WS-DAIR 표준 기반 그리드 메타데이터 카탈로그 인터페이스 구현

안선일, 김남규, 황순욱, 허태상

한국과학기술정보연구원
siahn@kisti.re.kr, ssgyu@kisti.re.kr, hwang@kisti.re.kr, tshuh@kisti.re.kr

Implementation of a Grid Metadata Catalog Interface based on the WS-DAIR standard

Sunil Ahn, Namgyu Kim, Soonwook Hwang, Taesang Huh

Korea Institute of Science and Technology Information

요 약

본 논문은 AMGA 그리드 메타데이터 카탈로그를 위한 WS-DAIR 표준 인터페이스의 설계와 구현 방안을 제시한다. AMGA WS-DAIR 구현은 기존 TCP 인터페이스 기반 AMGA 구현에서 제공하는 모든 기능들을 제공하면서도, WS-DAIR 표준 스펙 기반 인터페이스를 지원함으로써 그리드 상의 데이터베이스 접근 시스템들 간의 상호호환성을 향상시켰다. 또한 성능과 확장성 측면에서도 기존 AMGA 구현과 비슷한 수준을 목표로 구현되었다.

1. 서 론

AMGA(Arda Metadata Grid Application) [1]는 EGEE [2] gLite [3] 미들웨어의 공식 컴포넌트로써, 그리드 상에 저장된 파일들에 대한 메타데이터에 대한 접근을 제공한다. AMGA는 LHC 실험 [4]의 메타데이터 요구사항을 연구하기 위한 실험적 프로젝트로 시작되었으나, 현재는 다양한 커뮤니티들에 의해 메타데이터 카탈로그 서비스로써, 혹은 그리드 기반의 데이터베이스로써 널리 활용되고 있다.

OGF(open grid forum) [5]의 DAIS(data access and integration) 그룹은 그리드 상에서 데이터 접근과 통합을 위한 서비스 타입과 인터페이스 표준화를 위해 구성되었다. DAIS 그룹은 데이터 접근과 통합을 위한 상위 표준으로 WS-DAI(web service data access and integration) [6] 스펙을 정의하였고, 세부적으로 관계형 데이터베이스를 위한 WS-DAIR(web service data access and integration - relational realization) [7] 스펙과, WS-DAIX (web service data access and integration - xml realization) [8] 스펙을 정의하였다.

AMGA는 다양한 사용자 툴들과 통신을 위해 AMGA 전용의 TCP 스트리밍 기반 프로토콜과 메시지 패턴을 사용한다. AMGA는 또한 웹 서비스 기반 인터페이스를 제공하지만, 이는 기존의 어떤 표준 스펙에도 기반을 두지 않은 것이어서, 그리드 상의 다른 데이터베이스 접근 시스템들과 통합되어 활용될 수 없는 상호호환성 문제를 갖는다. 그러므로 AMGA를 위해 WS-DAIR 표준 인터페이스를 구현하는 것은 그리드 상의 데이터베이스 접근 시스템들 간의 상호호환성을 키울 뿐 아니라 여러 다른 연구 기관들이 보유한 데이터의 공유를 촉진시키는 중요한 걸음이 될 것이다.

2. 관련 연구

영국 Edinburgh 대학의 OGSA-DAI 팀은 OGSA-DAI 3.1 프레임워크를 기반으로 OGSA-DAI WS-DAIR 스펙을 구현하였다 [9]. Java와 Axis SOAP 툴을 사용하였으며, rpc/literal 바인딩을 이용하여 구현하였다. OGSA-DAI 프레임워크를 기반으로 하였기 때문에 여러 후위 DB를 지원하는 것이 가능하나 현재는 MySQL DB에 대해서만 시험되었다. 구현이 Java와 Axis 그리고 Apache 웹서비스를 이용하기 때문에 여러 플랫폼에서 실행될 수 있는 반면 처리율이 낮다는 단점이 있다.

