실증환경테스트베드 개발

일 자: 2014년 11월 30일

부 서: 첨단연구망센터/첨단연구망개발팀

제출자: 권 윤 주

KiSTi
한국과학기술정보연구원
Korea Institute of Science and Technology Information
305-806 대전광역시 유성구 어은동 52번지
TEL (042)869-0676 / FAX (042)869-0679
www.kisti.re.kr

실증환경테스트베드 개발

작성자: 권윤주

첨단연구망개발팀, 한국과학기술정보연구원

yulli@kisti.re.kr

최종수정일: 2014년 11월 30일

Abstract

네트워크 증속과 더불어 첨단 응용 연구분야에서는 네트워크기반의 도전적인 협업 연구들을 시도해왔다. 이를 위해 KREONET에서는 실질적인 방법으로서 고성능 시스템을 백본에 연결하여 응용 연구자들이 이용할 수 있는 환경을 구축해준다거나 필요에 따라 물리적으로 기가 수준의 네트워크를 응용 연구자의 연구실까지 연결시켜주는 등의 방법으로 첨단 응용 연구분야의 협업 연구를 지원하였다. 그러나 전자의 경우에는 응용 연구자가 원거리 시스템을 사용하는 데 있어서 시스템 사용 및 관리의 제한이 있었고, 후자의 경우에는 제한된 몇몇 연구에만 한시적으로 적용될 수 있는 데 그 한계가 있었다. 갈수록 다양한 응용 분야에서 네트워크 기반의 협업 연구를 진행해 나가는데 좀 더 유연하게 네트워크 자원을 제공해주기 위해서 KREONET에서는 네트워크를 포함한 실험환경을 제공할 수 있는 기반이 필요하다고 판단하고 있다. 이에 KREONET에서는 KREONET 백본에 연동되어 있는 시스템, 저장 장치등의 물리적 자원들을 사용자의 요구에 따라 할당하여 사용자가 가상 연구 환경을 제공받을 수 있는 실증환경테스트베드(RealLab)를 구축하고자 한다. 본 문서에서는 첨단 응용 분야에서의 네트워크를 통한 다양한 시도들을 지원하고자 구축하고 있는 실증환경테스트베드의 요구사항을 명세하고 시스템 설계 및 개발 내용을 기술하고자 한다.

Topics

1. 개요

(1) 목표

실증환경테스트베드는 첨단 응용연구 그룹들의 첨단 응용 연구를 지원하기 위한 “동적 컴퓨팅/네트워킹/스토리지 자원 제공 서비스”이다. 첨단 응용 그룹은 점점 더 정밀하고 유의미한 결과들을 창출하기 위해서 더 많은 데이터를 좀 더 빠른 시간 안에 공유하여 실험을 수행하고자 한다. 이러한 활동들을 지원하기 위해 연구망들이 존재하지만, 좀 더 네트워크라는 자원을 효율적으로 사용하도록 지원하기 위해서는 네트워크를 포함한 실험환경까지 제공할 수 있는 방법이 필요하다. 이에 본 문서에서는 첨단 응용 그룹 사용자들에게 동적으로 사용자 실험환경을 제공하고자하는 실증환경테스트베드에 대한 요구사항 명세 및 요구사항에 따른 시스템 설계를 기술하고자 한다.

(2) 실증환경테스트베드 서비스 시나리오

사용자는 실증환경테스트베드를 통해서 다음과 같은 시나리오로 자원을 제공받을 수 있다. 사용자는 계정을 생성하고 실험 환경을 구성하는 일련의 절차를 거친다. 이 절차내에서는 사용자가 필요로 하는 하드웨어적 시스템 자원 정의, 운영체제등의 소프트웨어적 정의를 마치면 실증환경테스트베드는 유휴한 시스템들을 검색하여 사용자들의 설정 요구사항들을 반영하여 시스템들을 할당해준다. 이후 사용자로부터 실험 중단 요청이 들어올 경우, 사용하고 있는 시스템의 사용자커널을 백업하여 추후 재사용할 수 있도록 저장하고 실험시스템의 전원을 종료한 후 서버 풀로 시스템을 반환한다. 또 사용자로부터 실험 종료요청이 들어올 경우, 해당 실험데이터들을 삭제하고 시스템의 전원을 종료한 후 서버 풀로 시스템을 반환한다.
그리고 실증환경테스트베드는 첨단 응용 그룹을 대상으로 하고 있으므로, 컴퓨팅자원 외에 저장공간을 할당받을 수도 있다. 할당받은 저장공간에는 사용자가 필요로 하는 대용량의 실험데이터를 저장할 수도 있고, 때에 따라 저장공간에 있는 데이터를 사용자 시스템으로 이동시킬 수도 있다.

2. 요구 사항

(1) 기능 요구사항

실증환경테스트베드가 필요로 하는 기능들에 대해서 기술한다.

가. 사용자 관리 기능

실증환경테스트베드에서는 사용자는 자원을 제공받는 대상이다. 이에 따라 실증환경테스트베드의 기능에는 자원을 요청하고 할당받을 수 있는 사용자에 대한 고려가 필요하다.

번호	1.1	이름	사용자 계정 추가
목적	테스트베드를 이용하는 사용자 계정 추가		
출력내용	없음		
내용			
• 본 테스트베드를 사용하기 위해서는 사용자마다 시스템에서 사용할 수 있는 계정을 필요로 한다. 따라서 새로운 사용자를 수용하기 위하여 사용자 계정 추가 기능 필요 • 사용자 계정 추가시 ID/PW 만 넣어서 생성한다.			

번호	1.2	이름	사용자 계정 삭제
목적	테스트베드를 이용하는 사용자 계정 삭제		
출력내용	없음		
내용			
• 사용자의 요청에 의하여 본 테스트베드에서 사용하는 사용자 계정은 삭제할 수 있다.			

나. 실험 관리 기능

실증환경테스트베드는 사용자그룹(한 명 이상의 사용자)의 실험을 지원하기 위하여 자원을 제공한다. 이에 실증환경테스트베드의 기능에는 사용자 그룹의 요구사항대로 실험 환경을 구성하고 관리할 수 있는 기능이 고려되어야 한다.

번호	2.1	이름	실험 생성
목적	사용자가 필요로 하는 실험 환경 구성		
출력내용	할당된 시스템의 IP 또는 도메인 네임		
내용			
• 사용자의 실험을 수행할 목적으로 시스템은 할당되어야 한다. • 이때, 가상화 기술 사용 여부에 상관없이 한 시스템은 한 실험에 할당되어야 한다. • 모든 시스템이 할당되었을 때 더 이상 할당할 수 있는 시스템이 없음을 메일 또는 웹을 통해 통지하여야 한다. • 사용자가 요구한 운영체제로 부팅되어 사용자(또는 실험)에게 할당하여야 한다.			

번호	2.2	이름	실험 중단
목적	장시간 실험이 진행되지 않을 때 실험은 반환하면서 실험내용을 유지할 수 있도록		
출력내용	없음		
내용			
• 실험 종료가 아니기 때문에 실험 내용은 모두 백업되어야 한다. • 운영체제 스냅샷을 찍어 백업한다. • 사용자계정/실험계정에 있는 데이터 파일들은 손실되지 않도록 보관한다.			

번호	2.3	이름	실험재시작
목적	중단되었던 실험을 사용자가 재시작하고자 할 때 실험 환경 구성		
출력내용	재할당된 시스템 IP 또는 도메인네임		
내용			
• 사용자가 실험 생성할 때의 환경으로 시스템을 할당한다. • 시스템 할당 이후 운영체제는 실험중단시의 운영체제 스냅샷으로 불러온다.			

번호	2.4	이름	실험 종료
목적	사용자가 실험을 종료하고자 할 때		
출력내용	없음		
내용			
• 사용자가 할당받았던 시스템 반환 • 실험 디렉토리에 저장되어 있던 데이터 삭제			

다. 운영체제 관리 기능

실증환경테스트베드에서는 사용자가 필요로 하는 운영체제로 자신이 할당받은 시스템을 운영시킬 수 있어야 한다. 이에 따라 실증환경테스트베드에는 다양한 운영체제에 대한 추가, 삭제, 백업 등의 관리기능이 고려되어야 한다.

번호	3.1	이름	운영체제 추가
목적	테스트베드에서 사용할 수 있는 운영체제를 추가시켜주기 위함		
출력내용	없음		
내용			
• 테스트베드에서 사용할 수 있도록 새로운 운영체제를 저장 공간에 추가시켜준다. • 이 때 네이밍 규칙을 준수해서 추가되는 운영체제의 이름을 만든다. • 테스트베드에 운영체제를 추가하는 행위는 시스템 관리자 권한을 가진 사용자만 할 수 있다. • 미리 본 테스트베드에서 사용가능 테스트를 마친 운영체제만 추가될 수 있다.			

번호	3.2	이름	운영체제 삭제
목적	테스트베드에서 필요없는 운영체제를 삭제시켜주기 위함		
출력내용	없음		
내용			
• 테스트베드에서 필요없는 운영체제를 삭제시켜준다. • 필요없는 운영체제라 함은 “너무 오래된 버전” 이거나 “사용할 때마다 오류를 일으키는 운영체제”를 의미한다. • 삭제하기 전에 현재 구동중인 실험 또는 중단된 실험 삭제하고자 하는 운영체제가 사용되고 있는 지 확인 후 삭제 절차를 진행한다.			

번호	3.3	이름	운영체제 스냅샷
목적	실험 중단시에 사용중인 시스템 상태를 백업하기 위함		
출력내용	없음		
내용			
• 실험 중단시에 시스템 상태 백업 • 운영체제 스냅샷은 운영체제가 저장된 공간에 위치 • 운영체제 스냅샷을 저장할 때 네이밍 규칙 마련 필요			

라. 사용자 데이터 관리 기능

실증환경테스트베드의 사용자 그룹은 기본적으로 첨단 과학 응용을 염두에 두고 있는데, 이들 그룹의 경우 상당히 대용량의 실험 데이터를 이용하여 실험을 수행하여야 한다. 따라서 실험시 사용되는 컴퓨팅 자원은 할당/반환이 발생할 수 있는 데, 컴퓨팅 자원을 반납하게 되는 경우에라도 실험 데이터는 유지될 수 있는 방법이 필요하다. 이에 실증환경테스트베드에서는 컴퓨팅 자원과 분리된 저장 자원을 사용자가 할당받고 데이터 저장할 수 있는 사용자데이터 관리 기능이 고려되어야 한다.

번호	4.1	이름	볼륨 생성
목적	실험에 요구되는 실험데이터를 저장할 수 있는 공간 생성		
출력내용	생성된 볼륨 이름		
내용			
• 사용자가 데이터 저장을 위해 필요로 하는 볼륨을 사용자가 원하는 사이즈로 할당(그 크기는 실증환경테스트베드에서 정의하고 있는 쿼터는 초과할 수 없다)			

번호	4.2	이름	볼륨 삭제
목적	불필요한 저장 공간 삭제		
출력내용	없음		
내용			
• 사용자가 생성했던 볼륨에 대하여 할당 해제하는 행위			

번호	4.3	이름	데이터 업로드
목적	실험에 요구되는 실험데이터를 사용자 시스템에서 실증환경테스트베드 내 사용자 환경으로 이동을 위함		
출력내용	없음		
내용			
• 외부 데이터를 실증환경테스트베드의 사용자 계정 디렉토리로 업로드			

번호	4.4	이름	데이터 다운로드
목적	실증환경테스트베드 내 사용자 환경에 있는 실험데이터를 사용자 시스템으로 이동하기 위함		
출력내용	없음		
내용			
• 실증환경테스트베드 내의 사용자 데이터를 외부 머신으로 다운로드			

(2) 기타요구사항

가. 성능 요구 사항

- 신뢰성 : 모든 수행에 있어 정확하게 수행되어야 한다.
- 수행속도 : 모든 기능은 최소한의 수행시간을 가진다. 백업 이외의 작업은 반응속도 30초 이내에 이뤄져야 한다.

