Grid 컴퓨팅 환경에서 JClarens를 이용한 Discovery Service 연구

심의규, 이무훈, 박희용, 최의인
한남대학교 컴퓨터공학과
e-mail : {ekshim, mhlee, hypark, eichoi}@dblab.hannam.ac.kr

A Study on Discovery Service using JClarens in Grid Computing Environment

Eui-Kyu Shim, Moo-Hun Lee, Hee-Yong Park, Eui-In Choi
Dept of Computer Engineering, Hannam University

요 약

Grid 컴퓨팅은 과학 분야에서 처음 시작되어 현재 과학 분야뿐만 아니라 e-business와 같은 더 많은 분야에서 폭 넓은 활용을 위하여 SOA(Services Oriented Architecture)를 채택하게 되었으며, SOA 프레임워크에 있는 기존의 어플리케이션과 프로그래밍 라이브러리를 통합하게 되었다. 특히 SOA에 기반을 둔 Clarens Grid 서비스 프레임워크는 권한 부여(authorization), 접근 제어(access control), Discovery Service를 제공한다. 본 논문에서는 Java로 구현한 JClarens를 이용하여 Discovery Service를 구현하는 두 가지 방법을 기술하고, 그 구현 방법으로 인해 발생할 수 있는 서비스 재등록으로 인한 네트워크 트래픽 증가에 대한 문제점을 진단하고 재등록을 위한 시간 간격을 차등화 함으로서 이 문제점을 해결할 수 있는 방안을 제안하였다. 따라서 Discovery Service는 Grid 컴퓨팅 환경에서 사용자에게 가장 적절한 자원을 발견할 수 있도록 지원 할 것이다.

1. 서론

Grid 컴퓨팅에 대한 연구는 1985년 I-WAY 실험을 기반으로 과학 분야에서 가장 먼저 시작되었다. I-WAY 프로젝트는 미국의 대형 슈퍼컴퓨터센터의 자원을 결합하는 수단으로 17개의 사이트를 통하여 고성능 컴퓨터와 첨단 가시화 환경을 구축하는 것이다. 이 실험에서는 고속 네트워크가 북아메리카를 가로질러 단시간에 17개 사이트의 첨단 자원을 연결하는데 사용되었다. 이 실험의 성공으로 많은 Grid 관련 연구가 시작되었으며, 이러한 네트워크가 필요하게 된 것은 과학연구 분야에서 테라바이트(terabytes) 나 페타바이트(petabytes), 엑사바이트(exabytes)에 이르는 대용량의 데이터 처리가 필요하였기 때문이다.

Grid라는 단어의 어원은 전력 Grid에서 유래되었으며, 전력은 콘센트에 플러그를 꽂는 것만으로도 어디에서 발전되고 있는가를 전혀 의식하지 할 필요없이 전기를 사용할 수 있다. 이와 같이 Grid의 어원도 지리적으로 분산된 컴퓨터를 결합시켜 그 처리능력을 전기처럼 발생 장소를 의식하지 않고 원하는 만큼 사용할 수 있게 한다는 생각에서 나온 것이다. 그리고 WWW(World Wide Web)은 네트워크를 통해 문장과 화상 등 컨텐츠(contents)를 공유할 수 있게 하였다. 하지만 이제 Grid의 개념에서는 이러한 컨텐츠의 공유뿐만 아니라 컴퓨터의 자원, 전문지식이나 인력 자원까지 포함하는 모든 자원을 세계가 공유하고자 하는 비전을 바탕으로 발전하고 있다.

이러한 Grid 컴퓨팅 환경은 다음과 같은 특징을 가지고 있다.

- 지역적으로 분산된 자원들을 이용
- 자원들은 운영체제와 같은 컴퓨팅 환경이 서로 다름
- Grid 컴퓨팅 환경을 이루는 각 사이트들이 보유하고 있는 자원들이 다름
- 자원들이 동적으로 추가되고 제거됨
- Grid 환경에서 사용자들의 요구사항이 서로 다름

Grid는 위와 같은 특징을 기반으로 다양한 서비스를 제공하기 위하여 많은 연구들이 진행 중이다. 그 중에서 서로 다른 요구사항은 Grid 환경이 매우 이질적인 성격을 가지고 있기 때문에 발생하게 된다. SOA는 이러한 이질적인 시스템 간의 자원 공유에 대한 문제를 효율적으로 해결할 수 있다. 그것은 바로 SOA는 이질적인 시스템 속에서 지역적인 제어가 가능하기 때문이다. 그리고 다음과 같은 SOA의 세 가지 특징으로 인해 Grid 컴퓨팅 환경에서 발생하는 문제점을 해결할 수 있다.

