REAL TIME TECH

빅데이터 실시간 분석 기술동향 및 적용사례

2013. 10. 08

(주)리얼타임테크

목 차

1. 빅데이터 개요

2. 빅데이터 분석 개요

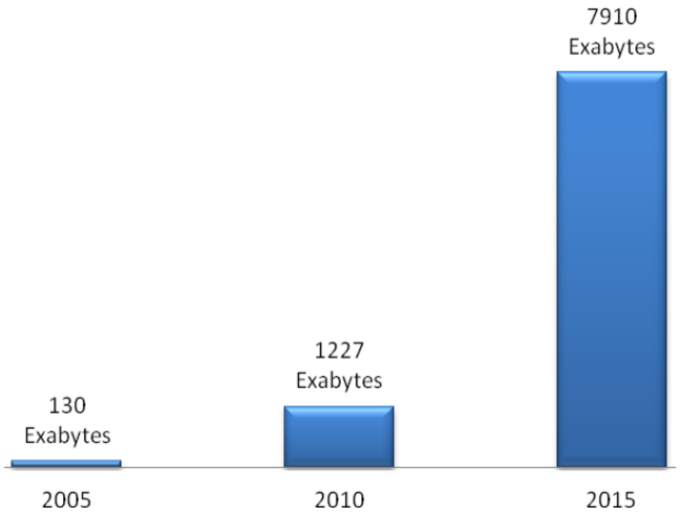
3. 빅데이터 분석 기술

4. 사례 연구

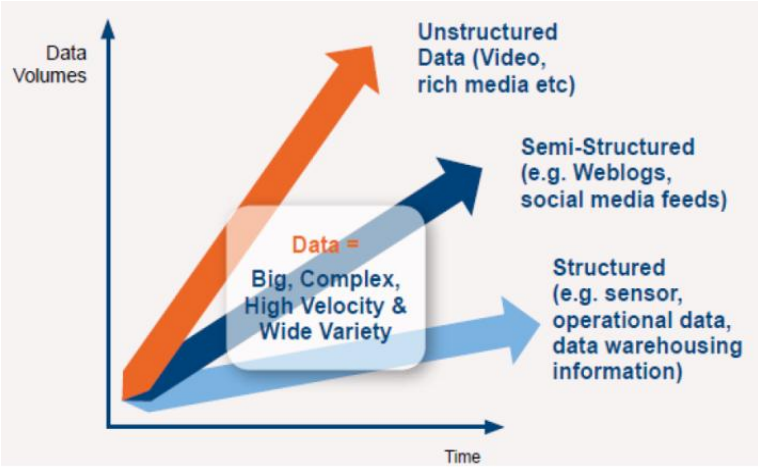
1. 빅데이터 개요

빅데이터 개요

❑ 빅데이터 기술의 등장 배경



Source : IDC Digital universe study(2011)



Data Volumes
Unstructured Data (Video, rich media etc)
Semi-Structured (e.g. Weblogs, social media feeds)
Structured (e.g. sensor, operational data, data warehousing information)
Data = Big, Complex, High Velocity & Wide Variety
Time

Source : IDC (2012)

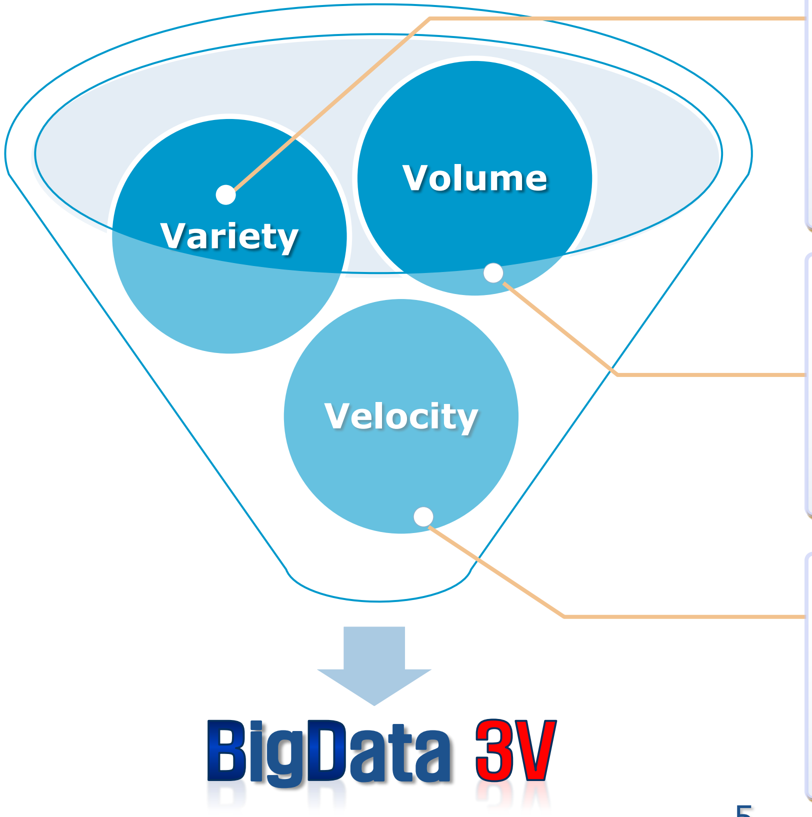
- ✓ Digital Universe: the total amount of data stored in the world's computers
- ✓ **The rapid rate(over 45%) of data growth**
- ✓ Problem of storage and processing speed, etc.

- ✓ **Over 90% of data : Unstructured and semi-structure data**
  - Conventional data processing ?
- ✓ The **frequency of data generation and delivery**
  - Should be applied to data in motion

빅데이터 개요

❑ 빅데이터 정의

"Big data technologies describe a ***new*** generation of technologies and architectures, designed to economically extract value from ***very large volumes*** of a wide ***variety*** of data, by enabling ***high-velocity*** capture, discovery, and/or analysis. " – *Definition of IDC*



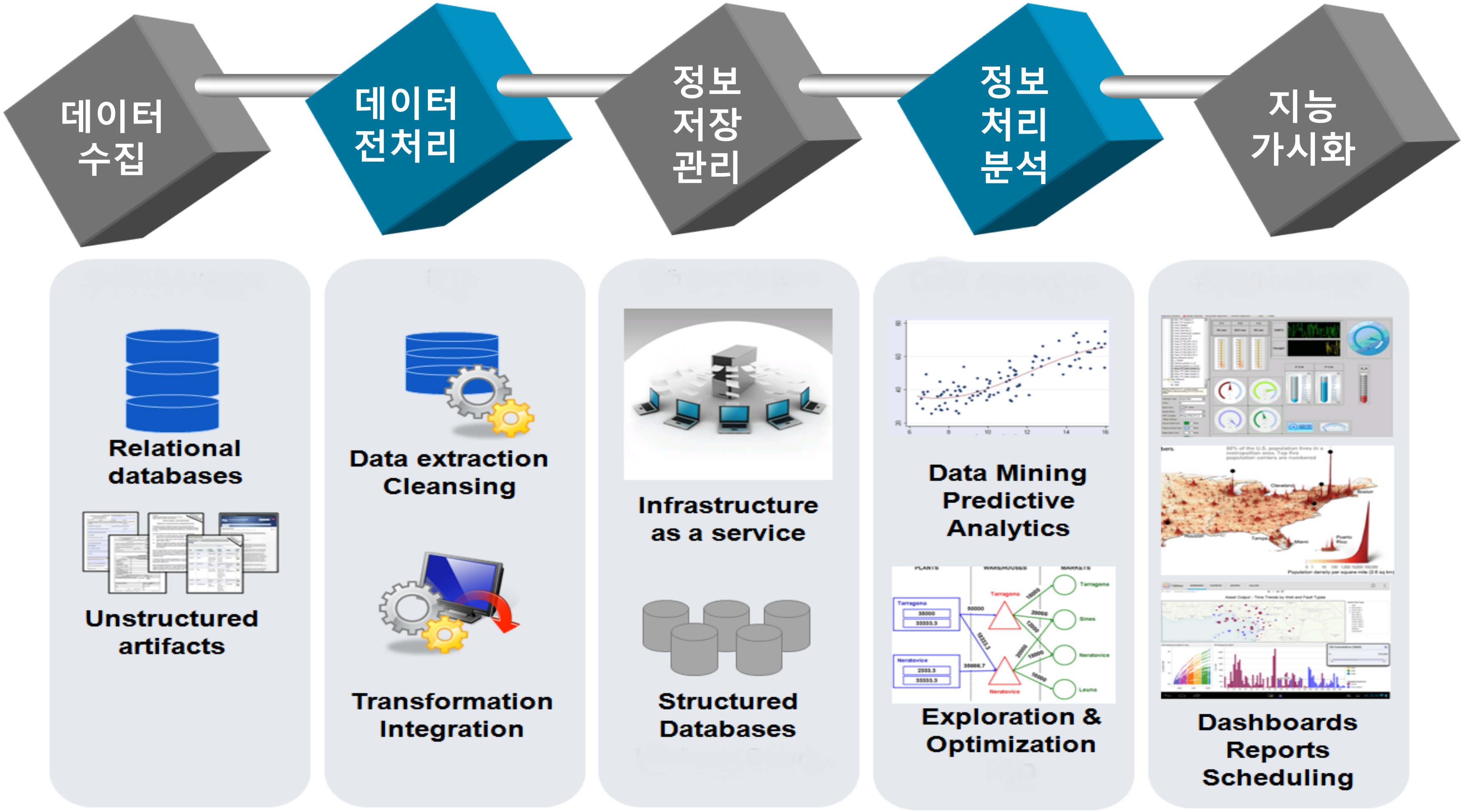
● **데이터의 다양화**
- ☞ 비정형 데이터(Unstructured Data) 처리 필요
- ☞ 시스템 유연성 지원
- ☞ 사용자 정의 프로세스 및 새로운 처리 모델

● **데이터의 대용량화(Beyond DBMS capacity)**
- ☞ 시스템의 확장성(Scalability)
- ☞ 분산 컴퓨팅 기술
- ☞ Parallelism

● **데이터의 고속 처리(분석)**
- ☞ 의사 결정 속도 중요, 지연 최소화
- ☞ 인메모리 컴퓨팅 및 슈퍼컴퓨팅 기술
- ☞ Stream processing

