K-ONE 기술문서 #2

다중 사이트 OpenStack 클라우드 환경에서 자원 레벨 가시성을 위한 SmartX Agent 설계 및 검증

Document No. K-ONE #2
Version 1.0
Date 2016-05-08
Author(s) 한정수

■ 문서의 연혁

버전	날짜	작성자	비고
초안 - 0.1	2016. 04. 16	한정수	초안 틀 작성
0.2	2016. 04. 18	한정수	1,2장 작성 완료
0.3	2016. 04. 19	한정수	3장 작성 완료
0.4	2016. 05. 06	한정수	4장 작성 완료
0.5	2016. 05. 07	한정수	1, 2장 내용 수정
1.0	2016. 05. 08	한정수	검토 및 오탈자 수정

본 문서는 2015년도 정부(미래창조과학부)의 재원으로 정보통신 기술진흥센터의 지원을 받아 수행된 연구임 (No. B0190-15-2012, 글로벌 SDN/NFV 공개소프트웨어 핵심 모듈/기능 개발)

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No. B0190-15-2012, Global SDN/NFV OpenSource Software Core Module/Function Development)

Contents

K-ONE #2. 다중 사이트 OpenStack 클라우드 환경에서 자원 레벨 가시성을 위한 SmartX Agent 설계 및 검증

그림 목차

K-ONE #2. 다중 사이트 OpenStack 클라우드 환경에서 자원 레벨 가시성을 위한 SmartX Agent 설계 및 검증

1. 다중 사이트 OpenStack 클라우드 및 가시성 개요

1.1. 목적 및 개요

o 본 문서에서는 다중 사이트로 이루어진 OpenStack 클라우드 기반의 SmartX Cloud Playground (i.e. Testbed)에 배포되어 운용 중인 SmartX Box와 클라우드 상의 가상자원을 대상으로 자원, 플로우 및 서비스와 같은 다양한 관점에서 상태를 파악할 수 있도록 도와주기 위해 자체적으로 디자인 및 개발 진행하고 있는 SW 도구인 OpenStack-MultiView (Multi-layer Visibility with SmartX Agent)에서 자원 레벨의 한정된 동적관찰 기능을 제공하는 SmartX Agent의 설계 및 검증 그리고 실제 활용 방법에 대해 상세히 기술한다. SmartX Agent는 다수의 원격지에 분산된 SmartX Box들을 대상으로 자원 레벨의 한정하여 동적으로 상태 파악을 하기 위한 기본적인 행동 지침을 제공한다. 이러한 작업을 통해 물리 박스 및 가상 자원 박스에 대한 다양한 상태 데이터를 수집하고 수집된 데이터들은 오케스트레이션 또는 알람과 같은 다양한 지능적인 행동을 위한 밑거름이 된다. 이러한 기능을 통한 소프트웨어 정의 인프라(Software-Defined Infrastructure) 관점으로 SmartX Cloud Playground를 효율적으로 운영이 가능하다.

1.2. SmartX Box

o 본 기술문서의 자원 레벨 가시성의 대상이 되는 SmartX Box는 ICT 인프라의 기본이 되는 컴퓨팅/네트워킹/스토리지라는 이질적인 자원들을 하나의 Box의 형태로 제공 가능한 Box를 의미한다. 이러한 융합형 자원 박스 개념 이전까지는 상기 자원들을 제공하기 위해서는 특수 목적으로 제작된 하드웨어에 이를 위해 작성된 소프트웨어만을 사용하도록 강요되어 왔다. 하지만 SDN/클라우드/NFV 기술이 각광을 받아 성숙되어 가고 있는 현재 추세와 더불어 x86 기반의 하드웨어 성능이 급격하게 향상되어 컴퓨팅 뿐만 아니라 NFV, 가상화 환경까지 원활하게 지원 가능함이 검증되어 왔다.

o 실제 이러한 추세에 따라 세계적으로 다양한 스타트업기업들이 융합형 자원 박스를 무기로 하여 사업 영역을 확장해 가고 있다. 2013년부터 뉴타닉스(Nutanix), Simplivity, Pluribus 등의 업체들이 컴퓨스토리지 (compu-storage), 서버-스위치 (server-switch) 로 불리는 컴퓨터, 스토리지, 네트워크 등을 결합하는 융합형 인프라 확산을 추구하면서, 용이한 설정, 유연한 운용 지원, 획기적 성능 개선 등을 무기로 scale-in/out 의 장점을 보여주면서 시장 호응을 얻고 있다.
이 뿐만 아니라 Private 클라우드 분야의 강세 기업인 VMware는 Evo:Rail 이라는 융합형 자원 박스의 통합적인 관리를 위한 솔루션을 14년부터 판매하고 있으며, IT 서비스 분야의 거대 기업 페이스북은 자신의 인프라 전체를 자체 제작한

융합형 자원 박스로 대체하였고, 이를 공개하여 오픈소스 융합형 자원 박스 하드웨어 개발을 위한 Open Compute Project를 주도적으로 이끌고 있다.

o 이러한 흐름에 대비하여 국내에서는 GIST의 Networked Computing Systems 연구실의 주도하에 12년~13년에 국내 5개 사이트, 국외 7개 사이트에 한국형 융합 자원 박스로써 SmartX Box를 설계, 배포하여 미래 인터넷 테스트베드의 운용을 시작하였다.[1] 시작 이래로 국내외 연동 사이트 및 Box 수와 같은 규모적 향상 뿐 아니라 초기 SDN, Cloud를 넘어서 IoT, BigData, FastData 등 다양한 연구주제를 위한 실험환경을 제공하는 질적 향상도 꾸준히 도모하고 있다.

o SmartX Box는 클라우드 컴퓨팅의 3대 자원 요소인 컴퓨트/네트워킹/스토리지 자원을 하나의 박스 단위로 확보하여 다양한 서비스에 적합한 자원을 유연하고 신속하게 제공하기 위한 한국형 초융합 자원 박스로 정의한다.

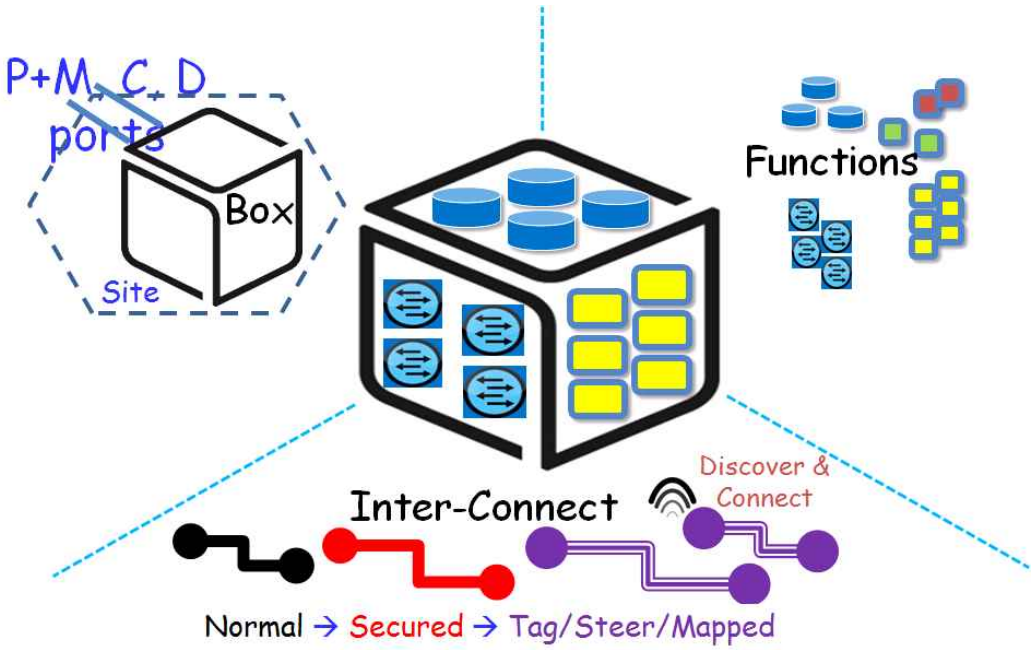

그림 1 SmartX Box 및 Box/Inter-Connect/Function의 3대 요소

o SmartX Box는 그림 1의 가운데 박스 심볼의 형태로 표현하며 박스 내부의 아이콘은 각각 우측 하단부에 컴퓨팅, 좌측 하단부에 네트워킹 그리고 상단부에 스토리지 자원을 의미한다. 따라서 이와 같이 이질적인 자원들을 하나의 박스에 포함된 그림을 SmartX Box의 심볼로 사용함으로써 SmartX Box가 초융합형 자원 박스임을 상징화 하고 있다.

o 그림 1과 같이 SmartX Box 자원집합을 기반으로 Box/Inter-Connect/Functions의 3대요소를 개념화해서 이뤄지는 IoT 서비스 개념은 소프트웨어-정의 패러다임에 따라 빅뱅수준의 속도로 발전하는 ICT 인프라/플랫폼/서비스 통합 기술의 발달 차원에서 바라볼 때 단순하고 명료하면서 미래에도 지속 가능한 기술 구도를 제시하고 있다.

o 신속성과 효율성을 무기로 하는 DevOps(개발/운영 병행체제) 방법론을 도입하여 오픈-소스 소프트웨어를 기반으로 소프트웨어-정의 및 가상화를 지원하는 초융합형(컴퓨트/네트워킹/스토리지를 통합한) SmartX Box 형태의 (오픈소스 White Box 하드웨어와 연계된) 자원집합을 요구되는 서비스의 부하에 유연하게 맞추면서 동시에 자원집합 전반에 걸친 활용도를 최적화하여 신속성과 경제성을 동시에 추구하는 개발/운용 자동화 기술이 요구된다.