나. 인터페이스 요구사항

- 국제화 : 영문으로 작성된 메시지로 출력한다.
- 웹인터페이스 : 웹을 통하여 쉽게 관리될 수 있어야 한다.

다. 기타요구사항

- 보안성 : 시스템 관리 인터페이스는 최고 관리자만 접근할 수 있게 한다.
- 유지가능성 : 기능을 쉽게 추가할 수 있도록 표준화시켜 프로그래밍한다.
- 호환성 : 인터넷 익스플로러/파이어폭스/사파리/크롬 등의 애플리케이션에서 모두 구동가능해야 한다.

3. 관련 연구

앞장에서 설명하고 있는 실증환경테스트베드의 요구사항들은 사용자 관리, 시스템 할당 및 운영체제 관리적 측면에서 클라우드 서비스와 유사한 기능을 포함하고 있다. 그리고 대용량 데이터 전송을 위한 기능을 추가로 필요로 하고 있다. 이에 본 장에서는 실증환경테스트베드 설계에 사용될 클라우드 소프트웨어인 OpenStack과 데이터 전송을 위해 사용될 고성능 파일 전송 툴에 대해서 소개하고자 한다.

(1) 오픈 스택

가. 개요

오픈스택은 IaaS 형태의 클라우드 컴퓨팅 오픈소스 프로젝트이다. 2012년 창설된 비영리 단체인 OpenStack Foundation에서 개발 리딩 및 유지관리 하고 있으며 아파치 라이센스 하에 배포된다. AMD, Intel, Canonical, Suse Linux, Red Hat, Cisco Systems, Dell, HP, IBM, NEC, VMware, Yahoo!등의 150개 이상의 회사가 이 프로젝트에 참가하고 있으며, 주로 리눅스 기반으로 운용과 개발이 이루어진다. 컴퓨팅, 저장 공간, 네트워킹의 가용자원을 제어하는 목적의 여러 개의 하위 프로젝트로 이루어져 있다. 대시 보드 프로젝트는 다른 하위 프로젝트의 운영 제어를 웹 인터페이스를 통해 담당한다. 오픈스택은 오픈소스 프로젝트를 지향한다. 커뮤니티는 6개월의 릴리즈 사이클로 개발을 진행하고 있다. 매 사이클의 기획단계에서는 OpenStack Design Summit을 개최하여, 개발자 작업을 지원하고, 로드맵을 설정하고 있다.

나. 오픈 스택의 구성 요소

오픈스택을 Austin을 시작으로 해서 Bexar, Cactus, Diablo, Essex, Folsom, Grizzly, Havana, Icehouse, Juno까지 개발되어 왔다. 본 문서에서는 실증환경테스트베드에서 사용하고 있는 버전인 Grizzly를 기반으로 OpenStack 구성요소의 내용을 기술하기로 한다.

1) Compute (Nova)

Nova는 IaaS 시스템의 중심이 되는 구성 요소이다. 언어는 Python을 사용하며, 병렬 프로그래밍을 위한 Eventlet, AMQP 통신을 위한 Kombu, 데이터베이스 접근을 위한 SQL Alchemy와 같은 외부 라이브러리를 사용한다. Nova는 전용 하드웨어나 특별한 소프트웨어 요구 조건 없이 일반적인 하드웨어에서 사용

가능하며, 기존 시스템과 써드 파티 기술과 통합될 수 있도록 설계되어있다. 자동적으로 컴퓨터 자원의 풀을 가동하여 관리할 수 있으며, 베어메탈과 HPC (High-Performance Computing) 등의 여러 가상화 기술을 이용할 수 있도록 되어있다. KVM, XenServer, Hyper-V, 리눅스의 LXC 등의 하이퍼바이저를 사용할 수 있다. 오픈 스택의 ARM 위에서 여러 종류의 하이퍼바이저를 동작시킬 수 있다.

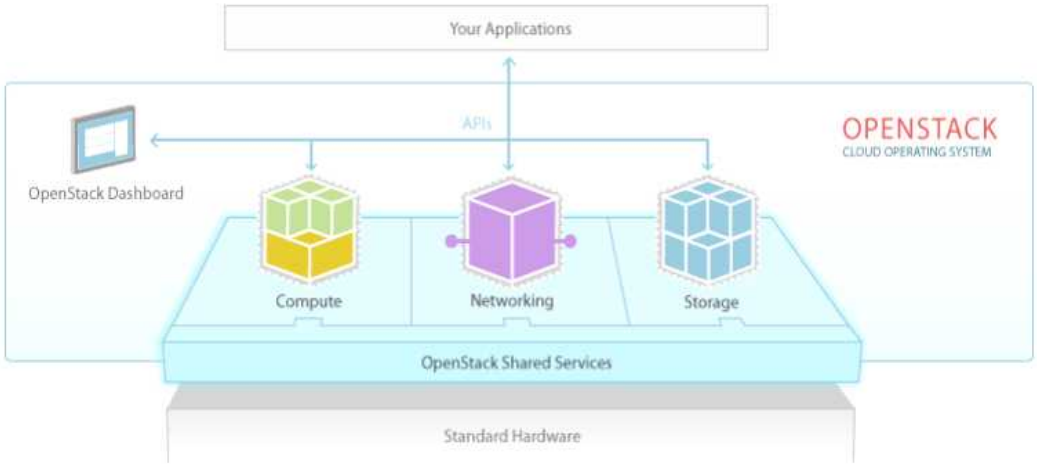


그림 2 Structure of OpenStack System

2) Object Storage (Swift)

Swift는 확장성과 용장성이 있는 오브젝트 스토리지 시스템이다. 오브젝트와 파일들은 데이터 센터를 중심으로 널리 퍼져있는 서버의 디스크 드라이브에 쓰여진다. 오픈 스택에서 제공하는 소프트웨어는 클러스터 간의 데이터 복제와 무결성을 보장한다. 스토리지 클러스터는 새로운 서버들을 추가하면서 확장된다. 만약 서버나 하드 드라이브에서 실패가 일어나면, 오픈 스택은 그 내용을 활성 중인 다른 노드로부터 클러스터 안의 새로운 곳으로 복제한다. 오픈 스택은 소프트웨어 로직을 사용하여 디바이스들로의 데이터 복제나 분배를 보장할 수 있기 때문에 비싸지 않은 일반적인 하드 드라이브와 서버들을 사용할 수 있다

3) Block Storage (Cinder)

블록 스토리지 시스템인 Cinder는 오픈 스택의 Compute 인스턴스가 블록 레벨 스토리지 디바이스를 안정적으로 이용할 수 있도록 해준다. 블록 스토리지 시스템은 블록 디바이스를 서버에 만들고, 마운트하고 마운트를 해제하는 것을 관

리한다. 블록 스토리지의 볼륨은 오픈 스택의 Compute로 완전히 통합되며, Dashboard를 통해 사용자가 필요에 따라 볼륨들을 관리할 수 있도록 해준다.

블록 스토리지는 데이터베이스 스토리지, 확장 가능한 파일 시스템, 로우 블록 레벨 스토리지에 접근할 수 있는 서버와 같이 성능에 민감한 시나리오에 적합하다.

4) Networking (Neutron or nova-network)

OpenStack에서는 nova-network가 전체적인 OpenStack의 네트워크를 담당하였다가 Folsom 배포판부터 네트워킹 컴포넌트가 nova에서 분리되어 Quantum 또는 Neutron이라는 이름으로 개발되고 있다. 그러나 Grizzly 배포판까지는 nova-network 또는 Neutron을 선택하여 OpenStack의 네트워크를 운영할 수 있다. 이러한 OpenStack의 네트워킹 컴포넌트는 네트워크와 IP 주소들을 관리하는 시스템이다.

이러한 주소 체계를 nova-network에서는 Flat Network와 VLAN Network를 이용하여 제공하였는데, Neutron으로 넘어가면서 프로젝트마다 프로젝트 내 인스턴스간 사용할 수 있는 폐쇄된 네트워크를 지원할 수 있는 기능도 포함하게 되었다. 이에 관리자들은 다수의 tenancy들을 지원 할 수 있는 OpenFlow 같은 Software-defined networking (SDN) 기술을 통합하여 이용할 수도 있다. 그러나 Neutron의 경우 운영상, 성능상의 문제로 Juno까지도 서비스 안정성이 떨어진다고 보고되고 있다.

5) Dashboard (Horizon)

Horizon은 관리자와 사용자가 클라우드 자원에 접근할 수 있는 사용자 그래픽 인터페이스를 제공한다. 이 인터페이스를 통해 추가의 관리 툴과 같은 써드 파티 프로덕트와 서비스를 제공받을 수 있다. 개발자들은 오픈 스택 API나 EC2 API를 이용하여 자동적으로 접속하고 자원을 관리하는 툴들을 만들 수 있다.

6) Identity Service (Keystone)

Keystone은 OpenStack의 사용자 인증 시스템으로써 LDAP와 같은 백엔드 디렉토리 서비스들과 통합될 수 있다. Keystone은 OAuth2.0 형태의 사용자 인증을 제공한다.

7) Image Service (Glance)

오픈 스택의 Glance는 디스크와 서버의 이미지를 탐색하고, 등록하고, 전송하는 기능을 제공한다. Glance는 오픈 스택 오브젝스 스토리지를 포함하여 디스크와 서버의 이미지를 다양한 백엔드에 저장할 수 있다. 저장된 이미지는 템플릿으로 사용될 수 있다. 그리고 개수에 제한받지 않고 백업을 저장하고 카탈로그할 수 있다. Glance의 이러한 API가 제공하는 표준 REST 인터페이스는 디스크 이미지에 대한 쿼리 정보를 포함하고 있어 클라이언트는 새로운 서버로 이미지를 스트리밍할 수 있다.

(2) 고속 파일전송 도구

가. UDT(UDP based Data Transfer Protocol)

1) UDT 개요

UDP 기반의 데이터 전송 프로토콜 (UDT)은 고속 광역 네트워크에서 대용량의 데이터 세트를 전송하기 위해 설계된 고성능 데이터 전송 프로토콜이다. 초기 버전은 초고속 네트워크 (1Gbps, 10Gbps)에서 개발되었지만, 최신 프로토콜 버전은 일반적인 인터넷 환경에서도 지원되도록 업데이트되었다. 예를 들면, 이 프로토콜은 랑데부 연결 설정을 지원하며, 이것은 UDP를 사용하여 NAT 방화벽을 통과하기에 좋다. UDT는 SourceForge에서 제공되는 오픈 소스이다. 고속 데이터 전송을 지원하는 가장 인기있는 솔루션 중 하나이며, 많은 연구 프로젝트와 시판품에 쓰이고 있다.

