학사학위 청구논문

NFVI 및 VNF 통합 매니지먼트 시스템

(Integrated Management System for NFVI and VNF)

2018년 11월 28일

숭실대학교 IT대학

전자정보공학부 IT융합

김 재 웅

학사학위 청구논문

NFVI 및 VNF 통합 매니지먼트 시스템

(Integrated Management System for NFVI and VNF)

지도교수 : 김 영 한

이 논문을 학사학위 논문으로 제출함

2018 년 11 월 28 일

숭실대학교 IT대학

전자정보공학부 IT융합

김 재 웅

김재웅의 학사학위 논문을 인준함

심사위원장 문 용 (인)

심 사 위 원 김 영 한 (인)

2018 년 11 월 28 일

숭실대학교 IT대학

목 차

표 및 그림 목차

국문초록

NFVI 및 VNF 통합 매니지먼트 시스템

정보통신전자공학부 김 재 웅
지도교수 김 영 한

클라우드 기반 인프라 시스템은 기업, 통신사업자, 포탈 등 다양한 분야의 기반 인프라로 사용되고 있다. 본 논문에서는 대표적인 클라우드 운영체계인 오픈스택환경에서 발생할 수 있는 다양한 장애원인을 모니터링하고 이들 장애간의 상관관계를 분석하여 장애원인을 찾는데 사용될 수 있는 장애분석기능을 집적하여 안정적인 클라우드 시스템의 운영기반을 구축하고 동작 등을 확인하였다. 장애모니터링 및 장애원인 분석기능 등은 오픈코드프로젝트인 Zabbix 및 Vitrage를 사용하였으며 Cloud OS인 Openstack을 활용하여 VIM을 포함한 기본 환경을 구축하고 각 기능을 집적 System UI를 통하여 통합적인 장애관리기능이 동작되도록 하였다.

특수한 하드웨어와 소프트웨어로 통신장비가 서비스되는 형태가 아닌 x86기반의 범용서버에 소프트웨어형태로 서비스되는 NFV환경에서 관제, 운용하는 사람의 필요한 기능들만 집약적으로 관리할 수 있다는 장점이 있어 5G의 상용화를 위해서는 필수적인 관리체계라고 사료된다.

I. 서 론

4차 산업혁명의 도래로 Telco들은 큰 위기에 빠지게 되었다. 먼저 AI, IoT, Cloud, Bigdata, Mobile 기술이 발달되면서 네트워크의 트래픽이 기하급수적으로 증가하게 되면서 ISP(Internet Service Provider)사업자인 Telco들에게는 이를 해결하기 위한 높은 수준의 CAPEX, OPEX가 요구된다.

또한, 비즈니스 패러다임 형태가 변화되었다. 기존의 SI(System Integration) 형태의 비즈니스가 아니라 오픈 애플리케이션 마켓 형태로 변화하면서 기존 Telco의 Revenue를 OTT(On The Top)사업자들이 가져가고 있다.

마지막으로, Vendor들의 통신장비에 종속되어있어서 진화된 아이디어가 진화된 서비스로 이루어지는 TTM(Time to Market)과정이 원활하지 않다.

이를 해결하기 위한 돌파구로 SDN/NFV(Software Define Network/Network Function Virtualization)기술이 연구개발 및 상용화 되어 Programmable 한 Open Networking이 실현되고 있다. 본 과제는 NFV 환경에서의 리소스와 VNF에 대해 차별화 된 고가용성 기능을 제공하여, 장애 원인 탐침 및 시스템 진단, 벤치마킹, 모니터링 기능을 포함하는 매니지먼트 서비스를 제공하고자한다. 결과적으로 5G상용화를 앞두고 있는 이 시점에서, 통신장비들을 한눈에 보고 제어하는 관제, 통신장비들을 고효율, 저비용 측면에서 더 높은 Performance를 낼 수 있게 설치하는 구축, 그렇게 구축된 통신장비들을 장애없이 서비스하는 운용측면에서 기여하고자 한다.

서론에 이어 본 논문은 2장에서는 NFV와 NFV Framework의 개념과 구성요소들에 대하여 간략히 기술하고, 3 장에서는 NFV환경의 VIM을 관리하는 Openstack 오픈소스에 대해서 알아보고, 4장과 5장에서는 장애모니터링 및 장애원인 분석기능 오픈소스인 Zabbix와 Vitrage의 개념과 활용방안을 6장에서는 모의실험결과 및 토의를 마지막으로 7장에서는 결론을 맺도록 한다.

II. NFV (Network function Virtualization) 기초 이론

NFV는 미들박스와 같은 네트워크 서비스 장비 내의 네트워크 기능들을 하드웨어 전용 장비로부터 고성능 범용 서버 등으로 분리시켜 소프트웨어적으로 유연하게 제어와 관리가 가능하도록 가상화시키는 기술을 말한다. 실제로 NFV를 구현하는 다양한 방법이 있으며, 이를 표준화하기 위해 유럽 표준화 기구인 ETSI에서는 Telco중심으로 2012년 말, NFV ISG를 구성해 활발한 표준 개발 작업을 현재 진행하고 있다.

