Big Data

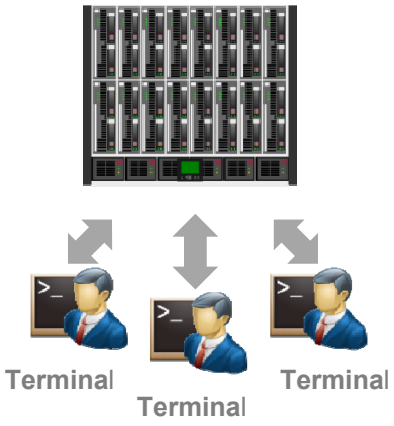
IT & Society

1960~ | 1990~ | 2005~ | 2012 | 2020~

Industrialized Society

Centralized Computing

Main Frames

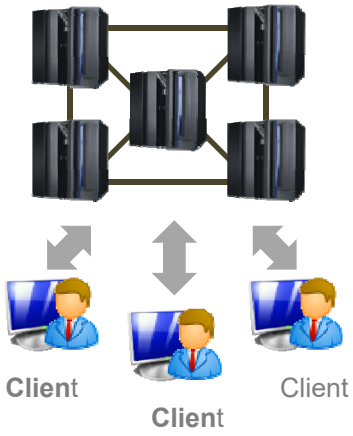


- Digitalization
- Mass Production
- Pareto Principle
- MIS

Informatized Society

Distributed Computing

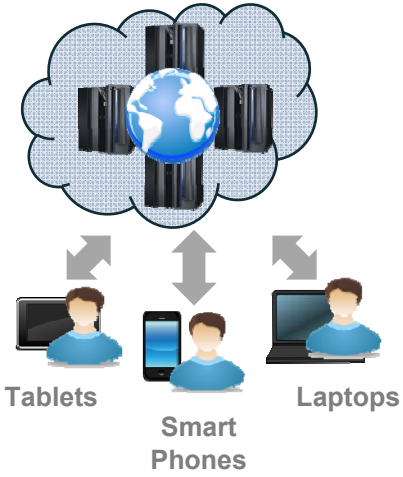
Internet



- High Speed Network
- Multi-item Small Prod.
- Long Tail
- eGov, ERP

Cloud Computing

Smart & Mobile
Virtualizaiton



- Smart Phones & TVs
- Personalizaton
- IT Fusion
- Platform Services

Optimized Society ?

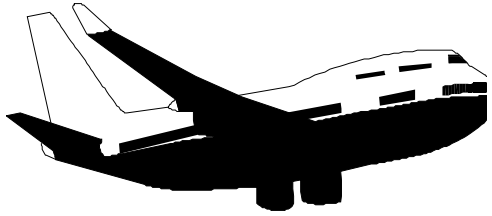
Enhancement

- Big Data
- IOT, M2M
- Brain Science
- ...

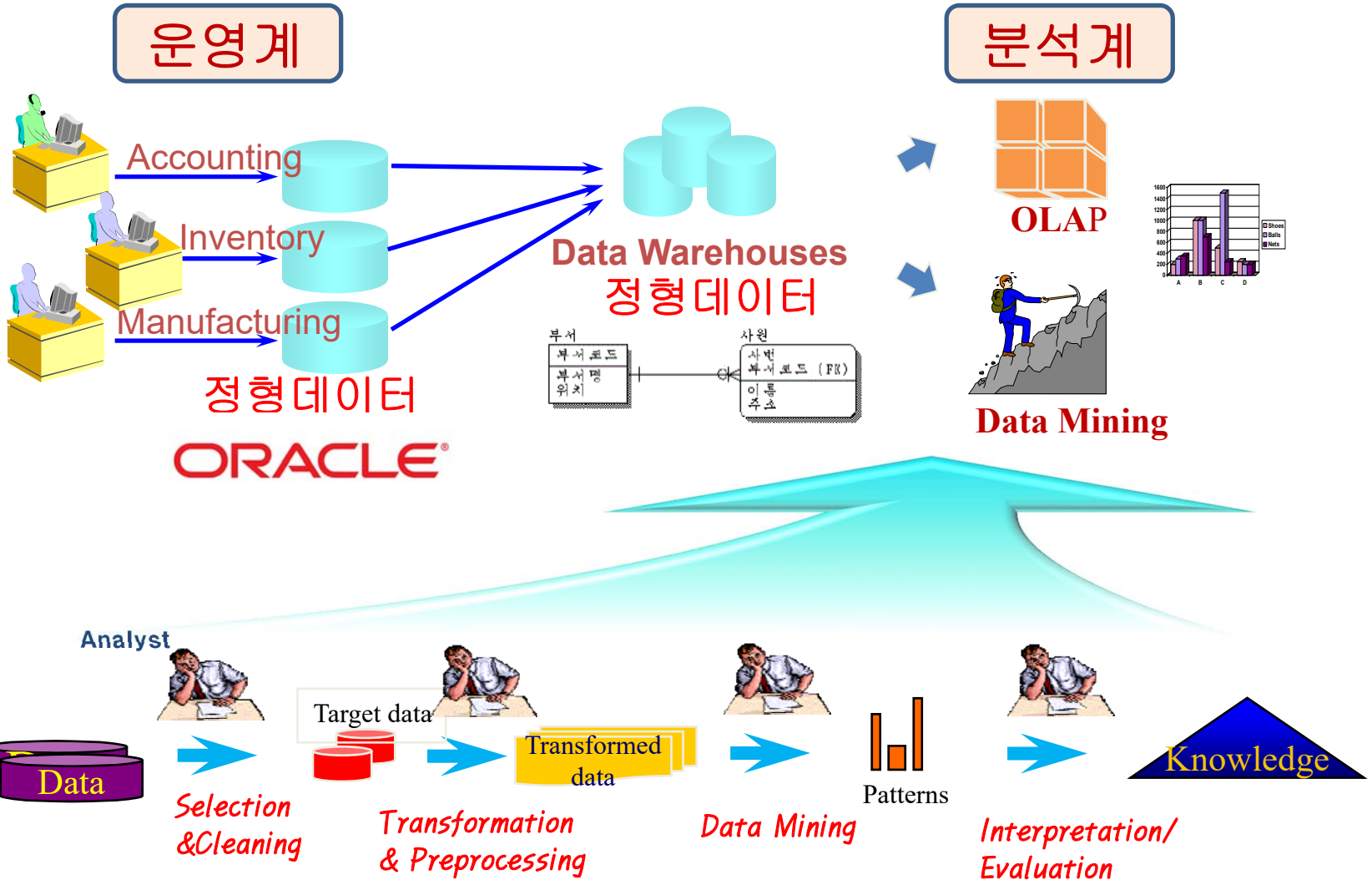
Front-Office Systems

OLTP – Online transaction processing

- Handle a business's daily activities & commerce
- ATM, airline reservation, catalog order, supermarket (bar-code data)
- Based on well-defined business & technical requirement
- Large volumes of simple transactions
- Rigid specs
- Maintained by IT professionals
- Process oriented

