논문 2021-16-20

도커 기반의 실시간 데이터 연계 및 처리 환경을 고려한 빅데이터 관리 플랫폼 개발

(Development of Big-data Management Platform Considering Docker Based Real Time Data Connecting and Processing Environments)

김 동 길, 박 용 순, 정 태 윤*
(Dong Gil Kim, Yong-Soon Park, Tae-Yun Chung)

Abstract : Real-time access is required to handle continuous and unstructured data and should be flexible in management under dynamic state. Platform can be built to allow data collection, storage, and processing from local-server or multi-server. Although the former centralize method is easy to control, it creates an overload problem because it proceeds all the processing in one unit, and the latter distributed method performs parallel processing, so it is fast to respond and can easily scale system capacity, but the design is complex. This paper provides data collection and processing on one platform to derive significant insights from various data held by an enterprise or agency in the latter manner, which is intuitively available on dashboards and utilizes Spark to improve distributed processing performance. All service utilize dockers to distribute and management. The data used in this study was 100% collected from Kafka, showing that when the file size is 4.4 gigabytes, the data processing speed in spark cluster mode is 2 minute 15 seconds, about 3 minutes 19 seconds faster than the local mode.

Keywords : Docker, Container, Big-data Management, Real-time Collection, In Memory Processing

Ⅰ. 서 론

인터넷의 발전으로 디지털 전환이 가속화되면서 모든 분야에서 급격하게 증가하는 데이터는 인공지능, 블록체인 등 혁신 기술들의 융합으로 만들어지는 4차 산업혁명에서 매우 중요한 역할을 담당하고 있다.

3개 부문으로 구분되는 데이터 산업은 그림 1과 같이 데이터 처리 및 관리 솔루션 개발을 의미하는 데이터 솔루션과 물리적 DB 설계, 데이터 이행, 데이터 가공, DB 성능 개선, 데이터 거버넌스, 데이터 분석 및 활용 등의 서비스를 의미하는 데이터 구축 및 컨설팅, 그리고 데이터 판매 및 제공을 의미하는 데이터 서비스가 있으며, 시장 규모는 데이터 솔루션 2조 409억 원, 데이터 구축 및 컨설팅 6조 4,922억 원, 데이터 서비스 8조 3,361억 원으로 모든 분야에서 성장하는 것으로 나타났다 [1].

최근에는 이러한 데이터에서 다양한 유형의 비즈니스 문제를 해결하는데 적합한 인사이트를 도출하기 위해 데이터를 안전하고 효율적으로 수집, 처리, 저장, 분석할 수 있는 데이터의 공유 및 활용에 대한 그 중요성이 더욱 강조되고

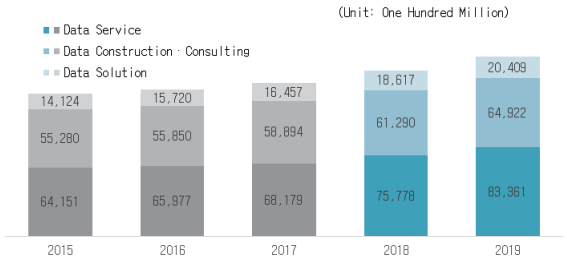

그림 1. 최근 5개년(2015~2019) 데이터 시장 규모
Fig. 1. Data Market Size During 5 Years(2015~2019)

있다 [2-5].

이에 본 연구에서는 데이터 사용을 최적화하여 조직에 대한 이익을 극대화할 수 있는 결정을 내리고 조치하기 위해 데이터를 분석하는데 필요한 빅데이터 관리 시스템에 초점을 두었다.

개발 과정은 데이터 수집 단계와 처리 단계로 구분되며, 어떤 환경으로든 실행에 필요한 모든 파일을 유연하게 사용할 수 있도록 하였다.

각 단계는 설치 환경 관점에서 다음과 같은 요구사항을 제시한다.

첫 번째, 대량의 데이터를 손실 없이 수신하고 전달하는데 필요한 요구사항으로 데이터의 분산 및 복제 구성을 손쉽게 할 수 있어야 하며, 데이터베이스, 파일시스템 등의 외부 시스템과 접속할 수 있어야 한다.

*Corresponding Author (tychung@gwnu.ac.kr)
Received: Jun. 15, 2021, Revised: Jul. 12, 2021, Accepted: Jul. 26, 2021.
D.G. Kim: Gang-won Research Institute of ICT Convergence (Ph.D)
Y.S. Park: Gang-won Research Institute of ICT Convergence (Ph.D. Student)
T.Y. Chung: Gangneung-Wonju National University (Prof.)
※ 이 논문은 2021년도 정부 (산업통상자원부)의 재원으로 한국산업기술진흥원의 지원을 받아 수행된 연구임 (R0006229, 차세대 생명·건강산업 생태계 조성사업).

ISSN : 1975-5066
http://dx.doi.org/10.14372/IEMEK.2021.16.4.153

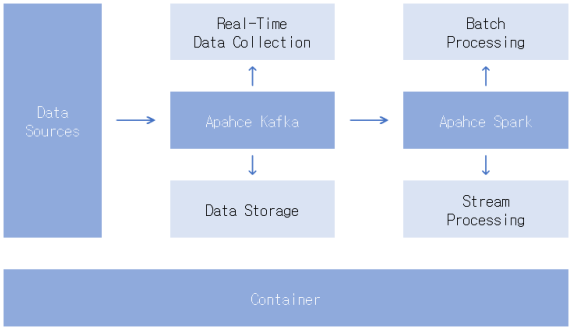

그림 2. 빅데이터 플랫폼의 구성 요소
Fig. 2. Components of the Bigdata Platform

두 번째, 머신러닝이나 딥러닝 등 반복적인 데이터 처리에 필요한 요구사항으로 빠른 성능을 위해 디스크가 아닌 메모리에서 수행되어야 한다.

세 번째, 실행에 필요한 서비스를 모두 패키지화하고 분리하는데 필요한 요구사항으로 구축, 배포, 복사 등을 다양한 환경에서 동일하게 이용할 수 있어야 한다.

본 연구의 목적은 그림 2와 같이 컨테이너 환경에서 카프카를 통해 수집된 원본 데이터를 스파크에서 처리할 수 있도록 시스템을 설계하고 구축하는 것이다. 먼저 원본 데이터를 확보하기 위해 도커에 웹 크롤러 관련 소스를 추가하였다. 이후 데이터를 실시간으로 수집하고 타 플랫폼과 연계할 수 있도록 도커에 카프카 관련 소스를 추가하였다. 마지막으로 수집된 데이터는 클러스터로 구성된 스파크에서 자원을 공유하여 신속하게 처리할 수 있도록 도커에 관련 소스를 추가하였다.

