웹2.0 기반 그리드 기술 동향 보고서

한국과학기술정보연구원

3 cm

기술동향보고서

5 cm

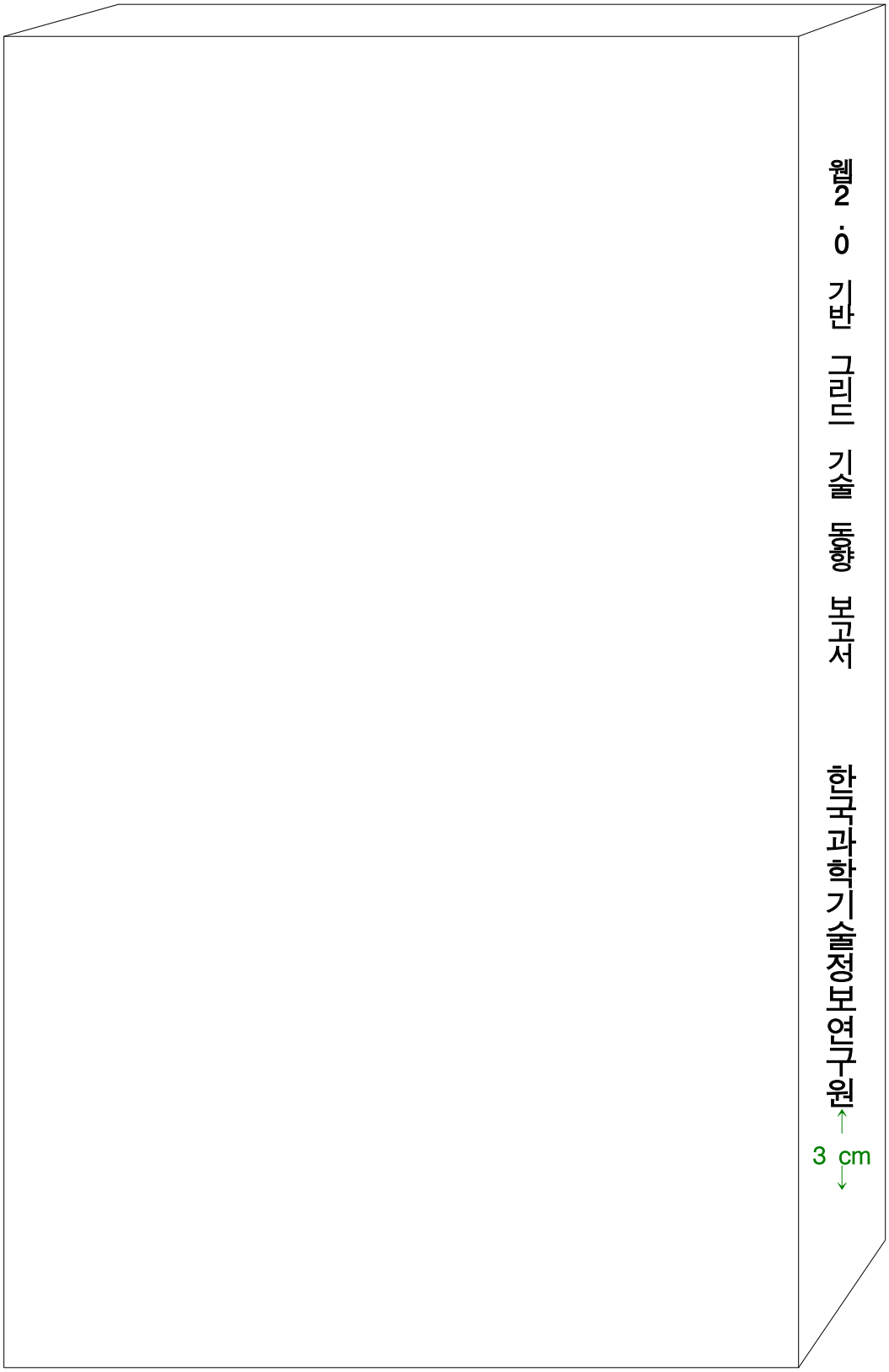
ISBN: 978-89-6211-296-2

웹2.0 기반 그리드 기술 동향 보고서

4 cm

한국과학기술정보연구원

겉표지 [뒷면]



웹2.0 기반 그리드 기술 동향 보고서

KISTI CI 미들웨어 연구팀

이필우, 함재균, 권오경, 곽재혁, 김상완

서울대학교

염헌영, 김형석, 유영진

요 약 문

Ⅰ. 제목

웹2.0 기반 그리드 기술 동향 보고서

Ⅱ. 보고서의 목적 및 필요성

웹2.0 기술은 최근 세계적으로 주목받고 있는 차세대 인터넷 서비스 기술로서 단순한 웹 인터페이스 기술을 떠나 사용자 커뮤니티 기반 플랫폼으로서 크게 주목받고 있으므로, 이를 슈퍼컴퓨터 기반 그리드 서비스에 웹2.0 개념과 기술을 적용하여 기존 그리드 서비스의 품질을 향상시키기 위한 연구가 요구된다. 또한 향후 웹2.0 기술과 그리드 기술과의 융합을 통해 그리드 서비스 환경의 진화를 가져올 것으로 예상되는 바, 웹2.0 기술을 슈퍼컴 기반 그리드 서비스에 효과적으로 적용하기 위한 기초 연구가 요구되며 이를 위하여 웹2.0 기술과 그리드 서비스 기술의 융합 가능성을 도출하고, 구현 가능한 서비스 모델을 창출하고자 한다.

Ⅲ. 보고서의 내용 및 범위

웹을 통해 모든 컴퓨팅 연구를 가능하게 하기 위한 인터페이스 모델을 정의하고 웹2.0 개념의 그리드 포털을 위한 새로운 기능을 제안하기 위하여 기존 그리드 웹 포털 기능을 분석하고 과학기술 컴퓨팅 분야에 적용할 수 있는 인터페이스 웹2.0 기술을 연구한다.
또한 웹2.0의 기술을 적용한 새로운 그리드 시스템 모델 설계를 할 수 있도록 그리드 미들웨어 표준 인터페이스인 WSRF와 웹2.0 서비스 제공기술을

비교 분석하고 웹2.0 기술을 바탕으로 한 그리드 미들웨어 인터페이스를 제안한다.
최종적으로 그리드 사용자 커뮤니티의 피드백을 활용하여 웹2.0의 협업 관련 기술 도입의 타당성 검토하고 사용자들의 자발적인 참여 및 공유 등을 통한 그리드 서비스 활성화 방안을 제시한다.

Ⅳ. 보고서 중점 사항

초기단계로 그리드 및 웹2.0 관련 기술 조사에 중점을 두었다. 그리드와 웹2.0에서 사용중인 사용자 인터페이스, 서비스 제공 모델 및 그리드와 클라우드 컴퓨팅(Cloud Computing) 간의 기술적, 개괄적 비교를 통하여 두 기술의 상호연동성에 대한 가능성과 타당성을 볼 수 있도록 하였다.

Ⅴ. 기대효과

웹2.0 기반기술을 활용하여 그리드 시스템을 구축할 경우, 사용자에게 친숙한 과학기술 커뮤니티 기반 플랫폼으로서 자리매김할 수 있고, 또한 기존 그리드 서비스의 품질을 향상시킬 수 있다. 웹2.0을 활용한 과학기술 인프라의 구축은 그리드 자원을 개방할 수 있는 가능성을 열어놓음으로써, 그리드 자원의 효율적인 활용을 제고하고 이에 기반한 과학기술 분야의 고급 응용 어플리케이션의 개발에 기여할 수 있다. 또한 OpenAPI 등을 이용하여 그리드 시스템의 개방을 유도함으로써 그리드 시스템의 활용 영역을 과학기술 분야뿐만 아니라 전반적인 정보기술 분야로 확대할 수 있을 것이고, 이를 통해 그리드 시스템을 활용한 다양한 응용 프로그램을 생산할 수 있는 기반기술을 제공할 수 있다.
웹2.0 기술을 활용한 그리드 시스템 구축은 현재 국내, 국제적으로 아직 연구된 바가 거의 없으며, 따라서 해당 기술을 표준화할 경우, 본 연구의 성과를 통하여 국제 표준화에 앞장설 수 있으며, 또한 확보된 기술의 선도성을 바탕으로 국익에 우선하는 방향으로 표준화 과정을 조율할 수 있을 것으로 기대한다.

목 차

제1장 개요 ····· 1

제2장 국내외 기술개발 현황 ····· 2

제3장 기술개발 내용 ····· 4

제4장 기대효과 ····· 25

제5장 참고문헌 ····· 26

그림차례

<그림 1> Cloud Computing의 개괄적인 구조 ····· 11
<그림 2> Grid와 Cloud의 접목 가능성 ····· 12
<그림 3> ROA와 SOA 비교 ····· 17
<그림 4> RSS 기동 방식 ····· 19
<그림 5> WS-BaseNotification 구현 ····· 22
<그림 6> Amazon S3의 보안 ····· 23
<그림 7> OpenID의 흐름도 ····· 24

제1장 개요

제1절 목적

웹2.0 기술은 최근 세계적으로 주목받고 있는 차세대 인터넷 서비스 기술로서, 사용자 커뮤니티 기반 플랫폼으로서 크게 주목받고 있다. 본 연구개발에서는 웹2.0의 기반기술에 대한 심도있는 접근을 통하여 과학기술 분야를 위한 새로운 사용자 인터페이스로서의 웹2.0의 가능성을 고찰하고자, 그리드 포털을 위한 웹2.0 기반의 새로운 인터페이스 모델과 그에 따르는 새로운 기능에 대한 연구를 수행하고자 한다.

인터넷 서비스 분야에서 확고한 위치를 잡아가고 있는 웹2.0의 경량 서비스 제공 기술이 가지는 장점을 그리드 미들웨어에 응용함으로써 그리드 미들웨어가 가지는 단점을 극복하고자 한다. 웹2.0의 mash-up 개념과 REST, XML-RPC, SOAP 방식 등의 OpenAPI 기술을 그리드 미들웨어 수준에 도입하는 것의 타당성을 검증하기 위하여, 기존의 그리드 미들웨어 표준 인터페이스인 WSRF와 웹2.0의 경량 인터페이스 기술을 비교 분석하고 실제로 웹2.0 서비스 제공 기술을 적용한 새로운 그리드 시스템 모델을 설계하고자 한다.

웹2.0의 철학인 참여, 공유, 개방의 개념을 과학기술을 위한 컴퓨팅 분야에 도입하여 사용자들의 자발적인 참여 및 공유 등을 통한 그리드 서비스 활성화 방안을 연구하고자 한다.