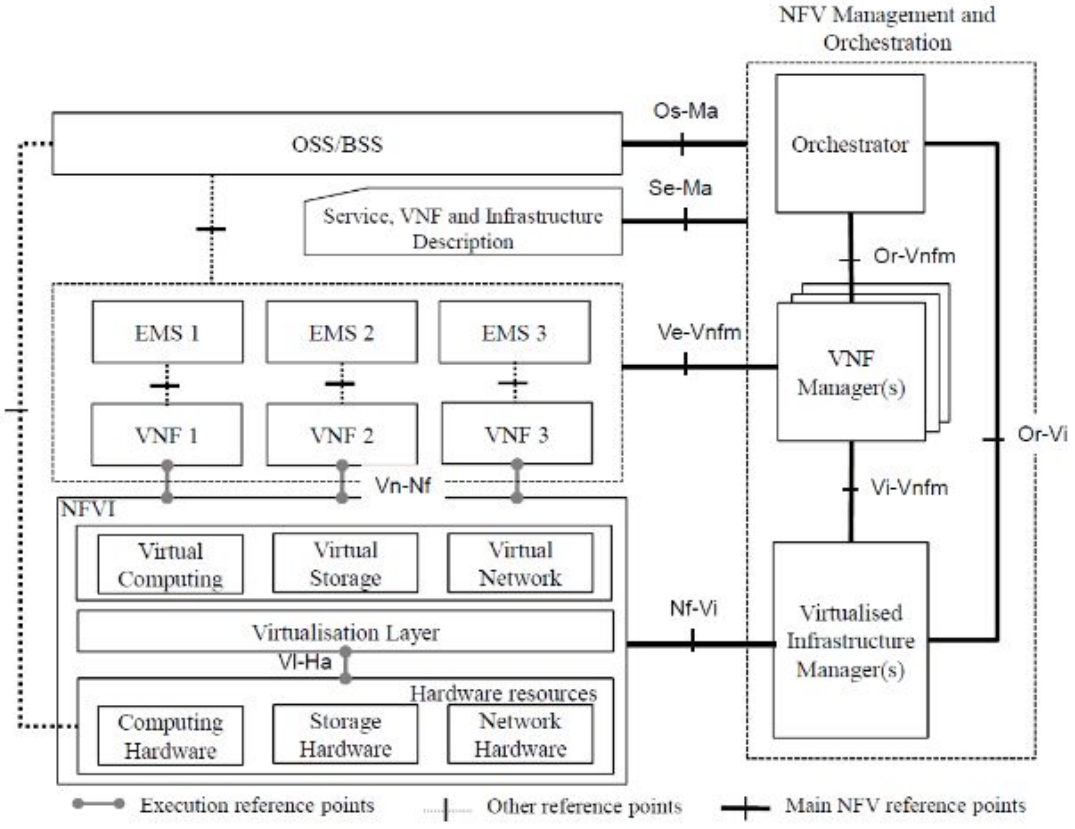


그림 1 : NFV reference architectural framework(ETSI GS NFV 002)

그림 1은 ETSI GS NFV 002에 기술된 NFV프레임워크 표준으로 가상화를 위한 가장 기본적인 자원에 해당하는 Infrastructure에서부터 최상위에 있는 운용 비즈니스 관리 지원 시스템에 이르기까지 NFV를 통한 Network Service를 구성, 제어, 관리하기 위한 통합 구조를 말한다.

II-1. EMS (Element Management System)

EM은 VNF 혹은 PNF에 대한 관리 기능을 수행하며, VNF의 경우에 VNFM과 연동해 VNF 인스턴스의 라이프사이클 오퍼레이션을 위한 가상 인프라 자원관리에 간접적으로 참여하기도 한다. 일반적으로 VNF 애플리케이션이나 PNF의 FCAPS(Fault, Configuration, Accounting, Performance Security)를 관리한다.

II-2. MANO (Management and Orchestrator)

MANO에서는 VIM을 통해서 인프라 자원을 관리하는 체계와 그위에 VNF를 관리하는 체계, 그리고 이를 통합하는 오케스트레이터가 독립적인 체계로 구성.

II-2-1. VIM (Virtual Infrastructure Manager)

NFV를 위한 NFVI를 관리하며, Computing, Network, Storage Resource 등을 통합 관리하고, 상위 인터페이스로는 가상 자원을 관리하며, 하위 인터페이스로는 하드웨어나 네트워크 컨트롤러 같은 다양한 인터페이스를 지원한다.

II-2-2 VNFM (Virtual Network Function Manager)

VNF 인스턴스의 라이프사이클을 관리가 주요 기능이며, 스케일링 및 인스턴스 관련 가상 자원 관리, 복구, 종료, 변경 등을 수행한다.

II-2-3 NFVO (Network Function Virtualiztion Orchestrator)

오케스트레이터는 상위 서비스 관점에서 네트워크 서비스의 라이프사이클 관리와 네트워크 서비스 구송요소 관리등을 주관하는 서비스 오케스트레이션 기능과, 네트워크 서비스를 구성하는 구성 요소들의 자원 측면에서 네트워크 서비스에 필요한 자원의 관리와 실행을 수행하는 자원 오케스트레이션 기능을 말한다.

III. 클라우드 컴퓨팅 및 Openstack 기초 이론

클라우드 컴퓨팅은 정보가 인터넷 상의 서버에 영구적으로 저장되고, 데스크톱, 태블릿컴퓨터, IT 기기 등과 같은 클라이언트에는 일시적으로 보관되는 컴퓨터 환경을 뜻한다. 즉, 구름(Cloud)과 같이 무형의 형태로 존재하며 하드웨어나 소프트웨어 등 컴퓨터 자원을 자신이 필요한 만큼 쓰고 이에 대한 사용요금을 지급하는 방식의 컴퓨팅 서비스로, 서로 다른 물리적인 위치에 존재하는 컴퓨팅 자원을 가상화(Virtualization) 기술로 통합해 제공하는 기술을 말한다.

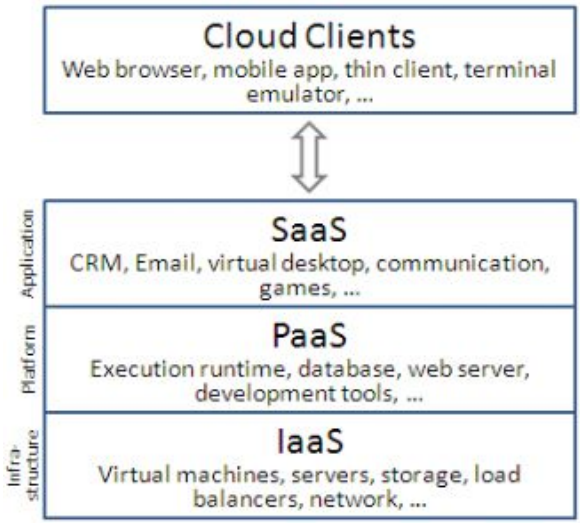


그림 2 : 클라우드 컴퓨팅 서비스 모델

클라우드 컴퓨팅 제공자들은 각기 다른 모델에 따라 자신들의 서비스를 제공하며, 이 모델의 세가지 NIST 표준 모델은 서비스로서의 인프라스트럭처(IaaS), 서비스로서의 플랫폼(PaaS), 서비스로서의 소프트웨어(SaaS)가 있다.

