J. Soc. Korea Ind. Syst. Eng Vol. 39, No. 2 : 129-137, June 2016
http://dx.doi.org/10.11627/jkise.2016.39.2.129
ISSN : 2005-0461(print)
ISSN : 2287-7975(online)

Correlation Analysis of Event Logs for System Fault Detection

Ju-Won Park*† · Eunhye Kim** · Jaekeun Yeom* · Sungho Kim*

*Div. of Supercomputing, KISTI
**Hyper-connected Communication Research Lab. ETRI

시스템 결함 분석을 위한 이벤트 로그 연관성에 관한 연구

박주원*† · 김은혜** · 염재근* · 김성호*

*한국과학기술정보연구원 슈퍼컴퓨팅본부
**한국전자통신연구원 초연결통신연구소

To identify the cause of the error and maintain the health of system, an administrator usually analyzes event log data since it contains useful information to infer the cause of the error. However, because today's systems are huge and complex, it is almost impossible for administrators to manually analyze event log files to identify the cause of an error. In particular, as OpenStack, which is being widely used as cloud management system, operates with various service modules being linked to multiple servers, it is hard to access each node and analyze event log messages for each service module in the case of an error. For this, in this paper, we propose a novel message-based log analysis method that enables the administrator to find the cause of an error quickly. Specifically, the proposed method 1) consolidates event log data generated from system level and application service level, 2) clusters the consolidated data based on messages, and 3) analyzes interrelations among message groups in order to promptly identify the cause of a system error. This study has great significance in the following three aspects. First, the root cause of the error can be identified by collecting event logs of both system level and application service level and analyzing interrelations among the logs. Second, administrators do not need to classify messages for training since unsupervised learning of event log messages is applied. Third, using Dynamic Time Warping, an algorithm for measuring similarity of dynamic patterns over time increases accuracy of analysis on patterns generated from distributed system in which time synchronization is not exactly consistent.

Keywords : Correlation Analysis, Unsupervised Learning, System Fault Detection

1. 서 론

일반적으로 시스템 관리자는 시스템에 결함이 발생한 경우 결함의 원인을 파악하기 위해 이벤트 로그 데이터를 분석한다. 이벤트 로그 파일에는 Warning, Error, Critical 등 결함의 등급을 의미하는 Severity level 값뿐만 아니라 결함의 원인을 추론할 수 있는 메시지를 포함하고 있기 때문에 시스템 관리자는 이벤트 로그의 메시지를 분석함으로써 결함의 발생 원인을 추론할 수 있다. 그러나 최근 들어 시스템의 규모가 커지고 복잡해짐에 따라 발생하는 이벤트 로그 데이터의 양이 많고 시스템 구성 모듈간 관계도 복잡해 관리자가 이벤트 로그 파일을 하나하나 분석하여 결함 원인을 파악하는 것은 거의 불가능하다[7]. 특히, 클라우드 관리 시스템으로 많이 활용되는 오픈스택(OpenStack)[6]의 경우 다수의 서버에 NOVA, NEUTRON, KEYSTONE 등 다양한 서비스 모듈이 연계되어 실행되

Received 18 April 2016; Finally Revised 15 June 2016; Accepted 17 June 2016
† Corresponding Author : juwon.park@kisti.re.kr

기 때문에 결함 발생 시 각 노드에 접속하여 서비스 모듈별로 이벤트 로그 메시지를 분석하는 것은 매우 어려운 작업이다. 또한 운영 체제, 장치 드라이버와 같은 시스템 레벨에서 발생하는 결함이 응용 서비스 레벨에 영향을 주기 때문에 시스템 레벨의 이벤트 로그와 응용 서비스 레벨의 이벤트 로그를 통합하여 분석해야 결함 원인을 파악할 수 있다.

본 논문에서는 시스템뿐만 아니라 응용 프로그램에서 발생하는 이벤트 로그 데이터를 하나의 서버에 수집하고 이를 메시지 기반으로 유사한 로그를 군집화하고 연관성을 분석함으로써 관리자가 결함의 원인을 신속히 파악할 수 있는 메시지 기반 로그 분석 방안을 제시한다. 이를 위해 먼저, 다수의 서버에서 발생하는 이벤트 로그 데이터를 수집하여 하나의 수집 서버로 전송함으로써 분산된 이벤트 로그 데이터를 수집한다. 둘째, 수집된 이벤트 로그 데이터를 메시지 기반으로 분석하기 위해 정규화를 진행한다. 셋째, 정규화 된 메시지를 계층적 군집(Hierarchical Clustering) 기법[4]을 통해 군집화한다. 마지막으로 Dynamic Time Warping(DTW) 기법[12]을 이용하여 군집화된 메시지 그룹간의 연관성을 분석한다.

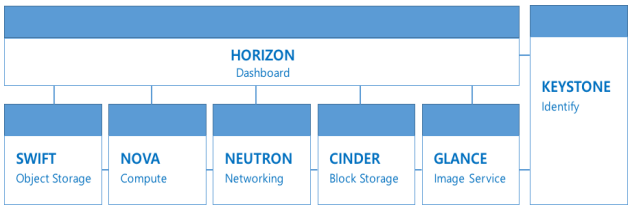
본 논문은 다음 3가지 측면에서 의의가 있다. 먼저, 시스템 레벨과 응용 서비스 레벨의 이벤트 로그를 모두 수집하여 로그간 연관성을 분석함으로써 결함 발생의 근본 원인을 파악할 수 있다. 둘째, 이벤트 로그 메시지를 비지도 학습(unsupervised learning) 기반으로 그룹화함으로써 모델 생성시 관리자가 메시지를 분류해줘야 하는 어려움을 해결할 수 있다. 셋째, 시간에 따른 동적 패턴의 유사도를 분석하는 DTW 기법을 활용함으로써 시간 동기화가 정확히 일치하지 않는 분산 시스템에서 발생하는 발생 패턴을 정확히 분석할 수 있다.