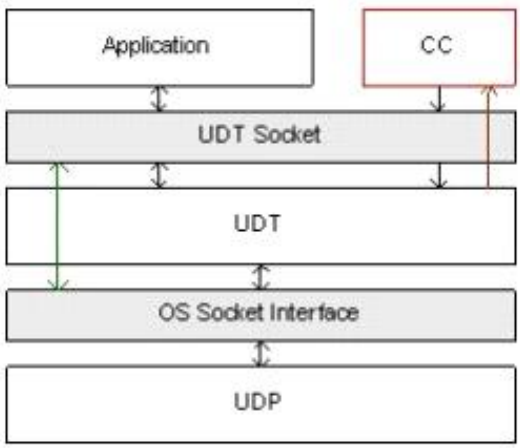


그림 3 UDT Process & Structure

2) UDT 특징

UDT는 User Datagram Protocol (UDP) 위에 혼잡 제어와 안정성 제어 메커니즘이 추가되어 설계되었다. UDT는 어플리케이션 레벨 연결이며, 안정적인 데

이터 스트리밍과 부분 신뢰성 메시징을 지원하는 이중 프로토콜이다.

◆ ACK 전송

UDT는 주기적인 ACK를 사용하여 패킷이 제대로 전달되었는지 확인하고, 네거티브 ACK를 사용하여 손실된 패킷에 대해 보고 받는다. 주기적인 ACK는 데이터 전송 속도가 빠를 때, 제어 트래픽을 줄여주는 데, 이런 상황은 ACK 개수가 데이터 패킷의 수보다는 시간에 비례하기 때문이다.

◆ AIMD의 증가 파라메터를 통한 제어

UDT는 AIMD(Additive Increase Multiplicative Decrease)식의 혼잡 제어 알고리즘을 사용한다. 이 알고리즘의 증가 파라메터는 이용 가능한 대역폭(패킷 페어 기술을 사용하여 측정)에 반비례한다. 그래서 UDT는 빠르게 높은 대역폭을 확인할 수 있으며, 최대 대역폭을 넘어섰을 때 안정성을 위해 속도를 줄일 수 있다. 감소 비율은 1/8과 1/2 사이이다. 이것으로 동기화 손실이라는 부정적인 영향을 줄일 수 있다. UDT의 패킷 전송은 전송 속도 제어와 윈도우 제어에 의해 제한된다. 전송 속도는 AIMD 알고리즘에 의해 업데이트된다. 혼잡 윈도우는 보조 제어 장치이며, 수신 데이터 도달 비율에 따라 설정된다.

◆ 혼잡 제어 설정

UDT는 C++ 클래스의 혼잡 제어에 대한 변수 셋를 공개하고 있으며, 사용자가 이 변수를 조작하는 콜백 함수를 정의 할 수 있다. 따라서 사용자는 이러한 콜백 함수의 일부 또는 전부를 오버라이딩하여 제어 알고리즘을 다시 정의 할 수 있다.

◆ 랑데부 연결 설정

UDT는 기존의 클라이언트/서버 연결 설정 모드와 새로운 랑데부 연결 설정 모드를 모두 지원한다. 후자는 두 피어에 모두 방화벽이 있을 때, 방화벽을 통과하기 위해 사용된다.

나. GridFTP

1) GridFTP 개요

GridFTP는 File Transfer Protocol(FTP)를 확장하여 빠른 속도로 신뢰성 있고 안전하게 데이터 전송을 하기 위해서 만들어졌다. 이 프로토콜은 Open Grid

Forum의 GridFTP working group에 의해서 정되었다. 여러 GridFTP 중 Globus 툴킷에서 제공하는 것을 가장 많이 사용한다.

GridFTP의 목적은 더 안정적으로 고성능 파일 전송을 제공하는 것이다. GridFTP는 슈퍼 컴퓨터 센터, 기타 과학 시설, 고에너지물리(Large Hadron Collider)와 같은 큰 프로젝트에 많이 사용된다. 또한 GridFTP는 스토리지와 액세스 시스템 간의 비호환성 문제를 해결한다. 이전에는 각 데이터 공급자가 액세스 함수 라이브러리를 제공하여 데이터를 이용 가능하게 하였다. 하지만 다양한 자원들로부터 데이터를 가져올 때, 각각 다른 액세스 함수를 필요로 하기 때문에 불편함이 있었다. GridFTP는 보편적인 FTP표준을 확장시키고, 여러 액세스 모드의 기능을 포함하고, 데이터에 접근하는 것에 통일된 방식을 제공한다. FTP가 가장 널리 쓰이고, 프로토콜을 확장하기 좋은 구조로 정의되어 있어 FTP를 토대로 사용한다.

2) GridFTP 특징

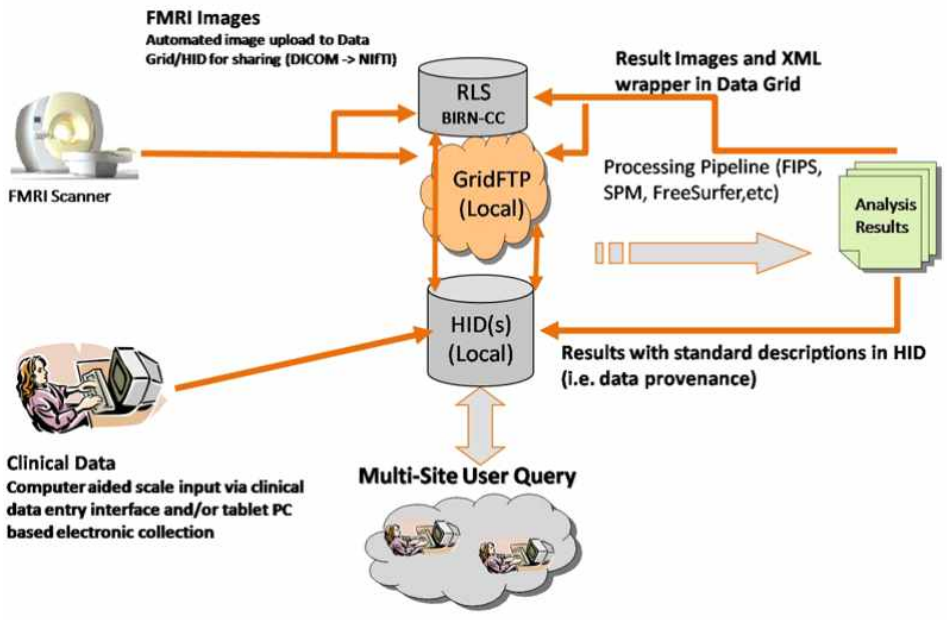


그림 4 The Architecture of GridFTP

◆ 그리드 보안 인프라 스트럭쳐

Grid FTP는 Grid Security Infrastructure(GSI)를 사용하여 보안을 유지한다. FTP는 패킷을 해킹당하기 쉬운 안전하지 못한 구조로 되어있어, SSH와 SSL 등을 보안을 위해 사용하였다. GSI는 사용자 지정 수준의 기밀성과 데이터 무

결성과 함께 파일 전송의 인증과 암호화를 제공한다.

◆ 데이터 전송의 써드 파티 제어

FTP의 편리한 기능 중 하나는 로컬 클라이언트가 서버 간 원격 전송을 할 수 있다는 것이다. GridFTP는 이러한 구조에 로컬에서 초기 설정을 할 때의 보안과 인증을 강화하였다.

◆ 병렬 데이터 전송

GridFTP는 여러 개의 TCP 스트림을 동시에 이용할 수 있도록 하여 전체 대역폭을 향상 시킬 수 있다. 파일들은 각기 다른 곳으로부터 동시에 다운로드 되거나 같은 곳에서 병렬 스트림으로 나뉘어 다운로드 된다.

◆ 스트라이핑 데이터 전송

데이터는 DPSS의 네트워크 디스크 캐시처럼 다수의 서버에 중첩되어 저장될 수 있다. GridFTP는 같은 데이터가 여러 서버에 분리돼 저장되어 있는 경우 이 데이터를 전송하기 위해 각 서버로 다수의 TCP 스트림을 열어 사용하는 스트라이핑 전송을 지원한다. 스트라이핑 전송은 병렬 전송보다 더욱 전체 대역폭을 향상시킬 수 있다.

◆ 부분 파일 전송

어떤 애플리케이션은 전체 파일보다는 파일의 일부분만을 필요로 할 수 있다. 예를 들면 커다란 데이터베이스 파일에서 일부분만을 요구하는 고에너지 물리학 분석이 있을 수 있다. 현재 FTP 프로토콜은 특별한 시작점에서 파일의 나머지 부분을 전송할 수 있다. GridFTP에서는 이러한 기능을 더욱 확장해 파일의 임의의 영역을 전송하도록 하는 커맨드를 제공한다.

◆ 자동 TCP 버퍼/윈도우 크기 협상

TCP 버퍼/윈도우 크기를 최적으로 설정한다면 데이터 전송 성능을 크게 향상시킬 수 있다. 그러나 수동적인 TCP 버퍼/윈도우 크기 설정은 자주 오류를 발생시키며 단순한 작업도 아니다. GridFTP는 커다란 파일과 대규모의 작은 파일들에 대한 수동 및 자동 TCP 버퍼 크기 협상을 지원한다. 이를 위해 표준 FTP 커맨드와 데이터 채널 프로토콜을 확장했다.

◆ 신뢰적이고 재시작 가능한 데이터 전송 지원

신뢰적인 전송은 데이터를 관리하는 많은 애플리케이션에게 중요한 요소이다. 일시적인 네트워크나 서버 고장과 같은 장애를 해결하기 위한 장애 복구 방법이 필요하다. FTP 표준은 실패한 전송을 재시작하는 기능을 지원하지만 많이 이용되고 있지는 않다. GridFTP는 FTP의 이러한 기능을 이용하며 새로운 데이터 채널 프로토콜을 지원하기 위해 이를 확장했다.

4. 실증환경테스트베드 설계

본 장은 실증환경테스트베드의 설계 내용 기술한다. 실증환경테스트베드는 기본적으로 OpenStack의 컴포넌트들을 기반으로 전체적인 구조를 갖추며, 부가적으로 사용자 데이터 전송의 편의를 돕기 위하여 고속전송도구를 OpenStack 컴포넌트에 플러그인 하는 방식을 추가 설계하였다. 실증환경테스트베드는 응용연구자에게 사용자 연구 환경을 제공하기 위해서 다음과 같은 구조를 갖는다.

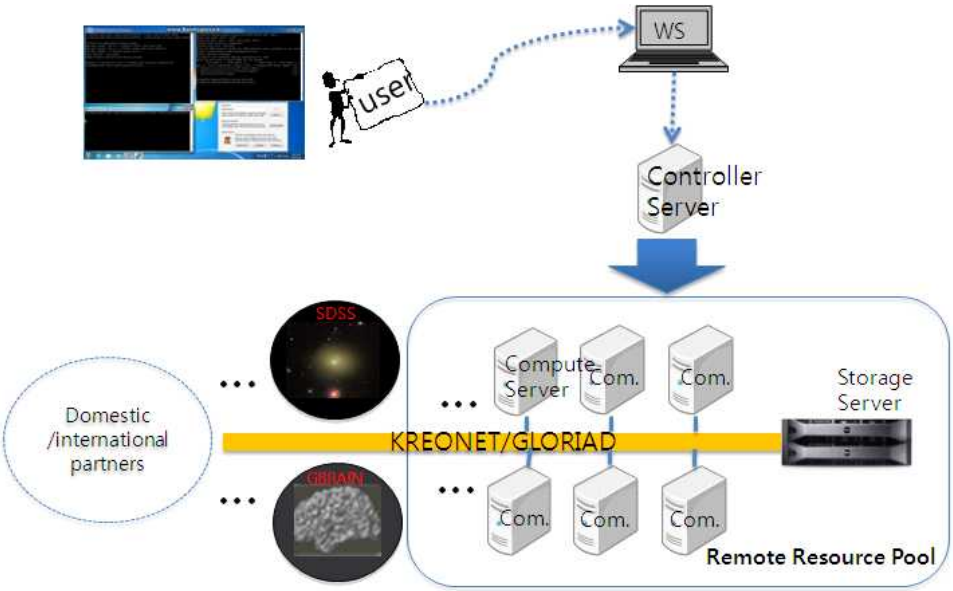


그림 5 실증환경테스트베드 물리적 구조

구성요소로는 사용자의 요청을 받아들일 수 있는 웹서버(Web Server, WS), 컴퓨팅자원을 제공하는 컴퓨트 서버들(Compute Servers), 실증환경테스트베드 서비스를 제공하기 위하여 사용자 인증, 이미지 관리, 시스템 할당등의 전반적인 서비스를 운영하는 컨트롤 서버(Control Server)가 있다. 이러한 요소들의 서비스들은 OpenStack 컴포넌트들을 이용하여 배치한다. 따라서 앞서 언급한 실증환경테스트베드의 요구사항을 기반으로 다음과 같이 시스템을 설계하였다.

