고 있다.

지역적으로 분산되어 있는 자원을 단일자원처럼 사용기 위해서는 보안, 스케줄링, 자원 할당, 자원 결합, 자원 정보 수집 등의 문제를 해결해야 한다. 현재 이러한 문제를 해결하기 위하여 그리드 미들웨어에 대한 많은 연구가 수행되고 있다. 본 고에서는 그리드 미들웨어 중 자원 관리 기술 및 원격지에 분산되어 있는 데이터를 접근하는 기술에 대하여 살펴본다. 특히 자원 요청의 구조, 자원 요청 기술 형식, 자원의 할당, 원격 데이터의 접근 등의 기술에 대해서 살펴보도록 한다.

2. 자원 관리 시스템

GRID는 지리적으로 분산된 고성능, 대용량의 자원들과 최단 장비들을 원격에서 동시적으로 사용할 수 있도록 한다는 목적으로 설계되었다. GRID 환경에서의 자원 관리 시스템은 다음과 같은 사항을 고려하여 설계되어야 한다[1].

1) 자원이 본질적으로 다른 관리자 영역에서 다른 조직체에 의해서 소유되고 운영된다.
2) 서로 다른 지역 자원 관리 시스템을 사용한다.
3) 가능한 한 자원을 제공하는 지역의 정책을 수정하지 않고 참여할 수 있어야 한다.
4) 동시에 다수의 자원을 요청하는 응용에 대한 고려가 있어야 한다.
5) Job이 실행중에도 자원의 요청과 특성이 변경될 수 있다.

이러한 요구사항들을 해결하기 위해 대표적인 그리드 미들웨어인 Globus[4]에서는 지역 자원 관리자(Local Resource Manager), 자원 중개자(Resource Broker), 그리고 자원 동시 할당자(Resource Co-allocator)와 같은 구성요소들과 각 구성요소들간의 정보를 교환하기 위해서 RSL(Resource Specification Language)이 사용된다.

기존의 자원 관리 시스템으로는 Network Batch Queueing System인 NQE[5], CODINE[7], LSF[28], PBS[9], LoadLeveler[10] 등이 있고 Wide-Area Scheduling System인 Callop[13], Legion[8], PRM[12], Condor[11] 등이 있다. 또한 Globus 미들웨어는 GRID 환경을 고려한 보다 효율적인 자원 관리 시스템을 제공하고 있다. 본 고에서는 Globus 기반 자원 관리 시스템에 대해서 살펴본다.

3. Globus의 자원 관리

Globus 시스템에서 자원을 관리하는 전체 구성도는 다음과 같다[1].

그림 1과 같이 Globus는 자원 중개자, 동시 할당자, 지역 자원 관리자로 이루어져 있다. 전체적인 요청을 간략히 살펴보면 응용 프로그램에서의 자원 요청은 중개자에게 전달되고, 각 특성에 맞는 중개자는 정보 서비스(Information Service)를 참고하여 자원 요청을 특정 지역 자원을 구별할 수 있을 때까

지 세분화시킨다. 이 과정에서 다수의 중개자가 참여할 수 있으며 세분화를 위해 다른 중개자에게 요청을 전달되기도 한다. 이렇게 세분화된 요청이 다중 자원을 요구할 경우, 자원 요청은 동시 할당자에게 전달되며 동시 할당자는 자원 요청에 명시된 지역 자원 관리자에게 전달한다. 지역 자원 관리자는 요청 받은 자원에 대해서 프로세스를 생성하고 그에 해당하는 job handler를 반환한다. 자원 요청을 명시하고 각 요소간 의사를 소통하는 수단으로는 RSL이 사용된다[1].