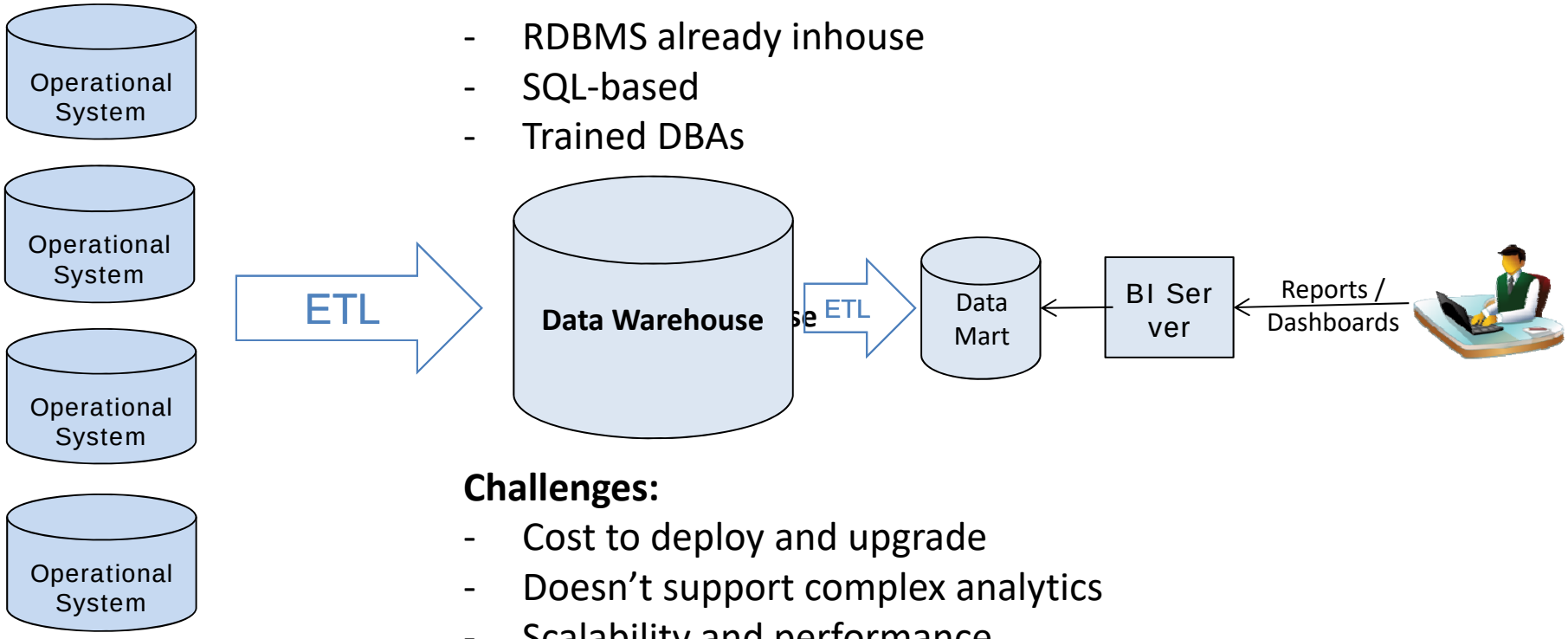


데이터분석- The first wave(1995)



Three Big Data platforms (systems)

1. **General purpose RDBMS** : First generation DW



Benefits:

- RDBMS already inhouse
- SQL-based
- Trained DBAs

Challenges:

- Cost to deploy and upgrade
- Doesn't support complex analytics
- Scalability and performance

Three Big Data platforms (systems)

2. Analytical platforms

1010data
Aster Data (Teradata)
Calpont
Datallegro (Microsoft)
Exasol
Greenplum (EMC)
IBM SmartAnalytics
Infobright
Kognitio
Netezza (IBM)
Oracle Exadata
Paraccel
Pervasive
Sand Technology
SAP HANA
Sybase IQ (SAP)
Teradata
Vertica (HP)

Purpose-built database management systems designed explicitly for query processing and analysis that provides dramatically higher price/performance and availability compared to general purpose solutions.

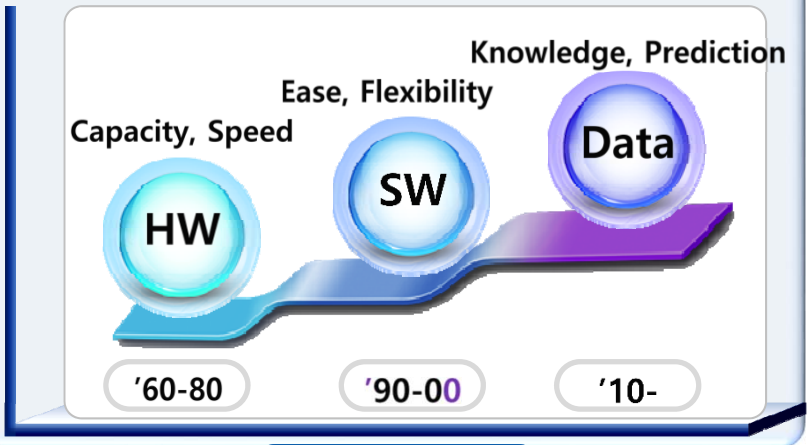
<u>Deployment Options</u>

-Software only (Paraccel, Vertica)
-Appliance (SAP, Exadata, Netezza)
-Hosted(1010data, Kognitio)