본 논문의 제 2장에서는 관련 연구에 대해 살펴본다. 제 3장에서는 본 논문에서 제시한 메시지 기반 연관성 분석 프레임워크를 설명하고 제 4장에서는 오픈스택 시스템을 대상으로 구현 방안을 제시한다. 제 5장에서는 실제 운영 중인 오픈스택의 이벤트 로그 데이터를 수집하여 분석한 결과를 제시하고 마지막으로 제 6장에서 결론을 맺는다.

2. 관련 연구

2.1 오픈스택

최근 대규모의 클라우드 시스템을 구축하고 서비스하기 위해 오픈스택을 많이 활용한다. 오픈스택은 컴퓨팅, 네트워킹, 스토리지 자원을 대규모 풀(pool) 형태로 구성하여 사용자의 요청시 자원을 동적으로 구성하여 제공하기 위한 오픈 클라우드 관리 시스템이다[6]. <Figure 1>은 오픈스택의 주요 구성 서비스를 보여주는 것으로 7개의 서비스가 유기적으로 연동되어 서비스된다.



<Figure 1> OpenStack Service Modules

- **NOVA** : 호스트 노드의 CPU, 메모리 등 컴퓨팅 자원을 이용하여 가상 노드를 만들기 위한 서비스로써 모든 컴퓨팅 노드에 설치된다.
- **NEUTRON** : 가상 노드의 네트워크 연결을 지원하기 위한 서비스로써 네트워크 노드와 컴퓨팅 노드에 설치된다. 일반적으로 네트워크 노드에 설치되는 l3 에이전트는 DHCP를 통해 동적 IP를 할당하며 컴퓨팅 노드에 설치되는 l2 에이전트는 호스트 노드에 브릿지를 생성하여 하나의 네트워크 카드를 다수의 가상 노드에서 공유하도록 지원한다.
- **CINDER** : 블럭(Block) 스토리지를 제공하는 서비스로써 가상 노드의 기본 저장 영역을 제공한다.
- **SWIFT** : 오브젝트(Object) 스토리지를 제공하는 서비스로써 이를 통해 아마존의 S3와 같은 서비스를 제공할 수 있다.
- **GLANCE** : 가상 노드 이미지를 관리하기 위한 서비스로써 다양한 형태의 가상 노드를 이미지로 생성하여 관리할 수 있다.
- **KEYSTONE** : 사용자의 접근 관리 및 인증을 위한 서비스를 제공한다.
- **HORIZON** : GUI를 통해 관리자가 쉽게 오픈스택 운영 및 관리를 할 수 있도록 구현된 서비스로써 웹 인터페이스를 통해 제공된다.

2.2 로그 메시지 포맷

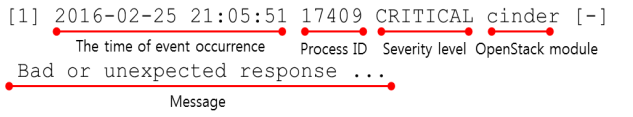
본 논문에서는 시스템 레벨과 응용 서비스 레벨에서 발생한 이벤트 로그 데이터를 통합하여 분석함으로써 결함의 원인을 파악하고자 한다. 이를 위해 시스로그데몬(syslogd)에서 출력되는 로그 데이터와 오픈스택에서 출력되는 로그 데이터의 포맷을 살펴본다. 먼저 시스로그 데이터는 Gerhards[2]에서 데이터 포맷과 전송 방법 등을 규정하고 있으며 본 논문에서는 헤더 정보를 중심으로

<Table 1>에서 제시한 포맷으로 이벤트 로그 정보를 수집한다.

<Table 1> Syslog Format

Field	Description
ID	Event log message ID
Received Time	The received time of the syslog event in server.
Device Reported Time	The reported time of the syslog event in device.
Facility	The facility code to specify the type of program that is logging the message.
Severity Level	The severity code to indicate relative importance.
Fromhost	The machine that originally sent the syslog event.
Message	A free-form message that provides information about the event.

둘째, 오픈스택 로그 데이터는 <Figure 2>와 같이 이벤트 발생 시각, 프로세스 ID, severity level, 이벤트 발생한 오픈스택 모듈, 메시지 총 5개의 필드로 구분되어 출력된다. 로그 발생 시각은 "년-월-일 시:분:초" 포맷으로 출력되며 severity level은 DEBUG, INFO, AUDIT, WARNING, ERROR, CRITICAL, TRACE 총 7단계로 구분된다. 오픈스택 이벤트 로그의 큰 특징 중 하나인 TRACE 로그 메시지는 결함의 원인을 파악하기 위해 역추적한 결과를 출력한 메시지로써 결함의 원인을 유추하는데 유용한 정보를 제공한다.