본 연구는 4차 산업혁명 시대의 도래로 데이터 경제가 급변하고 있는 상황에서 데이터 레이크 또는 데이터 웨어하우스에 데이터를 안정적으로 저장하고 처리할 수 있는 빅데이터 관리 시스템의 필요성을 제기했다는 점에서 의의가 있다고 할 수 있으며, 관련 서비스를 가상화 기술로 제시했다는 점도 시사하는 바가 크다고 할 수 있다.

본 연구의 구성은 다음과 같다. 2장에서 본 연구와 관련된 문헌을 고찰하고, 3장에서 제안하는 시스템을 설계하는데 필요한 기술들을 상세히 설명하였으며, 4장에서 결론 및 향후 계획을 정리하여 글을 마친다.

Ⅱ. 관련 연구

학술연구정보서비스에서 국내·외 학술 논문을 대상으로 최근 10년간 '도커'로 검색한 결과 그림 3과 같이 컨테이너 분야의 연구 빈도가 점진적으로 증가하고 있지만, 빅데이터 플랫폼에 컨테이너 기술을 활용한 연구는 상대적으로 매우 낮은 것으로 나타났다 [6].

이 중에서 본 연구자가 2021년 한국 ITS 학회 춘계학술대회에서 발표한 '도커 기반 빅데이터 수집 및 처리 엔진에 관한' 논문을 제외하면 [7], 서비스의 배포, 자동화 방법에 관한 솔루션을 도커 기반으로 제공하는 연구 [8], 애플리케

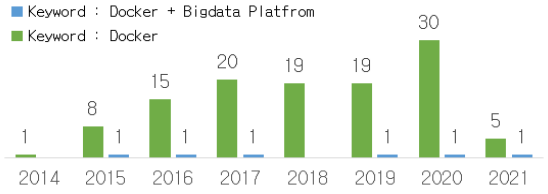

그림 3. 문헌 조사 : 키워드 (도커)
Fig. 3. Literature Review : Keyword (Docker)

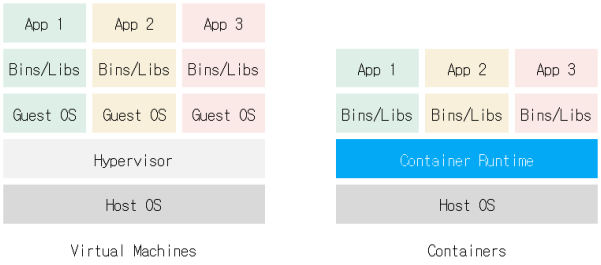

그림 4. 가상화와 컨테이너 비교
Fig. 4. Virtualization vs. Container

이션과 해당 환경에 상관없이 표준화된 소프트웨어 컨테이너 패키징 기술 (AlgoRun)을 도커 기반으로 구축하는 연구 [9], 엣지 디바이스에 필요한 아키텍처를 도커 기반으로 진행하는 연구 [10], 도커 컨테이너와 인스턴스를 관리하기 위한 쿠버네티스 연구 [11], DarkNet, Yolo, MaskRCNN 등의 딥러닝 기술을 통해 데이터를 분석하는 연구와 같이 [12], 데이터 수집 및 처리에 필요한 플랫폼이 아닌 컨테이너를 활용하는 기술이나 컨테이너에서 데이터를 분석하여 데이터의 인사이트를 확인한다는 점에서 본 연구와는 다르다는 것을 알 수 있다.

1. 서비스 배포 및 관리를 위한 도커 사용

도커는 컨테이너 기반의 오픈 소스 가상화 플랫폼으로 클라우드 컴퓨팅 환경에서 관리해야 하는 서버가 다양해짐에 따라 각종 소프트웨어의 설치와 배포를 손쉽게 해결하고, 컨테이너에서 다양한 종류의 서비스들을 이미지화하여 누구나 활용할 수 있기 때문에, 여러 사용자가 서버나 클라우드 환경 등 다양한 환경에서 사용할 수 있다 [13].

컨테이너는 그림 4와 같이 실제 구동 환경으로부터 추상화할 수 있는 애플리케이션을 쉽게 구성할 수 있기 때문에 개별 앱 구성과 관련된 세부 업무에 시간을 낭비하지 않고 배포 및 관리에 집중할 수 있다 [14].

2. 데이터 수집 및 연계를 위한 카프카 사용

카프카 이전에 데이터를 수집하는 과정에서 발생할 수 있는 문제는 다음과 같다.

첫 번째, 서버에 점점 더 많은 이벤트가 생기면서 높은 대기 시간이 필요해진다.

두 번째, 모든 기록 데이터에 새로운 시스템이나 DB를 도입하고 재구성하는 것에는 한계가 있다.

세 번째, 더 깊은 인사이트를 얻기 위해 모든 데이터에 대해 다양한 데이터 처리나 인공지능 모델을 학습하는 것은 불가능하다.

네 번째, 애플리케이션을 통해 데이터를 교환할 때 데이터 변형이 필요한 경우 변환이 쉽지 않다.

카프카는 이를 해결하기 위한 분산 메시지 배포 및 관리를 위한 메시징 시스템으로 발행 및 구독 모델을 분산 환경에서 처리하기 위한 분산 스트리밍 플랫폼이며, 메시지 배포의 안정성을 위해 다수의 서버로 구성된 클러스터로 운용된다 [15-17].

메시지 전송과 관련된 기존 서비스와 카프카 성능을 비교해보면 그림 5와 같이 카프카는 다른 서비스와 다르게 메시지 상실을 허용하지 않기 때문에 다음과 같은 특징이 있다.

첫 번째, 데이터의 연속적인 흐름을 수신하거나 전송할 수 있다.

두 번째, 실시간으로 애플리케이션으로 도와주는 과정을 제공한다.

세 번째, 분산형 플랫폼으로 규격에 맞게 구축할 수 있기 때문에 빈번한 읽기 및 쓰기나 대용량 데이터 볼륨을 쉽게 저장할 수 있다.

네 번째, 다양한 제품과 시스템에 쉽게 연동되어 약간의 커밋 정보만으로도 커밋 프로세스를 자동으로 관리하고, 기존 그룹 관리 프로토콜을 기반으로 동작하기 때문에 워커를 추가할 때마다 확장할 수 있다.

