공 학 석 사 학 위 논 문

클라우드 컴퓨팅 환경에서의 오픈소스 기반 로깅 시스템

2016년 2월

부 경 대 학 교 대 학 원

정 보 보 호 학 협 동 과 정

이 병 도

공 학 석 사 학 위 논 문

클라우드 컴퓨팅 환경에서의 오픈소스 기반 로깅 시스템

지도교수 신 상 욱

이 논문을 공학석사 학위논문으로 제출함.

2016년 2월

부 경 대 학 교 대 학 원

정보보호학협동과정

이 병 도

이병도의 공학석사 학위논문을 인준함.

2016년 2월 26일

주 심 이학박사 이 경 현 ㊞

위 원 이학박사 신 원 ㊞

위 원 이학박사 신 상 욱 ㊞

목 차

표 차례

그림 차례

Open Source based Logging System in Cloud Computing Environment

Byung-Do Lee

Department of Interdisciplinary Program of Information Security,
Graduate School, Pukyong National University

Abstract

Cloud computing that provide a elastic computing service is more complex compared to the existing computing systems. Accordingly, it has become increasingly important to maintain the stability and reliability of the computing system, and troubleshooting and real-time monitoring to address these challenges must be performed essentially. At this time, the handling of the log data is needed, but this work may be more difficult compared to the traditional logging system. Because log data in cloud computing environment has a lot of challenges such as atypical format, multi-layer and multi-point, volatility. In addition, there are another challenges in order to have the admissibility of the collected log data in court.

In this thesis, we design the logging service that provides the management and reliability of log data in a cloud computing environment. We implement the proposed cloud log system by using open sources on OpenStack platform, and then analyze the proposed system. The proposed system will present a new direction for the new logging system to the cloud service provider, and provide the service users with a new experience and a wide insight about the

current situation.

Ⅰ. 서론

1.1 연구 배경

클라우드 컴퓨팅은 ICT(Information and Communications Technologies)에서 아직도 성장 가능성이 높은 매력적인 분야로 평가받고 있다. IT 분야의 조사 및 자문회사인 가트너(Gartner)는 작년에 이어 올해 2015년에도 주목할 만한 '10대 전략 기술'[1]에 클라우드 컴퓨팅을 기재했으며, 시장 조사 및 컨설팅 기관인 IDC(International Data Corporation) 또한 'IDC 2015 예측'[2]에서 클라우드 컴퓨팅은 올해 거의 700억 달러의 고성장을 기록하고 2018년에는 1,260억 달러 규모로 늘어날 것이라고 전망했다. 뿐만 아니라 미국, 영국, 일본, 호주 등은 정부기관이 클라우드 컴퓨팅을 우선 도입하는 정책을 추진하고 있으며, 우리나라도 다소 늦었지만 클라우드 컴퓨팅의 발전 및 이용을 촉진하고 서비스를 안전하게 이용할 수 있는 환경을 조성하고자 올해부터 '클라우드컴퓨팅 발전 및 이용자 보호에 관한 법률'이 시행된다. 이처럼 클라우드 컴퓨팅은 여전히 매력적이며 그 가능성은 또한 그만큼 높다.

한편으로 가트너는 클라우드 컴퓨팅이 2015 하프 사이클(Hype Cycle)에서 각성기(Trough of Disillusionment) 단계로 접어들었음을 알렸다[3]. 이는 신기술에 대한 기대가 높은 버블기(Peak of Inflated Expectations) 단계를 지나 지나친 관심과 현실적인 제약에 부딪쳐 재조정기에 들었음을 의미한다. 이러한 제약의 일부분으로써 클라우드 컴퓨팅이 본질적으로 가지고 있는 제어권 손실, 법리상의 문제, 기존 컴퓨팅 플랫폼 환경에서 클라우드 컴퓨팅 플랫폼으로의 서드 파티 전환, 여러 보안 문제 등과 같은 한계

가 포함되어 있다.

클라우드 컴퓨팅 환경은 기존에 컴퓨팅 환경에 비해 더 큰 복잡성과 고유한 특성들로 인하여 다수의 보안 문제와 관리상의 문제를 일으키게 된다. 이러한 문제점들 중 하나가 로그 메시지를 다루는 문제다. 로그 메시지는 데이터 분석에서 중요한 역할 수행하는 요소이며, 일반적으로 서비스의 상태를 나타내거나 문제에 대응하기 위한 트러블 슈팅 그리고 실시간 모니터링 등에 이용되며 이 밖에 디지털 포렌식 분야에서도 이용된다. 그런데 현재 기존의 로깅 솔루션들은 클라우드 컴퓨팅과 같은 분산된 아웃소싱 플랫폼에서 발생하는 막대한 양의 로그 메시지에 대해서 적절하고 유연한 처리, 선택적 조회, 전송 및 저장 중에 보안 서비스 등이 부족한 실정이다.

따라서 본 논문에서는 클라우드 컴퓨팅 환경의 로깅 서비스가 지향해야할 방향에 대해 알아보고 로깅 서비스를 설계하여 분석을 수행하고, 오픈 소스를 이용하여 구현한다.

1.2 연구 목적 및 내용

본 논문은 현재 클라우드 컴퓨팅 환경에서의 제공하는 로깅 서비스가 직면한 어려움과 이를 극복하기 위해 로깅 시스템이 지향해야할 방향에 대해 논의한다. 그리고 이러한 지향 방향을 만족하는 로깅 시스템을 설계하여 실제 구현해 봄으로써 단기적으로 관리상의 이점과 신뢰성을 가진 로깅 서비스를 제공하는 것이지만 궁극적으로는 클라우드 컴퓨팅 기술의 미래 지향적 발전에 이바지할 수 있기를 목적으로 한다.

본 논문의 구성은 다음과 같다. 2장에서는 클라우드 컴퓨팅 환경에 대한 개괄적인 정의와 특성에 대해 알아보고, 클라우드 컴퓨팅 환경에서 발생하는 로그 메시지에 대한 분석과 클라우드 포렌식 상에 어려움에 대해 이야

기하며, 이를 대응하기 위한 요구사항들에 대해 조사하여 클라우드 환경에서의 로깅 서비스가 나아가야할 바람직한 방향을 제시한다. 3장에서는 이러한 요구사항들을 부합할 수 있도록 기능별로 여러 구성요소로 나누어 각각의 기능에 대해 정립하고, 생성된 로그 메시지가 수집되어 보관되기까지의 로깅 시스템 설계를 수행한다. 4장에서는 앞서 설계한 로깅 시스템을 바탕으로 실제 오픈 소스들을 이용하여 구현을 수행한다. 5장에서는 본 논문의 연구에 대한 결론을 제시하고, 향후 연구 과제에 대해 이야기하는 것으로 끝을 맺는다.

Ⅱ. 관련 연구

2.1 클라우드 컴퓨팅 환경

클라우드 컴퓨팅은 여러 단체 및 조직에서 다양하게 정의하고 있다. 대표적으로 미국 국립표준기술연구원(NIST; National Institute of Standards and Technology)에서는 클라우드 컴퓨팅을 “언제 어디서나 편리하게 최소한의 관리 노력과 서비스 제공자와의 상호작용으로 빠르게 제공되는 설정 가능한 공유 컴퓨팅 자원(예. 네트워크, 서버, 스토리지, 어플리케이션, 서비스) 풀에 필요에 따라 네트워크 접속을 가능하게 하는 모델”로 정의하고 있다[4]. 또한 NIST에서는 클라우드 컴퓨팅의 특성을 5가지, 서비스 모델은 3가지, 배치 방식은 4가지로 분류하고 있다.

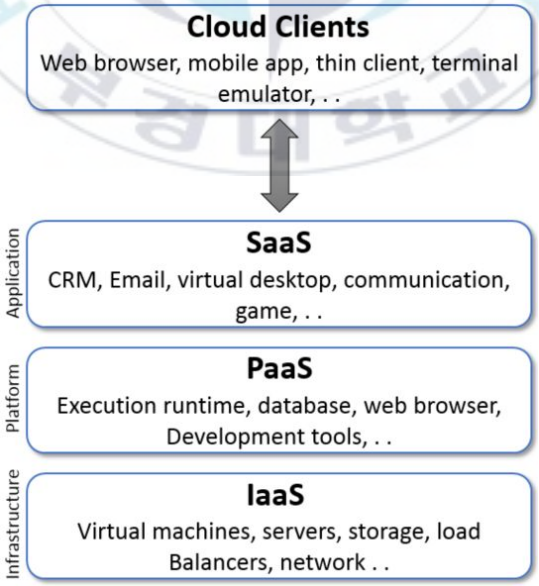
가. 클라우드 컴퓨팅 환경의 특성

NIST에서는 클라우드 컴퓨팅의 특징[4]을 다음 5가지로 기술하고 있다. 먼저 온 디맨드 셀프 서비스(On-demand self-service)는 사용자의 요구에 따라 서비스 제공자와의 상호작용 없이 자동으로 네트워크 스토리지와 서버 시간과 같은 컴퓨팅 기능을 독자적으로 제공한다. 광대역 네트워크 접근(Broad network access)은 다양한 클라이언트 플랫폼(예. 모바일 폰, 태블릿, 워크스테이션)에 의해서 이용을 촉진하는 표준 메커니즘을 통하여 접근이 가능하고 네트워크를 통해서 이용이 가능하다. 리소스 풀링(Resource pooling)은 멀티테넌트 모델을 이용하여 다수의 사용자에게 풀링된 컴퓨팅 리소스를 제공하며 다른 물리 및 가상 리소스가 사용자의 요

구에 따라 동적으로 할당과 해지가 일어난다. 빠른 탄력성(Rapid elasticity)은 역량이 탄력적으로 제공될 수 있음을 의미하며 일부 경우에는 자동으로 빠르게 요구에 상응하는 내외부의 정도를 조절할 수 있다. 측정 서비스(Measured service)는 클라우드 시스템이 서비스(예. 스토리지, 프로세싱, 대역폭, 사용자 계정 활성화) 유형에 따라 적절한 추상화 수준에서 측정 기능을 활용하여 자동으로 리소스 사용을 제어하여 최적화한다.

나. 서비스 모델

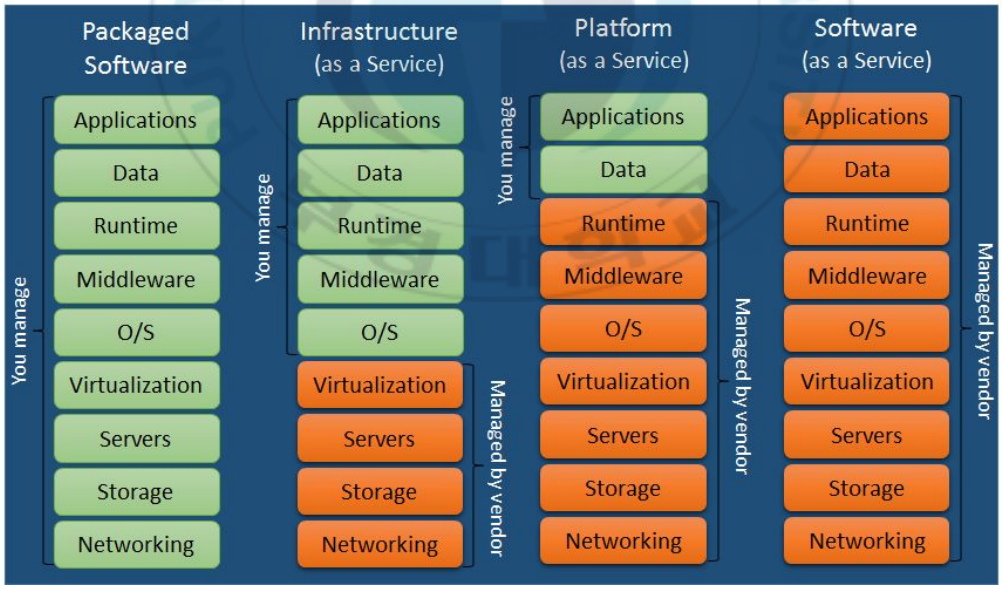
그림 1은 서비스 지향 아키텍처를 통해 모델에 따라 클라우드 컴퓨팅이 제공하는 서비스를 나타낸 것이다. 서비스 모델[4]은 Infrastructure as a service (IaaS), Platform as a service (PaaS), Software as a service (SaaS)로 분류된다.