빅데이터 개요

❑ 빅데이터 플랫폼의 구성



데이터 수집

데이터 전처리

정보 저장 관리

정보 처리 분석

지능 가시화

Relational databases

Unstructured artifacts

Data extraction Cleansing

Transformation Integration

Infrastructure as a service

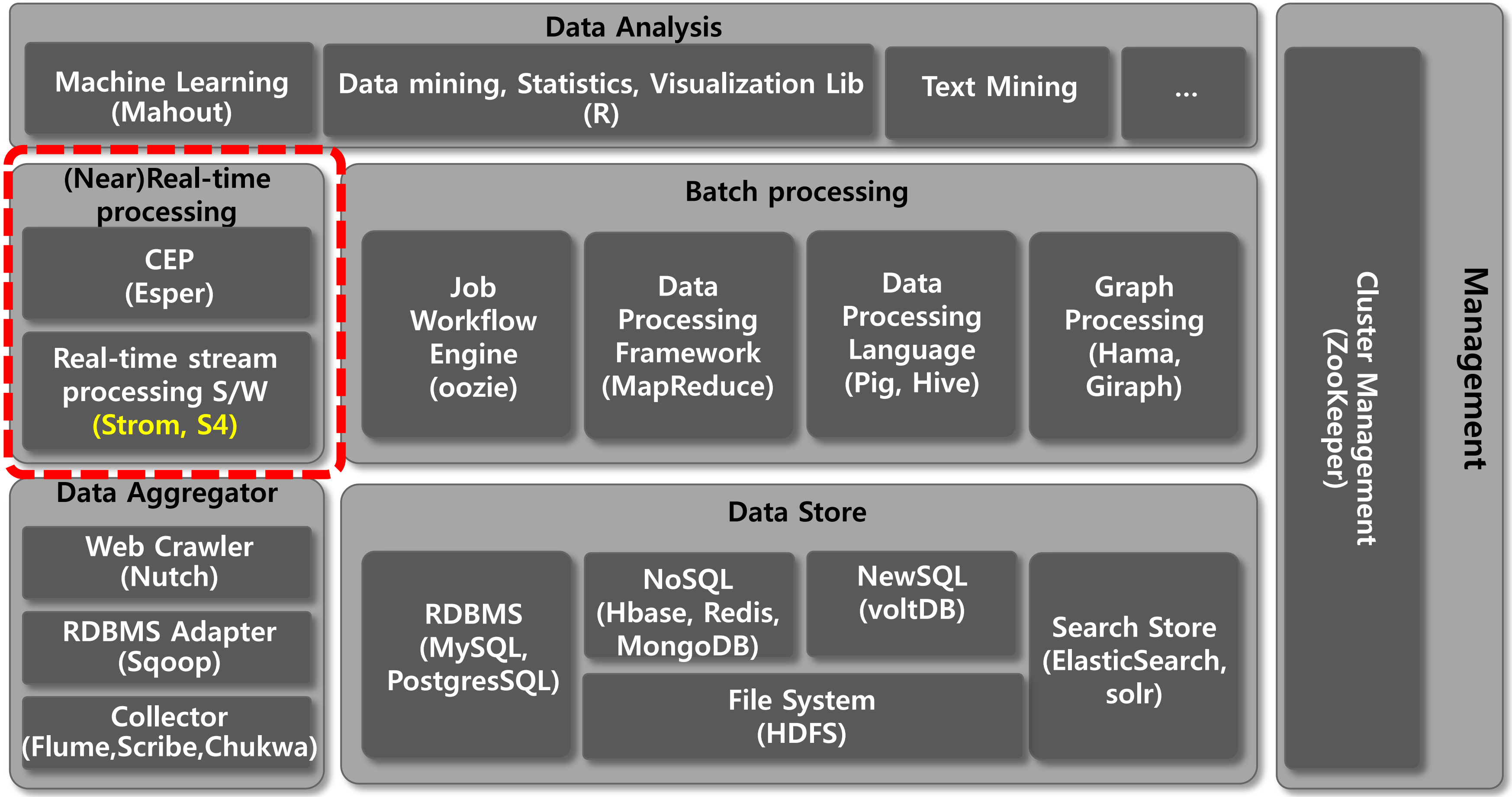
Structured Databases

Data Mining Predictive Analytics

Exploration & Optimization

Dashboards Reports Scheduling

❑ Open Source 기반 빅데이터 플랫폼(1/2)



Data Analysis

Machine Learning (Mahout)

Data mining, Statistics, Visualization Lib (R)

Text Mining

…

(Near)Real-time processing

CEP (Esper)

Real-time stream processing S/W (Strom, S4)

Batch processing

Job Workflow Engine (oozie)

Data Processing Framework (MapReduce)

Data Processing Language (Pig, Hive)

Graph Processing (Hama, Giraph)

Data Aggregator

Web Crawler (Nutch)

RDBMS Adapter (Sqoop)

Collector (Flume,Scribe,Chukwa)

Data Store

RDBMS (MySQL, PostgresSQL)

NoSQL (Hbase, Redis, MongoDB)

NewSQL (voltDB)

File System (HDFS)

Search Store (ElasticSearch, solr)

Management

Cluster Management (ZooKeeper)

빅데이터 개요

❑ Open Source 기반 빅데이터 플랫폼(2/2)

Category	Software	Description
Data Collection	Flume, Scribe, Chukwa	Collecting data from data source
	sqoop	Data delivery between HDFS and RDBMS
	Nutch	Web crawler
Data Store	HDFS	Distributed file system
	Hbase, Redis, MongoDB	Key-value based data-base management system
	voltDB	RDBMS supporting scalability and ACID
	Elastic search, Solr	Search engine
Real-time Analytics	Storm, S4	Real-time distributed and parallel data processing
	Esper	Processing stream data and providing high-level language
Batch Analytics	Oozie	Workflow scheduler for Hadoop job
	MapReduce	Batch distributed and parallel data processing
	Pig, Hive	Providing analytic operation and high-level language for big-data
	Goraph, Hama	Providing distributed and parallel programming model for big graph data
Mining	Mahout	Machine learning
	R	Statistics, data mining, visualization library
Management	zookeeper	Distribution coordinator for Cluster management

2. 빅데이터 분석 개요

빅데이터 분석 개요

❑ 분석 기술 발전 방향

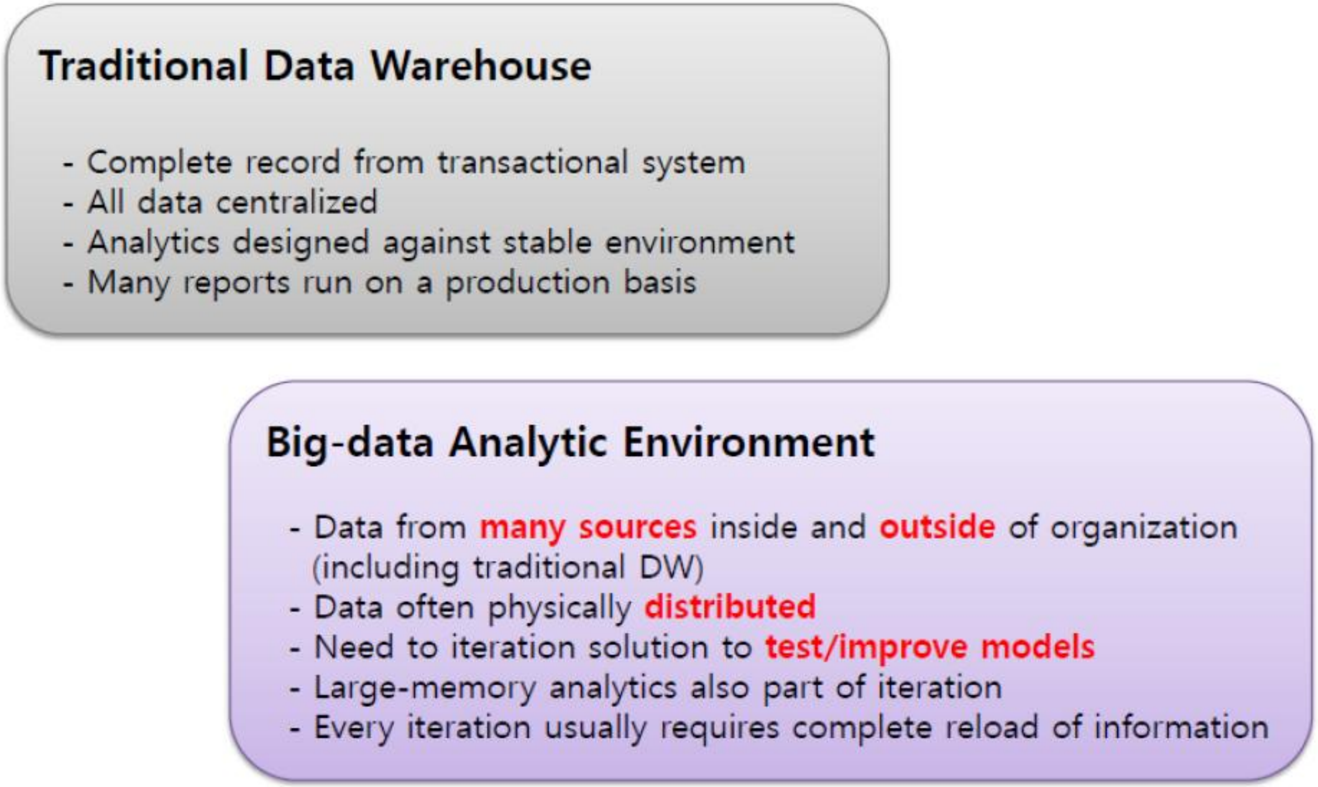
Flow of concept in Big-Data analytics

	Descriptive	Predictive	Prescriptive
Question	▪ What happened ? ▪ What is happening ?	▪ What will happen ? ▪ Why will it happen ?	▪ What should I do ? ▪ Why should I do it ?
Enables	▪ Business reporting ▪ Dashboards ▪ Scoreboards ▪ Data warehousing	▪ Data mining ▪ Text mining ▪ Web/Media mining ▪ Forecasting	▪ Optimization ▪ Simulation ▪ Decision modeling ▪ Export system
Outcomes	Well defined business problems and opportunities	Accurate projections of future states and conditions	Best possible business decisions and transactions

Past ⟷ Future

빅데이터 분석 개요

❑ 분석 환경 변화



Traditional Data Warehouse

- Complete record from transactional system
- All data centralized
- Analytics designed against stable environment
- Many reports run on a production basis

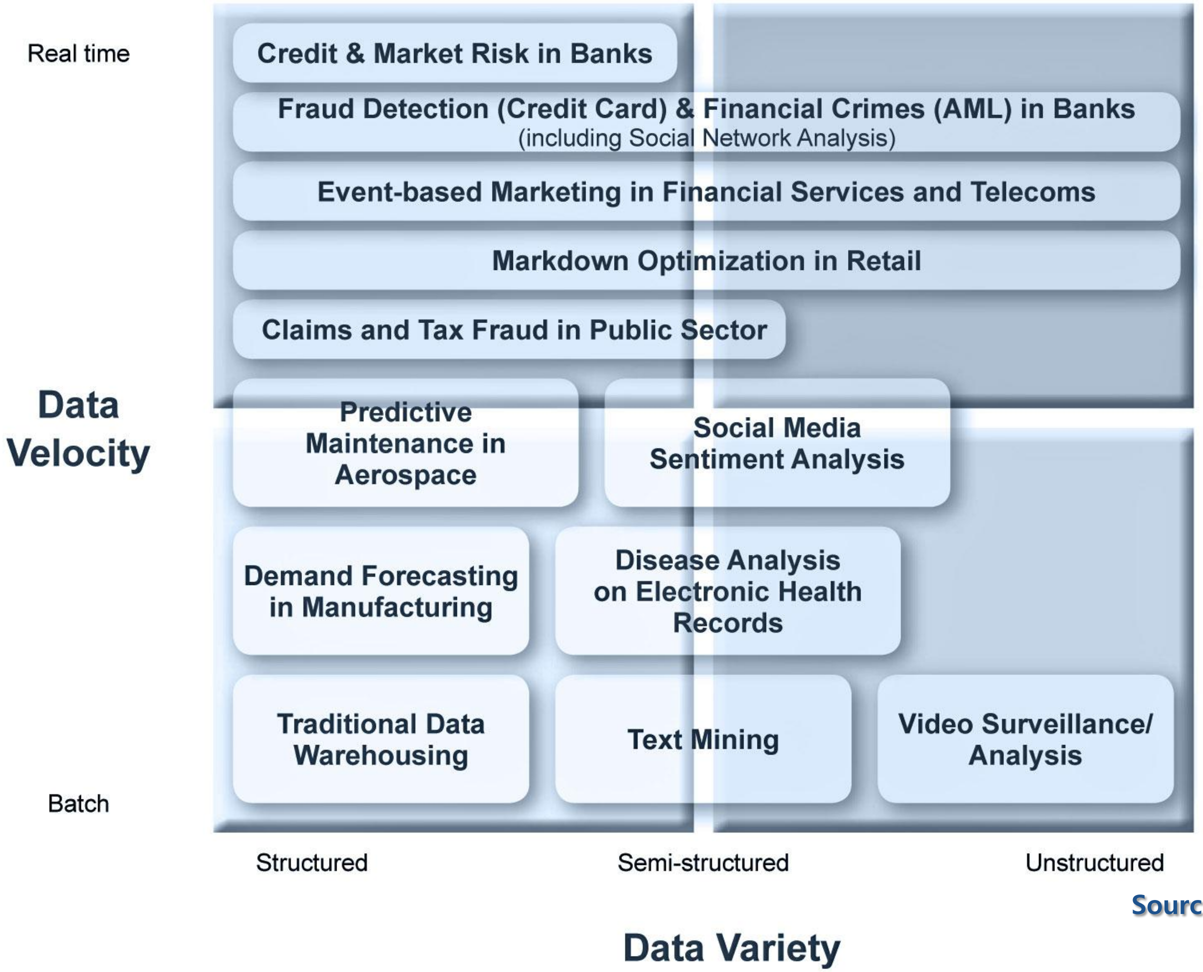
Big-data Analytic Environment

- Data from **many sources** inside and **outside** of organization (including traditional DW)
- Data often physically **distributed**
- Need to iteration solution to **test/improve models**
- Large-memory analytics also part of iteration
- Every iteration usually requires complete reload of information

http://wikibon.org/wiki/v/Enterprise_Big-data

빅데이터 분석 개요

❑ 분석 기술 적용 분야 (Potential Use cases)