1. 서비스로서의 인프라스트럭처(Infrastructure as a Service, IaaS)

서비스로서의 인프라스트럭처는 서버, 스토리지, 네트워크를 가상화 환경으로 만들어, 필요에 따라 인프라 자원을 사용할 수 있게 서비스를 제공하는 형태이다.

본 논문에서는 IaaS형태의 클라우드 오픈소스 프로젝트인 Openstack을 기반으로 한다.

2. 서비스로서의 플랫폼(Platform as a Service, PaaS)

서비스로서의 플랫폼은 앱을 개발하거나 구현할 때, 관련 인프라를 만들고

유지·보수하는 과정 없이 애플리케이션을 개발, 실행, 관리할 수 있게 하는 플랫폼을 제공한다.

3. 서비스로서의 소프트웨어(Software as a Service, SaaS)

서비스로서의 소프트웨어는 소프트웨어 및 관련 데이터는 중앙에 호스팅되고 사용자는 웹 브라우저 등의 클라이언트를 통해 접속하는 형태의 소프트웨어를 제공한다.

III-1. 오픈스택(Openstack)

OpenStack은 데이터 센터 전반에서 대규모 컴퓨팅, 스토리지 및 네트워킹 리소스 풀을 제어하는 IaaS(Infrastructure as a Service)형태의 클라우드 운영 체제로, 사용자가 웹 인터페이스를 통해 리소스를 공급할 수 있는 동시에 관리자가 제어할 수 있는 대시보드를 통해 관리된다.

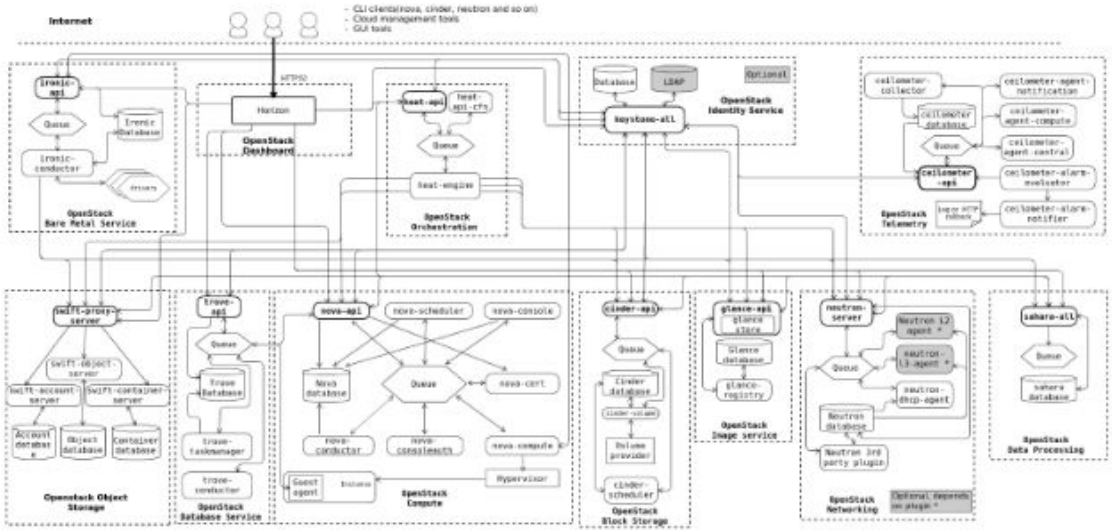


그림 3 : Openstack logical architecture

Openstack은 Openstack 서비스라는 이름의 몇 가지 독립적인 부분으로 구성되며 이를 각각 하나의 프로젝트라 한다. 각 프로젝트는 개별적인 기능을 수행하며 각각 최소한 하나의 API 프로세스가 존재한다. API 프로세스는 API 요청을 수신하고 사전 처리하여 서비스의 다른 부분으로 전달한다. [그림 2]는 가장 일반적인 오픈스택 아키텍처이며 프로젝트간의 작동을 나타낸다.

IV. Zabbix 기초 이론

IV-1. Zabbix 소개

IV-1-1. Zabbix

Zabbix는 엔터프라이즈 급 오픈소스 분산 모니터링 솔루션이다. Alexei Vladishev에 의해 만들어졌으며 현재 Zabbix SIA가 적극적으로 개발하고 있다. Zabbix는 네트워크의 수많은 매개 변수와 서버의 상태 및 무결성을 모니터링하는 소프트웨어이다. Zabbix는 클라우드 인프라를 운영하는데 있어서 서버 문제에 신속하게 대처할 수 있다. 또한, 저장된 데이터를 기반으로 뛰어난 보고 및 데이터 시각화 기능을 제공한다.