- 표준 인터페이스 정의 – 자원들을 위한 일반적인 인터페이스를 정의하여 사용 가능
- 구현의 독립 – 특정 운영체제나 프로그래밍 언어에 제약을 받지 않고 서비스를 구현할 수 있고, 그러한 서비스를 사용할 때에도 특정 운영체제나 프로그래밍언어에 제약을 받지 않음
- 표준기반 서비스 – 서비스들이 원격에 있는 서비스에 접근하거나 그 서비스를 선택하기 위한 기초적인 보안(security), 발견(discovery), 그리고 접근제어 기능을 제공

Clarens Grid Services Framework[1,2,3]를 이용하는 GAE(Grid Analysis Environment)[4]는 SOA의 대표적인 예이다. GAE는 데이터 분석을 위해 지역적으로 분산된 컴퓨팅 자원들을 이용하는 수많은 사용자들에게 일관된 컴퓨팅 환경을 제공한다. GAE SOA는 Caltech(California Institute of Technology)에서 진행 중인 Ultralight 프로젝트의 일환으로 단대단(end-to-end)에서의 네트워크 통합, 전역적인 자원 관리, 데이터 중심의 Grid 시스템에 초점을 두고 있다. 또한 Clarens는 Lambda station, PEAC(Proof Enabled Analysis Center)[5], Physh와 같은 프로젝트에서 Grid 서비스 프레임워크(Grid Service Framework)로 이용하고 있다.

본 논문은 Clarens에서 서비스를 등록하기 위해 발생하는 네트워크 트래픽을 감소시키기 위한 방안에 대해 설명하기 위해 Java로 구현한 Clarens인 JClarens를 활용하여 Discovery Service에서 발생할 수 있는 네트워크 트래픽 증가에 대한 문제를 해결하기 위한 방안을 제시한다.

본 논문의 구성은 다음과 같다. 2장에서 관련 연구에 대해 기술하고, 3장에서 Discovery Service의 구현에 대해 기술하며, 4장에서는 이러한 구현에 따른 문제점을 및 개선 방안에 대해 기술한다. 마지막으로 5장에서 결론 및 향후 연구 과제에 대해 기술한다.

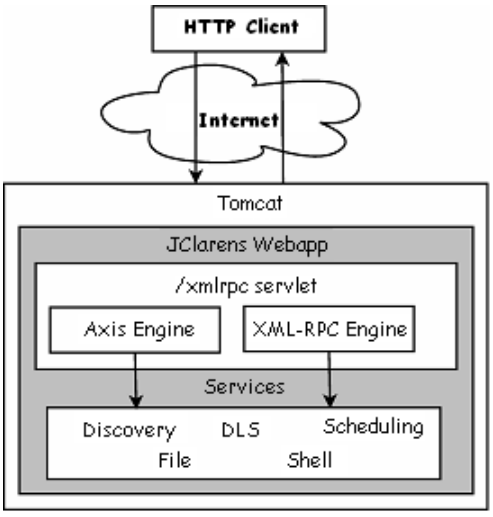
2. 관련 연구

2.1 Clarens Architecture

Clarens 프레임워크의 핵심은 권한 부여, 접근제어, Discovery Service, 그리고 추가된 Grid 서비스의 호스팅을 위한 표준 기반 서비스들의 집합을 제공한다. 여기에 추가된 서비스들은 작업 요청(job submission), 작업 스케줄링(job scheduling), 데이터 발견 및 접근과 같은 기능들을 제공한다. Clarens 프레임워크 내에서 사용되고 있는 보안을 위한 PKI(Public Key Infrastructure)나 원격 서비스를 가져오기 위한 SOAP/XML-RPC와 같은 표준 기술들은 분산된 환경에서 안전하고 신뢰할 수 있는 접근을 제공한다. Java와 Python을 이용하여 구현한 Clarens는 일반적인 표준 기반 서비스를 공유하며, PClarens는 mod_python 모듈의 아파치 웹 서버를 기반으로 하고 있다. PClarens의 서비스들은 Python을 사용하여 만들어 진다. JClarens는 톰캣 서블릿 컨테이너(Tomcat Servlet Container)에서 운영되는 단일 J2EE 웹 어플리케이션으로 개발된다.