제2절 중요성

최근 십년동안 분산 컴퓨팅의 화두는 그리드 컴퓨팅이었으며 다양한 연구 개발을 통해 괄목할만한 성과를 이루었다. 그러나 그리드 초창기에 기술적인 성공을 이루리라고 확신하던 Web Service기반 기술의 쇠퇴로 인해 그리드 시스템의 변화가 요구되는 상황이다. 다양한 과학기술 분야에 걸쳐서 그리드 시스템이 가지는 고성능 자원들을 사용하여 연구의 질을 높일 수 있음에도 불구하고 복잡한 그리드 내부 시스템 등으로 인해 아직은 활용도가 낮은 상황이다. 이에, 그리드 기반 기술인 Web Service 기술에 대한 집착보다는 인터페이스의 개방을 통해 새로운 영역을 개척하고 있는 웹2.0 기술에 대한 연구를 통하여 그리드 시스템과 접목시킬 수 있는 가능성에 대한 제고가 필요하다.

과학기술 연구는 연구성과에 대한 공유와 이를 비판하고 재생산하는 과정을 반복하여 순환한다. 따라서 연구성과의 생산과 유통에 있어서 웹2.0의 Weblog, RSS/Atom 등의 기술 등의 도움을 통하면 획기적인 공동연구 환경을 제공할 수 있다. 또한, 웹2.0의 RIA 개념을 과학기술 컴퓨팅 분야에 적용하여 AJAX, Flex 등의 다양한 기능을 가지는 응용프로그램을 과학자들에게 제공하여 연구과정의 효율성을 제고할 수 있을 것이다.

따라서, 기존의 그리드 기술과 웹2.0 기술에 대한 심도있는 연구가 필요하며, 두 기술에 대한 비교 분석과정을 통하여, 그리드 시스템 구축에 있어서 웹2.0 모델을 적용할 수 있는 가능성을 찾아보는 것이 필요하다.

제3절 기대 효과

웹2.0 기반기술을 활용하여 그리드 시스템을 구축할 경우, 사용자에게 친숙한 과학기술 커뮤니티 기반 플랫폼으로서 자리매김할 수 있고, 또한 기존 그리드 서비스의 품질을 향상시킬 수 있다. 웹2.0을 활용한 과학기술 인프라의 구축은 그리드 자원을 개방할 수 있는 가능성을 열어놓음으로써, 그리드 자원의 효율적인 활용을 제고하고 이에 기반한 과학기술 분야의 고급 응용 어플리케이션의 개발에 기여할 수 있다.

OpenAPI 등을 이용하여 그리드 시스템의 개방을 유도함으로써 그리드 시스템의 활용 영역을 과학기술 분야뿐만 아니라 전반적인 정보기술 분야로 확대할 수 있을 것이고, 이를 통해 그리드 시스템을 활용한 다양한 응용 프로그램을 생산할 수 있는 기반기술을 제공할 수 있다.

웹2.0 기술을 활용한 그리드 시스템 구축은 현재 국내, 국제적으로 아직 연구된 바가 거의 없으며, 따라서 해당 기술을 표준화할 경우, 본 연구의 성과를 통하여 국제 표준화에 앞장설 수 있으며, 또한 확보된 기술의 선도성을 바탕으로 국익에 우선하는 방향으로 표준화 과정을 조율할 수 있을 것으로 기대한다.

제2장 국내외 기술개발현황

제1절 국내 기술현황

KISTI는 국내의 그리드 기술의 확산 및 연구/개발을 위해 여러 산학 단체와 협력하여 과제를 수행하고 있다. 정보통신부와 함께 지난 2002년부터 국가 그리드 인프라(K*Grid)를 구축하여 과학 기술 연구분야에서 주로 활용해왔으며, 2005년부터는 그리드 미들웨어 개발 결과와 인프라를 기반으로 그리드 상용화를 위한 노력을 기울이고 있다. K*Grid 의 구축 과정에서, 시간과 장소에 구애받지 않고 일반 사용자들도 쉽게 그리드 자원을 활용할 수 있는 환경을 만들기 위해 웹기반의 그리드 서비스 플랫폼을 목표로 하였다. K*Grid Portal은 웹기반의 통합 사용자 인터페이스 개발, 웹기반의 이기종 그리드 자원에 대한 실시간 모니터링 기능 제공, 웹기반의 이기종 그리드 자원 계정 기능 제공들을 요구분석 하였지만, 웹2.0에 대한 고려가 아키텍쳐에 반영되어 있지 않기 때문에, 사용자들의 참여/공유/개방 과 관련한 지원이 현재로서는 부족한 실정이다.

OGF-KR은 정보통신부의 차세대 인터넷 기반 구축을 위한 국가 Grid 기본 계획의 일환으로 2001년에 설립된 GFK를 OGF의 한국 대표 포럼으로 공식적으로 추인되면서 한국의 그리드 커뮤니티를 대표하는 국제 표준화 포럼으로 발전시켜 나가기 위하여 재조직, 개명한 그리드 포럼이다. OGF-KR는 그리드 응용, 그리드 미들웨어, 그리드 인프라, 그리드 비즈니스로 분과되어 각기 접근성 및 표준화 등의 연구를 수행한다. 다만 진행중인 다수의 연구들이 전통적인 그리드 플랫폼에 국한되어 있어서, 현재 활발히 진행되고 있는 웹2.0 등과의 연계 기술 등에 관한 논의는 부족한 편이다.

국내에서의 웹2.0의 수익모델은 아직까지 콘텐츠, 사용자, 광고주를 중심으로 한 콘텐츠정기구독, 서비스이용 수수료, 광고 수익 등으로 Web1.0의 수익모델과 비교해 볼 때 거의 차이가 없다. 미국과 비교해 볼 때 국내의 웹2.0 서비스는 상위 10%의 웹사이트가 전체 트래픽의 80%를 차지하고 있으며, 이중 포털사이트가 전체 트래픽의 64%를 차지하고 있는 등 일부 주요 서비스에 트래픽이 극단적으로 집중되어 있는 상황이라 대부분의 Long tail ecosystem을 위한 환경과 수익성이 극히 미미한 상태이다. 또한 국내에서는 광고 및 UCC 기반 서비스 외에는 이렇다 할 많나 웹2.0 서비스 수익 모델이 전무한 상태이다. 전문가들은 UCC기반 서비스 모델 또한 콘텐츠를 제공하는 사용자와 이를 소비하는 이용자 모두에게 수익을 제공할 만한 비즈니스 플랫폼의 부재를 근본적인 문제로 지적하고 있다.

제2절 국외 기술현황

OGF (Open Grid Forum)는 그리드 관련 기술의 발전과 표준화 작업을 선도해오고 있는 세계적인 그리드 표준화 기구로서 그리드 기술의 보급과 정착을 위해 상호호환성을 위한 표준 인터페이스 연구개발에 주력하고 있다. 현재 OGF의 그룹은 크게 WG (Working Group), RG (Research Group), 그리고 CG (Community Group)로 분류되어 운영되고 있다.

웹2.0기술은 컨텐츠 유통과 상거래 방식의 변화를 요구한다. 컨텐츠의 생산과 유통방식의 변화는 웹2.0에서 블로그와 RSS, 그리고 Atom 등의 기술로 요약된다. 미국에는 국내와 달리 매우 다양한 분야에서 웹2.0 기술을 활용한 서비스가 널리 이루어지고 있다. “del.icio.us" 라는 소셜 북마크 서비스는 컨텐츠뿐만 아니라 컨텐츠의 위치(링크)를 가치 있는 데이터로 보고, 사용자들이 직접 중요한 데이터의 위치를 수집하고 공유하게 했다. 이 과정에서 기존에는 사용자 영역으로 보지 않았던 메타데이터까지도 태그를 통해 사용자가 생산하게 된다. 또한, 사용자 - 태그 - 데이터 간의 관계를 적극적으로 시각화하고 서비스화하여 사용자들이 새롭고 유용한 정보를 찾는데 활용하게 했다. "flickr"에서는 사용자가 사진을 올릴 수 있는 공간만 제공하는 것이 아니라, 이 사진을 웹에서 다른 사용자와 쉽게 공유할 수 있도록 했다. 사용자 간의 ‘관계 맺기’ 기능은 공유 기능을 강화하면서, 좀더 나와 관련있는 데이터를 쉽게 소비할 수 있게 한다.

최근에 들어 Grid Service와 웹2.0 을 연계 가능성에 대해 모색하는 시도가 이루어지고 있다. 인디애나 대학의 Geoffrey C. Fox는 Semantic research grid를 위해 e-Science 인프라에 웹2.0 서비스를 통합하는 연구를 제안하였다. 그리드 기술은 주로 원격의 Resources를 안전하게 접근하는 문제에 대해서 집중해왔지, scientific community를 어떻게 구성할 것인가의 문제는 해결하지 못하였다. 따라서 Geoffrey는 웹2.0의 tagging과 bookmarking 서비스를 도입하여 지역적으로 멀리 떨어진 다수의 연구기관에서 Research interest가 동일한 연구자들을 찾아줄 수 있도록 하는 CITEAM Portal을 개발하였다. 이를 "Tag-based Researcher Matchmaking"이라 한다. 이 포탈이 도입한 Bookmarking 서비스는 Connotea에서 제공하는 서비스를 Backend로 하여 구성하였다. 사용자가 포탈에 등록을 하고, 자기 자신의 Research interest 에 대한 설명을 tagging해 두고, 북마킹 해둔 paper 등을 이용해 프로파일 정보를 점점 축적해 나간다. 이러한 프로파일을 바탕으로 User Matchmaking이 이루어지고, 해당 연구자들에게 Collaboration의 기회를 제공해 주게 된다.

제3장 기술개발 내용

제1절 Grid와 웹2.0 기반 Cloud Computing

1 Cloud Computing의 개괄

Cloud Computing 이라는 것은 방대한 계산 자원들이 Third-Party에 의해 가동되고 소유되는 계산 방식을 의미한다. 이러한 방식은 다양한 인터넷 기술들을 "서비스" 의 형태로 고객들에게 제공할 수 있고, 이 덕분에 고객들은 리소스를 관리하는 Infrastructure에 대한 자세한 지식이 없이도 그것들을 활용할 수 있게 된다. Cloud Computing Infrastructure는 Computing/Storage 리소스들을 가상화한 데이터 센터를 기반으로 신뢰성있는 서비스를 제공하도록 구성되어 있다.

충분한 비용을 지불하고 원격의 리소스를 사용하여 원하는 서비스를 얻는다는 점에서 기존의 Utility Computing이나 On Demand Computing과 유사하다고 볼 수도 있다[1]. 하지만 Cloud Computing을 활용한 실제의 수익모델이 속속 등장하고 있고, REST와 같은 구체적인 아키텍쳐 스타일을 적극적으로 제안한다는 점에서 기존의 계산 방식들과는 차이가 있다.