3. 설계

그림 1은 AMGA WS-DAIR 구현 아키텍처를 보여준

다. 구현을 단순화하고 프로세스간 불필요한 통신을 피하기 위해 WS-DAIR 스펙의 모든 포트 타입과 인터페이스들을 하나의 서비스에 포함시켰다. AMGA WS-DAIR 구현은 WS-DAIR 스펙의 18개 연산(operation) 중 15개 인터페이스를 구현하였다. AMGA는 데이터베이스의 프로시져(Procedure)를 지원하지 않고, WS-DAIR 스펙의 3개 인터페이스가 프로시져와 관련되어 있어 구현되지 않았다.

AMGA는 WS-DAI 스펙에서 제시한 두 타입의 데이터 자원 접근 방법을 지원한다. 첫 번째는 "direct data access" 방법으로, 사용자의 데이터 질의에 대해 바로 결과를 돌려준다. 두 번째는 "indirect data access" 방법으로, 사용자의 데이터 질의에 대해 새로운 데이터 자원을 생성하고, 생성된 자원에 대한 레퍼런스를 반환한다. 사용자는 이 레퍼런스를 이용하여 신규 생성된 데이터 자원의 일부 또는 전부에 대해 다시 질의할 수 있다. 이 두 타입의 데이터 접근을 지원하기 위해 AMGA WS-DAIR는 두 타입의 리소스를 정의하였다. 첫 번째는 " direct data access "를 지원하는 SQLAccess [7] 타입으로, AMGA WS-DAIR 구현의 메타데이터 서비스를 위해 사용하는 기본 데이터베이스를 말한다. 두 번째는 "indirect data access"를 지원하는 SQLResponse [7] 타입으로, 사용자 질의 (SELECT)에 의해 SQLAccess 타입 자원으로부터 신규 생성되는 자원을 말한다.

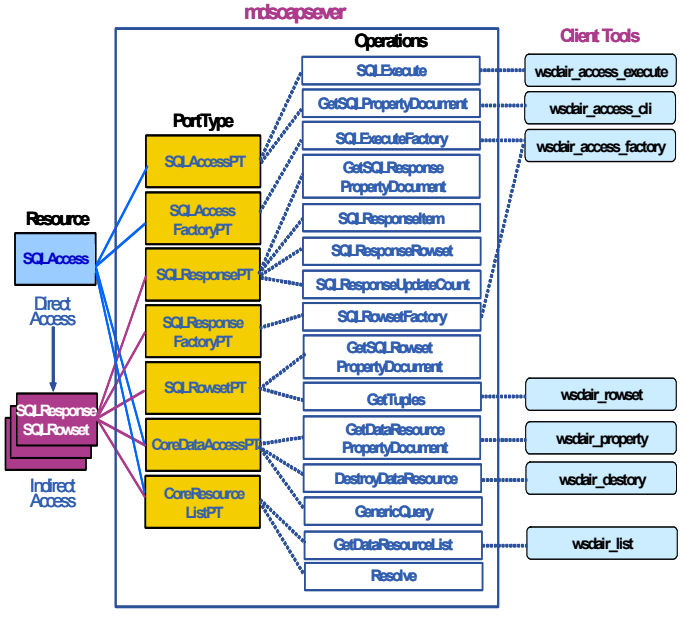

그림 1 AMGAWS-DAIR 인터페이스 구현 아키텍처

WS-DAIR 데이터 리소스 접근을 위해서는 데이터 자원에 대한 추상화된 이름이 필요하며 WS-DAI 스펙은 URI [10] 형식을 따를 것을 명세하였다. URI는 URL과 URN의 두 타입이 있으며, AMGA는 데이터 리소스 이름으로 URL 형식을 채택하여 데이터 리소스 이름만으로 데이터 리소스의 접근에 필요한 모든 정보를 얻을 수 있게 하였다. 데이터 리소스 이름은 아래 형식을 따른다.

https://<hostname>:<port>/<resource type>/<identifier>

WS-DAIR 스펙은 사용자 질의 결과를 표현하는 방법으로 WebRowSet [11] 형식을 최소한 지원할 것을 권장하고 있다. 이에 AMGA WS-DAIR 구현은 결과 데이터셋 (dataset)의 형식으로 JDBC의 WebRowSet 형식을 채택하였다.

