

CSE545

빅데이터처리 및 실습

(Big Data Processing and Practice)

Theory 02:

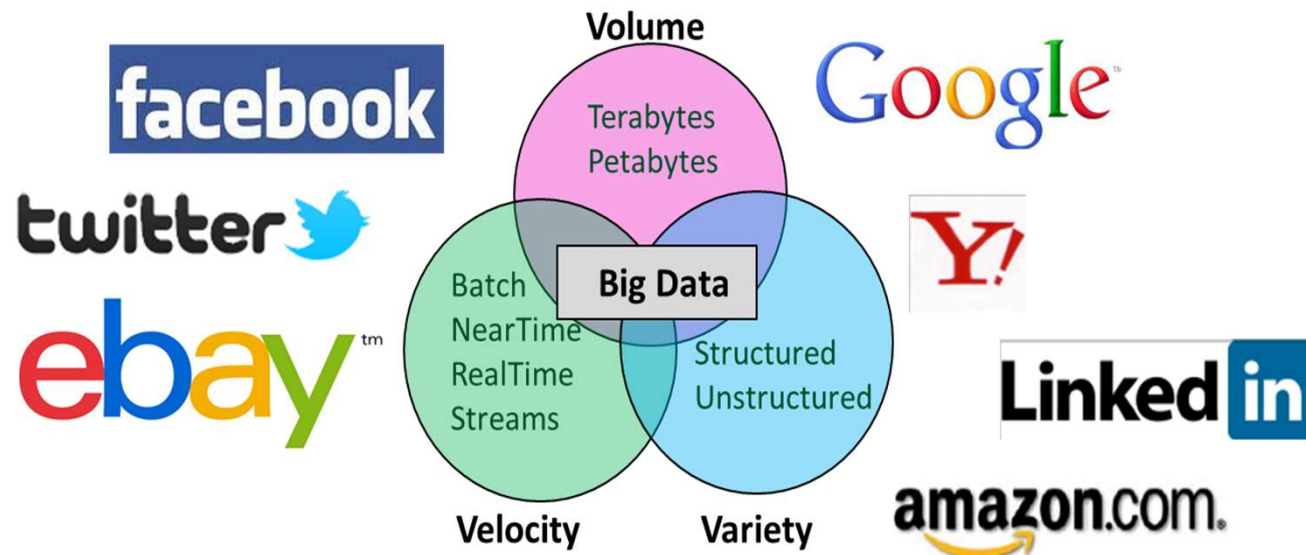
Overview of Big Data

담당교수: 전강욱(컴퓨터공학부)
kw.chon@koreatech.ac.kr

빅데이터의 개념과 처리 과정

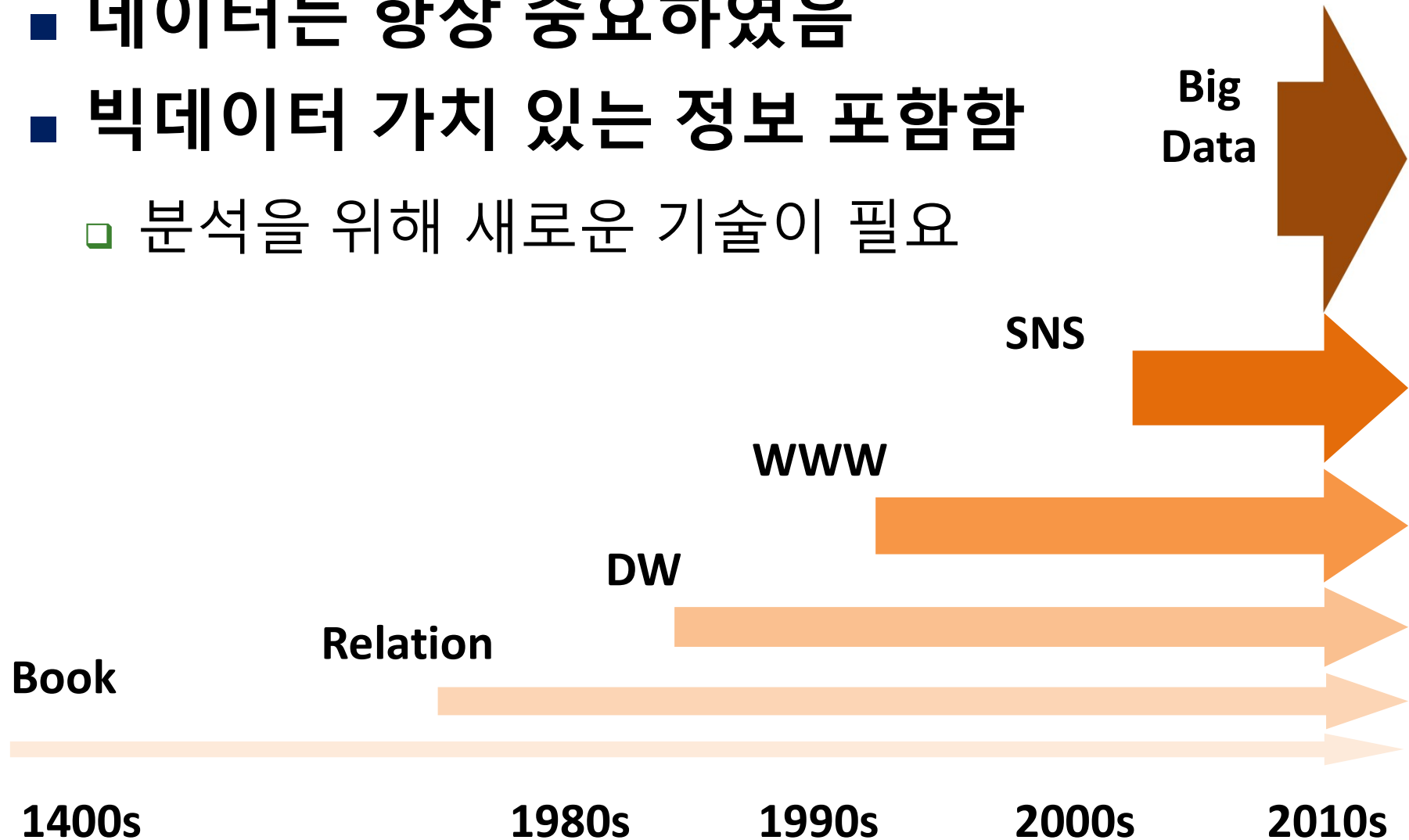
■ 빅데이터 등장 배경

- 1990년 이후 인터넷이 확산되면서 정형화된 데이터와 비정형화된 데이터가 무수히 발생하면서 정보 홍수개념이 등장
- 개인화 서비스와 SNS의 확산으로 기본 인터넷 서비스 환경 재구성
- 전 세계 디지털 데이터 양이 제타 바이트 단위로 2년마다 2배씩 증가하여 2020년에는 약 40제타 바이트가 될 것이라고 함
 - IDC Digital University의 보고서



왜 빅데이터가 중요한가?

- 데이터는 항상 중요하였음
- 빅데이터 가치 있는 정보 포함함
 - 분석을 위해 새로운 기술이 필요






빅데이터의 개념과 처리 과정

- 데이터양 증가로 기존의 데이터 저장, 관리, 분석기법으로 데이터 처리에 한계가 있어 정보 기술의 패러다임도 함께 변함
- 빅데이터 용어를 등장, 패러다임 지능화, 개인화된 시대를 빅데이터 시대라 함

	PC시대	인터넷시대	모바일시대	스마트시대
패러다임변화	디지털화, 전산화	온라인화, 정보화	소셜화, 모바일화	지능화, 개인화, 사물정보화
정보기술이슈	PC	초고속인터넷, WWW	모바일 인터넷, 스마트폰	빅데이터, 차세대PC, 사물네트워크
핵심분야	PC, OS	포털, 검색엔진	스마트폰, 웹서비스, SNS	개인화서비스, 상황인식
대표기업	MS, IBM	구글, 네이버	애플, 페이스북, 트위터	구글, 삼성, 애플, 페이스북
정보기술비전	1인 1PC	클릭 e-Korea	손안의 PC	IT everywhere

빅데이터 사례

- 최근에는 데이터가 중요하지 않다고 여겨지는 곳까지 **빅데이터 분석**을 시도
 - 데이터 플랫폼 구축, AI 적용 등

기업	핵심 데이터	매일 발생하는 데이터 양
	<ul style="list-style-type: none">방문자의 검색어와 클릭한 광고나 링크음식점 리뷰, 여행 정보, 지도 데이터, 교통 정보 등 일상 생활과 밀접한 각종 정보안드로이드 디바이스를 통한 사용자 정보	<ul style="list-style-type: none">6억 2,000만명의 방문자10억 건의 검색72억건의 페이지뷰
	<ul style="list-style-type: none">1억 2,000만명의 고객정보고객의 검색어와 상품 탐색 및 구매내역230만 종의 서적 데이터 베이스	<ul style="list-style-type: none">440만명의 방문자900만개의 상품 주문(2010년 크리스마스)
	<ul style="list-style-type: none">20억명의 회원, 1,000억 건의 친구관계회원의 관심사, 소속, 결혼여부, 심리 상태 등의 소셜 데이터 보유	<ul style="list-style-type: none">2억 5,000만 장의 사진27억 건의 '좋아요'와 댓글

빅데이터의 개념과 처리 과정

■ 빅데이터가 차세대 이슈로 떠오르는 이유

- 정보 통신 기술의 주도권이 데이터로 이동
- 공간, 시간, 관계, 세상 등을 담은 빅데이터
- 빅데이터는 미래 경쟁력과 가치 창출의 원천

구분	정보화시대(1세대)	스마트시대(2세대)
저장	관계형(정형) 데이터베이스, 데이터웨어하우스	비정형 데이터베이스, 가 상화, 클라우드 서비스
관리	지식관리시스템, 웹2.0	플랫폼, 소셜 네트워크, 집단지성
분석	경영정보, 고객정보, 자산정보분석(ERP, CRM, 데이터마이닝 등)	빅데이터 분석 (소셜 분석, 시각화)

정보화 시대와 스마트 시대의 데이터 처리 변화.

빅데이터의 개념과 처리 과정

■ 빅데이터 속성

- 규모(Volume): 데이터의 크기
- 다양성(Variety): 다양한 종류의 데이터를 수용하는 속성
- 속도(Velocity): 데이터를 빠르게 처리하고 분석할 수 있는 속성
- 정확성(Veracity): 데이터에 부여할 수 있는 신뢰 수준
- 가치(Value): 빅데이터를 저장하려고 IT 인프라 구조 시스템을 구현하는 비용

빅데이터의 개념과 처리 과정

- 이전에 빅데이터는 3V를 강조하였지만, 최근에 정확성 및 가치까지 강조하고 있음

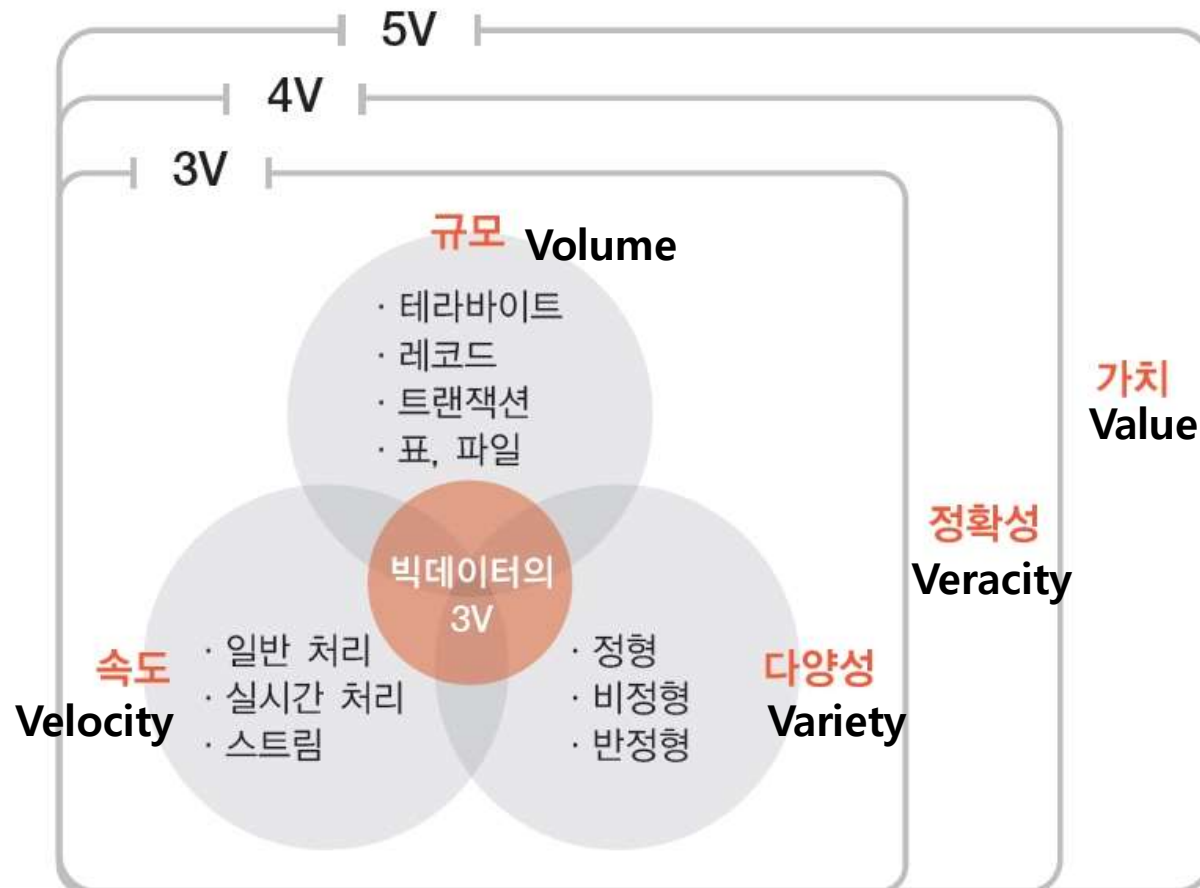


그림 1-2 빅데이터의 속성 [02]

빅데이터의 개념과 처리 과정

■ 빅데이터 정의

- 정형 데이터 + 비정형 데이터 + 데이터 관리/분석 인력 조직 + 데이터 관리/분석 기술 등 기존의 기술/인력/조직으로 다룰 수 없는 데이터 환경



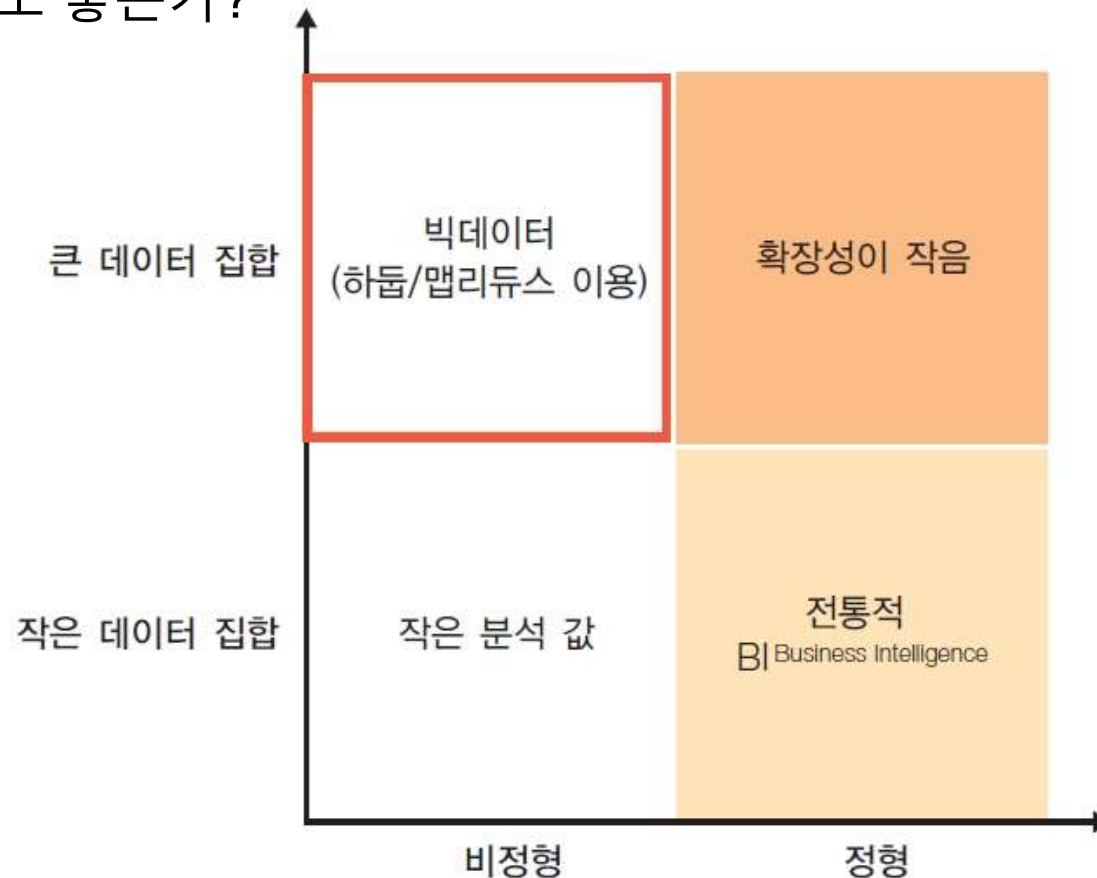
광의의 빅데이터 정의.