표1은 현재 기업이나 연구소에서 개발 및 보유하고 있는 Cloud Platform에 대한 비교이다[6]. 각각에 대해 간단히 살펴보면 다음과 같다.

EC2는 Linux 기반의 애플리케이션을 실행할 수 있는 가상 환경을 제공한다. Xen 기반의 가상 머신에 AMI(Amazon Machine Image)를 설치하여 원하는 서비스를 Deploy할 수 있다. S3의 경우 REST 기반의 Storage 서비스를 제공한다. Google App Engine은 사용자가 Python 으로 작성한 웹애플리케이션을 배포할 수 있도록 한다. Google App Engine은 datastore, URL fetch, image manipulation등의 API를 제공한다. 이 세가지 서비스에 대해서는 아래에 자세한 설명을 하도록 한다.

Microsoft Live Mesh는 사용자가 애플리케이션 및 데이터들을 저장할 수 있는 Storage 서비스를 제공한다. 이 데이터들은 전세계 어느 곳에서나 다양한 디바이스로 접근이 가능하게 된다. Sun network.com은 Solaris OS, Java, C, Fortran-기반의 애플리케이션을 실행할 수 있는 환경을 제공한다. 사용자가 실행에 필요한 라이브러리나 스크립트 파일을 묶어서 Sun Grid 포탈을 통해 업로드하면, 계산 리소스를 사용하여 결과를 얻을 수 있다. 엄밀히 이것은 Grid Computing의 범주에 포함되지만, Grid Service를 이용할 수 있는 인터페이스를 아주 단순화했다는 의미에서 Cloud Platform으로 분류를 한 것으로 여겨진다. GRIDS Lab Aneka는 SOA기반의

Table 1: Comparison of some representative Cloud platforms.

System / Property	Amazon Elastic Compute Cloud (EC2)	Google App Engine	Microsoft Live Mesh	Sun Network.com (Sun Grid)	GRIDS Lab Aneka
Focus	Infrastructure	Platform	Infrastructure	Infrastructure	Software Platform for enterprise Clouds
Service Type	Compute, Storage (Amazon S3)	Web application	Storage	Compute	Compute
Virtualisation	OS Level running on a Xen hypervisor	Application container	OS level	Job management system (Sun Grid Engine)	Resource Manager and Scheduler
Dynamic Negotiation of QoS Parameters	None	None	None	None	SLA-based Resource Reservation on Aneka side.
User Access Interface	Amazon EC2 Command-line Tools	Web-based Administration Console	Web-based Live Desktop and any devices with Live Mesh installed	Job submission scripts, Sun Grid Web portal	Workbench, Web-based portal
Web APIs	Yes	Yes	Unknown	Yes	Yes
Value-added Service Providers	Yes	No	No	Yes	No
Programming Framework	Customizable Linux-based Amazon Machine Image (AMI)	Python	Not applicable	Solaris OS, Java, C, C++, FORTRAN	APIs supporting different programming models in C# and other .Net supported languages

표 1 대표적인 클라우드 플랫폼들

Enterprise Grid 플랫폼이다[7]. Service Provider가 서비스를 시작하기 위해서는 Aneka Container를 띄우면 되는데, 이것은 Enterprise Grid의 서비스들과 상호작용하는 중간자로서의 역할을 하게 된다.

앞서 살펴본 것과 같이 현재 매우 다양한 Cloud Platform 또는 이 지속적으로 연구/개발되고 있으며, 이 플랫폼을 사용하게 될 기업들의 수도 크게 늘어날 것으로 보는 시각도 존재한다[8]. Cloud Computing의 인기는 웹2.0이라는 기술적 트렌드와 밀접하게 연관되어 있다. 사실 웹2.0 기술의 근간을 이루는 철학은 사용자들의 참여와 공유이다. 개별 사용자들이 콘텐츠의 생성자이자 소비자이고, 기존의 서비스들을 Mash-up 하여 새로운 서비스를 창출할 수 있다. 현재 Cloud Computing 기술을 선도하고 있는 대부분의 업체들의 경우 매우 간단한 서비스 인터페이스(REST)를 구현하였는데, 이러한 시도가 바로 사용자들의 서비스 참여를 독려하였

고, 그 결과 급속도로 널리 퍼지게 된 원동력이 되었다고 여겨진다. 실제로 Cloud Computing Platform을 제공하는 세가지 예를 살펴보도록 하겠다.

a) AWS: S3

S3는 어떤 양의 데이터든, 언제나 어디에서 업로드/다운로드 할 수 있는 간단한 웹서비스를 제공한다. 이것은 개발자가 Web-scale의 애플리케이션을 쉽게 작성할 수 있도록 해준다[9]. S3를 이루는 컴포넌트는 크게 Object와 Bucket으로 나눌 수 있다. Object는 S3에서 데이터를 저장하는 가장 기본적인 단위이다. Object는 데이터와 메타데이터로 이루어져 있는데 메타데이터는 Object를 기술할 수 있는 (이름, 값) 페어의 집합을 의미한다.

Bucket은 S3내의 Object들을 담을 수 있는 컨테이너 역할을 한다. 모든 Object는 어떤 Bucket에든 포함되어야 한다는 제약 조건이 있다. 이 때문에 Bucket은 S3에서 Object의 네임스페이스를 부여해주는 기능을 하고, 사용자 계정의 권한 체크와 연계된다. Bucket내의 Object들은 유일하게 식별할 수 있도록 하는 Key를 가지게 된다.

Amazon S3는 REST API를 구현하고 있다. SOAP 기반의 웹서비스와는 달리 REST 기반의 서비스는 매우 단순하고 XML 파싱 오버헤드가 적기 때문에 좋은 성능을 보장할 수 있다[10]. 사용자는 리소스에 대해 표준 HTTP 요청을 사용하여 생성, 삭제, 업데이트 등의 연산을 수행할 수 있다. 이 경우 리소스는 S3의 Bucket과 Object가 되는 것이고, 지원하는 연산은 PUT, GET, DELETE, POST 연산이 된다. 가령 HTTP Request의 헤더에 B 라는 Bucket 이름을 넣고, PUT 연산을 요청하면 B라는 이름의 새로운 Bucket을 추가하라는 명령이 된다. C라는 Bucket 이름과 GET 연산을 요청하면, Bucket C가 포함하고 있는 Object들의 URI 리스트를 가져오는 명령이 된다. O라는 Object 이름과 PUT 연산을 요청하면, O라는 새로운 Object를 생성하라는 명령이 된다. 단, 이 때 Object의 URI로부터 Object가 포함되어야 할 Bucket의 이름을 파악하여 적절한 권한을 체크한 다음에, 정상적으로 연산이 수행된다. 이처럼 Resource 들을 미리 정의하고, 해당 Resource에 적용가능한 연산의 종류를 공개하는 RESTful 서비스를 Resource-Oriented-Architecture(ROA) 스타일이라 정의하기도 한다.

S3 웹서비스를 제공함에 있어서 가장 중요한 이슈는 보안 문제이다. 비용을 지불하여 인증된 사용자가 허락된 만큼만의 리소스를 사용할 수 있도록 해야 하고, 허가받지 못한 요청은 서비스로부터 차단해야 할 것이다. S3에서는 REST API를 제공하는데, 이 때 HTTP 요청의 헤더부분에 "Authorization"이라는 필드를 두어 사

용자 자신의 Key ID와 메시지의 Signature를 포함하도록 되어 있다. S3 에서는 그 필드의 Key ID로부터 사용자와의 Shared Key를 유추할 수 있고, 이로 메시지의 Signature를 직접 계산하여 HTTP 요청 내의 Signature와 동일한지의 여부를 비교하기 때문에 메시지의 Integrity를 보장할 수 있게 된다.

사용자와 관련해서 다섯가지의 권한 타입이 정의되어 있다. Owner의 경우는 Bucket이나 Object를 생성한 사람에게 주는 권한이고, User-by-Email은 해당 e-mail과 일치하는 사용자에게 권한 부여, User-by-Canonical-Representation은 해당 Canonical ID와 일치하는 사용자에게 권한 부여, AWS User Group은 그룹에 해당하는 모든 사용자에게 권한 부여, AllUsersGroup은 모든 사용자에게 권한을 부여하는 것이다.

b) AWS: EC2

EC2는 Amazon이 운영하고 있는 거대한 규모의 데이터 센터의 계산 리소스를 필요한 만큼만 사용할 수 있도록 해주는 웹서비스이다. 이 계산 리소스는 애플리케이션이 요구하는 한 수분 단위로도 제공이 가능하며 사용한 만큼만 비용을 지불하면 되는 서비스 구조이다. EC2는 다음과 같은 목적으로 사용되는 경우가 많다.

첫 번째는 확장성있는 애플리케이션을 제작하는 경우이다. 사용자의 요구에 따라 필요한 리소스가 줄어들거나 늘어가는 경우에 대해서도 동적으로 사용하는 리소스의 양을 조절할 수 있기 때문에, 짧은 시간내에 아주 많은 수의 요청이 들어오는 시나리오에 대처할 수 있는 애플리케이션을 작성할 수 있다.

두 번째는 Batch Processing에도 유용하게 사용할 수 있는데, 비디오나 이미지 처리 혹은 금융 데이터 분석과 같이 오프라인으로 이루어져도 되지만 계산량이 매우 많은 작업들의 경우 EC2가 제공하는 풍부한 리소스를 사용하여 원하는 결과를 빠른 시간내에 얻어낼 수 있다. 세 번째는 Fault-Resilient 애플리케이션을 작성하는 문제인데, 애플리케이션을 여러 위치에 배치시킬 수 있으므로 특정 물리적인 위치에 오류가 발생하더라도 가용성을 높일 수 있게 된다.

EC2는 Xen기반의 가상 머신 환경을 이용하며, 따라서 원하는 애플리케이션을 디플로이하기 위해서는 설치환경이 포함된 이미지 파일(Amazon Machine Images)을 업로드해야 한다. 이후에는 SSH 로그인을 위한 Key pair를 생성하고, AMI로부터 가상 인스턴스를 생성한다. 네트워크 접근을 Authorize한 이후에는 가상 인스턴스에 접속해서 애플리케이션을 수행할 수 있다. 현재 EC2 서비스의 경우 SOAP은 지원하지만 REST는 지원하지 않는다.

"The New York Times"의 경우, 100개의 EC2 인스턴스와 Hadoop 애플리케이션

을 사용하여 S3에 저장된 4TB 상당의 이미지 데이터를 프로세싱 하여 110만개의 PDF 파일을 생성해 내는 것으로 알려져있다. 이 때 계산 리소스와 저장 공간 리소스를 사용하는데 드는 비용은 하루에 240$ 정도로서 매우 저렴하다.

c) Google App Engine

사용자가 작성한 웹애플리케이션을 Google이 만들어 놓은 Infrastructure 위에서 실행할 수 있도록 해주는 플랫폼이다[11]. 사용자가 특정 서비스를 운영하고 싶을 때, "App Engine"을 사용하면 따로 서버를 유지해야 할 필요가 없다. 단지 작성한 애플리케이션을 업로드하기만 하면 되므로, 서비스를 시작하는 것이 용이하고, 유지보수가 간편하며, 네트워크 트래픽이나 데이터 공간을 동적으로 확장하거나 축소하면서 운영할 수 있다.