WS-DAIR 스펙은 질의를 위한 언어를 강제하지 않는다. AMGA WS-DAIR 구현은 기존 AMGA 응용들의 지원을 위해 AMGA 언어를 지원하고, 다른 그리드 데이터베이스 응용들이 쉽게 AMGA를 활용하여 통합될 수 있도록 기본적인 SQL 언어를 지원한다. 대부분의 데이터베이스 구현들은 SQL을 지원하지만, 지원하는 SQL이 조금씩 다르며 서로 호환되지 않는다. 그러나 대부분의 데이터베이스들이 최근 SQL 표준을 준수하지 않지만 기존 표준인 SQL-92을 준수하는 것으로 알려져 있다. 이에 AMGA WS-DAIR 인터페이스는 SQL-92의 엔트리 레벨 <direct data statements>를 이해하고 실행할 수 있도록 구현하였다. 그리고 사용자 커뮤니티의 요청에 따라 JOIN과 LIMIT, OFFSET에 대한 지원을 추가하였다.

4. 구현

AMGA WS-DAIR 서버는 단독 실행이 가능한 멀티스레드 모델의 데몬 프로세스로 구현되었다. 프로세스가 시작되면 마스터 스레드와 설정된 수만큼의 일꾼 스레드가 시작된다. 마스터 스레드는 사용자로부터 연결 요청을 태스크 큐에 넣는 역할을 하고, 일꾼 스레드는 태스크 큐로부터 연결을 꺼내어 처리하는 역할을 한다. 정해진 수의 일꾼 스레드를 사용하는 것은 사용자 동시 접속이 많아져서 리소스에 과부하가 걸리는 것을 피하기 위해서이다. 각 스레드는 사용자의 질의를 처리하기 위해 데이터베이스 연결을 요구한다. 데이터베이스 연결을 초기화하는 것은 부하가 큰 연산이므로, 초기에 여러 연결을 생성하여 풀로 저장하고, 각 일꾼 스레드가 필요할 때 하나씩 꺼내어 사용한다. 기존 AMGA 구현과 같이 후위 데이터베이스와의 연결은 ODBC (Open DataBase Connectivity)를 이용하였다.

구현에는 SOAP 프로토콜의 구현인 gSoap [12] 라이브러리와 C++ 언어를 이용하였다. gSoap 라이브러리와 툴을 이용하면 웹 서비스 구현을 위해 필요한 스켈레톤(skeleton)과 스터브 (stub)를 생성한다. WS-DAI와 WS-DAIR 스펙에서 주어진 WSDL 파일은 수정 없이 그대로 사용되었다. WS-DAI와 WS-DAIR의 모든 인터페이스들은 하나의 서비스에 포함되어 구현되었고 document/literal 바인딩을 활용하였다. 모든 인터페이스들을 하나의 서비스에 포함하여 구현하는 경우 PropertyDocuement와 관련된 4개 인터페이스들에 대해 클라이언트가 주는 메시지의 형식이 같아서 서버에서는 이를 구분하기 위한 방법이 필요하다. AMGA WS-DAIR 구현에서는 데이터 자원을 접근하는 주소를 활용하여 이 4개 인터페이스를 구분하며 이를 위해 gSoap이 자동 생성한 코드를 약간 수정하여 사용하였다.

AMGA는 데이터베이스의 구조화된 관리를 위해 DB

테이블을 디렉터리에 맵핑하여 유닉스의 파일 시스템과 같은 계층 구조의 뷰(view)를 제공한다. AMGA는 내부적으로 마스터 테이블을 두고 어떤 테이블이 어떤 디렉터리에 맵핑되는지를 관리한다. AMGA WS-DAIR 구현은 새로 생성된 SQLResponse 타입 자원과 URL의 맵핑을 저장하기 위해 이 마스터 테이블을 활용한다.