* 광의: 넓은 뜻을 의미함.

빅데이터의 개념과 처리 과정

■ 빅데이터 위치

- 데이터의 규모가 작은 경우 빅데이터 프레임워크를 사용하여 데이터 처리를 할 때도 좋은가?



규모와 다양성에 따른 빅데이터의 위치.

빅데이터의 개념과 처리 과정

■ 빅데이터 종류와 유형 변화

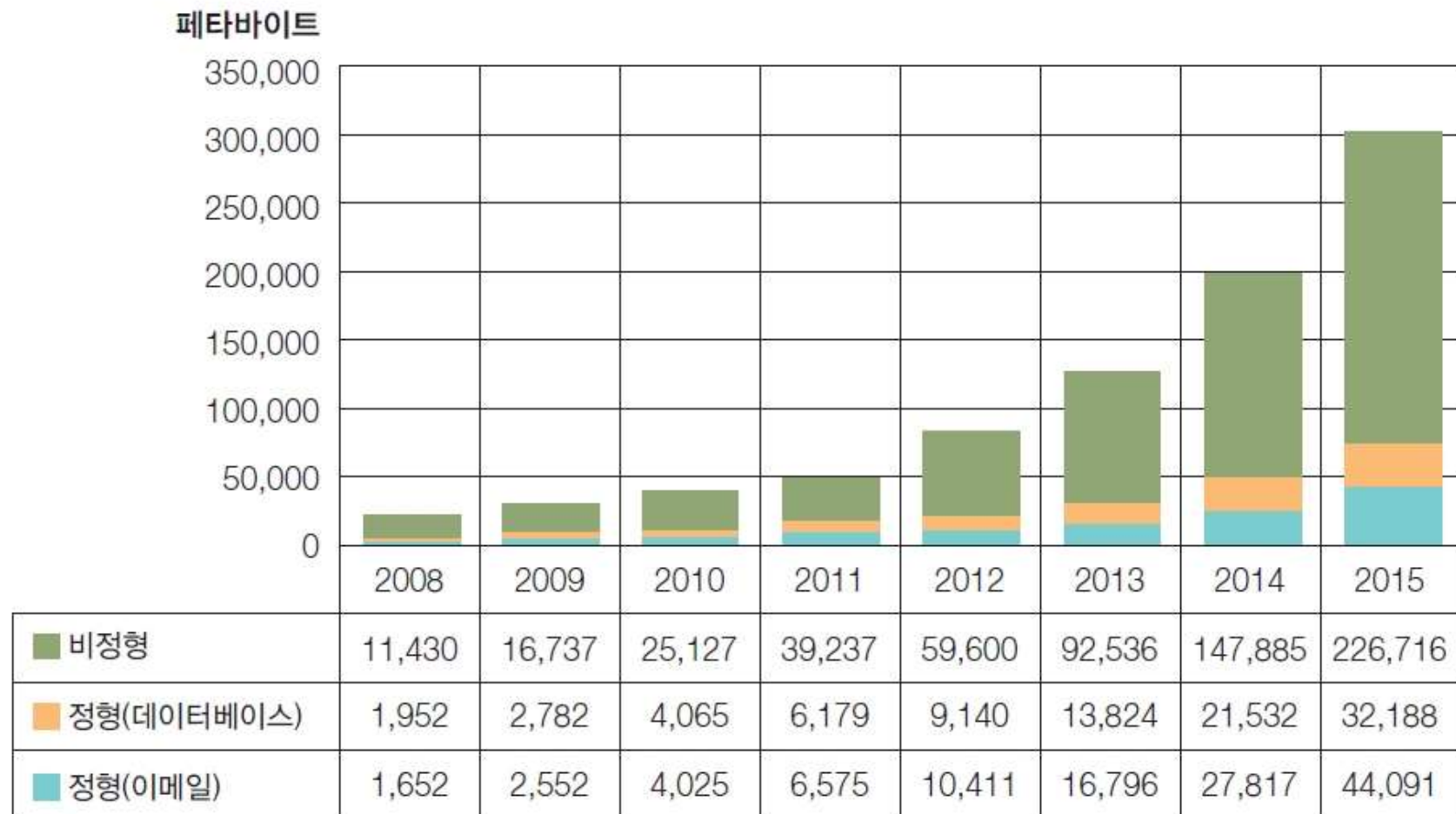
- 정형/반정형/비정형 등 다양한 타입의 데이터들을 다룸

종류	설명
정형	고정된 필드에 저장된 데이터 e.g., 관계형 데이터베이스, 스프레드시트
반정형	고정된 필드에 저장되어 있지 않지만, 메타데이터나 스키마 등을 포함하는 데이터 e.g., XML, HTML, 텍스트
비정형	고정된 필드에 저장되어 있지 않은 데이터 e.g., 텍스트 분석이 가능한 텍스트 문서, 동영상, 음성 데이터

빅데이터 종류.

빅데이터의 개념과 처리 과정

- 이미지, 동영상, SNS 데이터 등 다른 데이터 타입에 비하여 비정형 데이터가 많이 생기고 있음



정형과 비정형 데이터 유형의 변화.

빅데이터의 개념과 처리 과정

■ 전통적 데이터와 빅데이터 특징 비교

구분	전통적 데이터	빅데이터
데이터 원천	전통적 정보 서비스	일상화된 정보 서비스
목적	업무와 효율성	사회적 소통, 자기표현, 사회 기반 서비스
생성 주체	정부 및 기업 등 조직	개인 및 시스템
데이터 유형	<ul style="list-style-type: none"> 정형 데이터 조직 내부 데이터(고객 정보, 거래 정보 등) 주로 비공개 데이터 	<ul style="list-style-type: none"> 비정형 데이터(비디오 스트림, 이미지, 오디오, 소셜 네트워크 등 사용자 데이터, 센서 데이터, 응용 프로그램 데이터 등) 조직 외부 데이터 일부 공개 데이터
데이터 특징	<ul style="list-style-type: none"> 데이터 증가량 관리 가능 신뢰성 높은 핵심 데이터 	<ul style="list-style-type: none"> 기하급수로 양적 증가 <u>쓰레기(Garbage) 데이터 비중 높음</u> 문맥 정보 등 다양한 데이터
데이터 보유	정부, 기업 등 대부분 조직	<ul style="list-style-type: none"> 인터넷 서비스 기업(구글, 아마존 등) 포털(네이버, 다음 등) 이동 통신 회사(SKT, KTF 등) 디바이스 생산 회사(애플, 삼성전자 등)
데이터 플랫폼	정형 데이터를 생산·저장·분석·처리할 수 있는 전통적 플랫폼 예) 분산 DBMS, 다중처리기, 중앙 집중 처리	비정형 대량 데이터를 생산·저장·분석·처리할 수 있는 새로운 플랫폼 예) 대용량 비정형 데이터 분산 병렬 처리

데이터 분석 기술이 중요
(Statistics, Data Mining,
Machine Learning 등)

빅데이터의 개념과 처리 과정

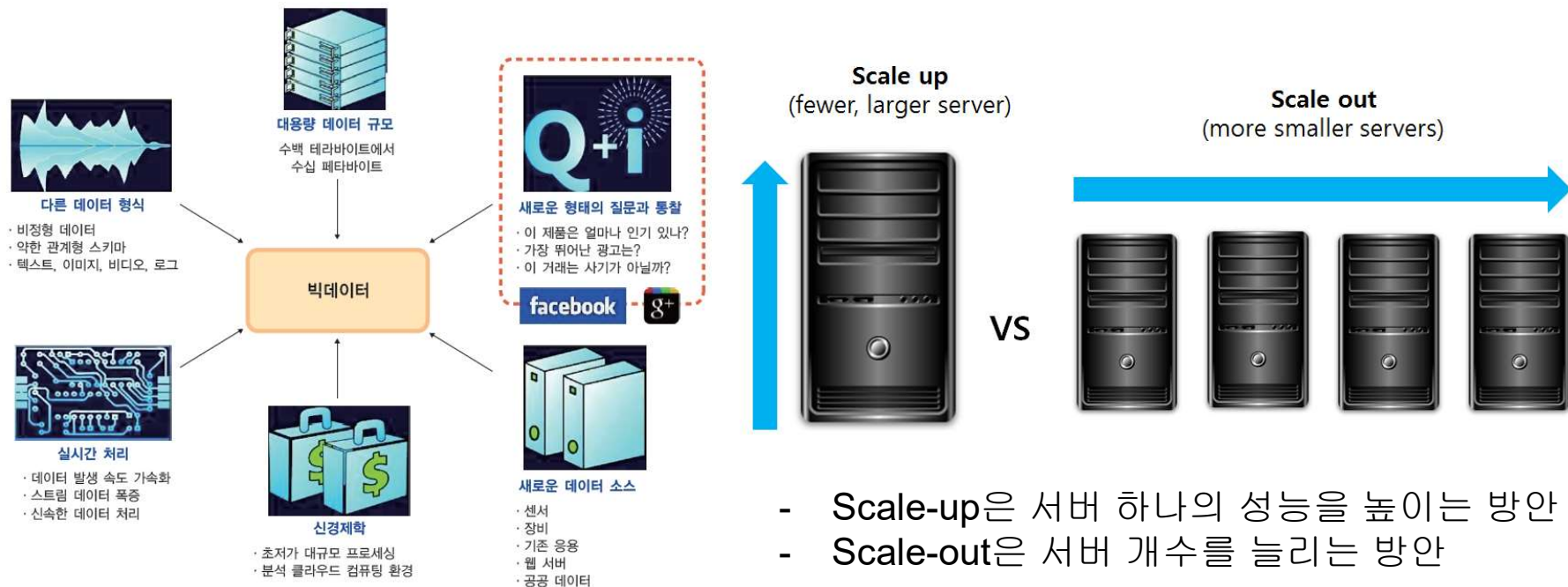
■ 빅데이터 처리 특징

- 처리 복잡도: 다양한 데이터 소스, 복잡한 로직 처리, 대용량 데이터 처리로 복잡도가 높아 분산 처리 기술 (또는 고성능 병렬처리기술) 필요함
 - 일반적으로 필요한 계산량은 데이터 양에 비례함
- 데이터 규모: 처리되어야 할 데이터 규모가 방대함
- 데이터 구조: 비정형 데이터의 비중이 높음
 - 예: 소셜 미디어 (Facebook), 콜센터 로그 등
- 분석 유연성: 잘 정의된 데이터 모델, 상관관계 등이 없어 기존 데이터 처리 방법에 비해 처리 및 분석 유연성이 높음
- 처리량: 동시 처리량이 높음(초당 몇GB의 데이터를 처리하는가?)

빅데이터의 개념과 처리 과정

■ 빅데이터 속성과 처리 특징

- 데이터가 생성되는 장비 등이 다양화되며, 데이터양이 커짐에 따라서 기존 방법으로는 적정 가격으로 저장 및 분석할 수 가 없음
 - 높은 라이선스 비용 → 오픈소스 기반의 SW 활용
 - 기존 분석 방법은 다양한 에러 발생 (O.O.M. 등) → Scale-out vs. Scale-up



빅데이터의 개념과 처리 과정

■ 빅데이터 처리 과정과 기술

이번 강의에서는 아래 내용에 집중



빅데이터 처리 과정.

빅데이터의 개념과 처리 과정

과정	영역	개요
생성	내부데이터	데이터베이스, 파일관리 시스템
	외부데이터	인터넷으로 연결된 파일, 스트림, 멀티미디어
수집	크롤링	검색 엔진의 로봇을 사용한 데이터 수집
	ETL	소스 데이터의 추출, 전송, 변환, 적재
저장	NoSQL	비정형 데이터 관리
	스토리지	빅데이터 저장
	서버	초경량 서버
처리	맵리듀스	데이터추출
	프로세싱	다중업무처리
분석	NLP	자연어 처리
	기계학습	기계학습으로 데이터의 패턴 발견
표현	가시화	데이터를 도표 등으로 표현

빅데이터 처리 과정별 기술 영역.

빅데이터의 개념과 처리 과정

■ 빅데이터 소스 생성과 수집 기술

- 내부 데이터 수집 : 자체적으로 보유한 내부 파일 시스템, 데이터베이스 관리 시스템, 센서 등에서 정형 데이터를 수집
- 외부 데이터 수집 : 인터넷으로 연결된 외부에서 비정형 데이터를 수집

방법	설명
로그 수집기	내부에 있는 웹 서버의 로그를 수집
크롤링	주로 웹 로봇으로 거미줄처럼 얹혀있는 인터넷 링크를 따라다니며 방문한 웹사이트의 데이터를 수집
센싱	각종 센서로 데이터 수집
오픈 API	데이터의 생성, 공유, 참여 환경인 웹2.0을 구현하는 기술로 필요한 데이터를 프로그래밍으로 수집

데이터 수집 방법.