Real time

Data Velocity

Batch

Credit & Market Risk in Banks

Fraud Detection (Credit Card) & Financial Crimes (AML) in Banks (including Social Network Analysis)

Event-based Marketing in Financial Services and Telecoms

Markdown Optimization in Retail

Claims and Tax Fraud in Public Sector

Predictive Maintenance in Aerospace

Social Media Sentiment Analysis

Demand Forecasting in Manufacturing

Disease Analysis on Electronic Health Records

Traditional Data Warehousing

Text Mining

Video Surveillance/ Analysis

Structured

Semi-structured

Unstructured

Data Variety

Source : SAS & IDC

3. 빅데이터 분석 기술

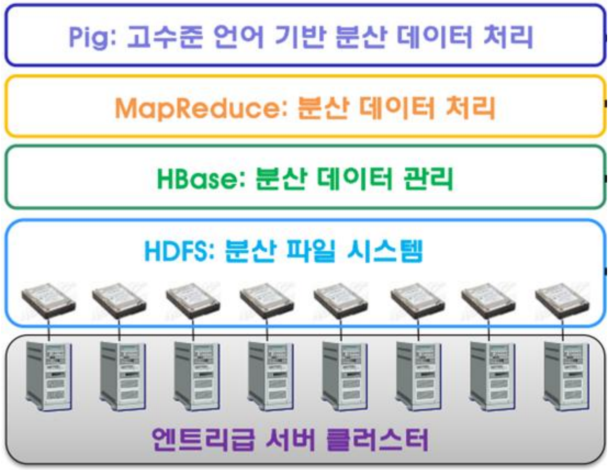
①빅데이터 배치 분석 기술
②빅데이터 실시간 분석 기술

빅데이터 배치(Batch) 분석 기술

❑ Hadoop overview

- ✓ Google 플랫폼의 클론으로 2004년 시작된 아파치 오픈 소스 프로젝트이며 현재, Big data 저장 / 분석 주류 플랫폼으로 성장

- ✓ Software platform that lets one easily write and run applications that process vast amounts of data. It includes:
  - – **MapReduce** – offline computing engine
  - – **HDFS** – Hadoop distributed file system
  - – **HBase** (pre-alpha) – online data access

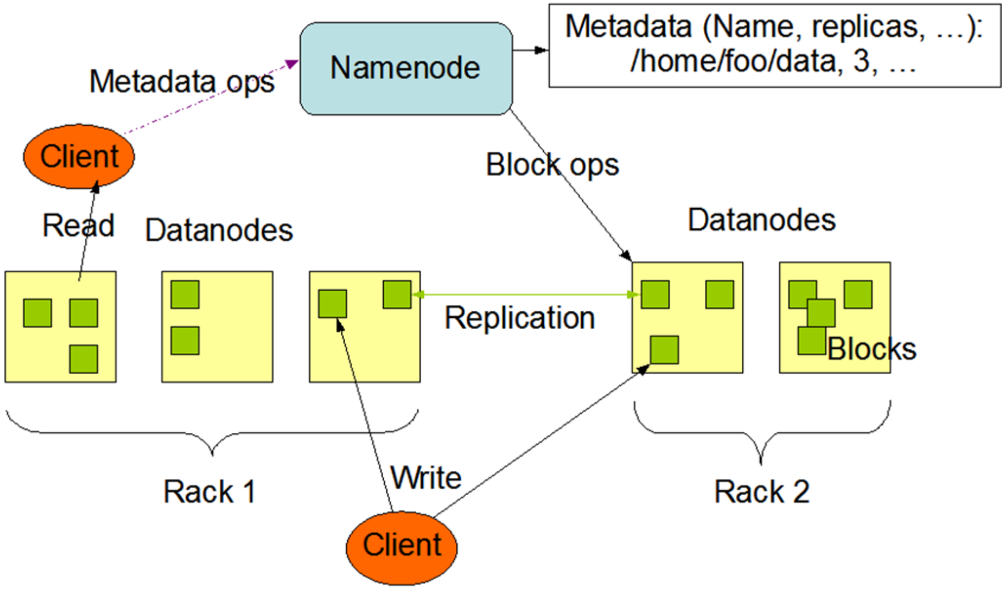


- ✓ Why Hadoop useful
  - – **Scalable:** It can reliably store and process petabytes.
  - – **Economical:** It distributes the data and processing across clusters of commonly available computers (in thousands).
  - – **Efficient:** By distributing the data, it can process it in parallel **on the nodes where the data is located.**
  - – **Reliable:** It automatically maintains multiple copies of data and automatically redeploys computing tasks based on failures.

빅데이터 배치(Batch) 분석 기술

❑ HDFS

- ✓ The **Hadoop Distributed File System (HDFS)** is a distributed file system designed to run on **commodity hardware**. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant.
  - **highly fault-tolerant** and is designed to be deployed on low-cost hardware.
  - provides **high throughput** access to application data and is suitable for applications that have large data sets.
  - relaxes a few POSIX requirements to enable streaming access to file system data.
  - part of the **Apache Hadoop Core project**.



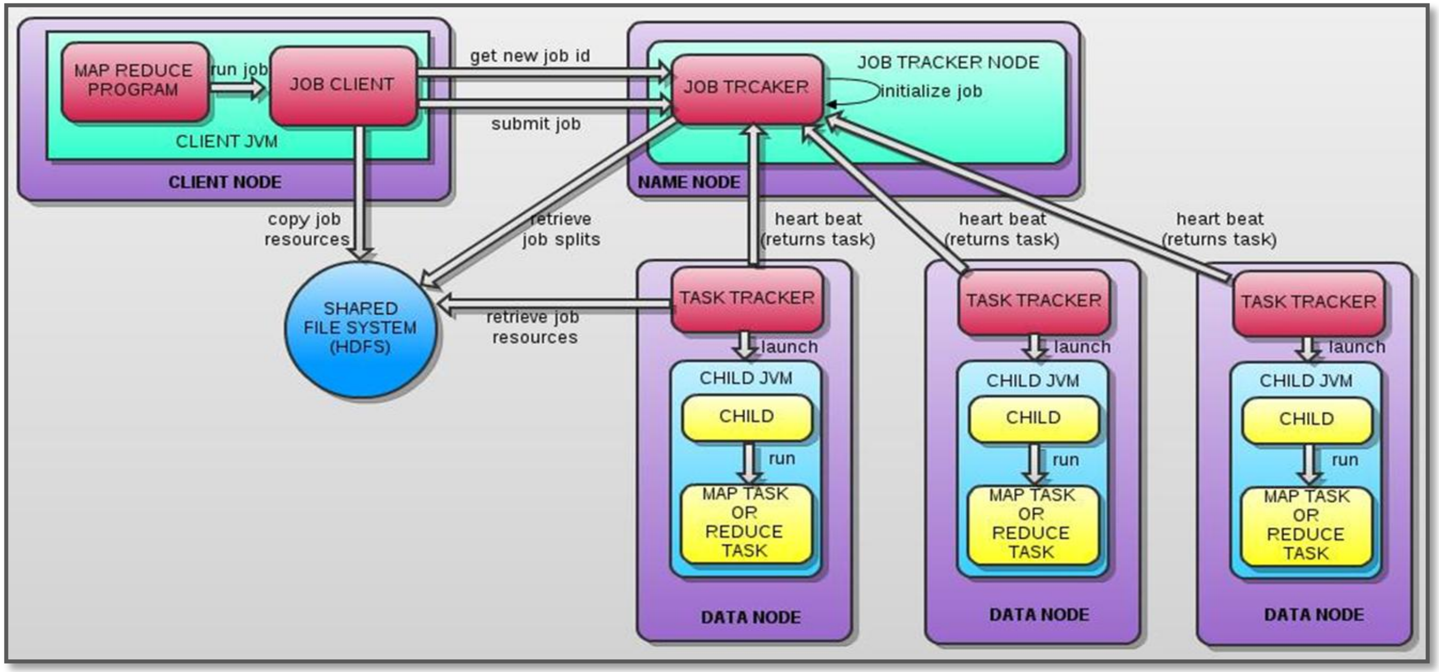
빅데이터 배치(Batch) 분석 기술

❑ MapReduce

- ✓ A programming model developed at **Google**
- ✓ Sort/merge based **distributed computing**
- ✓ Used extensively by more organizations (e.g., Yahoo, Amazon.com, IBM, etc.)
- ✓ It is **functional style programming**(e.g., LISP) **parallelizable across a large cluster of workstations or PCs**.

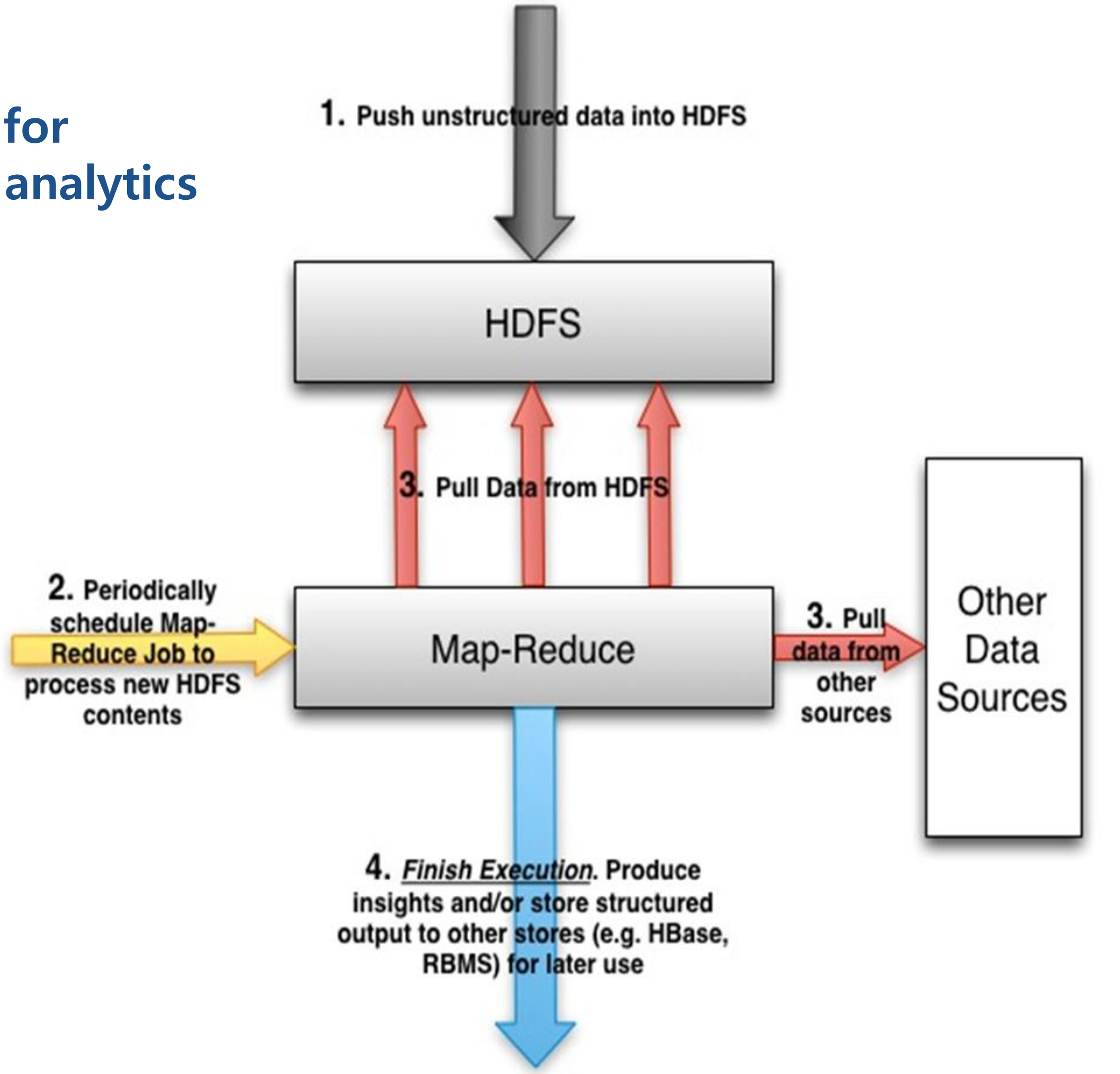
✓ **Key features for Hadoop 's success**

- – partitioning of the input data
- – scheduling the program's execution across several machines
- – handling machine failures
- – managing required inter-machine communication.



