제 18 회 한국정보처리학회 추계학술발표대회 논문집 제 9 권 제 2 호 (2002.11)

GASS 기반 그리드 데이터 I/O 의 접근성 개선에 관한 연구

권오경, 박형우, 이상산
한국과학기술정보연구원 슈퍼컴퓨팅센터
e-mail : {okkwon, hwpark, sslee} @hpcnet.ne.kr

The Accessibility Improvement of Grid Data I/O based on GASS

Ohkyoung Kwon, Hyoungwoo Park, Sangsan Lee
Supercomputing center, Korea Institute of Science and Technology Information(KISTI)

요 약

인터넷 기반의 컴퓨팅 환경이 발전함에 따라 자원과 정보의 공유를 위한 그리드 컴퓨팅이 나타나게 되었다. 그리드 컴퓨팅에서는 그리드 응용 수행시 프로그램과 데이터간의 위치가 다르고 분산되어 있는 경우가 많다. 현재 그리드 미들웨어 시장 표준인 글로버스 툴킷(Globus Toolkit™)에서 사용하는 GASS 와 같은 그리드 I/O 시스템들이 원격에 있는 관련 데이터들을 로컬 시스템의 데이터에 접근하는 것처럼 처리하는 것을 지원한다. 그러나 GASS(Global Access to Secondary Storage)에서는 원격지의 파일 시스템의 접근시 URL 을 이용하는데, 그리드 환경에서의 I/O 을 위한 파일 시스템은 복수의 파일 형태로 분산되어있기 때문에 하나의 그리드 응용을 수행할 때 URL 을 동적으로 바꿔저야 하는 문제점을 갖고 있다. 본 논문에서는 URL 보다 상위 개념의 주소 체계인 URI 을 도입하고 이를 RSL 과 연계하여 분산 환경의 그리드 I/O 에서도 간을 주소 형태로 I/O 처리가 가능하는 등 효율적인 그리드 I/O 를 위한 GASS 개선 방안을 제시 한다.

1. 서론

인터넷 기반의 컴퓨팅 환경이 발전함에 따라 네트워크 연결 이상의 새로운 서비스 개념이 나오기 시작했다[1]. 그 중 대표적인 것 중의 하나가 자원과 정보의 공유를 위한 그리드 컴퓨팅이다. 사용자들이 이때까지 접하지 못했던 여러 슈퍼컴퓨터의 큰 자원을 그리드를 통해 얻을 수 있게 된다.

응용 프로그램은 데이터를 사용하거나 출력하는 일이 빈번하다. 그리드에서는 프로그램과 사용되는 데이터의 위치가 다를 경우가 발생한다. 전통적으로 데이터를 프로그램이 실행되는 동일한 저장 장소에 수동으로 옮겨 놓는 방식을 사용한다. 하지만 GASS 같은 I/O 시스템들은 원격 데이터에 대한 접근을 현재 로컬에 있는 것 같이 사용하게 한다[2]. 본 연구에서는 그리드 미들웨어의 사실상의 표준인 글로버스 툴킷 [3] 내에 포함된 GASS 의 문제점을 살펴보고 개선점을 살펴 본다.

GASS 는 URL(Uniform Resource Locator)을 사용해서 I/O 에 대한 인터페이스를 통일화 한다. GASS 에서는 *globus_gass_open, globus_gass_close* 같은 함수를 제공하는데 URL 형식으로 데이터를 바로 접근이 가능하다. 하지만 GASS 서버인 경우에는 실행 할 때마다 포트가 계속 바뀌고 FTP 서버인 경우에도 실행하는 위치에 따라서 주소를 계속 바꿔야 한다. 즉 한번 프로그램을 컴파일 한 후에 다른 곳에서 작업을 하기 위해선 다시 컴파일을 해야 한다. 본 연구에서는 이동성(portablility)을 보장하기 위해서 URL 보다 더 큰 범위의 URI(Uniform Resource Identifier)를 사용하기로 한다.[4][5] URI 에서는 URL 뿐만 아니라 특정한 장소가 아닌 리소스에 대한 이름을 정할 수 있는 URN(Uniform Resource Name)[6]이 있다. URN 을 통해서 파일 이름을 다시 URL 로 바꾸는 구조를 설계한다.

응용 프로그램에서 각 계산 노드마다 생성 되는 데

이터들을 하나로 묶어야 될 필요가 많다. 현재의 GASS 시스템에선 파일들을 따로 생성한 후에 사용자가 수동으로 묶어 줘야 한다. 본 논문에서는 기존의 GASS API 를 이용해서 여러 계산 노드에서 하나의 파일에 순서대로 자동으로 쓰는 구조를 설계한다.