빅데이터의 개념과 처리 과정

■ 빅데이터 저장 기술

접근 방식	설명	제품
분산파일 시스템	컴퓨터 네트워크로 공유하는 여러 호스트 컴퓨터 파일에 접근할 수 있는 파일 시스템	GFS, HDFS, 아마존 S3
NoSQL	데이터 모델을 단순화해서 관계형 데이터 모델과 SQL을 사용하지 않는 모든 DBMS 또는 데이터 저장 장치	Cloudata, HBase, Cassandra
병렬 DBMS	다수의 프로세서를 사용하여 여러 디스크의 질의, 갱신, 입출력 등 데이터 베이스 처리를 동시에 수행하는 데이터베이스 시스템	VoltDB, SAP HANA, Vertica
네트워크 구성 저장 시스템	서로 다른 종류의 데이터 저장 장치를 하나의 데이터 서버에 연결하여 총괄적으로 데이터를 저장 및 관리	SAN, NAS

빅데이터의 개념과 처리 과정

■ 빅데이터 처리 기술

- 맵리듀스: 분산 병렬 데이터 처리 기술의 사실상의 표준
 - 일반 범용 서버(Commodity Machines)로 구성된 Cluster 시스템을 기반
 - Key-Value 쌍들을 입력으로 받아, 데이터분할 처리 및 처리 결과 통합 기술, job 스케줄링 기술, 작업 분배 기술, 태스크 재수행 기술이 통합된 분산 컴퓨팅 기술
 - 복잡한 기술이 사용자에게 숨겨짐
- 하둡: 맵리듀스 및 구글 파일시스템의 오픈소스 버전
 - 정형·비정형 빅데이터 분석에 가장 선호되는 솔루션
 - 하둡은 프로세싱 모델(MapReduce)과 분산파일시스템(Hadoop Distributed File System)으로 구성됨
- R: R 언어와 개발 환경으로 기본적인 통계 기법부터 모델링, 최신 데이터 마이닝 기법까지 구현 및 개선이 가능
- NoSQL (Not-only SQL): 전통적인 관계형 데이터베이스 RDBMS와는 다르게 설계된 비관계형 데이터베이스

빅데이터의 개념과 처리 과정

■ 일괄 처리(Batch Processing)

- Latency 보다 Throughput에 집중함
- e.g., Apache Hadoop, Apache Spark 등

■ 실시간 처리(Realtime Processing)

- Latency에 집중함
- e.g., InfoSphere Streams, Apache Storm 외에 Apache Kafka 등

■ 프로그래밍 지원 기술

- 활용성(Easy-to-Use)에 집중함 (적은 개발 공수로 데이터 처리하는데 집중)
- e.g., Apache Pig, Apache Hive

빅데이터의 개념과 처리 과정

- 인프라 기술을 포함한 빅데이터와 연계된 기술들
 - Cassandra
 - 분산 시스템에서 대용량 데이터를 처리할 수 있도록 설계된 오픈소스 기반 빅데이터 처리 시스템임
 - 원래 Facebook에서 개발하였으며, 현재 Apache 재단에서 프로젝트로 관리함
 - Hadoop
 - Google이 개발한 MapReduce를 오픈소스로 구현한 결과물
 - Yahoo!에서 개발하였으며, 현재 Apache 재단에서 프로젝트로 관리함
 - Hbase
 - Google의 BigTable을 참고로 개발된 오픈소스 분산 비관계형 데이터베이스
 - Powerset에서 개발하였으며, 현재 Apache 재단에서 프로젝트로 관리함
 - NoSQL
 - 전통적인 RDMBS와 다르게 설계된 비관계형 데이터베이스
 - e.g., Cassandra, Hbase, MongoDB 등

빅데이터의 개념과 처리 과정

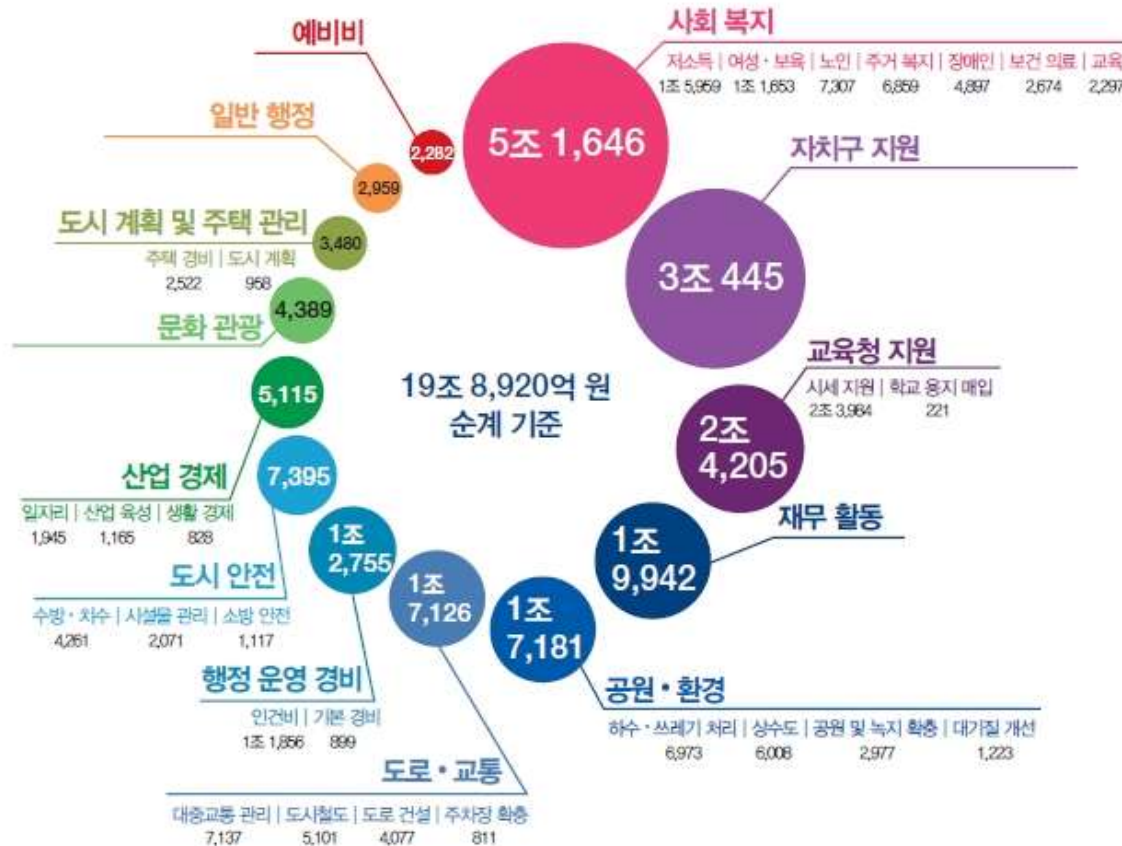
■ 빅데이터 분석 기술

- 통계분석: 모집단을 대표할 수 있는 표본 집단에서 정보를 수집하고 가설, 검정, 추론 과정을 거쳐서 분석하는 기술
 - 표본집단을 얼마나 잘 추출하는지가 중요
 - 표본론 + 추정 + 가설검정 + 다변량 분석

- 데이터마이닝: 대량의 데이터를 분석하여 숨겨진 패턴이나 규칙을 찾아내고, 유의미한 정보를 현실에 응용 반영하는 기술
 - 통계적 분석기법 + 분산 병렬 + DB + 기계학습 + 알고리즘 등 전반적인 것을 포함
 - 크게 4가지 종류: Association Rule Mining, Classification, Clustering, Outlier Detection
 - 머신러닝과의 차이는?

빅데이터의 개념과 처리 과정

■ 빅데이터 표현 기술



단위 %	점유비	증감률	단위 %
26.0	사회 복지	13.3	
15.3	자치구 지원	4.8	
12.2	교육청 지원	1.6	
10.0	재무 활동	-1.3	
8.6	공원·환경	-3.3	
8.6	도로·교통	-2.9	
6.4	행정 운영 경비	7.2	
3.7	도시 안전	44.3	
2.6	산업 경제	0.5	
2.2	문화 관광	6.7	
1.7	도시 계획 및 주택 관리	-15.0	
1.5	일반 행정	13.4	
1.2	예비비	-22.5	

순계 : 총예산에서 회계 간 중복분 제외한 실질적인 예산

그림 1-8 정보 표현의 간단한 예 [12]

빅데이터의 개념과 처리 과정

■ 빅데이터 활용 분야와 기대 효과

도메인	분석 대상 데이터	예상 효과
미국의 의료 사업	제약사 연구 개발 데이터, 환자 치료, 임상 데이터, 의료 산업의 비용 데이터	연간 \$3조, 0.7% 생산성 향상
유럽의 공공 행정	정부의 행정 업무에서 발생하는 데이터	연간 \$4.1조, 0.5% 생산성 향상
소매업	고객의 거래 데이터, 구매 경향	\$1조 + 서비스 업자 수익 \$7조 소비자 이익
제조업	고객 취향 데이터, 수요 예측 데이터, 제조 과정 데이터	60% 마진 증가 0.5~1.0% 생산성 증가
개인 위치 데이터	개인과 차량의 위치 데이터	개발 및 조립 비용 50% 감소 운전자본 7%감소

맥킨지에서 제시한 빅 데이터 활용 분야.

빅데이터의 개념과 처리 과정

■ 신한카드 사례

□ 추진 목적

- 카드 이용 패턴 분석을 통한 맞춤형 마케팅 전략 수립
- 고객 맞춤형 소비 생활 서비스 및 상품 제공으로 매출 증가 기대
- 고객의 성향을 분석하여 영업 전략 도출

□ 추진 내용

■ 트렌드 분석

- 고객의 다양한 요구를 명확하게 파악 후 상품에 반영
- 카드 결제 내역을 분석하여 향후 고객 유치 과정에 알맞은 카드 서비스를 제공할 수 있는 전략 수립

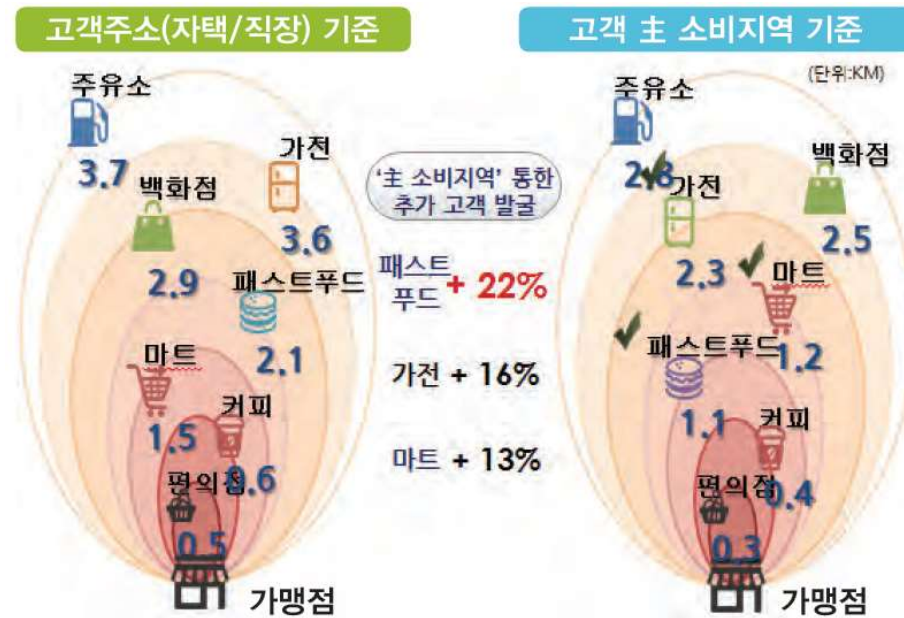
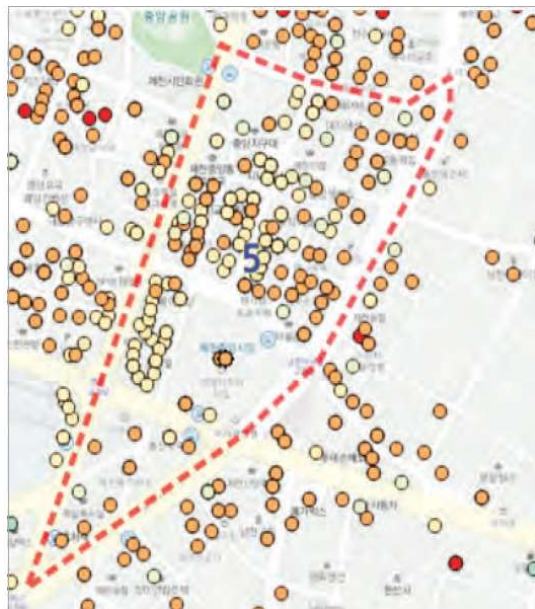


빅데이터의 개념과 처리 과정

■ 신한카드 사례

□ 카드 사용처 분석

- 특정 지역의 업종을 음식, 관광/레저, 편의점/쇼핑으로 구분하여 소비자가 결제한 상점에 대해 분석한 것으로 고객의 소비 성향과 카드 사용 집중 지역을 다룬 정리된 데이터 확보 가능
- 가맹점에서 파악하기 어려운 고객의 주 소비지역 거리 데이터를 통해 정확도나 이용도가 높은 타깃 발굴



빅데이터의 개념과 처리 과정

■ 활용 분야

- 공공분야- 국가적 차원에서 방대한 양의 데이터로 수자원 관리, 스마트 그리드, 재난 방재 영역 등을 포괄적으로 포함.
- 과학분야 – 산발적으로 흩어진 과학 데이터를 국가 차원에서 수집, 가공, 유통, 재활용할 수 있는 기반을 마련
- 의료 분야 - 의료 기록의 전자화, 병원 간 연구 데이터 공유로 빅데이터 도입과 활용이 확대됨
- 도소매 분야 – 이미 데이터를 활용 중이며 빅데이터 분석으로 수요 예측 및 선제적 경영 지원에 초점을 둠
- 제조분야 – 보유 데이터양 이 많고, 불량품 개선 비용 등 적용 효과를 계량화하여 빅데이터의 유용성을 확인할 수 있는 분야
- 정보 통신 분야 – 이동통신의 발전과 개인 단말기의 증가로 생성된 디지털 공간의 개인 데이터로 목표 마케팅, 개인화 서비스 확대

빅데이터의 개념과 처리 과정

■ 기대 효과

- 이상현상 감지- 업무에서 발생하는 이벤트를 기록하여 '정상 상태', '비정상 상태'를 표시하는 패턴을 파악하고, 이 패턴을 기초로 새로운 이벤트가 발생할 경우 이상 현상 여부를 판단함, 마케팅 분야에도 활용
- 고객 이탈을 사전에 감지한 T-Mobile
- 위키리크스 데이터 분석으로 효과적인 전술 정보 제공
- 아마존닷컴의 추천 상품 표시, 구글 및 페이스북의 맞춤형 광고

CSE545

빅데이터처리 및 실습

**(Big Data Processing and
Practice)**

**Theory 03: S/W Platform for
Managing Big Data**

담당교수: 전강욱(컴퓨터공학부)

kw.chon@koreatech.ac.kr

빅데이터의 수집 및 통합 기술

■ 개요

- 의미 있는 분석이 가능한 데이터는 보통 ‘양’에 의존하는데 인터넷과 컴퓨터의 발전으로 많은 양의 데이터(정형, 비정형)가 수집가능해짐
- 비정형 데이터의 처리에 어려움이 생김
- Hadoop, Spark 등 방대한 양의 데이터를 효과적으로 수집

빅데이터의 수집 및 통합 기술

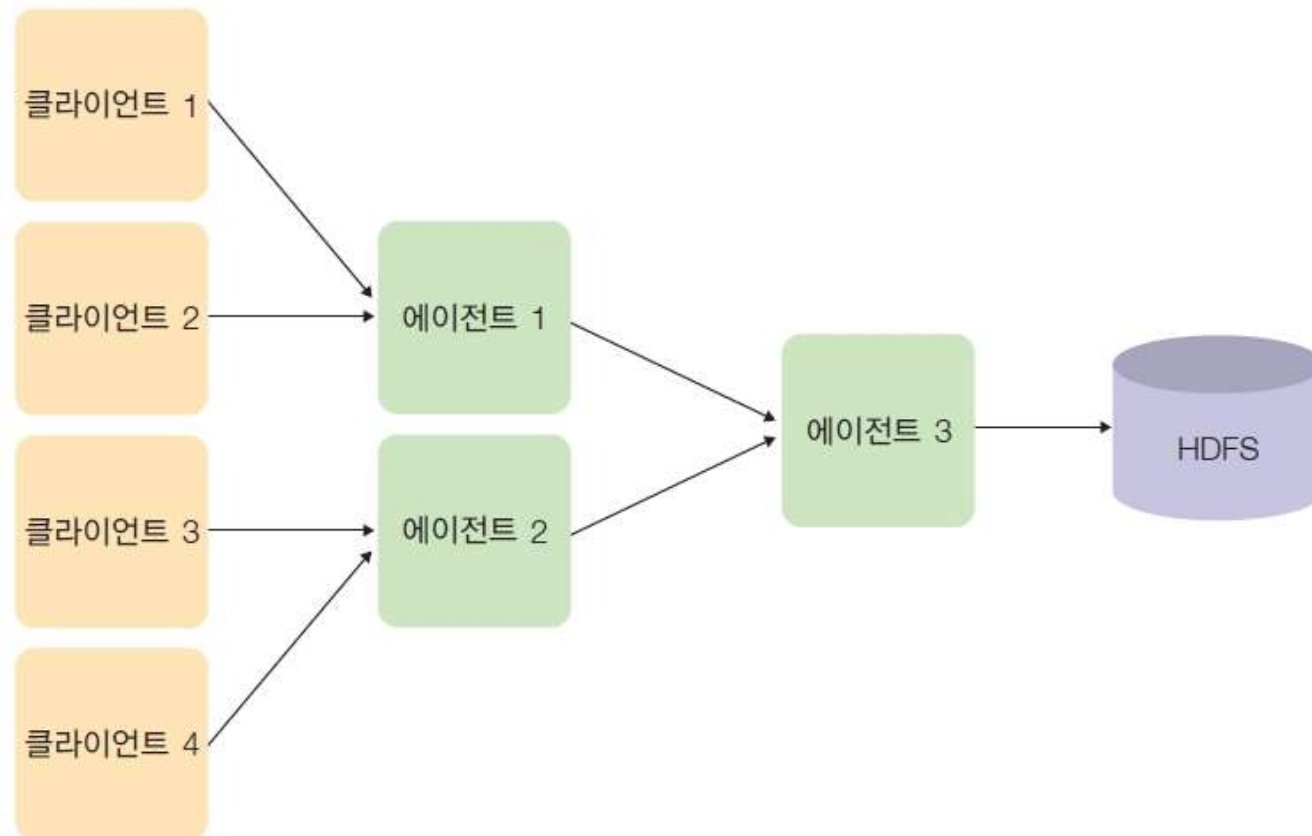
■ 빅데이터 주요 수집 및 통합 기술

제품/기술	최초 개발	최초공개	주요 기능 및 특징
Flume	Cloudera	2010년	이벤트 로그 수집
Chukwa	Yahoo	2008년	로그 수집 및 모니터링
Scribe	Facebook	2008년	로그 수집 서버
SQOOP	Apache	2009년	RDBMS와 NoSQL간 데이터 연동
Kafka	LinkedIn	2010년	메시지 전송 및 수집
OpenRefine	Google	2010년	대용량 데이터 정제

빅데이터 주요 수집 및 통합 기술.