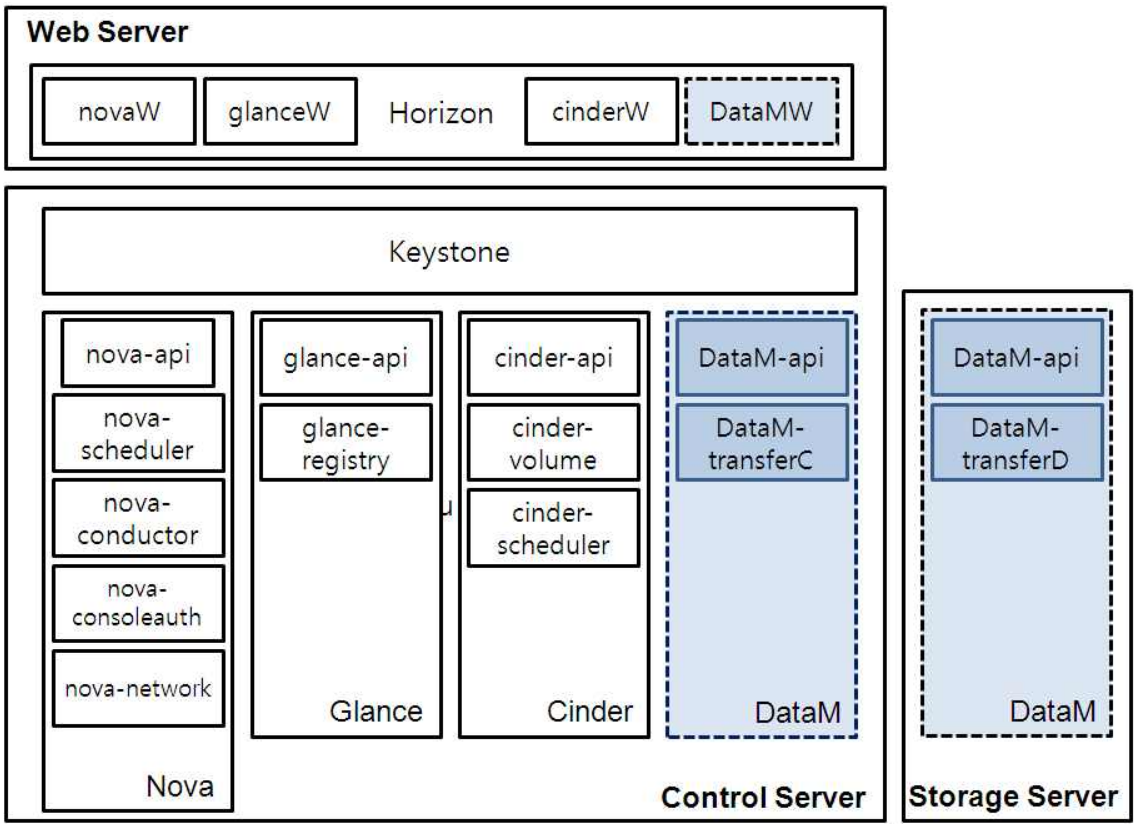


그림 6 실증환경테스트베드 시스템 설계

(1) 사용자 관리 기능

본 기능은 OpenStack에서 제공하고 있는 인증서비스인 Keystone을 이용하여 배치한다. 이를 통해 앞서 기능 요구사항 1의 "사용자 계정 삭제", "사용자 계정 추가"기능 구현이 가능하고, 더불어 사용자의 인가 정도에 따라 접근할 수 있는 서비스/행위 제어가 가능한 인증 기능도 제공한다.

(2) 실험 환경 관리 기능

OpenStack에서 제공하고 있는 컴퓨트 서비스인 Nova를 이용하여 실험환경관리기능을 제공한다. 실험 환경을 구성하기 위해서는 OpenStack의 Nova 컴포넌트의 다수 모듈들은 Control 서버와 Compute 노드에 기능에 맞게 분산되어 배치되어야 한다. 다시 말해서, 외부와의 인터페이스를 수행하는 nova-api, 요청내용을 큐잉하고 배분하는 nova-scheduler, vnc-console을 인증하는 nova-consoleauth, 할당받은 노드들의 네트워크를 구성하는 nova-network는 Control 서버에 배치되고, 하이퍼바이저를 통해 가상 머신 인스턴스를 할당해주는 nova-compute는 Compute 노드에 배치되어야 한다.

기능 요구사항 2의 역할을 수행하기 위해서 nova의 각 모듈들은 그 외 다른 컴포넌트들과 다음과 같은 관계를 갖으며, 실험환경 구성기능을 수행한다.

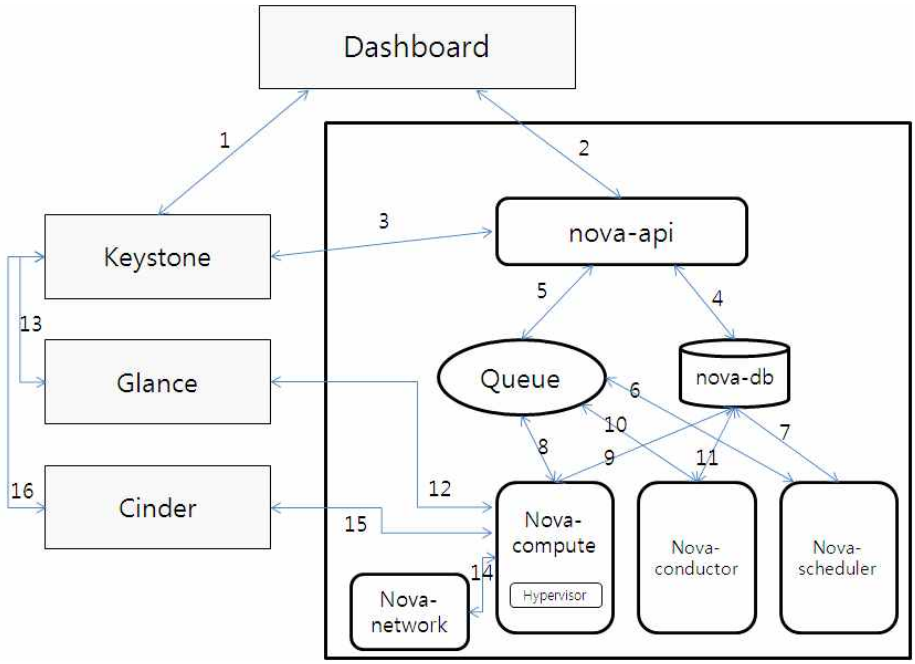


그림 7 가상머신 생성 프로세스

(3) 운영체제 관리 기능

본 기능은 OpenStack에서 제공하고 있는 이미지 서비스인 Glance를 이용하여 배치한다. Glance는 기본적으로 glance-api와 glance-registry라는 두 개의 데몬으로 구성된다. glance-api는 외부로부터의 요청을 처리하기 위한 역할을 하고 glance-registry는 내부적인 작업을 수행하기 위한 역할을 한다.

기능요구사항 3.1인 "운영체제 추가"는 아래 그림과 같은 절차로 수행된다. glance-api가 운영체제 추가 요청을 받은 후, glance-registry 데몬에게는 해당 이미지에 대한 메타데이터 저장을 요청하고 실제 이미지는 이미지 저장 공간으로 설정된 위치(ex. Swift)로 전송한다.

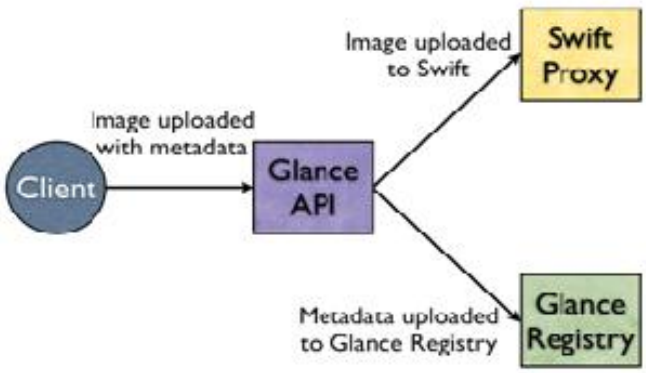


그림 8 이미지 업로드

기능 요구사항 3.2.인 “운영체제 삭제”는 glance-api가 glance-registry로부터 해당 이미지에 대한 저장위치를 얻어와서 저장공간에 있는 이미지를 삭제한다. 기능 요구사항 3.3인 “스냅샷 저장”은 “운영체제 저장”과 같은 방식으로 수행한다.

(4) 사용자 데이터 관리 기능

사용자 데이터 관리 기능은 OpenStack의 블록 스토리지 서비스인 Cinder를 이용하여 배치하고 추가적인 전송 기능에 대해서는 추가 개발이 필요하다. Cinder는 cinder-api, cinder-volume, cinder-scheduler라는 세 개의 모듈로 구성된다. cinder-api는 외부와의 요청을 담당하고, cinder-volume은 스토리지 서버와의 연결을 담당하고 cinder-scheduler는 cinder-api로부터의 요청을 받아 유휴한 스토리지 공간에 볼륨을 할당하거나, 할당된 볼륨을 인스턴스에 attach 하는 등의 실질적인 작업들을 수행한다.

기능요구사항 3.1인 “볼륨 생성”은 사용자가 필요로 하는 크기의 볼륨을 스토리지 서버에 생성시키는 것으로서, cinder-api를 통해 사용자의 요구사항이 전달되면 cinder-scheduler는 cinder-volume으로부터 스토리지 서버의 정보를 얻어 사용자가 요구한 볼륨을 생성한다.

기능요구사항 3.2인 “볼륨 삭제”는 할당되었던 볼륨을 해제시키는 것으로, cinder-api를 통해 사용자의 요구사항이 전달되면 cinder-scheduler는 스토리지 서버에 위치한 해당 볼륨을 삭제한다.

기능요구사항 3.3인 “데이터 업로드”와 3.4인 “데이터 다운로드”는 해당 볼륨에 외부 머신으로부터 사용자의 데이터를 저장하거나 해당 볼륨의 데이터를 사용자 머신으로 는 것으로서, 본 실증환경테스트베드에서는 DataM이라는 컴포넌트를 통해서 기능을 제공한다.