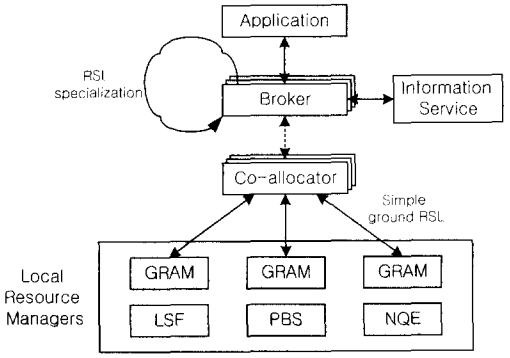


그림 1 GRID의 자원 관리 구조

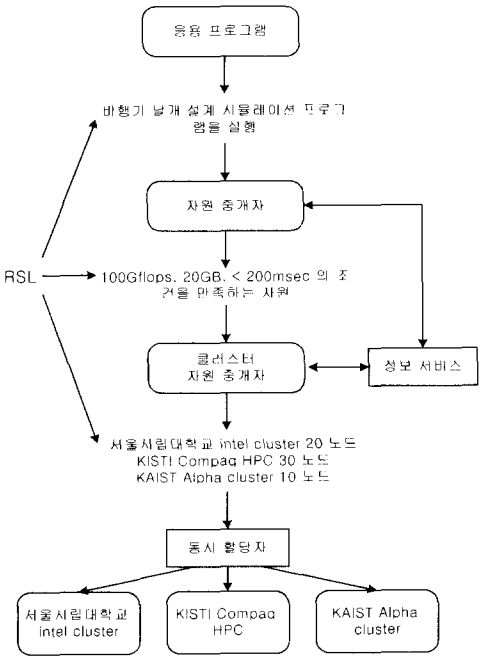


그림 2 자원 요청 흐름 예

그림 2는 Globus에서 자원 관리를 적용하는 예를 보이고 있다. 그림의 흐름을 살펴보면 다음과 같다.

1) 사용자가 비행기 날개 설계 시뮬레이션 프로그램을 GRID 환경에서 수행하기 위하여 자원 중개자에게 자원을 요청한다.
2) 자원 중개자는 응용 프로그램을 수행시킬 수 있는 자원을 찾기 위하여 정보 서비스에게 문의한다.
3) 정보 서비스로부터 받은 자원 정보를 토대로 자원 중개자는 서울시립대학교의 Intel-Cluster 20 노드, KISTI의 Compaq HPC 30 노드, KAIST의 Alpha-Cluster 10 노드를 선택하게 된다.
4) 클러스터 중개자는 다중 자원 요청을 하기 위해 동시 할당자에게 각각의 Sub-job으로 이루어진 요청을 전달한다.
5) 동시 할당자는 각각의 Sub-job을 세 기관의 지역 자원 관리자에게 전달한다.

4. 자원 명세 언어

Globus에서 사용되는 자원 명세 언어인 RSL (Resource Specification Language)은 자원 요청을 기술하는데 사용되며 Globus를 구성하는 구성요소간 의사 소통을 위해 사용된다. RSL의 문법은 그림 3에 나타나 있다

```
specification   :- request
request         :- multirequest | conjunction |
                   disjunction | parameter
multirequest    :- + request-list
conjunction     :- & request-list
disjunction     :- | request-list
request-list    :- ( request ) request-lit | ( request )
parameter       :- parameter-name op value
op              :- = | > | < | >= | <= | !=
value           :- ( [a..Z][0..9][ ])+
```

그림 3 RSL 문법[1]

다음은 그림 3의 문법을 토대로 RSL를 사용한 예이다.

```
& (executable = sort)
  (directory = /home/nobody )
```

```
(arguments = arg1 arg2)
(count = 1)
```

위 RSL은 실행 프로그램이 sort이고 Job Manager에 의해서 사용될 디렉토리는 /home/nobody, 매개변수는 arg1과 arg2, node는 1개를 요구하는 리스트로 구성되어 있으며 요구사항 모두를 만족해야 한다는 의미로서 &연산자(conjunction)를 사용하여 리스트를 결합하였다. 이 RSL에 사용된 모든 attribute들은 지역 관리자에 대한 것들로 구성되어 있다.