빅데이터의 수집 및 통합 기술

- Flume: 이벤트 로그 데이터를 효율적으로 수집하고 집계할 수 있는 로그 수집기, 안정성과 가용성이 높음

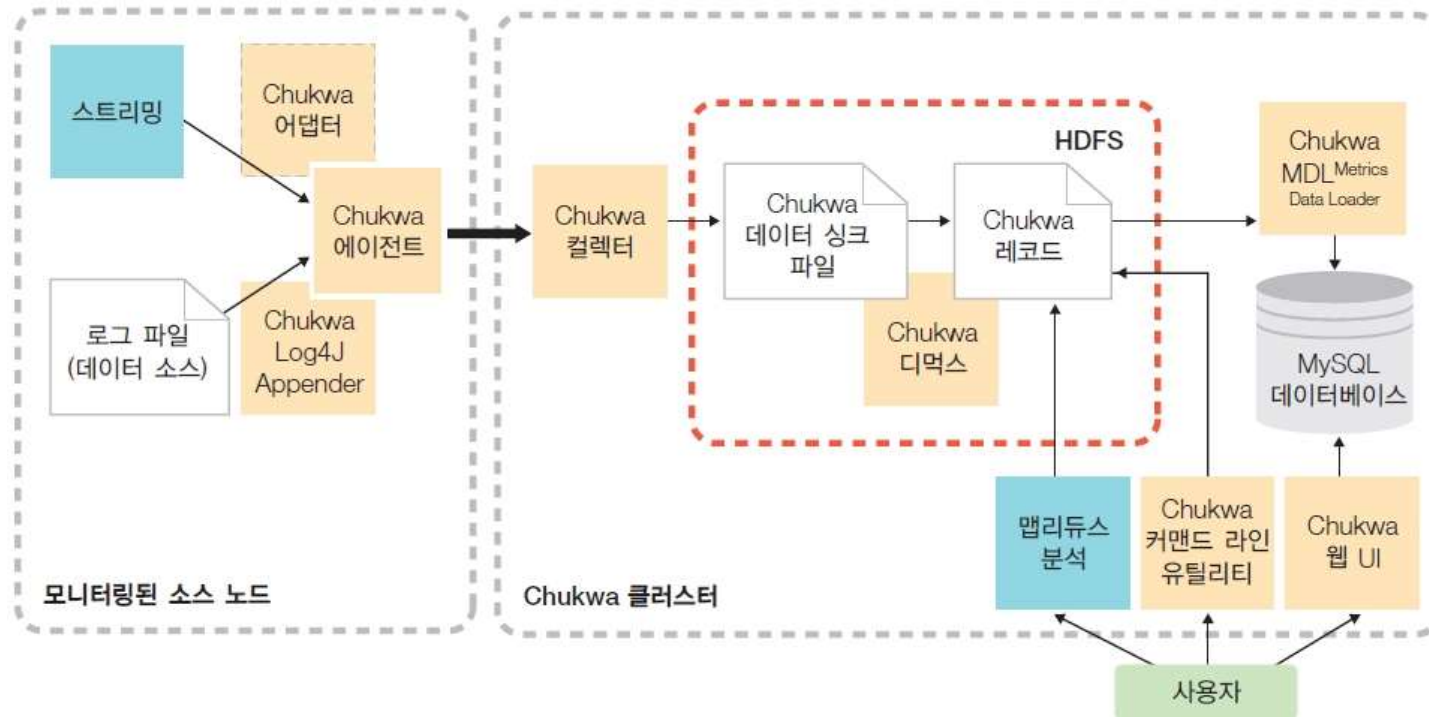


Flume의 데이터 흐름도.

빅데이터의 수집 및 통합 기술

■ Chukwa - 로그 데이터 수집 및 분석, 출력, 모니터링 시스템

- 하둡 기반으로 동작함 (반드시 하둡 설치)
- 응용 프로그램의 로그 저장 모듈을 수정하지 않아도 로그 수집 가능
- 하둡 분산 파일 시스템을 그대로 수용하고 실시간 분석이 가능한 장점



Chukwa 시스템 구성도.

빅데이터의 수집 및 통합 기술

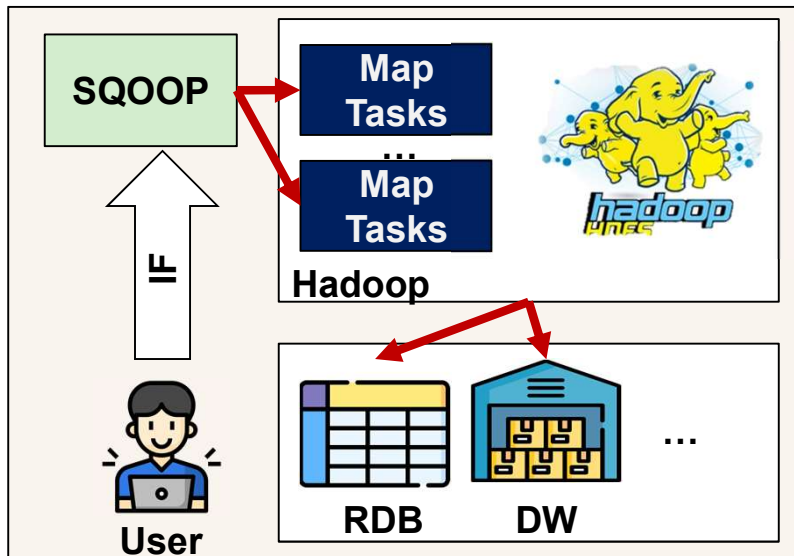
■ Scribe

- 분산형 로그 수집 서버, 오픈 소스 프로젝트
- 하루에 메시지 수십억 개를 저장가능, 중앙 서버 그룹과 노드 당 Scribe 서버 한대로 구성됨
- 기존 클라이언트를 수정하지 않고도 로그 데이터를 수집할 수 있는 크로스 플랫폼 클라이언트 라이브러리를 제공

빅데이터의 수집 및 통합 기술

■ SQOOP (SQL-to-hadOOP)

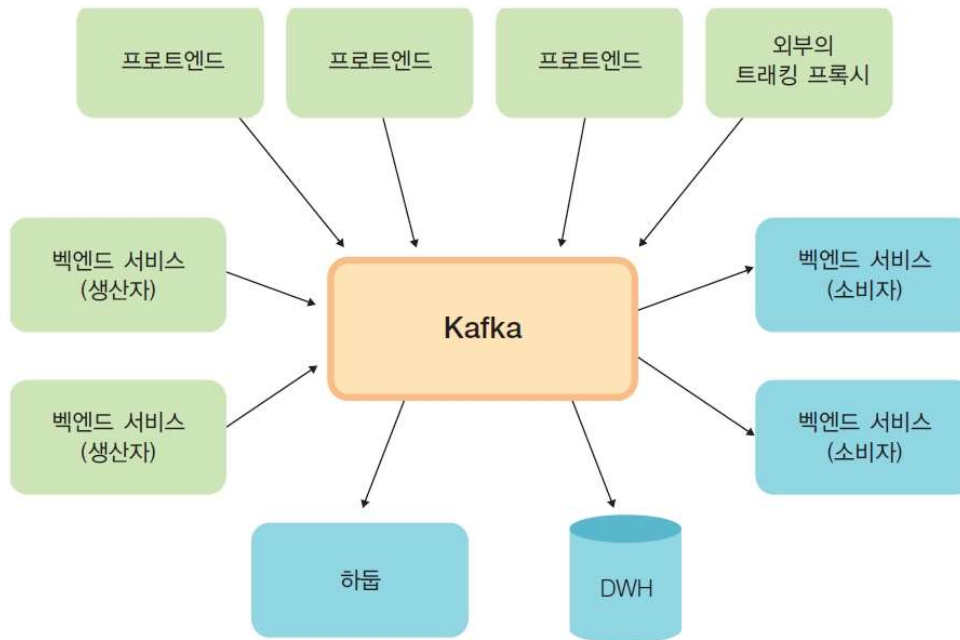
- 기존 RDBMS 에서 하둡으로 데이터를 이전하려고 시작한 프로젝트
- JDBC인터페이스를 사용하므로 MySQL, PostgreSQL, 오라클 등 다양한 데이터베이스 시스템을 지원함
- 아파치 하둡 기반 프로젝트인 Hive, Pig, Hbase 등과도 호환이 잘되어 RDMS와 NoSQL 간의 데이터 연동에 많이 사용됨



빅데이터의 수집 및 통합 기술

■ Kafka

- LinkedIn에서 개발 시작한 Kafka는 확장성이 좋고 처리량이 높은 분산 메시지 시스템으로 지속적이고 우수한 메시지 전달 성능을 보장함
- 실시간 데이터 처리에 널리 사용됨
 - Kafka + Spark Streaming, Kafka + Storm 등의 조합으로 플랫폼 구성



Kafka 시스템 구성도.

Realtime Data Pipelines 관련 SW Stack 구성 예시

Apache Hadoop, HBase, Phoenix

- Data Store for Realtime Data Ingestion

Kafka Ecosystem

- Apache Kafka Server
- Streams
- Kafka Rest Proxy
- Apache Avro
- Kafka Schema Registry
- Y! Kafka Manager

Distributed Computing

- Apache Spark (streaming)

빅데이터의 수집 및 통합 기술

■ JSON (JavaScript Object Notation)

- 인터넷에서 데이터를 주고받을 때 그 데이터를 표현하는 방식
- 기존 XML과 비슷하지만 용량이 작고 변환 속도가 빠름
- JSON은 데이터의 종류에 제한이 없어 웹의 데이터를 정형화 하는데 사용됨
- JavaScript 의 구문 형식을 따르지만 프로그래밍 언어나 플랫폼에 독립적임
- 텍스트로 되어 사람과 기계 모두 읽고 쓰기 쉬우며, 프로그래밍 언어와 플랫폼에 독립적이기에 서로 다른 시스템 간에 쉽게 객체교환 가능

```
"employees":[  
  {"firstName":"John", "lastName":"Doe"},  
  {"firstName":"Anna", "lastName":"Smith"},  
  {"firstName":"Peter","lastName":"Jones"}  
]
```

JSON Array 예제: 여러 개의 **Object**들을 포함하는 것이 가능함.

빅데이터의 수집 및 통합 기술

■ Thrift

- 확장성 높은 이기종 언어 서비스를 지원하는 소프트웨어 프레임워크
- 복잡한 비정형 데이터 객체의 정형화 기능, 정형화된 데이터를 취급할 수 있는 인터페이스 기능, 응용 계층과 무관한 하부 데이터 형식 지원 기능이 있음
- 단점 - 라이브러리 API의 업데이트가 잦고, XML의 직렬화 및 역 직렬화 기능을 제공하지 않으며, 코드가 복잡함
- 특징 - 버전닝 지원, 여러 언어 지원 및 언어별 소스 생성, 언어 간 직렬화 기능, 동기·비동기 API 제공, XML 설정 불필요, RPC 지원 등

빅데이터의 수집 및 통합 기술

■ Protocol Buffers

- 오픈 소스 직렬화 라이브러리
- 구글 내에 있는 많은 시스템과 다양한 운영체제 및 언어 환경에서 서로 통신할 수 있도록 직렬화 방법으로 개발한 라이브러리를 오픈 소스화 한 것
- XML에 비해 속도가 빠르고 데이터 크기가 작고, 코드가 간단하며, 개발이 쉬움
- 메시지를 연속된 비트로 생성, 이와 반대로 비트에서 원래의 메시지로 만들 수 있으며, 패킷을 전송하는 데 유용함

먼저 .proto 파일을 생성한 후 다음과 같이 작성한다.

```
message Person {  
    required int32 id = 1;  
    required string name = 2;  
    optional string email = 3;  
}
```

작성한 파일을 프로토콜 버퍼 컴파일러로 사용할 언어에 맞게 컴파일한다.
그러면 다음과 같이 호출하여 사용할 수 있다(C++로 가정).

```
Person person;  
person.set_id(123);  
person.set_name("Bob");  
person.set_email("bob@example.com");  
  
fstream out("person.pb", ios::out | ios::binary | ios::trunc);  
person.SerializeToOstream(&out);  
out.close();
```


빅데이터의 수집 및 통합 기술

■ Avro

- 데이터 직렬화 시스템
- Thrift, Protocol Buffers와 기능은 비슷함
- 기본적으로 데이터를 활용할 수 있도록 스키마와 인터페이스 기능을 제공, 여기서 생성한 스키마는 데이터와 전송하며, 이 스키마는 JSON 파서의 기능으로 정의
- 인코딩과 디코딩을 할 때 저장 공간을 많이 차지 하며 성능을 저하시킨다는 단점이 있음
- 지원 언어 – C, C++, Java, PHP, Python, Ruby 등

빅데이터 저장 및 관리 기술

- CAP 이론: “적절한 응답 시간 내 세 가지 속성을 모두 만족시키는 분산 데이터베이스 시스템을 구성할 수 없다”는 이론
 - 분산 데이터베이스 시스템은 반드시 네트워크 장애 등으로 인해 장애가 발생함
 - 분산 데이터베이스 시스템은 장애 회복을 위해 반드시 분할 허용성을 가지고 있어야 함