SQLAccessFactory 인터페이스는 SQLAccess 자원으로부터 새로운 SQLResponse 타입의 자원을 생성하는 인터페이스이다. 사용자가 SQLAccessFactory 인터페이스를 통해 질의하면 그 질의는 후위 데이터베이스에 실제 VIEW를 하나 생성하고, VIEW에 대해 추상화된 데이터 자원 이름이 사용자에게 반환된다. 추상화된 데이터 자원 이름은 마스터 테이블과 연계된 시퀀스를 활용하여 생성되고, 실제 VIEW의 이름과 추상화된 이름의 매핑은 마스터 테이블에 저장된다.

SQLResponse 자원은 하나 이상의 데이터 셋을 포함하는 자원이며, SQLRowsetFactory 인터페이스는 SQLResponse 타입의 자원으로부터 새로운 자원의 생성을 요청하는 인터페이스이다. SQLRowsetFactory 인터페이스를 통해 SQLRowset 타입 자원의 생성이 가능하고, SQLRowset 타입 자원은 SQLResponse의 여러 데이터 셋 중 한 데이터 셋만을 갖는다. SQLResponse 자원에 대한 질의는 데이터 셋 단위이며, SQLRowset 자원에 대한 질의는 데이터 셋에 포함된 튜플(tuple) 단위이다.

AMGA WS-DAIR 구현에서는 SQLRowsetFactory 인터페이스 호출에 대해 신규로 SQLRowset 자원을 생성하는 대신, 기존 SQLResponse 자원에 대한 데이터 리소스 이름을 바로 반환한다. Stored Procedure가 지원되는 경우에만 두 개 이상의 데이터 셋이 존재할 수 있으며, AMGA의 경우 프로시져를 지원하지 않기 때문에 SQLResponse 자원은 항상 하나의 데이터 셋만을 갖는다. 이때문에 SQLRowset 자원이 생성된다면 이 자원은 SQLResponse 자원과 항상 동일하므로 새로 생성하는 의미가 없다. 그러므로 AMGA WS-DAIR 구현에서는 SQLResponse 자원만을 생성하고 이 자원이 SQLRowset 관련 인터페이스들까지도 처리할 수 있도록 하였다.

```
$ wsdair_access_cli "SELECT EMPNUM FROM STAFF LIMIT 1"
STAFF.EMPNUM
E1

$ wsdair_access_factory "SELECT EMPNUM FROM STAFF"
https://localhost:8844/SQLResponse/responses284_5

$ wsdair_rowset -P 0 -C 1 https://localhost:8844/SQLResponse/response
s284_5
/SQLResponse/responses284_5.EMPNUM
E1

$ wsdair_list
https://localhost:8844/SQLResponse/responses284_5
https://localhost:8844/SQLAccess/Metadata

$ wsdair_destroy https://localhost:8844/SQLResponse/responses284_5
Success !!
```

그림 2 AMGA WS-DAIR 클라이언트 툴 사용

AMGA WS-DAIR 클라이언트는 8개 툴로 구성되어 있다. 그림 2의 wsdair_access_execute 툴은 AMGA WS-DAIR를 접속을 위한 쉘과 유사한 툴이다. 이 툴을 통해 AMGA 언어나, SQL 언어로 질의하고 결과를 출력할 수 있다. wsdair_access_cli 툴은 한 명령을 실행하고 그 결과를 출력하는 툴로 스크립팅에 유용하게 활용된다. wsdair_access_factory 툴은 간접 데이터 자원을 새로 생성하며, wsdair_destroy 툴은 간접 데이터 자원을 제거한다. wsdair_list는 모든 데이터 자원들을 열거하는 툴이며, wsdair_property는 데이터 자원의 속성을 출력한다. wsdair_rowset은 간접 데이터 자원으로부터 일부 데이터만을 선택하여 출력하는 툴이다. 마지막으로 AMGA에서는 직접 활용되지는 않지만 다른 WS-DAIR와의 호환성을 위해 SQLResponse 자원으로부터 SQLRowset 자원을 생성하는 wsdair_response_factory 툴도 개발되었다. 그림 2는 이들 클라이언트 툴들의 활용 예를 보여준다. 이들 툴 이외에도 사용자 응용프로그램의 제작에 사용할 수 있는 C++ API를 제공한다.