IV-1-2. Zabbix의 구성

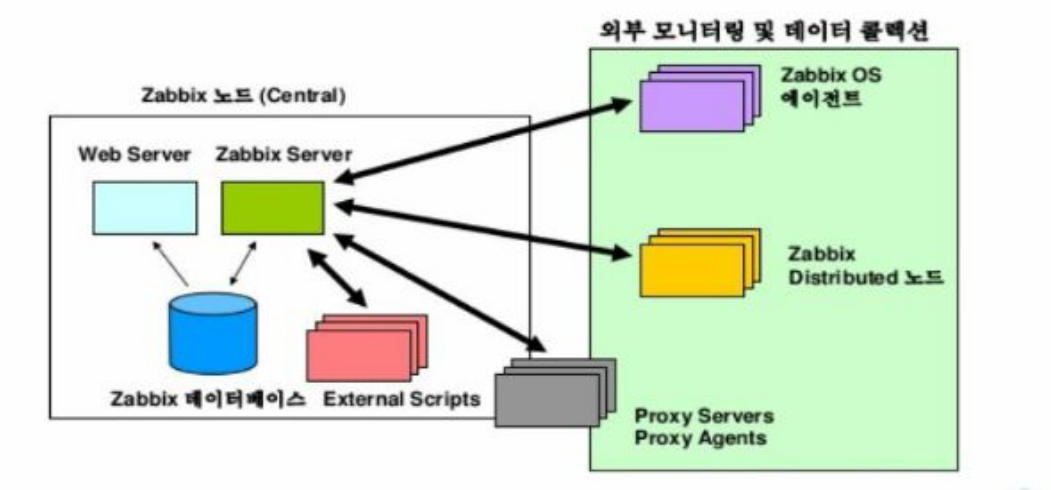


그림 4 : Zabbix 구성도

기본적으로 Zabbix는 여러 구성 요소를 지니고 있다. Zabbix Server는 Agent로부터 모니터링 데이터에 대한 정보를 수집한다. Server는 수집된 데이터를 바탕으로 통계를 내고 운영하는 것을 담당한다. Zabbix Database는 수집된 모든 데이터를 저장하게 된다. Database는 Mysql기반 데이터베이스를 이용하게 되는데, Mysql은 Database 관리 시스템 중에서 가장 많이 쓰이는 소프트웨어다. Web Server는 모든 플랫폼에서 사용자에게 쉽게 Zabbix를 액세스할 수 있는 웹 기반 인터페이스다. Openstack의 Dashboard처럼 item

혹은 template 등을 GUI(Graphical User Interface)방식으로 직접 생성할 수 있게 도와준다. Zabbix에서 제공되는 기능중의 하나인 graph를 기능을 visualization할 수 있게 된다. Zabbix proxy는 Zabbix server를 대신하여 데이터를 수집할 수 있다. proxy는 분산 모니터링 환경에서 아주 중요한 역할을 담당하게 된다. Zabbix에서 flexible하고 scalable한 기능을 제공하는 주요 인으로 server의 로드를 분산시키는 효과가 있다. 또한, proxy는 필수사항이 아닌 선택사항으로 운용할 수 있다. Zabbix agent는 모니터링 대상에 배포되며 로컬 리소스 및 응용 프로그램에 대하여 적극적으로 모니터링하고 수집된 데이터를 Zabbix server에 보고하게 된다. 다음은 분산 모니티링 환경 구성도의 한 예이다.

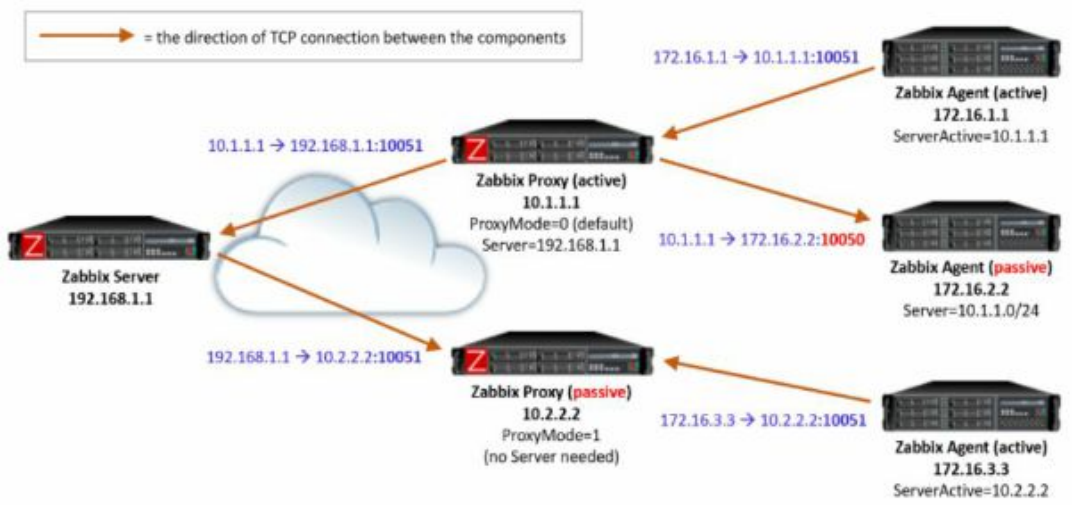


그림 5 : 분산 모니터링 구성도 예시

IV-1-3. Zabbix의 기능

오픈소스 기반 모니터링 소프트웨어인 만큼 Zabbix에서 제공하는 기능은 매우 다양하다. Zabbix에서 제공하는 기능은 다음과 같다.

① 데이터 수집
② 유연한 임계값 정의
③ 고도로 구성가능한 경고
④ 실시간 그래프 작성
⑤ 웹 모니터링 기능
⑥ 광범위한 시각화 옵션
⑦ 과거 데이터 저장
⑧ 간편한 구성
⑨ 템플릿 사용
⑩ 네트워크 discovery
⑪ 빠른 웹 인터페이스
⑫ Zabbix API
⑬ 쉽게 확장할 수 있는 Agent
⑭ 사용권한 시스템
⑮ 복잡한 환경에 대비

IV-2. Zabbix API

IV-2-1. REST API

Zabbix를 코드 레벨 단계에서 이용하기 위해 Zabbix에서 제공하는 여러 가지 기능 중에서 Zabbix API를 적극적으로 이용하기로 했다. Zabbix API는 REST API를 기반으로 한 프로그래밍 인터페이스다. REST API는 Representational State Transfer라는 용어의 약자로서 2000년도에 Roy Fielding의 박사 학위 논문에서 최초로 소개되었다. HTTP 프로토콜을 바탕으로 한 통신방식으로 웹의 장점을 최대한 활용할 수 있는 아키텍처로 REST를 발표했다. 다음은 REST API의 구성도이다.