다음은 그림 2의 예에서 나타난 요청을 RSL로 표현한 것이다.

```
+(&(excutable=aircraft_test)
   (count=20)
   (memory>=64)
   (resourcemanager=grid.uos.ac.kr:2200))
 (&(excutable=aircraft_test)
   (memory>=64)
   (count=30)
   (resourcemanager=grid.hpcnet.ne.kr:755))
 (&(excutable=aircraft_test)
   (count=10)
   (memory>=64)
   (resourcemanager=grid.kaist.ac.kr:3010))
```

위 RSL은 +연산자를 사용하여 다중요청(multi-request)을 표현하고 있으며 구체적으로 특정 자원 관리자를 가리키고 있는 것을 볼 수 있다. 이 RSL은 중개자에 의해서 더 이상 구체화되지 않고 동시 할당자에게 전달되며 동시 할당자는 다중요청을 세부분의 단일요청으로 분리하여 각각에 해당하는 지역 자원 관리자인 GRAM에게 전달하게 된다.

5. 자원 중개자

자원 중개자(Resource Broker)는 응용 프로그램으로부터 받은 추상적인 요청을 구체적인 요청으로 변환하는 일을 한다. 즉, 자원 중개자는 자원에 대한 정보를 요약하는 작업과, 자원을 선택하는 작업 등을 수행한다. 중개자는 이러한 역할을 수행하기 위해 정보 서비스의 한 형태인 MDS(Metacomputing Directory Service)에게 자원에 대한 정보를 문의한다. 자원 중개자는 요청의 성격에 따라 여러가지 종류가 있다. 자원 중개자의 예로는 거대한 모수 실험(Parametric Experiments)의 생성과 관리를 위한 Nimrod-G[16]와 계산 수학에서 Large Loosely Coupled Problem에 사용되는 AppLeS[15], 가시적인 요소를 가미한 GRS(Graphical Resource Selector)[1] 등이 있다. 자원 중개자가 구체화시킨 요청이 다중 자원을 요구할 경우에는 그 요청을 동시 할당자에게 전달하고, 단일 요청일 경우에는 그 요청을 직접 GRAM에게 전달한다.

6. 자원 동시 할당

응용에 의해서 생성된 자원 요청이 만약 단일 자원을 요구하는 요청일 경우에는 직접적으로 자원 관리자에게 전달되지만, 여러 자원을 동시에 요청할 경우에는 자원 중개자에 의해서 다중요청이 만들어지고 동시 할당이 이루어진다. Globus에서는 DUROC (Dynamically Updated Request On-line Co-allocator)[3]라는 구성요소를 이용하여 다중 요청을 처리하게 되는데, 이 동시 할당자는 요청을 여러 구성요소로 분리하고 각 요소들을 적당한 자원 관리자에게 전달하며 전달된 요소들의 상태를 감시하고 작업들을 중단하는 등 총체적으로 관리하는 역할을 수행한다. Globus는 all-or-nothing방식을 사용한다. 즉 전체를 할당할 수 있을 때 할당하며 만약 하나라도 문제가 발생하면 모든 할당을 중지하는 방식을 사용하여 테스트되었다. 하지만 이러한 방식은 효율성의 문제가 제기되고 있으며 현재는 실패한 할당에 대해서 동시 할당자가 밖의 자원 중개자에게 재 전송하게 되면 자원 중개자는 이를 수정하여 다시 동시 할당자에게 전달하는 방식을 연구하고 있다. 또한 GRAM은 예약 기능을 사용하여 미래의 특정 시간에 자원을 사용할 수 있게끔 지원하고 있으나 예약 기능을 지원하지 않는 지역 자원 관리 시스템으로 인하여 제한성을 가지고 있다.

