동적 서비스 설치 기능 구현을 통한 그리드상의 동적 자원 제공 시스템 개발

변은규[O†] 장재완[†] 김진수[‡]
한국과학기술원 전산학과
[†]{ekbyun[O], jwjang}@camars.kaist.ac.kr, [‡] jinsoo@cs.kaist.ac.kr

Grid-based On-Demand Resource Provisioning System Using Dynamic Service Deployment Mechanism

Eun-kyu Byun[O] Jae-wan Jang, Jin-soo Kim
Computer Science Department, Korea Advanced Institute of Science and Technology

요 약

그리드 컴퓨팅 기술은 이용하여 이종의 분산 자원들을 묶어서 사용할 수 있게 해 준다. 최근의 그리드는 이기종 자원간의 상호 호환성 증대를 위하여 웹 서비스 기술을 도입하여, OGSA와 WSFR 표준을 정의하였다. 그에 따라 그리드 자원과 그 위에서 동작하는 작업들은 웹 서비스의 형태로 구현되고 관리된다. Globus Toolkit은 이 표준들을 구현한 미들웨어로 그리드 연구에서 가장 널리 활용되고 있다. 서비스가 자원을 효율적으로 사용하면서 서비스 성능을 유지하기 위해서는 서비스가 원하는 시점에 원하는 양의 자원을 제공 받을 수 있어야 한다. 이를 위해서 유휴자원을 찾는 기능과 동적으로 자원에 서비스를 설치하는 기능이 반드시 필요하다. 그러나 기존의 Globus Toolkit은 이런 기능을 제공하지 않는다. 따라서, 본 연구에서는 Globus Toolkit에 동적 서비스 설치 기능을 추가하고 그것을 관리할 수 있는 도어 서비스를 구현하여, 동적 자원 제공 기능이 가능하도록 하였다.

1. 서 론

그리드 컴퓨팅[1] 기술은 지리적으로 분산된 이질적인 자원들을 하나로 묶어 복잡하고 어려운 문제를 해결할 수 있는 컴퓨팅 자원을 제공하는 것을 목적으로 한다. 관리 도메인이 다른 다수의 분산된 자원을 효율적으로 검색하고 공유하기 위해 VO(Virtual Organization)의 개념이 도입되었고 자원을 가진 사용자들이 쉽게 VO를 구성하고 사용할 수 있도록 미들웨어들이 개발 되었다. 그 중 공개 미들웨어인 Globus Toolkit[2]이 가장 널리 사용되고 있다.

최근 GGF[3]를 중심으로 그리드 컴퓨팅에 웹 서비스[4] 기술을 도입하려는 시도가 있었다. 웹 서비스는 플랫폼 독립적인 전송 방법이므로 이종의 자원을 다루는 그리드에 적합한 기술이기 때문이다. 그 첫번째 결과물로 OGSA (Open Grid Service Architecture)[5]가 제안되었다. Globus Toolkit version 3(GT3)는 사용자가 OGSA의 표준에 맞게 그리드를 구성할 수 있는 기능을 제공한다. 또한 그 이후 웹 서비스의 표준이 발전함에 따라 그에 맞는 새로운 그리드 표준인 WSRF(Web Service Resource Framework)[6]가 정의되었다. WSRF에 의거하여 구현된 Globus Toolkit 버전4(GT4)는 WSRF기반의 그리드 환경을 구성할 수 있게 해 준다. 두 기술 모두 그리드 자원을 사용하고, 자원의 상태를 모니터링 하는데 웹 서비스의 기술을 사용한다. 각 자원에 그리드 서비스(웹 서비스)를 실행시킬 수 있는 컨테이너를 설치하고 웹 서비스 프로토콜을 사용하여 상호 통신한다.