다섯 번째, 오픈 소스 라이선스를 따르기 때문에 무료로 사용할 수 있다.

3. 데이터 처리를 위한 스파크 사용

스파크는 인 메모리 컴퓨팅 및 분산 처리를 지원하며, 작업 처리를 관리하는 마스터 노드와 드라이버로부터 분산된 작업을 할당받아 처리하고 결과를 반환하는 하나 이상의 슬레이브 노드로 구성된다 [18].

속도 측면에서 기존 맵리듀스 프레임워크가 효율적으로 동작하지 못했던 반복적인 머신러닝 알고리즘과 대화형 쿼리 형태의 작업에서 놀라운 성능 향상을 보이며, 하둡에 비해 5~10배 빠른 결과를 보인다 [19].

스파크는 그림 6과 같이 운영 및 제어를 담당하는 드라이버 프로그램(Spark Context), 자원을 관리하는 클러스터 매니저, 실제 작업에 사용되는 워커 노드, CPU와 메모리 등의 자원을 할당받는 익스큐터로 구성되어 있다 [20].

Ⅲ. 제안 모델

본 연구에서 제안하는 모델은 도커 컨테이너를 기반으로 데이터 확보-연동-수집-처리 단계로 구성된다.

빅데이터 관리 플랫폼은 그림 7과 같이 도커 설치 후 ① 원본 데이터를 확보하기 위한 웹 크롤링 서비스 ② 데이터베이스와 연동하는 서비스 ③ 연동된 데이터를 카프카에서 수집 및 모니터링하는 서비스 ④ 수집된 데이터를 스파크에서 처리 및 모니터링하는 서비스로 요약할 수 있다.

○ Yes × No

Requirements	Message Q	Log Collection	ETL Tool	Kafka
Real-Time	○	○	×	○
Expandability	×	○	○	○
Persistence	×	×	×	○
Diversity	○	×	○	○
Transmission	○	×	○	○

그림 5. 기존 메시지 전송 서비스와 카프카 성능 비교
Fig. 5. Performance Comparison of Existing Message Transmission and Kafka

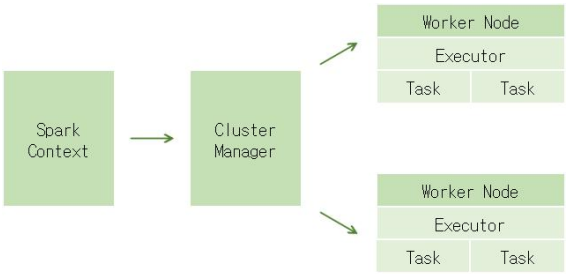

그림 6. 스파크 클러스터 모드
Fig. 6. Spark Cluster Mode

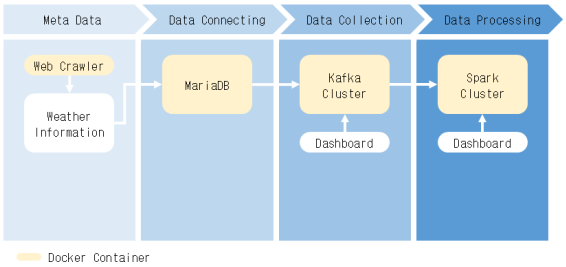

그림 7. 빅데이터 관리 플랫폼 흐름도
Fig. 7. Bigdata Management Platform Flow Chart

1. 도커 설치

1.1 도커 엔진

도커 엔진은 가상화를 수행하는데 필요한 그래픽 자원, 디스크 용량, 메모리 사용량 등의 리소스를 효율적으로 관리할 수 있다 [21].

설치 과정은 운영체제별로 달라지며, 본 연구에서는 우분투에서 도커 엔진을 설치하는 과정을 그림 8과 같이 설명하고자 한다 [22].

우분투 16.04 지원이 종료되어 18.04 버전부터 설치할 수 있다. 먼저 서버에 이전 버전이 설치되어 있다면 삭제가 필요하며, 새로운 버전을 설치하기 전에 패키지를 업데이트하고 저장소를 설정해야 한다. 이후 도커에서 제공하는 공식 GPG (GNU Privacy Guard) 키를 추가하여 x86_64/amd64 환경을 설정하고 최신 버전의 도커 엔진을 설치할 수 있다.

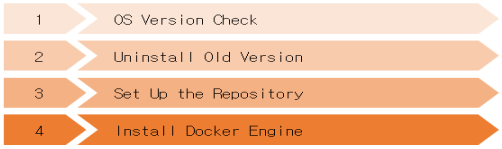

그림 8. 우분투에서 도커 엔진 설치하기
Fig. 8. Install Docker Engine on Ubuntu

그림 9. 우분투에서 도커 컴포즈 설치하기
Fig. 9. Install Docker Compose on Ubuntu

도커가 설치되면 이미지를 내려받기 위한 pull 명령어, 컨테이너 생성 및 접속을 위한 run 명령어, 실행 중인 컨테이너 목록을 확인하기 위한 ps 명령어, 실행 중인 컨테이너 내부에 접속하기 위한 exec 명령어가 많이 사용된다.

1.2 도커 컴포즈

도커 컴포즈는 도커 애플리케이션을 정의하고 실행하기 위한 도구를 의미하며, 서비스를 구성하는데 사용되는 파일 (YAML)은 공백 문자를 이용한 들여쓰기 구조체로 콜론 기호를 이용해서 키:값의 해쉬 형태로 한 줄에 표현하여 데이터를 상대적으로 이해하기 쉽도록 개발된 컴퓨터 언어를 의미한다 [23-24].

설치 과정은 도커 엔진과 다르게 매우 간단하며, 이는 그림 9와 같다.

앞에서 설치된 도커가 우분투 버전이기 때문에 리눅스에서 설치를 진행해야 하며, 깃허브의 컴포즈 저장소에서 최신 버전의 도커 컴포즈를 내려받을 수 있다. 도커 컴포즈가 설치되면 도커에서 개별적으로 실행되던 서비스들을 up 명령어를 통해 한 번에 실행할 수 있다.

2. 데이터

본 연구에서 사용되는 웹 크롤러 데이터는 그림 10과 같이 웹 페이지 사이를 이동하며 HTML 구조를 파싱하여 웹 링크 주소와 웹 데이터를 검색하고 수집하는 방식으로 수집의 양과 범위의 한계가 없다는 장점이 있으며 [25], 사용 방법은 그림 11과 같다.