[그림 1] 클라우드 컴퓨팅 서비스 모델

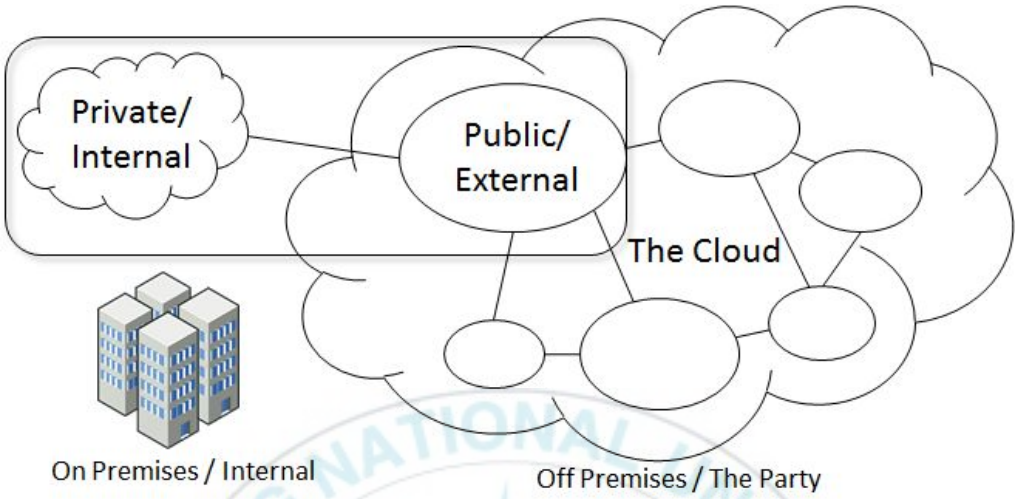
IaaS에서는 소비자에게 프로세싱, 스토리지, 네트워크, 다른 기본적인 컴퓨팅 리소스를 제공하여, 소비자는 운영체제와 애플리케이션을 포함하는 임의의 소프트웨어를 배치하고 수행할 수 있다. PaaS에서는 사용자에게 제공자가 지원하는 프로그래밍 언어, 라이브러리, 서비스, 도구를 이용하여 생성된 애플리케이션을 획득하거나 사용자가 생성한 클라우드 인프라로의 배치를 제공한다. SaaS에서는 클라우드 인프라에서 동작하는 제공자의 애플리케이션을 사용자가 이용할 수 있으며, 애플리케이션은 웹 브라우저와 같은 클라이언트 인터페이스나 프로그램 인터페이스를 통하여 다양한 클라이언트 장치로부터 접근가능하다.

그림 2는 위에서 기술한 서비스 모델들의 데이터 관리 영역을 나타낸 것이다.



[그림 2] 서비스 제공자의 관리 책임에 대한 데이터 시각화

다. 배치 모델



[그림 3] 클라우드 컴퓨팅 유형

그림 3은 클라우드 컴퓨팅의 배치 모델을 나타낸 것으로, 사설(Private) 클라우드, 커뮤니티(Community) 클라우드, 공용(Public) 클라우드, 하이브리드(Hybrid) 클라우드로 분류된다[4]. 사설 클라우드는 다수의 사용자(예. 비즈니스 부서)로 구성단일 조직의 독점적 이용을 위해 제공되며, 그 조직이나 서드파티, 혹은 그 조합으로 소유, 관리, 운영되며 소재지 내외에 존재할 수 있다. 커뮤니티 클라우드는 관심사(예. 임무, 보안 요구사항, 정책과 준수 사항)를 공유하는 조직에서 특정 소비자 커뮤니티의 독점적인 이용을 위해 제공되며, 커뮤니티, 서드파티, 혹은 그 조합 내 하나 이상의 조직에 의해 소유, 관리, 운영될 수 있으며 소재지 내외에 존재할 수 있다. 공용 클라우드는 일단적인 대중에게 열린 이용을 위에 제공되며, 비즈니스, 아카데미, 정부 기관, 혹은 그 조합으로 소유, 관리, 운영되며 클라우드 제공자의 소재지에 존재한다. 하이브리드 클라우드는 각각의 고유성이 유지되어 둘 이상의 구분되는 클라우드 인프라(사설, 커뮤니티, 공용)로 구성되

지만, 각각은 데이터와 애플리케이션 이동성(예. 클라우드 사이에 로드 밸런싱을 위한 이동)을 가능하게 하는 표준화나 독자기술로 바인딩 된다.

2.2 클라우드 환경에서의 로깅

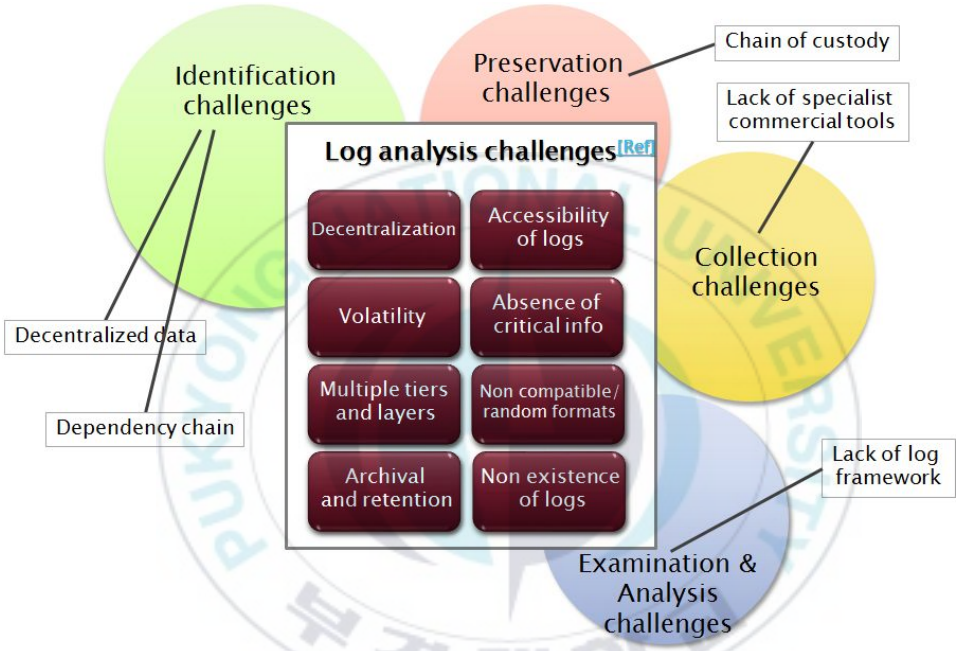
본 장에서는 클라우드 환경에서 발생하는 로그 메시지에 대한 특성과 법적 수용을 위한 신뢰성 및 분석 어려움에 대해 기술한다. 그리고 이러한 어려움을 해결하기 위한 요구사항이 무엇인지에 대해 추가적으로 언급한다.

가. 클라우드 환경에서의 로깅의 어려움

증거 능력(Admissibility)은 어떤 증거로의 수용을 판단하는 요구 사항을 규정하는 법적 개념이다. 그래서 증거 능력에 관한 기본 증거주의는 '증거와 관련된 신뢰성 보장'이 있다는 것을 판단하는 것이다[5].

이러한 신뢰성을 획득하기 위해서 클라우드 환경에서 생성되는 디지털 데이터는 기존의 디지털 포렌식 방법이 아니라 클라우드 포렌식[6]으로 다루어져야 한다. 일반적인 디지털 포렌식은 클라우드 컴퓨팅 시스템의 특성인 분산처리, 멀티 테넌시, 가상화, 동적인 환경으로 인하여 디지털 증거의 식별, 보존, 수집하는 것을 어렵게 하기 때문에 시스템에 직접 적용하는 것은 적절하지 않다. 이러한 문제점을 해결하기 위해 클라우드 환경에 적합한 디지털 포렌식 절차인 클라우드 포렌식이 필요하며, 디지털 포렌식 과정에 있어서 수사에 기본이 되는 로깅 또한 기존 방식과는 다른 방식으로 진행되어야 한다.

지금까지 디지털 포렌식 절차는 다양한 방식[7][8][9]으로 제안되어있으며, 일반적으로 식별(Identification), 보전(Preservation), 수집(Collection), 조사(Examination), 분석(Analysis), 표현(Presentation) 단계로 구성된 [7]의 DIP(Digital Investigative Process)방식에 따라 진행한다.



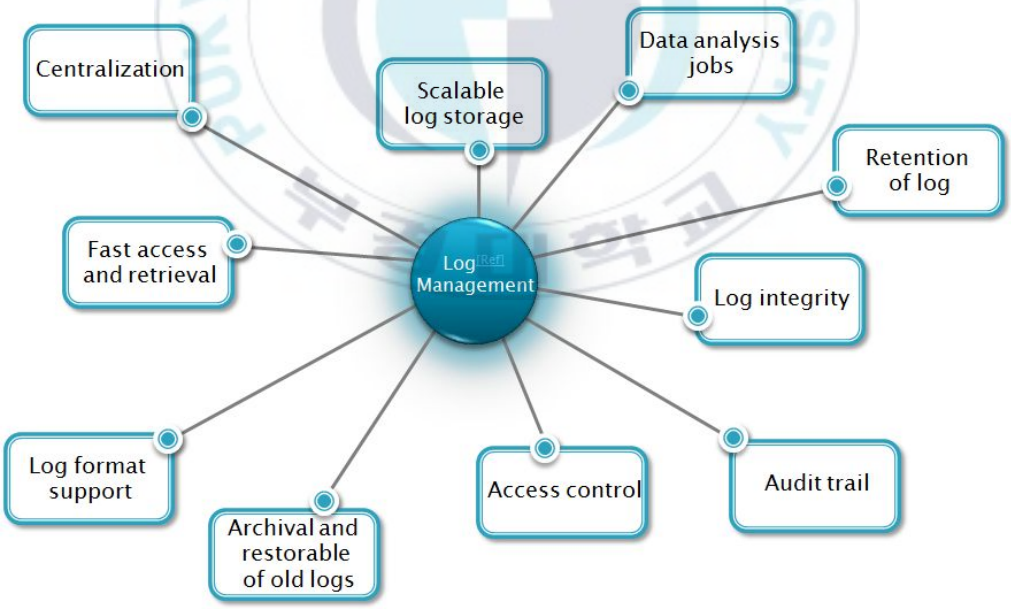
[그림 4] 클라우드 포렌식과 로깅의 어려움

이러한 DIP 방식에 따라 [10]은 클라우드 포렌식에 대한 기술적 어려움과 해결책 그리고 비교 분석에 대해 기술하고 있다. 각 단계별로 기술된 어려움들 가운데 로깅에 관련된 어려움으로 식별 단계에서 분산된 데이터(Decentralized data)와 의존성 체인(Dependency chain), 보존 단계에서 연계 보관성(Chain of custody), 수집 단계에서 전문가들이 사용하기 위한 상용도구의 결여(Lack of specialist commercial tools), 조사와 분석단계에서 로그 프레임워크의 결여(Lack of log framework)로 기술하고 있다. 이와

더불어 [11]은 디지털 포렌식을 위해 수행하는 클라우드 환경에서의 로그 분석의 어려움에 대해 기술하고 있다. 분석의 어려움으로는 분산된 로그, 로그 휘발성, 다 계층·단, 보관 및 유지, 로그 접근성, 로그 비존재성, 로그 중요 정보 부재, 비호환 및 임의 로그 형식으로 기술하고 있다. 이러한 어려움들을 정리하면 그림 4로 나타낼 수 있다.

나. 클라우드 포렌식과 관리를 위한 로깅 요구사항

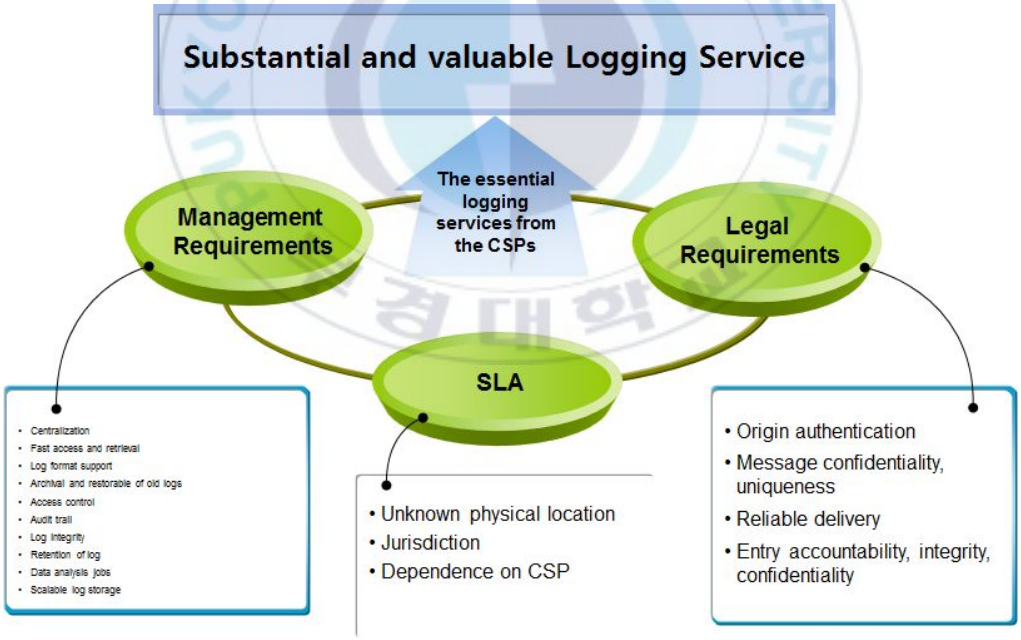
클라우드 환경에서 생성되는 로그 메시지에 대한 분석 어려움을 해결하기 위해서는 적절한 로그 관리가 선행되어야 한다. [11]에서는 로그 관리를 위한 요구사항으로 모든 로그의 중앙 집중화, 확장성 있는 로그 저장소, 신



[그림 5] 클라우드 컴퓨팅에서의 로그 관리 요구사항

속한 데이터 접근과 검색, 일부 로그 포맷 지원, 데이터 분석 작업 수행, 로그 레코드 유지, 로그 장기 보관과 필요시 복원, 접근 제어를 통한 분할된 데이터 접근 제어, 로그 무결성 보존, 로그 접근에 대한 감사 추적에 대해 기술하고 있다. 이는 그림 5와 같이 나타낼 수 있다.