그리드 상의 자원을 활용하여 사용자 들에게 서비스를 제공할 때, 정해진 양의 자원만을 할당 받아 서비스를 제공해야 한다면, 자원의 효율과 서비스의 성능이 많이 저하될 수 있다. 예를 들어, 특정 시간대에만 자원을 많이 사용하는 서비스의 경우, 자원을 많이 필요로 하는 특정 시간대의 성능을 위하여 그 서비스에 자원을 많이 할당한다면 대부분의 시간 동안 자원이 활용되지 않게 된다. 또한 자원의 효율을 위하여 그보다 적은 양의 자원만을 할당한다면, 서비스에 대한 요청이 많아지는 시간대에는 성능이 급격히 떨어지는 문제가 발생하게 된다. 따라서, 그리드 상의 자원을 효율적으로 사용하면서 서비스의 성능을 보장하기 위해 동적 자원 제공(On-Demand Resource Provisioning) 기법이 필요하다. 즉, 각 서비스의 요청 량에 따라 많은 자원이 필요한 시점에는 새로운 자원을 동적으로 할당 받아 사용하고 다시 요청 량이 감소하면 자원을 반환하여 다른 서비스가 사용할 수 있도록 하는 것이다. 웹 서비스 기반인 Globus Toolkit 3, 4 에서 이 동적 자원 제공이 가능하기 위해서는 2가지 기능이 제공되어야 한다.

1. 적합한 유휴 자원을 검색하는 일
2. 선택된 자원에 서비스를 동적으로 설치하여 실행 시킬 수 있는 환경을 만드는 일

기존의 Globus Toolkit에서는 서비스를 동적으로 설치하는 것이 불가능 하다. 자원에서 새로운 서비스를 실행시키기 위해서는 컨테이너를 정지시키고 설치를 한 후 재시작 해야만 한다. 따라서, 본 연구에서는, Globus Toolkit 기반의 그리드 환경에서 동적 자원 제공기능이 가능하도록, 위의 두 가지 기능을 제공하는 서비스들을 설계 및 구현하였다. 먼저 도어 서비스(Door Service)를 구현하여, 특정 서비스의 요청 량의 변화에 따라 자원을 선택하여 할당 받거나 반환하는 일을 처리 할 수 있게 하였다. 그리고, 동적인 서비스 설치를 위하여 GT3용으로 UFS(Universal Factory Service)라는 새로운 서비스를 개발하였고, GT4 용으로 동적 서비스 설치가 가능한 새로운 컨테이너를 개발하여, 자원을 일시 정지 시키지 않고도 새로운 서비스를 설치, 실행 시킬수 있는 기능을 제공하였다.

2. 관련 연구

OGSA에서는 모든 그리드 자원과 자원을 사용하는 작업들을 그리드 서비스라는 형태로 제공하도록 정의하고 있다. 그리드

서비스는 웹 서비스를 그리드 시스템에 맞게 확장한 것으로, 기존의 웹 서비스가 자원 및 서비스의 상태 정보를 장기간 유지하는 기능이 없는 단점을 보완하기 위해 새로이 제안된 기술이다. GT3 를 이용하여 사용자들은 OGSA규정에 맞게 그리드를 구성할 수 있다.

웹 서비스 기술이 발전함에 따라, 새로운 웹 서비스 표준들이 정의 되었다. 그리드 컴퓨팅에서도 이 새로운 웹 서비스 표준을 기반으로 WSRF(Web Service Resource Framework)[6]를 정의하였다. 이것은 OGSA와 달리 웹 서비스의 표준과 완벽히 호환이 가능하다. WSRF는 WS-Resource 라는 개념을 도입하여, 웹 서비스와 상태정보(State)를 함께 묶어 처리할 수 있는 방법을 정의하고 있다. 즉, 사용자가 사용하는 서비스 인스턴스는 서비스 코드와 그 코드가 사용하는 자원(Resource)의 결합으로 결정된다. 서로 다른 사용자가 같은 서비스를 사용할 때, 서로 다른 자원을 사용함으로써 각자 자신만의 작업을 수행할 수 있다. 또한 자원은 휘발성이 아니므로, 사용자가 자신의 작업정보를 유지할 수 있게 된다.

