JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 29 No. 3, pp. 127-136, March 2024
https://doi.org/10.9708/jksci.2024.29.03.127

Anomalous Pattern Analysis of Large-Scale Logs with Spark Cluster Environment

Sion Min*, Youyang Kim*, Byungchul Tak**

*Student, School of Computer Science and Engineering, Kyungpook National University, Daegu, Korea
*Student, School of Computer Science and Engineering, Kyungpook National University, Daegu, Korea
**Professor, Dept. of Computer Science and Engineering, Kyungpook National University, Daegu, Korea

[Abstract]

This study explores the correlation between system anomalies and large-scale logs within the Spark cluster environment. While research on anomaly detection using logs is growing, there remains a limitation in adequately leveraging logs from various components of the cluster and considering the relationship between anomalies and the system. Therefore, this paper analyzes the distribution of normal and abnormal logs and explores the potential for anomaly detection based on the occurrence of log templates. By employing Hadoop and Spark, normal and abnormal log data are generated, and through t-SNE and K-means clustering, templates of abnormal logs in anomalous situations are identified to comprehend anomalies. Ultimately, unique log templates occurring only during abnormal situations are identified, thereby presenting the potential for anomaly detection.

▸ **Key words**: Log analysis, Anomaly detection, Distributed system, Spark, Hadoop, Log template

[요 약]

본 연구는 Spark 클러스터 환경에서 대용량 로그를 분석하여 시스템 이상과의 연관성을 탐색한다. 로그를 활용한 이상 감지 연구는 증가하고 있으나, 클러스터의 다양한 컴포넌트의 로그를 충분히 활용하지 못하고 이상과 시스템의 연관성을 고려하지 않는다는 한계가 있다. 따라서 본 논문에서는 정상과 비정상 로그의 분포를 분석하고, 로그 템플릿의 출현 여부를 통해 이상 감지 가능성을 탐색한다. Hadoop과 Spark를 활용하여 정상과 비정상 로그 데이터를 생성하고, t-SNE와 K-means 클러스터링을 통해 비정상 상황에서의 로그 템플릿을 찾아 이상 현상을 파악한다. 결과적으로, 비정상 상황에서만 발생하는 고유한 로그 템플릿을 확인하며 이를 통해 이상 현상 감지의 가능성을 제시한다.

▸ **주제어**: 로그 분석, 이상 탐지, 분산 시스템, 스파크, 하둡, 로그 템플릿

• First Author: Sion Min, Corresponding Author: Byungchul Tak
*Sion Min (yehet5959@knu.ac.kr), School of Computer Science and Engineering, Kyungpook National University
*Youyang Kim (youyangkim@knu.ac.kr), School of Computer Science and Engineering, Kyungpook National University
**Byungchul Tak (bctak@knu.ac.kr), Dept. of Computer Science and Engineering, Kyungpook National University
• Received: 2024. 01. 08, Revised: 2024. 02. 28, Accepted: 2024. 02. 28.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

현대 시스템의 확장성과 복잡성이 급속도로 증가함에 따라, 많은 잠재적인 결함과 취약점이 도출되고 있다. 상용 시스템의 서비스 장애는 사용자 경험의 저하부터 금전적인 손실에 이르는 심각한 문제를 초래할 수 있다. 이러한 문제를 예방하고, 시스템의 안정성을 유지하기 위해 이상 현상을 정확하게 감지하는 것은 필수적인 작업으로 인식되고 있다. 로그 분석은 대규모 시스템에서 생성되는 대량의 실행 로그를 활용할 수 있으므로, 시스템의 이상 현상 탐지에 있어서 효과적인 해결책으로 각광받고 있다. 로깅(Logging)은 실무에서 널리 사용되며, 통계적으로 평균 소스 코드 58줄당 1줄의 로깅 코드가 존재한다고 알려져 있다[1]. 이에 따라, 최근 몇 년 동안 로그를 기반으로 시스템의 이상 현상을 감지하기 위한 다양한 방법론이 제안되고 있으나, 이러한 모델들이 실제 산업 현장에서 적용되기 위해서는 아직 극복해야 할 몇 가지 문제점이 남아있다.

먼저, 현재 모델의 검증에 사용되는 오픈소스 데이터 세트(Open-source log data set)들은 여러 제한 사항이 존재한다. Table 1에는 최근 이상 탐지 연구들[1-5]에서 활용되는 오픈소스 데이터 세트들이 정리되어 있다[6]. 이 중 레이블링(Labeling) 된 데이터 세트는 단 5개에 불과하며, 그중 현재 대규모 서비스를 대표할 수 있는 분산 시스템 로그는 3개에 불과하다. Hadoop, OpenStack 데이터 세트는 각각 48.61MB, 58.61MB로 크기가 제한적이며, 특히 HDFS(Hadoop Distributed File System), Hadoop 데이터 셋은 구동 애플리케이션(Application)이 단순 반복 맵리듀스(Map-Reduce) 작업에 한정되어 있다. 또한 로그 생성이 HDFS, YAML(YAML Ain't Markup Language) 등의 특정 컴포넌트(Component)에 한정되는 등의 제한 사항이 존재한다. 이에 본 논문에서는 다양한 애플리케이션을 구동하여 Spark 로그뿐만 아니라, 구동에 필요한 HDFS, YAML 등 Hadoop 컴포넌트를 포함하여 총 정상 로그 1,439,494줄, 총 비정상 로그 940,880,608줄의 대용량 로그 데이터 세트를 활용하여 분석을 수행하였다.

두 번째로, 로그 이상을 식별할 수 있는 패턴과 특징이 다양하며, 이러한 특징이 반드시 시스템의 이상과 직접적으로 연관되는 것은 아니므로 사전에 분석이 필요하다. Fig. 1에서는 기존 연구와 실증적 분석을 바탕으로 5가지의 로그의 비정상 패턴을 제시한다[7]. 로그의 비정상 패턴에는 부정적인 키워드(예: Error), 템플릿 개수(Template counts), 템플릿 시퀀스(Template sequences), 변숫값이나 분포의 변화, 시스템 성능 문제로 인한 로그 기록 지연 등이 있다. 최근 연구들은 주로 템플릿 시퀀스 이상을 감지하고 성능을 향상하는 데 초점을 맞추고 있으며, 이러한 모델들은 블랙박스로 동작하여 해당 시점의 이상 발생 여부만을 확인할 뿐, 실제 시스템 이상과 로그 이상 간의 연결을 명확히 하지 않는다는 한계가 있다. 따라서 모델을 사용할 경우 엔지니어는 모델의 감지 결과를 기반으로 로그에서 어떤 이상이 발생하였는지, 왜 발생하였는

Table 1. Available Open-source log data set.