IT & Society

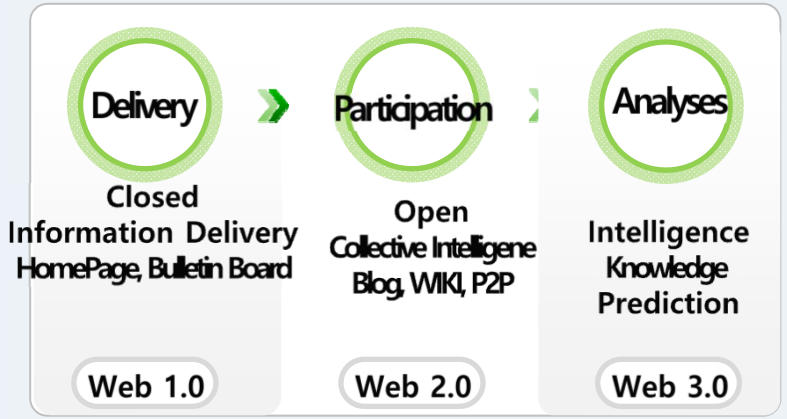
IT

- Key IT Technology

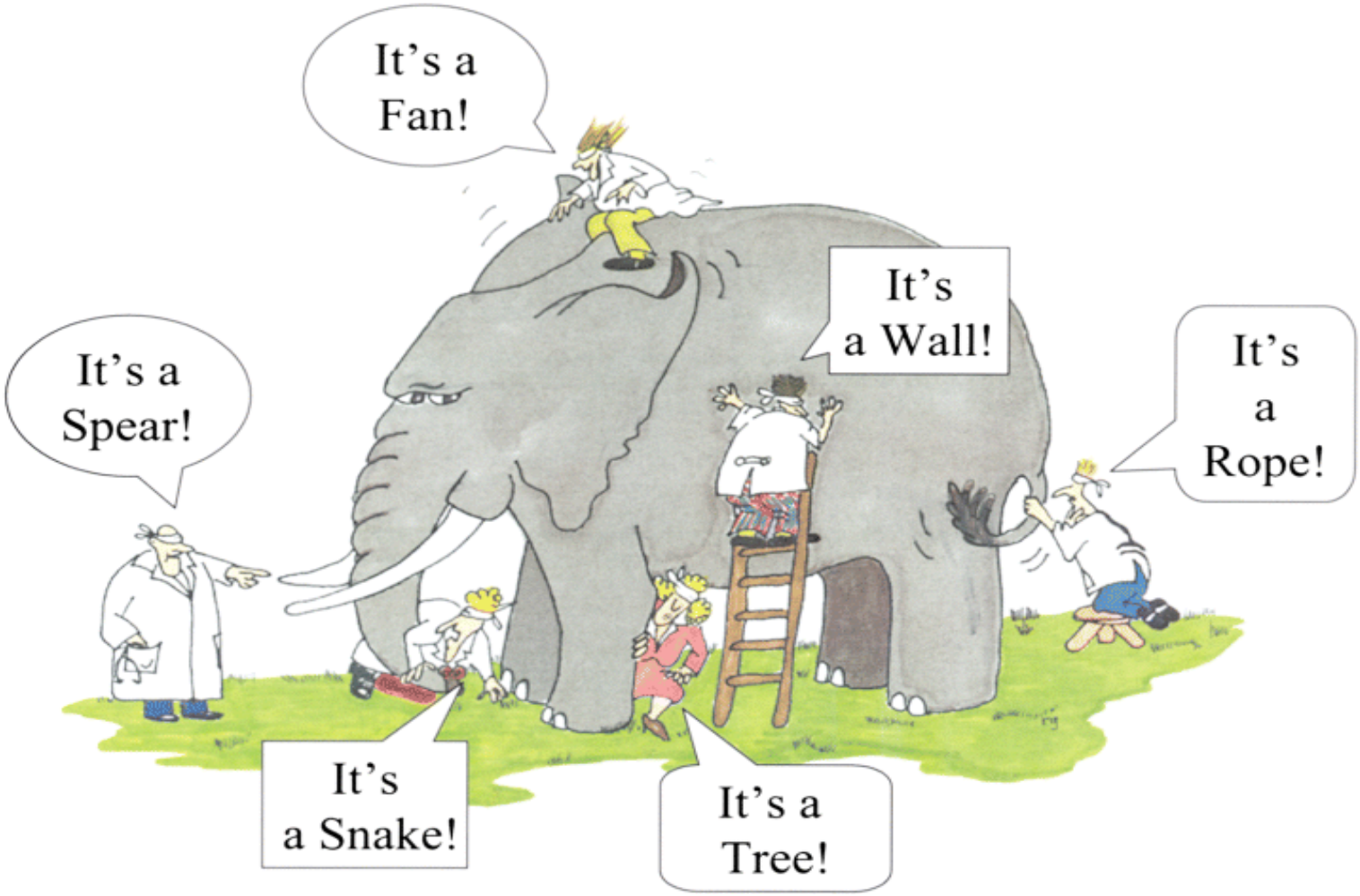


Web

- Key Methodology



Big Data - Definition?



데이터분석- The first wave(1995)

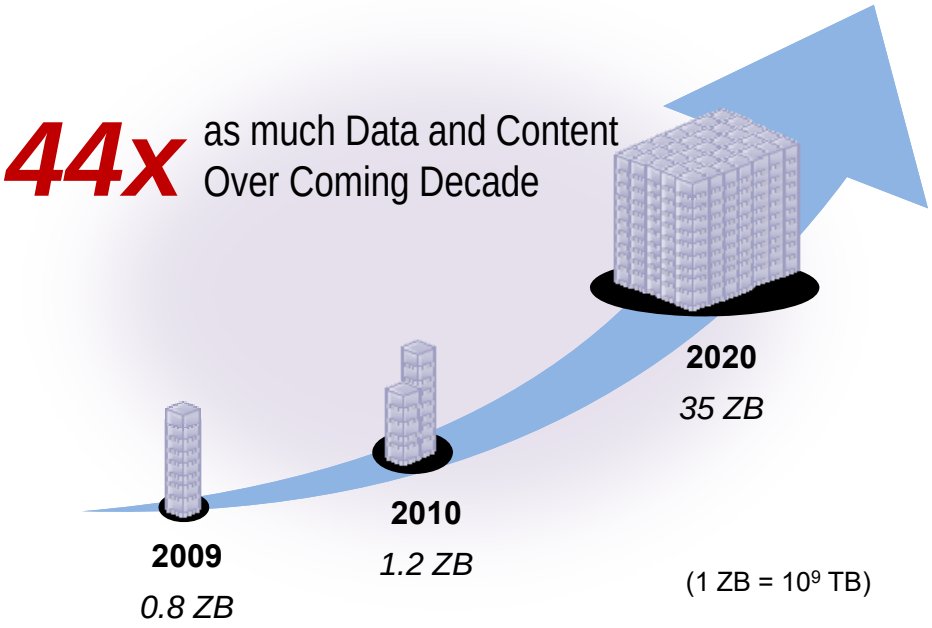
Monitoring

Prediction

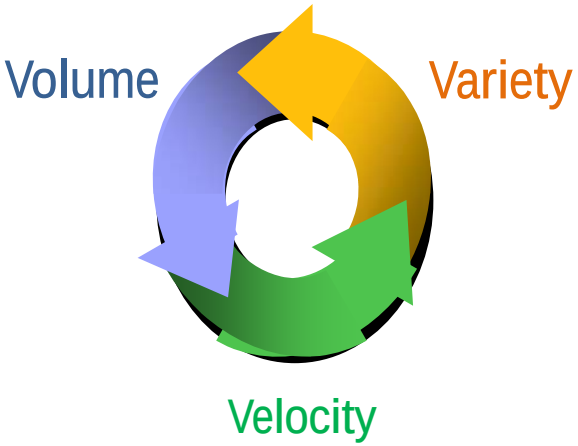
=> 데이터마이닝 & 인공지능 : 1997년 가트너 5대 핵심 기술

빅데이터 - The second wave(2011)

❖ 기존의 방식으로 저장/관리/분석하기 어려울 정도로 큰 규모의 자료



(1 ZB = 10^9 TB)



트위터

1억 1천만 tweet/일

휴대폰 발신/GPS

수억 건/일

웹로그

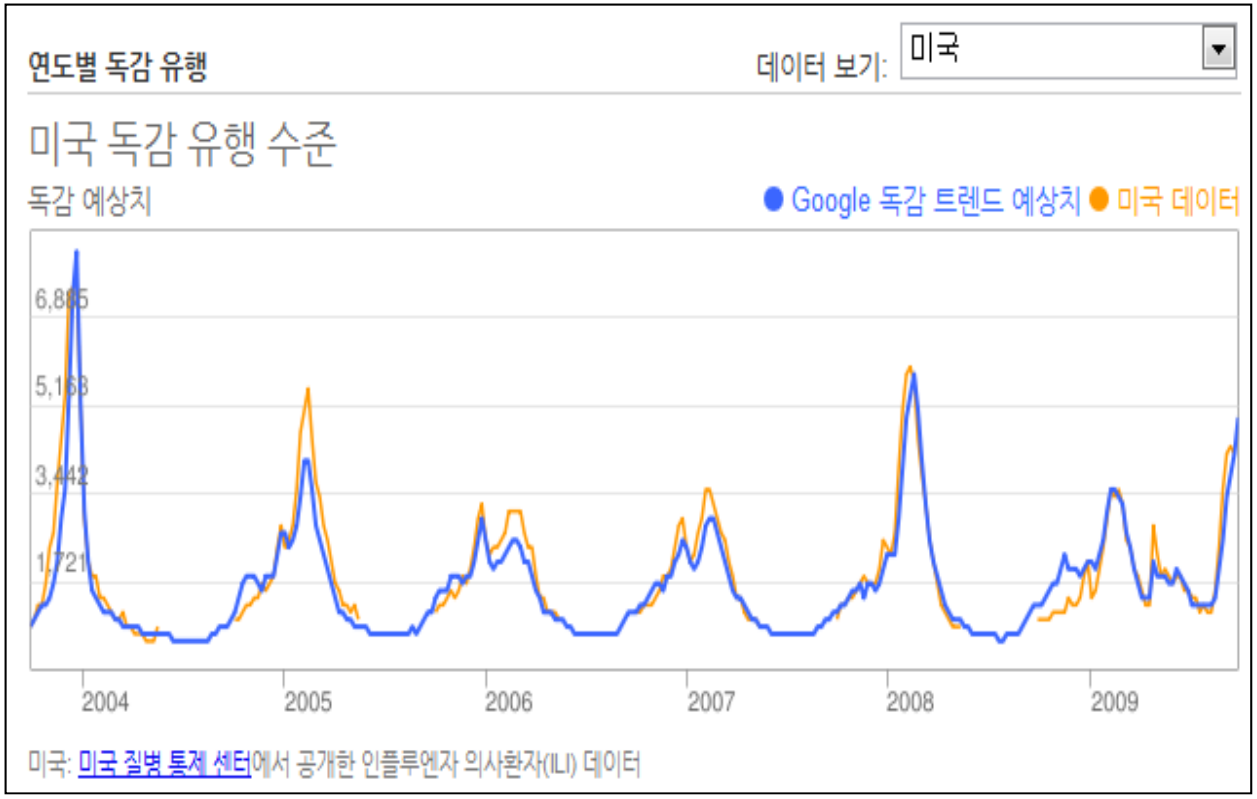
수십~수백 GB/일

센서데이터

빅데이터 - The second wave

❖ Google Trend

- Regional Flu infection rates vs Google key words

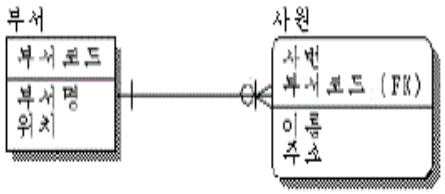


(http://www.google.org/flutrends/intl/ko/)