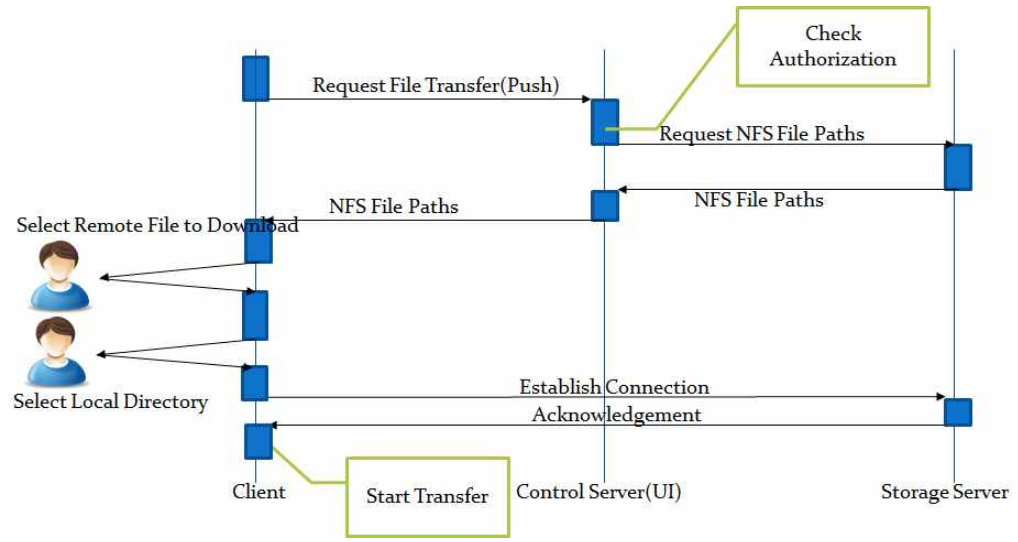


그림 9 File Transfer Process

위의 그림에서 보여주고 있는 것처럼, 사용자가 컨트롤 서버로 사용자가 선택한 볼륨의 디렉토리 정보를 요청하면 컨트롤 서버는 사용자의 요청이 유효한지 먼저 체크를 한 후 유효한 경우에 스토리지 서버로 볼륨 내용 전달을 요청한다. 스토리지 서버는 컨트롤 서버로부터 해당 요청을 받고 사용자가 요청한 볼륨의 디렉토리 정보를 전달한다.

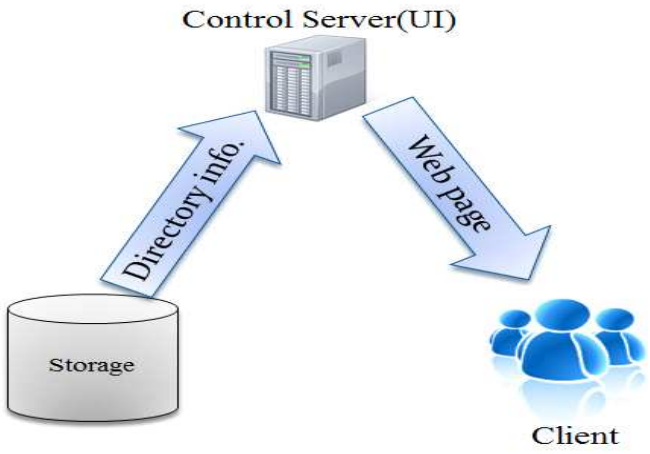


그림 10 Process of retrieving remote storage directory contents

이후 사용자가 사용자 머신으로부터 해당 볼륨으로 데이터를 전송하거나 사용자 머신으로 해당 볼륨의 데이터를 전송하게 되는 경우, 스토리지 서버와 클라이언트간 파일 전송 데몬을 이용하여 직접적으로 통신하게 된다.

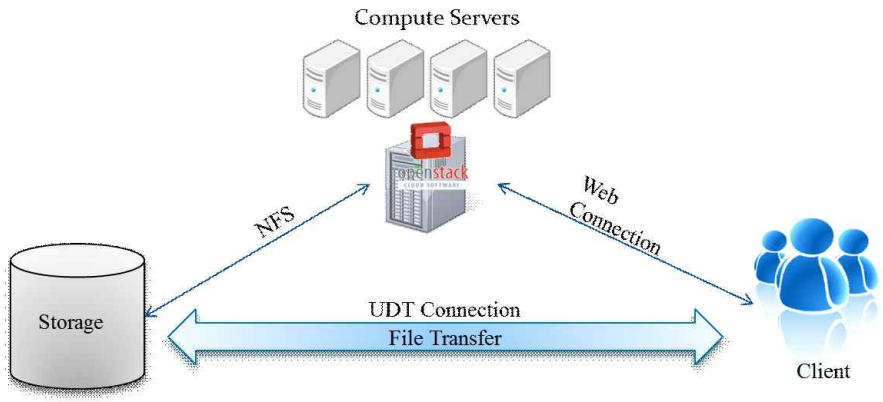


그림 11 File Transfer Process with external storage server

다시 정리하면, 스토리지 서버의 DataM-api는 컨트롤 서버로부터 사용자요청을 전달받아서 볼륨 디렉토리 정보를 리턴하는 작업과 사용자로부터 직접적으로 파일 전송 요청 받는 작업을 수행한다. 사용자가 볼륨 내의 파일을 전송 요청하거나 볼륨내로 사용자의 파일 전송을 요청할 경우 컨트롤 서버는 사용자에게 고속 전송 툴킷(ex. UDT client)을 사용자머신으로 다운받게 해주며, 그 후 그 툴을 이용하여 스토리지

서버의 DataM-transferD 모듈과 직접 통신하여 파일 전송 작업을 수행한다. 이러한 형태로 실증환경테스트베드에서는 요구사항 3.3 "데이터 업로드", 3.4 "데이터 다운로드" 기능을 제공한다.

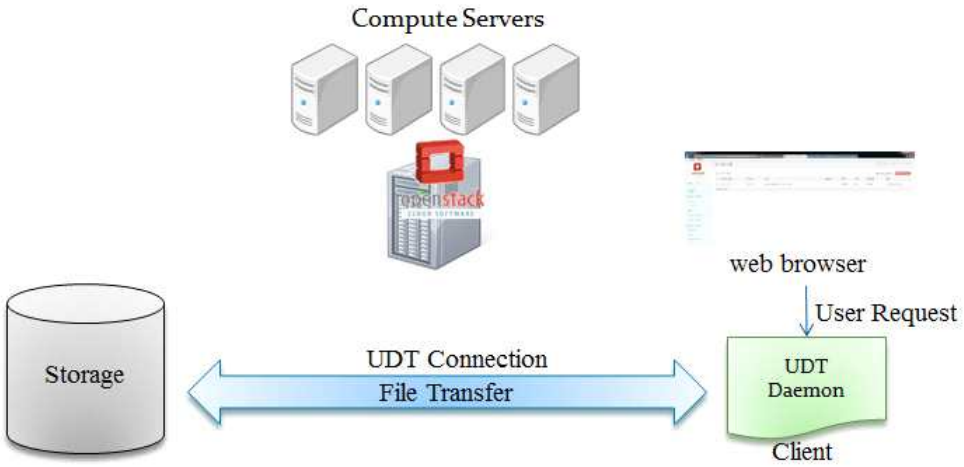


그림 12 File Transfer Operation on Client with OpenStack Web-page

5. 실증환경시스템 구현

실증환경테스트베드에는 아래 그림에서와 같이 노드 및 사용자 관리 등 전반적인 제어 및 관리를 할 수 있는 요소로서 컨트롤 서버가 필요하고, 사용자 데이터를 저장하기 위한 요소로서 스토리지 서버가 필요하고, 사용자의 컴퓨팅 자원을 지원하기 위하여 컴퓨트 노드들이 필요하다. 앞서 언급한 바와 같이 실증환경테스트베드를 구축함에 있어 OpenStack에서 제공하고 있는 기능들은 OpenStack 컴포넌트들을 설치하여 구축하고 그 외 기능에 대해서는 개발하는 방향으로 한다. 이에 본 장에서는 OpenStack 컴포넌트 설치 내용과 개발 내용에 대해서 기술하도록 한다.

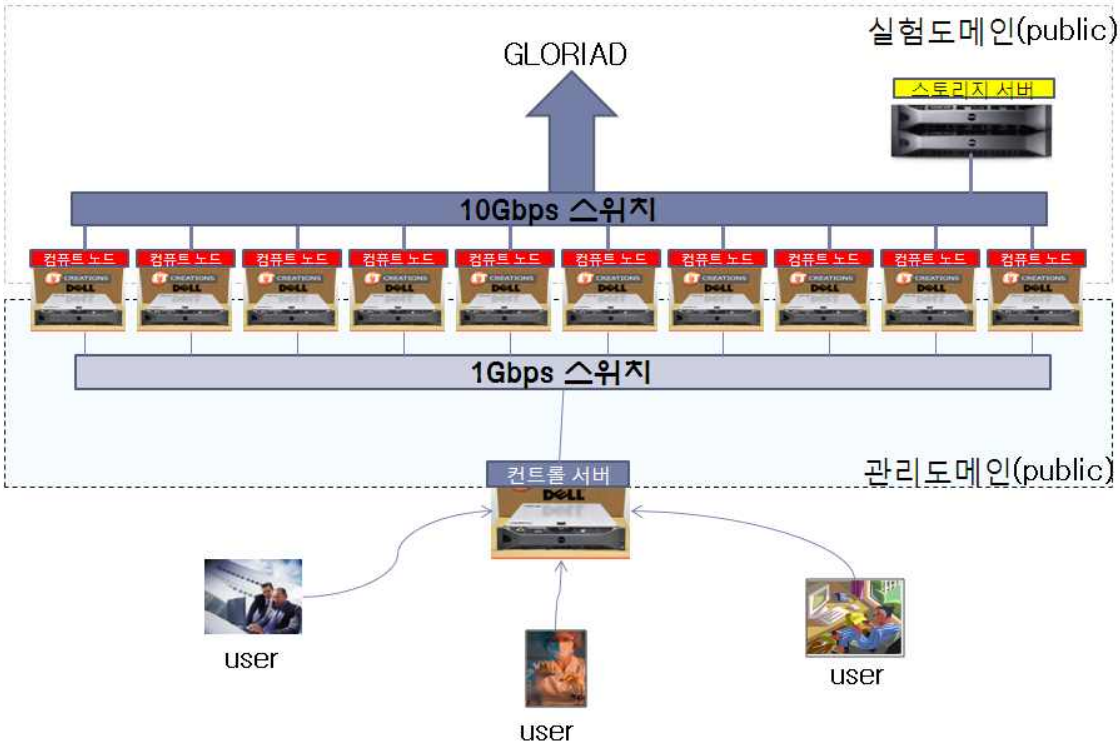


그림 13 실증환경테스트베드 물리적 구성

(1) OpenStack 컴포넌트 설치

◆ 컨트롤 서버

컨트롤 서버는 실증환경테스트베드의 전반적인 관리 기능을 제공해야 하므로, 다음과 같은 OpenStack 컴포넌트들을 설치한다.

- ✓ keystone
- ✓ glance
- ✓ cinder
- ✓ nova

추가로 컨트롤 서버는 스토리지 서버로 사용자 데이터 관리에 관한 통신이 필요하므로 다음과 같은 NFS 관련 모듈을 설치한다.

- ✓ nfs-common, portmap

◆ 스토리지 서버

스토리지 서버는 실증환경테스트베드에서 사용자데이터 관리 기능을 제공해야 하므로 실험자원을 할당하는 컴퓨터노드들이 이후 연동하여 사용할 수 있도록 NFS(Network File System) 관련 모듈을 설치한다.