동적으로 자원에 새로운 서비스를 설치하는 기술에 대한 연구는 많이 진행되어 왔다. Oceano[7]에서는 다수의 자원으로 구성된 서버팜을 여러 작업이 동시에 사용할 때, 각 작업의 사용량에 따라 각 작업에 할당된 자원의 양을 동적으로 조절하는 기술을 제안하였다. SODA[8]에서는 가상기계(Virtual Machine) 기술을 이용하여, 각각의 자원에 가상 기계를 설치하고 그 위에 새로운 서비스를 설치하는 방법을 제안하였다. OGSI.NET[9]은 마이크로소프트닷넷 기반의 OGSA 구현판으로, 본 논문에서의 UFS와 비슷한 기능을 제공하지만, .NET 프레임워크에서 작동 가능한 모듈만이 설치가 가능하다는 한계를 가지고 있다.

3. 시스템 구조

3.1 전체 시스템 구조

그림 1은 도어 서비스를 통해 서비스가 그리드 자원을 동적으로 활용하여 사용자에게 서비스를 제공하는 모습을 나타낸다. 그림 1은 GT3의 예를 나타낸 것으로 모든 자원의 컨테이너에는 UFS가 설치되어 있다. 각각의 서비스는 자신의 도어 서비스를 가지고 있다. 사용자는 도어서비스에 접속하여 서비스 요청을 보내고 도어 서비스는 그 요청을 적합한 자원에게 분배하는 역할을 한다. 또한 도어 서비스는 사용자의 요청 량이 증가하여 적합한 자원이 없을 때 자원을 검색하여 새로운 유휴 자원을 찾은 후 그 자원에 설치된 UFS를 통하여 자신의 서비스를 설치하게 된다. 그렇게 함으로서 더 많은 자원을 확보하여 증가된 요청을 처리할 수 있다.

GT4의 개선된 컨테이너도 구현의 내부적인 사항이 다를 뿐 기능적으로는 GT3의 UFS와 동일하다. 도어 서비스가 컨테이너에 새로운 서비스의 설치를 요청한 후 서비스 코드를 전송하면 컨테이너의 중지 없이 서비스를 설치하여 바로 실행이 가능하다.

3.2 *Universal Factory Service*

GT3에서 서비스를 실행 시키는 과정은 다음과 같다. 먼저, 서비스 팩토리를 서비스 컨테이너에 설치 해야 한다. 그 후 사용자는 서비스 팩토리에 서비스 인스턴스의 생성을 요청하게 된다. 그러면 팩토리는 컨테이너 내부에 새로운 서비스 인스턴스를 생성하고 그 주소를 사용자에게 넘겨준다. 사용자는 이후 그 주소를 가지고 서비스를 이용한다. 그러므로, 자원에 서비스를 설치하기 위해서는 컨테이너에 해당 서비스의 팩토리를 설치하면 된다.

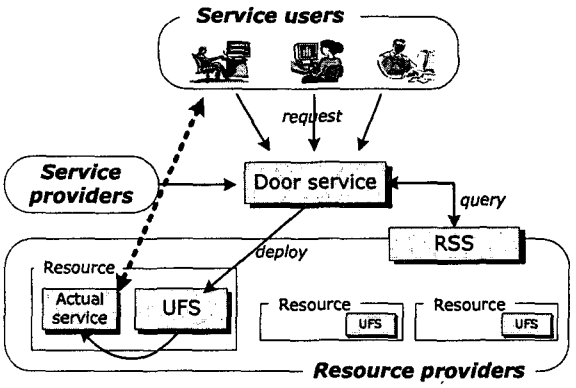

그림 1. GT3에서 도어 서비스와 UFS의 연동으로 서비스가 동적 자원 제공 기능을 사용하는 관계도

도어 서비스는 UFS의 createUFSInstance() 메소드를 통하여 새로운 서비스의 설치를 요청하면 UFS는 새로운 UFS instance를 생성한다. 이 생성된 UFS instance 자체가 새로운 팩토리의 역할을 하게 된다. 도어 서비스는 설치할 서비스의 코드를 생성된 UFS instance 에게 전송하고 서비스 클래스와 UFS instance를 연결 시킴으로써 UFS instance가 새로운 서비스의 팩토리로서 정상적으로 작동하게 한다.