2.2 JClarens Architecture

JClarens는 Java를 기반으로 Caltech에 의해 개발되어진 웹 서비스 프레임워크이다. 그리고 APIs를 결속시키기 위해 두 개의 구현 사이의 상호운용성을 보장한다. 또한 JClarens는 Java의 웹 기반 특성을 부각시키면서 통합하는 것을 유용하게 한다. 이미 UFL(University of Florida)의 SPHINX[6,7] Job Scheduling Middleware의 개발자들은 웹 서비스들을 호스팅하기 위해 JClarens를 이용한다. 게다가 JClarens는 Clarens의 프레임워크와 더불어 분산된 모니터링 시스템인 MonALISA[8]의 통합을 가능하게 했다. 그 결과로 Clarens를 위한 Discovery 서비스와 Monitoring 서비스를 지원하고 있다.



(그림 1) JClarens Architecture

그림 1과 같이 JClarens 아키텍처는 Java 서블릿 기술과 XML-RPC/SOAP에 기반을 두고 있다. 그리고 서비스는 Python의 모듈로써 클래스의 형태로 제공되어지며, 인터페이스 집합을 구현한다. JClarens 는 서블릿 엔진을 탑재한 어떤 웹 서버에도 배포되어질 수 있지만, 현재는 아파치 톰캣에 구

축하여 테스트가 진행 중이다. 클라이언트 간의 통신은 XML-RPC를 이용하고 있으며, 기존 Clarens의 클라이언트 간 호환성도 유지하고 있다. 즉, 이것은 다른 프로그래밍 언어로 구현된 Clarens의 클라이언트와 JClarens의 클라이언트 사이에 상호운용성을 보장하고 있다. 그리고 JClarens와 기존 Clarens는 peer-to-peer 서버의 집합으로써 배포를 고려하고 있다.

3. Discovery Service의 구현 방안

3.1 Core Service

동적인 Grid 환경에서 모든 권한부여와 접근제어 쿼리들을 위한 신뢰할 수 있는 권한부여자(central authority)가 없다. 따라서 Grid 서비스들의 각 지역 제공자는 그들의 지역 서비스들의 집합을 위해 이러한 특징을 제공해야한다. Clarens 프레임워크는 권한 부여과 접근제어 관리 기능들을 제공하는 시스템 서비스를 포함해 몇 개의 코어 서비스(core services)를 제공한다[9]. 그리고 그룹 서비스는 사용자들의 집단이 단일 개체로서 관리될 수 있도록 하여 접근 제어 기능들을 증대했다. 파일 서비스는 원격 시스템의 파일을 보거나 업로드하거나 다운로드할 수 있게 하기 위해 제한되는 메커니즘을 제공한다. 그리고 프록시 서비스는 쉘 명령들을 실행할 때와 다른 하나의 서비스로부터 호출된 서비스를 만들 때 클라이언트 프록시 크리덴셜(proxy credentials)을 관리한다.

현재 코어 서비스는 계속적으로 추가됨에 따라 모든 가능한 서비스를 포함하고 있는 하나의 큰 패키지를 만들기 보다는 사이트 관리자들이 서비스를 추가적으로 인스톨할 수 있도록 하는 형태로 운영되어 진다.

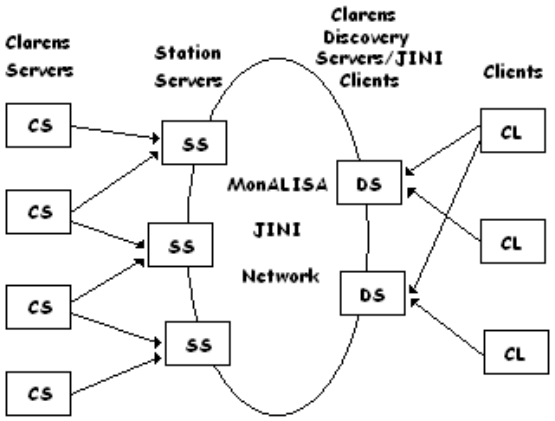
3.2 Discovery Service

분산된 서비스 환경에서 서비스들은 동적으로 추가 및 삭제되어짐으로 예측이 불가능하게 된다. 사용자들과 어플리케이션들은 이렇게 변화하는 정보를 지속적으로 획득한다는 것은 실질적으로 불가능하다. Discovery Service는 사용자들과 어플리케이션들에게 service를 위한 질의를 구성하여 동적인 환경에서 장소에 관한 최신 정보와 서비스의 인터페이스를 갱신시키는 기능을 제공한다. Discovery Service에서 등록은 서비스가 아직 사용가능하다는 것을 나타내기 위하여 규칙적인 간격으로 발생된다. 만일 서비스가 어떤 시간 내에서 Discovery Service에게 통지를 하지 못하게 되면 그것은 자동적으로 레지스트리에서 삭제된다. Discovery Service는 다음의 4가지 방법을 제공해야 한다.