논문의 구성은 다음과 같다. 먼저 2 장에서는 관련된 연구를 살펴본다. 3 장에서는 GASS 기반의 원격 데이터에 대한 I/O 시스템 설계를 하고 4 장에서는 3 장에서의 설계를 바탕으로 API 구현을 설명하기로 한다. 그리고 마지막 장에서 결론 및 향후 연구 방향에 대해서 기술한다.

2. 관련연구

Condor[7]에서는 영역(pool)안에 있는 모든 기계의 파일을 수동으로 옮길 필요 없이 접근 할 수 있다. 즉 원격에 데이터가 있어도 자신의 로컬 저장소에 있는 것처럼 사용할 수 있다. 원격 기계의 I/O 시스템 호출을 작업을 호출한 기계로 돌려주는 RPC(remote procedure call)로 바꾸는 방식으로 구성되어 있기 때문이다. GASS 는 로컬의 캐시를 통해 한번 가져와서 사용하는 반면 Condor 에서는 바로 사용하게 된다. 하지만 캐시가 지원이 안되고 분산된 여러 장소의 데이터는 사용자가 수동으로 옮겨 놓아야 하는 한계가 있다. 본 연구처럼 동적인 주소 체제를 통해 모으는 것이 쉽지 않다.

WebFS[8]와 UFO[9]는 GASS 처럼 URL 를 통해 파일 이름을 정함으로써 다른 파일 시스템에서의 접근을 가능하게 한다. WebFS 에서 UNIX 의 vnode 에 I/O 를 연결시키는 방법과 UFO 에서 UNIX 의 디버깅 트랩(trap) 인터페이스를 사용하는 방법은 이동성이 많이 떨어 진다. 본 연구에서 제시하는 방식을 Condor, WebFs,, UFO 에서도 적용시킬 수 있을 것이다.

3. 원격 데이터 I/O 설계

3.1. URI 기반 GASS 파일 접근 방법

C 언어의 경우에 파일에 대한 핸들을 얻기 위해서 *open* 이라는 시스템 호출(system call)을 통해서 이루어진다. 이것을 GASS 에서는 *globus_gass_open* 함수를 이용하는데 파일에 대한 주소를 URL 을 사용하여 나타난다. 인자에 관한 다른 옵션들은 *open* 과 같기 때문에 주소 부분만 바꿔주면 된다. *open*("a.txt", ...)과 같이 파일을 열었을 때 a.txt 는 로컬 저장소에 있는 것으로 생각해서 프로그램을 작성한다. 하지만 GASS API 를 사용하는 경우에는 다음과 같은 URL 의 형식으로 표현함으로써 로컬뿐 아니라 원격지의 장소까지 작성이 가능하다.

file:///PATH/a.txt
gsiftp://IPADDRESS/PATH/a.txt

하지만 URL 주소는 쉽게 변하기 때문에 변동적인 사항에도 대처할 수 있는 방법이 필요하다. 그래서 URL 보다 더 큰 범위의 URI(Uniform Resource Identifier)를 사용하기로 한다. URI 에서는 URL 뿐만 아니라 URN(Uniform Resource Name)를 사용이 가능하다. URL 이 위치 의존적인 설정을 해야 하는 반면 URN 은 위치에 상관 없이 동적으로 대응할 수 있는 특징이 있다.

URN 은 "urn:" <NID> ":" <NSS>과 같은 형태로 되어 있다. NID 는 네임스페이스에 대한 ID 이고 NSS 는 NID 에 해당하는 문자열이다. 여기서 URL 로 동적으로 변화하는 네임스페이스에 대한 NID 를 GridFTP 와 GASS 서버에 관한 두가지를 지원하기로 한다. NID 를 GridFPT 는 gridftp 라고 하고 GASS 서버는 gass 라고 하기로 한다. 로컬 저장소에 사용되는 파일 명을 NSS 에 넣기로 한다. 예를 들어 GridFTP 를 사용하기로 하고 로컬 저장소에 있는 파일명이 a.txt 이면 URN 은 urn:gsiftp:a.txt 이 된다.

컴파일시에 상위 노드에서 URN 주소인 urn:gsiftp:a.txt 로 했을 경우에 각 계산노드가 런타임시에 상위노드의 URL 주소인 gsiftp://IPADDRESS/PATH/a.txt 로 바꿔주게 된다. 이때 IP 주소와 작업이 이루어지는 디렉토리에 관한 값을 알아 내야 한다.