7. 지역 자원 관리자

GRAM(Globus Resource Allocation Manager)에 전달된 RSL 요청은 그 요청에 적합한 지역 자원을 인식할 수 있을 만큼 구체화되어 있다. GRAM은 Globus에서 지역 자원을 관리하는 책임을 지고 있으며, 지역 사이트의 자원 관리자들(LSF, CODINE, NQE, PBS 등)과 광범위 메타 컴퓨팅 환경 사이에서 인터페이스 역할을 수행한다고 볼 수 있다. GRAM

은 사용자와 자원간의 상호 인증, 지역 사용자 관리, Job Manager의 실행 등의 역할을 담당하는 Gatekeeper와 실제 프로세스를 생성하는 역할을 담당하는 Job Manager, 스케줄러의 현재 상황이나 정보를 나타내는 MDS의 구성요소로 구성되어 있다. 그 전체적인 흐름과 구성은 그림 4에 나타나 있다.

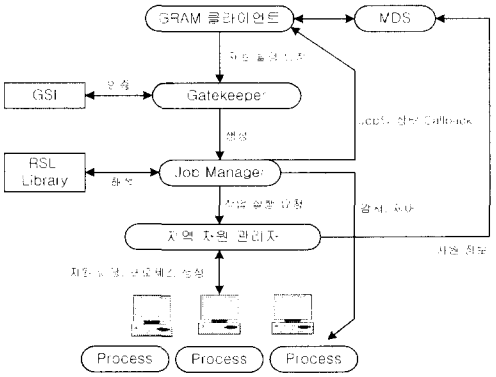


그림 4 GRAM의 구성 요소

그림 3에 나타나 있는 전체적인 흐름을 살펴보면 위에서 살펴보았듯이 응용 프로그램에서 생성된 요청은 자원 중개자에 의해서 특정한 자원을 지칭하는 요청으로 구체화되고 그 요청은 GRAM에게 전달된다. 첫 번째로 거치는 과정은 바로 인증 과정으로써 Gatekeeper는 GSI(Globus Security Infrastructure)를 호출하여 인증 과정 수행하고 인증 과정을 성공적으로 마치고 나면 Job Manager를 생성하게 된다. 생성된 Job Manager는 RSL 라이브러리를 기초로 RSL로 이루어진 요청을 해석하고 지역 자원 관리자를 통해서 프로세스를 생성하게 되며 그 프로세스를 감시한다.

8. 원격 데이터 접근

GRID 환경에서 응용 프로그램은 원격의 데이터를 엑세스하고 이동하는 경우가 종종 발생한다. 이러한 I/O서비스를 제공하는 데 요구되는 사항들은 다음과 같다[2].

1) 파일 접근에 대한 균일한 방식 제공
2) 스트림 I/O 서비스 지원
3) 변경이 잦은 자원들의 수용
4) 응용 프로그램의 최소한의 유지 비용
5) 응용 프로그램 프로그래머에 의한 전략 재정의 지원

Globus는 GASS를 이용하여 이러한 요구사항을 만족하는 I/O 서비스를 지원하고 있다. Unix나 표준 C를 사용하는 응용 프로그램들은 원격의 데이터를 접근하기 위하여 GASS API(Application Program Interface)를 사용할 수 있다.

GASS에서 지원하는 I/O 패턴은 그림 5에 나타나 있다. 현재까지 GASS에서는 파일에 쓰기 접근과 읽기 접근 연산을 동시에 수행하거나, 동시에 여러 개의 쓰기 접근 연산을 수행하거나, 파일의 일부분에 대해서만 읽기 연산을 수행하는 패턴을 지원하지 않고 있다.