5. 성능 평가

이 장에서는 AMGA WS-DAIR 구현의 성능을 측정하고 평가하고, TCP 기반 AMGA 구현과 비교하여 성능 차이를 평가하였다. 성능 측정을 위해 사용된 데이터와 질의문은 신약후보물질탐색 프로젝트인 WISDOM [13] 환경에서 실제 사용되는 예를 기반으로 하였다.

WISDOM 환경에서는 수만 개의 그리드 작업들이 실행할 태스크를 얻고, 입력 화합물 메타데이터를 얻고, 그리고 도킹 결과에 대한 메타데이터를 저장하기 위해 AMGA를 이용한다. 또한 사용자는 필요한 화합물 메타데이터를 입력하거나 검색하고, 도킹 결과를 분석하기 위해서 AMGA를 활용한다. 표 1은 성능 측정을 위해 사용된 질의를 보여준다. 처리율의 측정을 위해서는 작업 번호를 입력으로 화합물 번호를 돌려받는 가장 간단한 질의를 활용하여 초당 몇 개의 사용자 질의를 처리할 수 있는지를 측정하였다.

표 1 WS-DAIR 구현 성능 측정에 사용된 질의

	질의
처리율	SELECT ligand_id FROM /drugscreenerg/project/simulation WHERE simulation_id=??

AMGA WS-DAIR 성능 측정을 위해 활용된 기계는 8 GByte 메모리와 네 개 코어를 가진 Xeon 2.1GFlop 이고, 클라이언트와 서버로 LAN으로 연결된 2대의 동일한 사양의 기계를 사용하였다. 서버에는 AMGA와 PostgreSQL을 설치하였고, AMGA와 PostgreSQL의 최대 연결 수는 128로 설정하였다.

처리율의 측정은 클라이언트 기계에서 스레드의 수를 늘리면서 AMGA TCP 프로토콜 기반 구현과 WS-DAIR 구현이 초당 몇 개의 질의를 처리할 수 있는지를 측정하

였다. 각 클라이언트 스레드는 매번 질의를 위해 새로운 연결을 요청하였고, 서버는 별도의 인증을 요구하지 않도록 설정하였다. 그림 3은 클라이언트 스레드의 증가에 따른 TCP 프로토콜 기반 AMGA 구현과 WS-DAIR 구현의 처리율을 비교하여 보여준다. 그림 3을 보면 TCP 구현과 WS-DAIR 구현에서 처리율의 차이가 거의 없으며, 최대 초당 1400개의 질의를 처리할 수 있음을 알 수 있다.

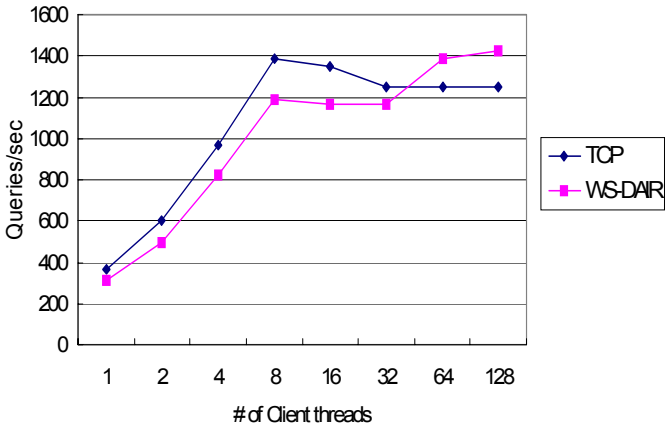

그림 3 클라이언트 스레드의 증가에 따른 처리율

6. 결론과 향후 연구

상호호환성 문제는 여러 그리드 프로젝트들의 핵심 이슈로 다루어지고 있으며, 본 논문은 AMGA를 위한 WS-DAIR 스펙의 구현과, 성능에 대해 설명하였다. AMGA WS-DAIR 구현은 기존 TCP 인터페이스 기반 AMGA 구현에서 제공하는 모든 기능들을 제공하면서도 WS-DAIR 표준 스펙을 지원하여 다른 WS-DAIR 구현들과 상호호환성을 보장한다.