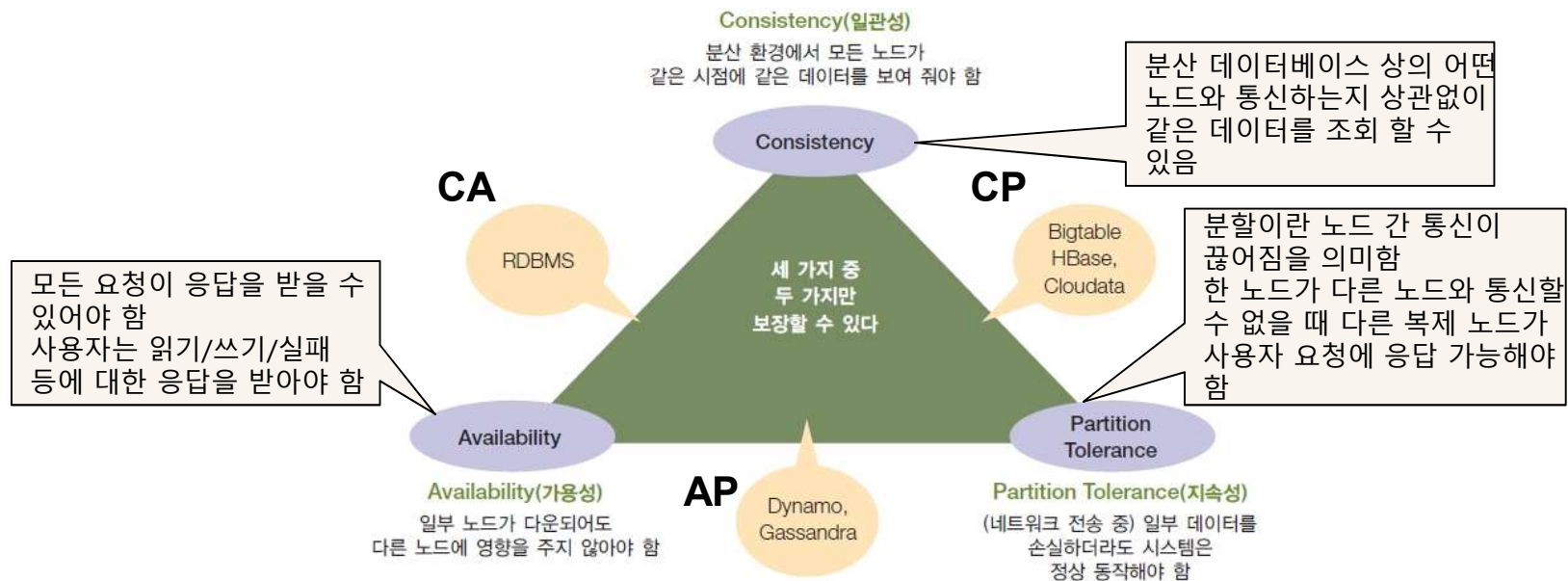


그림 3-1 CAP 이론

NOTE 2002년 버클리 대학의 Eric Brewer^{에릭 브루어} 교수가 발표한 CAP 이론은 분산 컴퓨팅 환경의 특징을 Consistency, Availability, Partition Tolerance 세 가지로 정의하고 있는데, 어떤 시스템이든 이 특성 모두를 동시에 만족하기는 어려우며 최대 두 가지만 만족할 수 있다는 것이다.

빅데이터 저장 및 관리 기술

■ 새로운 저장 기술의 등장 – NoSQL

□ NoSQL의 장 · 단점 및 특성

표 3-1 CAP 이론을 기준으로 한 RDBMS와 NoSQL의 비교

구분	설명	적용 예
RDBMS	일관성(C)과 가용성(A)을 선택	트랜잭션 ACID의 보장(금융 서비스)
NoSQL	일관성(C)이나 가용성(A) 중 하나를 포기하고, 지속성(P)을 보장	<ul style="list-style-type: none"> ▪ C+P형 : 대용량 분산 파일 시스템(성능 보장) ▪ A+P형 : 비동기식 서비스(아마존, 트위터 등)

표 3-2 RDBMS와 NoSQL의 장 · 단점 및 특성 비교

구분	RDBMS	NoSQL
장 · 단점	<ul style="list-style-type: none"> ▪ 데이터 무결성, 정확성 보장 ▪ 정규화된 테이블과 소규모 트랜잭션이 있음 ▪ 확장성에 한계가 있음 ▪ 클라우드 분산 환경에 부적합 	<ul style="list-style-type: none"> ▪ 데이터의 무결성과 정확성을 보장하지 않음 ▪ 웹 환경의 다양한 정보를 검색 · 저장 가능
특성	<ul style="list-style-type: none"> ▪ UPDATE, DELETE, JOIN 연산 가능 ▪ ACID 트랜잭션이 있음 ▪ 고정 스키마가 있음 	<ul style="list-style-type: none"> ▪ 수정 · 삭제를 사용하지 않음(입력으로 대체) ▪ 강한 일관성은 불필요 ▪ 노드의 추가 및 삭제, 데이터 분산에 유연

빅데이터 저장 및 관리 기술

■ NoSQL의 기술적 특성

NoSQL의 기술적 특성.

고정된 스키마는 DB에서 배우는 관계형 등의 데이터 모델을 따르는 데이터임

특성	내용
無 스키마	<ul style="list-style-type: none">데이터를 모델링하는 고정된 데이터 스키마 없이 키Key 값을 이용하여 다양한 형태의 데이터 저장 및 접근 가능데이터 저장 방식은 크게 열 Column, 값 Value, 문서 Document, 그래프 Graph 등의 네 가지를 기반으로 구분
탄력성Elasticity	<ul style="list-style-type: none">시스템 일부에 장애가 발생해도 클라이언트가 시스템에 접근 가능응용 시스템의 다운 타임이 없도록 하는 동시에 대용량 데이터의 생성 및 갱신질의를 대응할 수 있도록 시스템 규모와 성능 확장이 용이하며, 입출력의 부하를 분산시키는 데도 용이한 구조
질의Query 기능	수십 대에서 수천 대 규모로 구성된 시스템에서도 데이터의 특성에 맞게 효율적으로 데이터를 검색·처리할 수 있는 질의 언어, 관련 처리 기술, API 제공
캐싱Caching	<ul style="list-style-type: none">대규모 질의에도 고성능 응답 속도를 제공할 수 있는 메모리 기반 캐싱 기술을 적용하는 것이 중요개발 및 운영에도 투명하고 일관되게 적용할 수 있는 구조

파일 복사본을 임시 저장 위치에 저장하여 보다 빠르게 액세스 할 수 있도록 하는 프로세스임

빅데이터 저장 및 관리 기술

■ NoSQL의 분류

NoSQL의 분류.

데이터 모델	설명	제품 예
〈키, 값〉 저장 구조	<ul style="list-style-type: none">가장 간단한 데이터 모델범위 질의는 사용이 어려움(DB에서 지원하면 사용 가능)응용 프로그램 모델링이 복잡함	아마존 DynamoDB, 아마존 S3
문서 저장 구조	<ul style="list-style-type: none">문서에는 다른 스키마가 있음레코드 간의 관계 설명이 가능개념적으로 RDBMS와 비슷	아마존 SimpleDB, 아파치 CouchDB, MongoDB
열 기반 저장 구조	<ul style="list-style-type: none">연관된 데이터 위주로 읽는 데 유리한 구조하나의 레코드를 변경하려면 여러 곳을 수정해야 함동일 도메인의 열 값이 연속되므로 압축 효율이 좋음범위 질의에 유리	아파치 Cassandra

빅데이터 저장 및 관리 기술

■ 빅데이터 주요 저장 및 관리 기술

빅데이터 주요 저장 및 관리 기술.

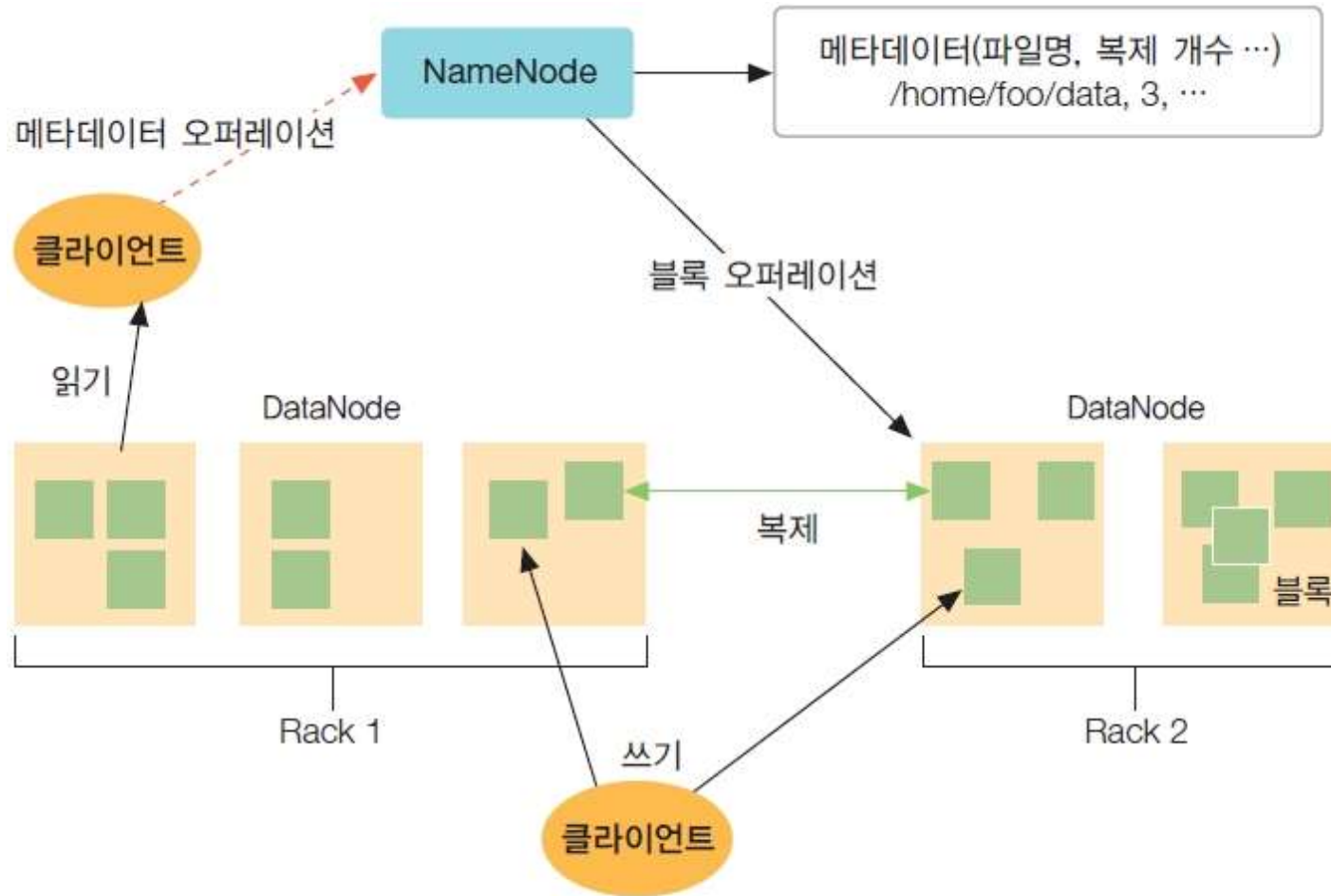
제품/기술	최초 개발	최초 공개	주요 기능 및 특징
S3	아마존	2006년	아마존의 인터넷 스토리지 서비스
HDFS	아파치	2008년	분산 환경에 기반한 데이터 관리
DynamoDB	아마존	2007년	SSD를 이용한 하둡 기반 데이터 분석·저장
MongoDB	10gen	2009년	DB의 수평 확장 및 범위 질의, 맵리듀스 연산
CouchDB	D. Katz	2005년	ACID 속성을 유지한 분산 데이터베이스
Cassandra	A. Lakshman	2008년	데이터의 저장 및 처리, 제한적 정렬
HBase	아파치	2007년	▪ 분산 클러스터 관리 및 복구 ▪ 데이터의 Get/Put/Scan/Delete 기능
Redis	VMware	2009년	정의된 데이터 타입을 이용한 구조화된 데이터 관리
Riak	Basho.com	2009년	링형 구조를 이용한 데이터 분산 저장
Hypertable	Zvents	2009년	▪ HBase와 비슷한 하둡 기반 데이터베이스 ▪ SQL과 비슷한 HQL 명령어 제공
ZooKeeper	아파치	2010년	여러 종류의 하둡 기반 시스템의 관리
Voldemort	LinkedIn	2009년	해싱을 이용한 빠른 조회, 동시 입력 데이터의 처리

빅데이터 저장 및 관리 기술

■ HDFS (Hadoop Distributed File Systems)

- 하둡은 아파치 진영에서 분산 환경 컴퓨팅을 목표로 시작한 프로젝트
- 파일 시스템은 분산 처리 환경에서 필수 조건으로 하둡은 HDFS를 제공함
- 하둡은 마스터-슬레이브 구조를 띰
 - 마스터 하나와 슬레이브 여러 개로 클러스터를 구성함
- HDFS에서는 마스터 노드를 NameNode라고 하며, 슬레이브 노드를 DataNode 라고 함
- HDFS는 대용량 파일을 클러스터에 여러 블록으로 분산하여 저장함
 - 이때 블록들은 마지막 블록을 제외 하고 모두 크기가 동일, 기본 크기는 64MB
- HDFS는 데이터 복제 기법을 지원함
 - 같은 데이터를 여러 노드에 분배
 - 특정 노드에 Failure 발생 시 복제본을 이용하여 복구

빅데이터 저장 및 관리 기술



HDFS의 전체 구성도.

빅데이터 저장 및 관리 기술

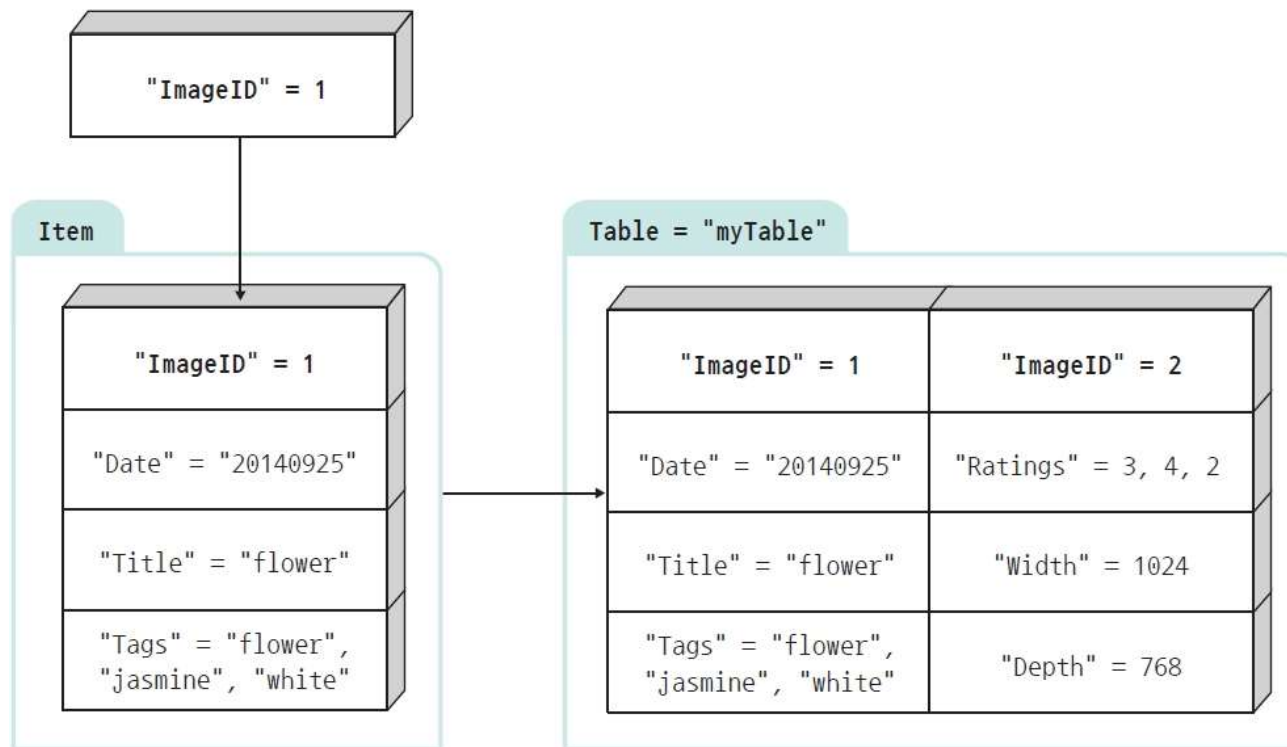
■ DynamoDB

- 아마존 웹에서 유·무료로 서비스
- 하드웨어 프로비저닝, 복제, 설정 패치, 사용하는 응용 프로그램에 따른 DB 자동 분할 기능 등을 지원하며, 사용자는 원하는 만큼 데이터베이스를 생성하거나 삭제하여 데이터를 저장할 수 있음
 - 프로비저닝: 사용자의 요구에 맞게 시스템 자원을 할당, 배치, 배포한 후 필요시 시스템을 즉시 사용할 수 있는 상태로 미리 준비해두는 것
- DynamoDB의 기본 데이터 모델
 - 속성, 항목, 테이블로 구성
 - 속성은 <이름, 값> 쌍으로 구성
 - 이름은 문자열이어야 하며, 값은 문자열, 숫자, 바이너리, 문자열 세트, 숫자나 바이너리 집합 등의 형태일 수 있음

```
"ImageID" = 1
"Title" = "flower"
"Tags" = "flower", "jasmine", "white"
"Ratings" = 3, 4, 2
```

빅데이터 저장 및 관리 기술

기본키는 DynamoDB 테이블의 항목을 구분한다. [그림 3-4]는 ImageID를 기본키로 지정한 속성을 나타낸 것이다. 테이블에는 이름(myTable)이 있으나 항목에는 이름이 없고, 기본키는 항목을 정의한다(예를 들어, 기본키 "ImageID"=1인 항목). 테이블에는 항목이 있고, 개별 영역에 정보가 저장된다. 테이블의 모든 항목은 동일한 기본키 체계로, 테이블을 생성할 때 기본키에 사용할 속성 이름을 지정한다. 테이블의 각 항목에는 고유한 기본키 값이 있어야 한다.