이러한 로그 관리가 바탕이 되었을 때, 다음 로그 메시지의 신뢰성 위한 [5]의 보안 요구사항이 적용되어야 한다. 보안 요구사항은 전송 단계와 저장 단계 각각에서 충족되어야 한다. 먼저 전송 단계에서는 출처 인증, 메시지 기밀성, 메시지 무결성, 메시지 고유성, 신뢰 전송이 이루어져야 한다. 저장 단계에서는 엔트리(Entry) 책임 추적성, 엔트리 무결성, 엔트리 기밀성이 이루어져야 한다.



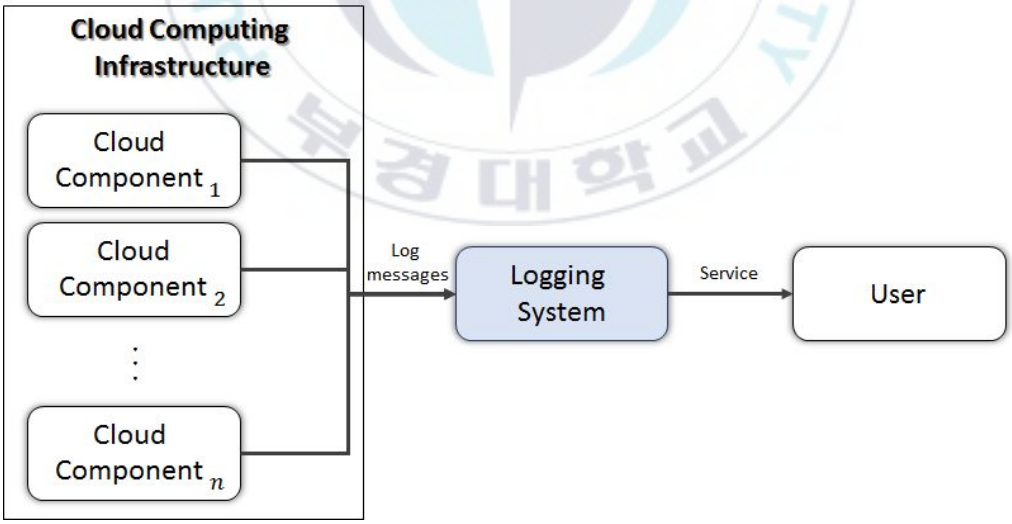
[그림 6] 클라우드 환경에서의 신뢰성 있는 로깅 방향

그림 6의 클라우드 환경에서 실질적이고 가치 있는 로깅 서비스는 로그

관리 요구사항인 [11]과 신뢰성 요구사항인 [5] 그리고 클라우드 포렌식에 해결책을 준수하는 [6]의 SLA(Service Level Agreement)을 바탕으로 구성되어야 한다. 이러한 로깅 서비스는 클라우드 다양한 사용자에게 유의미한 정보를 제공함은 물론이고, 다양한 보안 서비스와 법적 효력까지 제공해 줄 것이다.

Ⅲ. 클라우드 컴퓨팅 환경에서의 로깅 시스템 설계

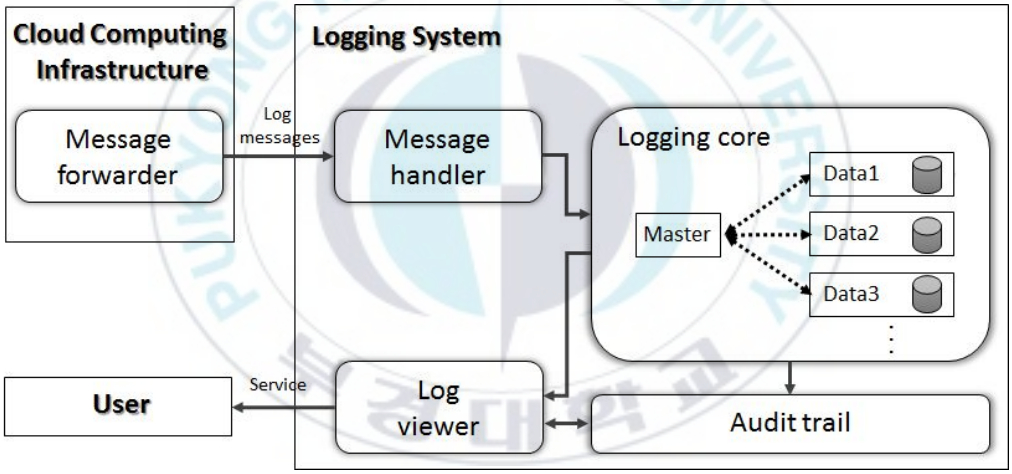
클라우드 컴퓨팅 서비스를 제공하는 인프라에서는 상당히 많은 양과 다양한 종류의 로그 메시지가 산재되어 발생한다. 이에 인프라 내에 물리적으로 구분되는 각 구성요소에 배치된 로그 메시지 전송 모듈은 로그 메시지가 발생할 때마다 실시간으로 로깅 시스템에 전송한다. 이러한 방식으로 다수의 모듈에서 로그 메시지가 로깅 시스템으로 중앙 집중화된 형태로 수집이 되며 수집된 로그 메시지는 메타 데이터 추가, 정규화, 파싱 등의 과정이 수행되어 저장된다. 이후에 사용자는 통합된 뷰를 통하여 저장된 로그에 대한 로그 정보, 질의, 시각화 등의 서비스를 제공받는다.



[그림 7] 제안 로깅 시스템 개요

3.1 로깅 시스템 구성요소

로깅 시스템(LS: Logging System)은 Message handler, Logging core, Log viewer, Audit trail로 구성되며, 클라우드 인프라의 각 호스트에 있는 Message forwarder는 Message handler에게 안전한 방식으로 로그 메시지를 전달하여 Log viewer가 로그 뷰를 사용자에게 제공한다. 그림 8은 로깅 시스템의 구성요소를 나타낸 것이며, 각 구성요소의 역할은 다음과 같다.



[그림 8] 로깅 시스템 구성요소

· Message forwarder(MF): 클라우드 컴퓨팅 인프라를 구성하는 호스트에 배치되며, 로컬 호스트에 있는 로그와 연관된 메타 데이터를 수집하여 Message handler에 전송하는 역할을 담당한다. 이때, 호스트의 낮은 자원 사용율과 안전한 로그 메시지 전송, 낮은 지연시간 그리고 전송 신뢰성을 제공한다.

· Message handler(MH): 수신된 로그 메시지와 연관 메타 데이터에 대하

여 정규화를 수행하며 Logging core에게 정규화된 로그 메시지와 레코드를 구성하는데 필요한 레코드 메타 데이터를 전달하는 역할을 담당한다.

· Logging core(LC): Message handler로부터 전달받은 정규화된 로그 메시지와 레코드 메타 데이터를 이용하여 실시간 분석을 수행한다. LC는 다수의 노드로 구성되는 클러스터로 동작하며, 노드는 하나의 마스터 노드(Master node)와 다수의 데이터 노드(Data node)로 분류된다. 마스터 노드는 클러스터 전체를 관리하고 외부의 명령을 전달 받으며, 데이터 노드는 로그 메시지 인덱싱과 검색을 수행하여 인덱싱된 데이터를 저장한다.

· Log viewer(LV): 이용자에게 로그 데이터를 볼 수 있도록 하는 뷰어로써 질의와 로그 정보 및 시각화를 제공하는 인터페이스 역할을 수행한다.

· Audit trail(AT): 로깅 시스템의 운용과 작업에 관한 로깅을 담당하며, 추가적으로 LC에 저장된 로그 파일에 접근하는 사용자에 대한 로깅을 수행한다.

3.2 로그 메시지 전송과 로그 데이터 정규화

구성요소들의 정상적인 동작을 위해 4 단계로 이루어진 초기화를 먼저 수행하여야 한다. 첫 번째 단계는 각 MF의 공개키 PK_{MF}와 개인키 SK_{MF} 그리고 로깅 시스템 LS의 공개키 PK_{LS}와 개인키 SK_{LS}를 생성하며, 이때 공개키 관리와 신뢰성 확보를 위해 공개키 인프라 PKI를 이용할 수 있다. 두 번째 단계에서 로깅 시스템과 각 MF는 내부시간을 신뢰되는 시간으로 동기화한다. 세 번째 단계에서 LC는 로그 데이터를 암호화하기 위해 이용되는 초기 IV와 블록암호 알고리즘 비밀키 K를 임의로 선택하여 안전하게 유지한다고 가정한다. 마지막 네 번째 단계에서는 수행한 초기화 작업의 내용과 결과를 AT에 기록한다.

로그 메시지 전송은 MF와 MH 사이에 발생한다. 우선 MF와 MH는 로그 메시지의 신뢰성 있는 전송을 위해 상호인증을 수행하고 적절한 세션 키를 설정한다. 로그 데이터가 발생하면 MF는 로그 데이터 원본 $OriLog_n$과 연관된 로그 메타 데이터 $LM-Data$을 생성하며, 이때 $LM-Data$은 로그를 전송한 클라우드 인프라의 호스트 $host$, 로그가 생성된 파일의 절대 경로 $file$, 로그 데이터 타입 $LType$으로 구성된다. 그리고 MF는 설정된 세션 키로 $OriLog_n$와 $LM-Data$에 암호화를 수행하며 이 암호문의 해시 값 $hashVal$을 생성하고 개인키를 이용하여 서명을 수행한다. 이렇게 생성된 암호문과 전자서명을 MH에 전송한다. 이러한 과정은 로그 데이터가 발생되는 즉시 수행된다.

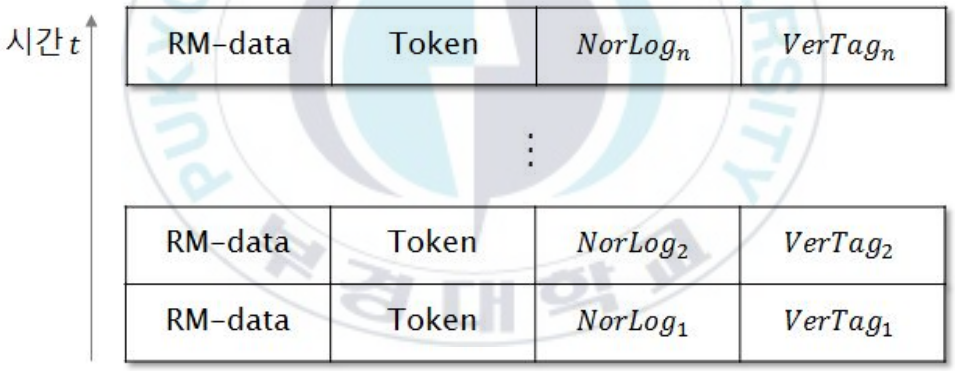
암호문과 전자서명을 전달받은 MH는 MF의 공개키 PK_{MF}를 이용하여 검증을 수행한다. 검증에 성공하면, 암호문을 복호화하여 $OriLog_n$과 $LM-Data$를 이용하여 정규화를 수행한다. 이 과정으로 $OriLog_n$와 $LM-Data$을 필드로 포함하는 정규화된 로그 데이터 $NorLog_n$으로 변환된다. 그리고 이후에 LC가 각 $NorLog_n$에 대한 레코드를 구성하기 위해 필요한 메타 데이터 $RM-Data$을 생성하여 $NorLog_n$와 함께 전달한다. 여기서 $RM-Data$에는 레코드가 속하는 인덱스 $Index$, 레코드 타입 $RType$, 레코드 Id 등이 포함된다.

3.3 로그 메시지 추가

그림 9는 일정 시간 구간 t동안 수집된 로그 메시지를 로그 파일로 나타낸 것이다. 로그 파일은 생성된 시간 순으로 로그 레코드가 추가되며, 각

로그 레코드는 $RM-Data$, 토큰 $Token$, $NorLog_n$, 앞의 3개의 필드와 이전 레코드들의 유효성을 검증하는 태그 $VerTag_n$으로 구성된다.