AMGA WS-DAIR 구현은 OGSA-DAI WS-DAIR 구현과 더불어 WS-DAIR 표준 참조구현의 하나로, WS-DAIR 구현들의 상호호환성 시험결과와 경험은 문서로 정리되고 OGF에 의해 표준화 문서로 공식 채택되었다. 향후 이 문서를 바탕으로 WS-DAI 스펙과 WS-DAIR 스펙이 다소 수정될 예정이다.

향후, 표준안 측면에서는 대용량 데이터 처리를 위해 스트리밍 (streaming)을 지원하는 WS-DAIR 스펙에 대한 연구가 진행될 필요가 있다. 그리고 AMGA WS-DAIR 구현 측면에서는 대용량 데이터 처리에 대한 부담을 최소화하는 방향의 연구가 진행될 것이다. 먼저 WebRowSet 양식으로 메시지가 전송되는 경우 실제 전송하려는 데이터에 비해 메시지가 크게 증가하는 경향이 있으므로, 다른 메시지 양식의 지원을 검토할 것이다. 또한 AMGA WS-DAIR 구현을 위한 C++ API 이외의 다양한 API 구현을 검토하고 있다.

참조 문헌

[1] N. Santos and B. Koblitz, "Metadata Services on the Grid," Nuclear Instruments and Methods in Physics Research, vol. 559, no. 1, pp. 53-56, 2006.

[2] F. Gagliardi, B. Jones, F. Grey, M. E. Begin and M. Heikkurinen, "Building an Infrastructure for Scientific Grid Computing: Status and Goals of the EGEE Project," Philosophical Transactions: Math., Physical and Engineering Sciences, vol. 363. pp. 1729-1742, 2005.

[3] gLite, "Lightweight Middleware for Grid Computing," Mar. 2009, [Online]. Available: http://glite.web.cern.ch/glite/. [Accessed: Mar. 30. 2010].

[4] LHC, "Large Hadron Collider," 2007, [Online]. Available: http://lhc.web.cern.ch/lhc/. [Accessed: Mar. 30. 2010].

[5] OGF, "Open Grid Forum," Sep. 2006, [Online]. Available: http://www.ogf.org/. [Accessed: Nov. 30. 2009].

[6] M. Antonioletti, et al., "Web Services Data Access and Integration (WS-DAI) Specification Version 1.0," OGF GFD.74, 2006.

[7] M. Antonioletti, et al., "Web Services Data Access and Integration–The Relational Realisation (WS-DAIR) Specification Version 1.0," OGF GFD.76, 2006.

[8] M. Antonioletti, et al., "Web Services Data Access and Integration–The XML Realisation (WS-DAIX) Specification Version 1.0," OGF GFD.75, 2006.

[9] E. Theocharopoulos and M. Jackson, "OGSA-DAI WS-DAIR 1.0," http://www.ogsadai.org.uk/documentation/ogsadaiwsdair1.0/, [Accessed: Mar. 30. 2010].

[10] URI Planning Interest Group, "URIs, URLs, and URNs: Clarifications and Recommendations 1.0," W3C/IETF, Sep. 2001, [Online]. Available: http://www.w3.org/TR/2001/NOTE-uri-clarification-20010921/. [Accessed: Nov. 30. 2009].

[11] J. Bruce, "WebRowSet XML Schema," 2001, [Online]. Available: http://java.sun.com/xml/ns/jdbc/webrowset.xsd. [Accessed: Nov. 30. 2009].

[12] A. Robert, "A Framework for Service-Oriented Computing with C and C++ Web Service Components," ACM Transactions on Internet Technologies, vol. 8, no. 3, 2008.

[13] N. Jacq, et al., "Grid-enabled Virtual Screening Against Malaria," Journal of Grid Computing, vol. 6, no 1, pp. 29-43, 2007.