Datasets	Existence of labeling	Size	Log generation nodes	Remarks
Distributed system				
HDFS	O	1.47 GB	HDFS nodes	MapReduce execution logs
Hadoop	O	48.61 MB	YAML nodes	MapReduce execution logs
Spark	X	2.75 MB	Spark nodes	Some logs lost.
ZooKeeper	X	9.95 MB	ZooKeeper nodes	
OpenStack	O	58.61 MB	OpenStack network node, control node, compute nodes	VM creation, suspension, deletion logs
Supercomputer				
BGL	O	708.76 MB	Blue Gene/L	
Thunderbird	O	29.60 GB	Thunderbird	

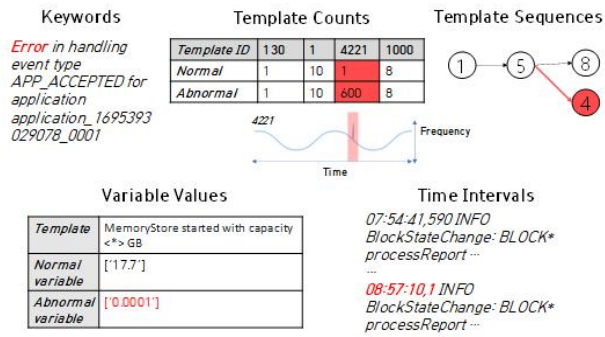

Fig. 1. Log's abnormal log patterns.

Table 2. Log example generated from Spark.

1	2023-12-31 10:56:44,574 DEBUG org.apache.spark.util.ShutdownHookManager: Adding shutdown hook
2	2023-12-31 10:56:44,734 DEBUG org.apache.hadoop.fs.FileSystem: Loading filesystems
3	2023-12-31 10:56:45,588 DEBUG org.apache.hadoop.ipc.client: Connecting to master/
...	.....

지, 정상 상황에서는 어떻게 동작해야 하는지, 로그 이상이 실제로 시스템 이상을 나타내는지를 다시 확인해야 한다. 이러한 이유로 사전에 정의한 로그 이상이 시스템 이상과의 연결 관계가 있는지를 이상 탐지 이전에 분석한 후 실제 이상 탐지에 활용해야 한다. 그러나 기존 이상 탐지 논문에서는 사용하는 로그 데이터 세트의 정상과 비정상에 대한 분석 없이 이상 탐지에 데이터 세트를 사용하고 있다. 따라서 본 논문에서는 로그의 비정상 패턴 중 템플릿의 개수 특성에 중점을 두어 시스템의 정상 로그와 비정상 로그 간 관계를 심층적으로 분석하여 로그 이상 탐지의 타당성에 대해 입증하고자 한다.

본 논문은 다음과 같은 기여를 제공한다. 첫 번째로, Spark 클러스터 환경의 모든 컴포넌트에서 여러 예제 애플리케이션을 활용하여 대용량의 로그를 생성하여 실제 분산 시스템의 환경에 가까운 분석을 수행한다. 두 번째로, 로그의 이상 패턴 중 하나인 템플릿 개수를 바탕으로 시스템 이상과의 연결 관계를 심층적으로 분석한다.

본 논문은 다음과 같은 구조로 구성되어 있다. 2장에서는 기본 개념을 소개한다. Backgrounds에서는 시스템 로그에 대한 기본적인 이해와 Hadoop과 Spark에 대한 정보를 제공한다. Related Works에서는 로그 분석을 위한 시스템 로그 데이터 세트 수집과 로그 데이터를 이용한 이상 탐지에 관한 이전 연구를 살펴본다. 3장에서는 분석 방법에 관해 설명한다. 분석은 크게 로그 데이터 준비, 로그 데이터의 특징 분석으로 이루어지며 추가로 로그 템플릿 출현 정보를 기반으로 한 이상 탐지를 간략하게 소개한다. 마지막으로, 4장에서는 본 논문의 결론을 제시한다.

II. Preliminaries

1. Backgrounds

1.1 System Log

로그(Log)는 시스템에서 동작 중에 발생하는 다양한 상황과 내부 상태를 기록한다. 현재 크고 복잡한 시스템에서는 대용량의 로그 데이터가 생성되며, 이는 시스템의 이상 탐지 및 문제 해결을 위하여 널리 사용된다.

로그는 시스템 실행을 추적하는 용도의 반정형 메시지이다. 로그에는 로그 출력 시간, 로그 레벨 (DEBUG, INFO 등) 로그가 출력된 컴포넌트, 그리고 로그 메시지 내용이 포함된다. Table 2는 Spark에서 생성된 로그 일부를 보여준다.

로그 분석을 위하여 출력된 로그를 그대로 사용하는 것이 아니라 전처리나 로그 파싱(Log Parsing)을 통하여 각 로그를 로그 유형에 해당하는 로그 템플릿(Log Template)으로 변환하고, 해당 템플릿을 사용하여 오류를 분석한다. 본 논문에서는 전처리를 통하여 로그 템플릿을 생성하고 이를 사용하여 오류를 분석한다.

1.2 Hadoop and Spark

Hadoop과 Spark는 대용량 데이터 처리 및 분석을 위한 오픈소스 프레임워크(Framework)로, 대규모 데이터 세트를 효과적으로 다룰 수 있는 기능을 제공한다. Hadoop은 배치 처리를 위한 목적으로 설계되었으며, 대규모 데이터를 클러스터(Cluster)에 분산하여 처리한다. Spark는 Hadoop의 기능을 활용하면서 대화형 질의, 스트리밍 처리(Streaming Processing), 기계학습 등 다양한 기능을 제공하는 프레임워크이다. 본 논문에서는 분산 시스템 환경에서 출력된 로그를 사용하여 오류 상황을 분석하기 위하여 Hadoop과 Spark를 활용하여 로그를 수집하고 이를 분석한다.