<Figure 2> OpenStack Log Format

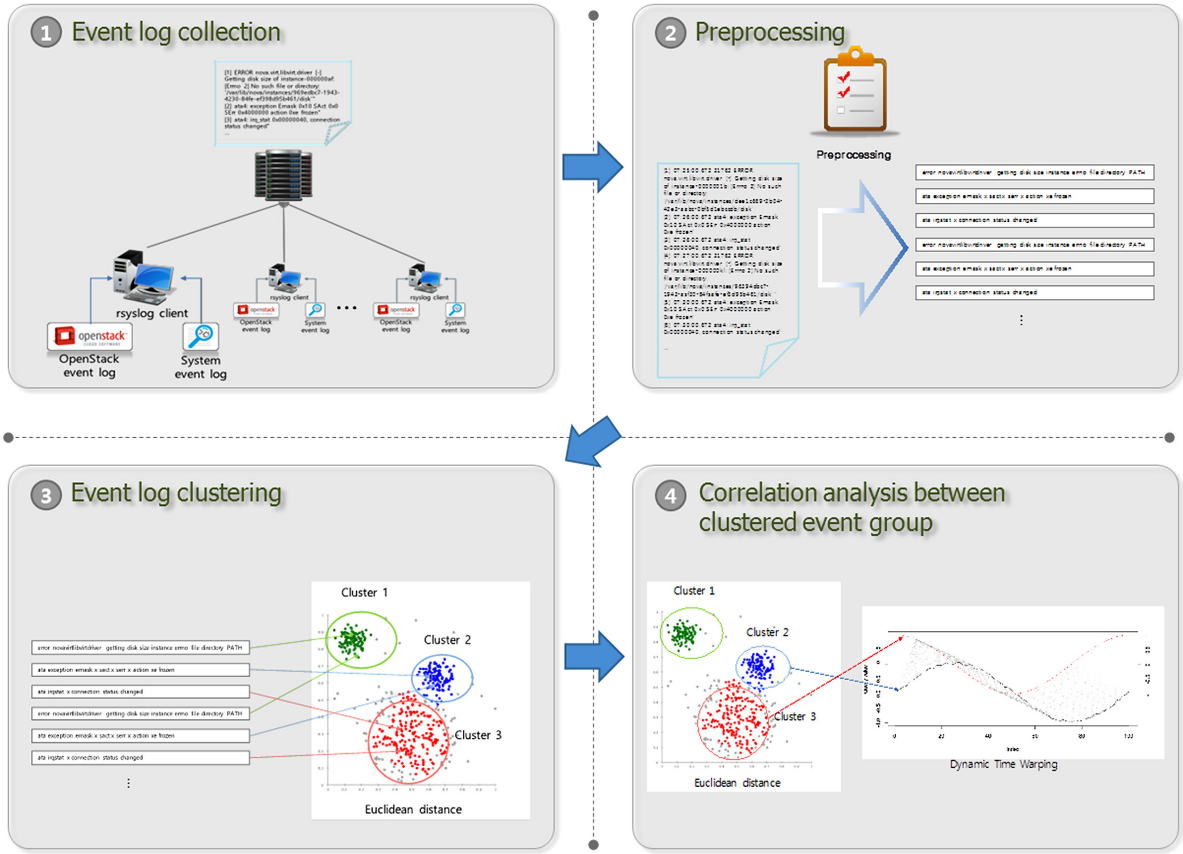
2.3 머신 러닝 기반 시스템 로그 분석 기법

최근 산업 분야에 데이터 마이닝 분야에서 연구되고 있는 기법들이 다양하게 활용되고 있다[3, 13]. Joo and choi [3]에서는 군집화 및 프로세스 마이닝을 기반으로 커리큘럼 마이닝 프레임워크를 제안하였으며 Yoo[13]에서는 군집분석을 활용하여 제품 플랫폼을 구성할 공동모듈 선정 방법을 제시하였다. 특히, 슈퍼컴퓨터를 비롯한 대규모 클러스터 시스템의 시스템 로그 분석을 위해 머신 러닝 기법을 활용한 논문이 증가하고 있다[5, 8, 11, 14-15]. Pitakrat et al.[8]에서는 시스템 이벤트 로그를 분류하고 머신러닝 기법을 이용하여 장애를 예측하는 SCAPE 기법을 제시하였다. 이를 위해 먼저 중복 데이터를 필터링하여 데이터양을 줄이고 지도 학습(supervised learning) 기법을 사용하여 이벤트 로그를 분류한다. 이렇게 분류된 데이터를 기반으로 이벤트 발생 시간 패턴을 검출하고 향후 발생할 이벤트 발생 시간을 예측하는 것이다. Oliner and Stearley[5]에서는 5개의 슈퍼컴퓨터에서 발생하는 이벤트 로그를 수집하고 알람 메시지를 구분하여 필터링함으로써 이벤트 로그 발생 패턴을 분석한 결과를 제시하였다. Zheng et al.[14]에서는 미국 아르곤 국립 연구소에 있는 Intrepid 시스템(IBM Blue Gene/P)에서 발생하는 이벤트 로그 정보를 수집하여 중요 이벤트(Fatal event)가 발생하기 전에 발생하는 이벤트 로그를 Genetic algorithm 기법으로 분석한 결과를 제시하였다. Sahoo et al.[11]에서는 대규모 클러스터 시스템의 능동적(proactive) 예측 및 관리 시스템을 제시하였다. 이를 위해 대규모의 클러스터 시스템에서 발생하는 로그를 수집하여 6개의 이벤트 타입과 4개의 클래스로 분류하고 time-series model, rule-based classification model, Bayesian network model을 적용하여 예측한 실험 결과를 제시하였다. Zheng et al.[15]에서는 필터링 작업 시에 중복된 정보는 제거하고 패턴 분석의 필수적인 정보는 보존함으로써 예측의 정확도를 향상시켰다.