1.3. SmartX Cloud Playground

o 클라우드 컴퓨팅이란 인터넷을 통해서 다양한 어플리케이션을 하나의 서비스로써 전달해주는 것과 이러한 서비스를 제공하기 위해서 운영 측면에서 데이터 센터 내의 하드웨어와 소프트웨어 모두를 의미한다 [2].

o 클라우드 운영자는 다수의 박스들을 마치 하나의 거대한 자원 풀로 엮어서 클라우드 사용자에게 가상화된 자원들을 제공할 수 있다. 자원 사용 측면에서 볼 때, 운영자들은 자신의 자원들을 가상화하여 유연하게 사용자들에게 제공해줄 수 있기 때문에 효율적으로 사용할 수 있게 되고, 반면에 클라우드 사용자들은 자신의 물리적인 IT 인프라를 구축하기 위해 필요한 비용은 경제적으로 절약할 수 있게 된다.

o OpenStack이란 이러한 장점이 있는 클라우드를 지원하기 위한 대표적인 클라우드 OS이며, 핵심적인 프로젝트인 Keystone, Nova, Neutron, Glance 등과 같은 여러 프로젝트들이 함께 엮어져서 제공되는 SW 모음이다.

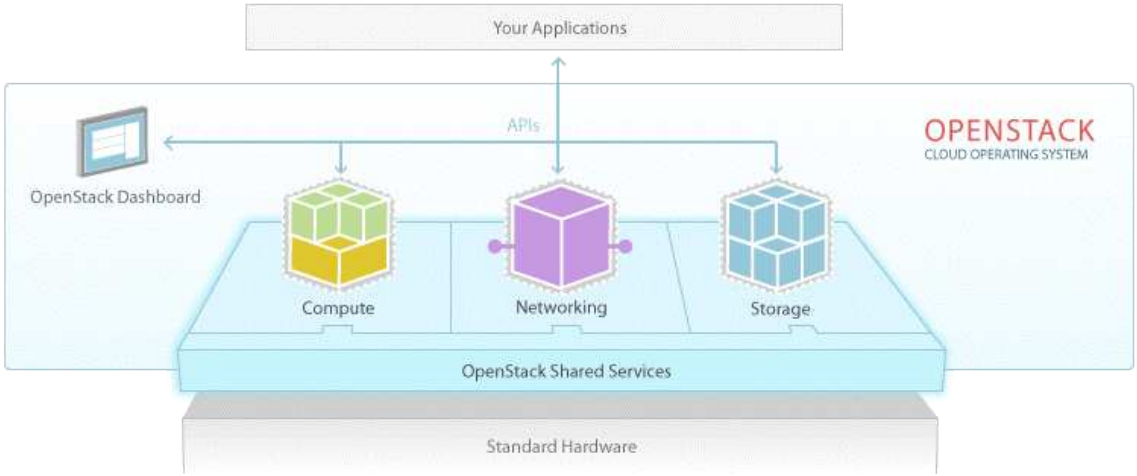

그림 2 OpenStack Cloud OS의 개념적인 구성도

o OpenStack이 나온 지 5년 지난 이 시점에서도 빠른 속도로 성숙해가고 있으며, 거대한 Vendor의 개발자들이 협력하면서 다양한 요구사항들을 반영해 크기를 키워나가고 있으며, 매년 두 번 열리는 OpenStack Summit 행사에서는 수천 명의 개발자 및 OpenStack 운영자들이 모여서 기술적인 논의가 이루어진다.

o 현재 ICT가 클라우드 중심의 패러다임으로 이동해가는 것에 주목하면서, 대표적인 오픈 소스인 OpenStack을 채택해서 실증형 연구를 위한 테스트베드 구축 및 운영을 해오고 있다.

o Playground란 사용자들이 하고 싶은 Play를 자유롭게 할 수 있게 제공되는 공간을 상징하고 사용자들이 Playground 위에서 하는 다양한 실증 실험들은 Play로 표현할 수 있다.

o SmartX Cloud Playground는 사용자들이 원하는 실증 실험을 할 수 있도록 SmartX Box들을 유연하게 엮어 컴퓨팅, 네트워킹, 스토리지로 대표되는 IT 자원들을 알맞게 제공할 수 있는 물리적인 인프라를 의미한다. SmartX Cloud Playgroud는 하나의 공통 인프라를 OpenStack 기반의 클라우드 환경으로 구축하고, DevOps 기반 자동화된 설치/설정 소프트웨어 도구를 활용하여 각 사용자의 요구사항에 따라 동적이고 유연하게 제공되는 실증 실험 환경을 의미한다. 즉, Cloud Playground는 하나의 공유 인프라 상에 각 사용자 별로 독립적으로 제공되는 실증 실험 공간이다.

o 국내 미래인터넷 관련 연구자들을 대상으로 클라우드 기반의 미래인터넷 실증 실험 테스트베드를 제공하고자 2012년 광주과학기술원 NetCS 연구실을 중심으로 KOREN NOC(Network Operation Center), 고려대학교, 포항공과대학교, 제주대

학교를 대상으로 초 융합형 자원 박스인 융합형 Box를 배포하고 이를 KOREN 연구망으로 연결하는 OF@KOREN Playground (i.e. 테스트베드)의 구축을 시작하였다. (그림 2)

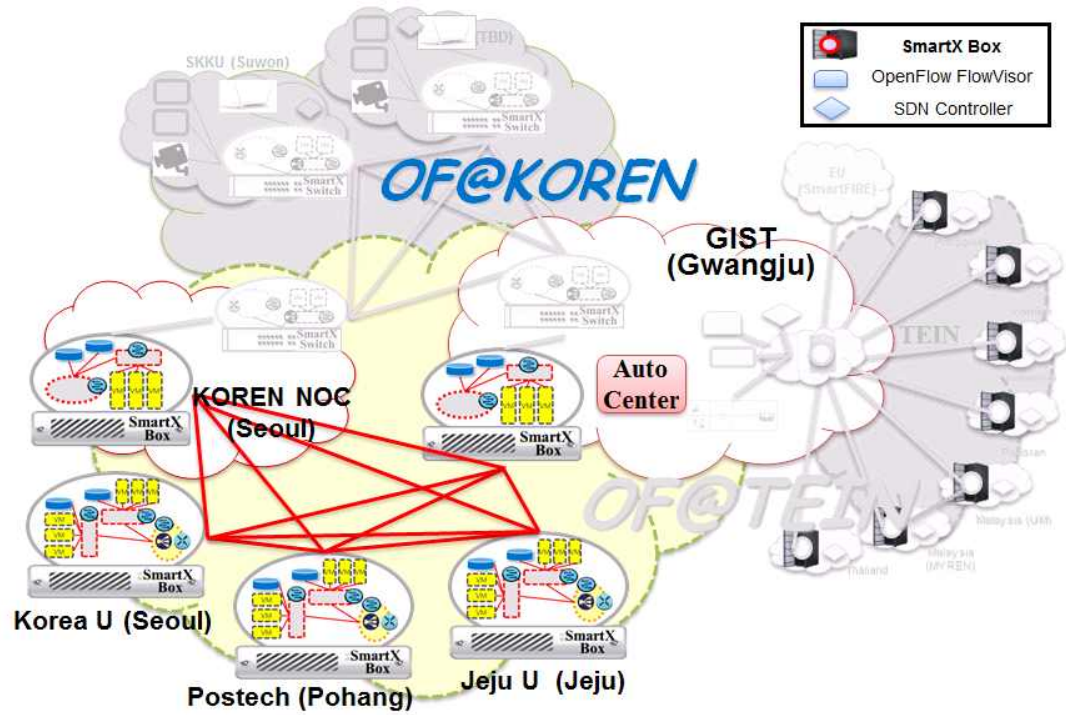

그림 3 SmartX Cloud Playground 구성도

o SmartX Cloud Playground는 그림 2에서 보는 것과 같이 지리적으로 국내 각지에 분산되어 있는 SmartX Box를 OpenStack으로 엮어 하나의 분산 OpenStack 클라우드로 구성함으로써 컴퓨팅, 네트워킹, 스토리지 자원들을 사용자들에게 제공하고 있다. 이 때 KOREN 연구망을 중심으로 SmartX Box들을 연결하여 구성된 물리적인 인프라는 Playground라고 할 수 있고, 이 인프라 상에 구성된 OpenStack은 각 사용자들에게 독립적인 Cloud 환경인 Playground를 제공한다.

1.4. SmartX Automation Framework

o SmartX Automation Framework이란 이러한 다수의 사이트의 OpenStack으로 구성된 SmartX Cloud Playground를 효율적으로 운영하고, 다양한 서비스들을 실증하기 위한 초융합형 SmartX Box 중심의 IoT-Cloud 인프라/플랫폼/서비스 통합에 대한 SmartX 자동화 소프트웨어 프레임워크를 의미한다.

o 아래 그림과 같이 크게 4가지의 도메인으로 구성되고 있으며 각 도메인에서는 자동화 수준으로 해당 도메인을 담당하기 위한 SW들로 구성되어 있으며, API 형태

로 제공해줘서 단지 SmartX Framework의 API를 활용하면 손쉽게 IoT-Cloud 인프라/플랫폼/서비스 통합을 위해 자동하 수준으로 Software-defined Infrastructure를 구성할 수 있도록 해주는 것이 SmartX Auotmation Framework의 주요한 목적이다.

o 또한 이러한 SmartX Automation Framework는 DevOps Tower라는 관제탑 요소 안에 들어가는 형태로 존재하며 신속성과 효율성을 무기로 하는 DevOps 방법론을 도입하여 인프라 전반에 걸친 자원집합의 활용도를 최적화하는 개발/운용 자동화 기술을 지원한다.

