범용 그리드 데이터 시스템의 구현

이상근, 황석찬, 최재영
숭실대학교 컴퓨터학부
{ seventy9,schwang }@ss.ssu.ac.kr, choi@computing.ssu.ac.kr

Implementation of Universal Grid Data System

Sangkeon Lee, Seogchan Hwang, Jaeyoung Choi
School of Computer, SoongSil Univ.

요 약

90년대 중반에 등장한 그리드(GRID)는 지역적으로 분산되어 있으며 소유자가 서로 다른 다양한 컴퓨팅 자원의 효과적인 공유를 목적으로 하며, 시스템의 구축에 필요한 스케쥴링, 자원 관리, 보안, 성능 측정 및 상태 모니터링 등의 문제를 해결하기 위한 다양한 미들웨어 및 개발 도구가 연구되고 있다.

그리드 시스템에서 사용되는 데이터는 FTP 서버, 파일 시스템, 데이터베이스 등 여러 장치에 저장되어 개별적인 인터페이스를 통해 접근된다. 각각의 인터페이스를 통하여 접근하므로 사용 방법이 어렵고 확장성이 떨어지는 단점이 있다.

이러한 문제점을 해결하기 위하여, 본 연구에서는 데이터베이스, FTP, 파일 시스템에 산재한 그리드 데이터를 동시에 지원하고, URL을 통해 단일 인터페이스로 접근하게 해 주며, 3계층 구조와 데이터 캐시를 이용하여 성능과 확장성을 제공하는 시스템을 구현한다.

1. 서 론

기존의 병렬 및 분산 시스템은 계산 성능 향상 및 시스템의 고가용성에는 효과적인 적용이 가능했으나, 여러 곳에 분산되어 소유자가 서로 다른 다양한 종류의 컴퓨팅 자원을 효과적으로 통합하기에는 어려움이 있었다.

90년대 중반에 등장한 그리드(GRID) 시스템은 다양한 컴퓨팅 자원을 효과적으로 공유 할 수 있는 수단을 제공하며, 기존의 분산 시스템에 비해 확장된 규모의 자원 공유와 높은 성능을 지원하고, 보다 혁신적인 응용프로그램의 개발을 위한 토대를 제공한다.[1]

이러한 특징으로 인하여, 그리드 시스템은 현재 다양한 과학 및 공학 분야에 적용을 위한 시스템의 개발에 널리 연구되고 있으며, 시스템의 구축에 필요한 스케쥴링, 자원 관리, 보안, 성능 측정 및 상태 모니터링 등의 문제를 해결하기 위한 다양한 미들웨어 및 개발 도구가 연구되고 있다.

본 논문에서는 그리드 시스템을 지원하는 여러 도구 중 그리드 시스템에서 필요로 하는 데이터를 관리하는 미들웨어의 구현에 대하여 설명한다. 본 논문 이후의 부분에서 그리드 시스템에 사용되는 데이터를 "그리드 데이터(Grid Data)"로 지칭하였고, 이 데이터를 관리하는 시스템을 "그리드 데이터 시스템(Grid Data System)"으로 정의하였다.

2. 관련 연구

그리드 데이터는 주로 FTP 서버, 파일 시스템, 데이터베이스에 저장되는데, 현재까지 연구된 그리드 데이터 시스템들의 특징을 살펴보면 다음과 같다.

파일 시스템을 지원하는 그리드 데이터 시스템의 예로 DPSS(Distributed Parallel Storage System)[2]를 들 수 있다. DPSS의 경우 DPSS 서버가 병렬 디스크(Parallel Disk)를 관리하며, 클라이언트 프로그램과 DPSS 사이에 DPSS 마스터가 미들웨어 역할을 하는 3 계층 구조로 구현되어 있다. 높은 성능과 확장성을 제공하지만, 병렬 디스크 상의 파일 시스템에 존재하는 데이터를 지원한다.

FTP를 지원하는 그리드 데이터 시스템의 예로는 GridFTP[3]가 있다. 이는 FTP 프로토콜을 그리드 환경에 맞게 확장한 GridFTP 프로토콜을 정의하고 이를 구현한 것이다. GridFTP를 사용하기 위해서는 FTP 클라이언트에서 GridFTP 프로토콜 확장을 지원해야 한다.

