情 報 保 護 學 會 誌
第 13 券 第 2 號, 2014. 09

효율적인 IaaS 클라우드 시스템 구축 방안

박 달 용*, 박 기 웅**

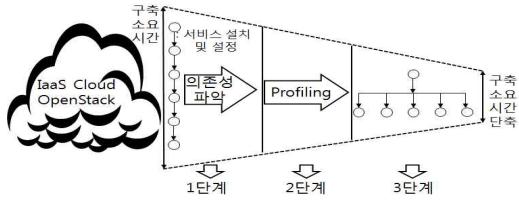
요 약

IaaS(Infrastructure as a Service) 클라우드 시스템인 OpenStack은 8개의 서비스로 구성되어 있으며 각각의 서비스를 구축함에 시간적인 Overhead가 크고 약 100여 가지의 명령어를 통한 구축되므로, 시스템을 구축함에 상당한 비효율성이 따른다. 따라서 본 논문은 이와 같은 구축 방법을 프로파일링하고, 서비스별의 의존성을 분석한 후 Script를 제작하여 병렬기반의 OpenStack 구축 방안을 제시하여, 클라우드 시스템을 구축한다. 일반적인 OpenStack 공식 가이드에서 제시하는 방법과 달리 각 서비스별의 의존성과 연관성을 프로파일링하여 개선된 OpenStack 구축 설계로 최대 400%까지 구축 속도 향상을 보인다.

Ⅰ. 서 론

최근, 클라우드 시장이 성장함에 따라 클라우드를 구축하는 문제가 화두에 올랐다 [1] . IaaS 클라우드 시스템 중 하나인 OpenStack [2] 은 가장 널리 사용되는 클라우드 시스템으로써, 대용량 시스템을 구성, 관리를 지원하며 구축 과정 시 약 30여 가지의 환경설정 파일과 약 100가지의 명령어를 실행해야 하며, 그에 따라 구축 시 시간문제가 발생한다. 또한, OpenStack을 스크립트로 구성되어 일괄적인 설치가 가능한 DevStack [3] 또한 순차적인 구축으로 인한 시간적 Overhead 발생과 설치되는 시스템 환경에 따른 환경설정 시 오류가 발생하여 구축에 어려움이 따른다 [?] .

따라서 본 논문에서는 이와 같은 문제를 해결하기 위해 3단계의 분석 절차를 통해 분석 방안을 제시한다. 〔그림 1〕에 도시된 바와 같이, 1단계는 서비스별 의존성과 연관성을 파악하며, 2단계는 이를 프로파일링하고, 3 단계는 이를 토대로 OpenStack구축을 최적화하여 OpenStack구축 시 시간적 성능 향상 방안을 제시한다.



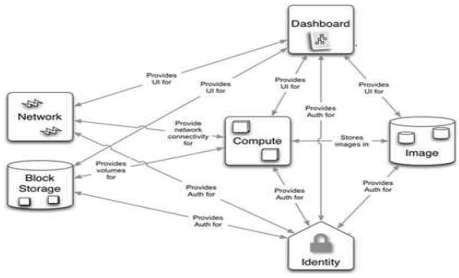
〔그림 1〕 효율적 구축 방안

본 논문의 구성은 다음과 같다. 2장에서 실험을 통해 각 서비스별 의존성을 파악하기 위한 알고리즘 제시와 프로파일링 분석결과를 제시한다. 3장은 2장의 분석 결과를 토대로 효율적인 구축방안을 제시한다. 또한 4장에서는 개선된 해결된 해결책을 기반으로 얼마나 속도향상이 이루어지는지 측정하며, 5장에서는 본 논문의 결론을 제시한다.

Ⅱ. 분 석

본 장에서는 효율적인 IaaS 클라우드 시스템 구축을 위한 서비스별 의존성을 파악 및 프로파일링 결과를

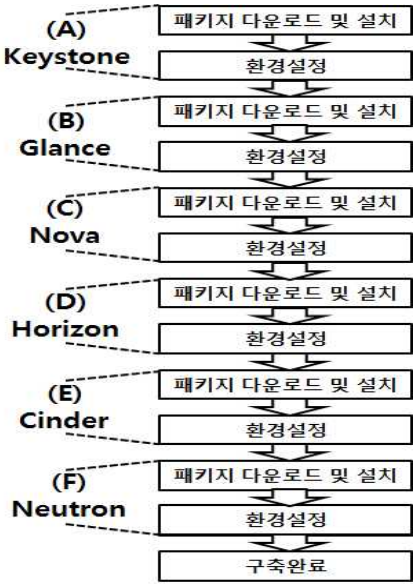
* : 박달용(주저자), 대전대학교 해킹보안학과 시스템보안연구실 석사과정(pdy0709@naver.com)
** : 박기웅(교신저자), 대전대학교 해킹보안학과 교수(woongbak@dju.ac.kr)