- ✓ nfs-kernel-server, nfs-common, portmap

◆ 컴퓨트 노드

컴퓨트노드는 실증환경테스트베드에서 사용자에게 컴퓨팅 자원으로서 할당되는 시스템으로서 컨트롤 서버로부터 자원 할당 명령을 통해 가상 머신을 할당할 수 있어야 한다. 이를 위해 컴퓨트 노드들에는 다음과 같은 OpenStack 컴포넌트를 설치한다.

- ✓ nova의 일부 모듈인 nova-compute

그리고 각각의 컴퓨트노드에는 nova-compute의 명령에 의해 실제 가상 머신을 할당할 수 있도록 kvm이나 xen 등의 하이퍼바이저가 설치되어 있어야 한다.

(2) 사용자 데이터 관리 개발

사용자 데이터 관리 기능 중에서 OpenStack의 cinder 컴포넌트에서 수용되지 못한 기능들에 대하여 개발이 수행되었다. 사용자 데이터 관리를 위하여 cinder에서는 볼륨을 생성하고 가상 머신인 인스턴스에 붙여주는(attach)하는 기능까지 제공해주고 있어서 데이터 저장공간을 포함한 사용자의 실험환경을 잘 구성할 수 있다. 그러나 사용자가 해당 데이터 공간에 사용자의 데이터를 업로드하거나 해당 데이터 공간에서 사용자의 머신으로 다운로드 하기 위해서는 인스턴스를 거쳐서 전송할 수 밖에 없어서 불필요한 병목이 예상된다. 이에 본 실증환경테스트베드에서는 사용자 데이터의 업로드와 다운로드에 있어서 사용자 머신과 스토리지 서버간에 전송할 수 있도록 개발하였다.

가. 디스크 볼륨 생성

실증환경테스트베드의 웹환경에서 사용자가 해당 볼륨의 파일을 전송하거나 해당 볼륨으로 파일을 전송하기 위해서는 디태치(detach)되어 있는 볼륨을 이미지 마운트시켜서 디렉토리 내용을 확인하는 과정이 필요하다. 동작순서는 다음과 같다.

- ✓ 사용자가 웹을 통해 컨트롤서버에 볼륨 생성 명령을 전달
- ✓ 컨트롤서버는 Cinder 모듈을 통해 NFS로 연결된 스토리지서버에 볼륨을 생성

✓ 스토리지서버는 해당 볼륨을 로컬 디렉토리에 마운트 (마운트 과정은 QCOW2 파일 시스템 이미지 마운트 과정을 통해 진행)

나. 리모트-로컬 파일 선택

사용자가 파일들을 선택하기 위해서는 실증환경테스트베드의 웹에 파일 브라우저를 통해 볼륨의 디렉토리 내용을 보여준다. 이를 위한 동작순서는 다음과 같다.

✓ Volumes창에서 Filebrowser 버튼을 선택하면 파일 탐색기가 나타남
✓ 화면에 보이는 파일들은 NFS 서버에 있는 가상머신에 속해있는 파일을 의미함
✓ 전송받을 파일(또는 전송할 폴더)을 선택
✓ 하단의 박스에 로컬의 폴더(또는 파일)를 선택
✓ 전송 버튼을 선택할 경우 Download(또는 Upload)가 수행됨

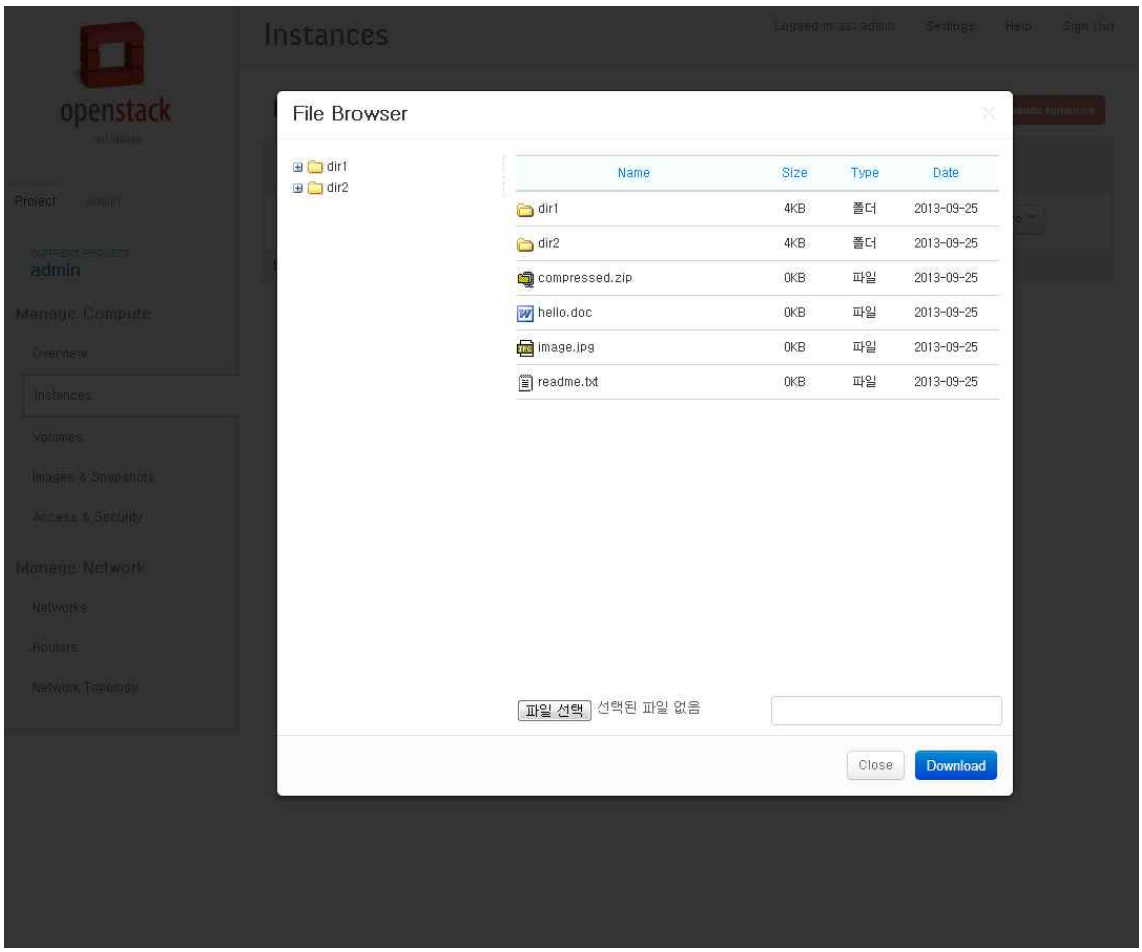


그림 14 UI for select file to down/upload

다. 파일 전송

파일 브라우저를 통해 스토리지서버와 사용자 머신간 업로드 및 다운로드 파일을 선택하고 나면 이후 두 객체간 전송이 이루어진다. 동작순서는 아래와 같다.

- ✓ Client는 Command line을 기반으로 스토리지서버로 명령어를 전달
- ✓ 스토리지서버는 Client로부터 전달받은 명령어를 분석
- ✓ Client가 스토리지서버로 “send” 명령어를 요청한 경우
  - ☞ Client는 자신의 로컬 영역에서 스토리지서버가 파일을 받을 준비가 되었는지 점검
  - ☞ 스토리지서버 Client로부터 파일을 받을 준비가 되었다면, Client에게 Ack를 전달
  - ☞ Client는 스토리지서버로부터 Ack를 받은 후, 전송할 파일에 대한 정보를컨트롤서버로 전달
  - ☞ 스토리지서버 Client로부터 파일 정보를 전달 받은후에, Client는 해당 파일을 스토리지서버로 UDT를 이용하여 전송
- ✓ Client가 스토리지서버로 “receive” 명령어를 요청한 경우
  - ☞ Client는 스토리지서버로 부터 파일을 전송받을 준비가 되었다는 Ack를 전달
  - ☞ Client가 스토리지서버로 Ack를 전달한 후에, 스토리지서버는 Client에게 해당 파일에 대한 정보를 전달
  - ☞ Client가 해당 파일에 대한 정보를 전달받은 후에, 스토리지서버는 해당 파일을 Client에게 UDT를 이용하여 파일을 전송

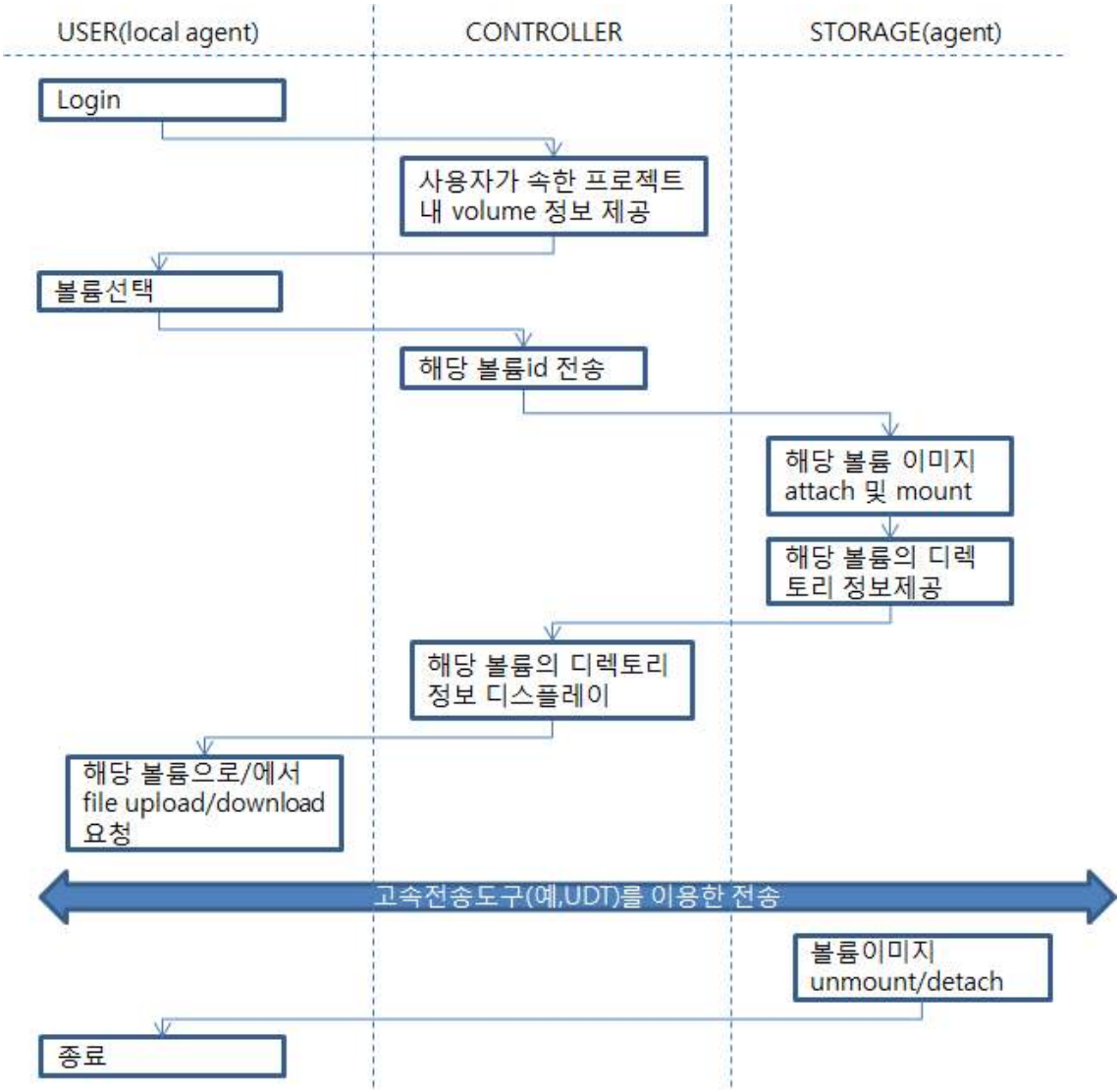
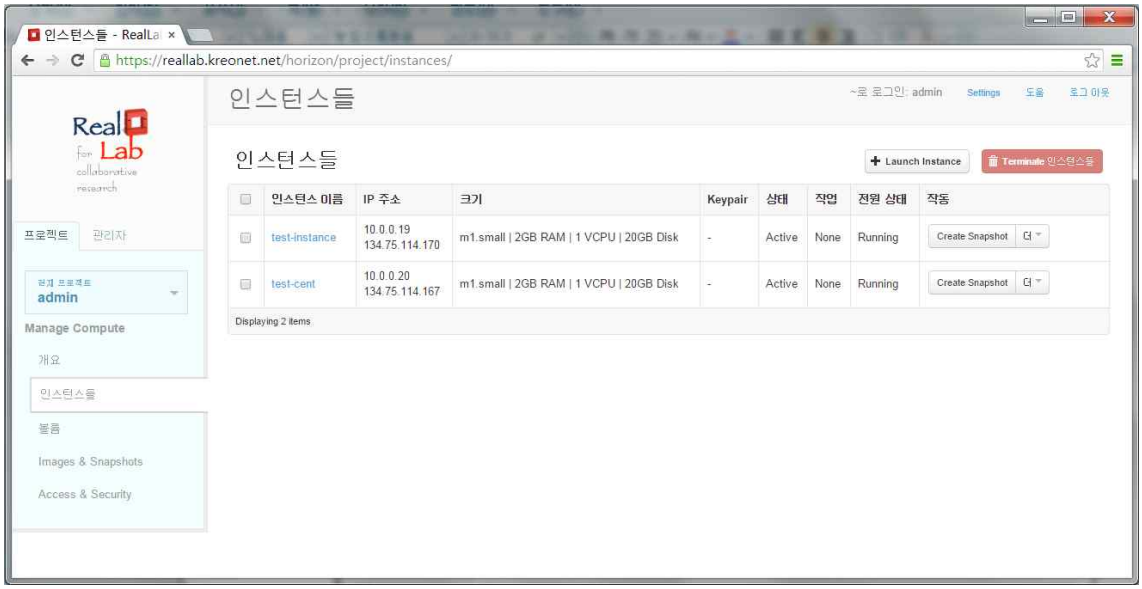