본 연구는 기존 연구와 비교해 다음 3가지 측면에서 의의를 찾을 수 있다. 먼저, 시스로그데몬에서 생성되는 이벤트 로그 메시지를 분석함으로써 x86 기반 클러스터 시스템에 적용이 가능하다. 기존의 많은 연구에서는 IBM Blue Gene 시스템의 이벤트 로그 데이터를 활용하여 분석하였다[5, 8, 15]. IBM POWER 시스템의 경우 이벤트 로그 메시지가 시스로그데몬에 비해 세분화되어 분류 및 예측에 효율적이지만 일반적으로 많이 활용되는 x86 기반 서버에는 적용할 수 없다는 문제가 있다. 둘째, 비지도 학습 기반의 머신 러닝 기법을 활용하여 모델 훈련 시 사용자가 이벤트를 분류할 필요가 없다. Oliner and Stearley [5], Pitakrat et al.[8]의 경우 이벤트 로그 메시지를 사용자가 직접 분류하는 지도 학습 기반의 머신 러닝 기법을 사용하여 이벤트 로그 데이터가 대량으로 생성되는 대규모의 분산 시스템에 적용하는데 어려움이 있다. 셋째, 시스템 레벨의 이벤트 로그와 응용 서비스 레벨의 이벤트 로그를 통합하여 연관성 분석 결과를 제시한다. 기존 연구에서는 시스템 레벨에서 발생하는 이벤트 로그 데이터만을 대상으로 메시지 분류 및 결함 예측 결과를 제시할 뿐 메시지 간의 연관 관계를 제시하지는 못했다[5, 8, 11, 14-15].

3. 제안 기법

슈퍼컴퓨터와 같이 대규모 시스템에서 신속한 결함 원



<Figure 3> Overview of the Proposed Framework

인을 분석하기 위해서는 시스템 레벨과 응용 서비스 레벨의 이벤트 로그를 모두 수집하고 통합하여 로그간 연관성을 파악해야 한다. 이를 위해 본 논문에서는 이벤트 로그 기반 연관성 분석 프레임워크를 제시하며 <Figure 3>에서 보는 바와 같이 4단계로 구분할 수 있다.

첫째, 분산된 다수의 서버에서 발생하는 이벤트 로그 데이터들을 하나의 서버로 수집한다. 둘째, 수집된 이벤트 로그 데이터를 정규화한다. 즉, 수집된 데이터에 필터링 기법을 적용하여 중복된 이벤트 로그를 제거하고 결함의 원인을 유추할 수 있는 메시지 부분을 추출하여 정규화시킨다. 셋째, 정규화된 로그 데이터를 메시지 기반으로 군집화한다. 이를 위해 먼저 메시지 전체를 기준으로 단어별로 분류한 후 단어의 사용 여부 및 사용 빈도수를 나타내는 메트릭 형태로 변환한다. 이렇게 변환된 메트릭의 유클리디안 거리(Euclidean distance)를 기반으로 계층적 군집분석을 적용하여 유사도에 따라 군집화한다. 넷째, DTW 분석 기법을 기반으로 군집화된 메시지 그룹간의 연관성을 분석한다. 일반적으로 동시에 발생한 로그 메시지들은 관련성이 높기 때문에 군집화된 메시지의 발생 시간 패턴을 비교함으로써 메시지간 연관성을 분석할 수 있다.

4. 구 현

본 논문에서는 최근 클라우드 서비스를 위해 많이 활용되고 있는 오픈스택 시스템을 대상으로 시스로그데몬에서 출력되는 이벤트 로그(시스템 레벨의 로그 데이터)와 각 서비스 모듈에서 출력되는 이벤트 로그(응용 서비스 레벨의 로그 데이터)를 수집하고 분석하여 이벤트 로그간 연관성을 제시한다.

4.1 로그 수집

오픈스택은 제 2.1절에서 언급한 바와 같이 다수의 서비스 모듈이 연동되어 실행되는 클라우드 관리 플랫폼으로 일반적으로 하나의 서버가 아닌 다수의 서버에 분산되어 실행된다. 또한 실행되는 서비스에 따라 서로 다른 파일에 로그가 출력된다. 예를 들어, 컴퓨팅 노드에 설치되는 NOVA 서비스의 경우 /var/log/nova 파일에, NEUTRON의 경우 /var/log/neutron 파일에 이벤트 로그가 각각 출력된다. 이와 같이 다수의 서버에 다수의 파일로 혼재되어 있는 로그 파일을 하나의 서버로 수집하기 위해 본 논문에서는 시스로그데몬을 활용한다. 즉, 오픈스택 이벤트 로그

메시지를 OpenStack[6]에서 제시한 포맷으로 시스템 로그에 출력하고 이를 rsyslog 클라이언트[10]를 통해 서버로 전송한다. 이를 위해 먼저, rsyslog 서버를 <Figure 4>와 같은 설정을 통해 활성화하여 rsyslog 클라이언트로부터 이벤트 로그 데이터를 수집하고 이를 데이터베이스에 저장한다.

```
$ModLoad immark
$ModeLoad imtcp
$InputTCPServerRun portNum
$ModLoad ommysql
 *.*        :        ommysql:database-server,data-
base-name,database-userid,database-password
```

<Figure 4> Rsyslog Server Configuration

둘째, 오픈스택에서 출력되는 메시지를 시스로그 형태로 시스템 로그 파일에 출력되도록 오픈스택의 각 서비스 설정 파일에 <Figure 5>와 같은 설정을 추가한다.

```
use_syslog=True
syslog_log_facility=LOG_LOCAL0
```

<Figure 5> OpenStack Configuration

셋째, 시스템 로그에 출력된 로그 메시지를 수집하여 rsyslog 서버로 전송하기 위해 오픈스택을 구성하는 모든 노드의 rsyslog.conf 파일에 <Figure 6>의 설정을 추가하여 rsyslog 클라이언트를 활성화한다.

```
*.* @@rsyslog_server_ip_address:portNum
```

<Figure 6> Rsyslog Client Configuration.

이와 같이 3단계의 설정을 NOVA 서비스에 추가할 경우 이벤트 로그 메시지가 /var/log/nova가 아닌 /var/log/message에 Facility 값을 16(LOG_LOCAL0)으로 설정되어 출력됨을 확인할 수 있다.