그 외에, 그리드 데이터 시스템에 2차 저장 장치를 통합하기 위해 사용이 가능한 개발 도구로 글로버스(globus) 프로젝트의 GASS(Global Access to Secondary Storage) API[4]가 있다.

3. 범용 그리드 데이터 시스템

1) 범용 그리드 데이터 시스템

관련 연구에서 살펴 본 바와 같이 그리드 데이터 시스템 구축에 관한 여러 연구가 있었지만, 데이터베이스에 저장된 데이터를 통합하는 데에 사용할 수 있는 범용적인 도구의 부재로

데이터베이스 관련 부분은 각각의 프로젝트에서 직접 구현하고 있는 실정이다.

또한, 데이터베이스, FTP, 파일 시스템에 산재하는 그리드 데이터를 모두 처리할 수 있는 그리드 데이터 시스템의 부재로 통합하는 데이터의 종류에 따라 다른 도구를 사용해야 하는 불편함이 있었다.

범용 그리드 데이터 시스템(Universal Grid Data System)은 데이터베이스, FTP 서버, 파일 시스템 상에 산재된 그리드 데이터를 URL을 이용하여, 동일한 인터페이스를 통해 접근이 가능한 시스템을 구현한 것이다. 이 후의 부분에서는 편의상 범용 그리드 데이터 시스템을 UGDS로 표기한다.

UGDS가 지원하는 시스템은 POSIX 표준을 따르는 UNIX 시스템과 커널 버전 2.2 이상의 리눅스 시스템이며, 통신 수단으로 TCP/IP 프로토콜을 사용하고, 데이터베이스는 현재 유닉스 및 리눅스용 오라클만을 지원한다.

2) UGDS의 구조

UGDS의 전체적인 구조는 그림 1과 같고 UGDS 클라이언트 API , URL 브로커, 데이터 서비스 모듈(Data Service Module)의 3계층으로 구성된다. 이 후의 부분에서 UGDS 클라이언트 API는 UGDS API로 데이터 서비스 모듈은 DSM으로 표기한다.

먼저, URL 브로커는 그리드 데이터에 접근하는 UGDS 클라이언트 프로그램이 URL로 데이터를 요구하면 적합한 DSM을 찾아 그 데이터를 클라이언트 프로그램에 전송해 준다. 그리고, 데이터 서비스 모듈의 상태를 모니터링하는 모듈이 있어, 데이터가 현재 사용 가능한지를 클라이언트 프로그램에 알려준다.

다음으로, 데이터 서비스 모듈(DSM)은 URL 브로커가 요청한 데이터를 URL 브로커로 전송하는 역할을 한다. 지원하는 그리드 데이터의 종류에 따라 DB-DSM, FS-DSM, FTP-DSM의 세 가지가 존재하고 각각 데이터베이스, 파일 시스템, FTP 서버의 데이터를 관리한다.

마지막으로, UGDS API는 UGDS 클라이언트 프로그램 구현 시 URL 브로커의 통신 규약에 따라 직접 프로그램을 작성하기가 어려우므로, 이를 지원하는 클라이언트 API를 C언어로 구현한 것이다. 그림 1에서의 실제 데이터 요청과 전송은 클라이언트 라이브러리 함수의 호출에 의해 발생한다.

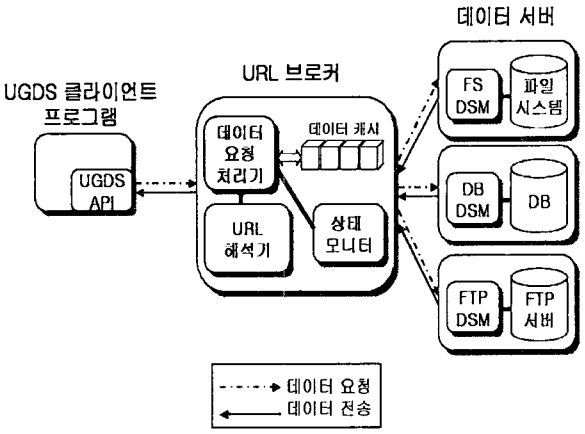

그림 1 UGDS의 구조

3) 데이터의 처리 과정

클라이언트 프로그램이 데이터를 요청하고 다시 클라이언트에 전송되기까지의 과정을 살펴보면 다음과 같다.