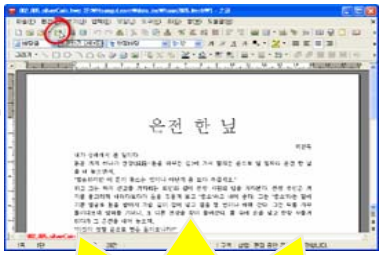
빅데이터 - The second wave

정형 데이터분석

빅데이터 분석



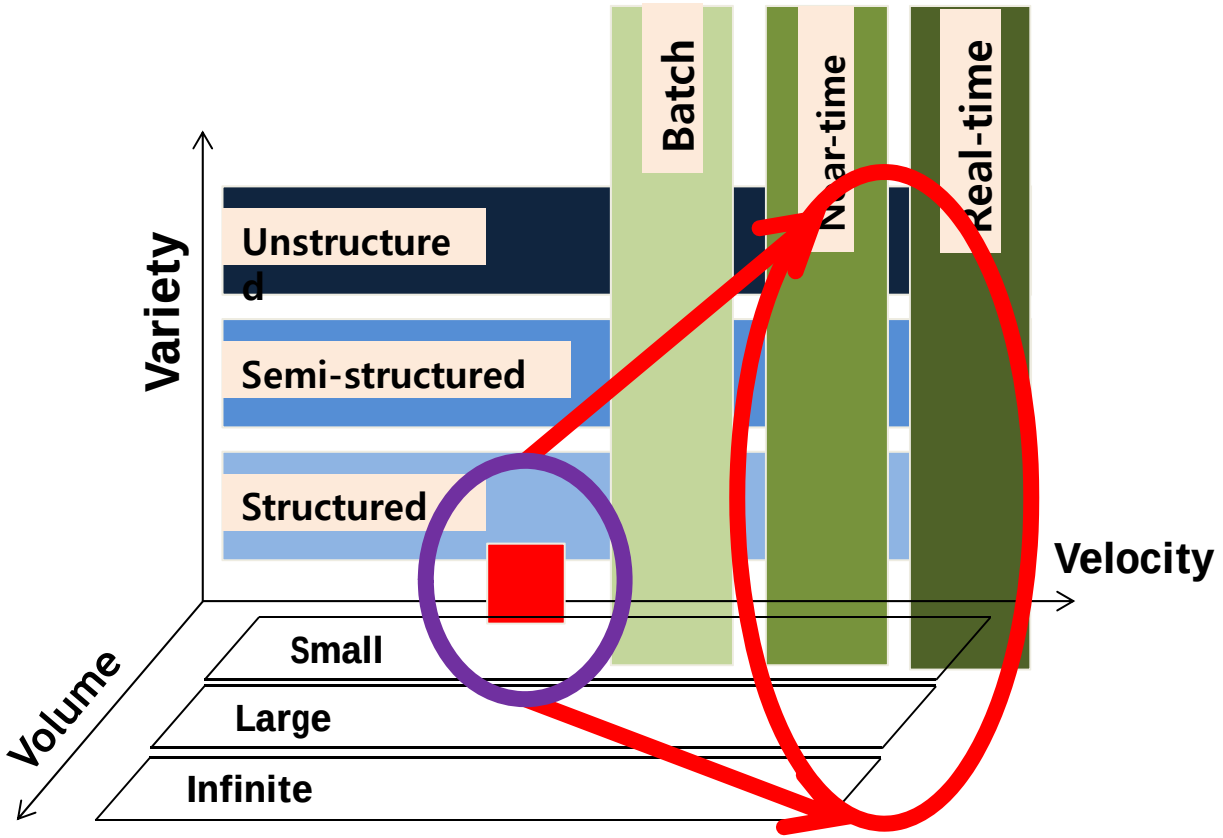
Variety



Volume

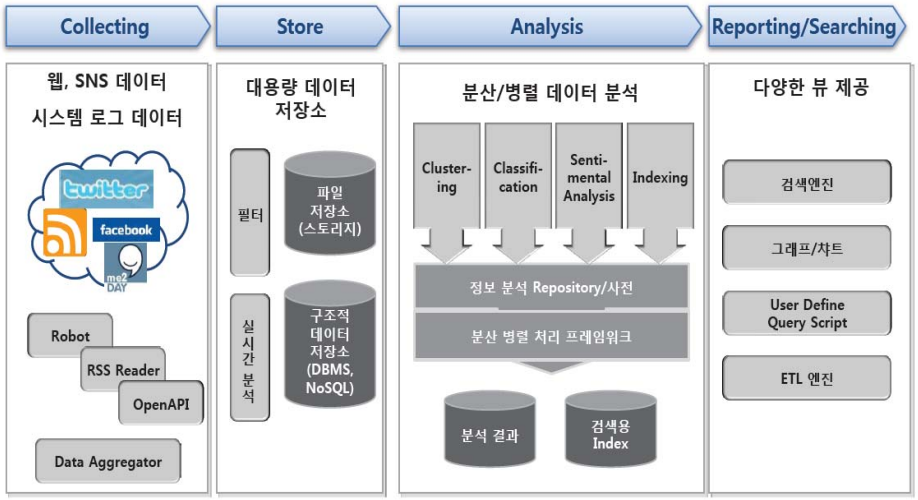
Velocity

빅데이터 - The second wave



빅데이터 3대 요소 - 분산병렬처리 기술

Hadoop



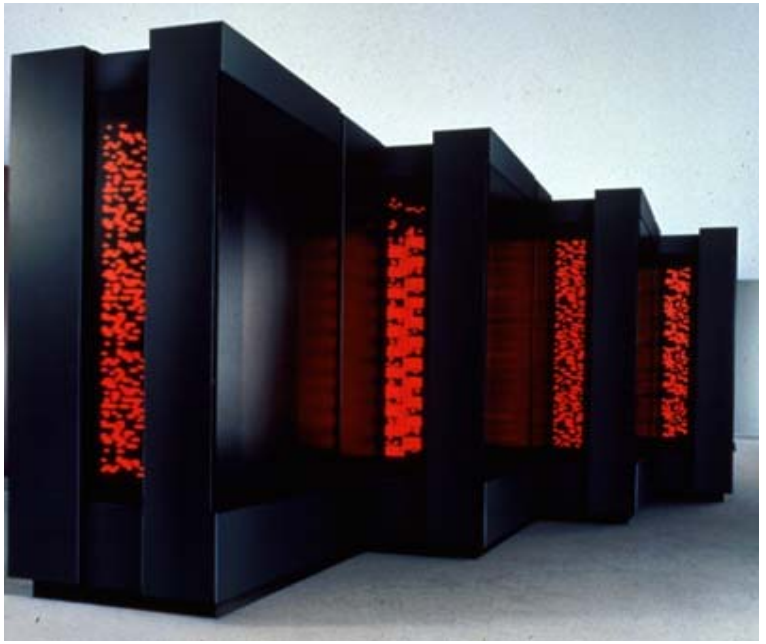
NoSQL

빅데이터 3대 요소 - 분산병렬처리 기술

Parallel Processing - MPP

Thinking Machine Corporation
Connection Machine nCube2 (1989)