글로버스는 상위 노드에서 계산 노드로 작업을 던질 때 RSL(Resource Specification Language)을 통해서 한다. 그림 1 에서 보듯이 계산 노드는 RSL 의 environment 속성을 통해서 환경 변수에 관한 값들을 알 수 있다. 즉 상위 노드의 IP 주소와 작업이 이루어지는 디렉토리에 관한 사항을 RSL 을 통해서 계산 노드는 얻을 수 있다.

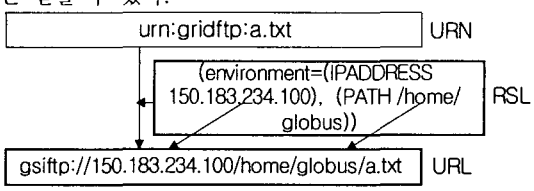

그림 1 URN 을 동적으로 URL 주소로 할당

그림 2 에서 실행 파일(execfile)과 입력 데이터인 a.txt 가 어떻게 각 계산 노드에서 작동하는지 살펴보자. 실행 파일은 GASS 의 스테이징(staging)기능을 통해서 자동으로 이동한다.

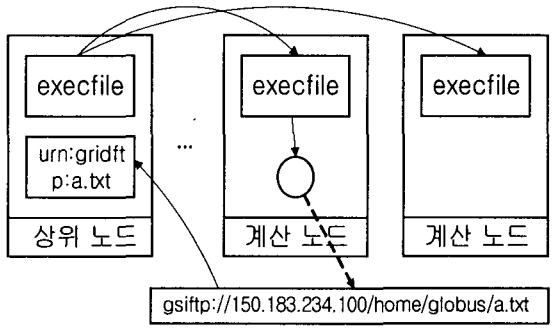

그림 2 실행파일과 입력파일이 각 노드들에서 작동하는 모습

첫 번째 계산 노드에서 a.txt 가 필요로 한다고 가정하자. globus_gass_open 함수를 이용해서 상위 노드에 있는 a.txt 를 접근해야 한다. URI 주소가 URN 형태이고 앞부분이 gridftp 라는 표시를 보고선 이 주소가 상위노드의 GridFTP 를 사용해서 가져와야 한다는 것을 알 수 있다. 그래서 주소를 URL 형태로 그림 1 처럼 바꾼다. *globus_gass_open* 함수는 URL 주소 형태로 각 계산 노드의 캐시에 데이터를 읽기 위해서 잠시 담아 두게 된다.

위에서 언급한 경우는 일반적인 분산 응용프로그램과 입력 파일의 위치가 같은 곳에 있는 경우이다. 하지만 앞에서 언급하였듯이 그리드가 활성화되면 될수록 데이터와 실행 파일의 위치가 달라지게 될 것이다. 이 경우에는 URI 의 부분집합인 URL 을 사용하면 된다. 그러나 URL 주소는 컴파일을 한 후에 변화 할 수 있다. 그림 3 과 같이 URL 자체에 환경 변수를 사용하면 동적인 변화에 대처 할 수 있을 것이다. 물론 앞과 같이 RSL 의 environment 속성을 통해서 URL 에 대한 값을 완벽하게 설정할 수 있다.

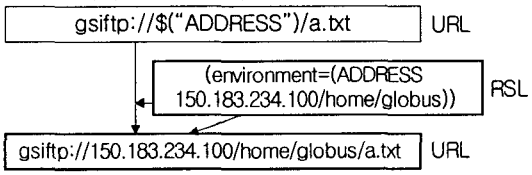

그림 3- 동적인 URL 주소 할당

3.2. 데이터 생성을 위한 설계

읽는(reading) 경우 뿐만 아니라 파일에 쓰는(writing) 경우에도 위의 3.1.과 같이 URI 를 사용해서 상위 노드에 쓸 수 있다.

일반적으로 여러 계산 노드에서 출력 값들을 하나의 파일로 만들어 내고 싶을 때가 많다. 기존에는 파일 이름을 out1.txt, out2.txt... 형태로 하나 하나씩 만든 다음에 out.txt 로 수동으로 붙여서 만든다. 본 연구에서는 이것을 동기화 시켜서 하나의 파일에 순서대로 붙일 수 있는 방법을 제시한다.