우선, 클라이언트 프로그램은 URL을 인자로 UGDS API를 호출한다.

다음으로, URL 브로커가 URL로 명시된 데이터 요청을 해석하고 만약 캐시에 데이터가 존재하면 데이터를 클라이언트 프로그램에 전송한다. 데이터가 존재하지 않을 경우 URL에서 DSM의 주소에 해당하는 부분을 추출한 다음 DSM에 데이터를 요청한다.

DSM은 URL 브로커가 데이터를 요청할 경우 이를 전송한다. 그 다음, URL 브로커는 캐시 내용을 갱신하고 클라이언트 프로그램에 전송한다.

4) UGDS를 이용한 시스템 구성의 예

UGDS를 이용한 시스템의 구성을 예를 들어 살펴보면 그림 2와 같다. 그림 2는 여러 개의 데이터베이스, 파일시스템, FTP 서버가 URL 브로커를 통해 클라이언트와 연결된 형태를 나타낸 것이다. 그림2에서 굵은 실선은 시스템의 구성 요소 간 네트워크 연결을 표시한 것이다. CA로 표시한 것은 그리드 시스템에서 작업을 수행하는 작업노드에서 동작하는 클라이언트 프로그램을 나타낸 것이다. BR로 표시한 것은 URL 브로커이다.

UGDS는 하나의 URL 브로커에 여러 클라이언트 프로그램이 접근 할 수 있고, 하나의 브로커가 여러 데이터 저장소에 연결될 수 있는 구조를 가진다. 성능의 확장이 필요한 경우, URL 브로커를 여러 개 사용하여 성능을 쉽게 확장 할 수 있고, URL 브로커가 미들웨어 역할을 하는 3 계층 구조를 사용하여 유연성 있는 시스템의 구성이 가능하다.

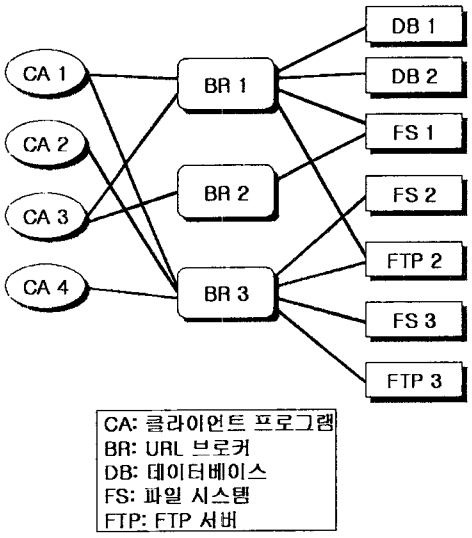

그림 2 UGDS를 이용한 시스템 구성 예

5) URL 표현 형식

UGDS에서 사용하는 URL 표현 형식은 일반적으로 사용하는 URL 표현 양식을 하며, 다만 데이터 베이스 상의 데이터를 지정하는 경우에는 DBMS가 설치된 서버의 주소, 테이블의 이름, 키로 사용될 칼럼, 키 값을 이용하여 *ugdsdb://<DB 서버 주소>/<테이블>/<키 칼럼>/<키 값>*의 형식으로 URL을 표현한다.

URL 표현 예)
ftp://data.testdb.com/testdata/test123,
ugdsdb://www.testd.com/test_protein/struct_id/122

6) 데이터 캐시

데이터 서비스 모듈에서 제공되는 모든 데이터는 클라이언트에 전송되기 이전에 항상 캐시를 거치는 구조로 되어있으며, URL 관리자가 데이터 서비스 모듈에 중복된 데이터 요청을 보내는 것을 방지하여, 성능이 향상된다.

URL 브로커의 캐시 공간이 부족할 때 가장 최근에 사용되지 않은 데이터를 먼저 제거한다.