4.2 전처리

오픈스택은 분산된 다수의 서버에서 다양한 서비스 모듈이 실행되고 서로 연동되어 운영되기 때문에 이벤트 로그 메시지의 종류가 다양하고 수도 매우 많다. 그러므로 필터링을 통해 중복된 이벤트 로그 메시지를 제거하고 메시지 필드의 내용 중에서 중요한 메시지만을 추출하여 정규화하는 전처리 작업이 필요하다. Pitakrat et al. [8]에서 제시한 바와 같이 추출된 메시지 내용을 의미론적으로 유사한 이벤트 로그로 군집화하기 위해 다음 정규화 과정을 실행한다.

먼저, 모든 문자를 소문자로 변환한다. 대소문자는 의미론적으로 구분이 없음에도 불구하고 대부분의 프로그래밍 언어에서 구분하여 처리하는 경우가 많다. 특히, 데이터 마이닝 분야에서 많이 활용되는 언어 중 하나인 R [9]의 경우 대소문자를 구분하여 비교하기 때문에 모든 문자를 소문자로 변환한 후 유사성을 계산한다. 둘째, 의미를 담고 있는 단어가 아닌 기호, 숫자, 조사 등을 제거한다. 예를 들어, -, +, ‘, : 와 같은 기호나 a, the, of, such 와 같은 관사 및 조사를 제거함으로써 군집화의 정확도를 높일 수 있다. 셋째, ID와 같은 고유값을 제거한다. 오픈스택의 경우 모든 가상 인스턴스에 ID를 부여하여 관리한다. 그러므로 동일한 현상의 이벤트 로그 메시지에도 각각 다른 인스턴스 ID가 포함되어 있기 때문에 ID을 제거하여 정확도를 높인다. 넷째, 디렉토리 경로는 PATH라는 단어로 대체한다. 오픈스택의 로그에는 파일의 위치를 표기하는 디렉토리 경로가 다수 포함되어 있으나 의미론적 입장에서는 파일의 위치를 설명하는 부수적인 의미를 내포하기 때문에 디렉토리 경로를 PATH라는 단어로 변경하여 군집화를 진행한다.

4.3 메시지 기반 군집화

전처리 작업을 통해 정규화된 메시지를 의미론적으로 유사한 것으로 군집화하기 위해 먼저 메시지 전체를 단어별로 분류하고 단어의 사용여부와 빈도수를 나타내는 메트릭으로 변환한다. 이를 기반으로 <Table 2>에서 보는 바와 같이 계층적 군집분석 기법을 통해 유사도를 계산하고 유사도가 일정값(threshold)보다 큰 경우 하나의 클러스터를 구성함으로써 군집화한다.

<Table 2> Context-aware Message Clustering

Input		Event log messages. Let $M = \{m_1, \cdots, m_n\}$ be a set of messages and $T = \{t_1, \cdots, t_p\}$ be the set of distinct terms in D.
Step	1	Compute the similarity between all pairs of clusters. $S(t_\alpha, t_\beta) = \left(\sum_{t=1}^{p} \lvert f_{\alpha,t} - f_{\beta,t} \rvert^2\right)^{1/2}$ where two messages m_α and m_β is represented by their term vectors t_α and t_β respectively, and $f_{m,t}$ denotes the frequency of term $t \in T$ in message $m \in M$.
	2	Merge iteratively the most similar two groups.
	3	Repeat step 2 until all the data are merged into a single cluster.
	4	Construct the clusters of messages : $S(t_\alpha, t_\beta) \geq \sigma$
Output		k clusters of messages

4.4 DTW 기법을 통한 군집화된 메시지 그룹 간 연관성 분석

일반적으로 동일한 결함에 의한 이벤트 로그 메시지들은 동일한 시간에 발생한다. 그러므로 이벤트 로그 메시지가 발생한 시간의 패턴을 비교 분석하여 메시지간 연관성을 파악할 수 있다. 본 논문에서는 이벤트 발생 패턴을 비교 분석하기 위해 DTW 기법을 활용한다. DTW는 시간 길이가 서로 다른 두 동적 패턴 사이의 유사도를 판별하기 위한 알고리즘으로 두 개의 순차 데이터의 시간 길이를 왜곡시킴으로써 두 패턴의 최적의 정합(matching)을 구하고, 해당 정합에서의 두 데이터 사이의 거리를 계산한다. 예를 들어, 길이가 각각 p, q인 두 이벤트 발생 시간 패턴 $X = (x_1, x_2, \cdots, x_p)$, $Y = (y_1, y_2, \cdots, y_q)$가 주어졌을 때, 패턴의 정렬을 위해 행렬이 만들어지며, 이 행렬의 (i, j)번째 요소는 유클라디안 거리 $(x_i - y_j)^2$를 이용한 $d(x_i, y_j)$를 포함한다. DTW 알고리즘은 연속성, 단조성의 조건을 이용하며, 두 패턴 간의 누적 거리 $D(i, j)$는 다음과 같이 계산된다.

$$D(i, j) = \min \begin{Bmatrix} D(i-1, j-1) \\ D(i-1, j) \\ D(i, j-1) \end{Bmatrix} + d(x_i, y_j)$$

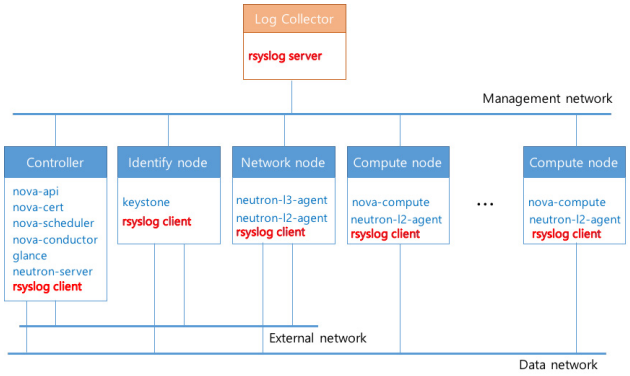
본 논문에서는 군집화된 메시지 그룹들의 발생 패턴 유사도를 DTW를 이용하여 계산하고 이를 기반으로 로그 연관성을 분석함으로써 동일한 결함 원인에 의해 발생하는 메시지 형태를 파악할 수 있다.