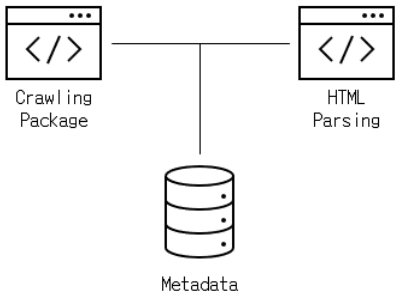

그림 10. 원본데이터 수집 방법
Fig. 10. To Collection Metadata

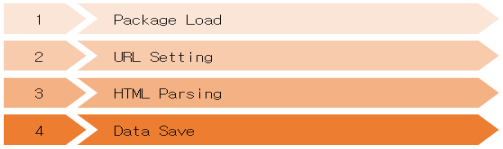

그림 11. 웹 크롤러 사용
Fig. 11. Use a Web Crawler

표 1. 웹 크롤러 데이터 정보
Table 1. Web Crawler Data Information

Date	Current temp	Dust	Ultra dust	Ozone
20210512	25	34㎍/m3	18㎍/m3	0.028ppm
20210512	24	33㎍/m3	20㎍/m3	0.031ppm
20210512	24	34㎍/m3	18㎍/m3	0.031ppm
⋮	⋮	⋮	⋮	⋮

웹 크롤러를 사용하기 위해서는 파이썬 패키지와 데이터 추출에 필요한 특정 사이트의 URL이 필요한데, 네이버 날씨 정보를 통해 이를 해결하고자 한다. 이후 패키지와 URL 설정이 완료되면 온도, 미세먼지 등의 정보를 파싱하여 CSV 파일로 저장할 수 있다. 이렇게 저장된 데이터는 표 1과 같다.

웹 크롤러를 통해 추출된 데이터는 날짜, 온도, 미세먼지, 초미세먼지, 오존 지수로 나타났으며, 이렇게 생성된 데이터는 카프카를 통해 수집되고 모니터링이 가능하며, 스파크에서 배치 처리, 스트리밍 등의 연산을 수행한다.

3. 카프카 클러스터 구축

본 연구에서 제안하는 카프카 클러스터는 대량의 데이터를 처리하는 분산 메시징 시스템으로 데이터를 수신하고 전달받은 데이터를 다른 시스템이나 장치에 보내기 위해 사용된다.

이러한 카프카 클러스터의 주요 구성 요소는 그림 12와 같이 데이터를 수신하고 전달하는 브로커, 메시지를 전송하는 프로듀서, 메시지를 취득하는 컨슈머, 분산 메시징의 메타 데이터를 관리하는 주키퍼로 구성된다.

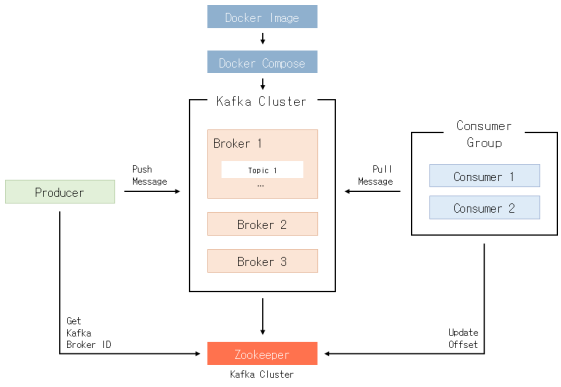

그림 12. 카프카 클러스터 구조
Fig. 12. Kafka Cluster Architecture

3.1 기존 카프카 클러스터의 단점

카프카는 사전 준비사항으로 설치 대상 서버에 Java 8+ 이상의 버전을 요구하며, 운영체제별로 설치 방법이 달라진다. 자바 설치가 완료되면 카프카 클러스터 설치 전에 주키퍼를 먼저 설치하고, 실행을 통해 작동 여부를 점검하여 다음 과정으로 진행할 수 있다. 이후 주키퍼에 오류가 없다면 카프카를 설치하는데, 다수의 서버를 하나의 연결하여 하나의 클러스터로 구성하기 때문에 브로커 정보, 카프카에 사용되는 호스트 아이피 및 포트 정보, 데이터가 저장될 폴더 및 파일 크기 단위 정보, 데이터 보관 기간 정보, 주키퍼 정보 등 환경 설정이 복잡하다. 또한 데이터를 추가하거나 기존 내용을 변경하는 과정에서 클러스터의 대기열 수가 증가하면 카프카 성능이 저하되는 원인이 되며 이를 관리하거나 감시할 수 있는 모니터링 도구가 부족하다. 그리고 이러한 모든 서비스는 로컬 환경에서 설치를 진행하기 때문에 서버 환경 (포맷, 패키지 업데이트, 라이브러리 버전 등)에 따라 문제가 발생한 서비스를 확인하고 재수정하는 등 배포와 관리가 쉽지 않고, 문제를 해결하는 과정에서 많은 시간이 소요된다.

3.2 도커 기반 카프카 클러스터의 장점

도커의 특징은 서비스에 필요한 모든 설정을 하나의 이미지로 저장하여 컨테이너별로 실행할 수 있다는 것이다. 이는 기존 카프카 클러스터를 구성하는데 필요한 모든 라이브러리나 프레임워크 등을 도커 이미지로 묶어서 공유하기 때문에 특정 환경에서만 실행되는 문제가 발생하지 않아 버그를 효율적으로 수정할 수 있게 된다.

따라서 이러한 특징을 정리하면 표 2와 같이 기존 카프카 클러스터와 비교하여 차이점을 확인할 수 있다.

3.3 도커 기반 카프카 클러스터 설치

먼저 카프카 클러스터에 필요한 설정을 Dockerfile 파일에 저장하면 각각의 서비스는 그림 13과 같이 docker-compose -build -d 명령어를 통해 한 번에 실행할 수 있다.