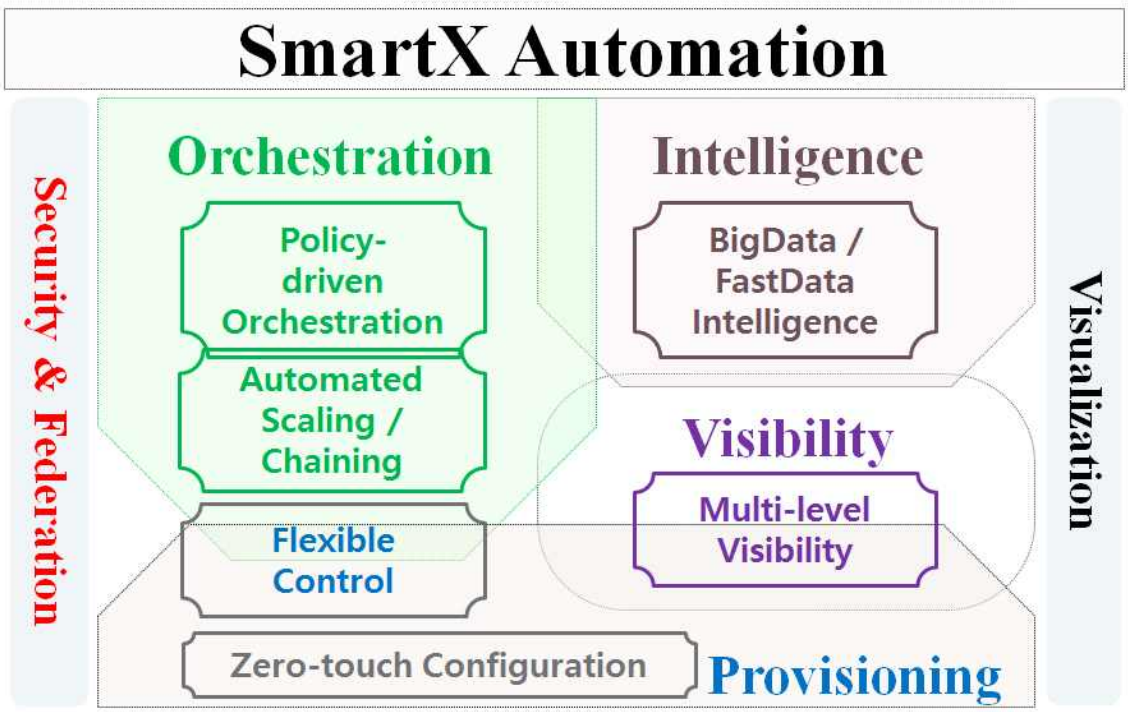

그림 4 SmartX Automation Framework의 주요 구성도

1.5. OpenStack-MultiView

o SmartXCloud Playground는 다중 사이트 환경에서 발생할 수 있는 이슈들을 다루는 것을 목적으로 하기 때문에 데이터센터처럼 하나의 사이트에 박스들이 집중되어 있는 것이 아니라 국내 각지에 분산 배치되어 있는 상태이다. 따라서 서비스들이 하나의 지역적으로 통일된 위치에 배포가 되기보다는 Function들로 나누어져 분산된 박스에 고루 분포되기 때문에 이들의 관계를 파악하거나 Function을 담을 수 있는 물리적인 박스자원의 상태파악이 중요하다. 따라서 물리적 자원 뿐

만 아니라 가상 자원 및 물리적 자원과 가상자원의 관계까지도 파악해줄 수 있는 모니터링 도구가 필요해지고 있는 상황이다.

o SmartX Cloud Playground는 기업, 연구소 차원에서 운영하는 인프라 와는 달리 그 목적이 다양한 실증 실험 환경을 제공하는 것에 있다. 따라서 급속도로 발전하는 다양한 소프트웨어와 사용자들의 요구사항에 맞춰 실증환경을 보다 유연하고 동적으로 파악할 수 있는 기능을 갖추어야 한다.

o 따라서 SmartX Cloud Playground의 특성인 지속적인 규모 확장, 다중 사이트 환경, 다양한 실증환경 제공이라는 요구사항에 맞춰 Playground를 상세히 파악하기 위해서는 Multi-level VIsibility를 구축하는 것이 반드시 필요하다.

o 따라서 위에서 언급한 SmartX Automation Framework의 도메인 중 하나인 Multi-level Visibility를 커버하기 위해서 OpenStack-MultiView라는 SW를 디자인 및 단계적으로 검증을 시작하고 있다. Multi라는 단어는 어느 관점에서 상태변화를 측정하는가에 따라서 자원 레벨, 플로우 레벨, 서비스 레벨과 같은 Multi-level을 의미하면서도 동시에 데이터의 종류에 따라 Operation Data, Application Data로 나눠질 수 있으며, 이러한 관점의 조합으로 다양한 형태의 가시성을 표현할 수 있고 이것이 MultiView라는 단어의 상징적인 의미이다.

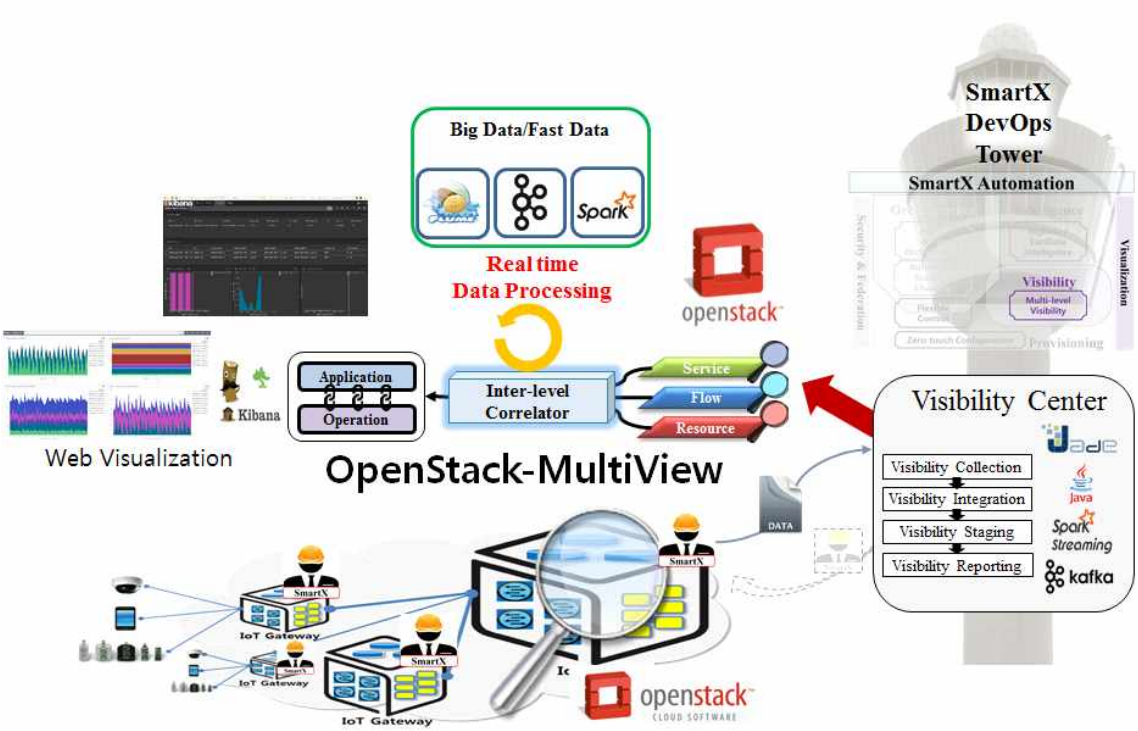

그림 5 OpenStack-MultiView의 기본적인 구도

o 위의 그림과 같은 구도로 OpenStack-MultiView는 DevOps Center 내에 위치한 SmartX Automation Framework의 Multi-level Visibility의 한 범주 내에 속하며, SmartX Cloud Playground에 Agent를 심고 Agent가 DevOps Tower에서 지시받은 행동을 수행하면서 기본적인 데이터들을 모으고, 데이터들을 따라서 위에서 언급한 SmartX Automation Framework의 도메인 중 하나인 Multi-level Visibility를 커버하기 위해서 OpenStack-MultiView라는 SW를 디자인 및 단계적으로 검증을 시작하고 있다.

o 현재 본 기술 문서에서는 OpenStack-MultiView에서 자원 레벨에 한정한 가시성 디자인 및 검증만을 진행하고 있으며, 차츰 다른 레벨을 디자인 및 검증을 끝낸 후에 하나로 통합시키면서 차츰 OpenStack-MultiView를 완성하는 단계로 진행해 나갈 것이다.

o 따라서 다음 장은 OpenStack-MultiView의 자원 레벨 가시성의 개요 및 요구사항을 정리하며, 3장에서는 자원 레벨 가시성을 위한 SmartX Agent 설계 그리고 마지막으로 4장에서는 SmartX Agent를 통한 자원 레벨 가시성을 검증해본다.