App Engine이 가진 몇가지 특징 중 하나는, 애플리케이션에 Sandbox 환경을 제공한다는 것이다. 사용자에 의해 업로드된 애플리케이션은 하부의 운영체제에 대한 제한된 접근만이 허용되어 안전하고 신뢰성있는 환경에서 실행될 수 있다. 예를 들면 애플리케이션은 App Engine이 제공하는 URL fetch나 email service와 같은 API를 통해서만 인터넷의 다른 컴퓨터에 접근할 수 있고, 다른 컴퓨터들 역시 표준 포트의 HTTP 요청만을 통해서 애플리케이션에 연결할 수 있도록 제한되어 있다. 또한 애플리케이션은 임의로 파일시스템에 내용을 쓸 수 없으며, 애플리케이션에 의해 업로드된 파일만을 읽어들일 수 있다. 애플리케이션은 반드시 App Engine의 "datastore" 서비스를 이용해서만 데이터를 저장할 수 있도록 제한된다.

App Engine은 쿼리 엔진과 트랜잭션을 수행할 수 있는 강력한 저장 서비스를 제공하는데 이를 datastore라 부른다. datastore는 아주 단순한 인터페이스를 통해 Google의 확장성 있는 Infrastructure를 사용할 수 있도록 해준다. 각 data object를 entity라 부르고, 각 entity는 하나 이상의 property를 가지게 된다. property는 (이름, 값)의 pair를 의미하고, 혹은 일대다 관계나 다대일 관계를 표현하기 위해 다른 entity로의 참조 역할을 하기도 한다.

2 Grid와 Cloud의 비교

Grid computing과 Cloud Computing 기술은 분산 컴퓨팅에 근원을 두고 있다는 점에서 유사한 부분이 매우 많이 존재한다. 분산 컴퓨팅이란 다수의 계산 자원, 저장 공간 자원 등을 포함하고 있는 하드웨어 및 소프트웨어 시스템을 다루는 개념이다. PC 자원이 점차 늘어나고, 인터넷과 같은 네트워크가 형성되면서 물리적으

로 흩어져있는 자원들을 효율적으로 관리하려는 연구가 다수 시도되고 있고, 이것이 Grid/Cloud Computing에서 추구하는 일이다. 분산 컴퓨팅의 주 목적중 하나가 확장 가능한 시스템을 만들고, 리소스의 사용률을 높이고자 한다는 점에서 최근의 연구 동향과 그 궤를 같이 한다고 볼 수 있을 것이다.

최근 EGEE에서 발표한 보고서[1]에 따르면 Grid Computing은 기술의 정점을 넘어 어느 정도 안정기에 접어들어 곧 상업적인 용도로 사용될 시점에 있고, Cloud Computing은 이제 막 기술의 정점에 올라 대중 매체의 각광을 받으며 소개되고 있다고 평가하고 있다. 일부에서는 Grid Computing을 구시대의 기술로 여겨, 새로운 기술인 Cloud Computing에 의해 침체될 것이라는 의견도 제시하고 있다.

하지만 아직 이러한 판단은 섣부르다고 여겨진다. Grid Computing과 Cloud Computing은 각각의 장점을 극대화할 수 있는 영역이 서로 다른 경향이 있다. Job을 실행하는데 있어서 둘은 서로 다른 패턴을 보이게 되는데 전자의 경우 정해진 시간내에 매우 많은 수의 작업을 실행하면서 다량의 리소스를 사용하는데 반해 후자는 상대적으로 적은 리소스로 오랜 기간 서비스하는 패턴을 보이게 된다. Grid Computing은 Cloud Computing이 가지지 못한 고수준의 서비스를 제공할 수 있으며, Cloud Computing은 Grid Computing이 가지지 못한 매우 쉬운 인터페이스를 제공할 수 있다. 물론 양쪽의 기술이 발달해가면서 서로의 장점을 배우며 모습이 바뀔 가능성은 있으나, 현재 둘간의 차이는 분명히 존재하고 어느 한쪽이 다른 한쪽을 완전하게 압도하지 못하는 상황이다. 둘의 기술을 비교한 결과가 아래에 있다.

Cloud Computing은 기업에 의해 주도되는 만큼, 내부 구조에 대해 자세히 분석

Features	Grid	Cloud(AWS/EC2+S3)
Ease of use	Heavy	Light
Ease of deployment	Heavy	Unknown
Data File Transfer Service	Yes	No
Workload Management	Yes	No
REST to storage	No	Yes
REST to computing	No	??
SOAP to storage	No	Yes
SOAP to computing	Yes	Yes
Support for job execution	Yes	Yes

표 2 Grid와 클라우드 컴퓨팅 비교[3]

된 결과물이 존재하지 않는다. 단지 서비스를 이용할 수 있는 API 정도만 공개되어 있을 뿐이다. 이를 다이어그램으로 나타내면 표2와 같다.

Cloud는 기업에서 구축한 서비스가 실행중인 플랫폼이라 볼 수 있다. User Interface는 공개되어 있기 때문에(REST), 사용자나 애플리케이션은 이를 통해 Cloud와 통신하여 서비스를 이용할 수 있고 이를 Front End라 볼 수 있다. Cloud의 Back End에서는 실제 계산 리소스, 저장 공간 리소스등을 관리하는 프레임웍이 존재한다. 이를 위해 Resource Interface를 사용해야만 하는데, 이것은 Cloud를 구축한 기업에서 임의로 정한 규약을 따를 뿐 외부에 공개되지는 않는다. 사용자의 입장에서는 Resource Interface를 알 필요 없이, User Interface만으로 원하는 서비스를 이용할 수 있다.

하지만, Resource Interface를 공개하지 않고 또한 표준화되어 있지 않기 때문에 다른 문제가 발생할 수 있다. 그것은 바로, 리소스를 제공할 수 있는 주체가 Cloud를 구축한 기업 자신밖에는 존재할 수 없다는 것이다. 기업에서 구축한 Cloud에 심각한 결함이 있거나, 기타 천재지변 등으로 인해 리소스에 문제가 발생한다면 서비스 전체가 멈추어 버릴 수 있는 상황이 생기게 되고, 이로 인한 손해는 매우 크게 될 것이다. 실제로 2008년 2월 15일에 Amazon의 S3가 수시간 동안 멈춘 사건이 발생했다. S3를 이용하는 업체들의 구체적인 피해수치는 밝혀진 바 없으나, 앞으로 더 많은 기업들이 Cloud Computing 플랫폼으로 옮겨올 가능성이 높다는 점에서 이러한 가용성의 문제는 더욱 심각하게 다가온다.

그리드의 경우에는 Resource Interface까지도 표준화가 되어 모두 공개가 되어 있기 때문에, 한 벤더에 Lock-in 되는 경우를 막을 수 있고 다양한 업체 및 연구소의 리소스를 결합할 수 있게 되어 있다. 따라서 특정 물리적인 위치의 리소스들에 문제가 생기더라도 전체 서비스가 중단하는 일은 막을 수 있을 것이다.

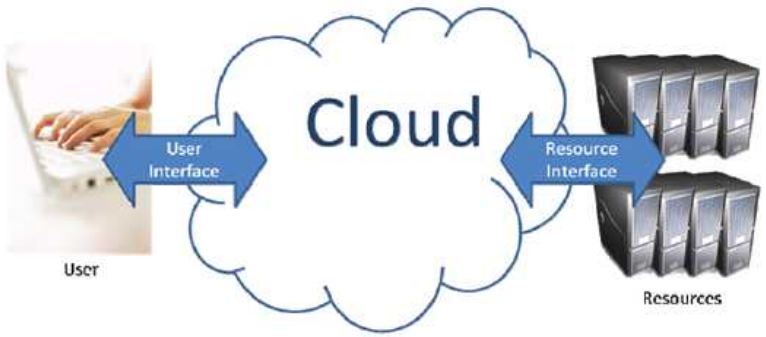


Figure 1: Cloud interfaces high-level diagram

그림 3 클라우드 컴퓨팅의 개괄적인 구조

하지만, 그리드의 대표적인 단점으로 꼽을 수 있는 것은 사용의 편의성 문제일 것이다. 수많은 기계들간의 상호운용성을 위해 여러 가지 표준화 작업이 이루어졌는데, 그 결과 서비스의 개발과 이용에 매우 많은 지식을 요구하게 되었고, 그 복

잡함으로 인해 넓은 사용자 층을 확보하는 데에는 어려움을 겪을 수밖에 없었다. 또한 지나치게 많은 기능을 포함한 프로토콜(예를 들면 SOAP)을 사용한 결과, 아주 단순한 서비스를 운영하더라도 높은 오버헤드[10]를 감수할 수 밖에 없었다. 그리드 컴퓨팅이 보여준 가능성에도 불구하고, 기업들이 Cloud Computing 쪽으로 관심을 기울이고 있다는 것은, 그만큼 사용의 편의성도 중요한 관심사라는 의미가 될 것이다.

그리드가 Technical Hype를 지나서 안정기에 접어들었고, 이제 산업에서 실제로 사용될 수 있는 수준으로 도약하고 있다는 의견[5]이 있다. 그리드 컴퓨팅이 한 단계 더 도약하기 위해서는 Cloud Computing의 장점들을 흡수할 필요가 있다고 여겨진다. 특히 사용자의 참여를 독려하는 웹2.0 기반의 사용자 인터페이스와, REST API와 같은 간결한 인터페이스가 그리드 컴퓨팅의 확산에 큰 도움을 줄 수 있을 것이라 여겨진다.