5. 실험 결과

5.1 실험 환경

본 논문에서는 한국과학기술정보연구원에서 실제 운영하고 있는 오픈스택 플랫폼의 이벤트 로그 데이터를 하루 동안 수집하여 제 4장에서 제시한 방법으로 R을 통해 분석한 결과를 제시한다. <Figure 7>은 구성된 오픈스택 플랫폼의 구성도를 보여준다.

<Figure 7>에서 보는 바와 같이 구성된 오픈스택 플랫폼은 총 31대로 구성되어 있으며 28대의 컴퓨팅노드와 네트워크노드, 제어노드, 인증노드로 1대씩을 사용하고 있다. 오픈스택 커뮤니티에서 제공하는 메뉴얼에서 제시한 바와 같이 각 서비스 모듈간 통신을 위한 관리 네트워크와 실제 데이터 전송을 위한 데이터 네트워크를 별도로 구성하였으며 외부 연결을 위해 제어노드, 인증노드, 네트워크노드만 외부 네트워크에 연결하였다. 시스로그데몬에서



<Figure 7> Experiment Environment

출력되는 이벤트 로그 데이터와 오픈스택의 로그 데이터를 서버로 전송하기 위해 모든 오픈스택 노드에 rsyslog 클라이언트를 설치하고 관리 네트워크에 rsyslog 서버를 연결하여 이벤트 로그 데이터를 수집하였다.

5.2 메시지 기반 군집화 결과

<Table 3>는 수집된 이벤트 로그의 메시지를 기반으로 의미론적으로 군집화한 결과를 보여준다. <Table 3>의 첫 번째 필드는 이벤트 로그 데이터가 발생한 호스트 정보이다. 두 번째 필드는 메시지가 발생한 오픈스택 모듈을 나타내는 필드이다. 제 2.2절에서 언급한 바와 같이 오픈스택의 경우 로그 데이터 포맷에 이벤트 로그가 발생한 서비스 모듈이 표기되기 때문에 결함이 발생한 모듈을 쉽게 파악할 수 있다. 세 번째 필드는 Facility code로 오픈스택의 경우 16(LOG_LOCAL0) 값으로 수집되었음을 확인할 수 있다. 네 번째 필드는 발생한 이벤트 로그 수이며 다섯 번째 필드는 메시지 기반으로 군집화한 결과이다. 본 실험에서는 오픈스택 모듈 필드와 Facility 필드 값을 이용하지 않고 메시지에 담겨있는 텍스트만을 이용한 비지도 학습 방식의 군집화 기법을 사용했음에도 두 번째, 세 번째 필드와의 연관성을 확인할 수 있다. 즉, 오픈스택에서 발생한 이벤트 로그의 경우 neutron.agent.l3_agent 모듈에서 발생하는 이벤트 로그만 8번과 9번 그룹으로 구분되었을 뿐 나머지 모듈에서 발생한 이벤트 로그 데이터는 하나의 그룹으로 구분되어 있음을 확인할 수 있다. 8번과 9번 그룹으로 구분된 이유는 각 그룹별 메시지와 원인을 보여주는 <Table 4>를 통해 동일한 오픈스택의 neutron.agent.l3_agent 모듈에서 발생한 이벤트 로그임에도 발생 원인은 Failed synchronizing routers와 Failed reporting state로 차이가 있음을 확인할 수 있다. 또한, Compute node 3에서 Facility 0(커널 메시지)로 발생된 메시지도 <Table 4>를 통해 메시지 형태는 커널 메시지로 동일하지만 내용에는 차이가 있음을 확인할 수 있다.