① register는 새로운 서비스들을 레지스트리에 추가하는데 사용된다.
② find_server는 어떤 검색 요청에 부합되는 서비스 호스트들의 위치 결정을 하기 위해 사용된다.
③ find는 어떤 검색 요청에 부합하는 서비스 인스턴스들의 위치를 결정하기 위해 사용된다.
④ deregister는 레지스트리에서 서비스를 제거하기 위해 사용된다.

그러나 서비스가 재등록하는데 실패하면 레지스트리에서 자동적으로 서비스는 제거된다. 그러면 재등록이 실패되었다고 해도 일정시간 후에 다시 그 서비스는 재등록을 하기 때문에 이로 인한 문제는 발생하지 않는다.

그림 2에서는 MonALISA Grid Monitoring System과 JINI를 기본으로 한 peer-to-peer 구현이다. MonALISA는 로컬 모니터링 데이터를 수집하는 Station Server를 사용하고, 다른 Station Server들과 또는 이 정보에 관심을 가지고 있는 다른 Client들과 일부 모니터링 데이터를 공유하는 JINI 기반의 모니터링 시스템이다.



(그림 2) Discovery Service

Discovery Service를 구현하는 방법은 두 가지가있다. 우선 첫 번째 방법은 Discovery Service가 MonALISA JINI 네트워크에 service registration들을 등록하기 위해 ApMon 라이브러리를 사용한다. 각 Discovery Service는 이 메모리 안의 캐쉬를 정기적으로 소멸시키고, service들은 다시 정기적으로 등록을 시도하는 방식으로 작동된다. 그리고 service 레지스트리가 메모리에 저장된 다음에는 그것이 server 전체에 걸쳐 지속적으로 server 재시동을 하지 않고, 그것이 다시 시작될 때 새로운 정보를 빠르게 레지스트리에 등록하기 때문에 service가 등록한 정보의 재사용에 대한 문제를 방지할 수 있다.

그리고 두 번째 방법은 Discovery Service에 MonALISA JINI 네트워크를 사용하지 않는 것이다. 그 대신에 서비스 등록을 저장하는 UDDI repository를 사용하는 것이다. UDDI는 간단히 JClarens service 레지스트리와 더 많은 기능을 가진 일반적인 UDDI 레지스트리 사이에서 연결을 해주는 중요한 역할을 수행한다. 이러한 두 번째 구현 방법은 register 방법에 있어서 UDDI 레지스트리로 Web Service를 추가하기 위해 해당하는 UDDI Service 호출을 생성하는 것이다. find와 find_server 방법은 UDDI 레지스트리에 질의를 수행하고 Discovery Service interface에 부합하는 것을 찾아내기 위해 결과를 재포맷한다. 이 UDDI 구현은 이미 사용하고 있는 기존의 UDDI repository들이 가지는 환경을 그대로 사용할 수 있다는 장점을 가진다.

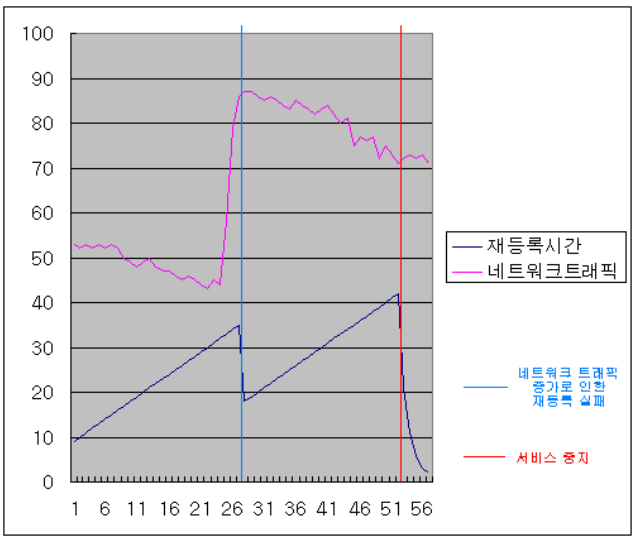
4. 문제점 및 개선 방안

3장에서 제시하고 있는 두 가지 구현의 문제점은 서비스가 증가하는 경우 서비스를 등록하기 위한 네트워크 트래픽이 증가하게 되는 문제점을 가지고 있다. 위 와 같은 방식의 구현에서 문제가 되는 부분은 일정시간이 지나면 서비스가 자동적으로 레지스트리에서 제거되기 때문에 발생하는 문제이다. 이렇게 제거된 것을 재등록하기 위해서 기존의 서비스들은 자신의 정보를 일정 간격으로 재등록을 시도하게 된다. Grid 환경을 지원해 주기 위한 고속의 네트워크의 상당한 부분을 이러한 재등록을 위해 사용하게 된다면 심각한 네트워크 트래픽이 발생될 것이다. 이러한 문제점을 개선하기 위하여 본 논문에서는 각 서비스마다 차등한 간격으로 서비스를 재등록하는 방법을 제안한다.