빅데이터 배치(Batch) 분석 기술

❑ Working model for offline-batched analytics


1. Push unstructured data into HDFS
HDFS
3. Pull Data from HDFS
2. Periodically schedule Map-Reduce Job to process new HDFS contents
Map-Reduce
3. Pull data from other sources
Other Data Sources
4. *Finish Execution*. Produce insights and/or store structured output to other stores (e.g. HBase, RBMS) for later use

빅데이터 배치(Batch) 분석 기술

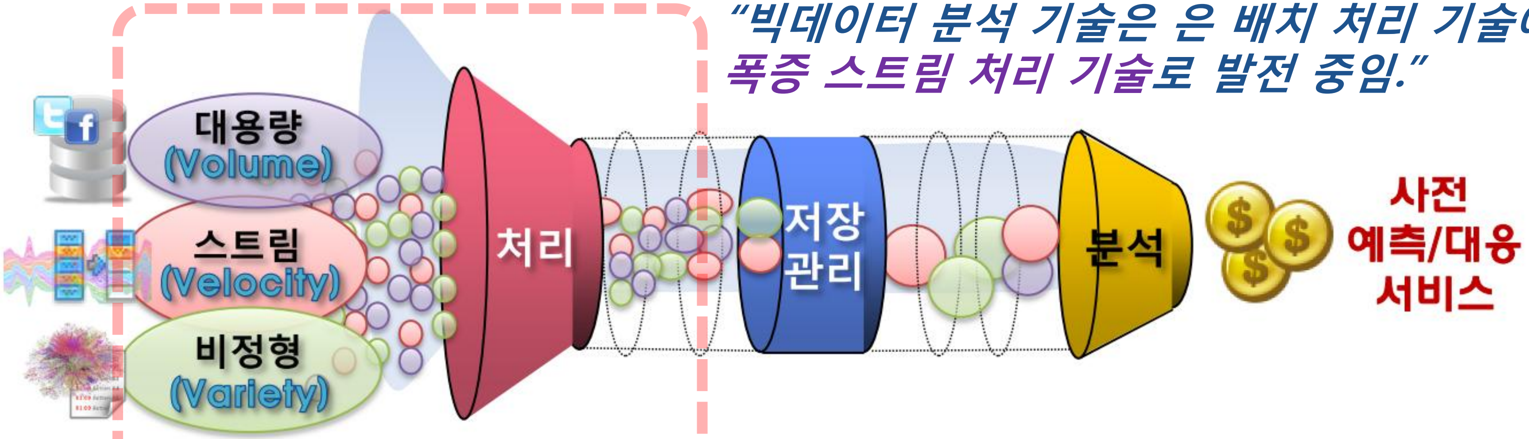
❑ Example applications of Hadoop

- A9.com – Amazon: To build Amazon's product search indices; process millions of sessions daily for analytics, using both the Java and streaming APIs; clusters vary from 1 to 100 nodes.
- Yahoo! : More than 100,000 CPUs in ~20,000 computers running Hadoop; biggest cluster: 2000 nodes (2*4cpu boxes with 4TB disk each); used to support research for Ad Systems and Web Search
- AOL : Used for a variety of things ranging from statistics generation to running advanced algorithms for doing behavioral analysis and targeting; cluster size is 50 machines, Intel Xeon, dual processors, dual core, each with 16GB Ram and 800 GB hard-disk giving us a total of 37 TB HDFS capacity.
- Facebook: To store copies of internal log and dimension data sources and use it as a source for reporting/analytics and machine learning; 320 machine cluster with 2,560 cores and about 1.3 PB raw storage;
- FOX Interactive Media : 3 X 20 machine cluster (8 cores/machine, 2TB/machine storage) ; 10 machine cluster (8 cores/machine, 1TB/machine storage); Used for log analysis, data mining and machine learning
- University of Nebraska Lincoln: one medium-sized Hadoop cluster (200TB) to store and serve physics data;
- Adknowledge - to build the recommender system for behavioral targeting, plus other clickstream analytics; clusters vary from 50 to 200 nodes, mostly on EC2.
- Contextweb - to store ad serving log and use it as a source for Ad optimizations/ Analytics/reporting/machine learning; 23 machine cluster with 184 cores and about 35TB raw storage. Each (commodity) node has 8 cores, 8GB RAM and 1.7 TB of storage.
- Cornell University Web Lab: Generating web graphs on 100 nodes (dual 2.4GHz Xeon Processor, 2 GB RAM, 72GB Hard Drive)
- NetSeer - Up to 1000 instances on Amazon EC2 ; Data storage in Amazon S3; Used for crawling, processing, serving and log analysis
- The New York Times : Large scale image conversions ; EC2 to run Hadoop on a large virtual cluster
- Powerset / Microsoft - Natural Language Search; up to 400 instances on Amazon EC2 ; data storage in Amazon S3

빅데이터 실시간 분석 기술

❑ 빅데이터 실시간 분석 플랫폼

"빅데이터 분석 기술은 은 배치 처리 기술에서 폭증 스트림 처리 기술로 발전 중임."



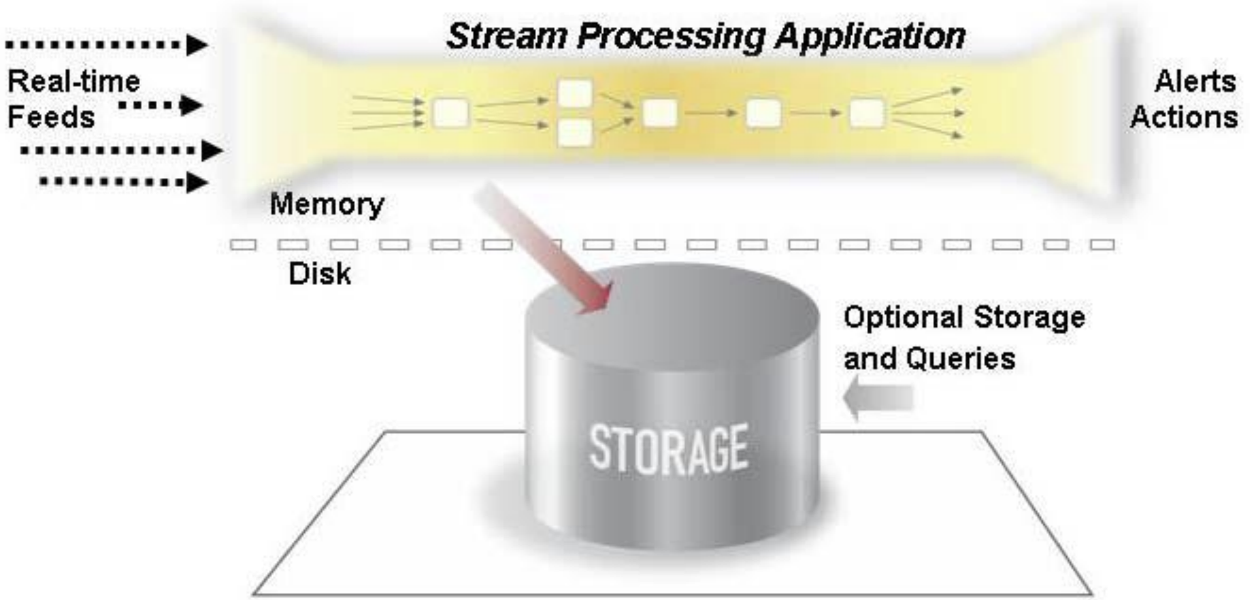
요구 사항	▪ 비정형 처리 유연성 ▪ 대규모 처리 확장성 ▪ 스트림 처리 실시간성	▪ 대용량 관리 확장성과 실시간성	▪ 정형/비정형 통합 분석 ▪ 대규모 실시간 분석
주요 기술	▪ 유연한 처리 인프라 ▪ 분산 병렬 처리 ▪ data in motion 연속 처리	▪ 분산 저장 관리 ▪ 인메모리 기반 저장 관리	▪ 비정형 데이터 분석 ▪ 빅데이터 처리, 관리 인프라와 연동

Source : ETRI

빅데이터 실시간 분석 기술

❑ Concept of stream processing

- ✓ Stream : Unbounded sequence of data
- ✓ Processing of data-in-motion
- ✓ Finite window data processing
- ✓ Continuous query processing



Source : EMC Blog posted by William Zhou Sep 2012

❑ Storm - overview

- ✓ Developed by BackType which was acquired by **Twitter**
- ✓ Lots of tools for data (i.e. batch) processing
  - – Hadoop, Pig, HBase, Hive, …
  - – None of them are real-time systems which is becoming a real requirement for businesses

Problems of MR

- ▪ Scaling is painful
- ▪ Poor fault-tolerance
- ▪ Coding is tedious

What we want

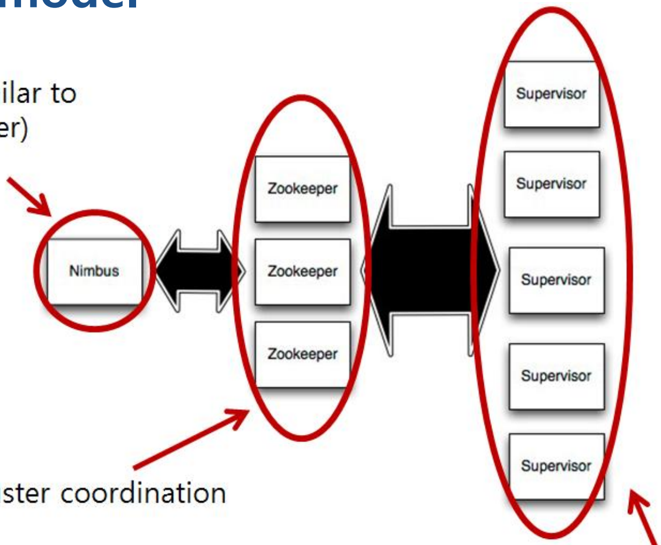
- ▪ Guaranteed data processing
- ▪ Horizontal scalability
- ▪ Fault-tolerance
- ▪ No intermediate message brokers!
- ▪ Higher level abstraction than message passing
- ▪ "Just works" !!