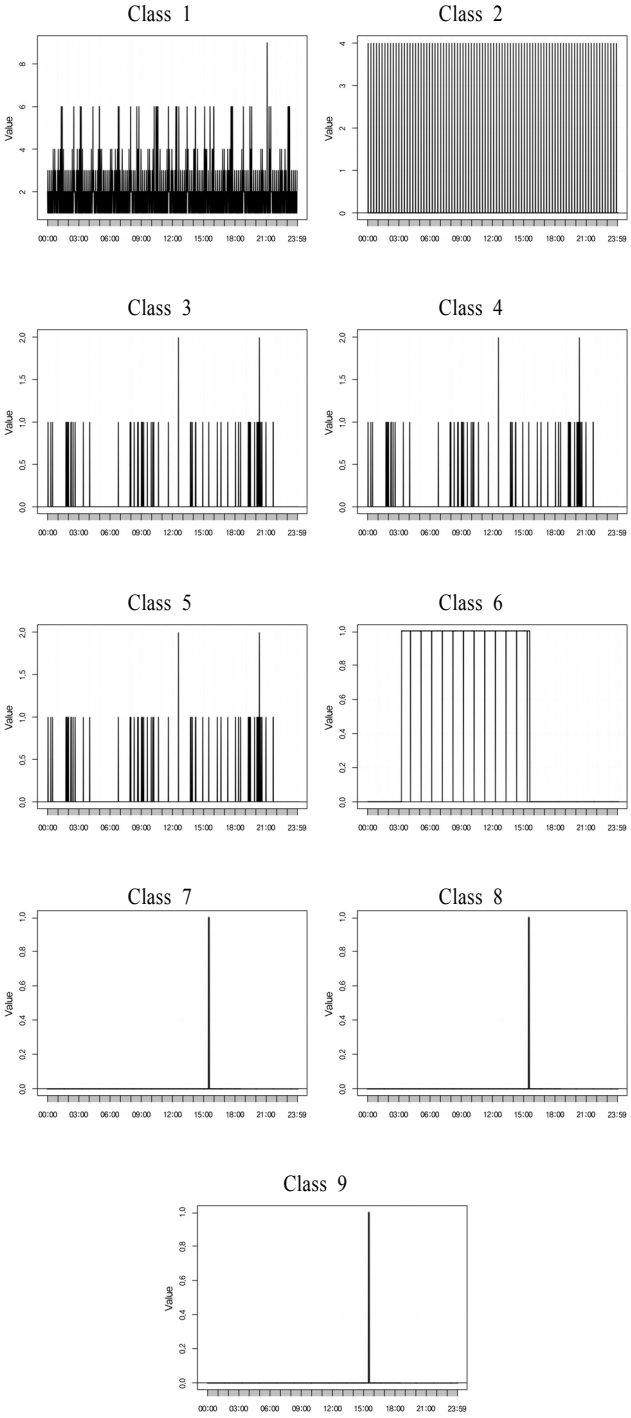
<Table 3> Clustering Results

FromHost	OpenStack Module	Facility	# of log	class
Controller	root	16	724	6
Controller	neutron.common.legacy	16	1	7
Network node	neutron.common.legacy	16	2	7
Network node	neutron.agent.l3_agent	16	3	8
Network node	neutron.agent.l3_agent	16	5	9
Login node	N/A	3	360	2
Compute node 1~28	nova.virt.libvirt.driver	16	32206	1
Compute node 3	N/A	0	65	3
Compute node 3	N/A	0	65	4
Compute node 3	N/A	0	65	5

<Table 4> Clustering Message and Description

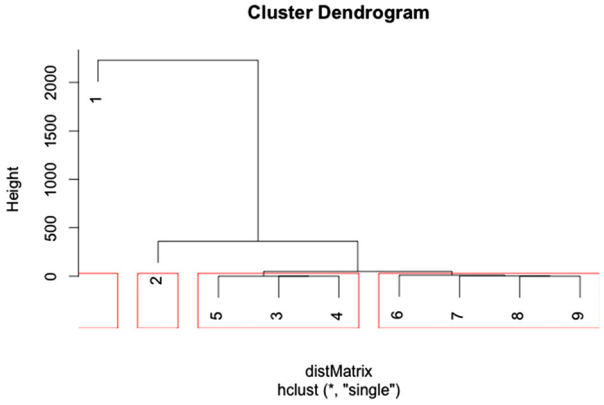
Class	Message	Description
1	07:25:00.672 31762 ERROR nova.virt.libvirt.driver [-] Getting disk size of instance-0000001b : [Errno 2] No such file or directory : …	Do not find the VM instance file in compute node.
2	host name not found: 3.centos.pool.ntp.org	Do not connect with NTP server in ntpd process
3	ata4 : exception Emask 0x10 SAct 0x0 SErr 0x4000000 action 0xe frozen	A kernal message to report SATA HDD read or write error.
4	ata4 : irq_stat 0x00000040, connection status changed	A kernal message to report SATA HDD read or write error.
5	SError : {CommWake DevExch }	A kernal message to report SATA HDD read or write error.
6	14:34:19.728 23037 ERROR root [-] Unexpected exception occurred 61 time(s)… retrying. 2015-11-17 14:34:19.728 23037 TRACE root Traceback (most recent call last) : …	A error message since neutron server failed to connect to AMQP server. This message is reported periodically.
7	15:26:00.217 12870 ERROR neutron.common.legacy [-] Skipping unknown group key : firewall_driver	A error message to report unknown group key in controller/network nodes.
8	15:29:40.291 13378 ERROR neutron.agent.l3_agent [-] Failed synchronizing routers 2015-11-17 15:29:40.291 13378 TRACE neutron.agent.l3_agent Traceback (most recent call last) : …	A error message since neutron l3 agent failed sysnchronizing routers in network node.
9	15:35:30.771 13378 ERROR neutron.agent.l3_agent [-] Failed reporting state! 2015-11-17 15:35:30.771 13378 TRACE neutron.agent.l3_agent Traceback (most recent call last) : …	A error message since neutron l3 agent failed reporting state in network node.

5.2 DTW를 통한 군집화된 메시지간 연관성 분석 결과



<Figure 8> Clustered Messages Occurring Time Patterns

<Figure 8>은 각 그룹별 이벤트 로그 데이터 발생 패턴을 보여주는 것으로 X축은 발생 시각(00:00~23:59)을 의미하여 Y축은 발생된 이벤트 로그 데이터 수를 의미한다.



<Figure 9> Correlation among Clustering Messages

<Figure 9>는 군집화된 메시지 그룹에서 이벤트 발생 패턴 유사도를 DTW를 통해 계산하고 거리값 15를 기준으로 그룹을 분류한 결과를 나타낸다. 그림에서 보는 바와 같이 3, 4, 5 그룹과 6, 7, 8, 9 그룹간 패턴 유사도가 높음을 확인할 수 있다. 특히, 3, 4, 5 그룹의 경우 <Figure 8>를 통해 이벤트 로그 발생 패턴이 동일함을 확인할 수 있고 이는 동일한 결함 원인으로 발생되는 이벤트 로그 메시지들로 유추할 수 있다. <Table 4>를 통해 3, 4, 5 메시지 그룹의 실제 내용 및 원인을 살펴보면 모두 SATA로 연결된 하드 디스크 사용에 결함이 있어서 발생하는 이벤트 로그 메시지 그룹들임을 확인할 수 있다.