- 64, 128,...,1024 nodes



ancestor(x,y) :- parent(x,y)
ancestor(x,y) :- ancestor(x,z) and parent(z,y)

• An Object-based Query Evaluation Scheme for Deductive Databases in Massively Parallel Computing Environment (International Conference on Data Engineering ICDE, 1989)

Hadoop

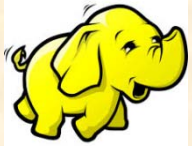
- Hadoop is a Platform which enables you to store and analyze large volumes of data.
- Hadoop is batch oriented (high throughput and low latency) and strongly consistent (data is always available).
- Hadoop is best utilized for:
  - Large scale batch analytics
  - Unstructured or semi-structured data
  - Flat files
- Hadoop is comprised of two major subsystems
  - HDFS (File System)
  - Map Reduce

Basic Features: HDFS

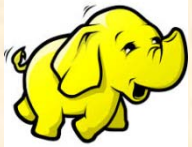
- Highly fault-tolerant
- High throughput
- Suitable for applications with large data sets
- Streaming access to file system data
- Can be built out of commodity hardware
- HDFS provides Java API for applications to use.
- A HTTP browser can be used to browse the files of a HDFS instance.

Basic Features: HDFS

- Masters

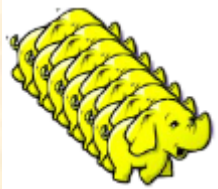


Name Node
(메타데이터 관리)

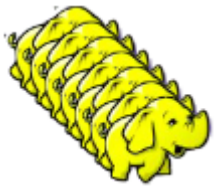


Job Tracker
(스케쥴러)

- Slaves



Data Nodes
(블록 데이터 저장소)

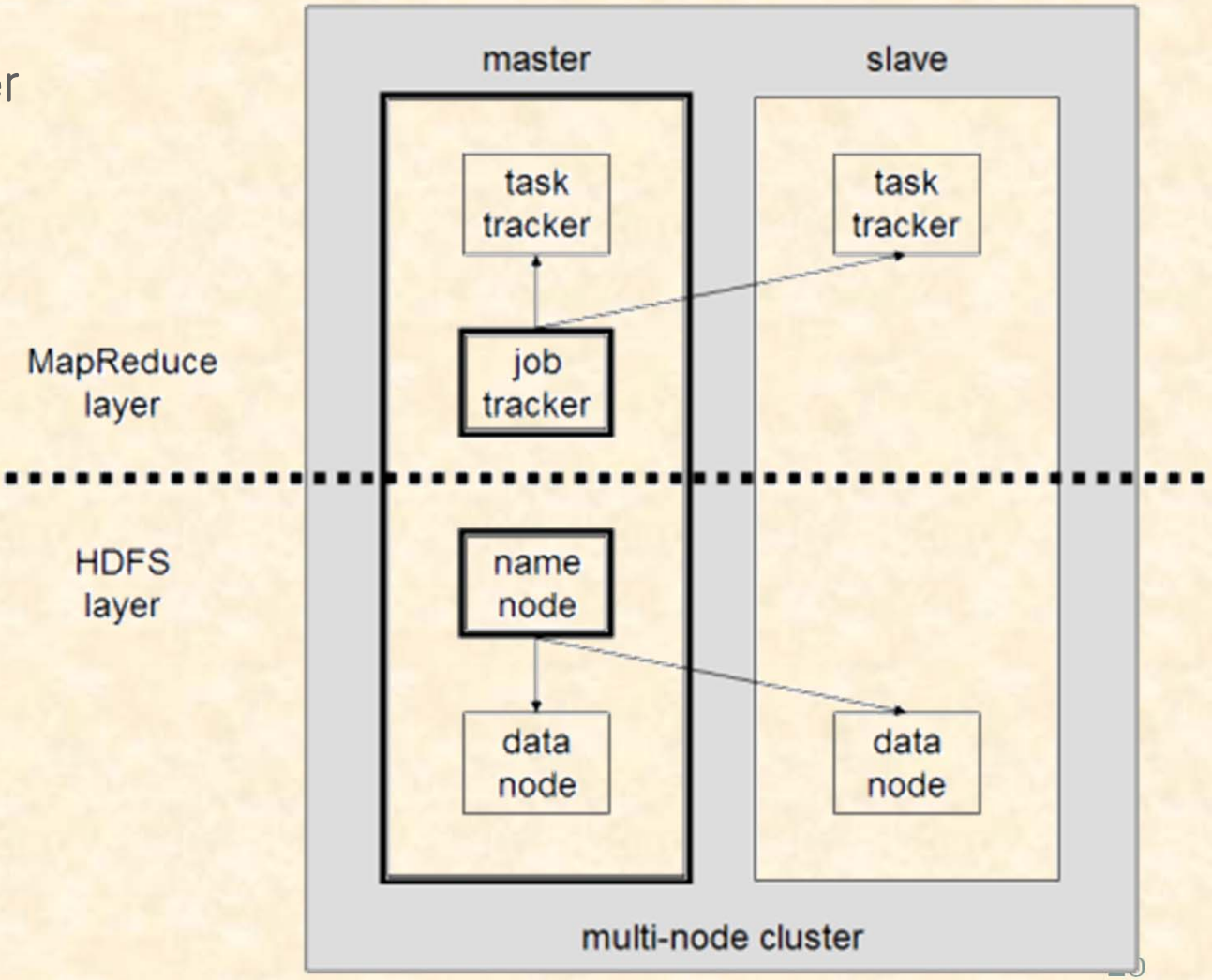


Task Trackers
(작업 실행자)

http://hadoop.apache.org/docs/r1.2.1/hdfs_user_guide.html

Basic Features: HDFS

- Multi-node Cluster
  - Name Node
  - Data Node
  - Job Tracker
  - Task Tracker



Namenode and Datanodes

- Master/slave architecture
- HDFS cluster consists of a single **Namenode**, a master server that manages the file system namespace and regulates access to files by clients.
- There are a number of **DataNodes** usually one per node in a cluster.
- The DataNodes manage storage attached to the nodes that they run on.
- HDFS exposes a file system namespace and allows user data to be stored in files.
- A file is split into one or more blocks and set of blocks are stored in DataNodes.
- DataNodes: serves read, write requests, performs block creation, deletion, and replication upon instruction from Namenode.