Storm provides real-time computation

- ▪ Scalable
- ▪ Guarantees no data loss
- ▪ Extremely robust and fault-tolerant
- ▪ Programming language agnostic

❑ Storm – architecture & stream processing model

- ➢ Storm cluster
  - – Distributed architecture as Master/Slave
  - – **Nimbus** : code distribution, task deployment, fault monitoring
  - – **Supervisor** : processing task control
  - – **Zookeeper** : cluster management



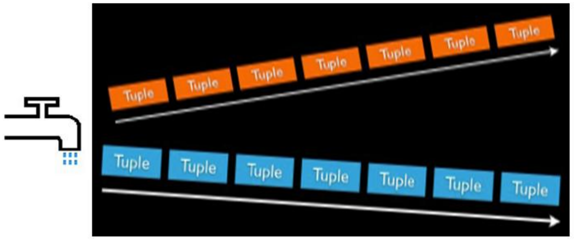
Master node (similar to Hadoop JobTracker)

Used for cluster coordination

Run worker processes

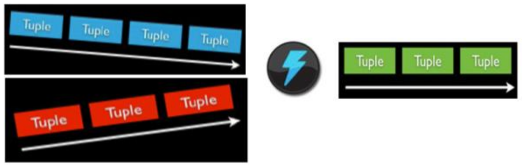
- ➢ Stream Processing model

Spouts



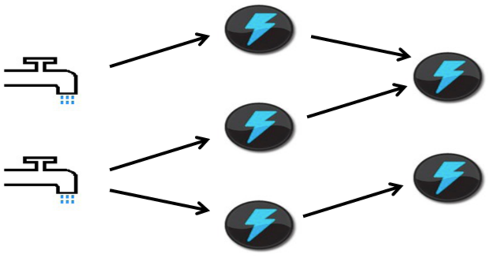
Source of streams

Bolts



Processes input streams and produces new streams:
Can implement functions such as filters, aggregation, join, etc

Topology

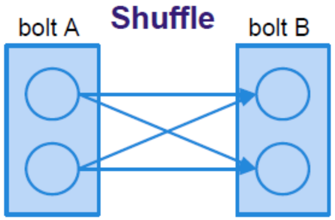


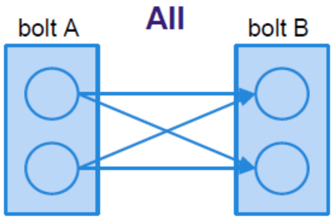
Network of spouts and bolts

❑ Storm – stream grouping

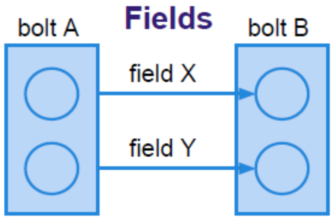
➢ <u>When a tuple is emitted which task does it go to ?</u>

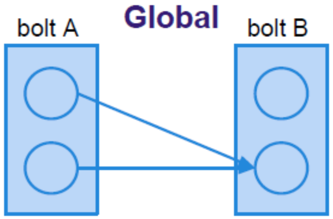
- **Shuffle grouping**
  pick a random task





- **Fields grouping**
  consistent hashing on a subset of tuple fields





- **All grouping**
  send to all tasks

- **Global grouping**
  pick task with lowest id

❑ Storm – Processing example(word count)

```
TopologyBuilder builder = new TopologyBuilder();
```

1. TopologyBuilder is used to construc topologies in Java

```
builder.setSpout("spout", new KestrelSpout(
          "kestrel.twitter.com", 22133, "sentence_queue"), 5);
```

2. Define a spout in the topology with parallelism of 5 tasks

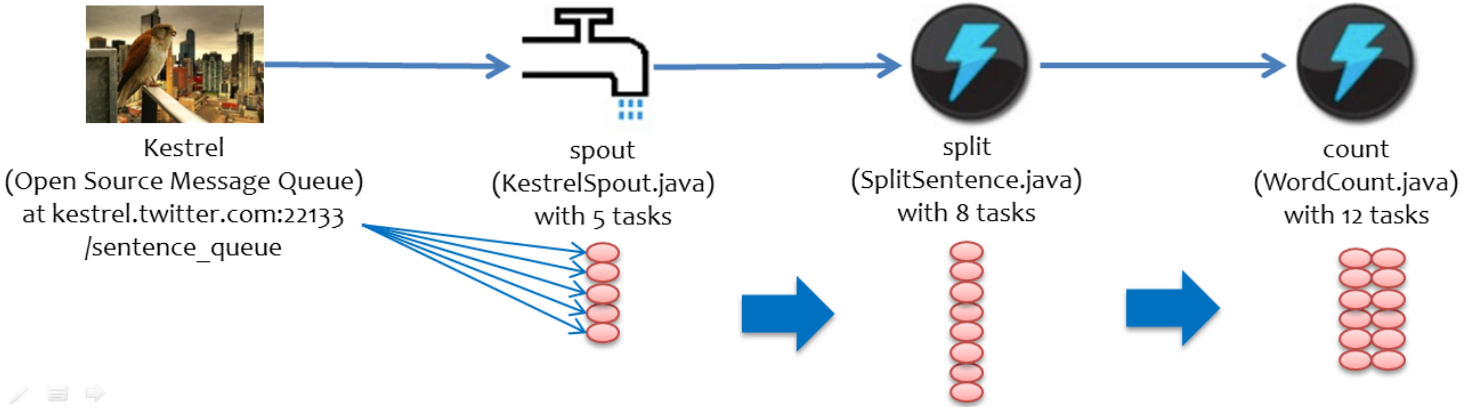
```
builder.setBolt("split", new SplitSentence(), 8)
          .shuffleGrouping("spout");
```

3. Split sentences into words with parallelism of 8 tasks

Consumer decides what data it receives and how it gets grouped

```
builder.setBolt("count", new WordCount(), 12)
          .fieldsGrouping("split", new Fields("word"));
```

3. Create a word count stream



Kestrel
(Open Source Message Queue)
at kestrel.twitter.com:22133
/sentence_queue

spout
(KestrelSpout.java)
with 5 tasks

split
(SplitSentence.java)
with 8 tasks

count
(WordCount.java)
with 12 tasks

❑ S4 - Overview

S4 distributed stream computing platform (Simple Scalable Streaming System)

"S4 is a general-purpose, **distributed**, **scalable**, fault-tolerant, pluggable platform that allows programmers to easily develop applications for **processing continuous unbounded streams of data**"

- ✓ ***Released by Yahoo!*** in October 2010
- ✓ An Apache Incubator project since September 2011
- ✓ Under the Apache 2.0 license

Proven

Deployed in production systems at Yahoo! to process thousands of search queries per second

Decentralized

All nodes are symmetric **with no centralized service** and no single point of failure.

Scalable

Throughput increases linearly as additional nodes are added to the cluster.

Extensible

Applications can easily be written and deployed using a simple API.

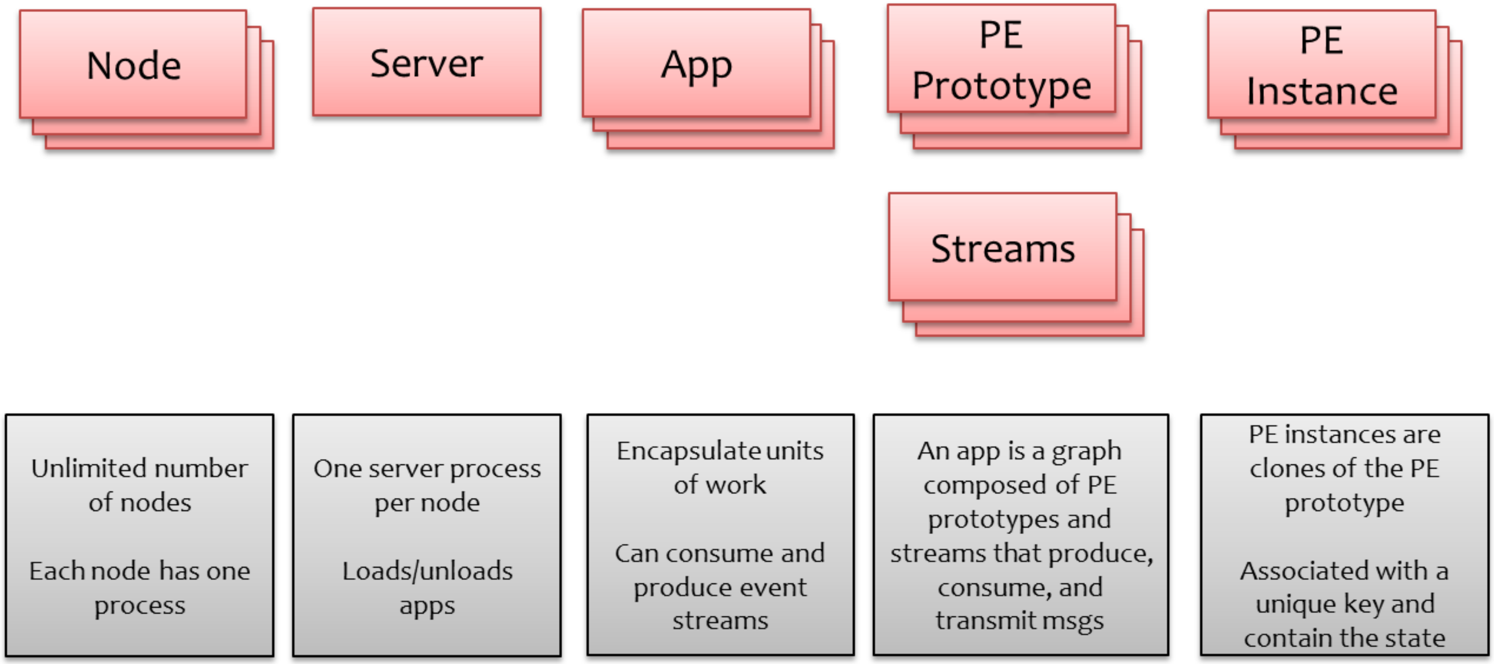
Cluster management

Using a communication layer built on top of **ZooKeeper**

Fault-tolerance

When a server in the cluster fails, a stand-by server is automatically activated to take over the tasks.

❑ S4 – Architecture



Node

Server

App

PE Prototype

PE Instance

Streams

Unlimited number of nodes

Each node has one process

One server process per node

Loads/unloads apps

Encapsulate units of work

Can consume and produce event streams

An app is a graph composed of PE prototypes and streams that produce, consume, and transmit msgs

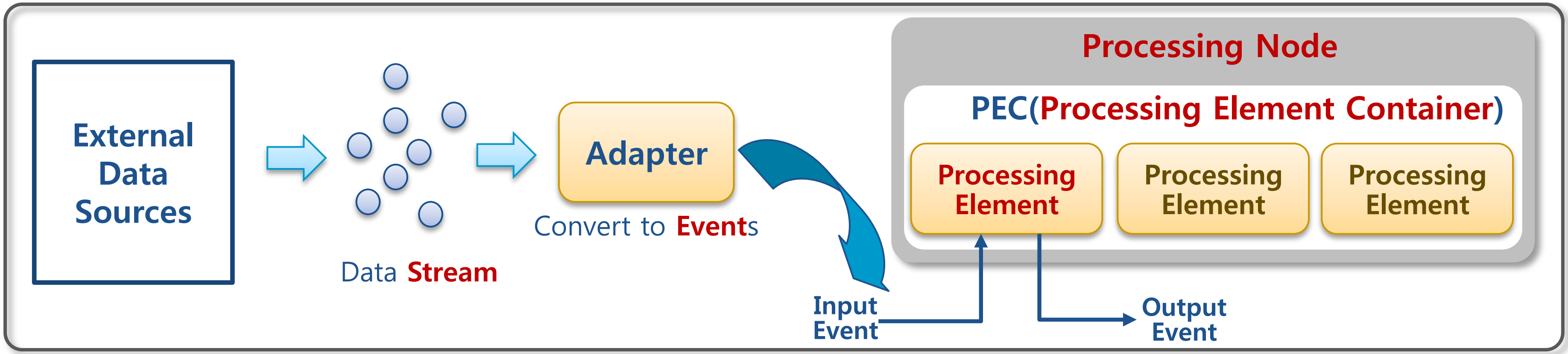
PE instances are clones of the PE prototype

Associated with a unique key and contain the state

✓ S4 is logically a message passing system

- computational units, called Processing Elements (PEs), send and receive messages (called Events)
- S4 framework defines an API which every PE must implement, and provides facilities instantiating PEs and for transporting Events

❑ S4 – Stream processing model



✓ **Stream : a sequence of “Events"**

✓ **Events**

- Arbitrary Java Objects that can be passed between PEs of the form (**K**, **A**)
  K : keyed attribute/value **A** : other attributes
- **Adapters** convert external data sources into Events that S4 can process
- Attributes of events can be accessed via getters in PEs
- Events are dispatched in **named streams**