2. 자원 레벨 가시성 및 구현 핵심 전략

2.1. 자원 레벨 가시성 개요

o 자원 레벨 가시성 설계 이전부터 다수의 물리적 박스들을 연결하여 하나의 클라우드 인프라를 구축하고 이러한 클라우드 환경을 위한 모니터링 도구들은 계속적으로 개발되어 왔고 많은 도구들이 나타나고 있다. 그러나 이러한 도구들은 대부분이 클라우드의 특징이 가상자원에만 집중적으로 모니터링을 하고 있다. 예를 들면 클라우드 OS의 대표적인 오픈소스인 OpenStack [3] 에서는 Ceilometer[4]라는 프로젝트를 통해서 클라우드 자원에 대한 Metering을 제공해주고 있다. 이는 SmartX Cloud Playground에서 가상자원을 모니터링 하기 위한 요소로 쓰일 수는 있지만 가상자원과 물리자원의 관계 및 물리자원의 대한 모니터링 부분은 여전히 못 채워주고 있는 상황이다.

o 반면에 클라우드 환경에서 벗어나게 되면, 물리 자원 박스에 대해서 모니터링을 하는 도구들이 많이 제공되고 있다. 특히 리눅스 기반의 서버 박스에서 손쉽게 사용할 수 있는 도구들이 많이 나와 있는 상태이다. top이나 mpstat은 CPU나 Core에 대한 모니터링을 해주는 도구이고, 메모리까지도 어떻게 사용되고 있는지 파악이 가능하다. netstat은 인터페이스에 들어오고 나가는 패킷에 대해서 파악이 가능하다. 따라서 이러한 도구들의 조합을 통해서 손쉽게 리눅스 기반의 서버 박스에서 자원에 대한 모니터링 부분을 할 수 있다.

o 따라서 단일 물리 자원 박스에 대해서 모니터링을 하는 도구들과 클라우드 환경 상에서 클라우드 가상자원을 대상으로 하는 모니터링 도구들을 적당히 활용해서 SmartX Cloud Playground 상에서 SmartX Box와 클라우드 가상 자원을 위한 통합적인 시각의 자원 레벨 가시성 필요성이 대두되기 시작하였다.

o 자원 레벨 가시성이란 SmartX Cloud Playground를 구성하는 다중 사이트 SmartX Box 들을 대상으로 전체적인 자원 입장에서 서비스를 이루는 Function들과 물리 자원들이 어떤 관계를 이루고 있고, 이러한 상황을 동적으로 판단할 수 있는 도구들을 하나의 큰 틀 하에서 엮음으로써 운영자가 쉽게 SmartX Box 자원과 클라우드 가상자원에 대해서 어떻게 사용되고 있는지 파악을 돕고, 나아가서 이러한 기술을 바탕으로 유연조정의 시발점이 될 수 있는 하나의 틀을 제공해준다.

2.2. 자원 레벨 가시성 요구사항

o SmartX Cloud Playground를 구축하는 대규모의 SmartX Box들과 클라우드 가상 자원을 대상으로 동적관찰을 제공해야 한다는 자원 레벨 가시성의 근본적인 목적에 부합하기 위해서 아래와 같은 요구사항을 최대한 만족하도록 자원 레벨 가시성을 설계하였다.

o Cloud, BigData, Container 등 현재 IT 분야의 기술 및 소프트웨어들이 엄청난 속도로 발전됨 따라 다양한 오픈소스 도구들이 나오고 있다. Cloud OS를 대표하는 OpenStack부터 가상스위치인 OpenvSwitch, 컨테이너 플랫폼인 Container까지 다양한 오픈소스 도구들로 인해서 이제는 누구나 쉽게 클라우드 환경을 구축해보고 테스트할 수 있는 시대가 도래되었다. 이러한 상황에 대응해서 여러 가지 모니터링 도구들이 쏟아져 나오고 있는 시점에서, 이미 존재하고 있는 도구들을 쉽게 엮어서 자신들의 목적에 맞는 가시성 도구를 만들어 내는 것이 중요하다. SmartX Cloud Playground에서 중요한 사항은 클라우드에서 다루고 있는 가상 자원 뿐만 아니라 가상 자원이 담기는 물리적 박스의 자원과 가상 자원들의 관계까지도 엮어낼 수 있어야 한다. 따라서 Physical 과 Virtual (p+v) Resource를 동적 관찰에 대상으로 볼 수 있어야한다.

o 자원 레벨 가시성은 이를 사용하는 운영자가 수집된 데이터를 가시화할 수 있어야하고, 이러한 수집된 데이터는 운영자에 의해 사용되어지거나 적절한 처리가 이루어질 수 있어야 한다. 따라서 데이터의 규격은 운영자가 쉽게 조작할 수 있는 규격에 맞춰서 정의가 되어야한다. 현재 다양한 데이터베이스 도구들로부터 조작이 가능한 데이터 형식으로는 JSON 형식이 있다. 관계형 데이터베이스가 아닌 NOSQL 데이터베이스에서 자료를 저장하는 형식으로도 많이 쓰이며, 데이터를 다루기가 쉽고 처리가 빠른 대표적인 데이터 형식이다 [5].

o 마지막으로 이러한 자원 레벨 가시성은 구축하려고 하는 사용자가 손쉽게 설치할 수 있어야하며, 다룰 수 있어야 한다. 즉 다루기 쉽고 누구나 쉽게 쓸 수 있는 오픈소스를 사용하되, 잘 엮어서 하나의 통합된 도구를 만들고 도구들안에 엮여져있는 연결성을 잘 정리해서 추가적으로 엮을 수 있는 틀을 마련해야한다. 이렇게 만들어낸 자원 레벨 가시성은 각 사용자들의 인프라의 상황에 알맞게 손쉬운 조정이 가능하며 자원의 대상을 플로우나 서비스까지 확장하는데 일조할 수 있게 된다.

3. 자원 레벨 가시성 및 SmartX Agent 설계

3.1. 자원 레벨 가시성 디자인

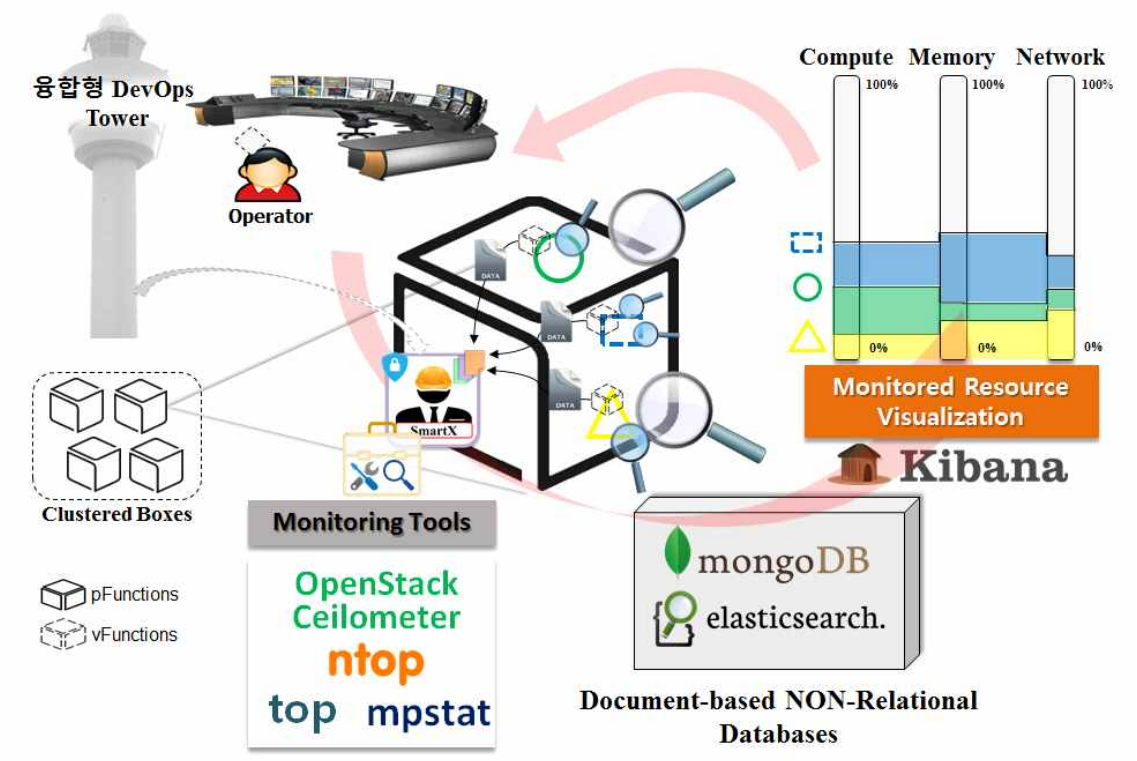

그림 6 자원 레벨 가시성의 설계도

o 자원 레벨 가시성은 위와 같은 그림으로 설계를 하였다. 그림에서 Clustered Boxes는 SmartX Box들이 OpenStack를 활용하여 SmartX Cloud Playground가 구축되는 대상 박스들을 엮어놓은 것을 말한다. 이러한 박스 내부에는 클라우드 자원을 활용한 가상 박스들이 배치될 수 있고 이러한 클라우드 가상 자원과 물리 자원 모두 동적관찰의 대상으로 보고 데이터를 수집하는 것이 목적이다.

o 목적에 부합하는 데이터 수집을 위해서 오픈도구들을 활용해 물리적 자원 모니터링, 클라우드 가상자원 모니터링, 데이터를 저장하기 위한 데이터 저장도구, 마지막으로 데이터들을 시각적으로 가시화 해주는 도구들을 선정해 하나로 엮는다. 이 때 이전 장에서 언급했던 것과 마찬가지로 데이터 형식은 JSON 형식을 따르되 누구나 쉽게 이용하고 설치 할 수 있는 오픈소스 도구들을 선정한다.

o 이렇게 엮어진 도구들을 관리해줄 수 있는 Tower 개념을 추가해서 SmartX DevOps Tower라 명명하고 여기서 SmartX Box 대상들에게 Agent를 전달하고

SmartX DevOps Tower가 Agent를 관리하고 대화할 수 있는 구조를 가지도록 한다. Agent는 각 SmartX Box내에 위치하며 위치한 Agent는 오픈소스 모니터링 도구들을 통해 물리 자원 및 클라우드 가상 자원의 데이터들을 모아서 데이터베이스에게 전달해주는 역할을 한다. 자원 데이터의 종류는 CPU, 메모리, 네트워킹을 기본적으로 하되 상황에 따라 특정 인터페이스에 해당하는 네트워킹 자원에 한정한다.