4. 실험

실험을 위해 URL 브로커, FS-DSM, FTP-DSM, UGDS 클라이언트 프로그램을 각각 800Mhz 펜티엄3 PC에 설치하였고, DB-DSM과 DBMS는 1.7Ghz 펜티엄4 PC에 설치하였다. 각각의 PC에 메모리는 256MB가 장착되어 있고 LAN으로 연결되어 있다. 운영체제는 레드햇 리눅스 7.2, DBMS는 오라클 8.1.6을 사용하였고, FTP 서버는 레드햇 리눅스에 포함된 것을 사용하였다.

실험에 사용한 데이터는 화합물의 정보를 저장하는 mol2 형식의 파일로 평균 8K 정도 크기의 데이터 10000개씩 FTP 서버, 파일 시스템, 오라클 DBMS에 저장하였다.

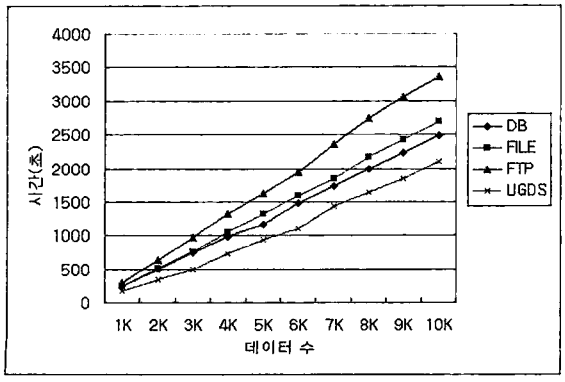

그림 3 실험 결과

그림 3에서 FILE, FTP, DB는 각각 파일 시스템, FTP 서버, 오라클 DBMS에 저장된 데이터를 전송하는데 소요된 시간을 측정한 것이고, UGDS는 FTP-DSM, FS-DSM, DB-DSM에 데이터를 3분의 1씩 나누어서 요청하였을 경우 소요된 시간을 측정한 것이다. 세 개의 DSM은 하나의 URL 브로커에 연결하였으며, 클라이언트 프로그램의 데이터 요청 횟수는 1000회부터 10000회까지 1000회씩 증가시키면서 실험하였다.

실험 결과, 데이터의 전송은 안정적이었고, UGDS를 사용한 경우 평균적으로 40% 정도의 성능 향상이 있었다. DSM 세 가지를 연결하는데 하나의 URL 브로커를 사용하였고, 단일 서브넷트웍에서 트래픽을 공유한 결과로 약간의 지연이 발생하였다.

5. 결론 및 향후 계획

UGDS를 사용하면 다양한 형태의 그리드 데이터를 손쉽게 효과적으로 통합할 수 있다. 파일시스템, FTP Server, 데이터베이스에 동일한 인터페이스로 접근할 수 있어, UGDS API 만으로도 세 가지 그리드 데이터를 손쉽게 처리할 수 있다. 또한, 이 세 가지 그리드 데이터를 단일 시스템에서 지원하므로, 두 가지 이상의 데이터 통합이 필요한 그리드 시스템을 구현하는 경우에도 구현의 어려움이 증가하지 않는다.

그러나, 현재의 구현으로는 데이터베이스 지원이 오라클 DBMS에 제한되어 있고, URL과 데이터가 정적으로 맵핑(mapping)되어서 사용자가 시스템을 설정하는데 제약이 있다.

차후, 위에서 언급한 문제점들을 해결하기 위하여 다양한 데이터베이스의 지원, 성능의 개선을 위한 통신 모듈과 데이터 캐시의 개선, 동적 URL 매핑 지원 등을 구현할 예정이다.

참고문헌

[1] Foster, I., Kesselman, C., Tuecke, S., "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", http://www.globus.org/research/papers/anatomy.pdf.

[2] Tierney, B., Johnston, W., Lee, J., Thompson, M., "A Data Intensive Distributed Computing Architecture for Grid Applications", Future Generation Computer Systems (an Elsevier Journal), vol 16, #5, April 2000, pp 473-481.

[3] Globus Project White Paper, "GridFTP: Universal Data Transfer for the Grid", http://www.globus.org/datagrid/deliverables/C2WPdraft3.pdf

[4] Bester, J., Foster, I., Kesselman, C., Tedesco, J., Tuecke, S., "GASS: A Data Movement and Access Service for Wide Area Computing Systems", Sixth Workshop on I/O in Parallel and Distributed Systems, May 5, 1999.