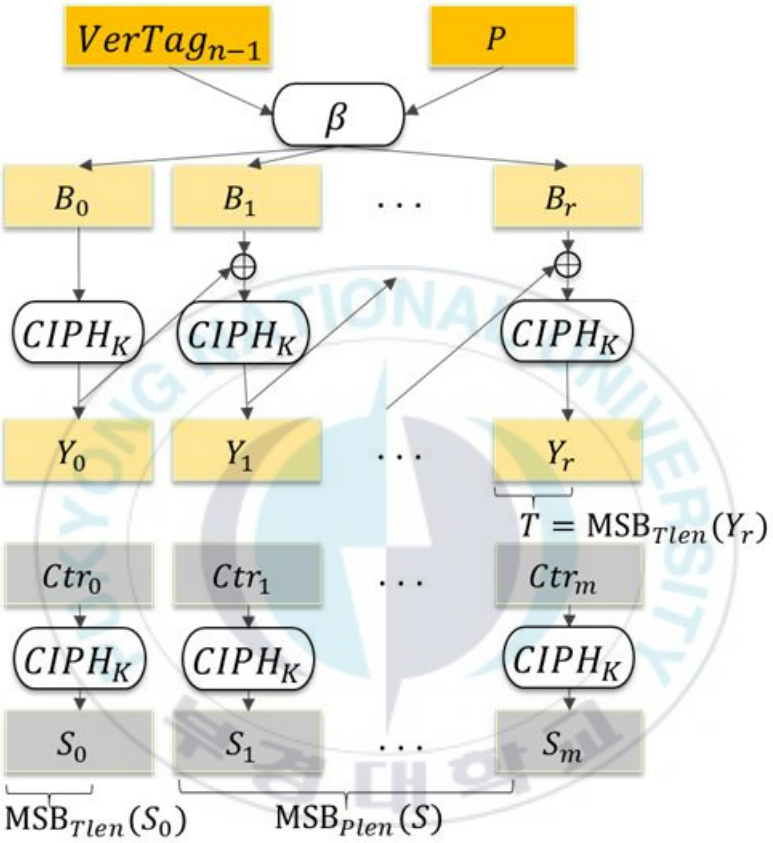
LC의 마스터 노드는 MH로부터 $RM-Data$와 $NorLog_n$을 전달받으며, $NorLog_n$에 대한 레코드를 생성하기 위해 다음 작업을 수행한다. 먼저 첫 번째 필드 $RM-Data$는 MH로부터 전달 받은 $RM-Data$을 그대로 이용한다. 두 번째 필드인 $Token$는 $NorLog_n$을 다수의 데이터 노드에 인덱싱 작업을 할당하여 검색 가능한 키워드를 생성해 구성한다. 마지막으로 $VerTag_n$ 필드는 각 레코드의 유효성을 판단하기 위해 $RM-Data$와 $Token$ 그리고 $NorLog_n$ 필드로 구성된 페이로드에 CCM(Counter with CBC-MAC)[12] 방식의 암호화를 적용하여 생성한다.



[그림 9] 로그 파일 구조

로그 파일은 저장 단계에서 기밀성과 무결성을 유지하기 위해 CCM 방식을 이용하여 각 레코드의 암호화와 검증 태그를 추가하여 저장한다. 이를 위해서 먼저 페이로드 $P=RM-Data \| Token \| NorLog_n$를 생성하고, P를 입력으로 하는 포맷 함수 β를 이용하여 블록 B_1, B_2. .., B_r을 생성한다. 여기서 B_0은 이전 검증 태그 $VerTag_{n-1}$을 이용하여 생성한다(단, 최초 B_0는

IV를 이용하여 생성한다). 그리고 생성된 각 블록 B_i에 전방향 암호화 함수인



[그림 10] 로그 레코드 암호화

$CIPH_K(B_i \oplus Y_{i-1})$ $(1 \le i \le r)$을 적용하여 Y_i를 생성한다. 마지막에 생성되는 Y_r에서 임의의 길이의 비트를 획득하는 $MSB_{Tlen}(Y_r)$을 수행하여 태그 T를 생성한다. 이후 카운터 블록 $Ctr_m = Ctr_{m-1} + 1$ $(1 \le m \le \lceil Plen/128 \rceil)$은 $CIPH_K(Ctr_j)$ $(1 \le j \le m)$을 이용하여 S_j을 생성한다. 여기에서도 Ctr_0는 이전 검증 태그 $VerTag_{n-1}$를 이용하여 생성한다(단, 최초 Ctr_0는 IV를 이

용하여 생성한다). 그리고 생성된 S_1, S_2, .., S_m을 순서대로 연결하여 S=S_1 ‖ S_2 ‖ .. ‖ S_m을 생성한다. 이렇게 생성된 값들을 이용하여 암호문 C는 $P \oplus MSB_{Plen}(S)$으로 획득하고, $VerTag_n$는 $T \oplus MSB_{Tlen}(S_0)$으로 획득한다. 이러한 과정은 그림 10으로 도식화 할 수 있다.

3.4 로그 파일 복호화와 검증

로그 레코드의 검증은 암호문 C가 복호화되었을 때의 평문이 유효한 것인지를 판단하는 역할을 한다. 로그 파일의 최상위 레코드 검증 태그 $VerTag_n$가 유효하다는 것은 해당 로그 파일의 복호화된 모든 레코드가 유효하다는 것을 의미한다.

암호화된 레코드의 복호화를 수행하기 위해 최초에 C의 길이가 $C \leq 0$ 인지 확인한다. 암호문의 길이는 0이거나 음의 정수이면 유효하지 않으므로 복호화를 중단시키고 그렇지 않으면, 암호화 과정과 같은 방법으로 카운터 블록 Ctr_j에 $CIPH_K(Ctr_j)$을 적용하여 S_j를 생성한다($1 \leq i \leq m$). 생성된 S_1, S_2, .., S_m를 연결하여 S=S_1 ‖ S_2 ‖ .. ‖ S_m을 생성한다. 이렇게 생성된 S를 이용하여 페이로드 P'는 $C \oplus MSB_{Plen}(S)$을 계산하여 획득한다. 그러나 획득한 P'는 검증과정에서 유효하다고 판단되어야만 받아들여질 수 있다. 검증과정에서 먼저, 복호화 과정에서 생성된 S_0으로 $VerTag_n \oplus MSB_{Tlen}(S_0)$을 계산하여 T를 복원한다. 그리고 획득한 P'와 $VerTag_{n-1}$를 입력으로 포맷함수 β를 이용하여 블록 B_0, B_1, B_2. .., B_r을 생성한다. 그리고 생성된 각 블록 B_i를 통하여 $CIPH_K(B_i \oplus Y_{i-1})$을 이용하여 Y_i를 생성하고 ($1 \leq i \leq r$), 마지막에 Y_r을 이용하여 T'를 생성한다. 이렇게 복원한 T와

T'를 비교하여 일치하지 않으면 복호화된 페이로드 P'가 유효하지 않다고 판단하고, 일치하면 유효하다고 인증한다.

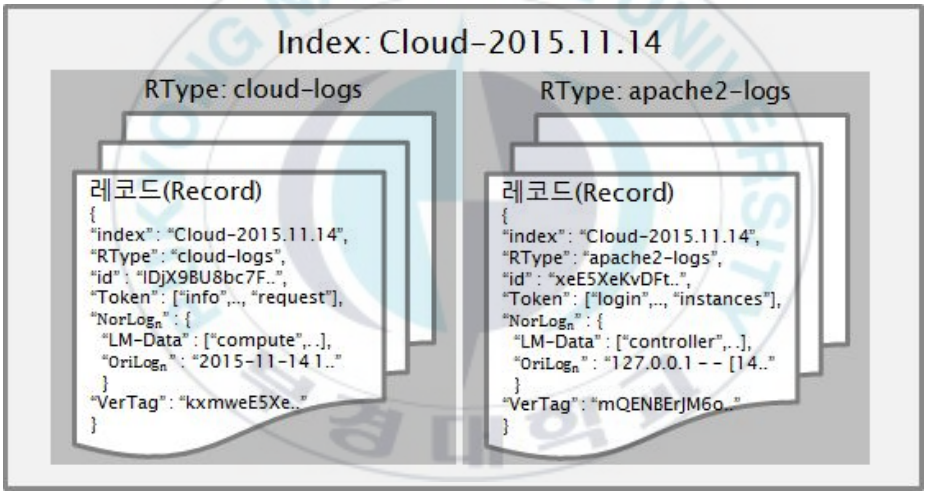
3.5 로그 파일 관리와 레코드 구성

로그 파일의 관리는 LC 내에 하나의 마스터 노드와 다수의 데이터 노드로 구성되는 클러스터를 통해 이루어진다. 이 클러스터는 로그 파일을 CCM 방식으로 암호화하여 다수의 데이터 노드에 분산 저장한다. 그리고 백업을 위해 사본을 생성하고 원본과 마찬가지로 암호화를 수행하여 분산 저장한다. 이러한 작업은 이후 사용자가 검색을 수행할 때 분산된 원본과 사본 모두를 이용하기 때문에 성능 향상 효과를 볼 수 있다.

로그 레코드의 정규화는 다양한 형식으로 표현된 로그들을 일관된 형식으로 유지하도록 하여 동일한 작업을 가능하게 한다. 그래서 정규화를 수행하기 위해서는 로그 레코드의 형식을 먼저 정의해야 한다. 로그 레코드는 $RM-Data$, $Token$, $NorLog_n$, $VerTag_n$ 필드로 구성된다. 그리고 레코드의 메타 데이터 $RM-Data$은 인덱스 $Index$, 레코드 타입 $RType$, 레코드 식별자 역할을 하는 Id로 구성되며 관계형 데이터베이스와 비교했을 때에 $Index$는 데이터베이스, $RType$는 테이블, 레코드는 행으로 비교될 수 있다. 그리고 $Token$ 필드에는 검색에 이용되는 키워드가 배열과 같은 형태로 존재한다. 다음 $NorLog_n$는 원본 로그 $OriLog_n$와 로그 메시지에 대한 메타 데이터 $LM-Data$로 분류되며, $LM-Data$는 로그를 전송한 클라우드 인프라의 호스트 $host$, 로그가 생성된 파일의 절대 경로 $file$, 로그 데이터 타입 $LType$으로 구성된다. 마지막으로 $VerTag_n$는 각 레코드의 유효성을 검증하기 위해 이용하며 CCM 방식의 암호화를 수행할 때에 해시 체인의 방식으

로 생성된다.

위 설명을 바탕으로 로그 레코드 구성의 예를 그림 11로 표현할 수 있다. 다음 그림에서 'Cloud-2015.11.14.' 인덱스에 'cloud-logs' 레코드 타입과 'apache2-logs' 레코드 타입이 존재하는 것을 볼 수 있다. 여기서 인덱스는 클라우드와 관련하여 2015년 11월 14일에 생성된 로그 메시지라는 것을 표현한다. 그리고 'cloud-logs' 레코드는 클라우드 인프라에서 생성된 로그 메시지이며 'apache2-logs' 레코드는 클라우드 사용자가 아파치 웹서버에 접속할 때에 생성된 메시지임을 나타낸다.



[그림 11] 레코드의 구성

3.6 제안 로깅 시스템 분석

이번에는 2장에서 살펴본 클라우드 환경에서의 로그 메시지 분석과 신뢰성 요구사항을 본 시스템이 얼마나 충족하는지에 대해 기술한다.

먼저 로그는 MH를 통해 LC에 모두 집중화되어 저장되며, 다양한 포맷의 로그에 정규화를 적용하고 파싱과 색인을 수행하여 분석을 수행한다. 이때, LC의 클러스터에 새로운 데이터 노드를 지속적으로 참여시킴으로써 선형적으로 인덱싱 성능을 향상시킴과 동시에 저장소의 확장성을 증가시킨다. 그리고 로그 레코드의 효율적인 유지를 위해 인덱싱된 로그 파일을 ISO 8601 표준에 따라 날짜별로 생성된 인덱스에 분리하여 저장하고 인덱스 내에 클라우드 컴퓨팅 로그, 웹 서버 로그, 부팅 로그 등과 같이 여러 유형으로 분류하여 저장한다. 로그 레코드 자체는 용량이 매우 작지만 수십, 수백만 개로 구성된 로그 파일이 수없이 생성되면 장기 보관에 문제가 발생한다. 이러한 문제는 압축을 이용하여 해결할 수 있다. 또한 다수의 데이터 노드에 인덱싱 되어 분산 저장된 로그 파일과 그 사본을 이용하여 신속한 접근과 검색이 가능하다. 로그 데이터의 무결성을 위해 소스에서 생성된 로그 메시지를 MF가 MH에 전송할 때에 SSL/TLS와 같은 통신을 이용하여 유지하며, 로그 메시지를 저장 할 때에도 검증 태그 $VerTag_n$을 이용하여 무결성을 유지한다. 감사 추적에 대한 요구사항은 AT에서 로그 파일에 접근하는 사용자에 대한 로깅을 수행함으로써 만족한다. 하지만 로그 메시지에 대한 접근 제어를 위해 적절한 접근제어 메커니즘이 적용되지 않았는데 이와 관련된 논의는 이후 4.4절에서 하기로 한다.

신뢰성 요구사항 중 메시지 출처 인증은 MF가 로그 메시지를 전송할 때 서명 수행과 로그 메시지를 생성한 호스트 이름과 같은 메타 데이터를 수집함으로써 획득할 수 있다. 그리고 전송 단계에서의 기밀성과 무결성은 인증서를 이용한 SSL/TLS 통신에서 제공하며 로그는 클라우드 인프라에서 단 한번만 생성되어 로그 파일은 신뢰하는 사용자만이 접근할 수 있으므로 고유성을 보존할 수 있다. 저장단계에서의 책임 추적성은 적절한 접근제어 메커니즘을 수행함으로써 획득하여야 하는데 이 역시 4.4절에서 논

의하기로 한다. 무결성과 기밀성은 CCM 방식을 이용한 암호화와 검증 태그 생성으로 확보할 수 있다.

위의 내용을 정리하면 표 1과 같다.