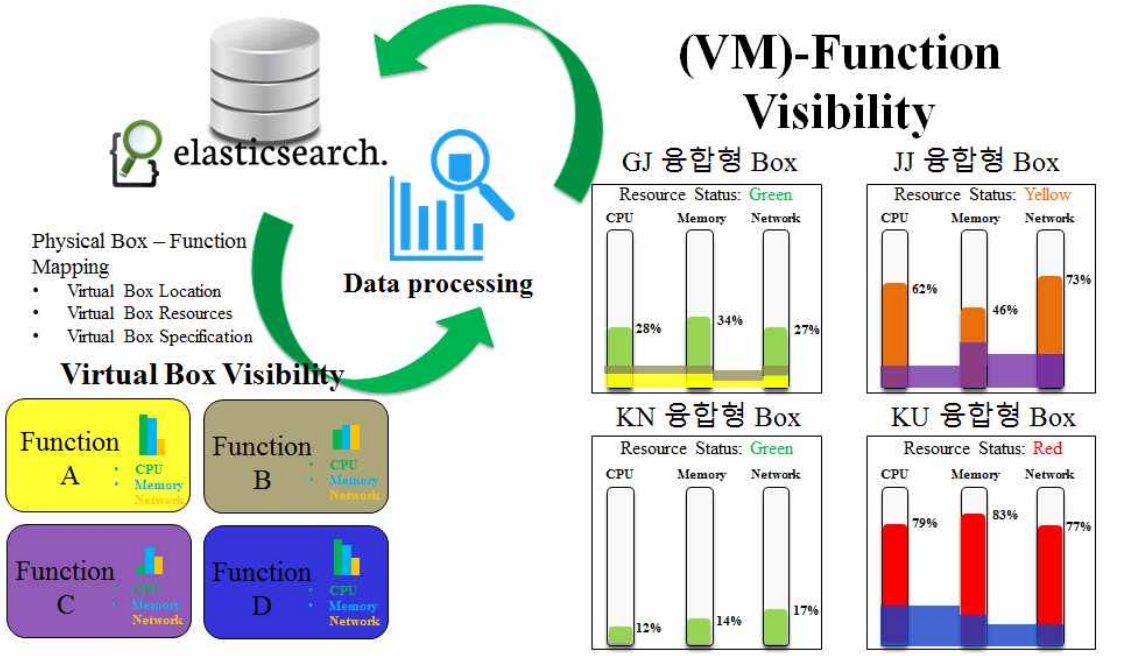

그림 7 물리자원과 가상자원과의 관계 나타내는 Function Visibility 디자인

o 물리자원과 가상자원 뿐만 아니라 서비스를 Function들의 모음으로 개념을 정의한다면 각 Function들과 물리 자원과의 관계를 나타낼 수 있어야 한다. 클라우드 환경에서는 기본적으로 가상머신 안에 Function이 위치하게 되므로 가상 Function (vFunction)에 대해서만 대상을 한정하여서 물리자원과의 관계를 나타낸다. 위 그림과 같이 각 Function들과 물리자원들과의 관계를 표시함으로써 현재 어떤 SmartX Box에 vFunction이 위치하고 있는지 한눈에 파악할 수 있도록 가시화를 한다.

3.2. 검증을 위한 SmartX Agent 소프트웨어 디자인

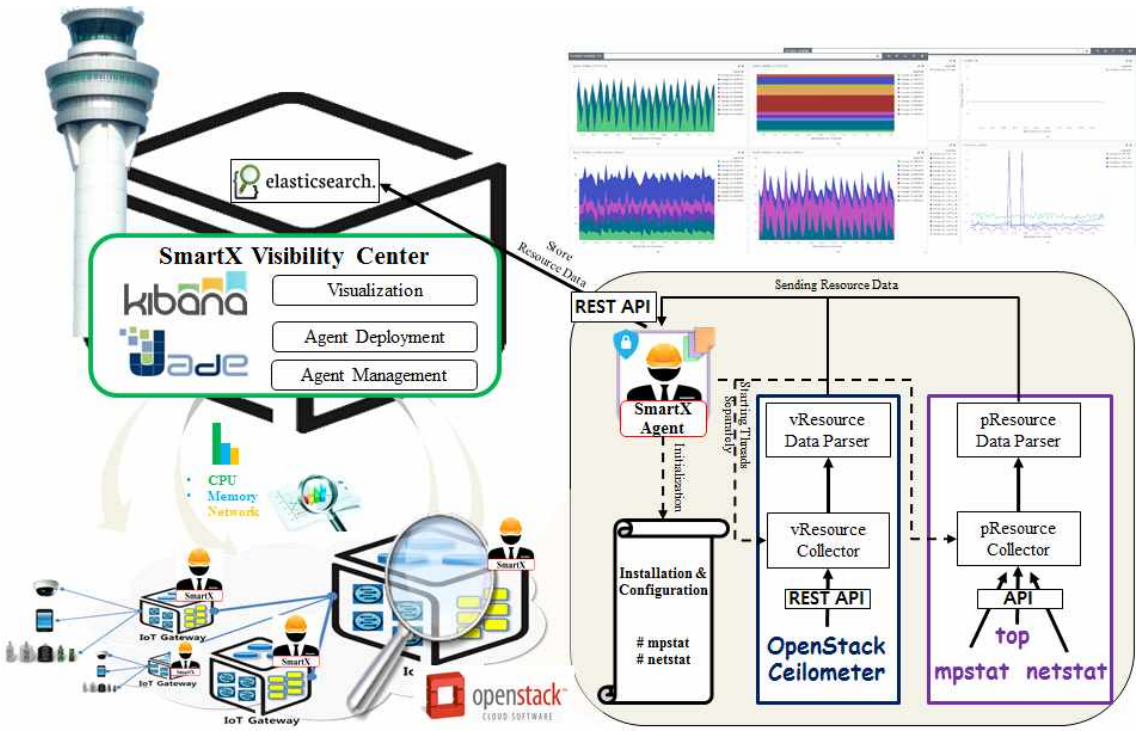

그림 8 자원 레벨 가시성을 위한 SmartX Agent 디자인

o 상기 절에서 제시한 디자인에 따라 SmartX Agent의 기능을 위해 위 그림과 같이 디자인 하였다. 크게 SmartX Automation Framework 의 SmartX Visibility 내의 사용될 소프트웨어들과 그리고 SmartX Agent 내에서 사용될 소프트웨어들을 그림과 같이 구분하였으며, 각 역할에 맞는 오픈소스 도구들을 다음과 같이 선정하고 이를 엮어서 자원 레벨 가시성을 구현하였다.

o SmartX Box의 자원에 대해서 모니터링을 하기 위한 오픈소스 도구로는 리눅스에서 많이 사용하고 있는 도구로 선정하였다. 물리 CPU를 파악하기 위해서는 mpstat라는 도구를 사용하였다. 이 도구는 CPU의 전체 사용량뿐만 아니라 각 코어마다 사용량을 개별적으로 확인할 수 있기에 CPU 자원의 사용량에 대해서는 유용하게 활용할 수 있다. 메모리의 자원 파악을 위해서는 리눅스의 대표적인 도구인 top을 활용하였다. top은 기본적으로 cpu와 프로세스사이의 관계를 나타내기 대표적인 도구이며, 손쉽게 쓸 수 있기 때문에 선택하였다. 마지막으로 네트워킹에 관해서는 ntop이라는 도구를 사용하였다. ntop은 이더넷 인터페이스에서 오고가는 패킷량을 측정할 뿐만 아니라 flow의 정보들까지 확인할 수 있어서 네트워킹을 모니터링하기에 유용한 도구이다. 또한 향후에 언급할 저장소인

Elasticsearch에 JSON 형태의 파일로 저장할 수 있는 간편한 API를 제공해주기 때문에 [6] ntop과 Elasticsearch를 같이 활용하고자 하는 경우에는 손쉽게 이용할 수 있다. 이렇게 활용되는 도구들은 모두 API를 활용해 데이터를 출력하고 출력된 데이터 중에서 알맞은 데이터 형식으로 파싱한 후 JSON 단위로 정보를 생성해낸다.