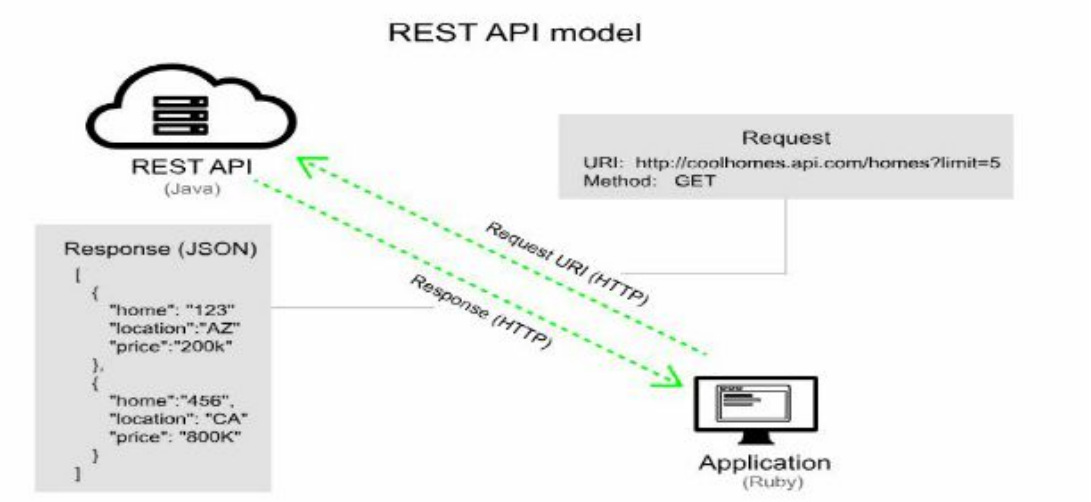


그림 6 : REST API 구성도

REST API는 자원(Resource), 행위(Verb), 표현(Representation)으로 구성되어 있다. 주고받는 자원의 경우 URI(Uniform Resource Identifier)로 지정하여 리소스에 접근할 수 있게 된다. 행위의 경우 HTTP Method를 기반으로 이루어져 있다. 대표적인 예로는 GET, POST, PUT, DELETE method들이 존재하고 각각의 역할은 다르게 정의되어 있다. 위의 구성도는 GET method를 이용한 REST API의 한 예로 주고받는 자원은 URI에 명시되어 있는 것을 볼 수 있고, API 서버로부터 Resource를 json 자료형으로 정의되어 얻어오는 것을 볼 수 있다.

IV-2-2. Zabbix API

bix API를 사용하게 되면 프로그래밍 방식으로 Zabbix의 구성 및 수정할 수 있고, 기록 데이터에 액세스할 수 있게 된다. Zabbix API는 웹 기반 API 이며 웹 프론트 엔드의 일부로 제공되고, JSON-RPC 2.0 프로토콜을 사용한다. Zabbix API를 기본적으로 이용하기 위해서는 접근 권한에 대한 허가 절차가 필요하다. 특정 포털사이트에서 로그인하는 과정과 많이 유사하다. 다음은 Zabbix API의 data flow이다.

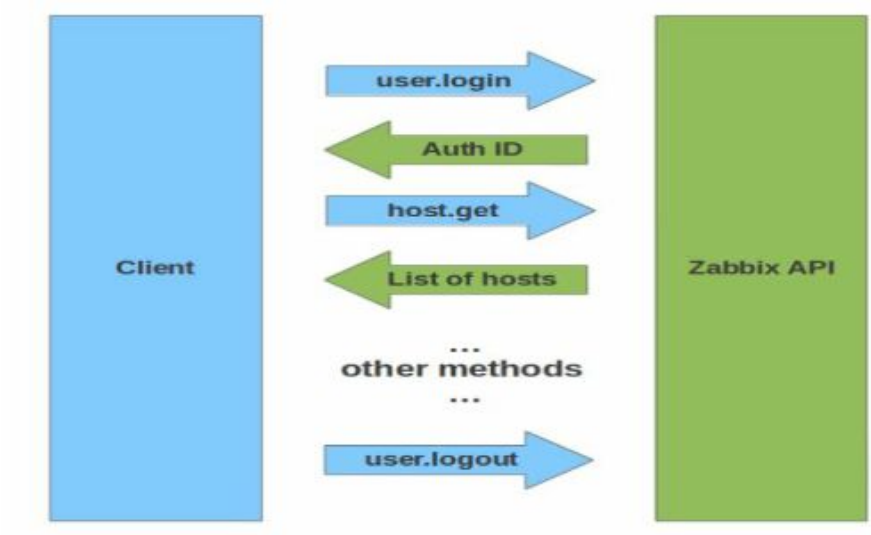


그림 7 : Zabbix API data flow

API Server에 접근하기 앞서서 접근 권한을 얻기 위한 Authentication 절차를 거치게 된다. Server에 접근하기 위한 URL 양식은 “ http://(zabbix_server_ip):(zabbix_server_port)/zabbix/api_jsonrpc.php” 이다. 서버에 접근 권한을 토큰을 통해 얻게 되면, 그 후에 정상적으로 Zabbix API에서 제공하는 여러 가지 method를 이용할 수 있다. 모든 정상적인 method 이용이 끝나게 되면 Zabbix API Server로부터 로그아웃하게 되는 절차를 진행하게 된다. 다음은 Authentication 절차에 대한 기본 양식이다.

```
{
    "jsonrpc": "2.0",
    "method": "user.login",
    "params": {
        "user": "Admin",
        "password": "zabbix"
    },
    "id": 1,
    "auth": null
}
```

그림 8 : Authentication Request

```
{
    "jsonrpc": "2.0",
    "result":
 "0424bd59b807674191e7d77572075f33",
    "id": 1
}
```

그림 9 : Authentication Response

IV-2-3. Zabbix API method

Zabbix API method에는 기본적으로 4가지 method가 존재한다. get은 Zabbix에 등록된 정보를 가져오는 역할을 담당한다. create는 특정 trigger나 item 등을 새로 생성하는 역할을 담당한다. delete는 특정 trigger나 item 등을 삭제한 역할을 담당한다. update는 trigger나 item 등에 대하여 상태를 업데이트 하는 역할을 담당한다. 아래 그림들은 특정 method에 대한 Request와 Response에 대한 결과를 나타낸다.