<표 1> 제안 로깅 시스템 분석

분류	요구사항	제안 로깅 시스템
분석	모든 로그의 중앙 집중화	○
	확장성 있는 로그 저장소	○
	신속한 데이터 접근과 검색	○
	다양한 로그 포맷 지원	○
	데이터 분석 작업 수행	○
	로그 레코드 유지	○
	로그 장기 보관과 필요시 복원	○
	접근 제어를 통한 분리된 데이터 접근 제어	×
	로그 무결성 보존	○
	로그 접근에 대한 감사 추적	○
신뢰성	전송 단계에서의 메시지 출처 인증	○
	전송 단계에서의 메시지 기밀성	○
	전송 단계에서의 메시지 무결성	○
	전송 단계에서의 메시지 고유성	○
	전송 단계에서의 신뢰 전송	○
	저장 단계에서의 엔트리 책임 추적성	×
	저장 단계에서의 엔트리 무결성	○
	저장 단계에서의 엔트리 기밀성	○

Ⅳ. 클라우드 컴퓨팅 환경에서의 로깅 시스템 구현

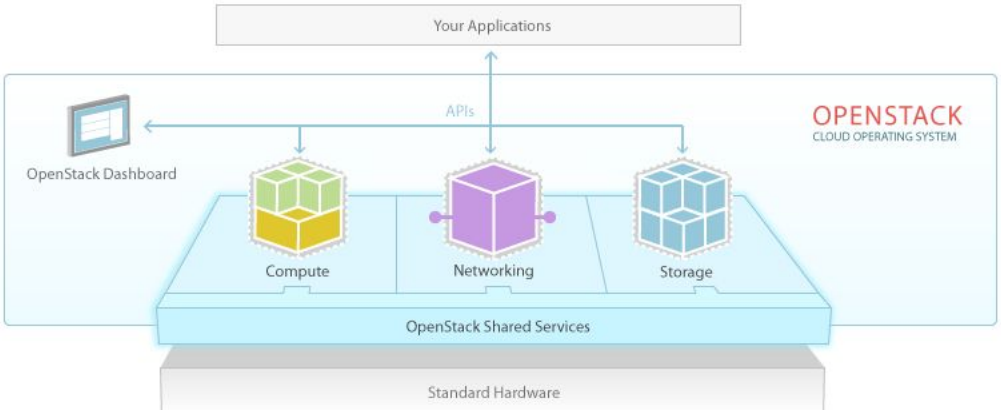
본 장에서는 이전 장에서 설계한 로깅 시스템을 구현하기 위해 필요한 오픈 소스를 선정하고 직접 구현해 봄으로써 제안 모델을 평가한다. 단 오픈 소스를 통한 구현으로 완벽하게 설계 동작을 나타내는 것에는 한계가 있기에 이러한 방식으로 구현이 가능하다는 것에 의의를 둔다.

4.1 클라우드 인프라와 로깅 시스템 구현을 위한 오픈 소스

클라우드 컴퓨팅 환경에서의 로깅 시스템을 구현하기 위해 클라우드 컴퓨팅 플랫폼으로는 오픈스택과 로깅 시스템으로는 로그스태시와 엘라스틱서치 그리고 키바나를 이용한다.

가. 오픈스택(Openstack)

오픈스택은 무료 오픈 소스 클라우드 컴퓨팅 소프트웨어 플랫폼으로 대시보드(Dashboard)나 Openstack API를 통하여 데이터센터 전반에 걸쳐 컴퓨팅, 스토리지, 네트워킹 자원의 거대한 풀을 제어한다. 또한 기업과 이기종 인프라에 이 기술을 적용할 수 있다[13].



[그림 12] 오픈스택 클라우드 컴퓨팅 플랫폼[13]

오픈스택 서비스[13]는 여러 서비스로 구성되어 있다. 주요 서비스만 살펴보면, Nova 프로젝트 이름을 가진 Compute 서비스는 오픈스택 환경에 있는 컴퓨트(compute) 인스턴스의 생명주기를 관리하며, 필요에 따라 가상머신의 생성, 스케줄링, 삭제를 담당한다. Neutron 프로젝트의 Networking 서비스는 Compute 서비스와 같은 다른 오픈스택 서비스에 대한 네트워크 연결 서비스와 사용자에게 네트워크와 부가 기능을 정의하기 위한 API를 제공한다. Cinder 프로젝트의 Block Storage 서비스는 실행중인 인스턴스에 영구적인 블록 스토리지를 제공하며, 플러그블(pluggable) 방식의 드라이버 아키텍처는 블록 스토리지 장치의 생성과 관리를 용이하게 한다. Keystone 프로젝트의 Identity 서비스는 다른 오픈스택 서비스에 대한 인증과 인가 서비스와 모든 오픈스택 서비스에 대한 종단(endpoint) 카탈로그를 제공한다. Glance 프로젝트의 Image 서비스는 가상 머신 디스크 이미지를 저장 및 검색을 수행한다. 끝으로 Horizon 프로젝트의 Dashboard 서비스는 인스턴스 생성과 IP 주소할당 그리고 접근 제어 설정과 같이 기본적인 오픈스택 서비스와 상호 작용하는 웹 기반의 포탈을 제공한다.

나. 로그스태시(Logstash)

로그스태시[14]은 실시간 파이프라이닝(Pipelining) 기능을 가진 오픈 소스 데이터 수집 엔진이다. 동적으로 다른 소스에서 생성되는 데이터를 통합할 수 있으며 원하는 방향으로 데이터를 정규화 할 수 있다.

추가적으로 로그스태시 포워더(Logstash Forwarder)[14]는 다른 곳으로의 프로세싱을 위해 로컬에 존재하는 로그 메시지를 수집하는 도구이다. 포워더는 로그스태시에게 로그 메시지를 전송하며 이때 lumberjack 프로토콜을 이용하여 전송과정에서 안정성과 낮은 지연시간, 낮은 자원 이용률 그리고 신뢰성을 제공한다.

다. 엘라스틱서치(Elasticsearch)

엘라스틱서치[15]는 루씬(Lucene)기반의 검색 서버이며, 분산 및 멀티테넌트 기능을 가진 오픈 소스 전문(full-text) 검색 엔진이다. 설정한 샤드(Shard) 수만큼 각 인덱스를 여러 샤드로 분할할 수 있으며, 각 샤드는 복수의 사본을 가질 수 있다. 이러한 샤드와 사본을 이용하여 읽기 및 검색 성능을 향상시킴으로써 고가용성을 제공한다. 이밖에도 복수의 인덱스와 하나의 인덱스 내에 여러 타입을 지정할 수 있어서 멀티테넌트 지원, HTTP RESTful API와 Native 자바 API 제공, Schema-free로 스키마를 자유롭게 구조화 할 수 있어서 사용자 커스터 마이징, 문서 지향(Document oriented)으로 모든 요소들을 JSON 문서 형식으로 저장, 실시간 처리, 작업의 일관성, 다양한 플러그인 지원 등의 특징을 가진다.

라. 키바나(Kibana)

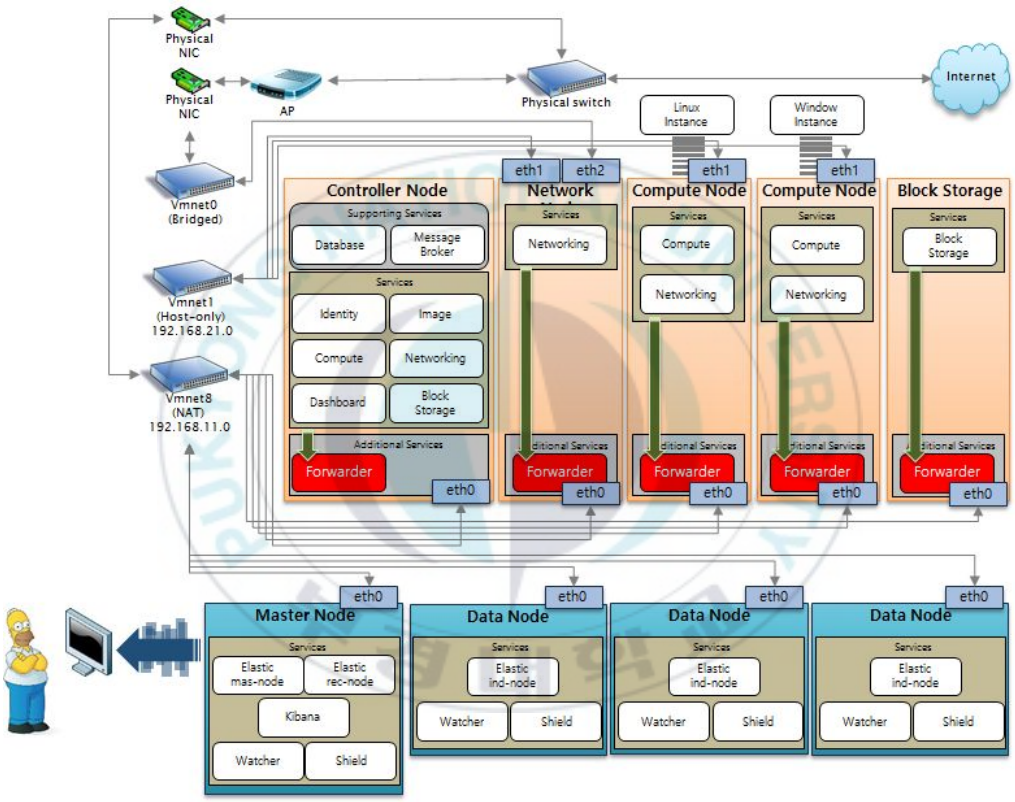
키바나[16]는 엘라스틱서치와 함께 동작하도록 설계되어 오픈 소스 분석과 시각화를 수행하기 위한 플랫폼이다. 엘라스틱서치에 저장되어 있는 데이터와 상호작용하여 검색과 뷰를 제공할 수 있으며 손쉽게 향상된 데이터 분석과 차트, 표, 지도 등과 같이 다양하게 데이터를 시각화할 수 있다.

4.2 로깅 시스템 설계

가. 전체 구현 개요

처음에 오픈스택의 각 노드에 배치된 로그스태시 포워더는 실시간으로 로컬에서 생성되는 로그와 이에 대응하는 로그의 메타 데이터를 함께 마스터 노드의 로그 스태시에 전송한다. 로그스태시는 전송받은 로그의 원본을 유지한 채 원본과 로그 메타데이터 파싱을 통해 JSON 형식의 로그 정규화를 수행하며, 이때 각 로그에 대한 해시 체인을 수행하여 설계에서 검증 태그에 해당하는 Fingerprint 필드에 각각을 추가한다. 이렇게 정규화된 로그 레코드는 Elasticsearch의 마스터 노드에게 전달된다. 마스터 노드는 동일한 클러스터를 형성하고 있는 다른 데이터 노드들에게 로그 레코드의 인덱싱을 할당하며, 인덱싱된 로그 레코드는 검색이 가능한 형태가 된다. 또한 인덱싱된 로그 레코드는 ISO 8601과 같은 형식의 날짜별로 바인딩 되어 인덱스를 형성하며 이 인덱스는 다수의 데이터 노드에 샤드의 형태로 분산 저장된다. 그리고 백업과 검색 성능의 향상을 목적으로 그 사본을 또

한 생성하여 분산 저장한다. 키바나는 이렇게 저장된 로그 레코드를 웹 인터페이스를 통해 사용자가 볼 수 있도록 하며, 사용자에게 검색과 데이터 시각화 등의 기능을 제공한다. 이와 같은 과정은 그림 13으로 나타낼 수 있다.