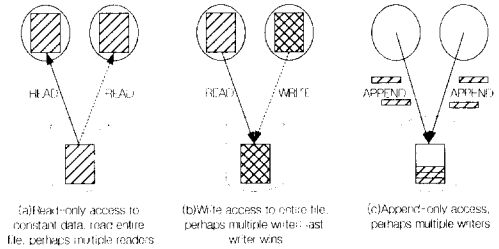


그림 5 GASS가 지원하는 I/O 패턴[2]

GASS는 캐쉬 기능을 사용하여 네트워크 대역폭을 제어할 수 있다. GASS는 응용 프로그램이 원격지 파일을 열고자 할 때, 캐쉬를 확인하여 그 파일이 캐쉬에 존재하는 경우에는 캐쉬의 데이터를 사용하고 reference count값을 증가시킨다. 그 파일이 캐쉬에 존재하지 않을 경우에는 원격지의 파일 전체를 열고 reference count의 값에 1을 기록한다. 그림 6은 GASS의 구조를 나타내고 있다.

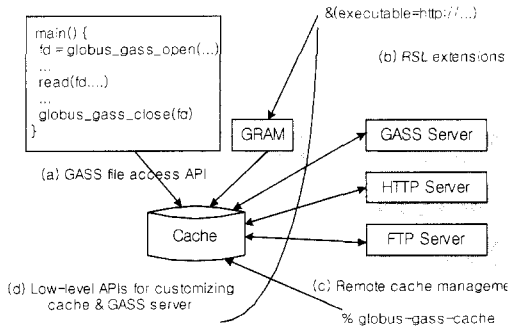


그림 6 GASS의 구조[3]

그림 6에서 볼 수 있듯이 현재 GASS는 FTP, HTTP, GASS server를 이용하여 데이터를 접근 할 수 있다. GASS는 캐쉬 기능을 사용하기 때문에 캐쉬에 적재되어 있는 파일을 언제 원격지에 이동시킬 것인가에 대한 결정 전략을 필요로 한다. GASS는 마지막으로 쓰기 연산을 한 프로세스가 파일을 닫을 때 원격지에 파일을 전송하고 캐쉬의 내용을 비우는 방법을 사용하고 있다. 반면, 원격지 파일이 추가 모드로 열렸을 경우에는 캐쉬에 파일을 적재할 필요 없이 스트림을 통하여 원격지의 파일을 만들거나 추가할 수 있게 하고 있다. 또한 GASS는 프로그래머로 하여금 기본적인 데이터 이동에 관한 전략을 수정하고 관리할 수 있도록 하는 메커니즘을 제공하고 있는데, 대용량의 데이터에 유용하게 사용될 수 있는 고수준의 메커니즘과 파일이 언제 캐쉬되는지에 대한 것 뿐만 아니라, 어디에 저장되는지를 명시하는 등, 다양하게 캐쉬의 행동을 제어할 수 있는 저수준의 메커니즘을 제공한다.

Globus의 GASS의 API[3]는 크게 4개의 API로 나눌 수 있으며 각각의 API는 다음과 같다.

1) GASS의 캐쉬를 관리하는 API : 캐쉬의 조작에 관련된 호출 정의
2) 파일 접근 관련 API : Unix의 open(), close(), C 언어의 fopen(), fclose()등의 호출 대체
3) 클라이언트 관련 API : 서버와의 요청 송수신과 관련된 호출 정의
4) 서버 관련 API : 클라이언트와의 요청 송수신과 관련된 호출 정의

9. 맺음말

최근 여러 컴퓨터 자원을 병렬로 결합하여 대규모 고성능 컴퓨터 시스템을 구현하는 GRID 기술에 대한 연구가 활발하게 진행되고 있다. GRID 시스템을 이용하면 기존에는 해결하기 어려웠던 대규모 병렬/분산처리 문제들을 해결할 수 있다. 본 고에서는 대표적인 GRID 미들웨어인 Globus에 대한 기술현황을 살펴보았다. 특히 Globus에서 제공하고 있는 자원 관리 시스템과 원격지에 있는 데이터에 대한 접근기술을 살펴보았다. 향후 자원 관리 시스템에는 사용자가 수행시키고자 하는 응용 프로그램이 어디에서 수행되어야 가장 효율적으로 수행될지 판단할 수 있는 Globus Queueing에 대한 연구와 효율적인 정보 서비스에 대한 연구가 이루어져야 할 것으로 생각된다.