표 2. 카프카 클러스터 비교
Table 2. Comparison of Kafka Clusters

Comparison Factor	Existing	Suggestion
Database Connect	×	O
Data Generate	×	O
Data Change	×	O
Data Linkage	×	O
Environment Setup	△ (Slow)	O (Fast)
Run Time	△ (Slow)	O (Fast)
Resource Management	×	O
Shared Environment	×	O
Deployment and Management	△ (Difficult)	O (Easy)

```
Creating mariadb ... Done
Creating kafka-schema-registry ... Done
Creating kafka-rest-proxy ... done
Creating kafka-connect ... done
Creating kafka-topics ... done
```

그림 13. 카프카 클러스터 서비스
Fig. 13. Kafka Cluster Services

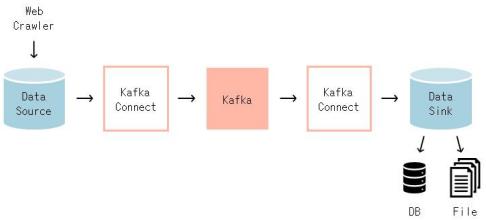

그림 14. 카프카 커넥트 구조
Fig. 14. Kafka Connect Architecture

```
echo '
{
  "name" : "kafka-connect",
  "config" : {
    "connector.class" : "io.confluent.connect.jdbc.JdbcSourceConnector",
    "connection.url" : "jdbc:mysql://mariadb:3306/testdb",
    "connection.user" : "root",
    "connection.password" : "passwd",
    "consumer.override.auto.offset.reset": "latest",
    "mode": "incrementing",
    "incrementing.column.name" : "currenttemp",
    "table.whitelist" : "test",
    "topic.prefix" : "web-crawler-",
    "tasks.max" : "3"
  }
}
' | curl -X POST -d @- http://localhost:8083/connectors --header "content-
Type:application/json"
```

그림 15. 카프카 커넥트 소스 명령어
Fig. 15. JDBC Source Kafka Connector for REST API

이러한 서비스를 정리해보면, 카프카 Connect는 MariaDB에 연계하기 위해, Rest Proxy는 데이터를 전송하기 위해, Schema Registry는 데이터 포맷을 확인하기 위해, Topics은 전송된 데이터를 확인하기 위해 사용된다.

3.4 카프카 커넥트를 통한 MariaDB 데이터 연계

데이터가 준비되면 그림 14와 같이 다른 플랫폼과 연결 및 작업을 위해 데이터를 카프카로 보낼 때 사용되는 JDBC (Java Database Connectivity)의 Source와 데이터를 카프카로부터 받아내는 JDBC의 Sink로 구성된 환경에서 REST API 제출하여 데이터베이스와 카프카를 연결하거나 로컬 시스템에 CSV나 TXT 등의 파일로 저장할 수 있다.

카프카 커넥트 소스에서 사용된 REST API 명령어는 그림 15와 같이 실행하는 커넥터의 이름과 클래스를 설정하고, 실행 중일 때 변경된 내용을 카프카에서 확인하는 과정을 추가한 뒤 대상 테이블과 토픽, 태스크 수를 지정하여 제출할 수 있으며, 이후 연계된 데이터는 카프카 대시보드에서 확인할 수 있다.

웹 화면에서 호스트 아이피와 포트 번호 8000으로 접속하면 그림 16과 같은 대시보드를 사용할 수 있으며, 토픽 (web-crawler-test)을 선택하면 Key, Value 등의 세부 정보를 확인할 수 있다.

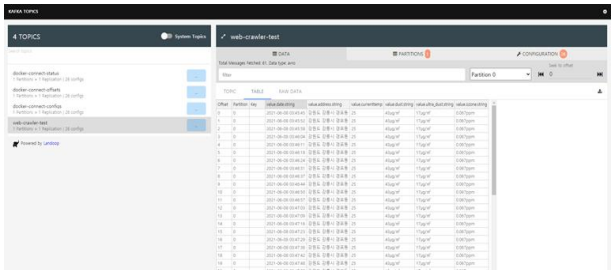

그림 16. 카프카 대시보드
Fig. 16. Kafka Dashboard

```
echo '
{ "name" : "kafka-connect-file",
  "config" : {
    "connector.class" : "org.apache.kafka.connect.file.FileStreamSinkConnector",
    "file" : "/weather.txt",
    "topics" : "web-crawler-test"
  }
}' | curl -X POST -d @- http://localhost:8083/connectors --header "content-
Type:application/json"
```

그림 17. 카프카 커넥트 싱크 명령어
Fig. 17. JDBC Sink Kafka Connector for REST API

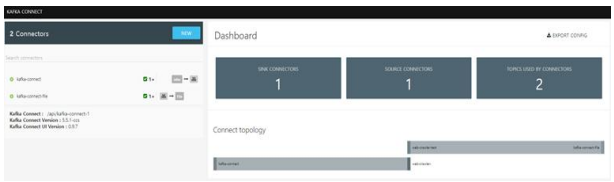

그림 18. 카프카 커넥트 대시보드
Fig. 18. Kafka Connect Dashboard

이렇게 수집된 데이터는 그림 17의 방법으로 카프카 커넥트 컨테이너 내부에 CSV 파일로 저장이 되며, 파일 위치는 루트 (root)에서 확인할 수 있다.

제출 여부는 그림 18과 같이 웹 화면에서 포트 번호 8003으로 접속하여 카프카 커넥트에 연결된 데이터베이스 정보와 연결된 토픽 목록을 확인할 수 있다.

3.5 실시간 데이터 수집 결과

테스트에 사용된 데이터는 표 3과 같이 2021년 5월 1일부터 31일까지 한 달 동안 카프카 커넥트를 통해 수집된 네이버의 날씨 정보로 시간 (Date), 위치 (Address), 온도 (Currenttemp), 미세먼지 (Dust), 초미세먼지 (Ultra _dust), 오존지수 (Ozone) 항목으로 나타났으며, 약 1억 2천 건의 데이터를 확인할 수 있다.