[그림 13] 로깅 시스템 구현 개요

나. 오픈스택 기반 클라우드 환경 구축

클라우드 컴퓨팅 플랫폼으로는 오픈스택(우분투 14.04 Kilo 버전)을 이용한다. 오픈스택을 이용한 클라우드 컴퓨팅 운영에는 컨트롤러(Controller),

System Information

Services | Compute Services | Block Storage Services | Network Agents

Filter

Name	Service	Host	Status
nova	compute	controller	Enabled
neutron	network	controller	Enabled
cinderv2	volumev2	controller	Enabled
glance	image	controller	Enabled
cinder	volume	controller	Enabled
keystone	identity (native backend)	controller	Enabled

Displaying 6 items

Services | Compute Services | Block Storage Services | Network Agents

Filter

Name	Host	Zone	Status	State	Last Updated
nova-cert	controller	internal	Enabled	Up	0 minutes
nova-consoleauth	controller	internal	Enabled	Up	0 minutes
nova-scheduler	controller	internal	Enabled	Up	0 minutes
nova-conductor	controller	internal	Enabled	Up	0 minutes
nova-compute	compute1	nova	Enabled	Up	0 minutes
nova-compute	compute2	nova	Enabled	Up	0 minutes

Displaying 6 items

Services | Compute Services | Block Storage Services | Network Agents

Filter

Name	Host	Zone	Status	State	Last Updated
cinder-scheduler	controller	nova	Enabled	Up	0 minutes
cinder-volume	block1@lvm	nova	Enabled	Up	0 minutes

Displaying 2 items

Services | Compute Services | Block Storage Services | Network Agents

Filter

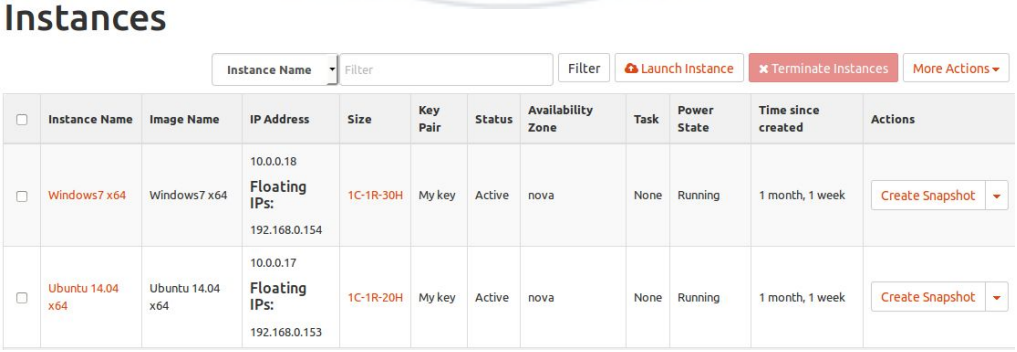
Type	Name	Host	Status	State	Last Updated
DHCP agent	neutron-dhcp-agent	network	Enabled	Up	0 minutes
L3 agent	neutron-l3-agent	network	Enabled	Up	0 minutes
Open vSwitch agent	neutron-openvswitch-agent	compute2	Enabled	Up	0 minutes
Metadata agent	neutron-metadata-agent	network	Enabled	Up	0 minutes
Open vSwitch agent	neutron-openvswitch-agent	network	Enabled	Up	0 minutes
Open vSwitch agent	neutron-openvswitch-agent	compute1	Enabled	Up	0 minutes

Displaying 6 items

[그림 14] 오픈스택 서비스

네트워크(Network), 컴퓨트(Compute), 블록 스토리지(Block Storage) 노드로 구성되는 4가지 종류의 물리 호스트가 최소 필요하다. 본 구축에서는 컨트롤러 노드 1개, 컴퓨트 노드 2개, 네트워크 노드 1개, 블록 스토리지 노드 1개로 구성되는 총 5개의 물리 호스트를 생성하여 클라우드 컴퓨팅 플랫폼을 구축하였으며, 다음 그림 14를 통해 각 노드에서 오픈스택 서비스가 정상적으로 작동함을 볼 수가 있다. 또한 그림 15에서 보이는 것처럼, 두 컴퓨트 노드에는 우분투(14.04 x64) 인스턴스(instance)와 윈도우즈 인스턴스가 하나씩 생성되어 일반 사용자 계정에 클라우드 서비스를 제공한다.

사용자는 할당된 Floating IP주소를 통하여 터미널이나 웹 인터페이스를 통하여 자신의 인스턴스를 이용할 수 있다. 이렇게 인스턴스를 이용하는 것은 물론 새로운 인스턴스의 생성 및 삭제 수정에 관한 컴퓨팅 작업, 네트워킹, 스토리지 작업 등을 수행할 때에 추가적인 로그 메시지 생성이 수반된다. 그런데 여기서 클라우드 컴퓨팅 환경은 다수의 인스턴스를 비롯한 다 수준의 클라우드 서비스가 동작하기에 로그 메시지는 광범위한 영역과 다양한 계층에서 엄청난 양으로 생성된다.



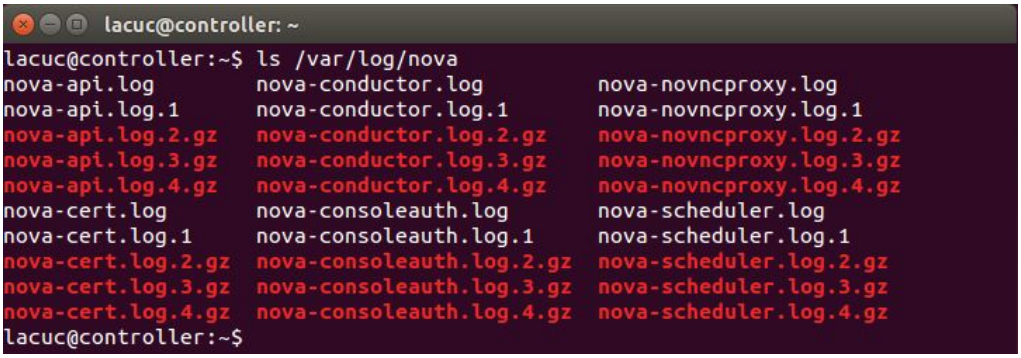
[그림 15] 현재 동작중인 두 인스턴스

오픈스택의 운영 가이드[17]에는 오픈스택이 수행되는 노드에서 획득할 수 있는 로그 메시지의 위치를 다음 표 2와 같이 나타내고 있다.

<표 2> 오픈스택 로그 위치[17]

노드 유형	서비스	로그 위치
Cloud controller	nova-*	/var/log/nova
Cloud controller	glance-*	/var/log/glance
Cloud controller	cinder-*	/var/log/cinder
Cloud controller	keystone-*	/var/log/keystone
Cloud controller	neutron-*	/var/log/neutron
Cloud controller	horizon	/var/log/apache2/
All nodes	misc (swift, dnsmasq)	/var/log/syslog
Compute nodes	libvirt	/var/log/libvirt/libvirtd.log
Compute nodes	Console (boot up messages) for VM instances:	/var/lib/nova/instances/instance-<instance id>/console.log
Block Storage nodes	cinder-volume	/var/log/cinder/cinder-volume.log

각 노드에서 수행되는 서비스별로 로그 디렉터리가 생성되며 이 디렉토리에는 다수의 로그 파일이 존재한다. 예로 그림 16은 컨트롤러 노드의 /var/log/nova 디렉터리에 있는 로그 파일들을 나타낸 것이다.



[그림 16] 컨트롤러 노드의 nova 서비스 로그 파일

이때 로그스태시 포워더는 위에 표기된 디렉터리 내에 있는 로그 파일에 로그가 추가될 때마다 마스터 노드가 생성한 인증서를 이용하여 로그스태시에 로그 메시지를 안전하게 전송한다.

다. 오픈소스 기반 로깅 시스템 구성

로깅 시스템을 구성하기 위해서 로그스태시 포워더(0.4 버전), 로그스태시(1.5.3 버전), 엘라스틱서치(1.7.1 버전), 키바나(4.1.1 버전)을 이용한다.

▪ **로그스태시 포워더**

포워더는 로그스태시에 로그 메시지를 전송하기 위해서는 다음 표 3과 같이 설정하는 것이 필요하다. JSON 형식을 이루고 있는 설정 파일은 network 객체와 file 배열로 구성되며 network 객체에는 로그 메시지를 한전하게 전송하기 위한 서버 목록과 이때 이용할 서버의 인증서 그리고 서버와의 연결을 유지하기 위한 timeout 관한 정보를 설정한다. file 배열에는 수집하려는 로그 파일의 각 경로를 기록하며 수집된 로그가 어떤 형식인지 설정할 수 있다.

<표 3> 컨트롤러에서의 포워더 설정

```
{
  # The network section covers network configuration :)
  "network":{
    # A list of downstream servers listening for our messages.
    # logstash-forwarder will pick one at random and only switch if
    # the selected one appears to be dead or unresponsive
    "servers":["192.168.11.211:5043"],
```

```
    # The path to your client ssl certificate (optional)
    #"ssl certificate": "./logstash-forwarder.crt",
    # The path to your client ssl key (optional)
    #"ssl key": "./logstash-forwarder.key",

    # The path to your trusted ssl CA file. This is used
    # to authenticate your downstream server.
    "ssl ca":"/etc/pki/tls/certs/logstash-forwarder.crt",

    # Network timeout in seconds. This is most important for
    # logstash-forwarder determining whether to stop waiting for an
    # acknowledgement from the downstream server. If an timeout is
reached,
    # logstash-forwarder will assume the connection or server is bad and
    # will connect to a server chosen at random from the servers list.
    "timeout":15
  },

  # The list of files configurations
  "files":[
    # An array of hashes. Each hash tells what paths to watch and
    # what fields to annotate on events from those paths.
    {
      "paths":[
        # single paths are fine
        "/var/log/nova/*.log"
      ],

      # A dictionary of fields to annotate on each event.
      "fields":{"type":"nova"}
    },{
      "paths":[
        "/var/log/cinder/*.log"
      ],
      "fields":{"type":"cinder"}
    },{
      "paths":[
        "/var/log/glance/*.log"
      ],
      "fields":{"type":"glance"}
    },{
      "paths":[
        "/var/log/neutron/*.log"
      ],
      "fields":{"type":"neutron"}
```

```
    },{
      "paths":[
        "/var/log/keystone/*.log"
      ],
      "fields":{"type":"keystone"}
    }
  ]
}
```

▪ 로그스태시

로그스태시는 포워더가 전달하는 로그 메시지를 처리하기 위해서는 표 4와 같은 설정이 필요하다. 설정 파일은 크게 input, filter, output 영역으로 구분된다. 먼저 input 영역은 이벤트의 소스를 명시하는 영역으로 포워더와 lumberjack 프로토콜을 이용하여 로그 메시지를 전달받으므로 프로토콜에 이용되는 포트번호와 이용되는 인증서 그리고 그 개인키의 경로를 설정한다. filter 영역은 이벤트에 대한 임시적 처리가 수행되는 부분이며 로그 메시지의 파싱을 수행하여 원본 로그에 기록된 시간으로 timestamp 필드의 인자 값을 대체하고 해시 체인을 수행하기 위하여 적절한 비밀 값과 해시 알고리즘, 필드 영역을 지정하여 설정한다. output 영역은 목적지로 이벤트 데이터를 전송하는 부분으로 엘라스틱서치를 출력 방향으로 설정하고 이때 엘라스틱서치에서 전송하는 로그 메시지를 도큐먼트로 처리할 때에 필요한 메타 데이터인 인덱스 이름, 클러스터 이름, 문서 형식을 지정하여 전달하고 엘라스틱서치가 수행되는 위치, 로그 메시지를 엘라스틱서치로 출력하기 위한 임시 노드 이름, 출력할 때의 로그 메시지 형식을 설정한다.

<표 4> 로그스태시 설정

```
input{
  lumberjack{
    # The port to listen on
```

```
    port=>5043

    # The paths to your ssl cert and key
    ssl_certificate=>"/etc/pki/tls/certs/logstash-forwarder.crt"
    ssl_key=>"/etc/pki/tls/private/logstash-forwarder.key"

    # Set this to whatever you want.

  }
}

filter{

    grok{
      patterns_dir=>"./patterns"
      match=>{"message"=>"%{TIMESTAMP_ISO8601:date}
%{NUMBER:pid:int}  %{AUDITLOGLEVEL:level}  %{NOTSPACE:module}
\[%{GREEDYDATA:program}\] %{GREEDYDATA:content}"}
    }
    date{
      match=>["date","YYYY-MM-dd HH:mm:ss.SSS"]
    }

    fingerprint{
      concatenate_sources=>true
      key=>"lacuc"
      method=>"SHA256"
      source=>["message"]
    }
}

output{
  elasticsearch{
    index=>"openstack-%{+YYYY.MM.dd}"
    host=>["localhost"]
    cluster=>"slse"
    node_name=>"o-rec"
    document_type=>"o-logs"
    codec=>json
  }
}
```

▪ 엘라스틱서치

엘라스틱서치는 로그스태시가 전달하는 로그 메시지를 다루기 위해 클러스터를 형성한다. 클러스터 slse은 마스터 노드 1개와 데이터 노드 3개를 생성하여 구성하며 마스터 노드와 데이터 노드 각각은 config/elasticsearch.yml 파일의 설정이 필요하다. 명령 수행의 창구 기능을 수행하는 마스터 노드의 설정은 다음 표 5와 같이 설정하며 동일한 클러스터를 형성하기 위해 클러스터 이름과 클러스터에 참여하는 노드의 이름, 노드의 역할을 지정하기 위해 마스터 역할 설정에 true, 데이터 노드 역할 설정에 false로 설정한다.

<표 5> 마스터 노드 설정

```
cluster.name:slse
node.name:"master“
node.master:true
node.data:false
```

반면 표 6은 데이터 노드 설정을 위한 것으로, 마스터 노드와 동일한 클러스터 이름과 클러스터에 참여하는 데이터 노드의 이름, 마스터 노드로 선출되지 않고 데이터 노드로 설정하기 위해 마스터 역할 설정에 false, 데이터 노드 역할 설정에 true로 설정한다. 그리고 데이터 노드는 http 통신 기능을 제한하여 REST API 접근을 차단하고 인덱싱과 저장을 하는 역할만을 수행하도록 한다.

<표 6> 데이터 노드 설정

```
cluster.name:slse
node.name:"data[:digit:]"
node.master:false
node.data:true
http.enabled:false
```

로그 파싱이 수행될 때에 필드 값들이 원하는 데이터 형식으로 저장되도록 맵핑을 설정하는 것이 필요하다. 명령을 통해 JSON 형식의 맵핑 스키마를 입력하여 설정할 수 있으며, 설정된 스키마는 다음 명령 'curl 'http://localhost:9200/openstack-*/_mapping?pretty''을 통해 표 7처럼 확인 할 수 있다.