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": [
            "hostid",
            "host"
        ],
        "selectInterfaces": [
            "interfaceid",
            "ip"
        ]
    },
    "id": 2,
"auth":"0424bd59b807674191e7
d77572075f33"
}
```

그림 10 : get request

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10084",
            "host": "Zabbix server",
            "interfaces": [
                {
                    "interfaceid": "1",
                    "ip": "127.0.0.1"
                }
            ]
        }
    ],
    "id": 2
}
```

그림 11 get response

V. Vitrage 기초 이론

V-1. Vitrage

Vitrage란 Openstack의 프로젝트 중 하나로 Openstack 경보 및 이벤트를 구성, 분석 및 확장하여 문제의 근본 원인을 파악하고 직접 탐지되기 전에 그 존재를 추론하는 RCA(Root Cause Analysis) 서비스를 제공한다.

V-1-1. Vitrage가 제공하는 기능

Vitrage는 물리적 하드웨어와 가상머신들 사이의 매핑이 가능하다. 또한 시스템 분석을 기반으로 해서 추론한 경보를 발생시켜 상태(문제가 있는지 없는지)를 수정가능하다. 또한 경보/이벤트에 대한 RCA 기능을 제공하며 위의 기능에 대한 UI 플러그인도 제공한다.

V-1-2. Vitrage 구조

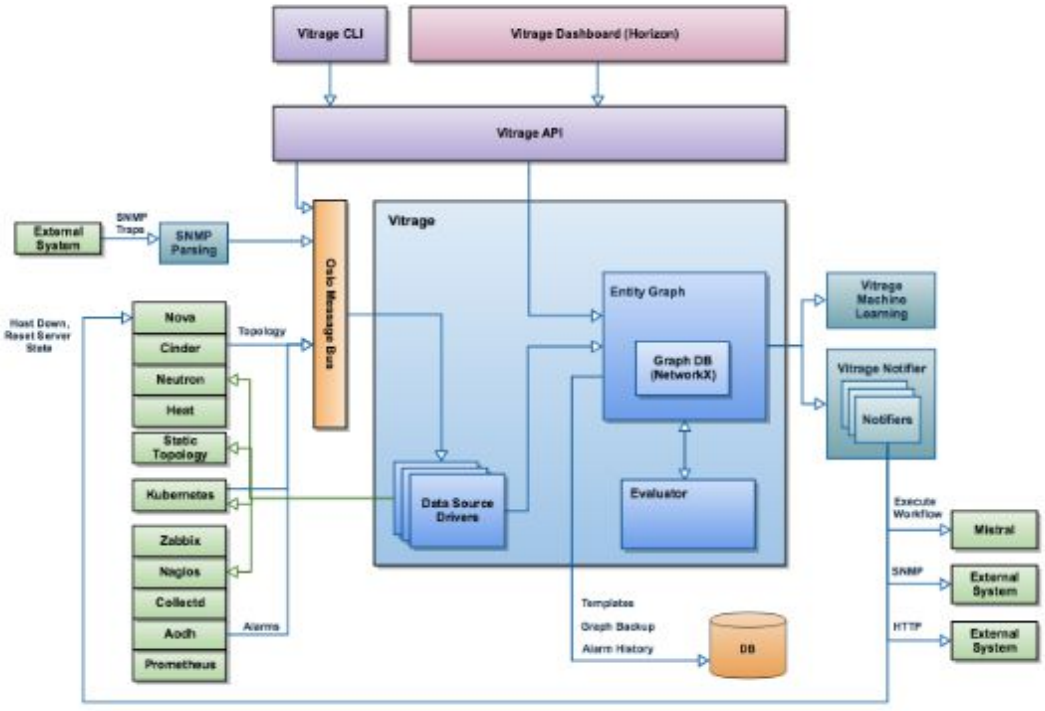


그림 12 : Vitrage 구조

1. Vitrage Data Source : 시스템의 상태와 관련하여 여러 리소스들에서 정보를 가져오는 책임이 있다. 여기에는 경보, 실제 뿐만 아니라 가상의 자원에 관한 정보가 포함 된다. 정보는 Vitrage 그래프로 처리 되면 현재는 Vitrage는 Nova, Cinder 및 Aodh OpenStack 프로젝트, Nagios 경보 및 정적 Physical Resources 데이터 소스에 대한 데이터들도 존재하다. 또한, 데이터 소스는 두가지 구성 요소로 구성된다. 드라이버는 정보 검색과 엔티티 큐에 입력을 처리하는 반면 Transformer는 검색된 정보를 그래프에 통합하는 방법을 정의 한다.

2. Vitrage Graph : Data Source에서 수집한 정보 뿐만 아니라 내부의 상호관계를 유지하고 있다. 또한 Vitrage Evaluator에서 사용하는 기본 그래프 알고리즘 모음(BFS, DFS 등)을 실행한다.

3. Vitrage Evaluator : Vitrage 그래프의 분석을 가지고 분석 결과를 처리한다. Vitrage에서 RCA 관계를 추가하거나 추론된 알람을 발생시키거나 추론된 상태를 설정하는 것과 같이 템플릿 기반의 다른 종류의 작업을 실행하는 역할을 한다.

V-2. Vitrage와 Zabbix 연동

네트워크 관리 시스템 중 하나인 Zabbix와 Vitrage를 연동시키면, Zabbix를 통해 모니터링한 결과를 Vitrage의 그래프에 표현 할 수 있으며, Vitrage의 시스템 분석 기능을 통해 RCA 서비스가 가능하다. Vitrage와 Zabbix를 연동시키기 위해선 몇가지 과정을 필요로 하며, 본 논문에서는 사용자의 편리한 사용을 위해 이를 자동화하도록 시스템을 설계하였다.