차등한 간격을 주는 방법은 다음 순서와 같다.

•단계1 : 초기 서비스가 레지스트리에 등록을 하면 초기의 재등록 시간을 서비스에게 할당해준다.
•단계2 : 서비스가 할당 받은 재등록 시간에 등록을 시도하여 성공을 하면, 재등록 시간을 1초 증가시키고, 재등록 시간을 초과하면, 현재 등록된 재등록시간의 1/2을 재등록시간을 할당하고 대기한다.
•단계3 : 서비스 재등록 시간이 초기 서비스 등록시간에 1/3이하가 되면 레지스트리에서 서비스 등록정보를 제거한다.

위와 같은 방법을 사용한다면, 각 서비스 별로 차등된 등록시간을 부여하여 재등록으로 인한 네트워크 트래픽을 분산시킬 수 있다. 그리고 기본적인 네트워크 트래픽에 따라서 등록시간을 초과한 경우에도 효과적으로 초기 재등록 시간을 변경함으로써 네트워크 트래픽으로 인한 서비스 불능 현상을 막을 수 있다. 또한 그림 3에서와 같이 일정 네트워크 트래픽만을 사용하여 서비스 등록을 위한 최상의 작업을 수행할 수 있다.



(그림 3) 네트워크 트래픽에 따른 재전송 시간

5. 결론

본 연구에서는 Grid 환경에서 사용자 또는 시스템이 필요로 하는 모든 자원에 대한 정보를 찾아주는 Discovery Service의 구현 방안을 기술하였다. 또한 이러한 방법으로 인해 발생되는 네트워크 트래픽 증가를 해결하기 위해 차등된 등록시간을 부여하는 방법을 제안하였다. 따라서 효과적인 자원의 발견을 위해 자원 등록 시에 발생되는 네트워크 트래픽을 절감할 수 있다.

향후에는 repository에서 제공해야 할 데이터 변경 통보, 데이터 분류 정보 관리, 데이터 보안, 데이터 중복에 대한 연구가 수행되어야 할 것이다.

참고문헌

[1] C. Steenberg, E. Aslakson, J. Bunn, H. Newman, M. Thomas, F. van Lingen, The Clarens Web Service ArchitectureCHEP 2003 La Jolla California
[2] C. Steenberg, J. Bunn, I. Legrand, H. Newman, M. Thomas, F. van Lingen, A. Anjum, T. Azim, The Clarens Grid-enabled Web Services Framework: Services and Implementation CHEP 2004 Interlaken
[3] C. Steenberg, E. Aslakson, J. Bunn, H. Newman, M. Thomas, F. van Lingen Clarens Client and Server Applications CHEP 2003 La Jolla California
[4] F. van Lingen, J. Bunn, I. Legrand, H. Newman, C. Steenberg, M. Thomas, P. Avery, D. Bourilkov, R. Cavanaugh, L. Chitnis, M. Kulkarni, J. Uk In, A. Anjum, T. Azim, Grid Enabled Analysis: Architecture, Prototype and Status. CHEP 2004 Interlaken
[5] M. Ballintijn, Global Distributed Parallel Analysis using PROOF and AliEn, in Proceedings of CHEP 2004.
[6] Jang-uk In, Paul Avery, Richard Cavanaugh, Laukik Chitnis, Mandar Kulkarni, Sanjay Ranka, SPHINX: A fault-tolerant system for scheduling in dynamic grid environments, in the proceesings of the 19th IEEE International Parallel & Distributed Processing Symposium (IPDPS 2005), Denver, Colorado, April, 2005.
[7] Jang-uk In, Adam Arbree, Paul Avery, Richard Cavanaugh, Sanjay Ranka, SPHINX: A Scheduling Middleware for Data Intensive Applications on a Grid, in the proceedings of Computing in High Energy Physics (CHEP 2004), Interlaken, Switzerland, September, 2004
[8] I. Legrand, MonALISA - MONitoring Agents using a Large Integrated Service Architecure International Workshop on Advanced Computing and Analysis Techniques in Physics Research, Tsukuba, Japan.
[9] A. Ali, A. Anjum, T. Azim, M. Thomas, C. Steenberg, H. Newman, J. Bunn, R. Haider, W. Rehman, JClarens: A Java Based Interactive Physics Analysis Environment for Data Intensive Applications, ICWS 2004 pp. 716-723