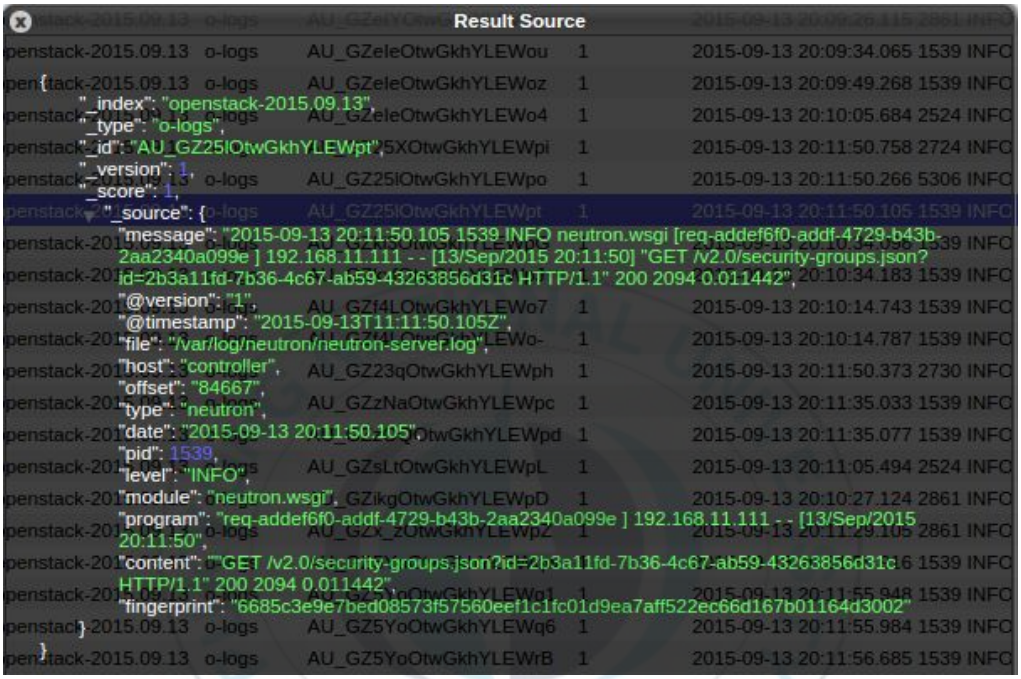
<표 7> 맵핑 설정 확인

```
{
  "openstack-2015.9.13" : {
    "mappings" : {
      "o-logs" : {
        "properties" : {
          "@timestamp" : {
            "type" : "date",
            "format" : "dateOptionalTime"
          },
          "@version" : {
            "type" : "string"
          },
          "content" : {
            "type" : "string"
          },
          "date" : {
            "type" : "string"
          },
          "file" : {
            "type" : "string"
          },
          "fingerprint" : {
            "type" : "string"
          },
          "host" : {
            "type" : "string"
          },
          "level" : {
            "type" : "string"
          },
          "message" : {
            "type" : "string"
```

```
            },
            "module" : {
              "type" : "string"
            },
            "offset" : {
              "type" : "string"
            },
            "pid" : {
              "type" : "long"
            },
            "program" : {
              "type" : "string"
            },
            "type" : {
              "type" : "string"
            }
          }
        }
      }
    }
  }
}
```

위의 모든 설정이 완료되면 엘라스틱서치의 head 플러그인을 통하여 로그 메시지와 관련 메타 데이터들에 대한 파싱이 정상적으로 수행되어 정확한 필드로 구분되는지를 그림 17처럼 확인할 수 있다. 그림에서 내장 필드인 _source 객체에 로그스태시가 전달한 로그 메시지가 저장되며, 객체 내에 message 필드에는 오픈스택 서비스에서 생성된 로그 메시지의 원본이 저장된다. 그리고 @timestamp, date, pid, level, module, program, content 필드는 원본 로그 메시지의 파싱을 통해 생성하며 객체 내 나머지 필드들은 로그에 대한 메타 데이터로 생성된 것이다. 특히 fingerprint는 표 4의 filter 영역의 설정으로 생성된 필드이며, 이것은 원본 로그 필드인 message에 대하여 SHA256 알고리즘과 비밀키를 이용하여 해시 체인을 수행한 결과 값을 나타낸다. 이 때, 설정에서 해시 체인 대상이 되는 필드를 원하는 것으로 자유롭게 선택할 수 있다. 마지막으로 _index, _type,

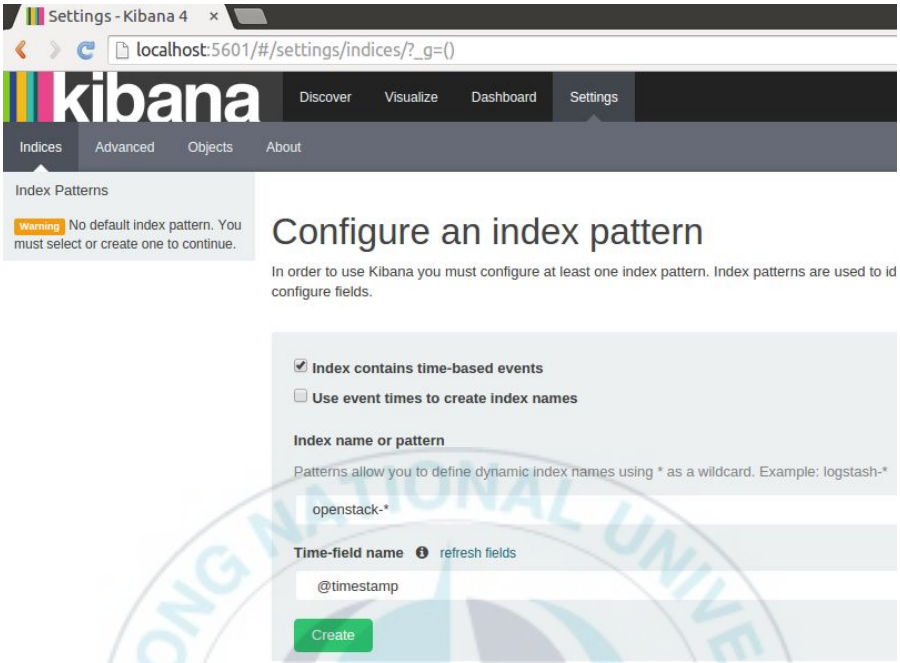
_version 필드는 로그 레코드의 도큐먼트를 관리하기 위한 필드로 활용된다.



[그림 17] 파싱이 수행된 로그 메시지

▪ 키바나

키바나에서는 특별한 설정이 필요한 것은 아니지만 초기 설정에서 불러올 인덱스 이름이나 패턴을 명시해야 한다. 다음 그림 18처럼 'openstack-' 뒤에 와일드카드 *를 이용하며 패턴으로 모든 로그 인덱스를 불러오거나 일부 특정 인덱스를 명시하여 불러와 모니터링에 이용한다.



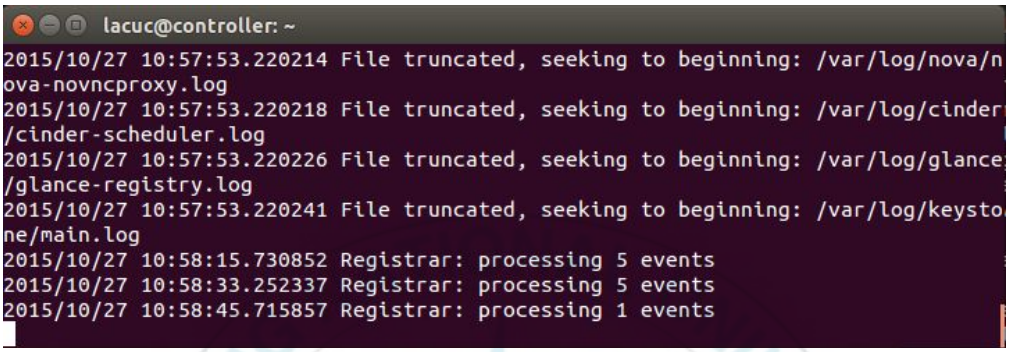
[그림 18] 인덱싱된 로그 파일 로드

4.3 로깅 시스템을 이용한 로그 모니터링

로그 메시지가 수집되는 호스트에서도 수집 상황의 일부를 확인할 수 있으며 그림 19에서처럼 호스트에서 포워더가 수집하는 로그의 로그 파일 이름과 수집되는 로그의 수량에 관한 정보를 실시간으로 확인할 수 있다.

그리고 엘라스틱서치에 의해 인덱싱되어 저장된 로그는 head 플러그인에서 날짜별로 생성된 인덱스가 다수의 샤드와 그 사본이 분산되어 저장된 모습을 그림 20으로 확인할 수 있다. 각 데이터 노드에 샤드와 사본이 저장되지만 마스터 노드에는 저장되지 않는다. o-rec 노드는 로그스태시에 의해 생성된 노드로 단순히 엘라스틱서치에 로그 메시지를 전달하기 위해

존재한다. 샤드와 그 사본은 암호화되어 저장되기 때문에, 파일 시스템 상에서는 로그 내용에 접근하는 것이 불가능하며 오직 터미널에서의 명령이나 RESTful API 명령으로만 접근할 수 있다.

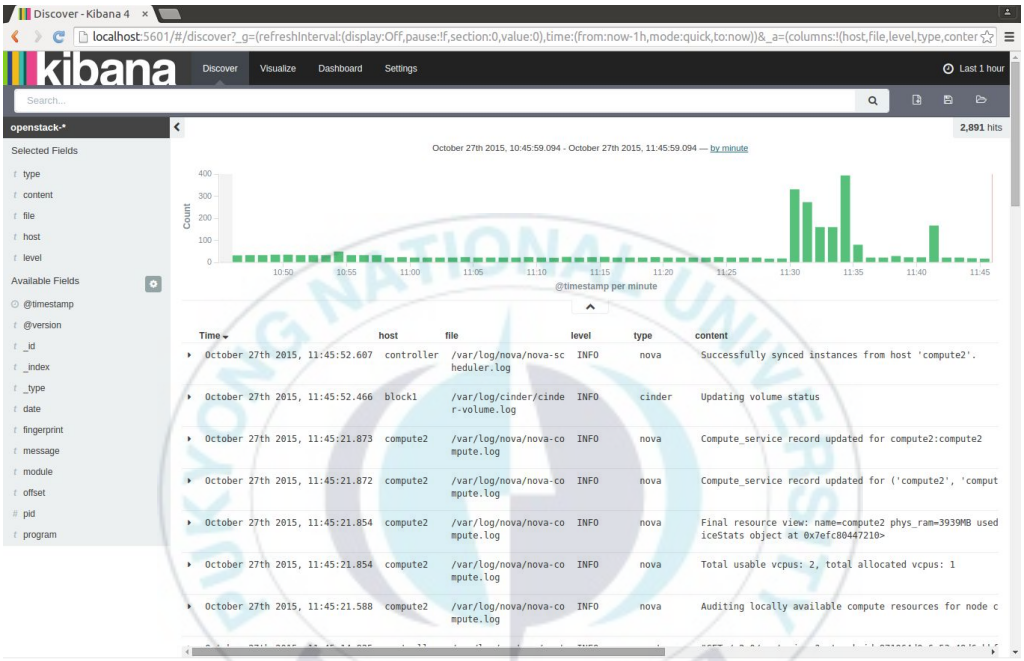


[그림 19] 컨트롤러 노드에서의 로그 수집

[그림 20] 로그 파일의 샤드와 사본

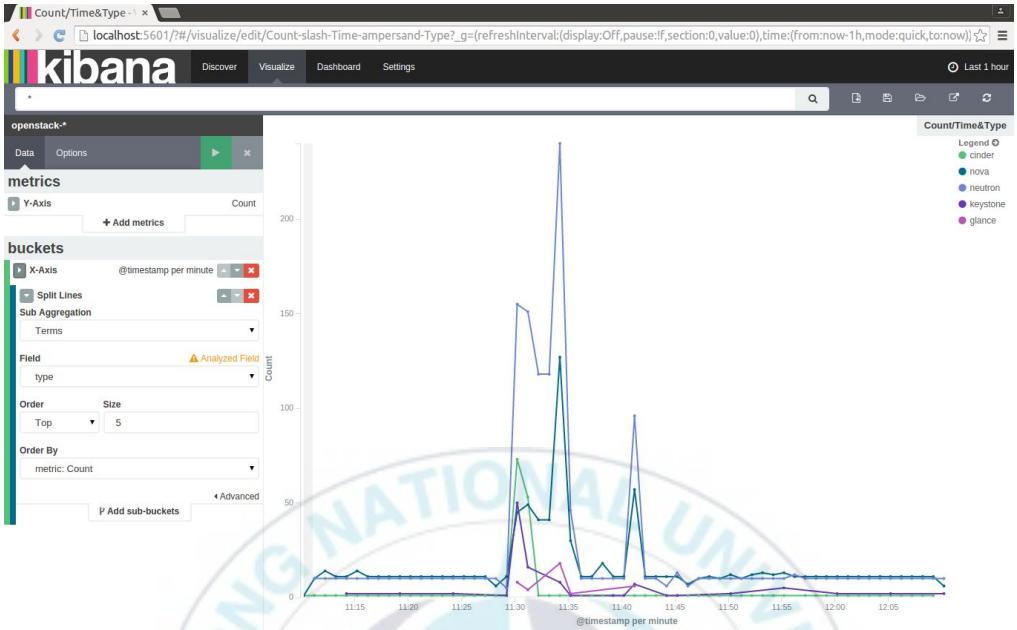
키바나에서 불러온 인덱스에 있는 로그 레코드들은 Discover 탭을 볼 수 있다. 다음 그림 21의 왼쪽 패널에서 필드를 사용자가 원하는 필드를 선택하여 값을 볼 수 있으며 선택될 필드에 따라 중앙 하단에서 각 레코드에서

필드 값을 확인할 수 있다. 그리고 상단의 검색창에서 사용자가 원하는 로그를 검색할 수 있으며 중앙 부분에 시간순서에 따라 생성된 로그의 수를 막대그래프를 통해 시각적으로 간단히 확인할 수 있다.