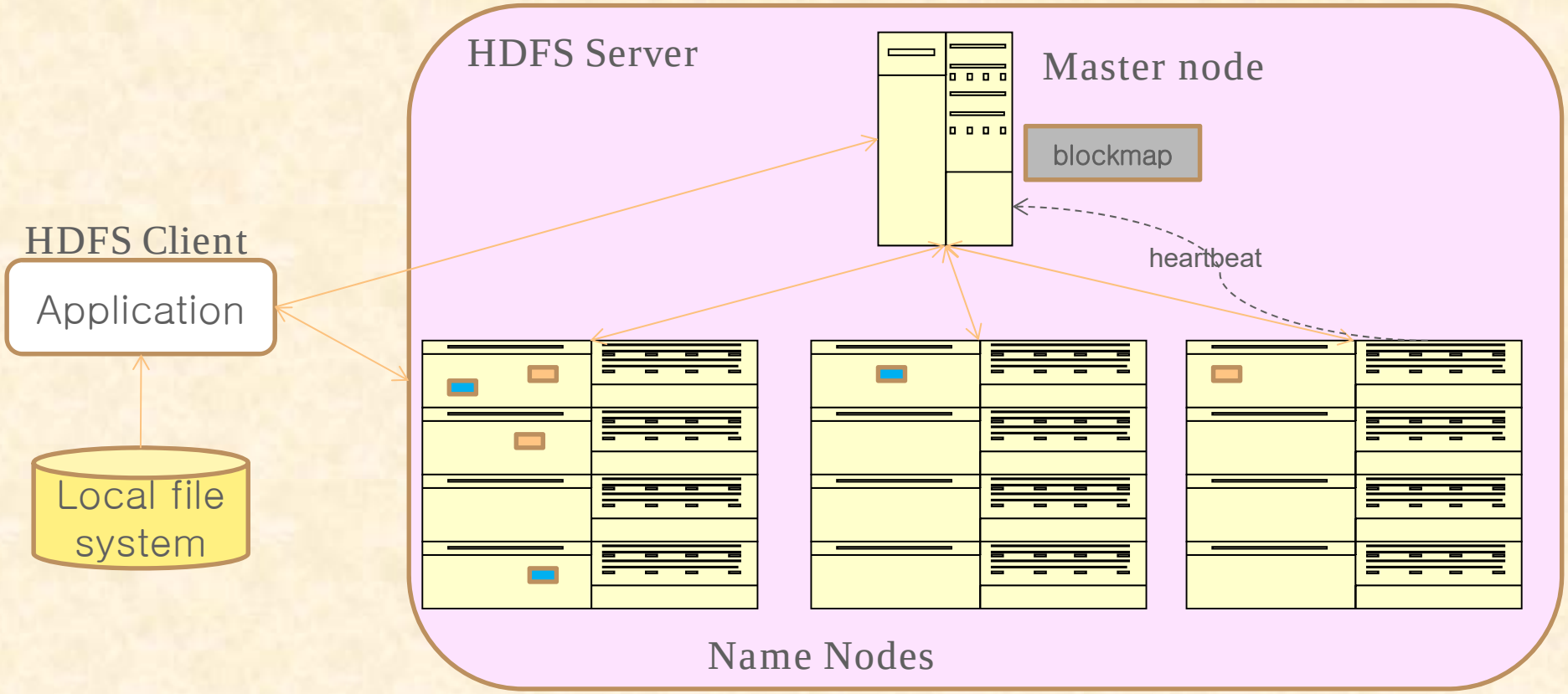
Data Replication

- HDFS is designed to store very large files across machines in a large cluster.

- Each file is a sequence of blocks.
- All blocks in the file except the last are of the same size.
- Blocks are replicated for fault tolerance.
- Block size and replicas are configurable per file.

- The Namenode receives a Heartbeat and a BlockReport from each DataNode in the cluster.
- BlockReport contains all the blocks on a Datanode.

Hadoop Distributed File System



MapReduce

"A simple and powerful interface that enables automatic parallelization and distribution of large-scale computations, combined with an implementation of this interface that achieves high performance on large clusters of commodity PCs."

MapReduce

- Large-Scale Data Processing
  - Want to use 1000s of CPUs
    - But don't want hassle of *managing* things
- MapReduce Architecture provides
  - Automatic parallelization & distribution
  - Fault tolerance
  - I/O scheduling
  - Monitoring & status updates

Map/Reduce

- Map/Reduce
  - Programming model from LISP (functional language)
- Many problems can be phrased this way
- Easy to distribute across nodes
- Nice retry/failure semantics

Map in Lisp

- (map *f list* [*list*$_2$ *list*$_3$ …])
- (map square '(1 2 3 4))
  - (1 4 9 16)
- (reduce + '(1 4 9 16))

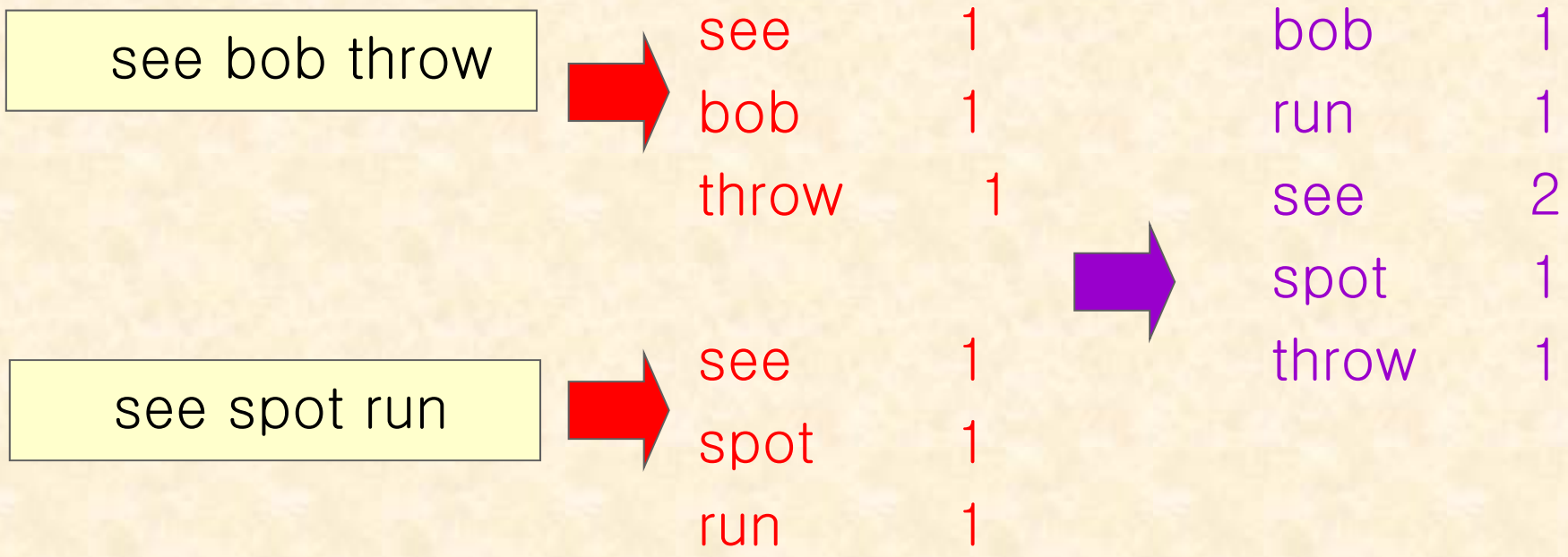
  => (+ 16 (+ 9 (+ 4 1)))

  ⇒30
- (reduce + (map square (map – l_1 l_2))))

Unary operator

Binary operator

Word Count



Can we do word count in parallel?

Partition Function

- Inputs to map tasks are created by contiguous splits of input file
- For reduce, we need to ensure that records with the same intermediate key end up at the same worker
- System uses a default partition function e.g., hash(key) mod R
- Sometimes useful to override
  - E.g., hash(hostname(URL)) mod R ensures URLs from a host end up in the same output file

MapReduce in Hadoop (2)