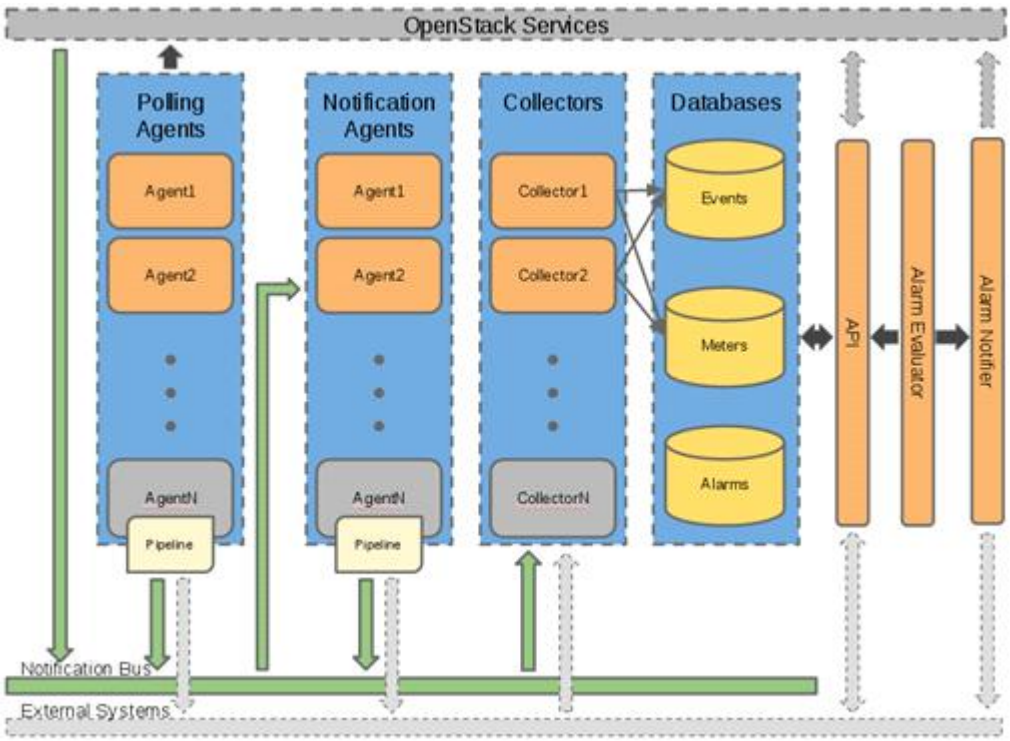

그림 9 OpenStack Ceilometer의 아키텍처

o 클라우드 가상 자원에 대해서 모니터링을 하기 위한 도구로는 OpenStack에서 제공하고 있는 Ceilometer를 활용하기로 했다. Ceilometer는 위 그림과 같은 구조로 디자인 되어있으며 OpenStack에서 과금 제도를 위해 존재하고 있는 프로젝트로써 크게 1. Metering 2. Rating 3. Billing 과 같은 단계로 구성되고 있다. 하지만 아직까지는 성숙되고 있는 프로젝트 중에 하나로써 1번의 Metering 부분에만 초점이 맞춰져 있다. 기본적인 속성은 가상머신을 제어하는 Hypervisor를 통해서 가상자원들의 정보를 주기적인 단위로 (보통이 10분) 모으며, 이렇게 모아진 데이터들은 데이터베이스에 저장되는 구조로 되어있다. Ceilometer API를 통해서 저장된 데이터를 가져올 수 있으며 이렇게 가져온 데이터들 중에 필요한 정보를 뽑아서 파싱한 후 알맞은 JSON 파일로 데이터를 만들어서 저장을 한다.

o 여러 오픈소스 모니터링 도구들을 활용하여 모아진 데이터들은 JSON 형태로 저장이 되는데 로그 기반의 데이터들을 저장하고 검색하는데 사용되는

Elasticsearch [7] 를 활용하여 JSON 데이터를 저장하였다. 데이터를 저장하기 위해서는 Curl 메소드를 활용한 간편한 방식으로 데이터를 저장할 수 있으며, 관계형 데이터베이스에서 칼럼(Column)에 해당되는 Field가 유동적이기 때문에 가상머신이 자주 생성되고 삭제되는 클라우드 환경을 고려할 때 알맞은 선택이라고 할 수 있다.

관계형 데이터베이스	엘라스틱서치
데이터베이스(Database)	인덱스(Index)
테이블(Table)	타입(Type)
로우(Row)	도큐먼트(Document)
컬럼(Column)	필드(Field)
스키마(Schema)	맵핑(Mapping)

그림 10 관계형 데이터베이스와 Elasticsearch의 비교도

o 마지막으로 가시화를 구현하기 위해서 Kibana [8] 라는 가시화 도구를 사용하였다. Kibana는 Elasticsearch와 연계하여 로그기반의 데이터들을 모으고 가시화하고 분석하는데 유용하게 쓰이는 도구이다. Elasticsearch에 들어가는 데이터들을 여러 형태의 그래프로 추출해낼 수 있으며, 약간이지만 추출된 정보를 통한 분석 기능도 제공하기 때문에 유용한 도구이다.

3.3. SmartX Agent 동작과정 설명

o SmartX DevOps Tower에서 SmartX Box에 뿌려진 Agent들이 기본적으로 SmartX Box 내에서 동작을 하면서 데이터를 모으고 저장소에 전달하는 역할을 한다. SmartX DevOps Tower에서부터 전달되는 SmartX Agent의 기본적인 수행 과정은 다음 아래 그림과 같다.

o SmartX DevOps Tower에 위치한 Visibility Center에서는 SmartX Box에 전달될 SmartX Agent들을 원격에서 관리 할 수 있는 시스템이 존재하고 있다. Jade (Java Agent Development Framework)는 이를 위한 도구로써 원격에서 Agent들을 모니터하고 심어진 Agent들이 동작을 잘 수행하는지 관리 하게 된다.

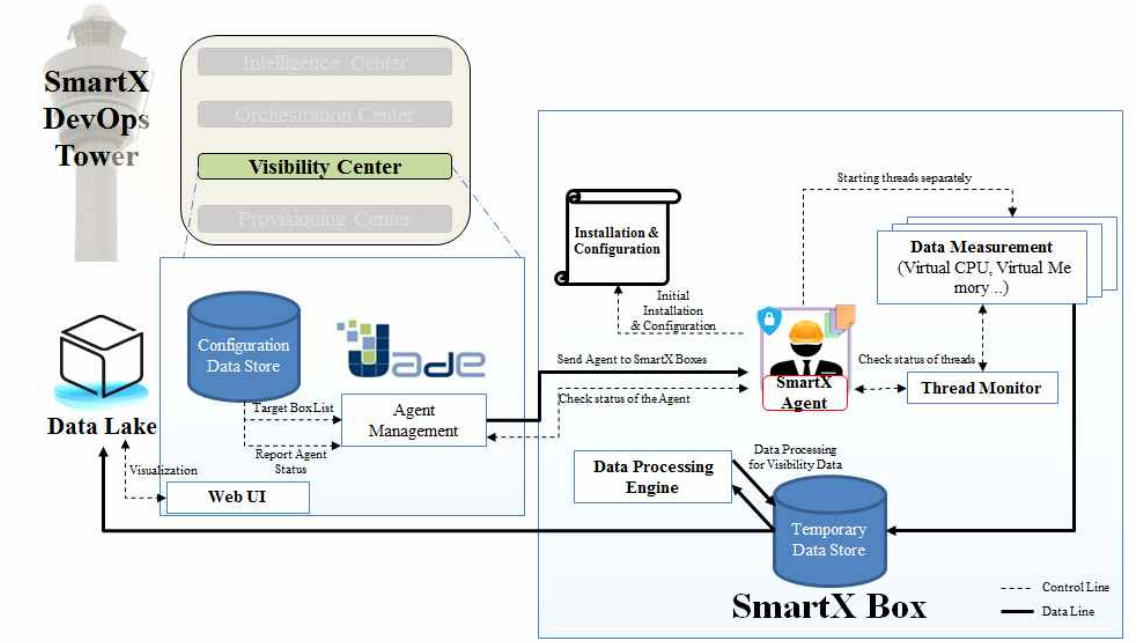

그림 11 SmartX DevOps Tower와 SmartX Agent의 동작과정

o SmartX DevOps Tower에서 SmartX Box에 뿌려진 Agent들은 기본적으로 SmartX Box 내에서 동작을 하면서 데이터를 모으고 저장소에 전달하는 역할을 한다. 이렇게 데이터를 전달하기 위해서 JSON 형태의 파일을 만들어야하는데, 이 작업도 Agent 단에서 이루어진다. 처음 전달된 Agent는 SmartX Box에 필요로 하는 모니터링 도구들을 설치하고 설정하는 작업을 거치게 된다. 이 작업이 끝난 후 각 모니터링 도구들의 API를 활용하여 데이터들을 측정한다. 이렇게 측정된 데이터는 임시 저장소에서 파싱을 통해 알맞은 JSON 형태로 바꾸고 SmartX DevOps Tower에 위치한 최종 저장소인 Elasticsearch에 전달한다. 그리고 중간에 물리자원과 가상 클라우드 자원과의 관계를 파악하기 위해 가상머신의 위치, 가상박스의 스펙 등을 활용해 어느 정도 처리를 거친 후 또 다른 형태의 JSON 형태의 파일을 만들고 최종 저장소로 저장이 된다. 최종 저장소에 저장이 되는 데이터베이스(인덱스)는 크게 3가지로 나뉘며 타임 스템프가 찍혀서 저장이 되는 Time Series 기반으로 저장을 하게 된다. 이렇게 저장이 된 데이터들은 Kibana의 웹 주소로 들어가서 확인할 수 있으며, 들어가서 자원들의 상태에 따른 그래프들을 생성해서 SmartX Box와 클라우드 가상 자원에 대해서 동적관찰을 가능하게 한다.