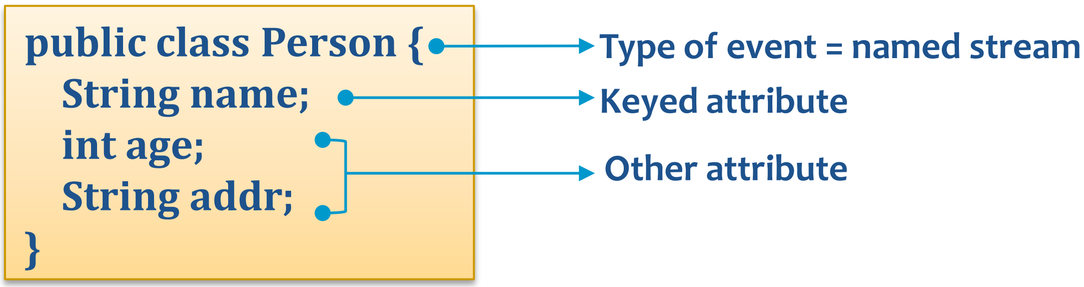
Events

```
public class Person {
   String name = “Lee”;
   int age = 30;
   String addr =
“Daejeon”;
}
```

❑ S4 – Stream processing model

✓ PE(Processing Element)

- **Basic computational units** in S4
- Consume events and can in turn emit new events and update their state
- Each instance of a PE is uniquely identified by four components:
  - its **functionality** as defined by a PE class and associated configuration,
  - the **named stream** that it consumes,
  - the **keyed attribute** in those events, and
  - the **value** of the keyed attribute in events which it consumes
- Every PE consumes exactly those events which correspond to the value on which it is keyed
- A PE is instantiated for each value of the key attribute
- This instantiation is performed by the platform



```
public class Person {
   String name;
   int age;
   String addr;
}
```

Type of event = named stream

Keyed attribute

Other attribute

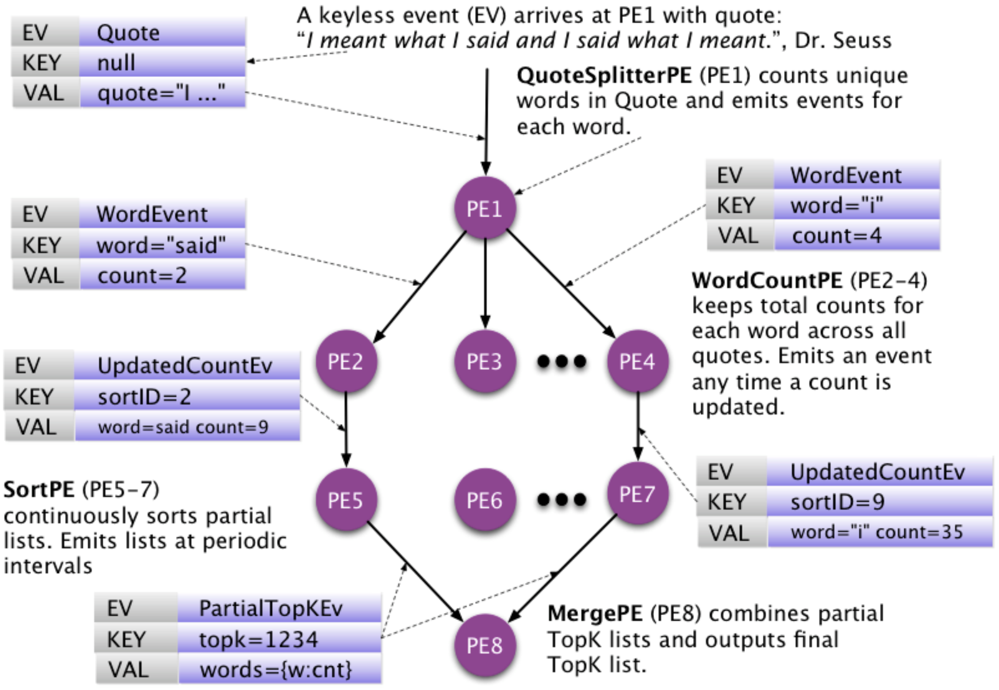
❑ S4 – Stream processing model

- ✓ Processing Node (PN)
  - Logical hosts to PEs
  - Responsible for listening to events, executing operations on the incoming events, dispatching events with the assistance of the communication layer, and emitting output events
  - S4 : route each event to PNs based on a hash function of the values of all known keyed attributes in that event
  - Event Listener : pass incoming events to the PEC
  - PEC : invoke the appropriate PEs in the appropriate order
    - Every keyless PE is instantiated once per PN
    - Only one PE prototype exists in a PN

- ✓ PE Container (PEC)
  - Holds all PE instances, including the PE prototypes
  - Responsible for routing incoming events to the appropriate PE instances

❑ S4 – processing example

- ✓ Word count example



A keyless event (EV) arrives at PE1 with quote:
"*I meant what I said and I said what I meant.*", Dr. Seuss

EV	Quote
KEY	null
VAL	quote="I ..."

QuoteSplitterPE (PE1) counts unique words in Quote and emits events for each word.

EV	WordEvent
KEY	word="said"
VAL	count=2

EV	WordEvent
KEY	word="i"
VAL	count=4

WordCountPE (PE2-4) keeps total counts for each word across all quotes. Emits an event any time a count is updated.

EV	UpdatedCountEv
KEY	sortID=2
VAL	word=said count=9

EV	UpdatedCountEv
KEY	sortID=9
VAL	word="i" count=35

SortPE (PE5-7) continuously sorts partial lists. Emits lists at periodic intervals

EV	PartialTopKEv
KEY	topk=1234
VAL	words={w:cnt}

MergePE (PE8) combines partial TopK lists and outputs final TopK list.

PE ID	PE Name	Key Tuple
PE1	QuoteSplitterPE	null
PE2	WordCountPE	word="said"
PE4	WordCountPE	word="i"
PE5	SortPE	sortID=2
PE7	SortPE	sortID=9
PE8	MergePE	topK=1234

빅데이터 실시간 분석 기술

❑ Twitter Strom vs Yahoo! S4

분류	Twitter Storm	Yahoo! S4
License	▪ Eclipse Public License	▪ Apache 2.0
시스템 구조	▪ Master/Slave	▪ Symmetric
연속 처리 모델	▪ 튜플 ▪ 태스크간 관계 DAG	▪ (Keys, attribute) 튜플 ▪ 이벤트 기반 Actor
용어	▪ Bolt	▪ Processing Element
Window	–	–
스트림 전달	▪ ZeroMQ	▪ Transport Protocol pluggable
태스크 노드 배치	▪ Master에서 결정	▪ 키 값에 의해 결정
입력 스트림 분배	▪ Shuffle, field, all, global, direct	▪ 이벤트 type & key
장애 대처	▪ 태스크 재배치 & 실행 ▪ Guaranteed message processing	▪ 태스크 재배치 & 실행

4. 사례 연구

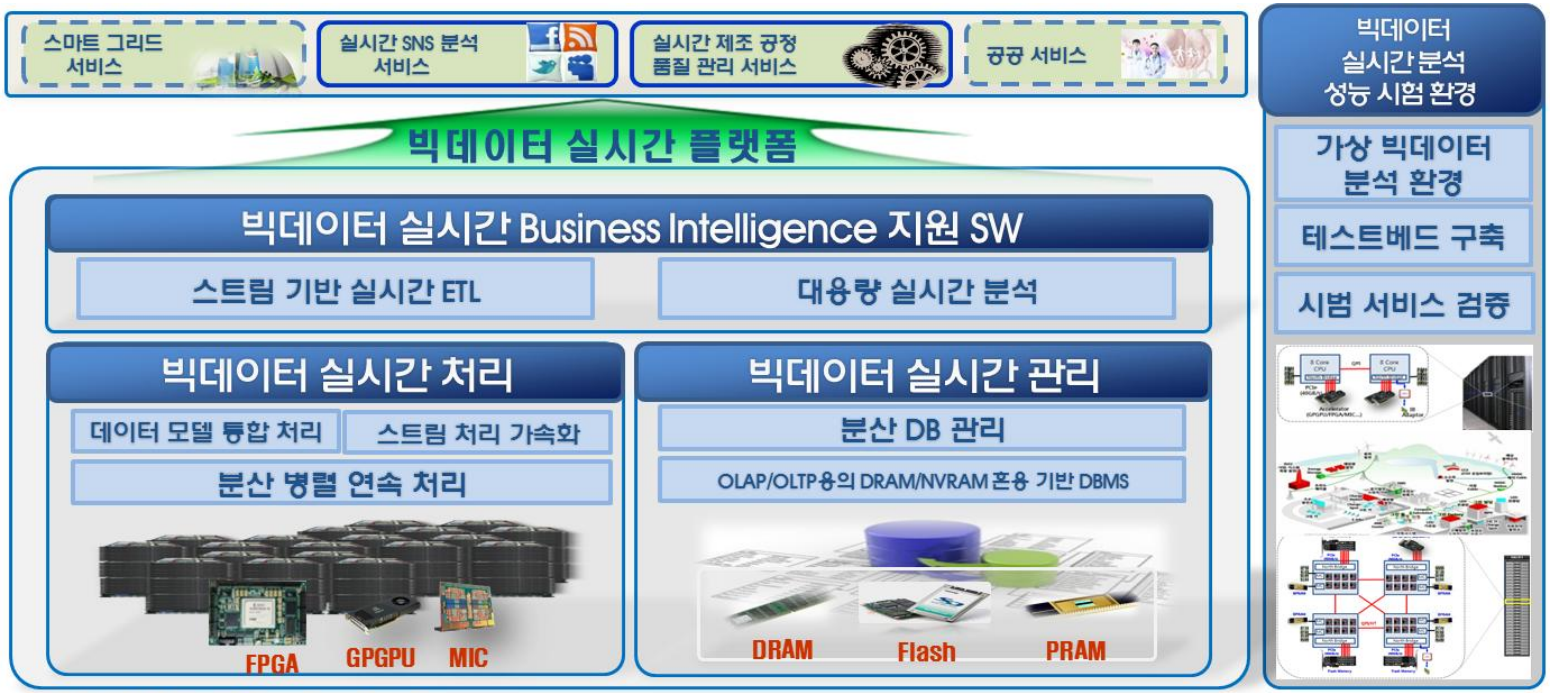
①빅데이터 실시간 플랫폼 개발 사례
②빅데이터 실시간 플랫폼 활용 사례
③In-Memory computing for Big data

빅데이터 실시간 플랫폼 개발 사례

❑ 프로젝트 : 차세대 메모리 기반의 빅데이터 분석·관리 소프트웨어 원천기술 개발
(ETRI, 2012.6 ~ 2017.5)

● 목표 : 빅데이터 실시간 처리, 관리 및 분석 플랫폼 핵심 기술 개발

- 성능가속장치 최적 활용을 통한 초당 1GB 급 의 실시간 스트림 처리
- 차세대메모리 활용을 통한 100 TB급 확장성, DRAM대비 성능저하 3% 이내 인 실시간 데이터 관리