2. Related Works

2.1 Collection of System Log Dataset for Log Analytics

대규모의 로그를 효율적으로 처리하기 위해 다수의 연구가 인공지능 기반의 로그 분석에 초점을 맞추고 있다. 그러나 이러한 기법 중 일부만이 실제 산업에 적용되고 있는데, 이는 공개 로그 데이터 세트 부족으로 인한 것이다.

Loghub[8]은 연구를 통해 분산 시스템, 슈퍼컴퓨터(Supercomputer), 운영체제, 모바일 시스템(Mobile System), 서버 프레임워크(Server Framework) 및 독립된 소프트웨어에서 수집한 19개의 로그 데이터 세트를 제공하고 있다. 그중 일부 데이터 세트만이 레이블링 되어 있으며, 이 데이터들은 다른 연구에서 공개된 데이터를 활용한 것이다.

HDFS 로그[9]는 Hadoop에서 맵리듀스 작업을 실행하여 생성되었으며, 전문가들이 HDFS 성능 저하를 의미하는 로그에 레이블을 부여하였다. 또한 BGL, Thunderbird 로그[10]는 슈퍼컴퓨터 로그로, 시스템 관리자와 협의하여 이상 기준에 따라 레이블링하였다.

오류를 주입한 로그들은 해당 오류 자체가 이상 현상을 나타내므로, 수동 레이블링에 드는 노력을 줄일 수 있다. [11]에서는 Hadoop에서 WordCount 및 PageRank 애플리케이션을 실행한 후 서버 전원 차단, 네트워크 연결 차단, 디스크를 수동으로 가득 차게 하는 하드웨어 오류를 시뮬레이션하여 394,308줄의 YAML 로그 데이터 세트 48.61MB

를 생성하였다. 또한 [2]에서는 OpenStack을 실행한 후 가상머신 생성, 중단, 삭제 로그를 수집하였다. 이 과정에서 가상머신 생성 중 OpenStack 컴포넌트인 neutron을 타임아웃(Time-out) 시키는 오류, 가상머신 삭제 중 libvirt API(Application Programming Interface) 오류, 가상머신 삭제 후 정리 과정 중 libvirt API 오류를 주입하여 207,820줄의 데이터 세트 58.61MB를 생성하였다.

본 연구에서는 Spark 애플리케이션을 실행하며 147종류의 설정 오류를 주입하였고, 이를 통해 생성된 942,320,102줄의 데이터 세트를 분석에 활용하였다.

2.2 Anomaly Detection Using Log Data

시스템 로그는 시스템의 다양한 현상을 기록하고, 시스템의 성능 저하나 오류가 발생하였을 때 이를 해결하기 위한 정보를 제공한다. 로그 데이터들은 다양한 정보를 내포하고 있어 시스템 문제의 진단에 유용하게 사용된다. 로그의 이상 현상을 탐지하는 방법으로는 변숫값과 빈도 특징을 기반으로 하는 PCA[9], 로그 간 양적 관계를 기반으로 하는 Invariant Mining[12], 로그를 사용하여 실행 흐름 그래프를 생성하는 방법[13], 그리고 로그 이벤트의 발생 빈도를 이용한 방법이 있다[14].

이중 Deeplog[2]는 딥러닝을 활용하여 템플릿 시퀀스의 이상을 감지하는 방법이다. 또한 로그 템플릿은 계속해서 변화하므로 로그 메시지의 문맥을 바탕으로 시퀀스를 생성하여 로그의 이상을 탐지하는 연구도 진행되고 있다. 예를 들어, LogRobust[1] 는 유사한 의미가 있는 로그 템플릿을 유사한 의미 벡터(Vector)로 표현하고, 이를 점진적으로 업데이트한다. LogAnomaly[3]는 유의어 및 반의어를 사용하여 변경된 로그 템플릿을 유사한 기존 템플릿으로 교체한다. PLELog[4]은 LogRobust와 동일한 탐지 방법을 사용하면서, 모델 학습 과정에서 필요한 데이터에 레이블을 자동으로 부여하여 감독 학습 접근법의 강점을 살린 반지도 학습 방법을 사용한다.

또한 NoTIL[15]은 성능 이상을 감지하는 연구로, 인접한 로그 간의 시간 간격과 다른 로그 이벤트 유형을 확인하여 시간 지연에 관한 이상 현상을 감지할 수 있다. 마이크로서비스(Microservice)에서 트레이스(Trace)의 이상을 로그와 스팬(Span)의 관계를 통해 탐지하는 DeepTralog[16]는 병렬과 비동기적 실행에서 교차하는 로그를 그래프로 처리할 수 있다.

최근에는 사전 학습된 대규모 언어모델 BERT(Bidirectional Encoder Representations from Transformers)를 사용하여 로그 파싱 오류의 영향을 낮추며 이상을 탐지하는 NeuralLog[5], EvLog[17]도 연구되고 있다.

Table 3. Injected errors by the configuration types.

Configuration type	Error count	Error example
path	8	/usr/bin, 10000
paths	8	hdfs:file.txt, @#
class	9	java.util.ArrayList
classes	9	www.google.com
boolean	5	true, false, 100
int	11	-1, 0, 1, 65535
ints	11	1/2/4/8, 4.3,2.2
ip	8	localhost, 8.8.8.8
ips	8	google.com,8.8.8.8
password	2	xyz, @#
timeduration	11	0s, 1ms, 100gb
timedurations	11	1ms,2ms,3ms,4ms
size	9	10kb, 100gb, xyz
string	7	hello,world
strings	7	www.google.com
float	11	3.1415926535
range	12	0-100, 100-200
Total	**147**	

1장에서 언급한 바와 같이 이상 탐지 이전에 탐지할 로그의 이상 패턴이 시스템 이상과 부합하는지에 대한 검증은 필수적이며, 이에 대한 분석이 기존 연구에서는 미흡하다는 점을 지적하였다.

따라서 본 연구에서는 기존의 이상 탐지 방법들을 보완하여 이상 현상 발생 시 출력되는 로그 중 정상 로그와 비정상 로그의 특성을 분석하고, 이를 통하여 로그를 이용한 이상 현상 탐지의 타당성을 판단하고자 한다.