3.4. 자원 레벨 가시성 SW 설정 및 실행방법

o 본 절에서는 각 SmartX Box에 JADE 4.4 버전이 설치되어있다는 가정 하에

SmartX Agent를 실행하는 과정을 나타낸다. 또한 OS는 Ubuntu 14.04.3 LTS에 맞춰서 설명을 진행할 것이며, 본 기술문서는 RedHat 기반의 OS에 대해서는 아직까지 지원하지 않는 것을 참고하기 바란다. 기본적으로 스크립트들은 sudo 권한을 요구하므로, 편의를 위해서 다음 커맨드를 입력해서 sudors 파일에 권한을 등록하는 것을 추천한다.

$ sudo -c 'echo "<계정명> ALL=(ALL:ALL) NOPASSWD: ALL">>/etc/sudoers'

o SmartX DevOps Tower는 Elasticsearch와 Kibana를 설치해 Agent가 모은 데이터들을 저장하고 가시화 하는 역할을 한다. 명령창에서 Visibility_Center.sh를 실행을 하게 되면 해당 융합형 Box에 자동으로 위 두 도구들을 설치하고 설정해준다.

$ bash ./<설치 경로>/Visibility_Center.sh

o 위 작업이 끝나면 SmartX Agent를 관리할 JADE의 메인 관리시스템을 DevOps Center에서 부팅시켜야 한다. 다음과 같은 명령어로 JADE를 실행 시킨다.

$ java –cp /<JADE 경로>/lib/jade.jar jade.Boot -name <platform name> -host <IP>

o platform Name은 관리하게 되는 JADE 시스템의 이름을 의미하며 IP 부분은 DevOps Tower가 위치한 곳의 IP를 명세하면 된다.

o JADE Agent 메인 관리 시스템이 동작하게 되면 각 SmartX Box에 SmartX Agent를 전달 및 실행해야 한다.

o SmartX Agent는 기본적으로 두 가지 유형이 있으며, 각 SmartX Box가 OpenStack에서 어떤 노드 역할을 하느냐에 따라서 다르게 배치해서 사용하게 된다. OpenStack의 Compute 노드용 SmartX Agent는 SmartX Box 물리 자원에 대해서 데이터를 수집하고 전달하는 SmartX Agent이며, OpenStack의 Controller 노드용 SmartX Agent는 클라우드 가상자원 뿐만 아니라 가상자원과 물리자원의 관계까지 포함해서 데이터를 수집하고 전달하는 역할을 한다.

o SmartX Agent의 위치는 SmartX Box 내에 어디에 있어도 상관이 없으며, DevOps Tower에서 Compute용 Agent를 전달하는 방법은 다음과 같다.

$ bash ./<설치 경로>/Deploy_Compute_Agents.sh <IP> <Box_Name>

<Directory> <VM_network_Interface> <External_Network_Interface>

o 위 스크립트 파일에 들어가는 IP는 SmartX Agent가 Deploy 될 IP를 말하며 Directory는 어떤 디렉토리에 설치를 할 것인지를 명세 하는 부분이다. 그리고 VM network Interface와 External Network Interface 부분은 해당 Box 내의 오픈스택를 구성할 때 맡았던 역할의 인터페이스 이름을 각각 명세하면 된다. 마지막으로 Box Name은 해당 Box의 이름을 입력하도록 한다.

o 마지막으로 SmartX DevOps Tower에서 Controller용 Agent를 전달하는 방법은 다음과 같다.

$ bash ./<설치 경로>/Deploy_Controller_Agents.sh <IP> <Box_Name> <Directory> <OpenStack_Env>

o IP, Box Name 및 Directory는 위의 부분과 동일하나 OpenStack Env는 OpenStack API를 접근하기 위한 관리자용 admin-openrc.sh의 파일 위치를 명세하면 된다.

o 실행이 되기 시작하면 주기적으로 데이터를 모으고 전달하는 과정이 반복적으로 일어나므로 백그라운드 환경에서 실행을 하는 것을 추천한다. 이렇게 실행이 되면 주기적으로 SmartX DevOps Tower에 데이터가 전달되게 되고 이렇게 전달된 데이터는 DevOps Tower에 접근할 수 있는 주소 IP 뒤에 포트번호 5602를 붙여서 Kibana 웹 URL에 접근해 볼 수 있다.

http://DevOps Tower가 위치한 IP:5602/

o 웹 URL에 접근하면 세 종류의 Index를 확인할 수 있으며 인덱스 안에 있는 다양한 필드들의 조합으로 그래프를 그려내 대시보드 탭을 통해서 가시화를 할 수 있다.

4. SmartX Agent를 통한 자원 레벨 가시성 검증

4.1. SmartX Cloud Playground 상의 자원 레벨 가시성 활용

o 본 장에서는 이전 장에서 설계하고 구현하였던 자원 레벨 가시성을 SmartX Cloud Playground 상에서 직접 검증해봄으로써 실제적으로 SmartX Box들과 클라우드 가상자원에 대해 동적파악을 하고 이를 쉽게 가시화 할 수 있는 것을 보여준다.

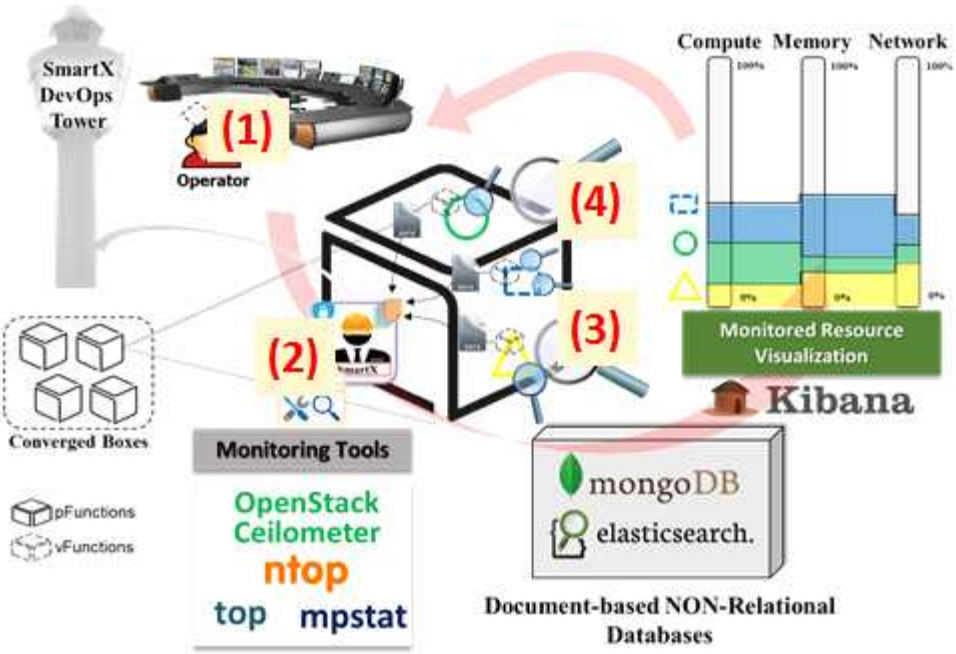

그림 12 자원 레벨 가시성 검증 시나리오

o 검증 시나리오는 위 그림과 같다. 현재 SmartX Cloud Playground로 활용되고 있는 SmartX Box 5개에 대해서 SmartX Agent를 심고 Agent로부터 얻어오는 데이터들을 Kibana 웹 URL에서 확인하는 과정을 거친다.

o 아래 그림은 Kibana의 웹 URL에 접속한 화면을 나타내고 있다. 이전 장에서 언급했던 것처럼 관계형 데이터베이스에서 데이터베이스에 해당하는 Index는 총 3개로 box_visibility, function_visibility, vbox_visibility로 구성되어 있다. 인덱스를 누르면 해당 인덱스에서 제공되는 필드들을 볼 수가 있다. Discover 탭에서는 현재 데이터들이 어떤 주기로 들어오고 있는지를 확인할 수 있으며, 대쉬보드에 데이터들을 표기하는 주기를 원하는 시간 간격으로 조정해서 나타낼 수 있다.

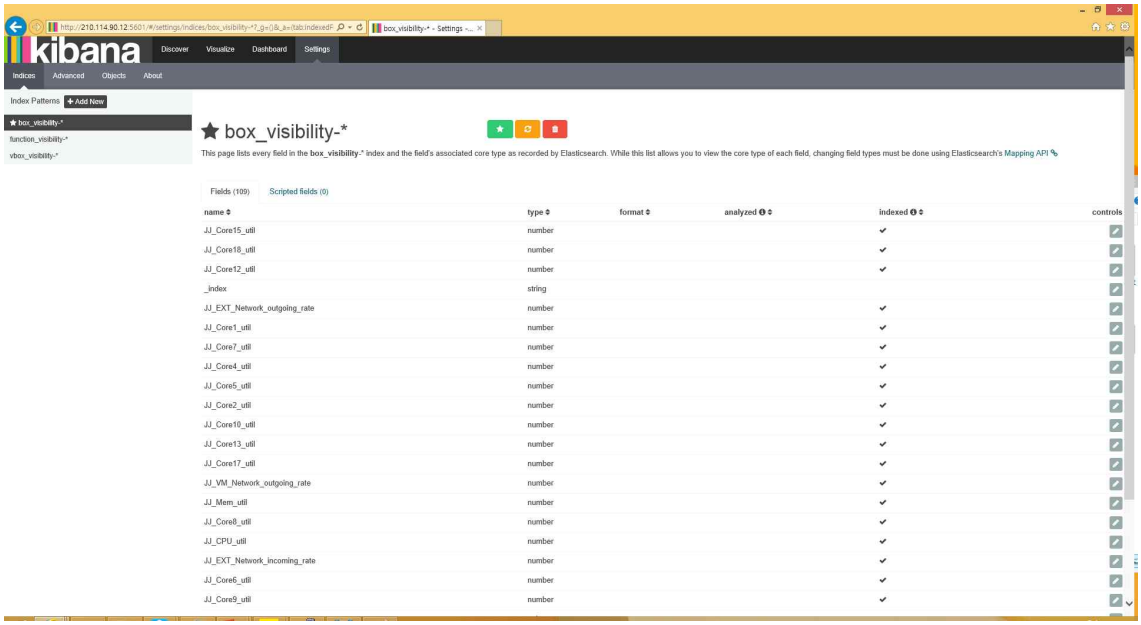

그림 13 Kibana 웹 URL에서 Index 확인 장면

o Visualize 탭에서는 보고 싶은 필드들을 선정해서 그래프를 그려주는 기능을 제공한다. 아래 그림은 특정 융합형 Box의 물리적 자원에 대해서 CPU, Core, 메모리, 네트워크 탭으로 구성된 그래프들을 가시화한 그림을 나타낸다. X 축은 시간이며 Y축은 해당 자원에 대한 사용률을 나타내고 있다. 본 그래프는 Line 그래프로 그렸으며 타임 주기를 변경함에 따라 다른 그래프들을 그려낼 수 있다.