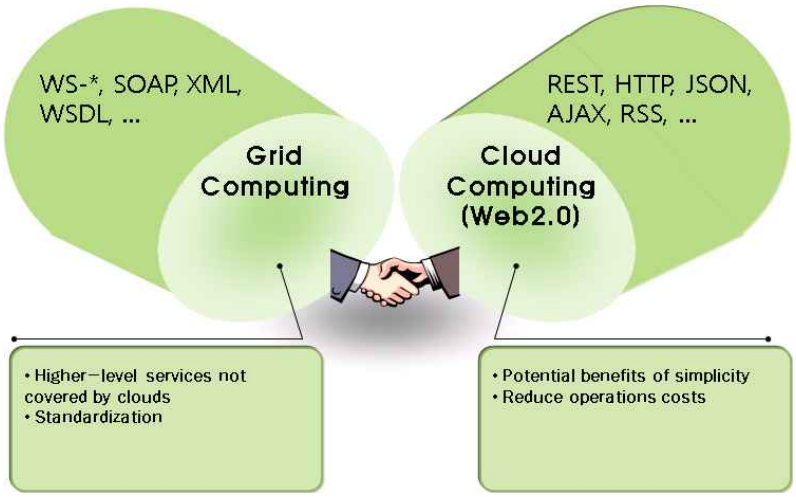


그림 4 Grid와 Cloud의 접목 가능성

제2절 사용자 인터페이스 비교 분석

1 웹2.0

AJAX

JavaScript, XML, HTML, CSS 기술을 통틀어서 AJAX 라 칭하는 경우가 많다. AJAX는 브라우저의 기술로서 웹서버와는 무관하다. 다시 말해 AJAX를 위한 서버라는 것은 특별히 존재하지 않는다. 어떠한 인터넷 서버라도 AJAX 페이지를 제공

할 수 있다.

AJAX 애플리케이션은 필요한 데이터만을 웹서버에 요청해서 받은 후 클라이언트에서 데이터에 대한 처리를 할 수 있다. 보통 SOAP이나 XML 기반의 웹 서비스 프로토콜이 사용되며, 웹 서버의 응답을 처리하기 위해 클라이언트 쪽에서는 자바스크립트를 쓴다. 웹 서버에서 전적으로 처리되던 데이터 처리의 일부분이 클라이언트 쪽에서 처리되므로 웹 브라우저와 웹 서버 사이에 교환되는 데이터의 양과 웹서버의 데이터 처리량도 줄어들기 때문에 애플리케이션의 응답성이 좋아진다[12].

Gadget

Gadget이란 특정한 기능을 가진 조그마한 기술적인 물체를 의미하는 말로써, 애플리케이션 개발에 있어서 독립된 응용프로그램을 만드는 대신 각각 다른 기능을 가진 다수의 Gadget을 하나의 환경에서 돌아가도록 하는 프로그램이 개발되고 있다. 대표적으로 Google Gadget, Microsoft Gadget 등이 있다.

구체적으로 Google Gadget의 경우 아주 간단한 시계와 같은 애플리케이션부터, 애니메이션, 사운드, 시각적인 효과 등을 가진 복잡한 애플리케이션까지 매우 다양한 형태가 공개되어 있다. 구글의 경우 Gadget을 개발할 수 있는 SDK를 공개하고 있기 때문에 Gadget API를 사용해서 미니 애플리케이션을 개발할 수도 있고, 구글 데스크탑의 Search API와의 연계도 가능하다. 또한 작성한 Gadget을 구글 데스크탑이나 iGoogle 홈페이지 또는 Mac Dashboard등의 다양한 위치에 배치할 수 있다.

Web Widget

Web Widget 이란 미니 웹 애플리케이션으로서 웹페이지나, 블로그 등에 삽입되어 쉽고 빠르게 방문자들에게 사용자에 특화된 정보들이나 부가기능을 제공하는 소프트웨어를 말한다[13]. Widget은 곧바로 사용할 수 있는 포맷을 가지고 있고, 시각적으로 화려하거나 사용자에 의해 개인화할 수 있다는 특징을 가지고 있다. Widget은 그림이나 그래프, 통계치, 결과 요약 등의 형태로 존재할 수 있다. 사용자는 단순히 코드를 복사해서 웹페이지에 삽입만 하면 바로 Widget을 사용할 수 있도록 되어 있다. Widget을 구성하는 코드는 자바스크립트이거나 아도비의 플래시 플러그인 혹은 윈도우즈의 실버라이트 플러그인 등 다양한 방법으로 구현할 수 있다.

Flex

최근 웹 애플리케이션의 프리젠테이션 레이어를 개선하고자 X-Internet, RIA 등의 개념이 소개되고 있는데, 이것을 선도하는 기술 중에 대표적인 것인 아도비의 Flex 이다. 아도비의 Flex는 플래시 플랫폼 기반의 리치 인터넷 애플리케이션을 쉽고 빠르게 개발할 수 있는 기술들을 총집합을 의미한다.

Multi-Tier 모델에서 Flex 애플리케이션은 프리젠테이션 레이어를 차지한다. 페이지 기반의 HTML 애플리케이션과는 달리, Flex 애플리케이션은 Stateful Client를 제공할 수 있기 때문에 화면에 중요한 변화가 나타난다 하더라도 매번 페이지를 리로딩할 필요가 없게 된다. Flex와 Flash Player는 서버 쪽의 컴포넌트와 효율적으로 데이터를 주고받을 수 있는 방법이 존재하기 때문에 굳이 클라이언트가 화면을 리로드하지 않아도 되는 경우가 많이 있다. 비록 현재 대부분의 브라우저들이 XMLHttpRequest와 같은 비동기적인 요청들을 처리할 수 있게 되면서 HTML 애플리케이션에서도 사용자의 페이지 리로딩을 최소화 할 수 있게 되기는 하였지만, 여전히 시각적인 화려함 등의 리치 클라이언트 구현 측면에서 Flex의 장점은 부각되고 있다.

제3절 서비스 제공 모델 비교 분석

1 Overview

Grid 기술은 웹 서비스 기술에 기반을 두고 있다. Grid 컴퓨팅의 사실상 표준인 Globus의 Grid 툴킷인 GT (Globus Toolkit)는 GT2에서 최근 GT4로 진화하면서 Web Service 기술을 핵심 서비스 모델로 설정하였다. Web Service 기술은 크게 SOAP, WSDL, UDDI의 핵심적인 스펙을 정의하는데, WSDL (Web Service Description Language)은 개개의 서비스를 표현하는 방식이고, SOAP (Simple Object Access Protocol) 은 분산 컴퓨팅에서 원격의 객체를 접근하는 방식을 뜻하며, UDDI (Universal Description, Discovery and Integration) 은 서비스를 찾기 위한 index 서비스의 일종이다.

Web Service 기술의 비약적인 발전은 WS-* 로 이름지어진 웹 서비스 관련 표준 스펙들을 통해 짐작할 수 있다. 메시지 스펙에 관련된 WS-Notification, WS-Addressing, 웹 서비스의 보안을 강화하기 위한 WS-Security, WS-SecureConversation, WS-Trust, 웹 서비스의 state를 관리하고 정의하기 위한 WSRF (Web Services Resource Framework) 관련 스펙 및 WS-Transfer 등 이 외에

도 수많은 WS-* 스펙이 제정되었다. Globus 진영도 이에 가세하여 최근 WSRF 및 WSN (WS-Notification),, WS-SecureConversation 등을 GT4에 적용시켜 Grid 서비스를 웹 서비스 기술의 스펙에 맞게 정의하고 구현하였다. Web Service 기술을 활용한 GT4의 구현은 표준 스택을 통하여 이종의 기기를 다루는 그리드 환경에서 상호연동성 (Interoperability)을 극대화시켜주는 장점을 지니고 있다. 이 외에도 그리드 기술은 X.509 PKI (Public Key Infrastructure) 기반의 GSI (Grid Security Infrastructure)를 통하여 그리드 자원에 대하여 강화된 보안 매커니즘을 제공한다.

웹2.0은 뚜렷한 기술적 정의는 없지만 웹을 플랫폼으로 사용하려는 것에 공통의 목적이 있다. 따라서, 웹에서 제공되는 다양한 서비스 - Google map, Amazon S3, Amazon EC2, 등 - 을 이용하여 새로운 응용과 서비스를 창출할 수 있게 하는 다양한 기존 기술들이 웹2.0이라는 이름으로 활용되고 있다. 기존에 이미 사용중인 기술을 활용하기 때문에 정해진 표준이 없이 필요에 따라 적절한 기술을 활용하여 다양한 서비스 제공자들이 제공하는 서비스를 사용할 수 있다는 것에 큰 특징이 있다. 특히 서비스 제공 모델에 있어서 웹2.0의 기술들은 효율성과 편의성을 공통의 지향점으로 삼고 있다. REST는 resource를 접근하는데 있어서 용이하면서도 표준화된 인터페이스를 제공하려는 아키텍쳐 스타일이고, RSS나 ATOM은 원하는 정보를 시시각각으로 업데이트하여 새로운 정보를 쉽게 획득할 수 있게 한다. 또한 JSON 등은 효율적인 데이터 전송을 제공해 주고, Wiki나 Folksonomies 등은 협업 및 커뮤니티를 손쉽게 구성하도록 도와주며 정보를 공유하는 데 있어서 큰 장점을 가지게 된다. 특히, 이들 기술들과 기존에 제공되고 있는 서비스들을 통합하여 Mashup이라는 형태의 새로운 응용을 만들 수 있는 프레임웍이 구축되어 새로운 응용과 서비스의 개발이 용이하다. 이와 같이 기술적으로는 새로운 것이 별반 있지 않지만, 이 기술과 서비스들을 적절히 활용한다면 Grid의 새로운 모델이 될 수 있을 것이다.

표3는 현재 사용되고 있는 Grid와 웹2.0 의 기술을 종합한 표이다. Grid와 Web Service 진영에서 사용하는 대부분의 기술이 WS-* 인 것에서 보이듯이 대부분이 표준화된 스펙을 사용하기 때문에 상호연동성에 있어서 뛰어난 장점을 보인다. 또한 정의된 다양한 표준은 서비스 설계시에 유연성을 더해줄 수 있으며, 이미 성숙한 기술이라는 장점도 있다. 반면에 웹2.0의 접근방법은 다양한 사용되는 기술도 다양하고 기존의 기술을 응용한 것임을 볼 수 있다. 특히 아직 Grid나 Web Service에서 제공하는 고급 기능에 부족한 면을 가지고 있지만 이들을 보완한다면 Grid를 통하여 새로운 응용 서비스를 생산하고, Grid 서비스의 활용도 또한 높일

	Grid / Web Service	Web 2.0 접근방법
Core Service Model	XML, WSDL, SOAP	XML JSON, RSS, ATOM REST with OpenAPI
Notification	WS-Notification, WS-Eventing	명시적인 기술 없음
Service Internet	WS-Addressing WS-MessageDelivery WSRM and MOTM	명시적인 기술은 없으나 TCP/IP 의 기능을 활용
Service Discovery	UDDI WS-Discovery	웹페이지등을 통한 접근
Management and State	WSRF, WS-Transfer	REST
Workflow and Transactions	BPEL WS-Choreography WS-Coordination	Mashup이나 MapReduce, JavaScript등을 이용할 수 있음
Security	WS-Security, WS-Trust, WS-Federation, SAML, WS-SecureConversation	SSL HTTP Authentication/Authorization OpenID
Portals and User Interface	WSRP, JSR-168 Portlets	Start Pages AJAX and Widgets Gagdets

표 3 Grid/Web Service와 웹2.0 접근방법 비교[15]

수 있을 것이다. 다음 장에서는 이들 기술들에 대한 자세한 논의를 한다.