참고문헌

[1] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke, "A Resource Management Architecture for metacomputing Systems," *Proc. of the 4th Workshop on Job Scheduling Strategies for Parallel Processing*, 1998.

[2] J. Bester, I. Foster, C. Kesselman, J. Tedesco, and S. Tuecke, "GASS: A Data Movement and Access Service for Wide Area Computing Systems," *Proc. of the 6th Workshop on I/O in Parallel and Distributed Systems*, 1999.

[3] http://www.globus.org

[4] I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit," *International Journal of Super computer Application*, vol.11, no.2, pp.155-128, 1997.

[5] Cray Research, Document Number IN-2153 2/97, 1997.

[6] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, and S. Tuecke, "A Directory Service for Configuring High-performance Distributed Computations," *Proc. of the 6th IEEE Symp. on High Performance Distributed Computing*, pp.365-375, 1997.

[7] GENIAS Software Gmbh, "CODINE: Computing in Distributed Networked Environments," 1995.

[8] A. Grimshaw, W. Wulf, J. French, A. Weaver, and P. Reynolds, Jr., "Legion: The Next Logical Step toward a Nationwide Virtual Computer," Technical Report CS-94-21, Department of Computer Science, University of Virginia, 1996.

[9] R. Henderson and D. Tweten, "Protable Batch System : External Reference Specification." Technical Report, NASA Ames Research Center, 1996.

[10] International Business Machines Corporation, Kingston, NY, "*IBM Load Leveler: User's*

Guide," September 1993.

[11] M. Litzkow, M. Livny, and M.Mutka, "Condor - a hunter of idle workstations," *Proc. of the 8th Intl Conf. on Distributed Computing Systems*, pp.104-111, 1988.

[12] B. C. Neuman and S. Rao, "The Prospero Resource Manager: A Scalable Framework for Processor Allocation in Distributed System," *Concurrency: Practice Experence*, vol.6 no.4, pp.339-355, 1994.

[13] J. Weissman, "Gallop: The Benefits of Wide-area Computing for Parallel Processing," Technical Report, University of Texas at San Antonio, 1997.

[14] S. Zhou., "LSF: Load Sharing in Large-scale Heterogeneous Distributed Systems," *Proc of the Workshop on Cluster Computing*, 1992.

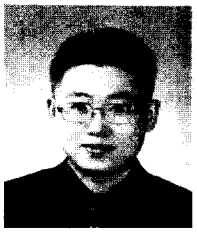
[15] F. Berman, R. Wolski, S. Figueira, J. Schopf, and G. Shao, "Application-level Scheduling on Distributed Heterogeneous Networks," *Proc. of the Supercomputing '96*, 1996.

[16] D. Abramson, R. Sosic, J. Giddy, and B. Hall, "Nimrod: A Tool for Performing Parameterised Simulations using Distributed Workstations, *Proc. of the 4th IEEE Symp. on High Performance Distributed Computing*, 1995.

김 학 두

1994. 3~2001. 2 서울시립대학교 컴퓨터·통계학과(학사)
2001. 3~현재 서울시립대학교 석사과정
관심분야(GRID), 병렬처리, 클러스터링 시스템
E mail:ncomidas@sidae.uos.ac.kr

김 진 석



1986~1990 한국과학기술대학 전산학 학사
1990~1992 KAIST 전산학 석사
1992~1997 KAIST 전산학 박사
1997~1998 미국 M.I.T. Laboratory for Computer Science Postdoc Fellow
1998~1999 전자통신연구원 (ETRI) 슈퍼컴퓨터센터 초빙 연구원
1999~현재 서울시립대학교 컴퓨터통계학과 조교수
관심분야 병렬처리 시스템, (GRID)컴퓨팅, 멀티미디어
E mail:jskim@venus.uos.ac.kr