V-2-1. Vitrage 설정

1. /etc/vitrage/vitrage.conf의 데이터 소스 목록에 zabbix를 추가한다.

```
[datasources]
types = zabbix, nova.host, nova.instance, .nova.zone, static, aodh, cinder.volume,
neutron.network, neutron.port, heat.stack
```

2. 다음 섹션을 /etc/vitrage/vitrage.conf 에 추가한다.

[zabbix]

url = http : //< ip >/ zabbix

password = zabbix

user = admin

config_file = / etc / vitrage / zabbix_conf . yaml

밑줄 친 ip에는 zabbix agent의 ip주소를 넣어야 한다.

3. 이 내용으로 /etc/vitrage/zabbix_conf.yaml을 만든다.

zabbix :

- zabbix_host : Zabbix server

  type : nova . host

  name : Zabbix server

zabbix_host에는 모니터링 대상을 zabbix에 등록했을 시에 사용했던 zabbix agent의 이름을 넣어야 한다. type의 내용은 모니터링 대상에 따라 달라진다. Instance를 모니터링 한다면 nova.instance, 물리 장치를 모니터링 한다면 nova.host를 써야한다. name에는 Vitrage entity 그래프에 표현된 모니터링 대상의 ID를 넣어야 한다.

4. devstack/openstack에서 vitrage 서비스를 다시 시작한다.

VI. 모의실험 결과 및 토의

(1) 오픈스택 vitrage

그림 13 : vitrage entity graph

vitrage의 entity graph에서 노드를 클릭 할 시 우측에 설계한 관리용 서비스들이 나오는 것을 확인 할 수 있다.

(2) 관리용 서비스

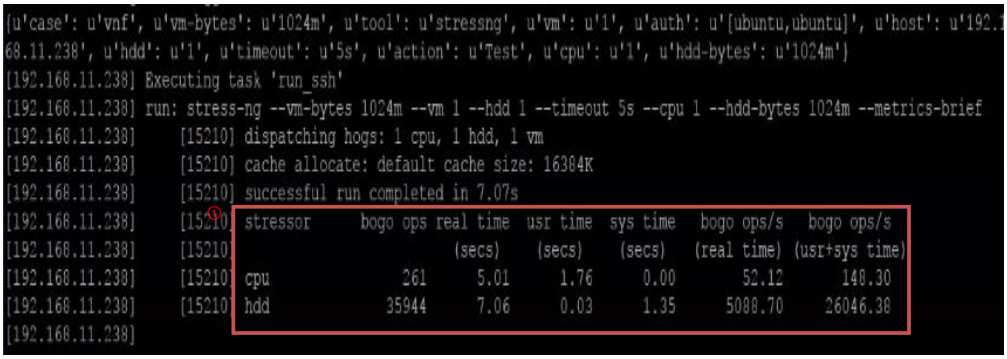


그림 14 : testing 결과 예시

그림 4의 ①은 testing의 결과이며 각 값의 의미는 다음과 같다.

결과	의미
bogo_ops	프로그램 동작 동안 스트레스를 반복 횟수.
real_time (secs)	스트레스 테스트 프로그램의 평균적인 지속 시간(초). 특정 스트레스 요인의 모든 시간을 스트레스 요인의 수로 나눈 값.
usr_time (secs)	모든 스트레스 테스트 인스턴스를 실행하는데 소비된 총 유저 시간.
sys_time (secs)	모든 스트레스 테스트 인스턴스를 실행하는데 소비된 총 시스템 시간.
bogo_ops (real_time)	실행 시간을 기준으로 초당 총 bogo 작업,
bogo_ops (usr+sys time)	누적 사용자 및 시스템 시간을 기준으로 초당 총 bogo 작업.

표 1. testing 결과 의미

testing의 결과를 통해 관리하고자 하는 VM의 모니터링 기준점을 세울 수 있다.

그림 15 : Monitoring을 위한 zabbix agent 스크립트 예시

그림 5는 monitoring을 위해 자동으로 생성되는 zabbix agent 설치 스크립트이다. 각 부분은 다음의 기능을 갖는다.

① - 사용자의 네임과 비밀번호를 설정 후 zabbix-agent를 설치한다.

② - 입력 받은 정보를 통해 IP address 및 호스트 네임을 셋팅한다.

③ - 자빅스 서버로부터 원격커맨드가 가능하게 EnableRemoteCommand의 값을 1로 바꾸어주며 zabbix-agent.conf 파일을 수정한다.

이를 통해 관리하고자하는 VM에 쉽게 zabbix-agent를 설치 할 수 있다.

(3) zabbix

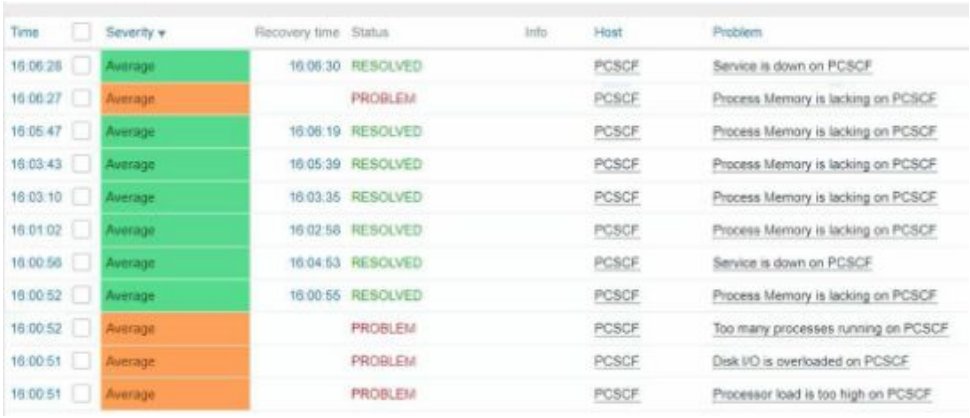
그림 16 : zabbix agent 등록

zabbix 서버에서는 monitoring의 스크립트를 통해 agent가 설치된 VM과 연결이 진행되고 연결 현황을 그림 6과 같이 확인 할 수 있다.