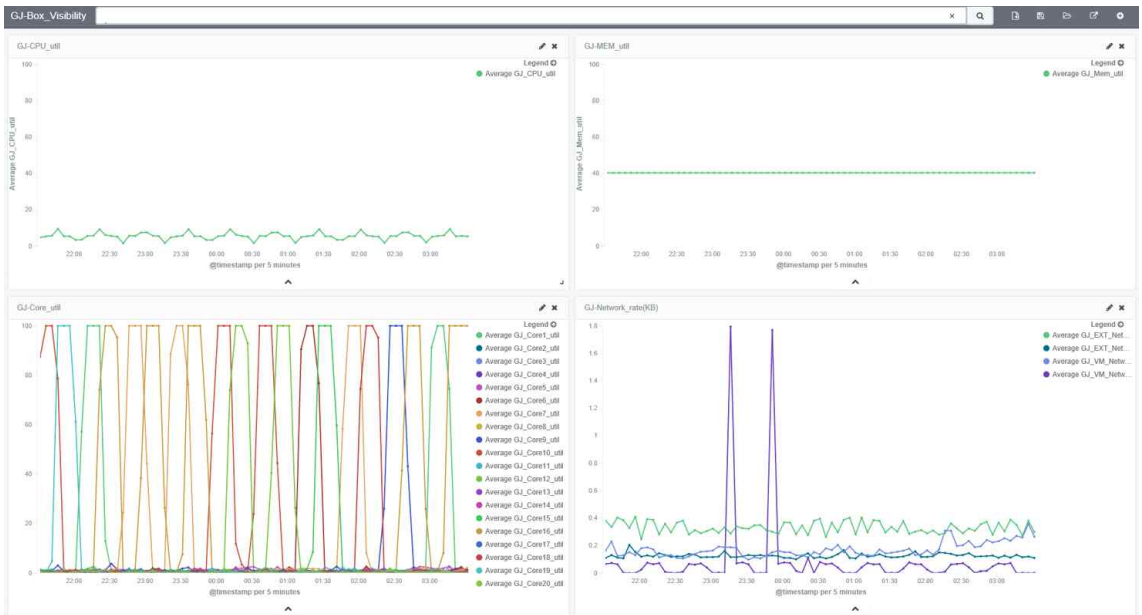

그림 14 광주 SmartX Box에 대한 물리 자원 레벨 가시성

o 다음 그림에서는 클라우드 가상자원에 대한 Resource Visibility를 나타내고 있는 그림이다. 광주 융합형 Box에 생성된 가상머신의 CPU, Memory 및 들어오고 나가는 패킷의 전송량을 보여주고 있다. 위 그림과 같이 Line 그래프로 그렸으며 시간에 따라 사용량이 변화하는 것을 관찰 할 수 있다.

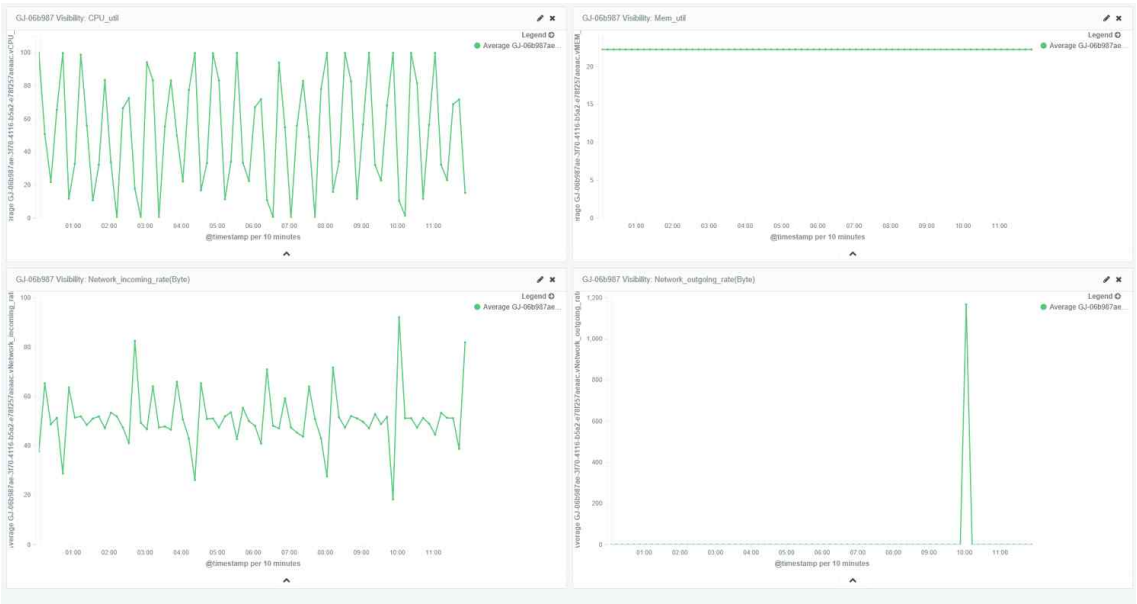

그림 15 SmartX Box내에 위치한 클라우드 가상 자원에 대한 자원 레벨 가시성

o 마지막으로 다음 그림에서는 융합형 Box와 클라우드 가상자원에 대한 관계를 Resource Visibility로 나타내고 있는 그림이다. 대상 융합형 Box는 위와 같이 광주를 사용하였으며, 위의 두 그래프와는 달리 Area Chart를 사용하여 가시화하였다. 각 차트에서 보이는 색깔은 광주 융합형 Box 내에 위치한 가상머신들을 나타내고 있으며 각 가상머신들이 융합형 Box에서 얼마나 자원을 차지하고 사용하고 있는지를 동적으로 한 눈에 파악할 수 있다. 각 차트 위에서 오른쪽 순서대로 CPU, Memory, 네트워크 패킷 유입량, 네트워크 패킷 유출량을 나타내고 있다. 아래 그림을 해석하면 보라색깔이 나타내고 있는 가상머신은 네트워크 활동이 많은 가상머신이라고 판단할 수 있으며 녹색색깔이 나타내고 있는 가상머신은 CPU 컴퓨팅 자원을 많이 사용하고 있는 가상머신임을 확인할 수 있다. 이러한 정보들을 통해서 융합형 Box와 클라우드 가상 자원에 대해 동적으로 관찰할 수 있는 기반을 마련하고 있으며, 이러한 동적파악 능력을 활용해 융합형 Cloud Playground를 효율적으로 운영할 수 있도록 유연 조정의 기능을 위해 사용될 수 있다. 아직은 그 단계까지 가지는 못했지만 위 과정을 통해서 융합형 Box 자원 인프라로 구성된 융합형 Cloud Playground를 동적으로 파악할 수 있는 Resource Visibility를 구축하였음을 검증할 수 있다.

그림 16 SmartX Box와 클라우드 가상자원의 관계를 나타낸 자원 레벨 가시성

References

[1] J. Kim et. al., "OF@TEIN: An OpenFlow-enabled SDN testbed over international SmartX Rack sites," in *Proc. APAN –Networking Research Workshop*, Aug, 2013.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica et al., "A view of cloud computing," Communications of the ACM, vol. 53, no. 4, pp. 50–58, 2010.

[3] OpenStack, http://openstack.org

[4] Ceilometer, http://ceilometer.org

[5] Z. U. Haq, G. F. Khan, and T. Hussain, "A comprehensive analysis of XML and JSON web technologies," in Proc. New Developments in Circuits, Systems, Signal Processing, Communications and Computers (CSSCC), March 2015.

[6] Exploring your traffic using ntopng with ElasticSearch+Kibana http://www.ntop.org/ntopng/exploring-your-traffic-using-ntopng-with-elasticsearchkibana/

[7] Elasticsearch, https://www.elastic.co

[8] Kibana, https://www.elastic.co/products/kibana

K-ONE 기술 문서

- 이 문서의 기술적인 내용은 프로젝트의 진행과 함께 별도의 예고 없이 변경될 수 있습니다.

- 본 문서와 관련된 문의 사항은 아래의 정보를 참조하시길 바랍니다.
  (Homepage: http://opennetworking.kr/projects/k-one-collaboration-project/wiki, E-mail: k1@opennetworking.kr)

작성기관: K-ONE Consortium
작성년월: 2016/05