GASS 에서 원격지에 쓰기 위해서 *globus_gass_open* 함수를 사용해서 같은 파일을 계속 열게 되면 다음과 같다. 원격지에 데이터를 쓸 때 *globus_gass_open* 함수로 열었던 파일은 로컬의 캐시에 먼저 쓰게 된다. 마지막에 *globus_gass_close* 함수를 호출하게 되면 실제로 원격지에 옮겨가는 구조로 되어 있다. 원격지로 이동중에 다른 노드에서도 같은 파일에 이동 된다면 끝날 때까지 기다린다. 캐시를 사용해서 *globus_gass_close* 함수가 사용될 때 만 멈춰서(block) 기다리면 되므로 상당히 효율적으로 보인다. 하지만 파일이 상당히 크고 계산 노드 수가 많을 때는 파일을 쓰는 동안에 오래 걸린다. 뿐만 아니라 네트워크 속도(latency)때문에 순서대로 데이터가 만들어지지도 않는다. 그리고 글로버스 2.0 버전에서는 GridFTP 에 대해서는 첨가(append) 부분이 구현되어 있지 않아서 GASS 서버만 사용 할 수 있다.

파일 이름을 out.txt 로 열지만 동기화 순서 번호에 따라서 out1.txt, out2.txt, ...로 파일을 원격지에 쓴 다음에 마지막에 자동으로 out.txt 로 합쳐주는 방법을 사용하기로 한다. out1.txt 를 원격지에 쓰는 동안에도, out2.txt 를 동시에 쓸 수 있으므로 큰 파일의 경우에도 병렬적으로 파일을 쓸 수 있는 장점이 있다. 그리고 파일 뒤에 붙은 숫자로써 동기화를 맞출 수가 있다.

4. 구현

본 장에서는 3 장에서 설계한 두 부분에 대해서 C 언어로 API 를 구현 할 것이다. 그리고 포트란의 서브루틴(subroutine)이 C 언어의 API 를 호출하는 것을 제공할 수 있을 것이다.

4.1. URI 기반 GASS 파일 접근 구현

글로버스에서 제공하는 파일에 대한 접근 API 는 *globus_gass_open, globus_gass_close, globus_gass_fopen, globus_gass_fclose* 이다. 뒤의 두개는 파일에 대한 접근을 위한 URL 방식의 API 이다.

본 논문에서는 앞에서 제안한 URI 에 대한 방식을 이용해서 새로운 API 인 *gass_open, gass_close, gass_fopen, gass_fclose* 를 설계한다.

```
gass_open("urn:gsiftp:a.txt", O_RDONLY);
        ⇩
local_hostname = getenv("IPADDRESS");
cur_path = getenv("PATH");
sprintf(address, "gsiftp://%s/%s/a.txt", local_hostname, cur_path);
globus_gass_open(address, O_RDONLY)
```

그림 4 *gass_open* API

gass_open 함수는 그림 4 처럼 동적인 URI 주소를 URL 의 형태로 변환한 뒤에 *globus_gass_open* 함수를 호출하는 형태로 구현 할 수 있고 *gass_close* 함수는 *gobus_gass_close* 함수를 바로 호출 하면 된다. 즉 함수 *gass_open* 과 *gass_fopen* 은 함수 *globus_gass_open* 과 *globus_gass_fopen* 의 주소부분이 URL 에서 URI 로 바뀌는 것이지 다른 인자들은 같다. 함수 *gass_close* 와 *gass_fclose* 또한 인자들이 함수 *globus_gass_close* 와 *globus_gass_fclose* 과 같다.

4.2. 데이터 생성을 위한 설계 구현

같은 파일을 여러 노드에서 동시에 접근해서 파일 핸들을 얻고 싶을 때 다음과 같은 API 를 사용하면 된다.

int *gass_sync_open*(char* filename, int oflag, int sync_order, int max_order_digit, ...);

```
gass_sync_open ("out.txt", 700, 1, 5);
        ⇩
gass_open(".tmp_out.txt00001", 700);
```

그림 5 *gass_sync_open* API

filename 은 *gass_open* 과 같이 URI 에 대한 이름이다. sync_order 를 통해서 동기화에 대한 순서를 맞추게 된다. 그리고 max_order 는 동기화가 이루어지는 자리

수 이다. 그림 5 에서 보듯이 파일 이름 out.txt 하나에 대해서 실제로는 상위 노드에 .tmp_out.txt00001 로 만들어 짐을 알 수 있다. 원격에 대한 파일 핸들에 대해서 열었으면 다음과 같이 닫고서 마지막엔 합쳐주는 API 가 필요하다.