그림 15 개선된 블록 스토리지 접근 플로우차트

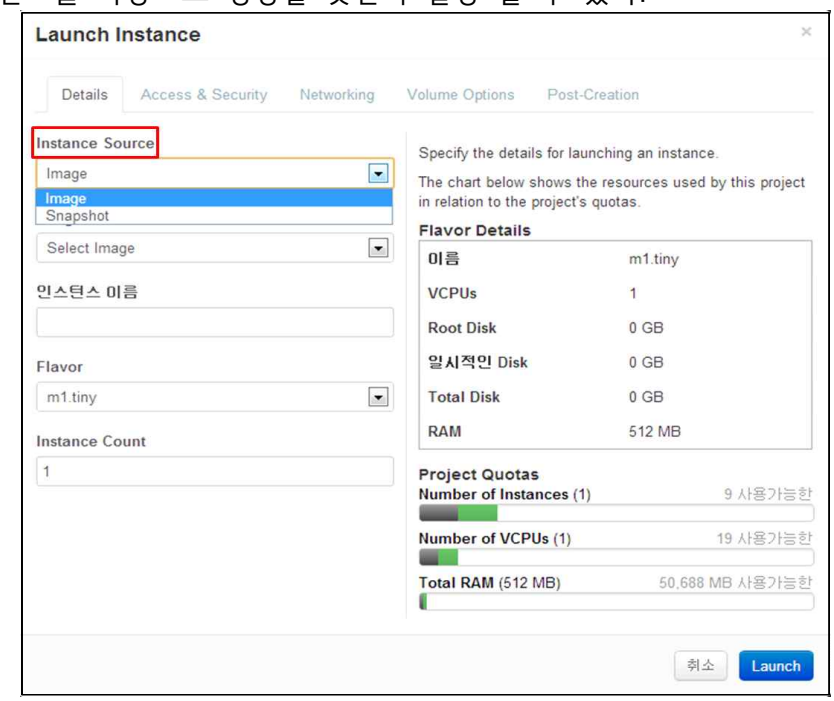
6. 실증환경시스템 사용 예

(1) 동적 연구 환경 구성

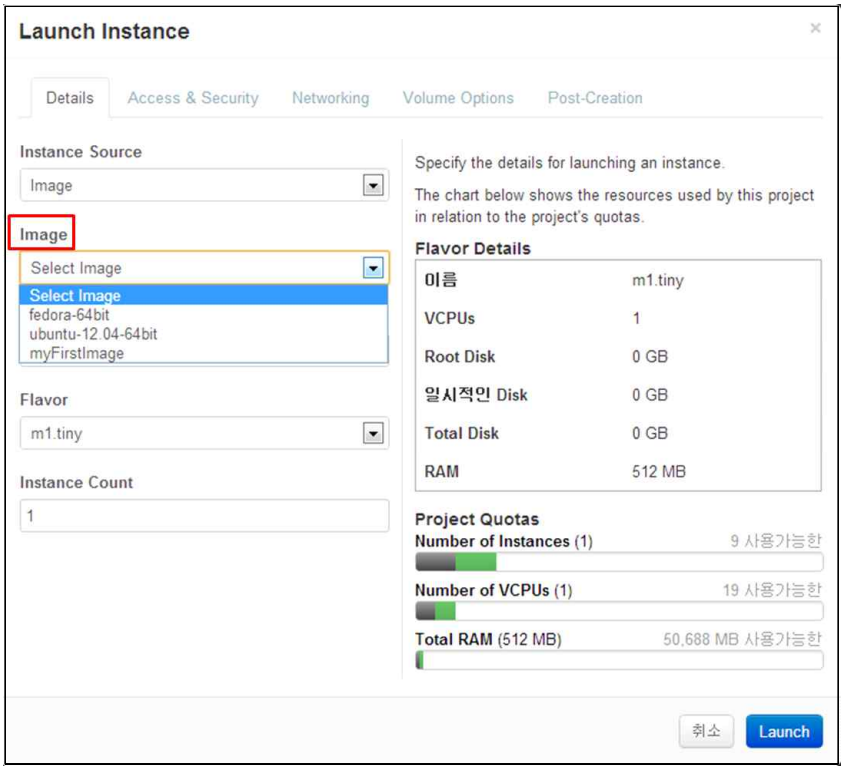
① 프로젝트 항목에서 인스턴스들 메뉴를 통해 현재 시스템에 생성된 실험환경을 확인할 수 있으며, 또한 새로운 실험환경을 생성할 수 있다.
② 새로운 실험환경을 생성하기 위해서는 Launch Instance를 선택한다.



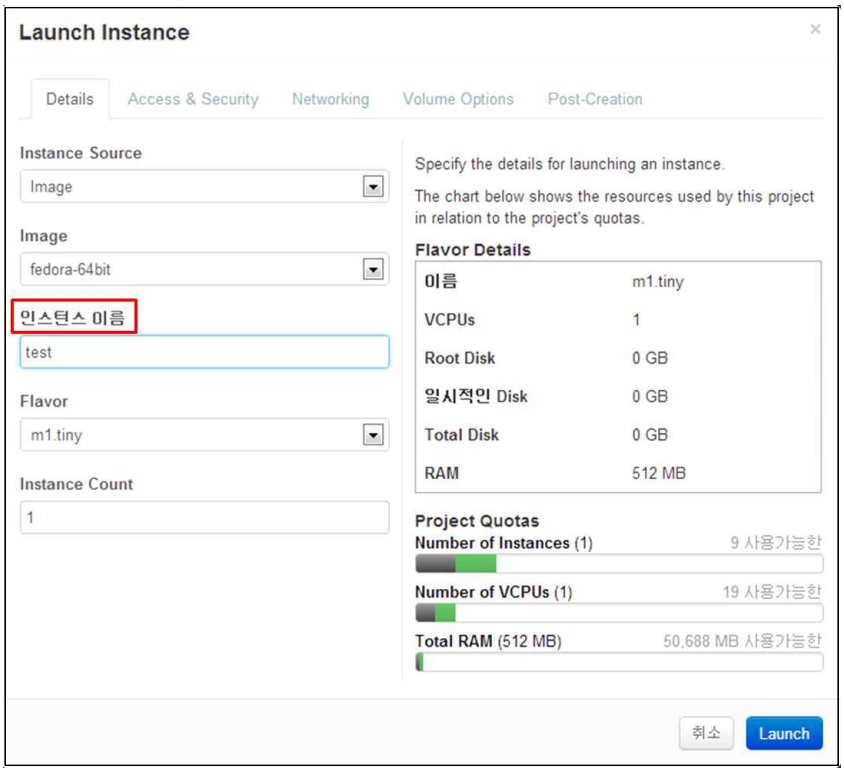
③ Instance Source 메뉴를 통하여 신규 블록 장치를 통해 생성할 것 인지 다른 인스턴스를 바탕으로 생성할 것인지 결정 할 수 있다.



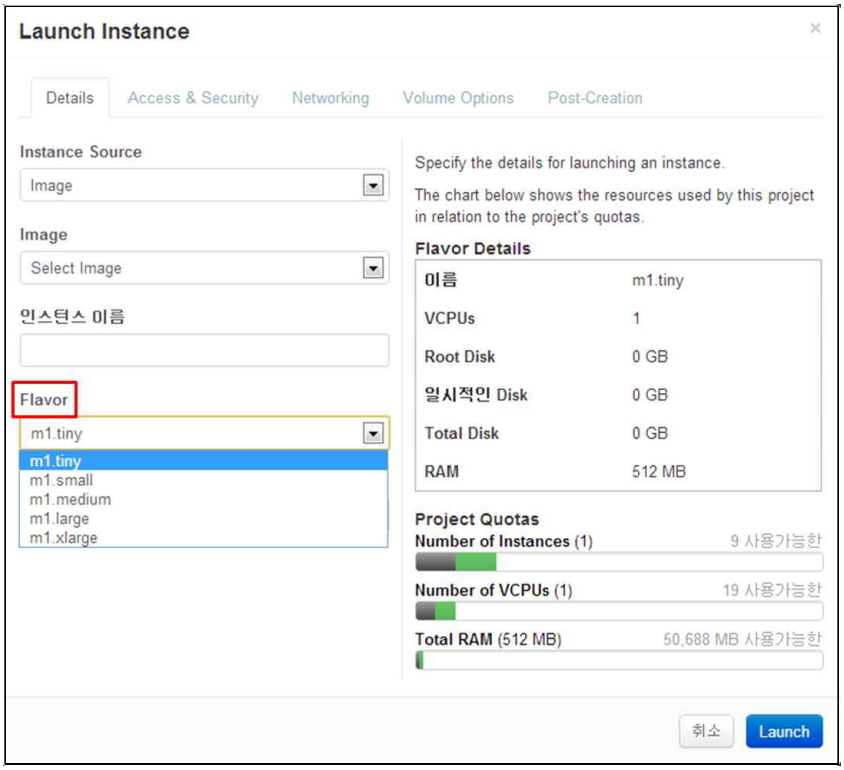
④ Image 메뉴를 통해서 실험 시스템의 운영체제를 선택할 수 있다.