III. The Proposed Scheme

1. Log Data Preparation

1.1 Normal & Abnormal Log Data Generation

본 논문에서는 시스템 이상을 Hadoop과 Spark에서 발생하는 설정으로 인한 오류로 정의하고 해당 설정 오류 로그를 분석한다. 또한 오류가 없는 정상으로 동작한 상황에서 발생한 로그도 수집하여 분석한다. 정상 상황은 Hadoop과 Spark의 설정 파일에 명시된 기본값(Default value)으로 Spark 애플리케이션들을 실행하였을 때 추출되는 로그를 말한다. 본 논문에서 로그 추출을 위하여 실행한 애플리케이션들은 Spark에서 제공하는 예제 애플리케이션으로 DeveloperApiExample, DecisionTreeExample, Analytics로 총 3개이다[18].

오류 로그를 수집하기 위하여 해당 연구에서는 Hadoop의 core-site.xml 설정 파일에 오류를 주입한 후, 3개의 애플리케이션을 실행하여 로그를 수집하였다. Hadoop의 core-site.xml에 존재하는 설정은 총 461개이며, 해당 설

정들에 주입할 오류를 17개로 분류하였다. 설정 오류들은 실제 환경에서 발생할 수 있는 오류의 값들로 구성된다. Table 3은 각 설정 종류별 주입한 오류 종류와 예시를 나타낸다.

또한 분산 시스템 환경을 구성하기 위하여 Docker를 사용하여 1개의 master 컨테이너(Container)와 3개의 slave 컨테이너를 생성하였다. 애플리케이션을 'spark-submit'을 통하여 master 컨테이너에서 클러스터 모드(Cluster mode)로 실행한 후 애플리케이션이 종료되면 4개의 컨테이너의 각 컴포넌트에서 생성된 로그를 하나의 파일에 시간순으로 정렬하여 수집하였다. core-site.xml에 존재하는 416개 설정에 Table 3의 설정 타입별로 오류를 주입하고 3개의 애플리케이션을 실행하여 총 1,248개의 오류 상황에 대한 로그를 수집하였다. 분석에 사용된 정상 로그는 총 1,439,494줄로 DeveloperApiExample에서 227,671줄, Analytics에서 944,280줄, DecisionTreeExample에서 267,543줄이며. 비정상 로그는 총 940,880,608줄로 DeveloperApiExample에서 76,979,624줄, Analytics에서 673,781,555줄, DecisionTreeExample에서 190,119,429줄이다. 실험은 메모리(Memory) 328GB, CPU는 48코어(Core) Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz인 서버(Server) 아래 각 2코어 CPU, 16GB RAM, 운영체제 Ubuntu 20.04.6 Docker 컨테이너 4개로 클러스터를 구성하여 수행하였다.

1.2. Template Construction via Log Preprocessing

로그 분석은 출력된 로그를 템플릿으로 변환하는 작업을 요구한다. 본 연구에서는 전처리를 통하여 로그 템플릿을 생성하였다. 전처리를 위하여 먼저 로그에 포함된 시간, 요청 ID(request-ID), 로그가 출력된 컴포넌트를 제거하였다. 또 모든 단어를 소문자로 변경한 뒤, 숫자가 포함된 단어, 특수문자가 포함된 단어를 제거하였다. 전처리 후에는 로그별 ID를 생성하였다. 본 연구에서 생성된 로그 템플릿의 수는 총 4,346개이다.

2. Log Feature Vector Similarity Comparison

시스템 이상 로그의 양상을 분석하기 위하여 정상 상황에서 출력된 로그와 오류 상황에서 출력된 로그 간의 유사도를 비교한다. 유사도 분석은 1,248개 각 비정상 로그와 각 이상을 발생시키기 위하여 실행한 애플리케이션의 정상 로그 간의 유사도를 비교한다. 유사도 비교는 코사인 유사도(Cosine Similarity)를 활용한다. 예를 들어, 'hadoop.http.authentication.type' 설정에 오류를 주입한 후 DeveloperAPIExample 애플리케이션을 실행하여 수집한 오류 로그는 DeveloperAPIExample 애플리케이션을 실행한 정상 로그와 코사인 유사도를 비교하였다.

Table 4. One-hot vector showing the log template appearance status per error type.

Configuration type	hadoop.http.authentication.type					
Executing app	DeveloperAPIExample					
one-hot vector	0	1	2	...	4344	4345
	0	1	0		1	1

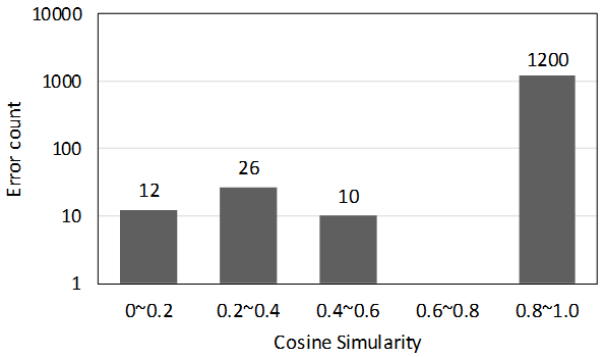

Fig. 2. Error count histogram by the similarity.

2.1 Log Feature Vector Generation per Error

코사인 유사도를 사용하여 로그 간의 유사도를 분석하려면 먼저 오류별로 출력된 로그를 벡터로 변환하여야 한다. 이를 위하여 정상 상황에서 출력된 로그와 오류별 출력된 로그를 원-핫 벡터(One-hot vector)로 표현하였다.

원-핫 벡터란 특정 로그 템플릿의 출력 여부를 0과 1로 나타내는 벡터를 말한다. 원-핫 벡터의 크기는 생성된 템플릿의 수인 4,346이다. Table 4는 한 오류 상황에서 생성된 원-핫 벡터를 보여주는 예시로, 원-핫 벡터 행의 상위 행은 로그 템플릿 번호를, 하위 행은 각 로그 템플릿의 출력 여부를 나타낸다.