표 3. 카프카 클러스터에 사용된 테스트 데이터
Table 3. Test Data Use for Kafka Cluster

Collection	2021.05.01~2021.05.31
Sample	122,408,524
Variable	Date, Currenttemp, Dust, Ultra_dust, Ozone
Type	CSV
Size	4.4GB

```
Python 3.8.3 (default, Jul  2 2020, 16:21:59)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> Import pandas as pd
>>> df = pd.read_csv(web-crawler.csv)
>>> df.isnull().sum()
date            0
currenttemp     0
dust            0
ultradust       0
ozone           0
dtype: int64
>>> df["date"].head().values
Array([2021.05.01 10:01:03, 2021.05.01 10:01:06, 2021.05.01 10:01:09,
2021.05.01 10:01:12, 2021.05.01 10:01:15)
>>> df["date"].tail().values
Array([2021.05.31 17:27:39, 2021.05.31 17:27:42, 2021.05.31 17:27:45,
2021.05.31 17:27:48, 2021.05.31 17:27:51)
```

그림 19. 카프카 커넥트를 통한 테스트 결과
Fig. 19. Test Result from Kafka Connect

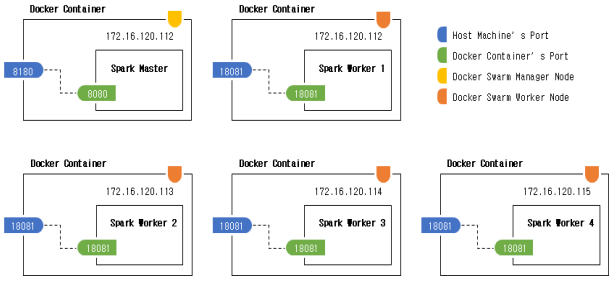

그림 20. 스파크 클러스터 구조
Fig. 20. Spark Cluster Architecture

수집된 데이터의 누락된 값과 정해진 시간 내에서 데이터가 정상적으로 수신되었는지 확인하기 위해 파이썬이 설치되어 있는 로컬 환경에서 그림 19의 방법으로 테스트를 진행하였다.

한 달 동안 수집된 4GB의 데이터를 확인한 결과 누락된 값은 존재하지 않으며, 추출 값 또한 3초 간격으로 데이터가 정상적으로 수신되는 것을 확인하였다.

4. 스파크 클러스터 구축

본 연구에서 제안하는 스파크 클러스터는 여러 대의 서버에서 작동하는 방식으로 마스터에서 하나의 애플리케이션을 담당하는 드라이버가 Spark Context를 생성하고 관리하면, 이를 실제로 담당하고 처리하는 워커에서 CPU 코어, 메모리 등의 자원을 할당하는 익스큐터를 실행한다.

이러한 스파크 클러스터는 그림 20과 같이 도커 스웜을 통해 여러 대의 서버를 하나의 서버처럼 연결할 수 있으며, 자원을 추가하고 메모리를 이용한 데이터 저장 방식을 제공함으로써 머신러닝 등 반복적인 데이터 처리가 필요한 분야에서 높은 성능을 보인다.

4.1 기존 스파크 클러스터의 단점

스파크 클러스터는 마스터 인스턴스를 실행하고, 그 뒤에 워커 인스턴스 연결이 필요하다. 스파크 마스터는 클러스터 내의 워커 노드 중 하나에서 실행되며, 스파크 드라이브와 Spark Context를 통해 조정된다. 먼저 다수의 서버를 연결하기 위해 오버레이 네트워크 등의 통신 설정과 SSH 등 네

트워크 보안 설정 작업이 필요하다. 통신 설정이 완료되면, 마스터와 워커 노드별로 호스트와 포트 정보를 설정하고, 카프카 클러스터에서 설치한 자바 경로를 스파크 환경 변수에 반영해야 한다. 그리고 스파크 마스터 노드가 설치된 서버에서 스파크를 실행하여 서비스를 구축할 수 있으며, 워커 노드에서 처리되는 데이터 상태를 확인하기 위한 Jobs 설정, REST API와 관련된 설정, 스파크 마스터와 워커 노드 설정 등 환경 설정이 복잡하다. 또한 CPU, 메모리 등의 자원 추가가 필요한 경우 기존 서비스를 종료하고 다시 시작해야 연결 오류가 낮아지기 때문에 효율적인 자원 관리에 한계가 있다. 그리고 이러한 모든 서비스는 로컬 환경에서 설치를 진행하기 때문에 네트워크 설정에 따라 노드 연결에 실패하거나 서버 환경에 따라 문제가 발생한 서비스를 확인하고 재수정하는 등 배포와 관리가 쉽지 않고, 문제를 해결하는 과정에서 많은 시간이 소요된다.

4.2 도커 기반 스파크 클러스터의 장점

도커의 특징은 앞에서 언급한 바와 같이 스파크 클러스터에 필요한 모든 라이브러리나 프레임워크 등을 도커 이미지로 묶어서 공유하기 때문에 특정 환경에서만 실행되는 문제가 발생하지 않아 버그를 보다 효율적으로 관리하고 대응할 수 있다.

따라서 이러한 특징을 정리하면 표 4와 같이 기존 스파크 클러스터와 비교하여 차이점을 확인할 수 있다.

4.3 도커 기반 스파크 클러스터 설치

스파크 클러스터를 구축하기 위해서는 먼저 그림 21과 같이 각각의 호스트에서 사용하고 있는 네트워크와 상관없이 컨테이너 간에 네트워크가 연결되어 새로운 서비스가 추가되는 경우 미리 생성한 오버레이 네트워크의 서브넷 IP를 할당받아 컨테이너를 관리할 수 있는 네트워크 설정이 필요하며, 도커에서 제공되는 도커 스웜 모드를 통해 사용할 수 있다.

그리고 스파크 클러스터에 필요한 설정이 Dockerfile 파일에 저장되면 앞에서 언급한 빌드 명령어를 통해 한 번에 모두 실행할 수 있으며, 관련 서비스는 그림 22와 같다.

이러한 서비스를 정리해보면, 카프카 스파크 마스터는 클러스터 내의 리소스 관리 역할을 담당하며, 스파크 워커는 CPU 코어와 메모리 할당량 등 계산과 실제 할당된 처리를 실행한다.

스파크 클러스터는 웹 화면에서 호스트 아이피와 포트 번호 8080으로 접속하면 그림 23과 같은 대시보드를 통해 연결된 노드 정보와 자원을 확인할 수 있다.

대시보드를 확인한 결과 스파크 클러스터에서 사용 가능한 자원은 Memory 106.7GB와 Cores 40개로 나타났다.

4.4 데이터 처리 속도 비교

데이터 처리 속도를 비교하기 위해 로컬 환경과 스파크 클러스터 환경에서 테스트를 진행하였으며, 테스트에 사용된 데이터는 카프카 커넥트에서 수집된 네이버 날씨 정보로 표 5와 같이 4단계로 분할된 데이터를 통해 처리 시간을 비교하고자 한다.