DynamoDB의 기본키 구성 예.

빅데이터 저장 및 관리 기술

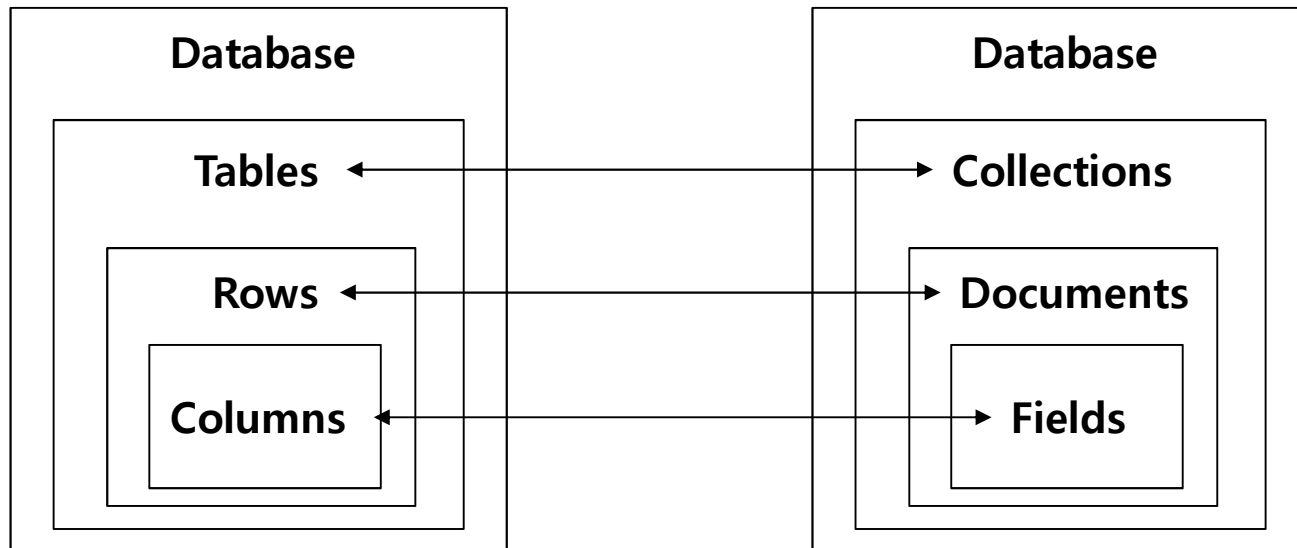
테이블 : 내 이미지						
기본키	기타 속성					
ImageID =1	ImageLocation = https://amazonaws.com/buket/img_1.jpg	Date = 1260653179	Title = flower	Tags = Flower, Jasmine	Width = 1024	Depth = 768
ImageID =2	ImageLocation = https://amazonaws.com/buket/img_2.jpg	Date = 1252617979	Rated = 3, 4, 2	Tags = Work, Seattle, Office	Width = 1024	Depth = 768
ImageID =3	ImageLocation = https://amazonaws.com/buket/img_3.jpg	Date = 1285277179	Price = 10.25	Tags = Seattle, Grocery, Store	Author = you	Camera = phone
ImageID =4	ImageLocation = https://amazonaws.com/buket/img_4.jpg	Date = 1282598779	Title = Hawaii	Author = Joe	Colors = orange, blue, yellow	Tags = beach, blanket, ball

DynamoDB의 테이블 구성 예.

빅데이터 저장 및 관리 기술

■ MongoDB

- 신뢰성과 확장성에 기반한 문서 지향 데이터베이스
 - 방대한 양의 데이터에서 낮은 관리 비용과 사용 편의성을 목표로 하는 MongoDB는 10gen이 오픈 소스로 개발한 것으로, 상업적인 지원이 가능
- MongoDB에서 저장의 최소 단위는 문서
 - 각 문서들은 RDBMS의 테이블과 비슷한 컬렉션이라는 곳에 수집하며, 각 컬렉션은 데이터베이스에서 관리



빅데이터 저장 및 관리 기술

■ MongoDB (계속)

- MongoDB는 자동-샤딩 (Auto-Sharding)을 이용한 분산 확장이 가능함
 - 샤딩은 데이터를 분할하여 다른 서버에 나누어 저장하는 과정
 - 기존 DBMS의 범위 질의, 보조 인덱스, 정렬 등 연산과 맵리듀스 등 집계 연산을 함께 지원
 - 데이터는 BSON 형태로 저장하며, C++로 작성
 - 서버의 용량이 부족할 때 새로운 샤드 노드를 추가하면 자동으로 재분산을 수행
- MongoDB는 조인과 트랜잭션 기능을 지원하지 않음
 - 데이터를 역정규화 하여 사용

빅데이터 저장 및 관리 기술

■ MongoDB (계속)

□ MongoDB를 사용하는 이유

- 신뢰성: 서버 장애에도 서비스는 계속 동작
- 확장성: 데이터를 샤딩하여 수평확장(Scale-out) 가능
- 유연성: 여러가지 형태의 데이터를 손쉽게 저장
- 인덱스 지원: Collection 당 최대 64개의 인덱스 생성 가능

빅데이터 저장 및 관리 기술

■ Cassandra

- <키, 값> 구조의 오픈소스 NoSQL 시스템으로 페이스북에 적용하여 사용함
- Master-Slave 구조인 Hadoop 등에서 나타나는 Single Point Failure 문제가 없음
 - Non-centralized 구조를 띠고 있음 (Master 노드가 없음)

facebook IBM ebay

DELL CISCO Microsoft SanDisk

Cassandra 활용처.

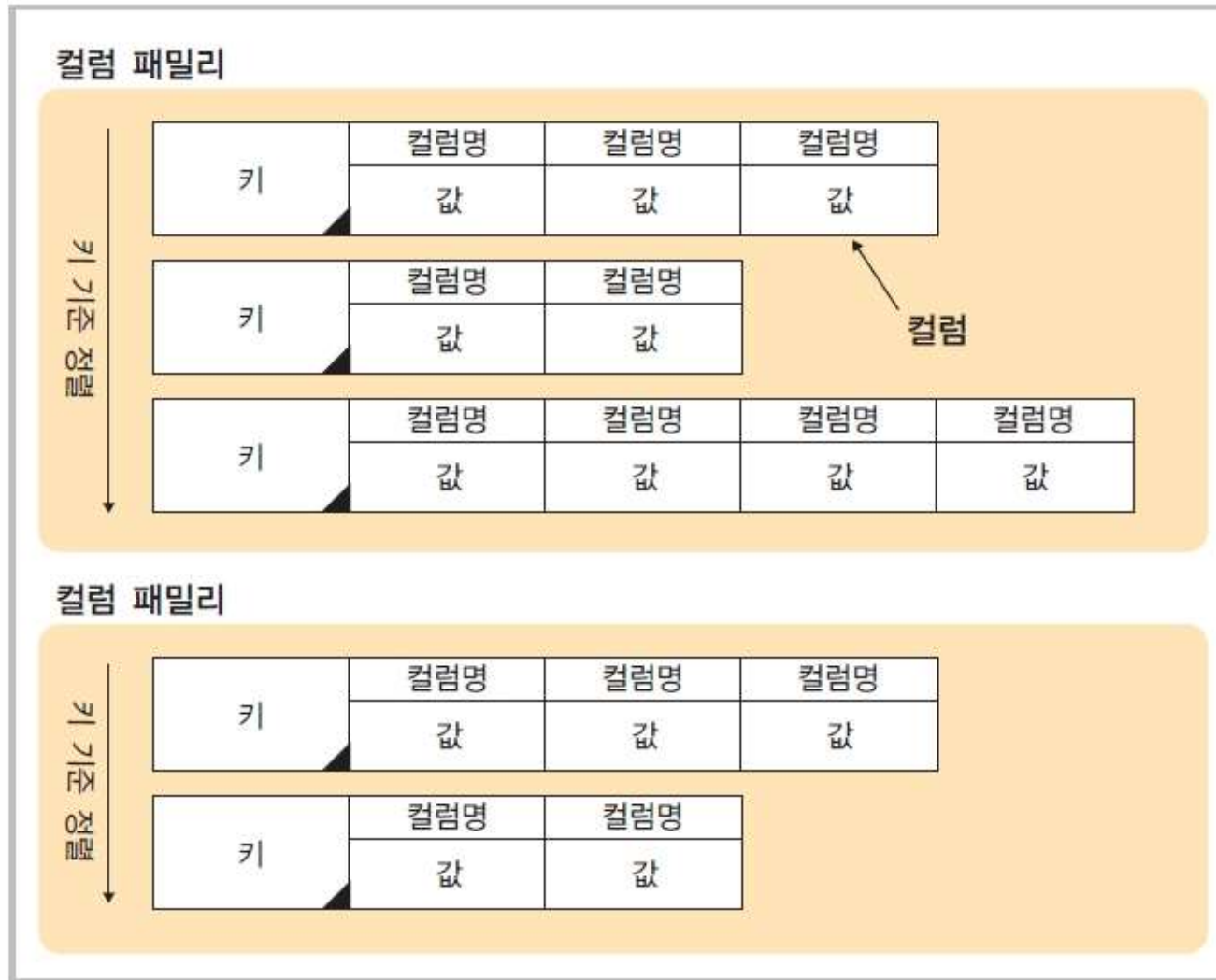
빅데이터 저장 및 관리 기술

■ Cassandra (계속)

- 토큰링 배경의 키 구간이 설정되어 있어 서버(노드)의 추가 및 제거만으로도 전체 저장 공간의 유연한 확장 및 축소가 가능함
- 다른 서버(노드)에 데이터 복제본을 구성하여 특정 노드에 장애가 발생해도 서비스에 영향을 주지 않고, 데이터가 유실되지 않음
- 수정·추가·삭제가 빠름
 - 실제 스토리지 구조에 적용하기 전에 먼저 CommitLog에 변경 사항을 기록함
- 1차 인덱스는 열 집합의 열 이름, 2차 인덱스는 열의 값을 기반함
- 데이터 전송 프로토콜로 Thrift를 사용하며, 사실상 언어에 의존 없이 모든 환경에서 이용 가능함
- 물리 파일 저장 구조로 SSTable (Sorted, String, Table)을 사용함
 - 물리 파일을 저장하는 구조 자체가 1차 정렬 조건에 맞추어 사전 정렬된 형태를 유지한다

빅데이터 저장 및 관리 기술

키 영역



Cassandra 데이터 저장 구조.

빅데이터 저장 및 관리 기술

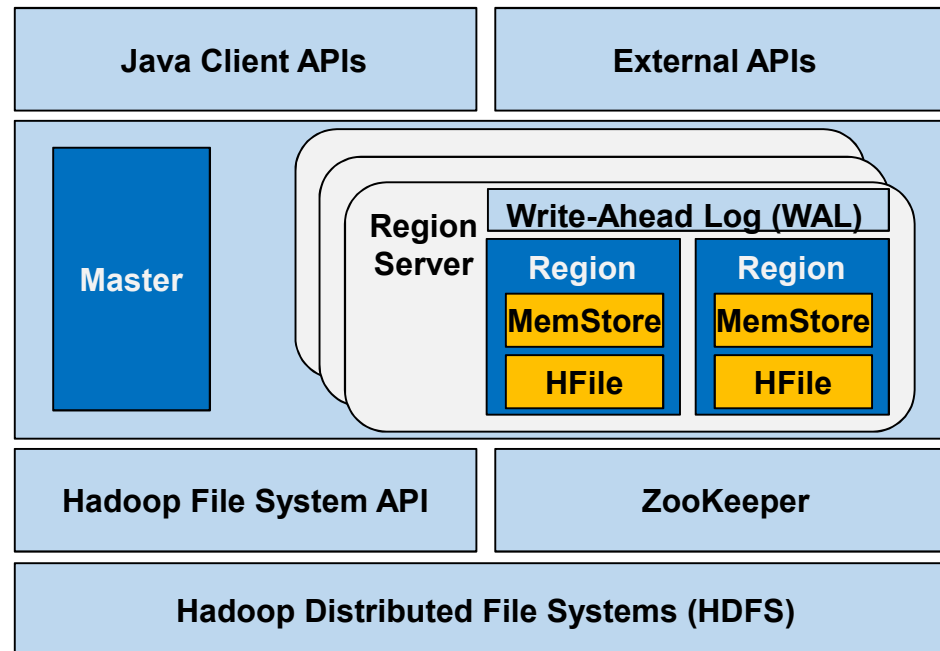
■ HBase (Hadoop dataBASE)

- 행, 열 그룹, 열 이름, 타임스탬프를 이용한 테이블 구조
- 하둡 파일 시스템 위에 설치되며, ZooKeeper를 노드 관리에 사용
- 읽기와 수정은 즉시 실행되며, 맵리듀스 연산은 일괄 처리됨
- 각 프로세스는 자신의 레코드를 비동기적으로 업데이트함
- 영역 서버 (Region Server) 간의 시스템 대체 작동과 불량 클러스터 복구 기능, Get/Put/Scan/Delete의 네 가지 동작을 지원함
- 데이터 모델은 열 집합 기반의 저장소로 구성됨

빅데이터 저장 및 관리 기술

■ HBase (계속)

- 마스터 서버는 리전 서버에 특정 영역을 할당하는 일 담당 및 리전 서버 간 부하 분산 처리 담당 (Zookeeper 이용하여)
- 과부하가 걸린 서버의 부하를 덜어주고 여유 공간이 많은 서버에 영역 (Region)을 옮겨주는 작업을 수행



Hbase 구성요소.

빅데이터 저장 및 관리 기술

개념적 저장 공간

로우키	타임스탬프	컬럼 "contents:"	컬럼 "anchor:"		컬럼 "mine:"
"com.cnn.www"	t9		"anchor.cnnsi.com"	"CNN"	
	t8		"anchor.my.look.ca"	"CNN.com"	
	t6	"<html>..."			"text/html"
	t5	"<html>..."			
	t3	"<html>..."			