2.2 Error Log Analysis using Cosine Similarity

1,248개의 오류 상황 로그를 모두 원-핫 벡터로 변환한 뒤, 정상 로그의 원-핫 벡터와 코사인 유사도를 계산하였다. Fig. 2는 각 코사인 유사도 구간별 해당 구간에 속하는 오류의 개수를 보여준다. 유사도 분석 결과, 정상 벡터와의 유사도가 0.8~1.0인 오류 수는 1,200개로 나타났으며, 0.6~0.8은 0개, 0.4~0.6에는 10개, 0.2~0.4에는 26개, 0.2 이하는 12개로 확인되었다. 또한 Fig. 3은 각 구간에 속하는 오류 로그에서 정상 상황에서 출력된 로그 템플릿을 제외 후 오류 상황에서만 출력되는 로그 템플릿의 평균 개수

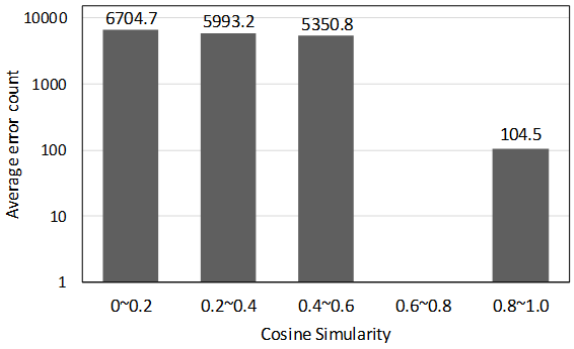

Fig. 3. Average error log count histogram by similarity.

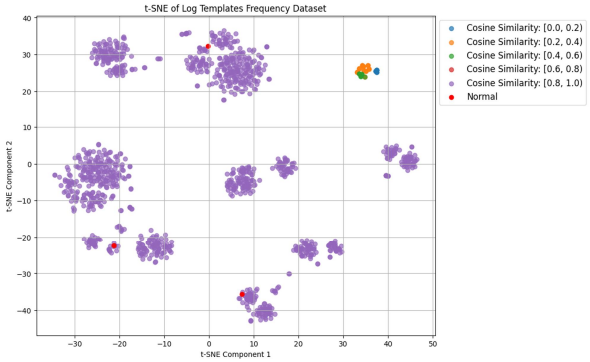

Fig. 4. Dimension reduction results using t-SNE

Table 5. Normal and abnormal case log template count.

Criteria	Number of types
Error situations	4,346
Normal situations	2,517
Appear only in error situations	1,829
Appear only in normal situations	0

를 보여준다. 유사도가 0.8~1.0인 구간에서는 평균 104.5개의 오류 로그가 출력되었고, 0.4~0.6에는 5350.8개, 0.2~0.4에서는 5993.2개, 0.2 이하 구간에서는 6704.7개가 출력되었다.

이 결과는 대부분의 비정상 로그 벡터와 정상 로그 벡터의 로그 템플릿들의 등장 양상이 매우 유사하다는 것을 보여준다. 이는 구간별 평균 오류 로그 개수에서도 확인할 수 있다.

유사도 0.8 이하 구간에서의 오류 로그는 비교적 많이 출력되지만, 대부분의 비정상 로그가 속한 유사도가 매우 높은 0.8~1.0 구간에서는 로그 대부분이 정상 로그와 유사하므로 오류 로그의 출력 수가 약 67배 적게 나타난다. 그러나 정상 로그와 완전히 일치하는 형태를 보인 오류 로그는 존재하지 않으므로, 특정 구간에서는 오류에서만 나타나는 로그 템플릿이 존재할 수 있다.

결과적으로, 오류 상황에서만 나타나는 고유한 로그 템플릿이 존재할 수 있음을 확인하였다. 또한 오류가 발생한 상황에서는 일반적인 정상 실행과는 다르게 애플리케이션의 동작이 변경되어 정상 실행과는 다른 로그가 출력될 수 있음을 확인하였다. Table 5에서는 오류 상황과 정상 상황에서 발생한 로그 템플릿의 종류를 보여준다. 초기 단계에서, 오류 상황별로 정상 로그를 제외한 오류 로그만을 수집하였으며, 그 결과 4,346개의 로그 템플릿 중 1,829개가 오류 로그 템플릿으로 수집되었다. 여기서 오류 로그 템플릿이란 정상 상황에서는 출력되지 않고 오류 상황에서만 나타나는 로그를 의미한다. 추가로 오류 특징 벡터들이 더 다양한 로그 템플릿을 포함하고 있음을 확인하였다. 이는 프로그램이 정상적인 흐름에서 벗어날 때 고유한 로그 템플릿이 다양하게 발생한다는 점을 입증해 준다.

2.3 Normal and Error Log Distribution Analysis with t-SNE and K-means Clustering

2.3절에서는 결과를 심층적으로 탐색하기 위해 t-SNE(t-distributed Stochastic Neighbor Embedding)를 통한 시각화와 K-means 클러스터링을 활용한 정상 및 오류 로그 분포 분석 및 오류 로그와 시스템 이상과의 관계를 분석하였다.

2.3.1 Dimension Reduction with t-SNE and Error Log Distribution Analysis

t-SNE은 고차원 데이터를 저차원 공간으로 변환하는 비선형 차원 축소 방법이다. 이를 통해 데이터의 복잡한 구조와 패턴을 보존하면서 차원을 축소하고, 그 결과를 시각화함으로써 데이터를 효과적으로 분석할 수 있다. 본 연구에서는 원-핫 벡터를 사용하여 오류 상황과 정상 상황의 로그 유사도를 구한 후, t-SNE를 활용하여 벡터의 차원을 축소하였다. 이를 통해 정상 상황 로그와 오류 상황 로그의 분포를 시각화하고 분석하였다. t-SNE의 사용은 로그 데이터의 복잡성을 고려하였을 때, 데이터 간의 국소적인 구조를 잘 보존하는 특성 때문에 이 연구에 적합하다고 판단되었다.