그룹 6, 7, 8, 9의 경우 제어노드의 Neutron-server와 네트워크노드의 Neutron l3 agent에서 발생하는 이벤트 로그 메시지로 두 모듈간 연결에 결함이 있어서 발생한 이벤트 로그 메시지다.

본 논문의 실험 결과에서 주목할 만한 결과는 6 그룹과 패턴 유사도가 높은 그룹을 발견한 것이다. <Figure 8>과 같이 7, 8, 9 그룹의 발생 패턴의 유사도가 높은 것은 쉽게 파악할 수 있지만 6 그룹의 경우 패턴 유사도가 높은 그룹을 발견하기 어렵다. 실제로 Neutron 서버와 AMQP 사이의 연결 문제로 인하여 발생한 이벤트 로그 메시지(Class 6)의 경우 오픈스택 모듈 필드는 root로 표기되어 오픈스택 서비스 모듈과의 연관성을 유추하기 어려우며 이벤트 로그 개수만으로도 다른 그룹과의 연관성을 유추하기 어렵다. 그러나 본 연구에서는 이벤트 로그 메시지 군집의 DTW 기반 유사도 분석을 통해 6 그룹의 발생 패턴과 유사도가 높은 오픈스택 서비스 모듈을 파악할 수 있었다.

6. 결 론

시스템 관리자는 시스템에 결함이 발생한 경우 결함의 원인을 파악하기 위해 이벤트 로그 데이터를 분석한다. 그러나 시스템은 규모가 크고 복잡해짐에 따라 발생하는 이벤트 로그 데이터양이 많고 시스템 구성 모듈간 관계도 복잡해 관리자가 이벤트 로그 파일을 하나하나 분석하여 결함 원인을 파악하는 것은 거의 불가능하다. 이를 위해 본 논문에서는 시스템뿐만 아니라 응용 프로그램에서 발생하는 이벤트 로그 데이터를 하나의 서버에 수집하고 이를 메시지 기반으로 유사한 로그로 군집화하고 연관성을 분석함으로써 관리자가 결함의 원인을 신속히 파악할 수 있는 메시지 기반 로그 분석 방안을 제시하였다. 또한 제안된 분석 기법을 구현할 수 있는 방안도 제시하고 실제로 운영 중인 오픈스택 시스템의 이벤트 로그 데이터를 수집하여 분석 결과를 제시하였다.

References

[1] Feinerer, I. and Hornik, K., Package tm, *Tech. Rep.*, CRAN, 2013.

[2] Gerhards, R., The syslog protocol, RFC 5424, 2009.

[3] Joo, W.-M. and Choi, J.Y., Curriculum mining analysis using clustering-based process mining, *Journal of Society of Korea Industrial and Systems Engineering*, 2015, Vol. 38, No. 4, pp. 45-55.

[4] Kaufman, K. and Rousseeuw, P.J., Finding groups in data : An introduction to cluster analysis, *John Wiley and Sons*, 2009.

[5] Oliner, A. and Stearley, J., What supercomputers say : A study of five system logs, in Proc. of *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2007, Edinburgh, pp. 575-584.

[6] OpenStack, OpenStack open source cloud computing software, [Online]. Available at : http://www.openstack.org/.

[7] Park, Y.-S., Yoon, B.-N., and Lim, J-H., An empirical study on faults prediction for large scale telecommunication software, *Journal of the Korean Society for Quality Management*, 1999, Vol. 27, No. 2, pp. 263-276.

[8] Pitakrat, T., Grunert, J., Kabierschke, O., Keller, F., and Hoorn, A., A framework for system event classification and prediction by means of machine learning, in Proc. of *the 8th International Conference on Performance Evaluation Methodologies and Tools. ICST* (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2014, pp. 173-180.

[9] R development core team, R : A language and environment for statistical computing, [Online], Available : http://www.r-project.org.

[10] RSYSLOG : The rocket-fast system for log processing.

Available: http://www.rsyslog.com.

[11] Sahoo, R.K., Oliner, A.J., Rish, I., Gupta, M., Moreira, J.E., MA, S., Vilalta, R., and Sivasubramaniam, A., Critical event prediction for proactive management in large-scale computer clusters, in Proc. of *the 9th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003, pp. 426-435.

[12] Sakoe, H. and Chiba, S., Dynamic programming algorithm optimization for spoken word recognition, *IEEE Transactions on Acoustics*, *Speech and Signal Processing*, 1978, Vol. 26, No. 1, pp. 43-49.

[13] Yoo, J., Module communization for product platform design using clustering analysis, *Journal of Society of Korea Industrial and Systems Engineering*, 2014, Vol. 37, No. 3, pp. 89-98.

[14] Zheng, Z., Lan, Z., Gupta, R., Coghlan, S., and Beckman, P., A practical failure prediction with location and lead time for Blue Gene/P, in Proc. of *International Conference on Dependable Systems and Networks Workshops (DSN-W)*, 2010, pp. 15-22.

[15] Zheng, Z., Lan, Z., Park, B.H., and Geist, A., System log pre-processing to improve failure prediction, in Proc. of *IEEE/IFIP International Conference on Dependable Systems and Networks*, 2009, pp. 572-577.

ORCID

Ju-Won Park | http://orcid.org/0000-0003-1388-1583
Eunhye Kim | http://orcid.org/0000-0001-7812-9160
Jaekeun Yeom | http://orcid.org/0000-0001-6727-1823
Sungho Kim | http://orcid.org/0000-0001-5448-3923