스마트 그리드 서비스

실시간 SNS 분석 서비스

실시간 제조 공정 품질 관리 서비스

공공 서비스

빅데이터 실시간 플랫폼

빅데이터 실시간 Business Intelligence 지원 SW

스트림 기반 실시간 ETL

대용량 실시간 분석

빅데이터 실시간 처리

데이터 모델 통합 처리

스트림 처리 가속화

분산 병렬 연속 처리

FPGA

GPGPU

MIC

빅데이터 실시간 관리

분산 DB 관리

OLAP/OLTP용의 DRAM/NVRAM 혼용 기반 DBMS

DRAM

Flash

PRAM

빅데이터 실시간분석 성능 시험 환경

가상 빅데이터 분석 환경

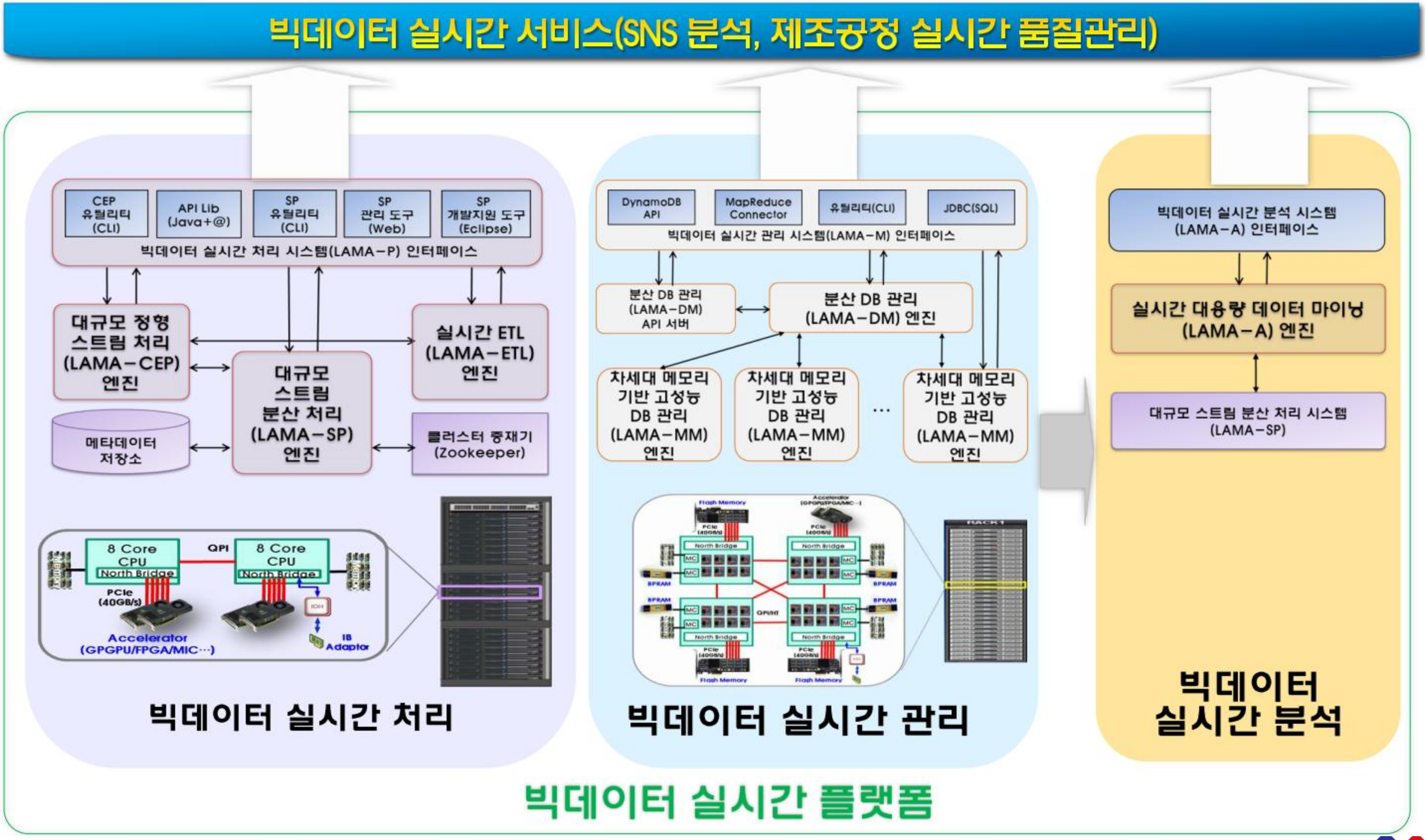
테스트베드 구축

시범 서비스 검증

❖ ETL: Extract, Transform, Load
❖ NVRAM : Non Volatile RAM

빅데이터 실시간 플랫폼 개발 사례

❑ 빅데이터 실시간 분석 플랫폼 구성도



빅데이터 실시간 서비스(SNS 분석, 제조공정 실시간 품질관리)

CEP 유틸리티 (CLI)
API Lib (Java+@)
SP 유틸리티 (CLI)
SP 관리 도구 (Web)
SP 개발지원 도구 (Eclipse)
빅데이터 실시간 처리 시스템(LAMA-P) 인터페이스

대규모 정형 스트림 처리 (LAMA-CEP) 엔진
대규모 스트림 분산 처리 (LAMA-SP) 엔진
실시간 ETL (LAMA-ETL) 엔진
메타데이터 저장소
클러스터 중재기 (Zookeeper)

빅데이터 실시간 처리

DynamoDB API
MapReduce Connector
유틸리티(CLI)
JDBC(SQL)
빅데이터 실시간 관리 시스템(LAMA-M) 인터페이스

분산 DB 관리 (LAMA-DM) API 서버
분산 DB 관리 (LAMA-DM) 엔진
차세대 메모리 기반 고성능 DB 관리 (LAMA-MM) 엔진
차세대 메모리 기반 고성능 DB 관리 (LAMA-MM) 엔진
…
차세대 메모리 기반 고성능 DB 관리 (LAMA-MM) 엔진

빅데이터 실시간 관리

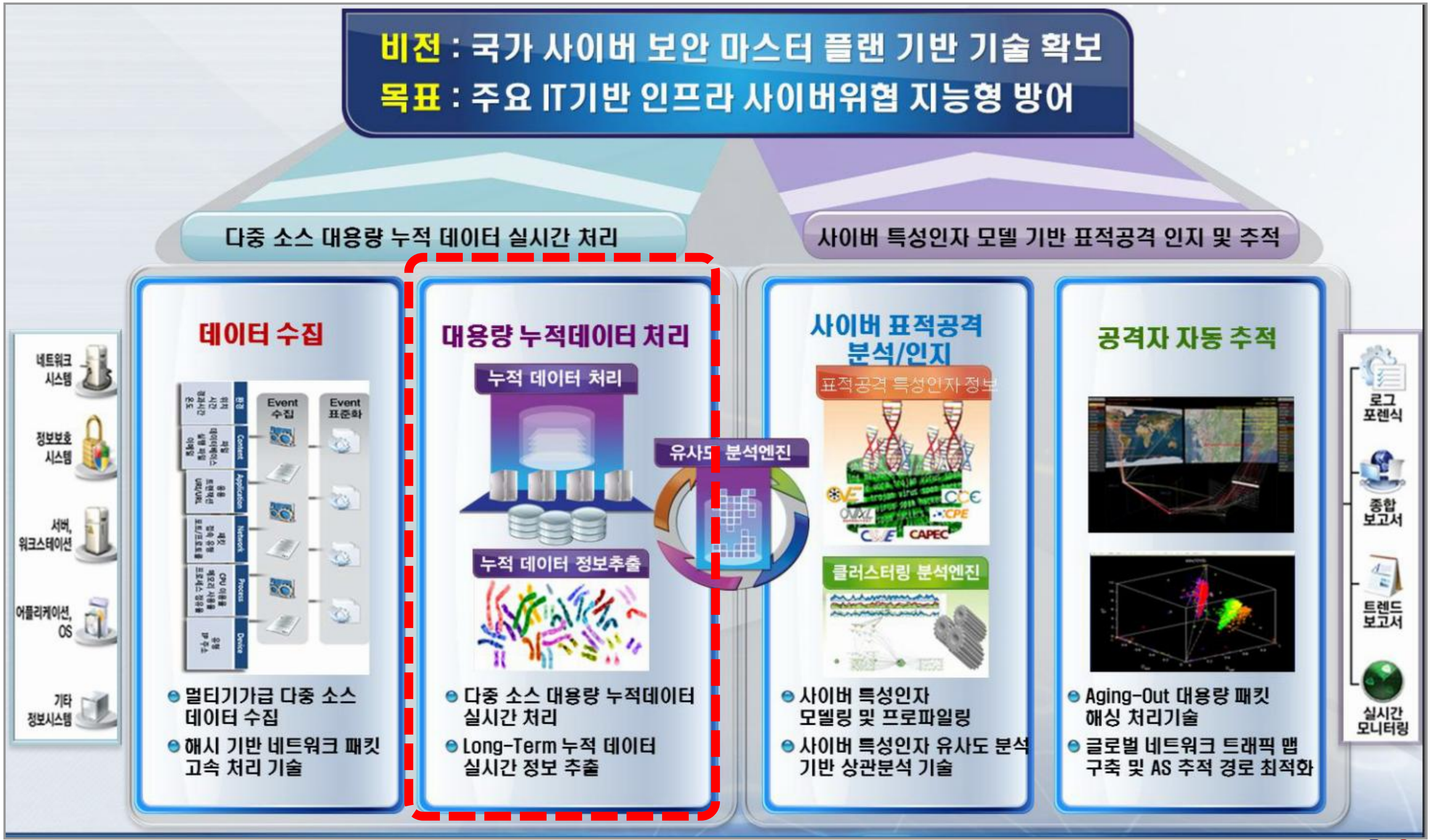
빅데이터 실시간 분석 시스템 (LAMA-A) 인터페이스
실시간 대용량 데이터 마이닝 (LAMA-A) 엔진
대규모 스트림 분산 처리 시스템 (LAMA-SP)

빅데이터 실시간 분석

빅데이터 실시간 플랫폼

빅데이터 실시간 플랫폼 활용 사례

❑ 프로젝트 : 사이버 표적공격 인지 및 추적 기술 개발 (ETRI, 2013.3 ~ 2017.2)



비전 : 국가 사이버 보안 마스터 플랜 기반 기술 확보

목표 : 주요 IT기반 인프라 사이버위협 지능형 방어

다중 소스 대용량 누적 데이터 실시간 처리

사이버 특성인자 모델 기반 표적공격 인지 및 추적

데이터 수집

- 멀티기가급 다중 소스 데이터 수집
- 해시 기반 네트워크 패킷 고속 처리 기술

대용량 누적데이터 처리

- 다중 소스 대용량 누적데이터 실시간 처리
- Long-Term 누적 데이터 실시간 정보 추출

사이버 표적공격 분석/인지

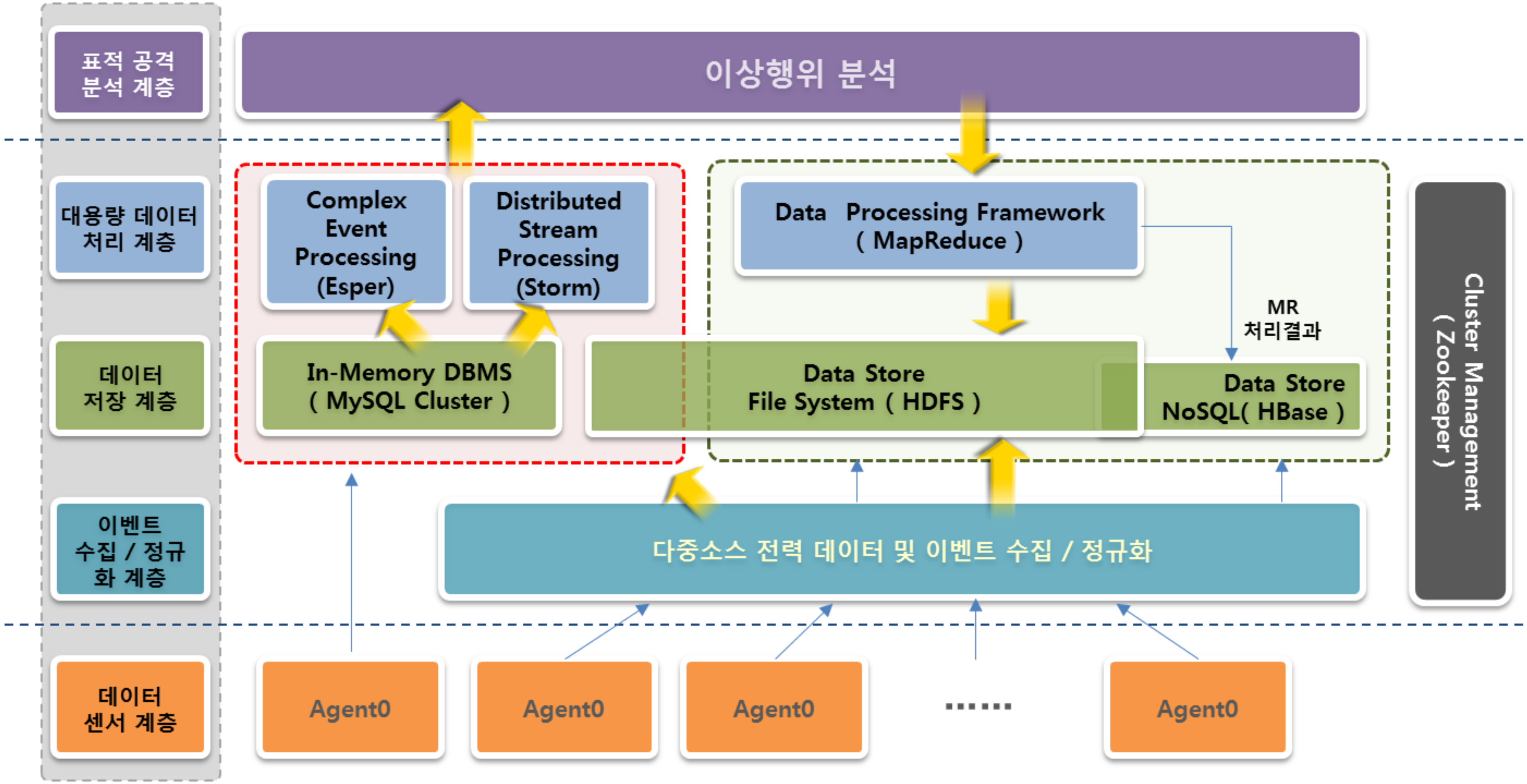
- 사이버 특성인자 모델링 및 프로파일링
- 사이버 특성인자 유사도 분석 기반 상관분석 기술