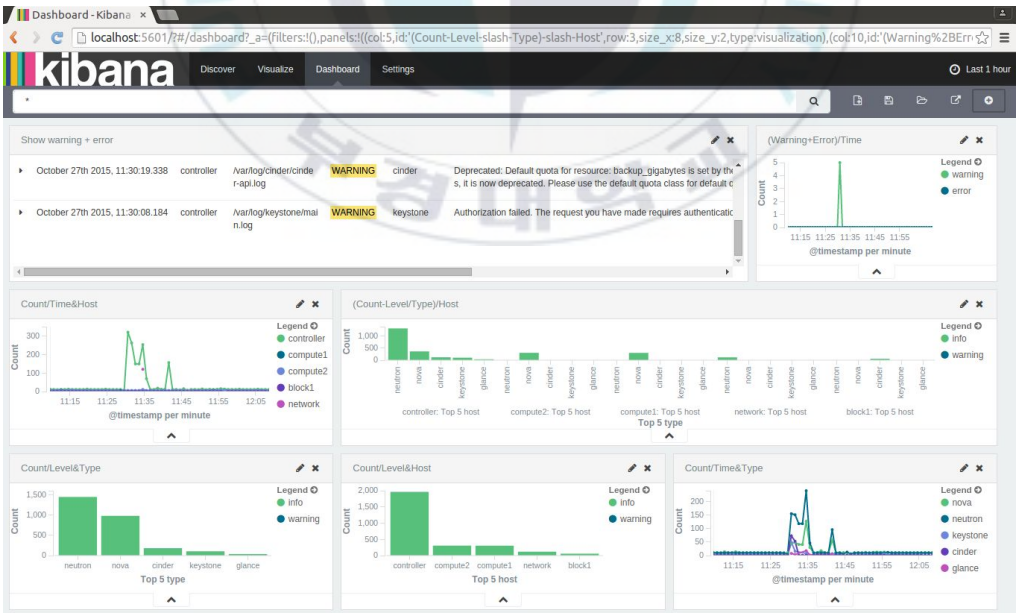


[그림 21] 키바나 Discover

두 번째 탭은 Visualize 탭으로 색인된 로그에 대한 시각화를 제공한다. 시각화 기법에는 면적그래프, 데이터 테이블, 선도표, 마크다운 위젯, 수치, 파이 그래프, 타일 맵, 수직 막대 도표를 이용할 수 있다. 시각화를 설정할 때에 뷰 옵션, 여러 종류의 통합 기법, JSON을 이용한 시각화로 좀 더 상세하게 작업을 수행할 수 있다. 그림 22는 오픈스택의 물리적인 호스트가 아니라 논리적인 서비스 단위로 로그가 얼마나 생성되었는지를 선도표로 나타낸 것이다. 이렇게 생성된 시각화 결과는 외부와의 공유를 위해서 html로 삽입할 수 있도록 하는 코드와 URL을 제공하며 이를 통하여 의사



[그림 22] 키바나 Visualize



[그림 23] 키바나 Dashboard

결정권자를 비롯한 다양한 사용자에게 보다 효과적으로 정보를 전달할 수 있다.

Discover 탭에서의 질의 결과물과 Visualize 탭에서의 시각화 결과물 각각은 저장이 가능하며, 이렇게 저장된 결과물은 Dashboard 탭에서 생성된 사용자 Dashboard에 원하는 목적에 따라 배치된다. 이를 통하여 사용자는 Dashboard에서 다수의 시각화와 질의 결과물을 한꺼번에 모니터링 할 수 있다. 그림 23은 Discover 탭과 Visualize 탭에서 생성한 검색 결과와 시각화 자료들을 이용하여 Dashboard를 생성한 하나의 예이다. 이렇게 생성된 Dashboard 또한 html 삽입 코드와 URL을 제공하여 외부 사용자와 공유가 가능하다.

4.4 로깅 시스템의 한계 및 해결방안

3.6절의 제안 로깅 시스템 분석에서 보았듯이 관련 연구에서 기술한 요구사항 대부분을 충족하였지만 로그 메시지의 접근제어와 책임 추적성을 만족시키지는 못했다. 이는 저장된 로그 데이터에 대한 접근 제어 메커니즘이 결여되었기 때문에 나타난 결과이다. 이 문제점을 해결하기 위해 클라우드 컴퓨팅 환경에 적합한 접근 제어 메커니즘 도입이라는 솔루션이 필요하지만 이것은 단순하게 해결될 수 있는 문제가 아니다. NIST에서는 클라우드 조사관이 인가되지 않고 데이터에 접근하는 개체를 분명하게 식별하는 것은 사용자의 클라우드 계정에 대한 인가와 접근 제어가 데이터 보호 규칙을 충족하지 않을 수 있기 때문에 어려운 문제라고 기술하고 있다[6]. 이러한 주장은 [18][19][20][21][22][23]에서도 뒷받침되고 있으며 이 문제를 해결하기 위해 앞으로 전문가 집단의 많은 연구가 필요할 것으로 보인다.

V. 결론 및 향후 과제

본 논문에서는 클라우드 컴퓨팅 환경에서 제공하는 로깅 서비스가 직면하고 있는 로그 분석과 신뢰성에 대한 어려움과 이를 극복하기 위한 요구사항들을 통하여 로깅 서비스가 지향해야할 방향에 대해 기술하였다. 또한 이러한 요구사항들을 부합하기 위한 로깅 시스템의 설계와 그 간접적인 구현을 수행하였다.

그 결과 설계한 로깅 시스템의 분석에서 로그 관리와 신뢰성 요구사항의 대부분을 충족하였지만 저장 단계에서의 엔트리 책임 추적성과 분리된 데이터 접근 제어를 만족시키기에는 충분하지 못하였으며, 이를 해결하기 위해서 저장 단계에서의 로그 접근 제어 메커니즘이 필요함을 확인할 수 있었다. 또한 실제 로깅 시스템의 구현에서도 오픈 소스를 이용하여 진행하다보니 설계상의 CCM을 이용한 방식으로 로깅 시스템을 완벽하게 구현하는데 한계가 있어 설계 로깅 시스템을 데모의 형식으로만 보여줄 수밖에 없었다. 그러나 제안 로깅 시스템은 클라우드 컴퓨팅 환경에서의 실질적인 로깅 서비스가 지향해야할 방향을 제시하였을 뿐만 아니라 로깅 서비스를 제공하는 주체인 클라우드 서비스 제공자(CSP)에게 로깅 서비스에 대한 새로운 모델을 제공한다. 뿐만 아니라 로그 데이터를 획득할 수 없었거나 단편적으로 획득할 수 있었을 뿐이었던 기존의 클라우드 컴퓨팅 플랫폼과 비교하여 로그 데이터에 실시간으로 종합적이고 지능적인 분석을 수행하며 로깅 서비스를 제공받는 다양한 사용자들에게 기존 로깅 서비스에서는 제공받지 못한 새로운 사용자 경험을 느끼게 해 줄 수 있을 것이다. 이는 결국에 현재 상황에 대한 넓은 통찰력을 제공할 수 있을 것이라고 사료된다.

향후에는 클라우드 컴퓨팅 환경에 적절한 로그 접근 제어 메커니즘을 구

성하여 로깅 시스템에 배치함으로써, 모든 요구사항 조건을 만족하는 로깅 시스템으로 보완할 할 것이며 이러한 제안 로깅 시스템이 얼마나 많은 클라우드 컴퓨팅 플랫폼에서 이식될 수 있는지에 대한 이식성과 시스템 성능을 향상시키기 위한 방안에 대해 연구할 계획이다.

참고문헌

[1] GARTNER, I. N. C. Gartner identifies the top 10 strategic technology trends for 2015.

[2] GENS, Frans. IDC Predictions 2015: Accelerating Innovation－and Growth－on the 3rd Platform. 2015.

[3] Burton, Betsy; Walker, Mike J. Hype Cycle for Emerging Technologies, 2015. Gartner, July, 2015.

[4] MELL, Peter; GRANCE, Tim. The NIST definition of cloud computing. National Institute of Standards and Technology, 2009, 53.6: 50.

[5] ACCORSI, Rafael. Log Data as Digital Evidence: What Secure Logging Protocols Have to Offer?. In: Computer Software and Applications Conference, 2009. COMPSAC'09. 33rd Annual IEEE International. IEEE, 2009. p. 398-403.

[6] NIST CLOUD COMPUTING FORENSIC SCIENCE WORKING GROUP, et al. NIST Cloud Computing Forensic Science Challenges (Draft NISTIR 8006)(2014).

[7] PALMER, Gary. A road map for digital forensics research-report from the first Digital Forensics Research Workshop (DFRWS). Utica, New York, 2001.

[8] MCKEMMISH, Rodney. What is Forensic Computing? Australian Institute of Criminology. 1999.

[9] KENT, Karen, et al. Guide to integrating forensic techniques into

incident response. NIST Special Publication, 2006, 800-86.

[10] PICHAN, Ameer; LAZARESCU, Mihai; SOH, Sie Teng. Cloud forensics: Technical challenges, solutions and comparative analysis. Digital Investigation, 2015, 13: 38-57.

[11] MARTY, Raffael. Cloud application logging for forensics. In: Proceedings of the 2011 ACM Symposium on Applied Computing. ACM, 2011. p. 178-184.

[12] DWORKIN, Morris. Recommendation for block cipher modes of operation: The CCM mode for authentication and confidentiality. US Department of Commerce, Technology Administration, National Institute of Standards and Technology, 2004.

[13] www.openstack.org. OpenStack Open Source Cloud Computing Software.

[14] www.elastic.co/products/logstash. elastic.

[15] www.elastic.co/products/elasticsearch. elastic.

[16] www.elastic.co/products/kibana. elastic.

[17] docs.openstack.org/ops/. OpenStack Open Source Cloud Computing Software.

[18] RUAN, Keyun, et al. Cloud forensics definitions and critical criteria for cloud forensic capability: An overview of survey results. Digital Investigation, 2013, 10.1: 34-43.

[19] RUAN, Keyun, et al. Cloud forensics. In: Advances in digital forensics VII. Springer Berlin Heidelberg, 2011. p. 35-46.

[20] RUAN, Keyun; CARTHY, Joe. Cloud computing reference architecture and its forensic implications: A preliminary analysis. In:

Digital Forensics and Cyber Crime. Springer Berlin Heidelberg, 2013. p. 1-21.

[21] RUAN, Keyun, et al. Key terms for service level agreements to support cloud forensics. In: Advances in Digital Forensics VIII. Springer Berlin Heidelberg, 2012. p. 201-212.

[22] RUAN, Keyun; CARTHY, Joe. Cloud Forensic Maturity Model. In: Digital Forensics and Cyber Crime. Springer Berlin Heidelberg, 2013. p. 22-41.

[23] CHEN, Yanpei; PAXSON, Vern; KATZ, Randy H. What's new about cloud computing security. University of California, Berkeley Report No. UCB/EECS-2010-5 January, 2010, 20.2010: 2010-5.

감사의 글

이제 석사 생활 막바지에 이르니 여러 감정이 뒤엉켜 복잡한 생각이 들기도 하지만 차분하게 뒤를 돌이켜 보면 많은 사람들의 도움이 있었다는 것을 새삼 느끼게 됩니다. 그래서 작지만 이 기회를 빌려 감사의 마음을 전하려 합니다.

가장 먼저, 저를 오랜 시간동안 뒷바라지 해주시며 기다려 주신 어머니에게 감사의 뜻을 표합니다. 가까이 있고 자주 보기 때문인지 표현하기 더 힘들고 어색해서 잘하지 않게 되지만 항상 고맙다는 말 전하고 싶습니다.

그리고 저를 비롯한 연구실 구성원을 항상 친근하고 편하게 보듬어주시며 진심어린 조언과 올바른 연구 방향으로 이끌어 주신 신상욱 교수님, 멀리서 묵묵히 지켜봐주신 이경현 교수님, 처음 정보보호학으로 저를 이끌어 주신 신원 교수님에게 감사함을 표합니다.

또한 낯선 연구실로 처음 들어왔을 때에 따뜻하게 맞이해준 태림 선배와 수빈 선배 그리고 주영 선배를 비롯한 연구실 선배들, 소정이, 완석이, 효민이, 기웅이를 비롯한 연구실 동생들, 학부 때부터 석사 과정까지 함께 동고동락한 명학이 형, 서로에게 힘이 되어준 수환이와 은아, 보면 볼수록 좋은 우주최강 여신 다빈이, 다말하기 힘든 모든 분들까지

정말 감사하고 행복하셨으면 좋겠습니다.

2016.02

이 병 도 드림