3.3 향상된 GT4 컨테이너

GT4에서 서비스의 설치는 컨테이너가 정지된 상태에서 서비스 코드, server-config.wsdd, jndi-config.xml 파일들을 지정된 위치에 복사를 하는 과정이다. 그리고 난 후 컨테이너가 시작될 때, 각 서비스들의 server-config.wsdd, jndi-config.xml 파일들을 읽어서 AxisServer의 환경(EngineConfiguration)과 JNDI 네이밍 서비스에 등록을 함으로써 원거리 사용자가 서비스에 정상적으로 접근이 가능해 진다. 즉 서비스가 설치가 되기 위해서는, 1)서비스 코드의 전송, 2)웹 서비스 엔진인 AxisServer의 EngineConfiguration에 추가, 3)서비스와 리소스등의 관계를 기술하는 JNDI 네이밍 서비스의 갱신 의 3단계가 완료 되어야 서비스를 정상적으로 실행 할 수 있다. 따라서, 본 연구에서 GT4-Java 컨테이너를 수정하여, 컨테이너 가 동작중인 상태에서 위의 세 단계의 과정을 수행하는 모듈을 호출할 수 있도록 만들었다.

컨테이너는 일종의 웹 서버로서, HTTP POST 명령으로 SOAP 메시지를 받아 그것을 파싱하여 AxisEngine을 통해 서비스를 실행하고 그 결과를 HTTP 프로토콜을 통해 전달하는 역할을 수행한다. 수정된 GT4 컨테이너에서는 HTTP POST의 내용을 확인하여, 도어 서비스가 새로운 서비스의 설치를 요청하는 내용을 보냈을 경우에는, 서비스 설치 모듈을 호출하여, 위의 세 단계의 설치 과정을 실행하게 된다.

4. 실 험

도어 서비스와 동적인 서비스 설치 기법을 활용함으로써 서비스의 성능과 자원의 활용 효율이 얼마나 증가하는지를 알아 보기 위해 실험을 수행하였다. 실험을 위하여 8대의 펜티엄 Xeon 2.8Ghz PC와 4대의 펜티엄3 850Mhz SMP PC로 이루어진 그리드를 구성하였다. 각각의 자원에 UFS를 포함한GT3를 설치하고 3종류의 다른 서비스를 제공하도록 하였다. 각각의 서비스의

도어 서비스들을 한 대의 Xeon PC에 설치하고, 또 다른 한 대의 Xeon PC에는 클라이언트 프로그램을 설치하여 연속적으로 서비스 요청을 생성하도록 하였다. 아래 표 1은 각 서비스의 시간의 변화에 따른 요청 비율의 변화를 나타낸다.

그림 3은 시간에 따라 각 서비스가 점유한 자원의 양을 나타낸다. 세 서비스 각각이 요청 비율이 변함에 따라 필요한 양의 자원만을 사용함으로써, 자원이 각각의 서비스에게 골고루 분포되는 모습을 보여준다. 그림4는 서비스의 실패 비율을 동적 자원 제공 기법을 사용했을 때와 그렇지 않았을 때를 비교한 그래프이다. 동적 자원 제공을 사용함으로써, 서비스의 성능이 월등히 좋아졌음을 확인할 수 있다.

서비스	Phase 1	Phase 2	Phase 3	Phase 4
Raja	1.0	2.0	1.0	0.0
JIU	1.0	0.5	1.5	1.5
Jspeex	1.0	0.5	0.5	1.5