2 Core service model

일반적으로 현재 분산 컴퓨팅 분야에서 크게 대두된 아키텍쳐는 ROA (Resource-Oriented Architecture)와 SOA (Service-Oriented Architecture) 이다. 두 아키텍처가 각각 REST (REpresentational State Transfer) 와 SOAP을 기반으로 하기 때문에 웹2.0 기술과 Grid 기술을 논의 할 때 이 두 아키텍처가 중요한 요소가 된다.

SOA

SOA는 앞에서 언급한 것처럼 SOAP을 기반으로 XML, WSDL 등을 활용하여 분산컴퓨팅 환경에서 개개의 서비스를 중심으로 접근하는 아키텍처이다. 각 작업 단위의 개별 서비스는 다양한 operation을 제공하며, Client는 SOAP과 XML, WSDL을 통하여 각 operation을 호출할 수 있게 된다.

ROA

ROA는 SOA와는 달리 서비스가 중심이 아닌 resource를 중심으로 수행이 되는

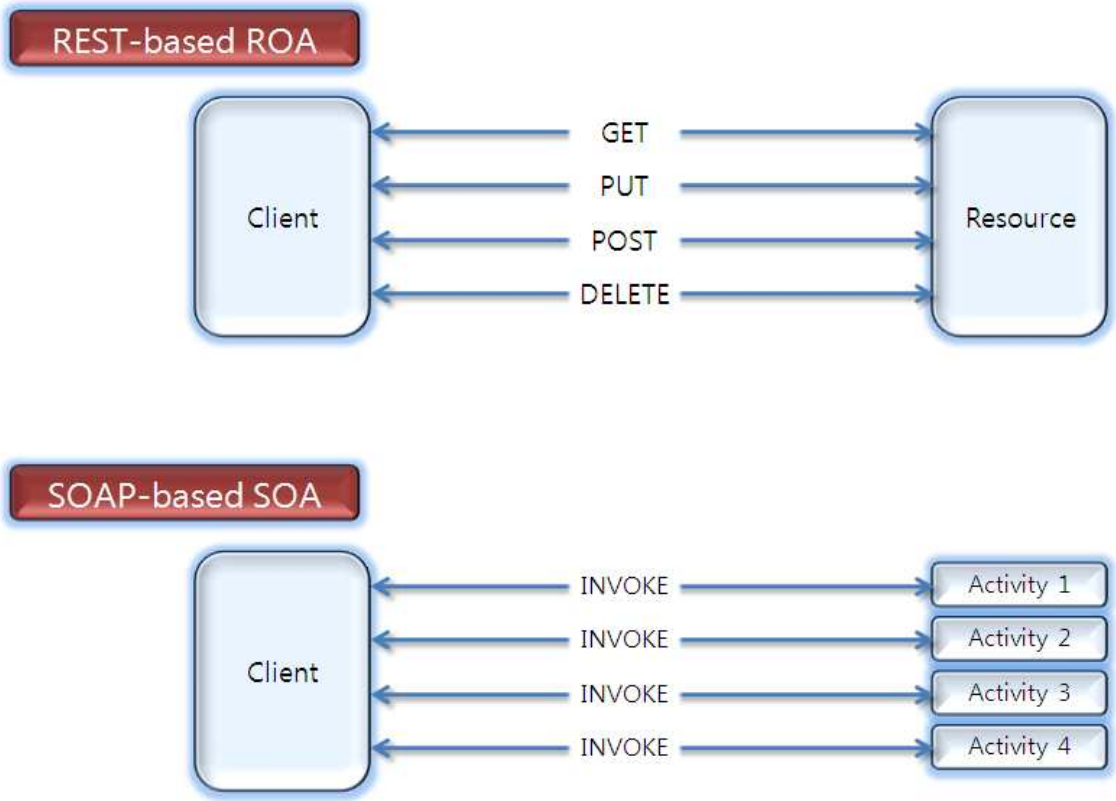


그림 5 ROA와 SOA 비교

아키텍쳐이다. 각 서비스가 정의하는 리소스가 있으며, 이에 대한 접근은 GET (resource를 다운로드하는 operation), PUT 혹은 POST (resource를 업로드하는 operation), DELETE (resource를 지우는 operation) 등의 소수의 표준화된 operation을 통하여 이루어진다.

REST or SOAP?

SOA와 ROA 이 두 아키텍처의 차이는 REST와 SOAP의 차이로 볼 수 있다. SOAP이 다양한 operation을 제공하는 객체에 대한 접근방법이라면, REST는 state를 가지고 있는 resource에 대한 표준화된 접근방식이다. 그림5에서 보이는 것처럼 두 아키텍처는 resource와 activity 두 가지 중에서 어떤 것을 중요시 하느냐에 따라 구분된다. 현재 제공되고 있는 많은 OpenAPI들이 REST 인터페이스를 제공하지만, SOAP 인터페이스 또한 많이 사용되고 있다. 이는 각 서비스가 지닌 특징에 따라 달라지게 된다. 즉, 제공하고자 하는 응용프로그램이 정보를 가지고 있는 resource에 대한 접근가능성을 주요 대상으로 한다면 (예를 들어 Amazon의 S3) ROA 방식의 REST 스타일로 설계하는 것이 용이하다. 이와는 반대로, 응용프로그램이 resource에 대한 operation을 중요시 한다면 SOAP을 사용한 SOA 방식의 설

계가 적당하다.

SOAP은 복잡한 구조와 메시지의 비효율성, 그리고 복잡한 WSDL로 인해 무겁다는 평가를 받고 있지만 신뢰도, 보안, resource 관리 등 고급 기능을 제공하는 WS-* 기술들은 SOAP 기반의 서비스에 다양한 기능을 제공해준다. 그렇게 때문에 Grid에서는 SOAP기반으로 다수의 WS-* 관련 기술들을 적용하고 있다. 이에 반해 REST 기반 서비스는 SOAP기반에서 사용할 수 있는 WS-*의 고급 기능을 사용하지 못하는 단점이 있지만 효율적이고 표준화된 인터페이스를 제공하여 resource 중심의 서비스에 최적화되어 있다. 이에 Google과 Amazon등 산업체에서 각광받고 대중적으로 사용되고 있다.

JSON

JSON은 Ajax 애플리케이션을 위한 유용한 포맷으로서 JavaScript 객체와 스트링 값 사이를 빠르게 변환할 수 있게 해준다. XML은 똑같은 텍스트 효과를 제공하지만, JavaScript와 XML 사이의 전환을 위해 적지 않은 오버헤드를 감수해야 한다. JSON의 가장 큰 가치는 JavaScript가 데이터-포맷 언어로서가 아닌 JavaScript로서 역할을 할 수 있다는 점이다. JSON 문자열을 별다른 파싱할 필요없이 바로 “eval" 에 적용하게 되면, 리턴값이 바로 JavaScript 객체가 되어버리기 때문에 편하고 빠르다.

JSON은 두 개의 구조를 가지고 있다. 첫 번째는 (이름, 값) 쌍의 집합으로서 다른 프로그래밍 언어에서의 Dictionary나 Hash table 과 같은 자료구조에 해당한다. 두 번째는 순서화된 리스트로서 array, vector, 혹은 sequence 와 같은 자료구조에 해당한다. 최근에는 보안이나 REST 등의 이슈와 연관지어 좀 더 확장된 환경에서 JSON을 다루려는 시도가 이루어지고 있는데[14] JSONP, JSONPath, JSPON, JSONT 등의 기술이 연구되고 있다.

RSS/ATOM

RSS(Really Simple Syndication)은 XML 기반의 문서 표준으로 뉴스나 블로그 사이트에서 주로 사용하는 콘텐츠를 표현방식이다. RSS가 등장하기 이전에는 원하는 정보를 얻기 위해 해당 사이트를 직접 방문하여야 했으나, RSS 관련 프로그램(혹은 서비스)을 이용하여 자동 수집이 가능해졌기 때문에 사용자는 각각의 사이트 방문 없이 최신 정보들만 골라 한 자리에서 볼 수 있다.

ATOM은 RSS의 대안으로서 제시된 포맷이다. 둘의 차이점이 몇가지 있는데, 첫

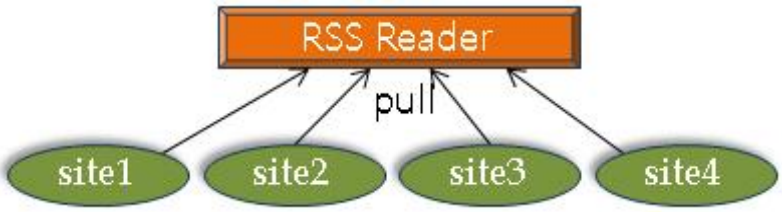


그림 6 RSS 기동 방식

번째는 컨텐츠 모델에 관한 내용이다. RSS의 경우 Payload 부분에 일반텍스트나 escaped HTML를 포함할 수 있는데, 둘중 어느 포맷으로 담겨있는지를 표시할수 있는 방법이 없다. 반면 ATOM은 명확하게 담고있는 컨텐츠의 타입이 어떤 것인지를 레이블을 달아둘수 있으며 위 두 타입을 포함하여 XHTML, XML, BASE64 바이너리 인코딩, 문서나 외부 멀티미디어 컨텐츠로의 참조 등 매우 다양한 Payload를 포함할 수 있다.

두 번째 차이점은 Modularity 이다. RSS에서 사용되는 element들은 일반적으로 다른 XML에서 재사용될 수 없다. 하지만 ATOM의 문법에서는 어떠한 element들도 ATOM feed의 문맥 밖에서 재사용될 수 있도록 디자인되어 있다. 이 밖에도 Date 포맷이라던지 국제화와 관련해서 RSS가 지원하지 못하는 것들을 ATOM이 지원할 수 있도록 확장된 규약을 가지고 있다.

3 Resource management

현존하는 대부분의 웹 서비스는 웹상에 존재하면서 state (데이터 값등을 지칭함)를 저장하고 있는 resource를 접근하는 방법을 제공한다. 하지만 위에서 언급한 것처럼 기존의 웹 서비스는 SOA 구조를 지니고 activity를 기준으로 설계된 아키텍쳐이기 때문에, 서비스 인터페이스에 resource에 대한 명시적인 정의가 없었기 때문에 resource를 빈번히 사용하면서도 표준에 입각한 인터페이스가 지원되지 않았다. 따라서, resource를 검색하고, 해당 resource에 대한 상태를 추출하거나 설정하는 인터페이스의 정의가 필요하게 되었고, 이에 따라 resource에 대한 명세도 필요하게 되었다.