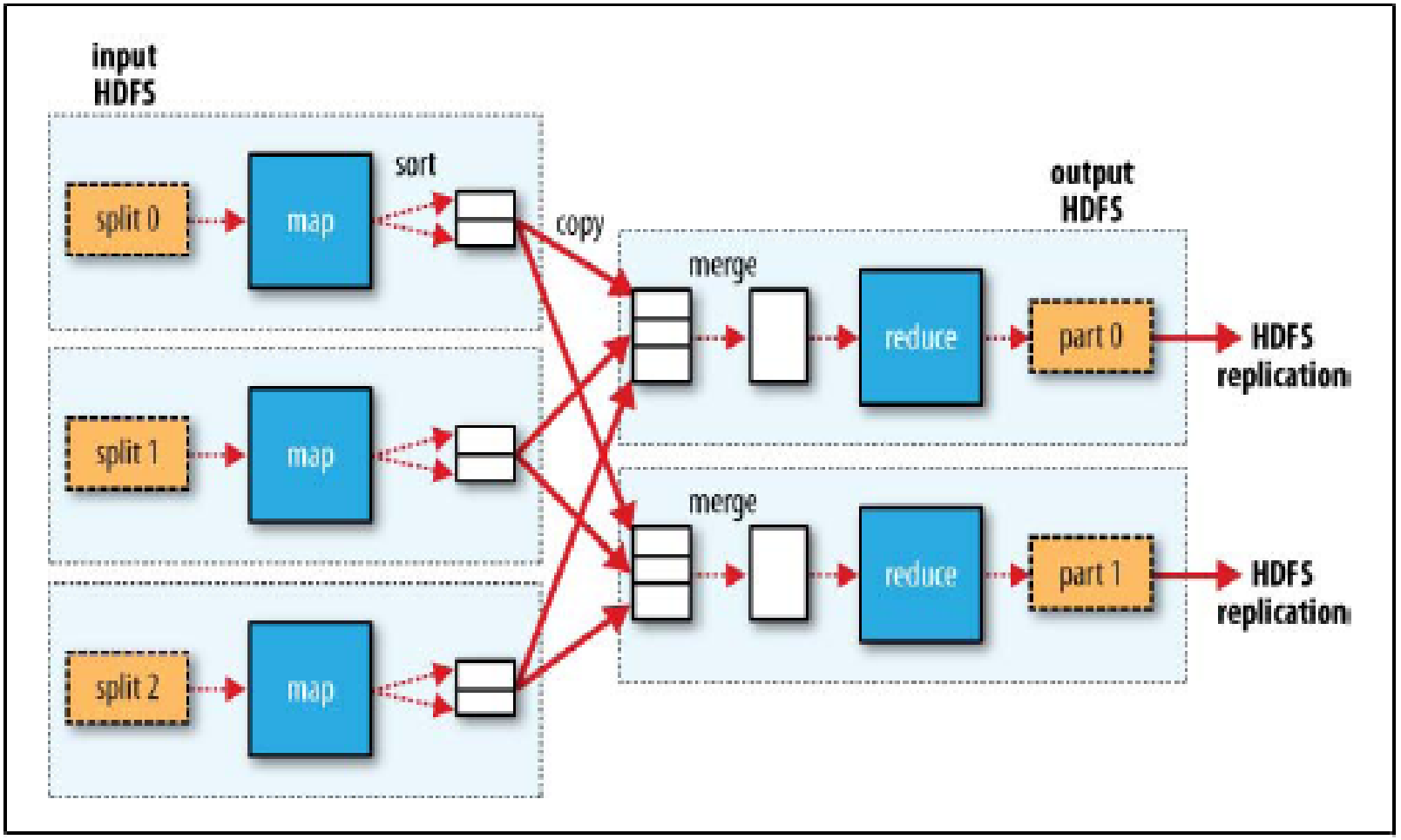


Figure 2-3. MapReduce data flow with multiple reduce tasks

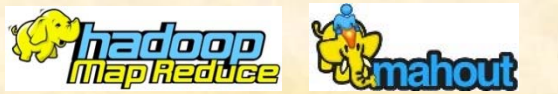
MapReduce Benefits

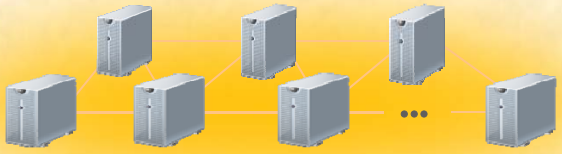
- Greatly reduces parallel programming complexity
  - Reduces synchronization complexity
  - Automatically partitions data
  - Provides failure transparency
  - Handles load balancing

빅데이터 기술 – 디스크 기반 접근

데이터 마이닝/분석

- 맵리듀스, Mahout





질의 처리

- NoSQL, Hive

로그 데이터

선 저장 후 분석

What are Data Streams? – Definition

- Common characteristics of new applications

  – Data input as continuous, ordered data streams

Definition of Data Streams

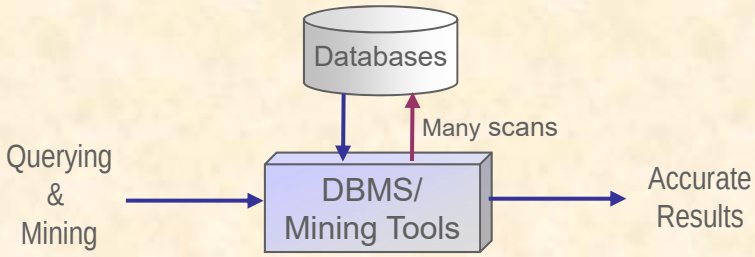
Continuous, unbounded, rapid, time-varying streams of data elements

- Goal:

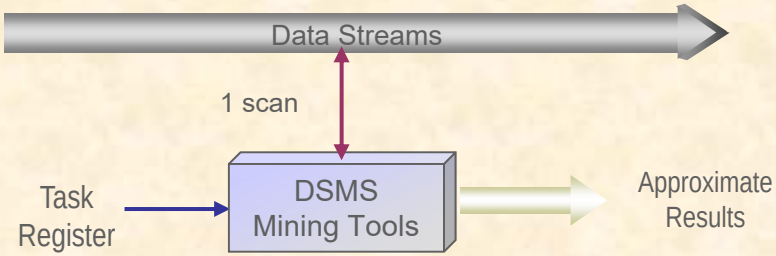
  Mine patterns, process queries and compute statistics on data streams in *real-time*