표 1. 시간에 따른 사용자의 서비스 요청량의 변화

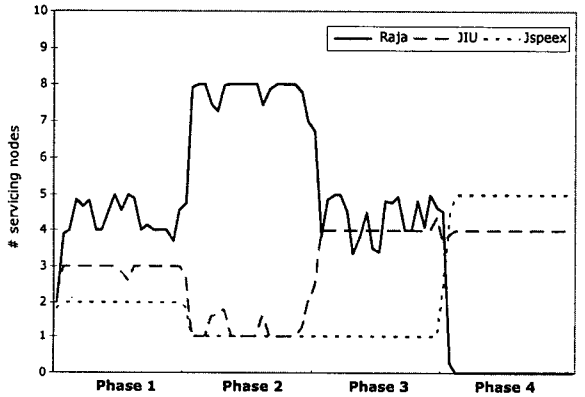

그림 2. 서비스 요청 비율의 변화에 따른 각 서비스의 자원 점유량의 변화 그래프

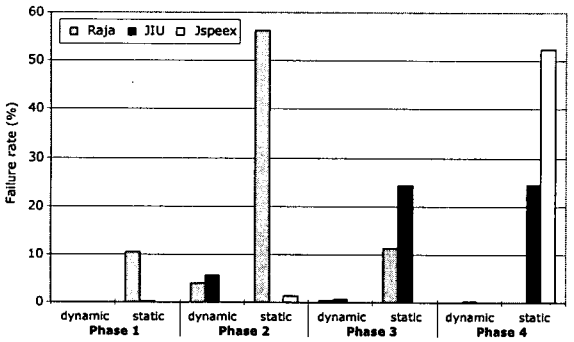

그림 3. 동적 자원 제공 기법을 사용했을때와 그렇지 않을때의 서비스 실패율의 비교

5. 결론 및 향후 연구과제

본 연구에서는 웹서비스 기반의 Globus Toolkit으로 구성된 그리드 환경에서 서비스에 자원을 동적으로 제공하는 시스템 구조를 설계 및 구현하였다. Globus Toolkit3, 4 각각에 UFS와 새로운 컨테이너를 구현함으로써 자원의 정지 없이 새로운 서비스를 설치하는 기능을 구현하고 그것을 이용하는 Door Service를 구현함으로써, 서비스가 사용하는 자원의 양을 동적으로 조절할 수 있게 하였다. 그리고 GT3 기반의 실험을 통하여, 우리의 동적 자원 제공 기능을 이용하여, 일정 수준 이상의 서비스 성능을 유지하면서도 그리드 자원을 효과적으로 활용할 수 있음을 확인하였다

현재 구현된 기능으로는 이미 실행되고 있는 작업의 인스턴스를 다른 자원으로의 이전하는 것은 불가능하다는 한계를 가지고 있다. 하지만 WSRF기반의 서비스에서 상태 정보를 가지고 있는 Resource의 이전을 가능하게 하는 기능을 개발하면, 작업의 인스턴스까지도 다른 유휴자원으로 이전하여 작업의 성능을 향상시킬 수 있을 것이다.

6. 참 고 문 헌

[1] I. Foster, C. Kesselman, and S. Tuecke, " The Anatomy of the Grid: Enabling Scalable Virtual Organizations," International Journal of Supercomputer Applications, 2001.
[2] The Globus Alliance, http://www.globus.org/
[3] Global Grid Forum, http://www.gridforum.org/
[4] F. Curbera, M. Duftler, R. Khalaf,W. Nagy, N Mukhi, and S. Weerawarana, " Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI," IEEE Internet Computing, 2002.
[5] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, " The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration," Open Grid Service Infrastructure WG, Global Grid Forum, 2002.
[6] WSRF, http://www.globus.org/wsrf/
[7] K. Appleby, S. Fakhouri, L. Fong, G. Goldszmidt, M. Kalantar, S. Krishnakumar, D. P. Pazel, J.Pershing, and B. Rochwerger, " Oc'eano- SLA Based Management of a Computing Utility," Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management, 2001.
[8] X. Jiang, D. Xu, " SODA: a Service-On-Demand Architecture for Application Service Hosting Utility Platforms," Proceedings of the IEEE International Symposium on High Performance Distributed Computing, 2003.
[9] G. Wasson, N. Beekwilder, M. Morgan, and M. Humphrey, " OGSI.NET: OGSI-compliance on the .NET Framework, " Proceedings of the IEEE International Symposium on Cluster Computing and the Grid, 2004.