Fig. 4는 t-SNE의 시각화 결과를 나타낸다. 시각화 결과, 코사인 유사도 비교 시 정상 벡터와 높은 유사도를 보인 유사도 구간 0.8~1.0의 오류 로그 벡터 1,200개는 Fig 4의 정상 로그들과 상대적으로 가까운 거리에, 0~0.8 유사도 구간에 속하는 오류 로그 벡터 48개는 정상 로그들과 가장 거리가 먼 우측 상단에 몰려서 나타났다. 또한 정상 특징 벡터 3종류와 매우 가까이 분포하는 오류 상황 벡터들이 있는 반면, 동떨어진 양상을 보이는 벡터들도 존재하였다. 이 결과는 코사인 유사도를 통해 측정된 로그 벡터

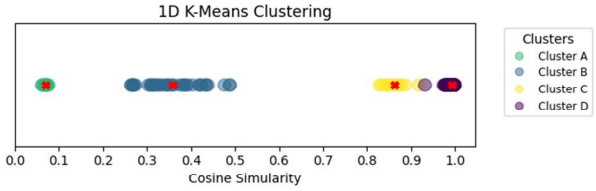

Fig. 5. Cosine similarity K-means clustering

Table 6. Silhouette scores for optimal number of clusters

Number of clusters	Silhouette scores
2	0.9816838
3	0.9818385
4	0.9825208
5	0.9541370
6	0.9522139
7	0.7147309

Table 7. Cosine similarity cluster's Centroids

Cluster	Centroids
A	0.06933743
B	0.35828375
C	0.86228988
D	0.99310877

간의 유사성이 t-SNE에 의한 시각화에서도 잘 반영되었음을 보여준다. 즉, 코사인 유사도가 높은 벡터들은 t-SNE 공간에서도 가깝게 위치하고, 그 반대의 경우에는 멀리 위치하는 경향이 있다.

2.3.2 Log Grouping with K-means Clustering

K-means 클러스터링은 데이터를 K개의 클러스터로 묶는 머신러닝(Machine Learning)의 비지도 학습 기법이다. 이 기법은 데이터 간의 거리를 계산하여 그룹화하므로 속도가 빠르고 대용량 데이터 처리에 적합하다. 따라서 방대한 양의 로그 데이터를 빠른 속도로 그룹화하기 위하여 K-means 클러스터링 기법을 사용하였다. 또한 2.2 절의 결과를 바탕으로 각 구간을 클러스터로 묶어 공통적인 특징을 찾아내고자 하였다. K-means 클러스터링은 scikit-learn 라이브러리의 KMeans 모듈을 사용하였다. Silhouette 분석을 사용하여 Silhouette score를 구한 후, 클러스터의 수를 4로 결정하였다. Table 6은 Silhouette 분석의 결과를 보여준다. 중심점을 선정하는 알고리즘으로는 K-means++을 사용하였다. 최대 반복 횟수는 300번으로 설정하고, 이 횟수 이전에 모든 데이터의 중심점 이동이 없으면 종료하도록 하였다.

Fig. 5는 K-means 클러스터링 결과를 클러스터별로 그룹화한 것을 나타낸다. 각 클러스터의 중심점은 X로 표시하였다. 클러스터 A는 12개, 클러스터 B는 36개, 클러스터 C는 10개, 클러스터 D는 1,190개의 코사인 유사도로 구성되어 있다. 각 클러스터의 중심점은 Table 7에 나와 있다. 정상 로그와의 유사도가 가장 높은 클러스터 D의 특징 벡터는 애플리케이션마다 Analytics가 397개, DecisionTreeExample이 396개, DeveloperApiExample이 397개로 균일하게 분포되어 있다. 이는 다른 클러스터도 동일한 양상을 보인다. 즉, 이상이 로그에 반영되는 정도가 애플리케이션에 따라 달라지는 것이 아니라, 어떤 설정 오류인지에 따라 다양한 정상과는 다른 패턴이 나타난다고 볼 수 있다.

Table 8. Configuration error frequency with in the cluster

Cluster A	
Configuration type	**Frequency**
hadoop.security.auth_to_local	3
hadoop.security.auth_to_local.mechanism	3
hadoop.security.authentication	3
hadoop.security.group.mapping	3
Cluster B	
Configuration type	**Frequency**
fs.AbstractFileSystem.hdfs.impl	3
fs.defaultFS	3
fs.getspaceused.classname	3
fs.permissions.umask-mode	3
hadoop.http.authentication.type	3
hadoop.http.filter.initializers	3
hadoop.rpc.protection	3
hadoop.rpc.socket.factory.class.ClientProtocol	3
hadoop.rpc.socket.factory.class.default	3
...	...
Cluster C	
Configuration type	**Frequency**
fs.AbstractFileSystem.file.impl	3
hadoop.security.credential.provider.path	3
hadoop.security.dns.nameserver	3
io.compression.codecs	1
Cluster D	
Configuration type	**Frequency**
adl.feature.ownerandgroup.enableupn	3
hadoop.security.group.mapping.ldap.bind.users	3
hadoop.security.group.mapping.ldap.bind.user	3
fs.s3a.s3guard.consistency.retry.limit	3
fs.s3a.s3guard.consistency.retry.interval	3
fs.s3a.s3guard.cli.prune.age	3
tfile.io.chunk.size	3
hadoop.security.group.mapping.ldap.bind.password	3
hadoop.security.group.mapping.ldap.base	3
fs.protected.directories	3
...	...

각 클러스터에 속하는 설정 오류들과 빈도의 일부를 Table 8에 기재하였다. 클러스터 A는 Hadoop의 보안(Security)과 관련된 설정 오류로만 이루어져 있다. Hadoop이 보안 모드에서 실행될 수 있도록 구성된 경우, Hadoop의 사용자는 Kerberos 프로토콜(Protocol)을 통해

인증을 받게 된다. 이 중 프로토콜의 사용자 명, 규칙, 인증 설정, 그룹 매핑(Mapping) 설정에 오류를 발생시키는 경우 로그 분포가 정상과 크게 달라진다. 클러스터 B에는 데이터 노드(Data node)의 작업을 위하여 필요한 HDFS의 URI(Uniform Resource Identifier) 설정, HDFS 사용 공간 추정 옵션 설정, 그리고 HTTP(Hypertext Transfer Protocol), RPC(Remote Procedure Call) 같은 네트워크(Network) 설정 등이 포함되어 있다. 클러스터 C에는 파일 크기 설정, 클라이언트(Client)의 연결 옵션 설정 등이 포함되어 있다. 클러스터 D에는 Azure, S3와 같은 클라우드(Cloud) 설정, Key-Value 형태의 TFile 구성, 그룹 매핑 설정 등이 포함되어 있다. 본 논문에서는 애플리케이션을 구동할 때 클라우드와 같은 외부 저장소를 사용하지 않고, 기본적인 파일 형식을 사용하였다. 즉, 클러스터 D가 정상과 유사한 형태의 분포를 보인 것은 같은 Hadoop core의 설정이지만 애플리케이션 구동에 필수적인 옵션이 아니므로 설정에 오류를 주입하여 오류 데이터를 만들었더라도 애플리케이션 실행에 영향을 끼치지 않아 정상 상황에서 출력되는 로그가 대부분을 차지한다는 것을 알 수 있다.