서비스	특징
Keystone(Identity)	서비스들의 인증관리
Glance(Image)	가상머신들의 이미지 관리
Cinder(Block Storage)	블록 스토리지 관리
Neutron(Network)	네트워크 관리
Nova(Compute)	인스턴스 관리
Horizon(Dashboard)	오픈스택 구현 서비스 제어를 위한 웹 인터페이스

〔그림 2〕 OpenStack 각 서비스 요소별 기능 [4]

나타낸다. OpenStack은 총 6개의 서비스로 구성되어 있으며 서비스들의 역할은 〔그림 2〕와 같다. 이와 같은 서비스들이 모두 설치되었을 때 비로소 정상적인 OpenStack이 동작하게 된다.



〔그림 3〕 OpenStack 기반 IaaS 구축 절차

OpenStack에서 제공하는 공식 문서 [4] 는 아래 〔그림 3〕처럼 각각의 서비스별로 순차적인 설치 및 설정을 하며 OpenStack을 구성하게 된다. 따라서 일반적인 설치자들은 가이드에서 제시하는 방안을 그대로 사용하며, 순차적인 구축을 진행한다.

〔그림 3〕에 도시된 구축 방법에 따라 구축이 진행될 경우, 각 설치 단계가 병렬적으로 구축이 될 수 있음에도, 순차적으로 진행되어 비효율적 구축이 이루어진다. 따라서 본 논문에서는 효율적인 구축을 위해, 각 설치 단계별 의존성 관계를 분석하는 알고리즘을 고안하였다.

알고리즘은 총 2가지로 이루어진다. 첫 번째 알고리즘은 다운로드 및 설치와 환경설정에 대한 순서 의존성 분석 알고리즘으로써, 다운로드 및 설치와 설정 순서에 대한 오류를 검증한다. 두 번째 알고리즘은 첫 번째 알고리즘의 분석을 바탕으로 설정 의존성을 판단하게 된다.

```
1 PROCEDURE Sequence_Dependency_Check
2 INPUT α:다운로드 및 설치,β:설정
3 BEGIN
4    IF(NoERR(Install α→β)) output_seq(α→β)
5    END IF
6    IF(NoERR(Install β→α)) output_seq(β→α)
7  END IF
8 END
```

〔그림 4〕 설치·설정 의존성 조사 알고리즘

〔그림 4〕의 설치·설정 의존성 알고리즘에서는 α, β의 순서 의존성을 조사한다. 먼저 α가 진행된 후 β가 진행되었을 시 오류가 나지 않는다면 α, β순서를 출력하여 오류가 없는 설치 순서 방안을 제시한다. 또한 먼저 β가 진행된 후 α가 진행되었을 시 오류가 나지 않는다면 β, α순서를 출력하여 오류가 없는 설치 순서 방안을 제시한다. 본 논문에서 제시한 OpenStack의 서비스별 설치·설정 의존성 검사에서는 다운로드 및 설치 후 설정을 해야 한다는 결과를 도출하였다. 설치, 설정 의존성 조사가 완료된 후, 각 서비스의 설정에 대한 의존성 조사를 진행하였다.

```
1 PROCEDURE Configuration_Dependency_Check
2 Proc = 오류가 없는 설정 Procedure
3 U = Proc의 집합
4 N = U 내부 Proc 개수
5 ← : 집합에서 2개의 요소를 선택하여 이동
6 BEGIN
7   REPEAT UNTIL N 〉 3
8      IF(Config(T ← U) = No_Error)
9          output(1st element of T)
10    END IF
11  END REPEAT
12 END
```

〔그림 5〕 설정 의존성 파악 알고리즘

〔그림 5〕의 설정 의존성 파악 알고리즘은 서비스들의 설정 의존성을 파악한다. 먼저 모든 서비스가 차례대로 2가지가 설정되어 오류가 나지 않는 조건에서, 첫