표 4. 스파크 클러스터 비교
Table 4. Comparison of Kafka Clusters

Comparison Factor	Existing	Suggestion
Network Setting (SSH, Port, etc.)	△ (Independent)	O (Universal)
Data Processing	O	O
Multi-Server Connecting	×	O
Environment Setup	△ (Slow)	O (Fast)
Run Time	△ (Slow)	O (Fast)
Resource Management	×	O
Deployment and Management	△ (Difficult)	O (Easy)

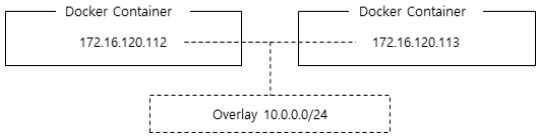

그림 21. 도커 스웜 오버레이 네트워크
Fig. 21. Docker Swarm Overlay Network

```
Creating spark-master ... Done
Creating spark-worker1 ... Done
Creating spark-worker2 ... done
```

그림 22. 스파크 클러스터 서비스
Fig. 22. Spark Cluster Services

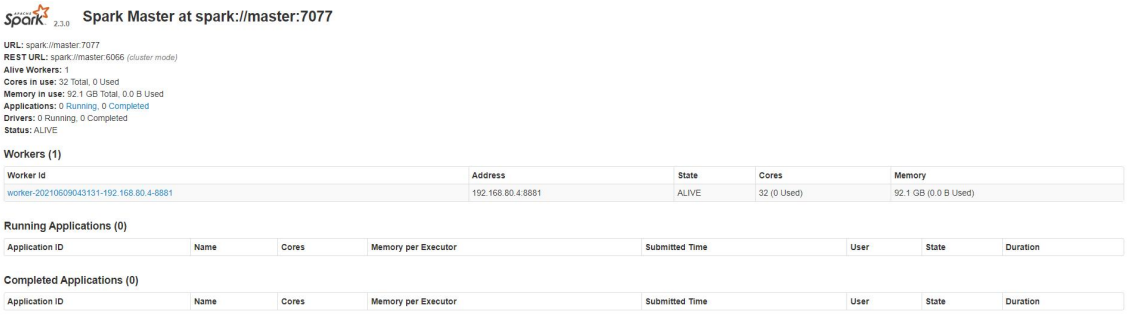

그림 23. 스파크 대시보드
Fig. 23. Spark Dashboard

표 5. 수집된 데이터 분리
Table 5. Split of Collect Data

Step	Date	Size
weather_1	2021.05.01~2021.05.07	0.87GB
weather_2	2021.05.08~2021.05.14	0.98GB
weather_3	2021.05.15~2021.05.21	1.13GB
weather_4	2021.05.22~2021.05.31	1.42GB

표 6. 로컬 환경과 스파크 클러스터 환경 비교
Table 6. Comparison of Local and Spark Cluster Environments

Step	Dataset	Local Mode	spark Cluster Mode
1	weather_1	00:01:48	00:00:38
2	weather_2	00:02:14	00:00:55
3	weather_3	00:02:49	00:01.16
4	weather_4	00:03:26	00:01:44
5	weather_1~4 combine	00:05:34	00:02:15

각각의 결과는 표 6에서 확인할 수 있으며, 분할된 데이터를 모두 병합하는 과정이 추가되었다.

모든 결과에서 로컬 환경보다 스파크 클러스터 환경의 데이터 처리 속도가 더 우수한 것으로 나타났으며, 모든 데이터가 병합된 5단계에서 스파크 클러스터 환경의 처리 속도가 로컬 환경보다 3분 19초 더 빠른 것으로 나타났다.

Ⅳ. 결 론

본 연구는 도커 기반의 실시간 데이터 연계 및 처리 환경을 고려하여 소멸하기 쉬운 정보를 빨리 파악하고 조치할 수 있는 빅데이터 관리 플랫폼으로 다음과 같은 특징이 있다.

첫 번째, 카프카 클러스터에서 데이터를 수집하는 기존 방식은 호스트 서버에 원본 데이터를 1차로 저장하고, 저장된 파일을 카프카로 전달하여 2차로 데이터를 확인하고, 확인된 데이터를 다른 데이터베이스로 저장하기 위해 3차로 데이터를 가공하는 과정이 필요하다. 이에 본 연구에서 제안하는 카프카 클러스터는 로컬 시스템이 아닌 특정 데이터베이스 (MariaDB)에 데이터가 실시간으로 수집되는 방식으로 특정한 작업 형태를 카프카 커넥트에서 실행함으로써 불필요한 반복 작업을 줄일 수 있다. 이를 통해 데이터 파이프라인 생성이 가능해지면서 다른 플랫폼에 원본 데이터 (1차)를 연계하게 되면 별도의 추가 작업 없이 로컬이나 다른 시스템에 저장하거나 처리하는 등의 작업이 가능해진다.

두 번째, 스파크 클러스터를 구축하는 기존 방식은 타 호스트 간의 연결을 위한 SSH 설정이 필요하며, 마스터와 워커 노드에 해당하는 서버에서 호스트 정보를 입력하고, 실행에 필요한 설정 파일들을 각각의 서비스 모듈에서 실행하는 과정이 복잡하기 때문에 이와 관련된 전문 지식이 부족하다면 연결이 쉽지 않아 단일 서버인 로컬 환경을 많이 사용하게 된다. 이에 본 연구에서 제안하는 스파크 클러스터는 호스트 아이피가 다른 여러 대의 서버를 하나처럼 관리할 수 있는 도커 스웜 모드를 제공하며, CPU, 메모리 등의 자원이 추가로 필요할 경우 서비스 업데이트를 통해 대응할 수 있는 기능을 제공한다. 또한 카프카에서 연계된 데이터나 다른 데이터베이스를 통해 연계된 데이터를 스파크에서 직접 분석할 수 있기 때문에 데이터를 처리하는 시간을 단축할 수 있다.

세 번째, 모든 서비스는 도커 기반에서 실행되기 때문에 서버 오류, 재설치, 포맷 등의 다양한 문제에서 빠르게 대응할 수 있다.

본 플랫폼은 깃랩 (GitLab) 프로젝트에서 확인 가능하며 [26], 다운로드부터 구축까지 연구자가 별도의 작업 없이 간편하게 사용할 수 있음을 확인하였다.

추후 연구에서는 스파크 클러스터에서 데이터 분석이 가능한 주피터 노트북과 같은 웹 대시보드 환경을 추가하여 인공지능 모델 학습 및 결과를 확인할 수 있는 리포팅 기능을 제공할 계획이다.

References

[1] H.R. Yu, Y.M. Na, “2020 Data Industrial White-paper: Market Status,” Korea Data Agency, No. 23, pp. 110-141, 2020 (in Korean).