int *gass_sync_close*(int fd, int last_flag);

gass_sync_open 함수를 통해서 열었던 파일 핸들에 대해서 *gass_sync_close* 함수를 통해서 닫아 줘야 한다. 이때 last_flag 를 통해서 마지막 출력 데이터인지 나타내야 한다. last_flag 가 1 일 때 상위 노드에 대해서 파일들을 합치는 작업이 gsi-ssh 를 통해서 이루어지게 된다. 다음과 같이 gsi-ssh 를 이용해서 서로간의 그리드 인증서를 확인 함으로써 상대편 계정의 명령어를 바로 실행할 수 있다.

ssh <상위노드> ls .globus/.tmp_filename* | xargs cat > PATH/filename
ssh <상위노드> rm -f .globus/.tmp_filename*

gass_fopen 과 같이 파일에 대해서만 따로 *gass_sync_fopen, gass_sync_fclose* API 까지 필요하다.

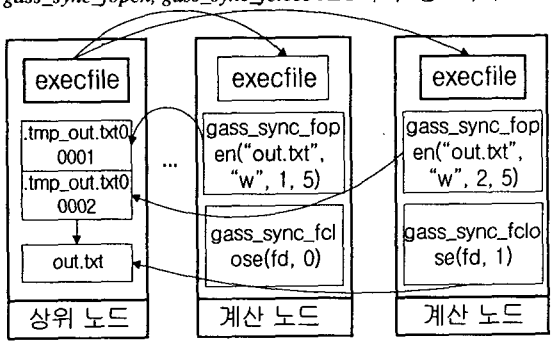

그림 6 각 계산 노드가 상위 노드에 데이터 출력하는 흐름

그림 6 에서 *gass_sync_fopen* 과 *gass_sync_fclose* 함수를 이용해서 이루어지는 작업을 볼 수 있다. 첫 번째 계산 노드가 상위노드에 .tmp_out.txt0001 을 작성하고 두 번째 계산 노드가 마찬가지로 .tmp_out.txt0002 를 작성한 다음에 마지막에 *gass_sync_fclose* 함수를 통해서 out.txt 가 자동으로 만들어 지게 된다.

5. 결론 및 향후 연구

그리드 기반 응용 프로그램은 프로그램과 데이터간의 위치가 다를 경우가 많다. GASS 는 원격 데이터에 대한 접근을 현재 로컬에 있는 것 같이 사용하게 한다. GASS 는 여러 원격지의 파일 시스템을 URL 을 사용해서 통일 시킨다.

URL 은 위치에 따라서 동적으로 값을 바꿔줘야 하므로 여기선 URI 를 사용해서 동적으로 설계를 하였다. URN 의 이름에 urn:gridftp(GridFTP), urn:gass(GASS 서버)를 명시 함으로써 동적으로 URL 로 바꾸거나, URL 주소 자체에 환경 변수를 넣는게 가능하다. 동적으로 컴파일을 새로 하지 않아도 여러 노드에서 사용할 수 있게 해준다. 이것을 가능하게 해주는 API 인 *gass_open, gass_close* 를 본 논문에서 소개 하였다.

여러 계산 노드에서 원격지에 파일을 작성을 하는 경우에 동기화 시켜서 하나의 파일로 만들어주는 방법도 제시하였다. *gass_sync_open* 함수를 통해서 하나의 파일에 순서대로 첨가하는 방식으로 사용자는 이용할 수 있다. 용량이 큰 데이터에 대해서도 병렬적으로 동시에 하나의 파일로 작성이 가능하였다.

참고문헌

[1] I. Foster and C. Kesselman, editors. The Grid: Blueprint for a Future Computing Infrastructure. Morgan Kaufmann Publishers, 1999.

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

[3] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. International Journal of Supercomputer Applications, 11(2):115{128, 1997.

[4] Naming and Addressing: URIs, URLs., http://www.w3.org/Addressing/

[5] URI Syntax, http://www.ietf.org/rfc/rfc2396.txt

[6] URN Syntax, http://www.ietf.org/rfc/rfc2141.txt

[7] M. Litzkow, M. Livny, andM. Mutka. Condor - a hunter of idle workstations. In Proc. 8th Intl Conf. on Distributed Computing Systems, pages 104{111, 1988.

[8] A. Vahdat, P. Eastham, and T. Anderson. WebFS: A global cache coherent file system. Technical report, Department of Computer Science, UC Berkeley, 1996.

[9] A. D. Alexandrov, M. Ibel, K. E. Schauser, and C. J.Scheiman. Extending the operating system at the user level: The UFO global file system. In 1997 Annual Technical Conference on UNIX and Advanced Computing Systems (USENIX'97), January 1997.