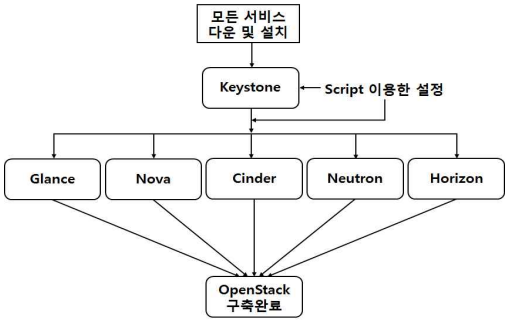
번째의 서비스설정을 출력하며, 첫 번째 서비스설정을 완료시켜 둔 뒤 나머지 서비스들을 차례대로 2가지 쌍으로 설정하여 오류검증을 반복적으로 수행한다. 본 논문에서의 수행 결과, Keystone과 나머지 서비스와의 설정 의존성이 존재하였다.

Ⅲ. 클라우드의 효율적 구축방안

본 절에서는 2절에서 실험했던 결과를 바탕으로 효율적인 구축 방안을 제시한다. Keystone과 나머지 서비스들은 서로 의존성과 연관성이 있으며 나머지 각각의 서비스들의 의존성과 연관성은 없는 것으로 나타났다. 따라서 각각의 서비스들을 순차적인 구축이 아닌 병렬적 구축 시에도 문제없이 동작하게 된다. 또한 〔그림 6〕과 같이 Script를 제작하여 서비스별 구축을 병렬적으로 진행하였다.

```
#!/bin/bash

echo -e '\E[40;33m'"\033[1m"[*]  HELLO SYSCORE" \033[0m"
source keystone_source

echo -e '\E[40;31m'"\033[1m"[*]  NEUTRON SETTINGS" \033[0m"


sed -i "s/\[filter:authtoken\]//"g /etc/neutron/api-paste.ini


echo "[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
auth_host = controller
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = neutron
admin_password = openstacktest" >> /etc/neutron/api-paste.ini
```

〔그림 6〕 도출된 OpenStack구축 Script 일부



〔그림 7〕 도출된 Script에 따른 OpenStack 구축 Procedure

〔그림 7〕과 같이 모든 서비스를 병렬적으로 다운받아 일괄적으로 설치하도록 apt-fast 명령어를 통해 일괄적 다운, 설치를 진행한다. 그 후 keystone 서비스의 구축을 진행한 뒤, 나머지 모든 서비스의 병렬적인 구축을 Script를 통해 일괄적으로 진행한다.

Ⅳ. 성능 평가

본 절에서는 개선 전 OpenStack 구축 시 설치 소요시간과 Script 기반 개선 후 설치 소요시간을 나타낸다. 실험환경은 CPU Xeon E5-2609(2.5GHz)를 사용하였고 Memory 32GB를 사용하였으며 Storage 환경은 SSD로 진행하였다. 클라우드 환경은 ubuntu12.04에서 진행되었다. 순차적으로 설치하는 OpenStack 구축 공식가이드로 OpenStack 구축 시 측정결과는 약 2,400초로 측정되었다. 본 논문에서 제시한 Script 기반 개선된 구축 해결책을 바탕으로 OpenStak 구축 시 약 480초가 소요된다. 개선 전 소요시간은 약 2,400초로 각 서비스당 약 300초의 시간이 소요된다. 개선 후 설치 소요시간은 약 480초로 최대 400%까지 속도 개선이 이루어졌다. 이는 3절에서 언급하였듯이 서비스별 의존성 검사와 프로파일링을 통해 구축과정을 최적화하였고, 그에 따른 Script를 통해 구축을 진행하였기 때문이다.

Ⅴ. 결 론

본 논문에서는 일반적인 IaaS 클라우드 시스템인 OpenStack 구축과정에서 일반화된 알고리즘을 제작하여 각각의 서비스 의존성과 연관성을 분석하였으며, 이에 따라 개선된 구축 방안을 제시하였다. 이 일반화된 알고리즘을 통해 다른 시스템들의 구축 시 의존성을 검사하여 개선된 구축과정을 알아낼 수 있을 것이다. 또한, 총 소요 시간을 측정하여 개선 전후의 구축방법에 대한 비교를 진행하였다. 개선된 설치방안을 제시하여 공식 구축 방안보다 최대 400%의 속도 개선을 나타냈다.

참 고 문 헌

〔1〕클라우드 지원센터, "2014년 상반기 클라우드 시장/정책 동향 보고서", 클라우드지원센터, 2014.06

〔2〕OpenStack, http://www.openstack.org, 2014

〔3〕DevStack, http://devstack.org. 2014

〔4〕OpenStack Documentation, http://docs.openstack.org/, 2014