[2] H.S. Son, “A Study on the Legislative Direction of Data Acts about Digital New Deal,” Korean Constitutional Law Association, Vol. 27, No. 1, pp. 203-252, 2021 (in Korean).

[3] H.L. Kim, “A Knowledge Model of Data Map for Semantically Representing National Data,” Journal of Digital Contents Society, Vol. 22, No. 3, pp. 491-499, 2021 (in Korean).

[4] J.M. Bang, “Data-based Administration for the Substantialization of Public Data Utilization-Focused on US Federal Data Strategy-,” The Journal of Comparative Law, Vol. 21, No. 1, pp. 87-126, 2021 (in Korean).

[5] S.M. Chun, S.Y. Suk, “Development and Implementation of Smart Manufacturing Big-Data Platform Using Opensource for Failure Prognostics and Diagnosis Technology of Industrial Robotl,” IEMEK J. Embed. Sys. Appl., Vol. 14, No. 4, pp. 187-195, 2019 (in Korean).

[6] http://www.riss.kr/index.do

[7] D.G. Kim, T.Y. Chung, “A Study on Bigdata Collection and Processing ENgine of Docker Based,” Korea Institute of Intelligent Transportation Systems Proceedings of the Spring Conference, pp. 593-594, 2021 (in Korean).

[8] P. Hoenisch, W. Ingo, S. Stefan, Z. Liming, F. Alan,

"Four-Fold Auto-Scaling on a Contemporary Deployment Platform Using Docker Containers," Lecture Notes in Computer Science, Vol. 2015, No. 9435, pp. 316-323, 2015.
[9] A. Hosny, V.L. Paola, R. Laubenbacher, T. Favre, "AlgoRun: a Docker-based Packaging System for Platform-agnostic Implemented Algorithms," Bioinformatics, Vol. 32, No. 15, pp. 2396-2398, 2016.
[10] A. Eiermann, M. Renner, M. Großmann, K. Udo R. "On a Fog Computing Platform Built on ARM Architectures by Docker Container Technology," Communications in Computer and Information Science, Vol. 2017, No. 717, pp. 71-86, 2017.
[11] C.W. Tien, T.Y. Huang, C.W. Tien, T.C. Huang, S.Y. Kuo, "KubAnomaly: Anomaly Detection for the Docker Orchestration Platform with Neural Network Approaches," Engineering reports, Vol. 1, No. 5, pp. 1-20, 2019.
[12] R.Y. Jang, R. Lee, M.W. Park, S.H. Lee, "Development of an AI Analysis Service System based on OpenFaaS," Journal of the Korea Contents Association, Vol. 20, No. 7, pp. 97-106, 2020 (in Korean).
[13] T.Y. Kim, J.R. Lee, T.H. Kim, I.G. Chun, J.M. Park, S.G. Jin, "Kubernetes Scheduler Framework Implementation with Realtime Resource Monitoring," IEMEK J. Embed. Sys. Appl., Vol. 15, No. 3, pp. 129-137, 2020 (in Korean).
[14] https://cloud.google.com/containers/?hl=ko
[15] I.S. Jeong, "Apache Kafka : From Application Development to Pipeline, Internet of Things data hub Construction," Hanbit Media, pp. 1-388, 2020 (in Korean).
[16] W.S. Ryu, "A System Design for Real-Time Monitoring of Patient Waiting Time based on Open-Source Platform," Journal of the Korea Institute of Information and Communication Engineering, Vol. 22, No. 4, pp. 575-580, 2018 (in Korean).
[17] https://kafka.apache.org/27/documentation.html#connect
[18] R.Y. Myung, H.C. Yu. S.K. Choi, "Performance Optimization Strategies for Fully Utilizing Apache Spark," KIPS Transactions on Computer and Communication Systems, Vol. 7, No. 1, pp. 9-18, 2018 (in Korean).
[19] S.Y. Ko, J.H. Won, "Processing Large-scale Data with Apache Spark," The Korean Journal of Applied Statistics, Vol. 29, No. 6, pp. 1077-1094, 2016 (in Korean).
[20] S.M. Baek, "Spark Two Programming for the Bigdata Analysis," Wikibooks, pp. 1-630, 2018 (in Korean).
[21] https://en.wikipedia.org/wiki/Docker_(software)
[22] https://docs.docker.com/engine/install/ubuntu/
[23] https://docs.docker.com/compose/
[24] https://en.wikipedia.org/wiki/YAML
[25] D.H. Han, Y.K. Lee, "Design of Action-Based Web Crawler Structural Configuration for Multi-Website Management," KIISE Transactions on Computing Practices, Vol. 27, No. 2, pp. 98-103, 2021 (in Korean).
[26] https://gitlab-gemscrc.gwnu.ac.kr/dgkim1108/bigdata-analysis-platform

Dong Gil Kim (김 동 길)

2011 Industrial System Engineering from Gangneung-Wonju University (B.S.)
2013 Industrial Engineering from Gangneung-Wonju University (M.S.)
2018 Industrial Engineering from Gangneung-Wonju University (Ph.D)
2018~Gang-Won Research Institute of ICT Convergence (Data Analysis Group, Senior Researcher)

Career:
2018 Gang-won Research Institute of ICT Convergence
Field of Interests: Machine Learning, Deep Learning, Bigdata Analysis Platform, Data Engineering
Email: dgkim1108@gwnu.ac.kr

Yong-Soon Park (박 용 순)

2003 Electrical Engineering from Gangneung National University (M.S.)
2014 Electrical Engineering from angneung-Wonju National University (Ph.D. Candidate)
2003~2005 HAMPEX (Assistant Engineer)
2005~Gang-Won Research Institute of ICT Convergence (Group Leader)

Career:
2005 Gang-won Research Institute of ICT Convergence
Field of Interest : Bigdata, AI, ML
Email: ee9415@gmail.com

Tae-Yun Chung (정 태 윤)

1987 Electrical Engineering from Yonsei University (B.S.)
1989 Electrical Engineering from Yonsei University (M.S.)
2000 Electrical & Computer Engineering from Yonsei University (Ph.D)
2001~Electronics Engineering at Gang-neung-Wonju National University (Prof.)

Career:
1989~1996 Samsung Advanced Institute of Technology
1996~2001 Samsung Electronics
2004~Gangwon Research Institute of ICT Convergence (President)
Field of Interest : Video Processing, IoT, Big data, AI
Email: tychung@gwnu.ac.kr