물리적 저장 공간

로우키	타임스탬프	컬럼 "contents:"
"com.cnn.www"	t6	"<html>..."
	t5	"<html>..."
	t3	"<html>..."

로우키	타임스탬프	컬럼 "anchor:"	
"com.cnn.www"	t9	"anchor.cnnsi.com"	"CNN"
	t8	"anchor.my.look.ca"	"CNNcom"

로우키	타임스탬프	컬럼 "mine:"
"com.cnn.www"	t6	"text/html"

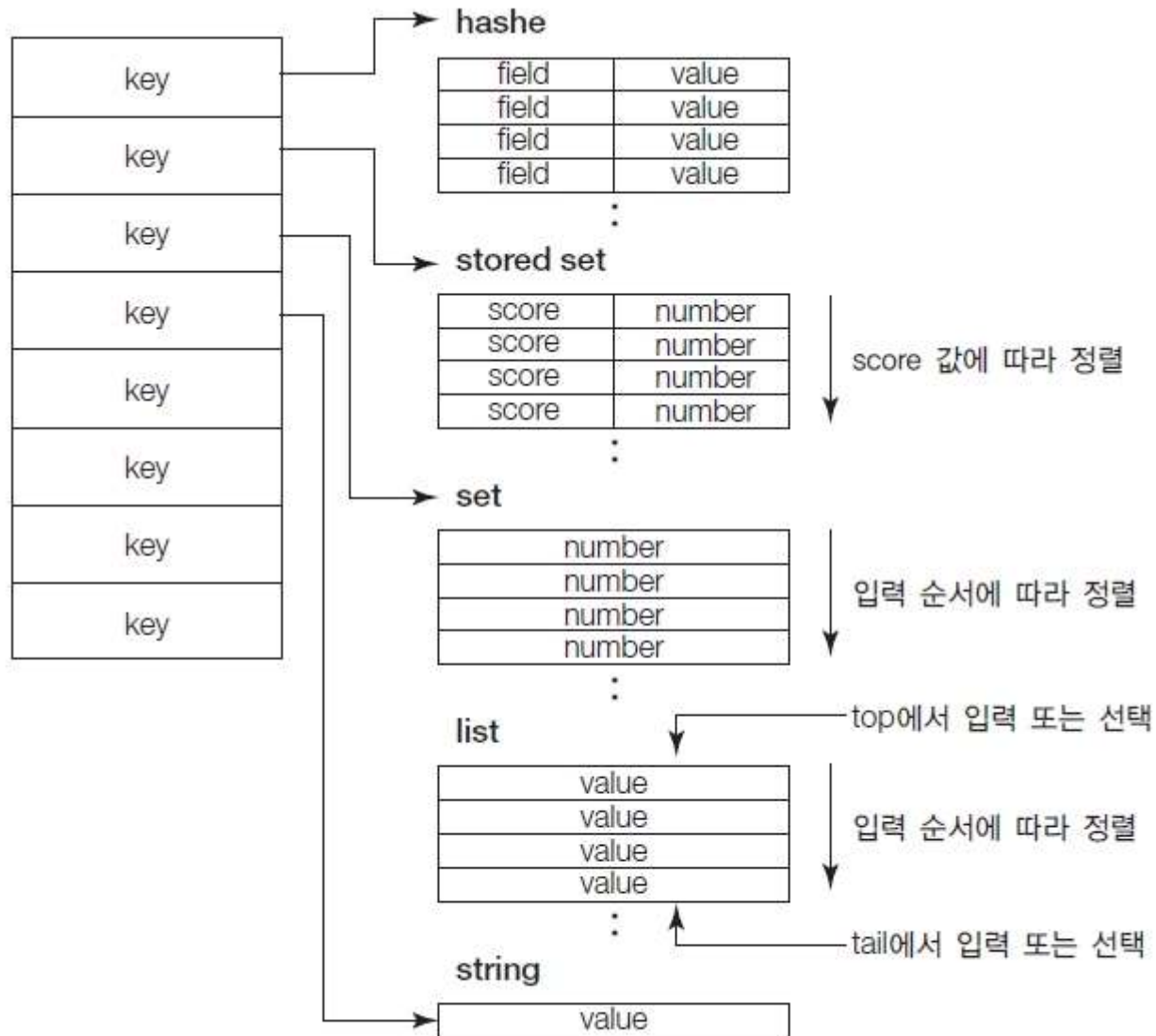
Hbase 개념적 및 물리적 저장 공간 비교.

빅데이터 저장 및 관리 기술

■ Redis (Remote Dictionary System)

- 메모리 기반의 <키, 값> 저장 공간, NoSQL 이나 인메모리 솔루션으로 분류하기도 함
 - 다양한 데이터 구조를 지원하며, 메모리에 저장된 내용을 지속시키려고 파일로 싱크하는 기능을 제공
 - 데이터 캐싱시 자주 사용됨
- 데이터 타입
 - String : 일반적인 문자열로 최대 512MB까지 지원, 문자열뿐만 아니라 숫자와 JPEG 등 바이너리 파일도 저장 가능
 - Set : string의 집합. set 간의 연산을 지원하는데 이것으로 교집합, 합집합, 차집합 등을 빨리 얻을 수 있음
 - stored set : 일종의 가중치가 설정된 데이터 타입. 데이터는 오름차순으로 내부 정렬되며, 정렬되어 있는 만큼 score 값 범위에 따른 질의, 톱 랭킹에 따른 질의 등이 가능
 - Hashe : 값 내의 <필드, 문자> 쌍으로, RDBMS의 기본키 한 개와 문자열 필드 하나로 구성된 테이블
 - list : 문자열의 집합으로 저장되는 데이터로 형태는 set과 비슷하지만, 일종의 양방향 연결 리스트임. list 앞과 뒤에서 Push/Pop 연산을 이용하여 데이터를 삽입·삭제할 수 있고, 지정된 인덱스 값을 이용하여 지정된 위치에 데이터를 삽입·삭제할 수 있음

빅데이터 저장 및 관리 기술

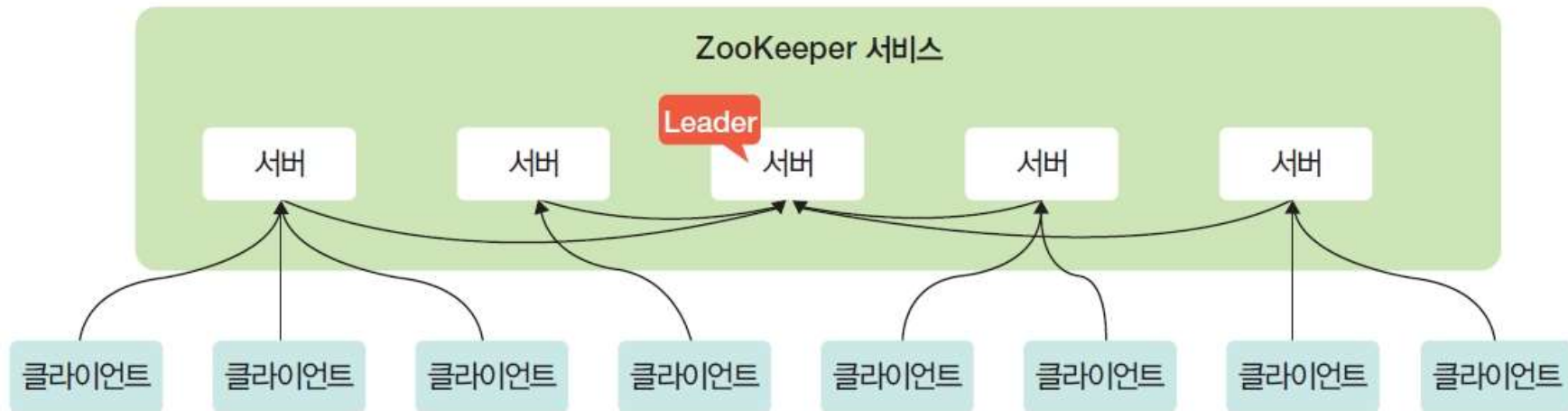


Redis의 저장 구조.

빅데이터 저장 및 관리 기술

■ ZooKeeper

- 하둡의 분산 처리 시스템(Hadoop, Chukwa, Pig 등)을 관리하는 분산 처리시스템을 일괄적으로 관리하는 시스템
- 다중 서버 집합을 묶어 관리해 주는 시스템으로 야후의 분산 코디네이터임
- 분산 처리 시스템의 장애 문제 해결을 위해 개발됨



ZooKeeper의 기본 동작 구조.

빅데이터 처리 기술

■ 개요

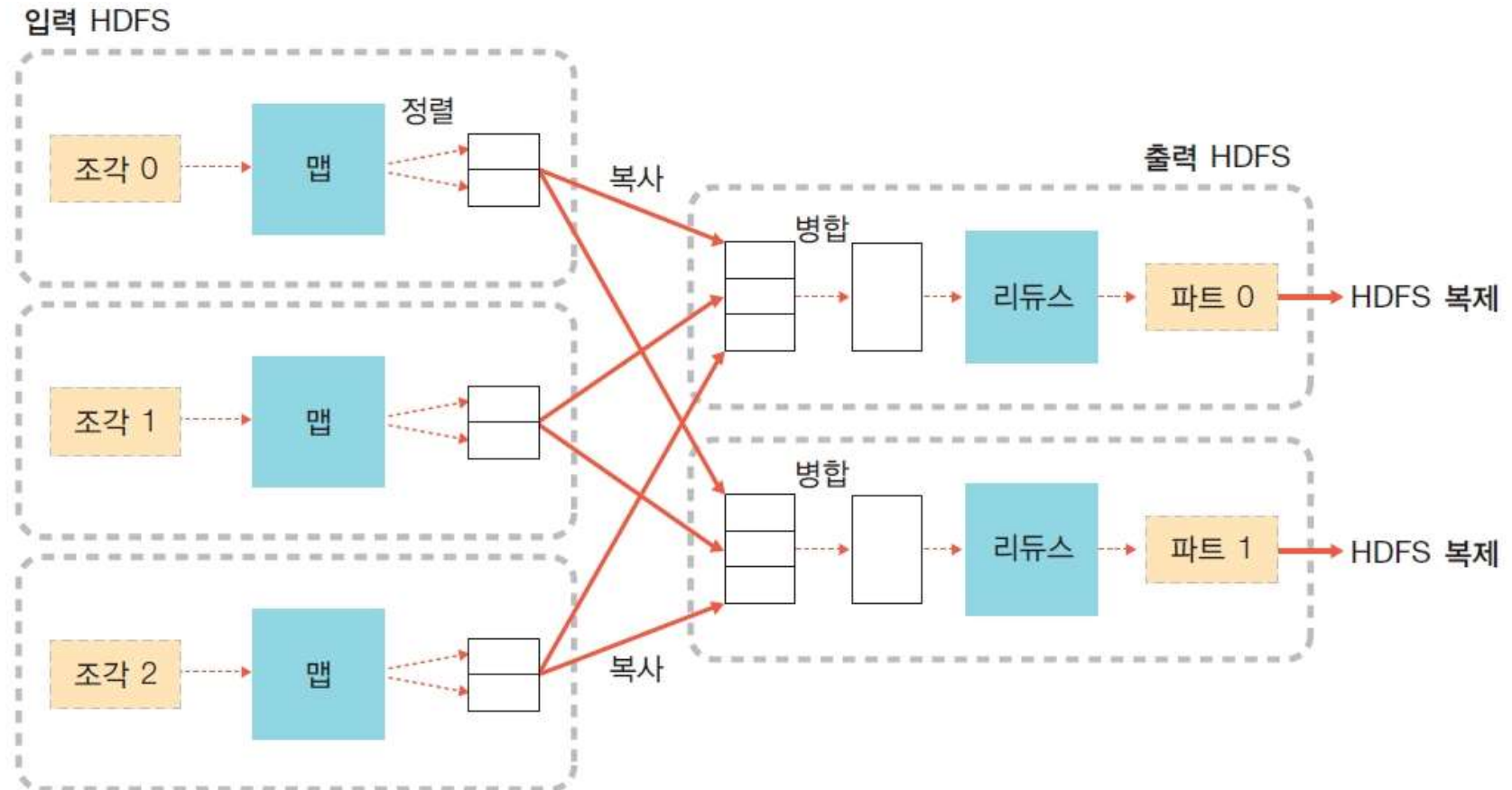
- 기존의 데이터 처리 방식이 방대한 양의 데이터를 한 번에 얼마나 빠르게 처리하는지에 초점을 맞췄다면, 현재의 데이터 처리 기술은 저장된 방대한 양의 데이터를 사용자가 원하는 부분에 맞춰 원하는 시간에 처리하는 데 초점을 둔다

빅데이터 처리 기술

■ Hadoop

- 여러 컴퓨터로 구성된 클러스터를 이용하여 방대한 양의 데이터를 처리하는 분산 처리 프레임워크. 엔진 형태로 되어 있는 미들웨어와 소프트웨어 개발 프레임워크로 구성되어 있음
- 즉시 응답해야 하는 트랜잭션 처리보다는 데이터를 모은 후 처리하여 작업을 완료해야 응답을 주는 방식으로 설계되었음
 - 따라서 어느 정도의 시간이 소요되는 방대한 양의 데이터 처리에 적합함
- 맵리듀스의 분산 처리 구조를 사용하며 맵리듀스는 하나의 큰 데이터를 여러개의 조각으로 나누어 처리하는 맵 단계와 처리된 결과를 하나로 모아서 취합한 후 결과를 도출해 내는 리듀스 단계로 구성되어 있음

빅데이터 처리 기술



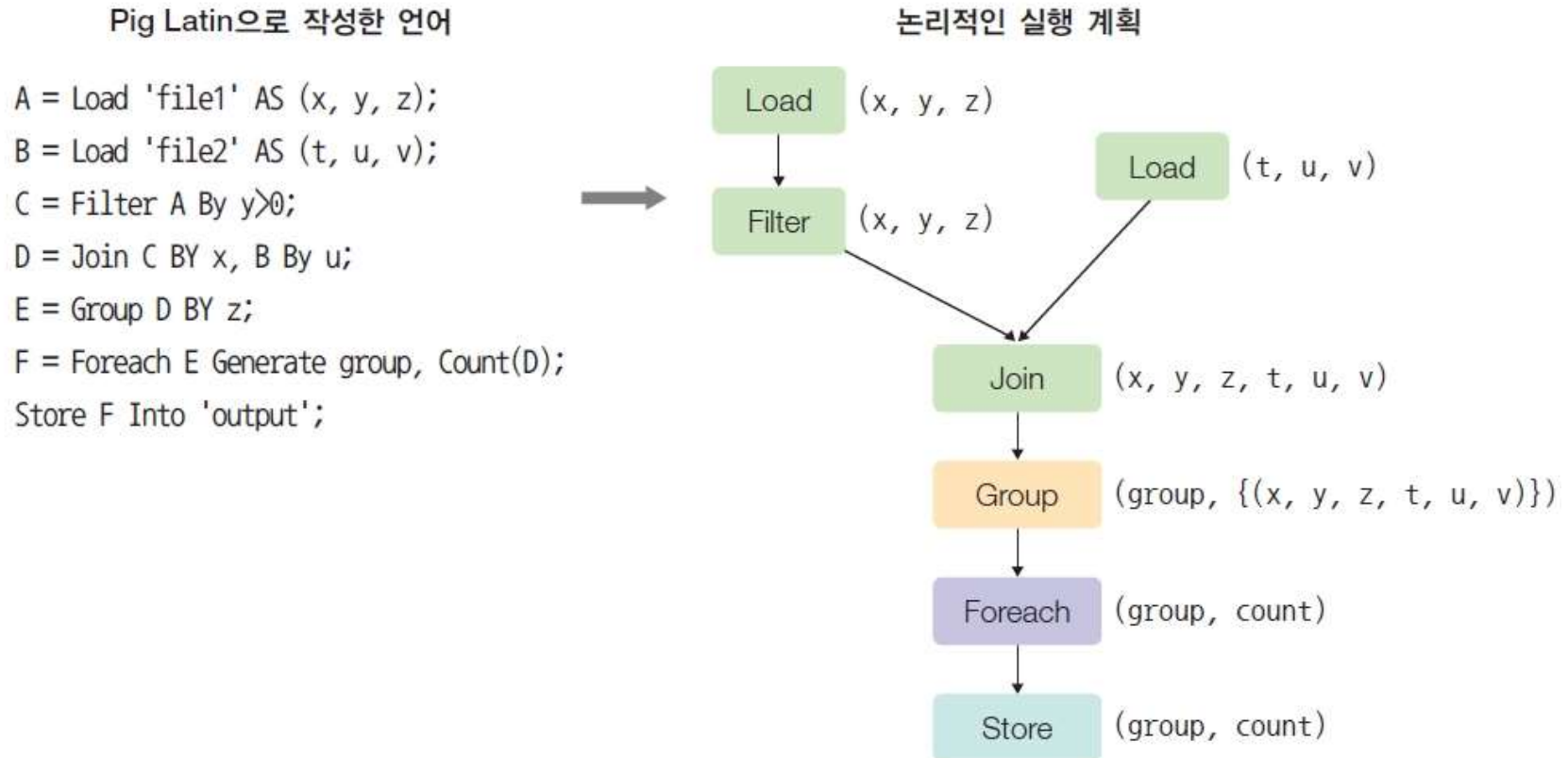
MapReduce 실행 단계.