그림 17 : agent 트리거 등록

또한, testing을 통해 얻은 VM의 성능을 토대로 VM에 대해 트리거 설정을 통해 알람 설정의 기준점을 설정 할 수 있다.

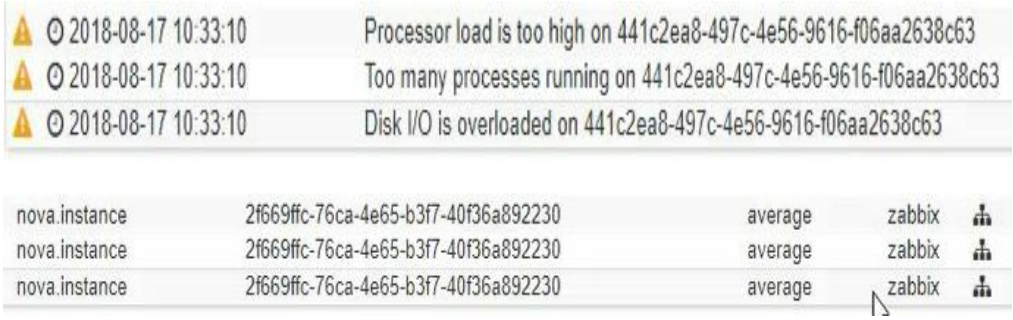
(4) 장애 발생 알림



Time	Severity	Recovery time	Status	Info	Host	Problem
16:06:28	Average	16:06:30	RESOLVED		PCSCF	Service is down on PCSCF
16:06:27	Average		PROBLEM		PCSCF	Process Memory is lacking on PCSCF
16:05:47	Average	16:06:19	RESOLVED		PCSCF	Process Memory is lacking on PCSCF
16:03:43	Average	16:05:39	RESOLVED		PCSCF	Process Memory is lacking on PCSCF
16:03:10	Average	16:03:35	RESOLVED		PCSCF	Process Memory is lacking on PCSCF
16:01:02	Average	16:02:58	RESOLVED		PCSCF	Process Memory is lacking on PCSCF
16:00:56	Average	16:04:53	RESOLVED		PCSCF	Service is down on PCSCF
16:00:52	Average	16:00:55	RESOLVED		PCSCF	Process Memory is lacking on PCSCF
16:00:52	Average		PROBLEM		PCSCF	Too many processes running on PCSCF
16:00:51	Average		PROBLEM		PCSCF	Disk I/O is overloaded on PCSCF
16:00:51	Average		PROBLEM		PCSCF	Processor load is too high on PCSCF

그림 18 : zabbix 서버 내부 장애 알림

관리하는 VM에서 장애가 발생시 zabbix 서버에서는 어떠한 VM에서 어떠한 문제로 인하여 장애가 발생하였는지 설정해놓은 트리거를 통해 알 수 있으며 장애의 해결 여부와 경도를 확인 할 수 있다.



2018-08-17 10:33:10	Processor load is too high on 441c2ea8-497c-4e56-9616-f06aa2638c63			
2018-08-17 10:33:10	Too many processes running on 441c2ea8-497c-4e56-9616-f06aa2638c63			
2018-08-17 10:33:10	Disk I/O is overloaded on 441c2ea8-497c-4e56-9616-f06aa2638c63			
nova.instance	2f669ffc-76ca-4e65-b3f7-40f36a892230	average	zabbix	
nova.instance	2f669ffc-76ca-4e65-b3f7-40f36a892230	average	zabbix	
nova.instance	2f669ffc-76ca-4e65-b3f7-40f36a892230	average	zabbix	

그림 19 : 오픈스택 vitrage 내부 장애 알림

또한, 오픈스택 vitrage의 알람 탭을 통하여 장애가 발생한 VM에 대해 발생한 장애의 원인과 발생 시간을 알 수 있다.

Ⅶ. 결 론

본 논문에서는 기존의 vitrage가 가지는 단점을 보안하고 나아가 관리 기능을 통한 발생한 문제에 대한 정보를 알 수 있는 시스템을 설계하고자 하였다. 이를 위해 vitrage와 zabbix, 관리 컴포넌트를 연결하는 system을 설계하였다. 관리 서비스 구성요소에는 checking, testing, monitoring의 기능이 있으며 이를 통하여 관리자는 관리하고자 하는 VM과 VNF에 대해 관리 및 문제에 대한 정보를 얻고 zabbix를 통한 모니터링 및 RCA 서비스를 받을 수 있었다. 향후 추가적인 관리 서비스 구성요소의 추가를 통하 사용자가 이용할 수 있는 관리 서비스의 다양성을 더하고 이를 통해 zabbix에서 장애를 분별할 수 있는 종류를 추가하는 작업이 필요할 것이라 생각된다.

참고문헌

[1] 위키피디아 – 클라우드컴퓨팅 https://en.wikipedia.org/wiki/Cloud_computing

[2] 오픈스택
https://docs.openstack.org/install-guide/get-started-logical-architecture.html

[3] Zabbix Documentation 3.4
[https://www.zabbix.com/documentation/3.4/start]

[4] MySQL 위키백과 [https://ko.wikipedia.org/wiki/MySQL]

[5] REST 위키백과 [https://ko.wikipedia.org/wiki/REST]

[6] https://docs.openstack.org/vitrage/latest/contributor/zabbix_vitrage.html

[7] 클라우드 컴퓨팅 시장 및 정책동향, 융합연구정책센터, 2018.03.05.

[8] 금융권 클라우드 서비스 현황 분석, 전자금융과 금융보안원, 2015. 10

[9] What is NFV MANO?
,https://www.sdxcentral.com/nfv/definitions/nfv-mano/

[10] https://wiki.openstack.org/wiki/Tacker

[11] 비트라지 https://wiki.openstack.org/wiki/Vitrage

[12] 자빅스 https://www.zabbix.com/features