웹 서비스 표준 제정 관련 산업체 및 관련 기관에서는 이에 따라 웹 서비스에 resource의 개념을 도입하여 다양한 스펙을 제정하였다. 이는 주로 OASIS 진영과 MS, BEA 등의 산업체에서 주도하고 있는데 각각 WSRF와 WS-Transfer (후에 WS-RT로 대체) 라는 이름의 스펙을 제정하고 있다.

WSRF

WSRF는 OASIS 및 Globus 진영에서 제정한 스펙으로 다음 네가지의 세부적인 스펙을 포함하고 있다.

WSRF-ResourceProperties - resource의 state를 추출하고 설정하는 방법에 대한 스펙을 정의한다. GetResourceProperty, GetMultipleResourceProperties, SetResourceProperties, QueryResourceProperties 등의 operation을 구현한 웹 서비스는 위의 표준화된 접근 방법을 통해 resource에 대한 접근이 가능해진다

WSRF-ResourceLifetime - resource의 생성과 소멸을 할 수 있는 표준화된 operation을 정의한다

WSRF-ServiceGroup - 서비스와 resource를 통합할 수 있는 방법을 제공한다

WSRF-BaseFaults - WSRF관련 에러를 정의한다

WS-RT

MS를 포함한 산업체 진영에서 제정한 WS-Transfer 스펙은 WSRF와 동일한 목적으로 만들어졌으나, WSRF에 비해 resource를 세부적으로 접근할 수 있는 operation에 있어서 부족한 점을 가지고 있다. 이에 이들 진영에서 이를 극복하기 위하여 WS-ResourceTransfer라는 스펙을 새로 제정하여 WS-Transfer에서 제공하는 사용하기 쉬운 resource operation (GET, PUT, DELETE) 외에 WSRF의 fragment access 기능을 지원하여 WS-Transfer의 단점을 보완하고 있다.

이들 두가지 resource 관련 스펙은 공통적으로 resource에 대한 정의를 기술하고, 이를 접근할 수 있는 표준 인터페이스를 제정하는 것에 중점을 두고 있다. 웹2.0 기술에서 SOAP을 대체할 만한 아키텍처로 대두된 REST 역시 resource에 중심을 둔 방법이다.

위에서 언급했듯이 REST는 resource에 대해서 PUT, GET, DELETE 등의 가장 간단하면서도, 필수적인 operation을 표준으로 제공하여 서비스를 만들자는 아키텍쳐 스타일이다. Amazon S3에서의 REST는 이렇게 구현된다. S3에서는 파일하나가 resource를 나타내며 접근가능한 URI로 식별된다. 사용자가 resource를 저장하기 위해서 (S3의 문맥에서라면 File을 upload하기 위해서)는 PUT 혹은 POST operation을 사용하여 resource를 저장한다. 저장된 resource의 state를 추출하기 위해서는 (S3의 문맥에서라면 File을 download하기 위해서) GET operation을 호출하여 해당 resource를 가져오게 된다(이 외에도, S3에서는 각 파일에 대한 access control 또한 resource로 정의하여 access control을 수정하거나 추출하는 데에 PUT, GET 등의 operation을 사용하게 된다.). 이는 WSRF나 WS-RT에서 제공하고

자 하는 resource에 대한 표준 인터페이스를 제공하는 것이고, 이를 Grid서비스에도 적용할 수 있을 것이다. 하지만, SOA기반으로 이루어진 그리드 서비스들을 제한된 operation을 사용하는 REST 방식으로 제공하는 것이 수월하지는 않다. 다행히, 그리드 서비스들의 operation수가 그렇게 많지 않기 때문에 이는 Grid 서비스의 RESTful 서비스로의 변화에 대한 가능성을 열어 준다. 예를 들어, WS-GRAM의 주요 서비스인 ManagedJobExecution의 경우, ManagedJobExecutionFactoryService의 createJob이 거의 유일한 operation이다. 사용자는 이후에는 별다른 operation을 사용하지 않고 Notification 서비스를 이용하여 resource의 state에 대한 것만 처리하게 된다. ManagedJobExecution을 REST 방식으로 해석한다면, job은 resource가 될 것이고, PUT 혹은 POST는 job submission, GET은 job의 상태를 얻어오는 operation, DELETE는 job을 소멸시키는 operation으로 볼 수 있을 것이다.

4 Notification

Grid에서 Notification 스펙은 중요한 위치를 가진다. 가장 흔한 예제로 Job submission을 들 수 있다. Grid 자원을 필요로 하는 사용자가 Job 스펙을 만들어서 Grid 서비스로 넘기면, Grid 서비스는 해당 job에 관련된 resource를 만들고, job을 실행하게 된다. 사용자나, 혹은 job의 state를 시시각각으로 모니터링할 필요성이 있을 경우에는 모니터링 시스템은 주기적으로 resource의 state를 검사할 필요가 있으나 이를 publish/subscribe 시스템을 활용하여 push방식을 쓰게 되면 resource의 변화를 효율적으로 알 수 있게 된다. 이 스펙은 resource의 state에 관련되어 있기 때문에, WSRF와 물려서 스펙이 만들어졌는데, WSRF와 마찬가지로, Globus 진영에서는 WS-Notification이라는 이름으로, 그리고 BEA 등 산업체 진영에서는 WS-Eventing이라는 이름으로 스펙을 각각 제정하였다.

WS-Notification에는 WS-BaseNotification, WS-Topics, WS-BrokeredNotification 세 가지의 spec이 정의되어 있으며 WS-BaseNotification과 WS-Topics를 통하여 Web Service 상에 손쉽게 publish/subscribe 시스템을 구축할 수 있다. WS-Eventing은 WS-BaseNotification과 동일한 기능을 정의한 스펙이고, WSRF나 WS-RT처럼 WS-Notification과 WS-Eventing을 통합하려는 시도가 이루어지고 있다.

WS-Notification에서는 subscriber에 Web Service 컨테이너를 띄우는 방식으로 Event를 처리한다. 그림7에서 보이는 것처럼 subscriber가 처음 WS-Topics를 사용하여 관심있는 resource의 상태 조건을 정의하고, 이벤트를 처리할 서비스 컨테이

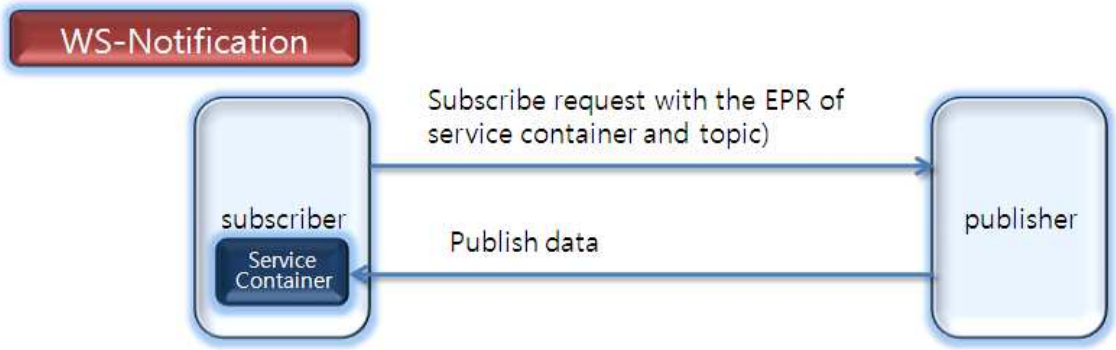


그림 7 WS-BaseNotification 구현

너를 실행한 후에 해당 서비스의 EPR (Endpoint Reference)를 토픽과 함께 publisher에게 보낸다. publisher는 해당 토픽과 EPR을 등록하고 이벤트가 발생할 때마다 EPR에 메시지를 전송한다.

반면에, 웹2.0에는 notification를 지원하는 기술이 아직 없다. 가능한 방법으로는 자바스크립트의 XmlHttpRequest를 활용하여 주기적인 polling을 하는 방법이 있다. 혹은 RSS나 ATOM을 사용하여 유사하게 구현할 수 있다. 하지만, 이 방법 모두 관심있어하는 이벤트의 조건이나 명확한 이벤트 전달 방법이 정의되지 않았기 때문에 그대로 적용하기에는 무리가 있다.

5 Security

Grid나 웹2.0을 활용한 Cloud Computing 등 내부 자원을 외부 서비스로 제공하는 데에는 보안이 가장 중요한 문제이다. Grid에서는 이를 PKI와 X.509 인증서를 기반으로 한 GSI를 사용하고 있다. 사용자와 Grid 서비스 모두 신뢰할 수 있는 CA로부터 인증서를 발급받았을 경우에 서로를 인증하는 Mutual Authentication, 사용자 대신에 사용자임을 증명할 수 있는 Proxy Certificate 및 Delegation 등 다양한 보안 기술들이 제공된다. 여기에 WS-SecureConversation을 사용하여 메시지의 무결성 (Integrity)이나 기밀성 (Confidentiality)을 보장할 수 있게 된다. 실제 GSI에서는 사용자는 모르게 AuthenticationService가 인증 및 권한검사를 담당하고 있다. A라는 서비스로 요청을 보낼 때 사용자 모르게 이 요청은 A-authService라는 서비스로 redirect되게 되는데, 이는 다시 서버쪽에서 AuthenticationService로 redirect되어 실제 초반 인증과 권한을 검사하고 정상적으로 완료되어 Secure Context가 맺어졌을 경우에 선택된 옵션에 따라서 메시지를 전자서명(무결성 보장) 혹은 암호화(기밀성 보장)하여 다시 원래의 A라는 서비스로 요청이 전송하게 된다.

반면에 보안 관련 웹2.0 기술은 아직 널리 쓰이고 있지 않다. GSI와 같은 PKI, X.509 기반의 보안 기술은 아직 쓰이지 않고 있으며, Amazon의 S3에서 사용하는 대칭키 기반의 인증이나 사용자/패스워드 기반의 Single Sign On을 위한 OpenID가 주로 쓰이고 있다.