⑤ 인스턴스 메뉴를 통해서 실험 시스템의 이름을 설정할 수 있다.

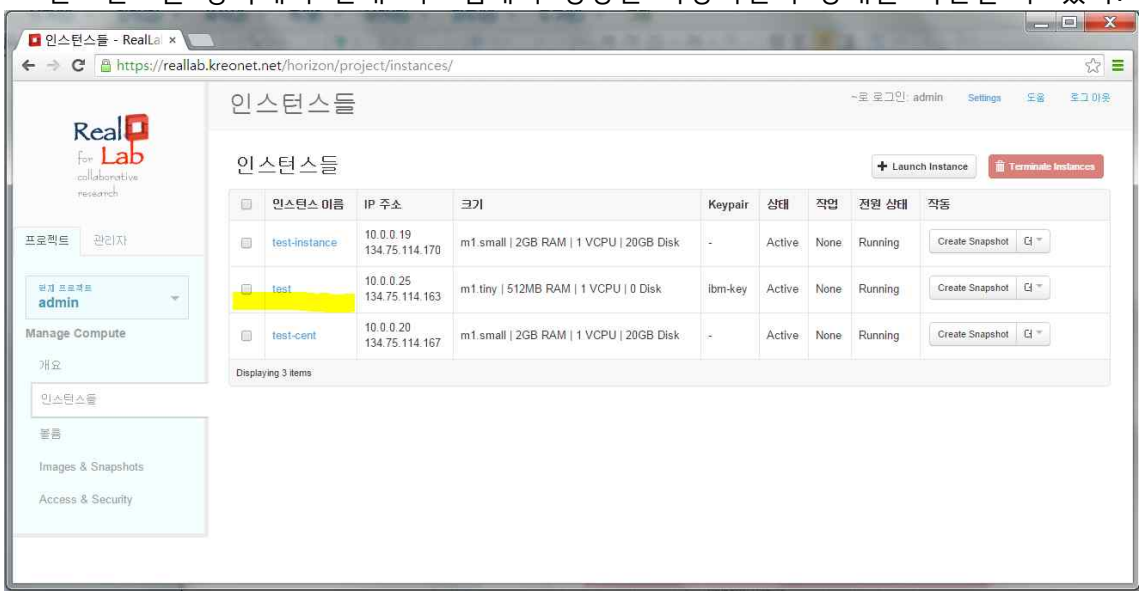


⑥ Flavor 메뉴를 통해서 실험 시스템의 리소스(Flavor Details)를 설정할 수 있다.



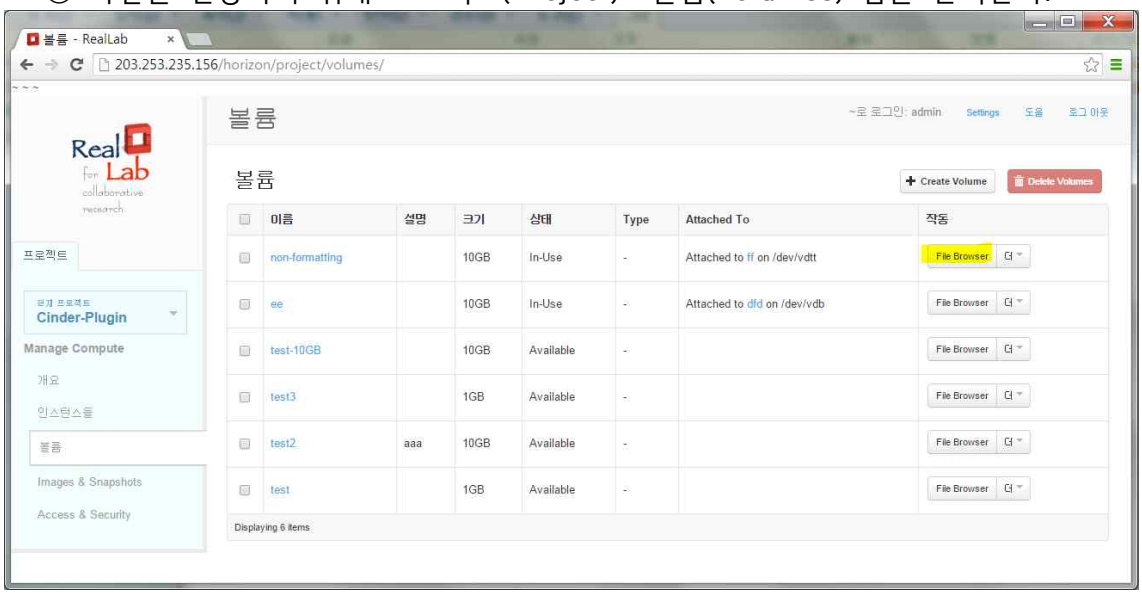
⑦ 마지막으로 Launch 버튼을 클릭해주면 실험 시스템이 생성된다.

인스턴스들 항목에서 현재 시스템에서 생성된 가상머신의 상태를 확인할 수 있다.

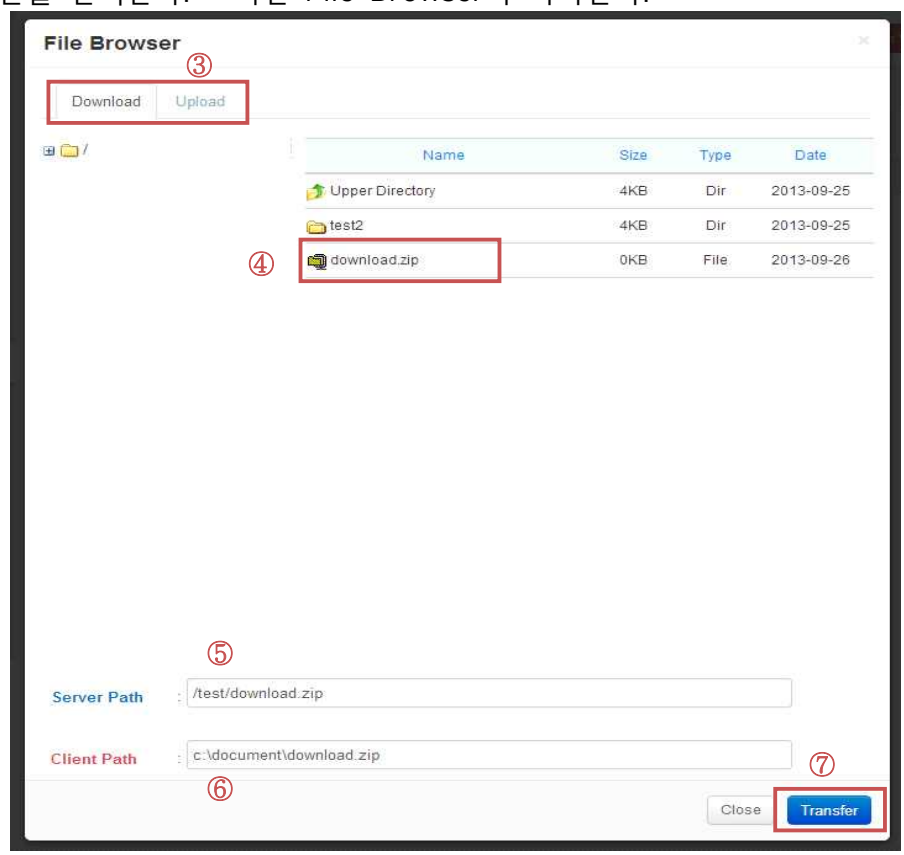


(2) 사용자데이터 관리

① 파일을 전송하기 위해 프로젝트(Project)->볼륨(Volumes) 탭을 선택한다.



② 메인 화면의 Volumes 리스트에서 파일을 전송할 volume의 File Browser 버튼을 선택한다. 그러면 File Browser가 나타난다.



③ 다운로드 또는 업로드 탭을 선택한다.
④ 다운로드 할 파일(또는 업로드 할 경로)를 선택한다. 그러면 Server Path의 경로가 자동으로 업데이트 된다.
⑤ 필요에 따라 Server Path를 수동으로 입력한다.

⑥ 다운로드 할 경로(또는 업로드 할 파일)의 Path를 입력한다.
⑦ 전송 버튼을 누르면 전송이 시작된다.

7. 결론

본 문서에서는 종단에 있는 응용연구자들이 국가 연구망의 고대역폭 백본을 이용하여 자신의 응용 실험들을 진행할 수 있는 실증환경테스트베드의 요구사항 및 시스템 설계, 그리고 구현에 대하여 기술하고 있다. 지금까지 백본의 고도화와 그에 부응하는 기술적 진보가 있었음에도 연구망의 직접적 수혜대상이 되어야 하는 대부분의 응용연구자들은 혜택을 보기 어려웠다. 그 이유는 실험실 내 시스템 자체의 성능이나 실험실에 연결되어 있는 네트워크 성능 등이 고성능 실험을 보장하기 어려운 수준이었기 때문이다. 이러한 연구 환경에 대한 문제를 해결하는 하나의 방편으로 우리는 물리적으로 고성능컴퓨팅/대용량저장공간/고대역폭네트워킹이 잘 구성되어 있는 하나의 연구환경 인프라를 다수의 응용 연구 그룹이 이용할 수 있는 실증환경테스트베드 서비스를 제공하고자 한다.

특히 본 테스트베드는 사용자에게 제공하게 되는 실험시스템을 사용자가 요구하는 운영체제로 로딩하여주고 사용자에게 루트 권한을 주는 등 응용 연구자의 실험실 환경을 이용하는 것처럼 사용자의 편의성에 설계 초점을 두었다. 아직까지는 개발하는 부분에 있어서의 검증과 그에 따른 추후작업들이 남아지만, 실증환경테스트베드는 원격으로 사용자가 필요로 하는 고성능, 고대역폭의 연구환경을 구성해 주고, 사용자에게 실험 수행 시스템에 대한 권한을 위임하며, 또 실험 수행 시스템의 온/오프에 상관없이 사용자에게 상시적으로 데이터에 접근할 수 있도록 함으로써 사용자에게 많은 편의성을 제공하고자 한다. 본 실증환경테스트베드 이용을 통해 사용자는 고성능 시스템 구비 및 고대역폭의 네트워크를 구입 또는 임대에 따른 비용을 절감할 수 있을 것으로 기대된다.

참고문헌

[1] Bulk Data Transfer Techniques for High-Speed Wide-Area Networks, Brian L. Tierney, http://fasterdata.es.net/assets/fasterdata/Tierney-bulk-data-transfer-tutorial-Sept09.pdf, 2009

[2] UDT: UDP-based Data Transfer for High-Speed Wide Area Networks, Yunhong Gu and Robert L. Grossman, Computer Networks (Elsevier). Volume 51, Issue 7. May 2007

[3] UDT 관련 홈페이지, http://udt.sourceforge.net/

[4] gridFTP 관련 홈페이지, http://toolkit.globus.org/toolkit/docs/latest-stable/gridftp/

[5] OpenStack 홈페이지, http://www.openstack.org

[6] OpenStack 기술문서: OpenStack Install and Deploy Manual - Ubuntu, 2013.1