3. Anomaly Detection Based on Log Template Appearance Information

추가적인 분석을 통해, 오류에서만 나타나는 고유한 로그 템플릿을 이용하여 특정 로그 템플릿의 출현 여부를 활용하여 이상 현상을 탐지할 수 있는지를 조사하였다. 이는 본 연구의 핵심 주제를 넘어서는 부가적인 분석이지만, 이상 탐지 방법론의 유효성을 더욱 강조하고 본 연구의 중요성을 재확인하는 데 이바지하였다. 수집된 오류 로그 템플릿이 1,248개의 오류 상황 중에서 총 몇 번 출력되었는지를 분석하고 이를 비율로 나타내었다. Table 9에는 출력 빈도가 가장 높은 로그 템플릿별로 순위, 출력 비율, 로그 메시지가 제시되어 있다.

전체 오류 로그 중에서 출력 비율이 가장 높은 로그 템플릿은 ‘disconnected’, ‘shutdown’과 같은 키워드로 검색이 가능한 로그 템플릿이다. 출력 비율이 가장 높은 템플릿은 오류 상황에서 시스템에서 빈번하게 출력되는 로그를 의미한다. 또한 출력 비율 순위 3의 로그 템플릿은 로그 레벨에 따른 필터링을 통해 찾을 수 있는 템플릿이다. 그러나 순위 2의 로그 템플릿은 키워드 검색이나 필터링만으로는 찾을 수 없는 로그이다.

이러한 결과를 통해, 정상 로그와 비정상 로그의 특성을 템플릿의 출현 여부와 빈도 기반으로 구분 지을 수 있으며,

Table 9. Most frequent log templates.

Rank	Print ratio	Level	Component
1	87.7%	INFO	spark.executor.yarncoarsegrainedexecutorbackend
Log messages			
driver from * disconnected during shutdown			

Rank	Print ratio	Level	Component
2	84.5%	INFO	yarn.serer.nodemanager.containermanager.launcher.containerlaunch
Log messages			
could not get pid for *, waited for * ms.			

Rank	Print ratio	Level	Component
3	72.4%	WARN	netty.channel.nio.nioeventloop.selector
Log messages			
select() returned prematurely * times in a row; rebuilding selector ip netty channel nio *.			

이는 가장 일반적으로 사용되는 로그 이상 패턴 분석 방법인 키워드 검색, 로그 레벨 필터링(Filtering)만으로는 모든 이상 현상을 탐지하는 데에 한계가 있음을 확연히 보여주었다.

IV. Conclusions

본 논문은 Spark를 이용하여 분산 시스템 환경을 구성하고, 다양한 애플리케이션을 통해 대규모 로그 데이터를 생성하여 로그 템플릿의 빈도와 시스템 이상에 대한 분석을 수행하였다.

기존 이상 탐지 연구에서 벤치마크로 사용되는 오픈소스 데이터 세트들의 한계를 인지하고, 이를 개선하기 위해 다양한 애플리케이션을 통해 모든 컴포넌트에서 생성된 대규모 로그 데이터를 생성하고 분석을 수행하였다. 또한 로그 이상 패턴의 식별이 반드시 시스템의 이상과 직접적으로 연관되지 않음을 지적하며 이에 따른 사전 분석의 중요성을 강조하였다. 본 논문은 로그의 비정상 패턴 중 템플릿 개수와 빈도 특성에 초점을 맞추어 정상과 비정상 로그 사이의 관계를 심층적으로 분석함으로써 로그 이상 탐지의 타당성을 입증하였다.

분석 결과, 대부분의 비정상 로그 벡터와 정상 로그 벡터의 로그 템플릿들의 특징 벡터가 상당히 유사하였으나, 오류 상황에서만 나타나는 고유한 로그 템플릿을 통해 정상과 비정상을 구분 지을 수 있음을 확인하였다. 또한, 비정상 특징 벡터들이 더 다양한 로그 템플릿을 포함하고 있

음을 확인하였다. 이는 프로그램이 정상적인 흐름에서 벗어날 때 다양한 로그 템플릿이 발생한다는 것을 시사하였다. 이러한 유사도는 애플리케이션에 따라 변하는 것이 아니라, 설정 오류의 종류에 따라 달라지는 것을 발견하였다. 특히, 가장 정상과 유사도가 높은 로그 클러스터는 Hadoop core의 설정에 오류를 주입하였음에도 불구하고, 애플리케이션 구동에 필수적인 설정이 아니었기 때문에 정상과 가장 유사한 결과를 보여주었다. 추가로, 비정상 특징 벡터에만 속하는 로그 템플릿의 출현 여부를 통해 이상 현상 탐지 가능 여부를 조사하였다.

결론적으로, 본 논문은 분산 시스템에서의 로그 분석을 통해 로그 기반 이상 감지의 타당성을 입증하였다. 이러한 연구 결과는 실제 환경에서의 로그 기반 이상 탐지 시스템 개발에 있어 유용한 기초 자료로써 활용될 수 있을 것이다.

ACKNOWLEDGEMENT

This study was supported by the BK21 FOUR project (AI-driven Convergence Software Education Research Program) funded by the Ministry of Education, School of Computer Science and Engineering, Kyungpook National University, Korea (4199990214394). This work was also supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. NRF-2021R1A 5A1021944).