빅데이터 처리 기술

■ Pig

- 아파치 하둡 세부 프로젝트 중 하나로, 절차적 데이터 처리 언어 프레임워크
- Pig는 고수준 언어로 데이터 분석을 프로그래밍할 수 있는 방대한 양의 데이터 분석 플랫폼이며, 이를 평가할 수 있는 인프라도 함께 제공하며 큰 특징은 대규모 병렬 처리에 대응할 수 있는 구조라 대규모 데이터 처리가 용이함
- Pig Latin의 특징
 - 데이터 흐름을 명시적으로 보여 줄 수 있는 코드 작성이 가능함
 - 이해하기 쉽고 유지보수가 용이
 - 시스템이 코드 실행을 자동으로 최적화하므로 사용자는 효율성을 생각하지 않고 프로그래밍 내용에만 집중할 수 있음
 - Pig에서 제공하는 Pig Latin은 int, long, double 등 기본형 외에 릴레이션(Relation), 백(Bag), 튜플(Tuple)과 같은 고수준의 구조를 제공하며, Filter, Foreach, Group, Join, Load, Store 등 관계(Relation; Table) 연산도 지원함, 사용자 지정 함수도 쉽게 정의 가능
 - Pig Latin으로 작성한 데이터 처리 프로그램은 논리적인 실행 계획으로 변환되고, 이것은 최종적으로 맵리듀스 실행 계획으로 변환됨

빅데이터 처리 기술



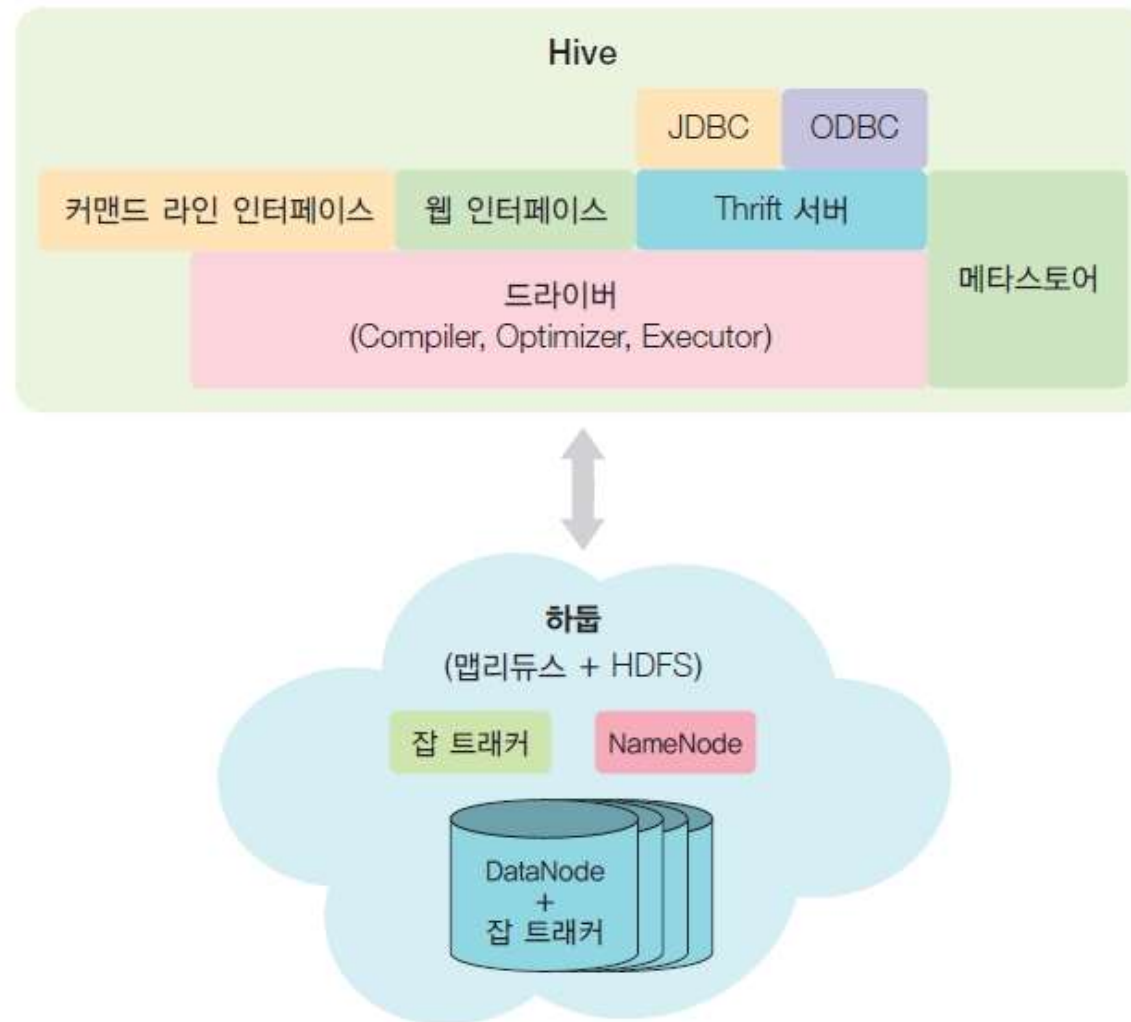
Pig의 동작 과정.

빅데이터 처리 기술

■ Hive

- 하둡에서 동작하는 SQL 프로그램을 구현할 수 있는 Hive는 하둡 기반의 데이터웨어하우스 인프라로, 관계형 데이터베이스에 익숙한 개발자에게 **훌륭한 인터페이스**를 제공함
 - Hive는 SQL과 같이 선언적으로 데이터를 처리할 수 있음
 - Hive는 HDFS나 HBase와 같은 빅데이터의 원본을 HiveQL 질의 언어를 이용하여 분석
- 맵리듀스 기반의 실행 부분과 데이터가 저장된 공간의 메타데이터 정보, 사용자나 응용 프로그램에서 질의를 입력 받아 실행시키는 실행 부분으로 구성됨
- 스칼라 값, 집합, 테이블 수준에서 사용자 정의 함수를 지정할 수 있는 기능을 제공하여 사용자 확장을 지원

빅데이터 처리 기술



Hive의 구성도.

빅데이터 처리 기술

■ Mrjob

- 하둡 이후로 Python을 이용하여 하둡을 쉽게 프로그래밍할 수 있게 도와주는 프레임워크가 등장
 - e.g., 대표적인 것이 바로 Hadoopy, Dumbo, Mrjob임
- 미국의 맛집 추천 사이트로 유명한 Yelp에서 만든 Python 라이브러리
- 데이터를 처리하는 코드를 작성한 후 한곳에서 작동하는 엘라스틱 맵 리듀스나 하둡 클러스터에서 동작하는 프레임워크를 제공
- 추상화나 내장 연산은 지원하지 않지만, 각 단계에 디버거를 실행하여 실제 코드가 내부에서 어떻게 동작하는지 파악 가능함

빅데이터 처리 기술

- Mrjob을 사용한 Yelp 웹 사이트 검색 결과 화면

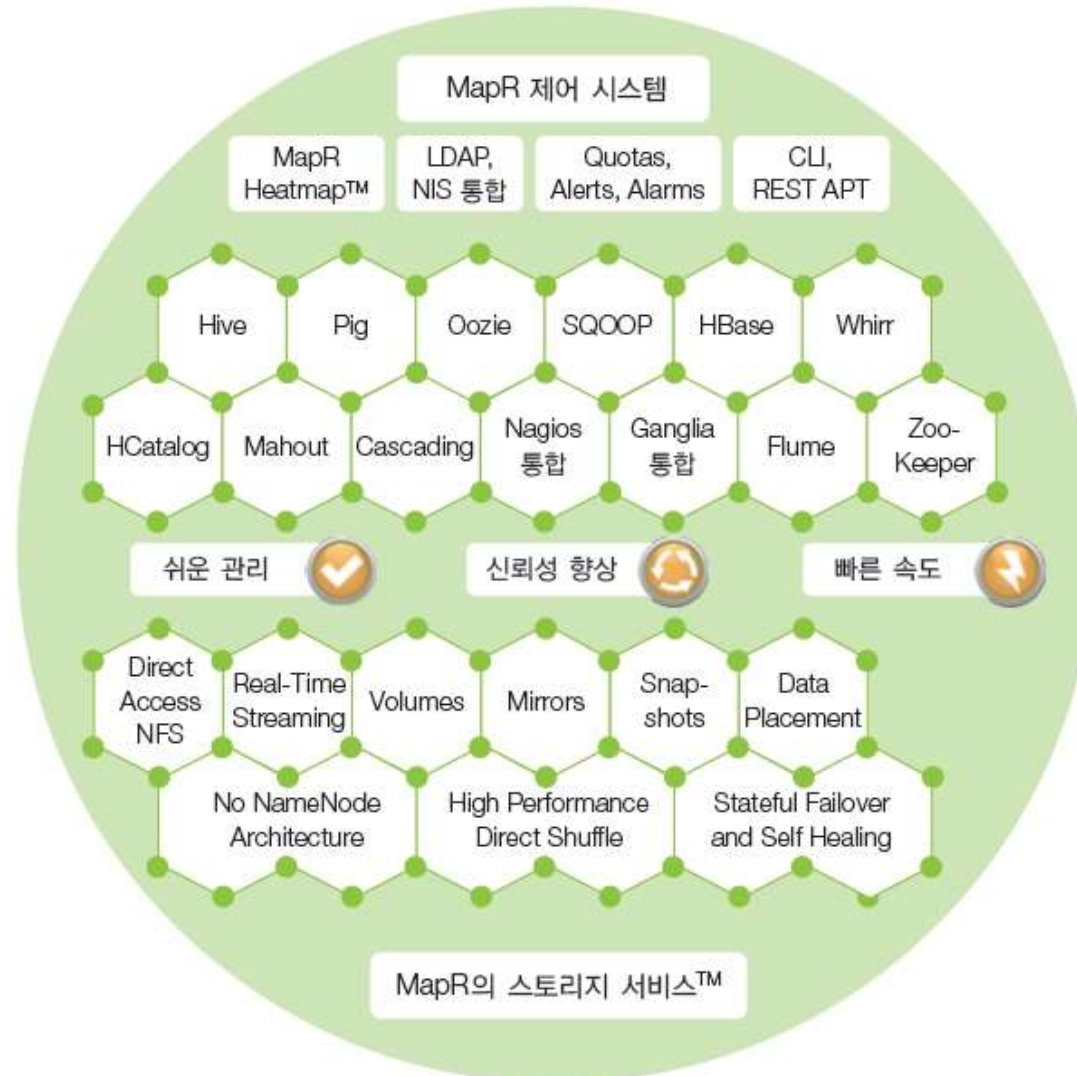


빅데이터 처리 기술

■ MapR

- 아파치 하둡의 배포판 공급 업체인 MapR Technologies에서 높은 신뢰성을 요구하는 기업에 제공하는 상업용 하둡임
- HDFS를 대체할 수 있는 자체 파일 시스템(MapRFS)으로 운영하며, 분산된 네임 노드가 있어 개선된 신뢰성을 제공함
 - HDFS에서 지원하는 기능은 거의 모두 제공
 - Shard 할 때 256MB의 Chunk 사용 → MetaData의 공간이 줄어들어 더 많은 수의 파일들을 저장/읽기/쓰기 가능함
 - I/O 할 때 단위가 8KB 단위 → Cluster의 오버헤드가 줄어듦
- MapR의 새로운 파일 시스템은 성능이 더욱 개선되고, 백업이 쉬우며, 네트워크 파일 시스템과 호환할 수 있는 등 데이터 전송의 단순화를 제공하여 맵리듀스 처리량을 향상하고 입출력 프로세싱 속도도 개선됨
- MapR의 프로그래밍 모델은 하둡과 같지만, 핵심 프레임워크를 둘러싼 인프라스트럭처를 개선하여 기업에 적합한 통합 솔루션이라 할 수 있음

빅데이터 처리 기술



MapR의 프레임워크 및 장점.

빅데이터 처리 기술



Azkaban

■ Azkaban

- LinkedIn의 오픈 소스 프로젝트인 Azkaban은 각 서비스가 여러 개의 연산들을 통합하여 비즈니스 로직으로 실행
 - 이것으로 사용자가 원하는 워크플로우를 정의할 수 있도록 개발한 배치스케줄러
- Azkaban은 사용자가 세부 작업까지 관리하기 어려운 문제를 해결하려고, 스케줄링 기법을 이용하여 진행한 작업을 서로 관련 있는 여러 단계로 나눠 처리하기 번거로운 세부 과정들을 처리
- 로그 기능을 제공하여 오류가 발생하면 이를 찾아 관리자에게 이메일로 상황을 알려 주며, 웹 인터페이스로 작업의 진행 상황을 파악
- 유닉스 명령어나 Java프로그램을 불러오는 최소한의 명령어로 된 텍스트 파일로 처리할 일을 구성하고, 복잡한 내부 구현은 각 명령어나 Java 프로그램으로 구현

빅데이터 처리 기술

Home

Executing Schedule Jobs

Scheduled Jobs

Name	Next Execution	Period	
Azkaban_WordCount	01-05-2012 02:10:00 UTC	1 day, 0 minutes	Remove

Current Time: 01-05-2012 02:08:22 UTC

Azkaban의 실행 화면.

빅데이터 처리 기술

■ Oozie

- Azkaban과 함께 하둡 기반의 워크플로우 제어 시스템
- 프로세스를 실행할 때 워크플로우에 관련된 결정을 특정 단계에서 내릴 수 있도록 XML 파일을 이용하여 설정
 - 작업들을 비순환 그래프(DAG: Direct Acyclic Graph)로 정의
- Java 서블릿 컨테이너에서 실행하는 Java 웹 응용 프로그램으로, 사용자가 시스템의 기능을 확장할 수 있는 API도 지원
- Oozie의 워크플로우 동작들은 원격 시스템에서 작업을 시작
 - 이 워크플로우는 제어플로우 노드와 동작 노드를 포함
 - 제어 플로우 노드는 워크플로우의 시작과 끝(start, end, fail 노드)을 정의 하며, 워크플로우의 실행 경로를 제어하는 메커니즘(decision, fork, join 노드)도 제공
 - 여기서 동작 노드는 계산·처리 작업을 실행시키는 원리

빅데이터 처리 기술



Oozie의 워크플로우.

감사합니다.

Contact: kw.chon@koreatech.ac.kr