공격자 자동 추적

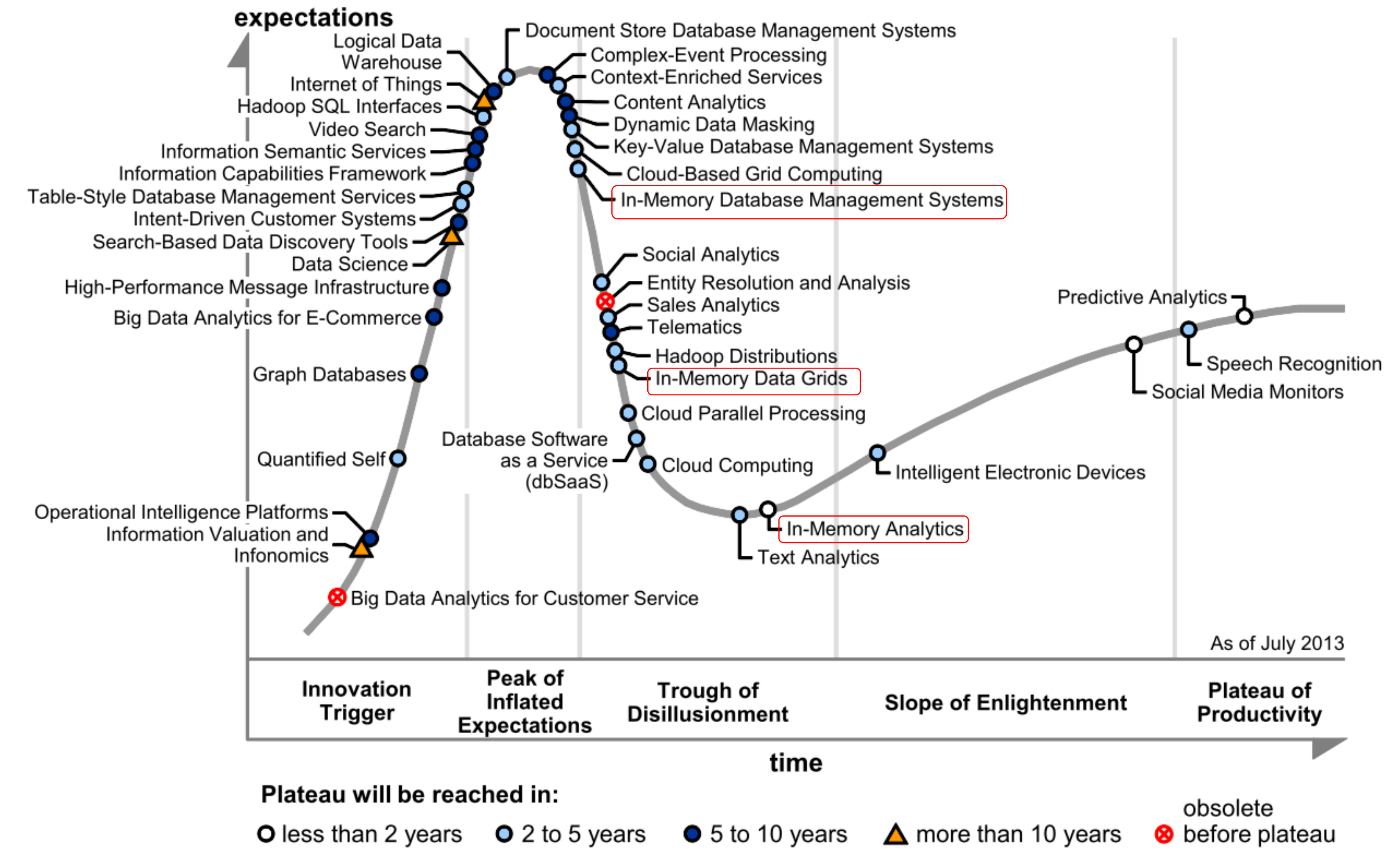
- Aging-Out 대용량 패킷 해싱 처리기술
- 글로벌 네트워크 트래픽 맵 구축 및 AS 추적 경로 최적화

빅데이터 실시간 플랫폼 활용 사례

❑ 대용량 누적 데이터 및 실시간 데이터 처리 플랫폼 구성도 (오픈 소스 활용)


표적 공격 분석 계층
이상행위 분석
대용량 데이터 처리 계층
Complex Event Processing (Esper)
Distributed Stream Processing (Storm)
Data Processing Framework (MapReduce)
MR 처리결과
Cluster Management (Zookeeper)
데이터 저장 계층
In-Memory DBMS (MySQL Cluster)
Data Store File System (HDFS)
Data Store NoSQL(HBase)
이벤트 수집 / 정규화 계층
다중소스 전력 데이터 및 이벤트 수집 / 정규화
데이터 센서 계층
Agent0
Agent0
Agent0
....
Agent0

In-Memory computing for Big Data



Source: Gartner (July 2013)

[Hype Cycle for Big Data]

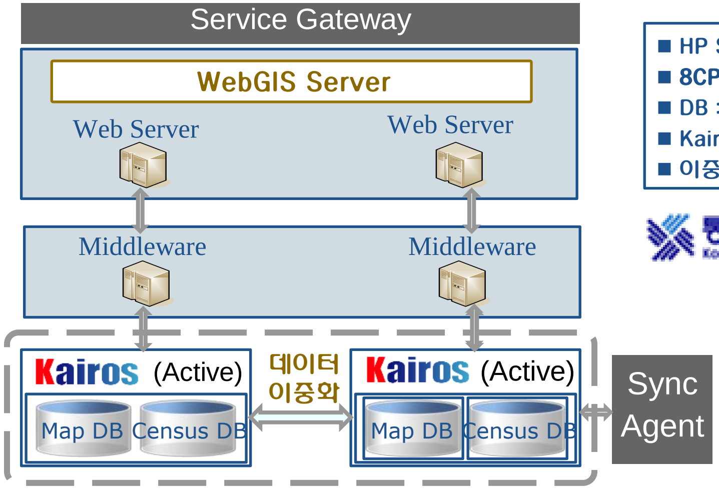
❑ 적용 사례 1 : 실시간 공간 통계 분석/제공 시스템 (통계청)

통계청 통계 네비게이터 시스템

1) 국민 생활과 밀접한 **상세지역 생활통계정보를 지역별 공간 정보와 연계하여** 웹 기반 대국민 서비스를 제공하는 **공간 빅데이터 시스템**으로, **Kairos 적용을 통한 고속의 Web 기반 통계 GIS 서비스 실현**

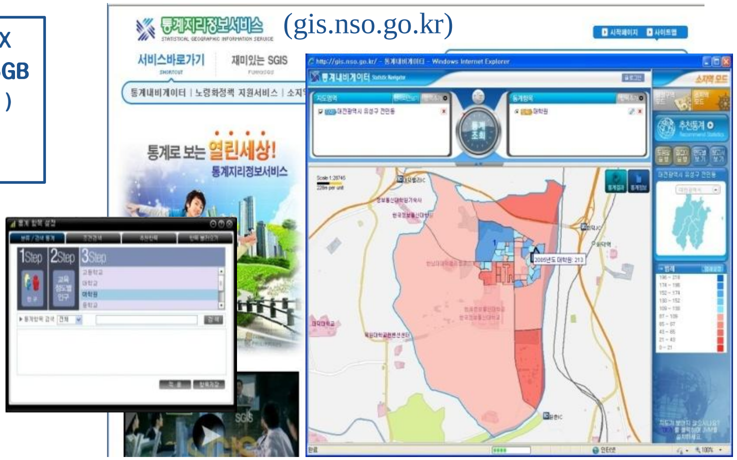
2) 기존 외산 소프트웨어를 기반으로 구축되었던 시스템을 국산 기술과 국산 웹 기술 기반의 신규 시스템으로 대체하여 성공한 사례임

3) 데이터의 실시간 갱신을 통한 서비스의 신뢰성 확보



- ■ HP Superdome : HP-UX
- ■ 8CPU x Quad core, 256GB
- ■ DB : 100GB (2012연재)
- ■ Kairos Spatial 4.8
- ■ 이중와를 통한 HA 구연

통계청
Korea National Statistical Office



In-Memory computing for Big Data

❑ 적용 사례 2 : 교통정보 실시간 수집/가공/분석 시스템 (현대/기아 자동차)

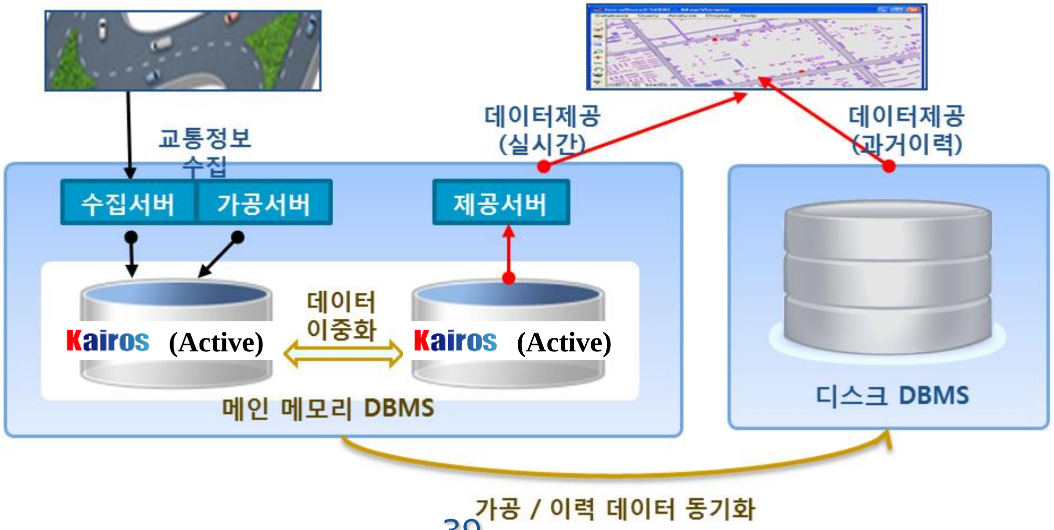
현대/기아 자동차 교통정보시스템 고도화 구축

1) 현대/기아 자동차의 교통정보 빅데이터 처리에 디스크DBMS의 성능한계로 In-Memory DBMS를 도입하여 운영되고 있는 **빅데이터 분야**의 대표적인 성공사례

2) 현대/기아 자동차 본사의 In-Memory DBMS의 첫 적용사례

3) 가공시간 단축으로 기존 대비 더 정확한 교통정보 제공을 통해 양질의 서비스를 제공함

4) 차량의 단말기(카드, 내비게이션 등)를 이용한 교통제공서비스 연동 가능



REAL TIME TECH

Thank You !

www.realtimetech.co.kr