REFERENCES

[1] Xu Zhang, Yong Xu, Qingwei Lin, Bo Qiao, Hongyu Zhang, Yingnong Dang, Chunyu Xie, Xinsheng Yang, Qian Cheng, Ze Li, Junjie Chen, Xiaoting He, Randolph Yao, Jian-Guang Lou, Murali Chintalapati, Furao Shen and Dongmei Zhang. 2019. “Robust log-based anomaly detection on unstable log data.” in Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software, pp. 807–817, Tallinn, Estonia, August 2019. DOI: 10.1145/3338906.3338931

[2] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. “Deeplog: Anomaly detection and diagnosis from system logs through deep learning”, in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1285–1298, October 2017, DOI: 10.1145/3133956.3134015

[3] Weibin Meng, Ying Liu, Yichen Zhu, Shenglin Zhang, Dan Pei, Yuqing Liu, Yihao Chen, Ruizhi Zhang, Shimin Tao, Pei Sun and Rong Zhou. “LogAnomaly: Unsupervised Detection of Sequential and Quantitative Anomalies in Unstructured Logs” in Proceedings of the 28th International Joint Conference on Artificial Intelligence, Vol. 7. pp. 4739–4745, August 2019, DOI: 10.5555/3367471.3367702

[4] Lin Yang, Junjie Chen, Zan Wang, Weijing Wang, Jiajun Jiang, Xuyuan Dong, and Wenbin Zhang. “Semi-supervised log-based anomaly detection via probabilistic label estimation.” in Proceedings of 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). IEEE, pp. 1448–1460, May 2021, DOI: 10.1109/ICSE43902.2021.00130

[5] Van-Hoang Le and Hongyu Zhang. “Log-based anomaly detection without log parsing”, in Proceedings of the 36th IEEE/ACM International Conference on Automated Software Engineering, pp. 492-504, Melbourne, Australia, November 2021, DOI: 10.1109/ASE51524.2021.00051

[6] Loghub, https://github.com/logpai/loghub

[7] Nengwen Zhao, Honglin Wang, Zeyan Li, Xiao Peng, Gang Wang, Zhu Pan, Yong Wu, Zhen Feng, Wenchi Zhang, Kaixin Sui and Dan Pei. “An Empirical Investigation of Practical Log Anomaly Detection for Online Service Systems” in Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software, pp. 1404-1415, Athens, Greece, August 2021. DOI: 10.1145/3468264.3473933

[8] Jieming Zhu, Shilin He, Pinjia He, Jinyang Liu and Michael R. Lyu. “Loghub: A Large Collection of System Log Datasets for AI-driven Log Analytics”, in Proceedings of 2023 IEEE 34th International Symposium on Software Reliability Engineering, pp. 355-366, Florence, Italy, October 2023, DOI: 10.1109/ISSRE59848.2023.00071

[9] Wei Xu, Ling Huang, Armando Fox, David Fatterson, and Michael I Jordan. “Detecting large-scale system problems by mining console logs”, in Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles, pp. 117-132, October 2009, DOI: 10.1145/1629575.1629587

[10] Adam Oliner and Jon Stearly. “What Supercomputers Say: A Study of Five System Logs”, in Proceedings of 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, pp. 575-584, July 2007, DOI: 10.1109/DSN.2007.103

[11] Qingwei Lin, Hongyu Zhang, Jian-Guang Lou, Yu Zhang and Xuewei Chen. “Log Clustering Based Problem Identification for Online Service Systems”, in Proceedings of 2016 IEEE/ACM 38th International Conference on Software Engineering Companion, pp. 102-111, Austin, TX, USA, May 2016, DOI: 10.1145/2889160.2889232

[12] Jian-Guang Lou, Qiang Fu, Shengqi Yang, Jiang Li and Bin Wu. “Mining Program Workflow from Interleaved Traces”, in Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining 2010, pp. 613-622, Wachington, DC, USA, July 2010. DOI: 10.1145/1835804.1835883

[13] Xiao Yu, Pallavi Joshi, Jianwu Xu, Guoliang Jin, Hui Zhang and Guofei Jiang. “CloudSeer: Workflow Monitoring of Cloud Infrastructures via Interleaved Logs”, in Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 489-502, Atlanta, Georgia, USA, March 2016. DOI: 10.1145/2872362.2872407

[14] R. Mahindru, H. Kumar and S. Bansal, "Log Anomaly to Resolution: AI Based Proactive Incident Remediation," in Proceedings of IEEE/ACM International Conference on Automated Software Engineering, Melbourne, Australia, 2021, pp. 1353-1357, DOI: 10.1109/ASE51524.2021.9678815.

[15] Baril Xavier, Coustié Oihana, Mothe Josiane and Teste Olivier. “Application Performance Anomaly Detection with LSTM on Temporal Irregularities in Logs.” in Proceedings of the 29th ACM International Conference on Information & Knowledge, pp. 1961-1964, October 2020, DOI: 10.1145/3340531.3412157.

[16] Chenxi Zhang, Xin Peng, Chaofeng Sha, Ke Zhang, Zhenging Fu, Xiya Wu, Qingwei Lin and Dongmei Zhang. "DeepTraLog: Trace-Log Combined Microservice Anomaly Detection through Graph-based Deep Learning," in Proceedings of the 44th International Conference on Software Engineering, Pittsburgh, PA, USA, 2022, pp. 623-634, DOI: 10.1145/3510003.3510180.

[17] Yintong Huo, Cheryl Lee, Yuxin Su, Shiwen Shan, Jinyang Liu and Michael R. Lyu. “EvLog: Identifying Anomalous Logs over Software Evolution” in Proceedings of 2023 IEEE 34th International Symposium on Software Reliability Engineering, pp. 391-402, Florence, Italy, October 2023, DOI: 10.1109/ISSRE59848.2023.00018

[18] Apache Spark Examples, https://github.com/apache/spark/tree/master/examples/src/main/scala/org/apache/spark/examples

Authors

Sion Min received the B.S. degree in Computer Science and Engineering from Kyungpook National University, Daegu, Korea in 2024. She is interested in log analysis, cloud computing, and distributed systems.

Youyang Kim received the M.S. degree in Computer Science and Engineering from Kyungpook National University, Daegu, Korea in 2023. She is interested in log analysis, troubleshooting, root cause analysis.

Byungchul Tak received his B.S. degree from Yonsei University in 2000, M.S. from KAIST in 2003 and Ph.D. degrees in Computer Science and Engineering from the Pennsylvania State University at University. Park in 2012. Dr. Tak joined the faculty of the Department of Computer Science at Kyungpook National University, Dague, Korea, in 2017. He is currently an Associate Professor in the Department of Computer Science. His research interests are in cloud computing, distributed systems, operating system and big data analytics.