Amazon S3의 인증구조

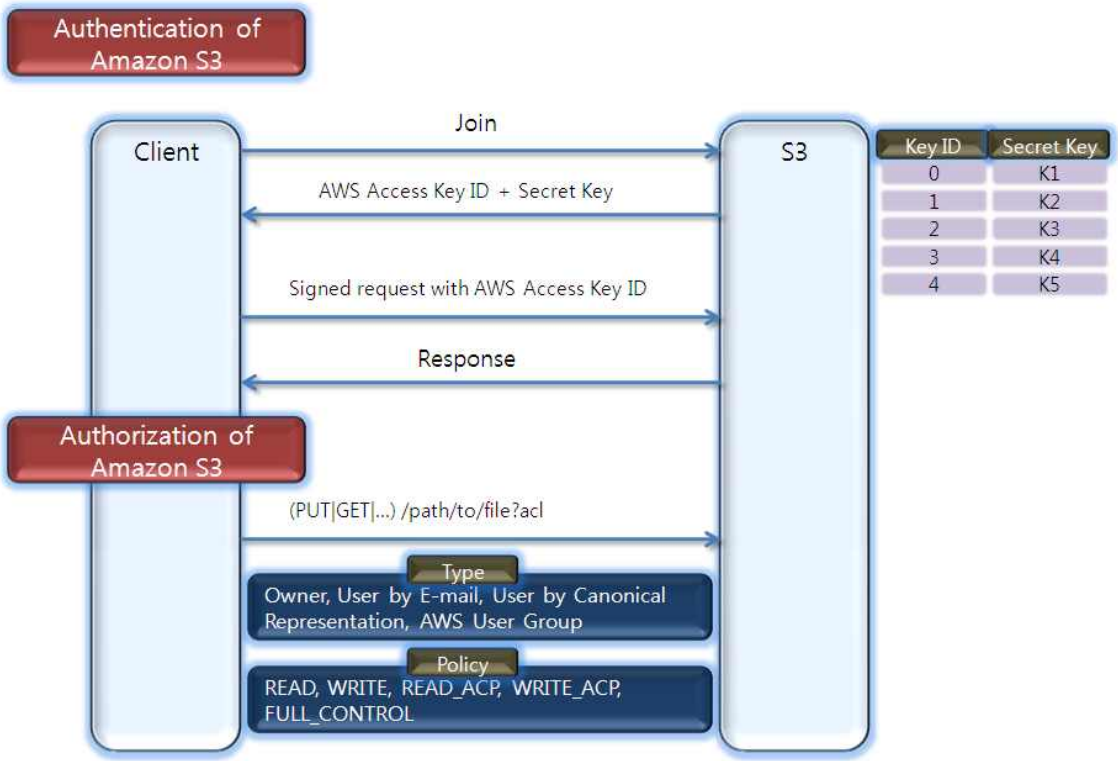


그림 8 Amazon S3의 보안

Amazon S3는 대칭키를 사용한 전자서명을 기반으로 사용자를 인증한다. 그림8에서 보이는 것처럼, S3 서비스를 사용하기 위해서 각 사용자는 자신 고유의 비밀키 KS를 발급받는다. S3에서는 비밀키 KS외에 비밀키를 접근할 수 있는 키ID를 사용자에게 보내준다. 사용자 측에서는 S3에 파일을 업로드하거나 다운로드하기 위해서 PUT, GET 등의 operation을 사용하는데, HTTP의 구조상 stateless connection이기 때문에 모든 요청에 자신의 비밀키를 사용하여 사인하고 생성된 전자서명은 동봉하여 S3에게 전송한다. HTTP의 stateless 특성으로 인하여, Client와 S3가 키를 공유하기 위해서는 Client측에서 자신의 비밀키 KS의 키ID를 전자서명과 함께 보내야 한다. 이때, S3는 받은 요청이 어떤 사용자에게서 온 것이지 인증하기 위하여, 전송된 키ID를 가지고 키테이블에서 해당 사용자의 비밀키 KS를 얻어낸 후에 보내온 전자서명과 일치하는지 확인하는 과정을 통해 사용자를 인증

한다.

OpenID

OpenID는 웹 사이트마다 빈번히 입력하는 ID/Password를 한 번 입력하여 로그인 함으로써 여러 사이트를 동시에 로그인하는 Single-Sign-On을 가능케 하는 기술이다. OpenID는 현재 매우 각광받고 있는 기술로서, AOL, Microsoft, Sun, Novell 등 약 만여개 이상의 사이트가 OpenID를 지원하고 있으며 다수의 OpenID 서버가

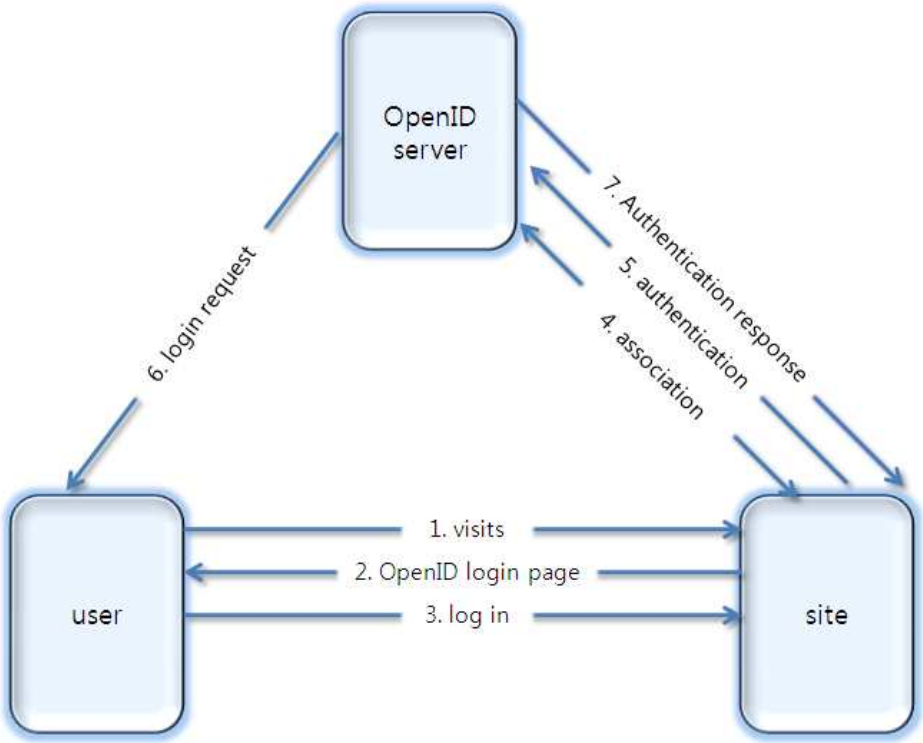


그림 9 OpenID의 흐름도

서비스를 하고 있다.

그림9은 OpenID의 흐름도를 보여주고 있다. 우선 각 사용자는 OpenID Server에 자신의 ID를 생성해야 한다. 사용자의 ID는 URI로 구성되어 있으며 이는 각 사용자를 뜻하기도 하지만, 해당 사용자가 사용중인 OpenID 서버를 가리키기도 한다. 사용자가 OpenID를 지원하는 사이트 A에 로긴을 하려고 하면, 사이트 A는 사용자에게 OpenID 로그인 화면을 보여준다. 사용자가 자신의 OpenID와 비밀번호를 입력하면 사이트 A는 사용자가 입력한 OpenID를 보고 해당 사용자의 OpenID 서버를 알 수 있게 되어, OpenID 서버와 연관 과정을 거친후 사용자가 입력한 인증정보를 전송하게 된다. 이 때, OpenID 서버는 보안을 위하여 사용자에게 실제로 사용자가 사이트 A를 신뢰할 것인지 확인한 후에 로그인 정보를 처리하게 되고,

사용자는 OpenID 서버로 로그인을 하게 된다. 최종적으로 OpenID 서버가 해당 인증 정보를 사이트 A에게 전송하게 되면 로그인 과정이 끝나게 된다. 추후에 OpenID를 지원하는 또다른 사이트 B에 사용자가 로그인을 하게 되면 사이트 B는 이미 로그인정보가 저장되어 있는 OpenID 서버를 통하여 사용자를 인증하게 된다.

제4장 기대효과

웹2.0 기반기술을 활용하여 그리드 시스템을 구축할 경우, 사용자에게 친숙한 과학기술 커뮤니티 기반 플랫폼으로서 자리매김할 수 있고, 또한 기존 그리드 서비스의 품질을 향상시킬 수 있다. 웹2.0을 활용한 과학기술 인프라의 구축은 그리드 자원을 개방할 수 있는 가능성을 열어놓음으로써, 그리드 자원의 효율적인 활용을 제고하고 이에 기반한 과학기술 분야의 고급 응용 어플리케이션의 개발에 기여할 수 있다. 또한 OpenAPI 등을 이용하여 그리드 시스템의 개방을 유도함으로써 그리드 시스템의 활용 영역을 과학기술 분야뿐만 아니라 전반적인 정보기술 분야로 확대할 수 있을 것이고, 이를 통해 그리드 시스템을 활용한 다양한 응용 프로그램을 생산할 수 있는 기반기술을 제공할 수 있다.

웹2.0 기술을 활용한 그리드 시스템 구축은 현재 국내, 국제적으로 아직 연구된 바가 거의 없으며, 따라서 해당 기술을 표준화할 경우, 본 연구의 성과를 통하여 국제 표준화에 앞장설 수 있으며, 또한 확보된 기술의 선도성을 바탕으로 국익에 우선하는 방향으로 표준화 과정을 조율할 수 있을 것으로 기대한다.

참고문헌

[1] "An EGEE Comparative Study: Grids and Clouds - Evolution or Revolution?" - EGEE-Grid-Cloud-v1.1.doc, 03/05/2008.

[2] “웹2.0 기술 현황 및 전망”, 전종홍, 이승윤, 전자통신동향분석 제21권 제5호 2006년 10월.

[3] “Using 웹2.0 for Scientific Applications and Scientific Communities", Marlon E. Pierce, Geoffrey C. Fox, Jong Y. Choi, Zhenhua Guo, Xiaoming Gao, and Yu Ma

[4] “RESTful Web Services vs. "Big" Web Services: Making the Right Architectural Decision", Cesare Pautasso, Olaf Zimmermann, Frank Leymann

[5] "WS-* vs. REST: Mashing up the Truth from Facts, Myths and Lies", Sanjiva Weerawarana, QCon2007.

[6] “Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities", Rajkumar Buyya, Chee Shin Yeo, and Srikumar Venugopal, HPCC2007.

[7] “Aneka: Next-Generation Enterprise Grid Platform for e-Science and e-Business Applications", Xingchen Chu, Nadminti K., Chao Jin, Venugopal S., Buyya R., IEEE International Conference on e-Science and Grid Computing, Dec. 2007.

[8] "http://www.gartner.com/it/sym/2008/spg10/spg10.jsp", 가트

너 보고서.

[9] "http://developer.amazonwebservices.com", 아마존 웹서비스 API 문서.

[10]“http://www.lynchconsulting.com.au/blog/index.cfm/2008/3/13/CFMX--SOAP -vs-REST-benchmarks”, REST와 SOAP의 성능비교(비공식 벤치마크).

[11] "http://code.google.com/appengine", Google App Engine.

[12] "http://en.wikipedia.org/wiki/AJAX", AJAX 위키피디아.

[13] “http://www.widgets-gadgets.com/2007/08/what-is-web-widget.html”, 위젯 설명.

[14] "http://www.json.com/2007/09/24/advanced-json

-presentation" , Advanced JSON.

[15] Grids challenged by a 웹2.0 and Multicore Sandwich, Geoffrey Fox, Marlon Pierce.