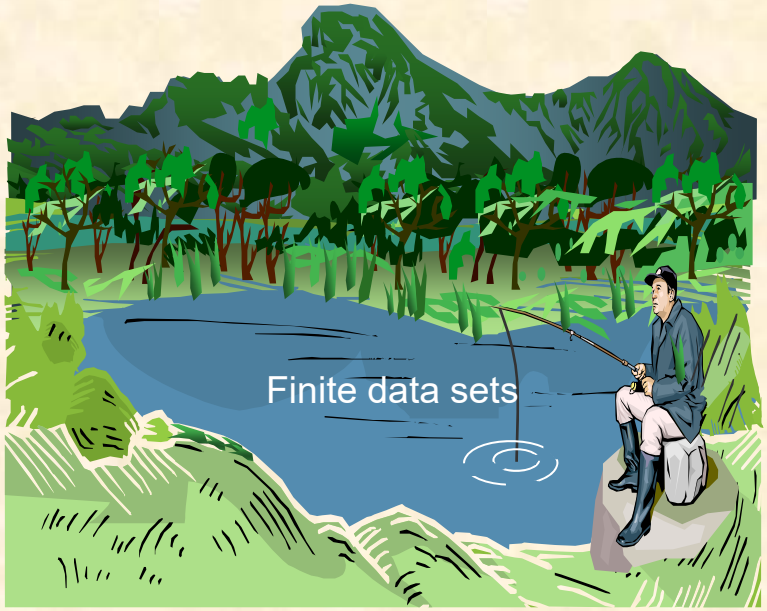
What are Data Streams? – Definition

Conventional Data Processing



Data Stream Processing



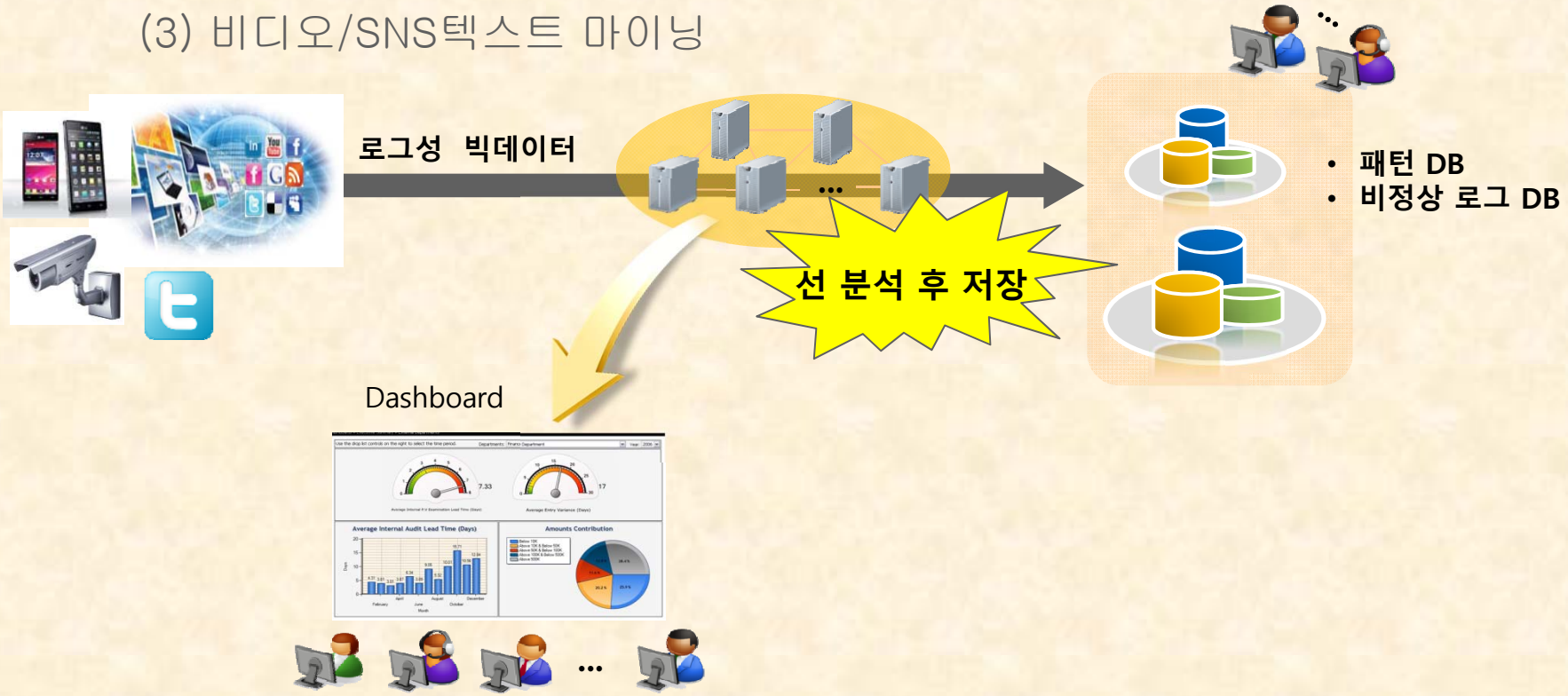


실시간 빅데이터 기술 - 데이터스트림

실시간 인메모리 데이터스트림 분석 기술

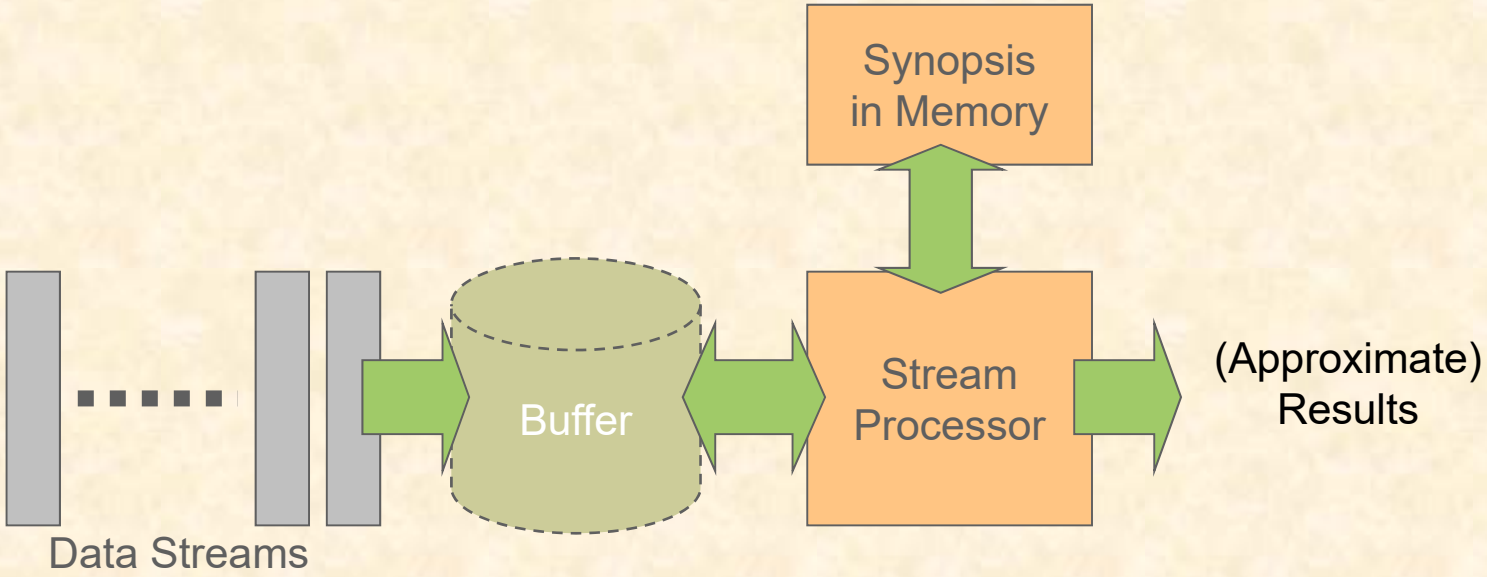
(ESPER, STORM, SPARK,FLINK)

(1) 실시간 데이터스트림 마이닝
(2) 실시간 연속질의 처리(상황인지, 주기, Top-k)
(3) 비디오/SNS텍스트 마이닝



Computation Model

- The pass-thru nature of data streams
  - Single pass: each record is examined at most once
  - Bounded (memory) storage: for storing synopsis
  - Real-time: processing time per record must be low

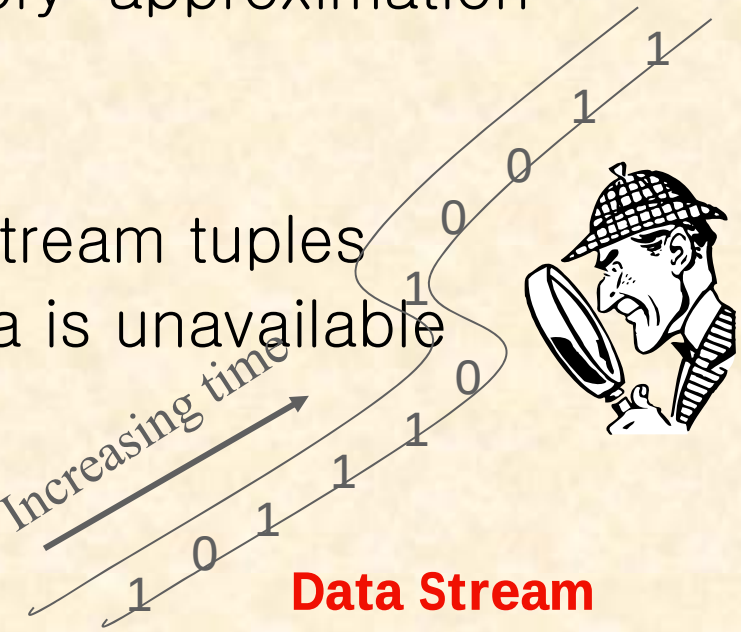


Computation Model

- To access or aggregate past data
- Need bounded-memory history-approximation

- Synopsis?
  - Succinct summary of old stream tuples
  - Like indexes, but base data is unavailable

- Basic synopsis computation
  - Samples
  - Histograms
  - Wavelet representation
  - Sliding Windows
  - ….



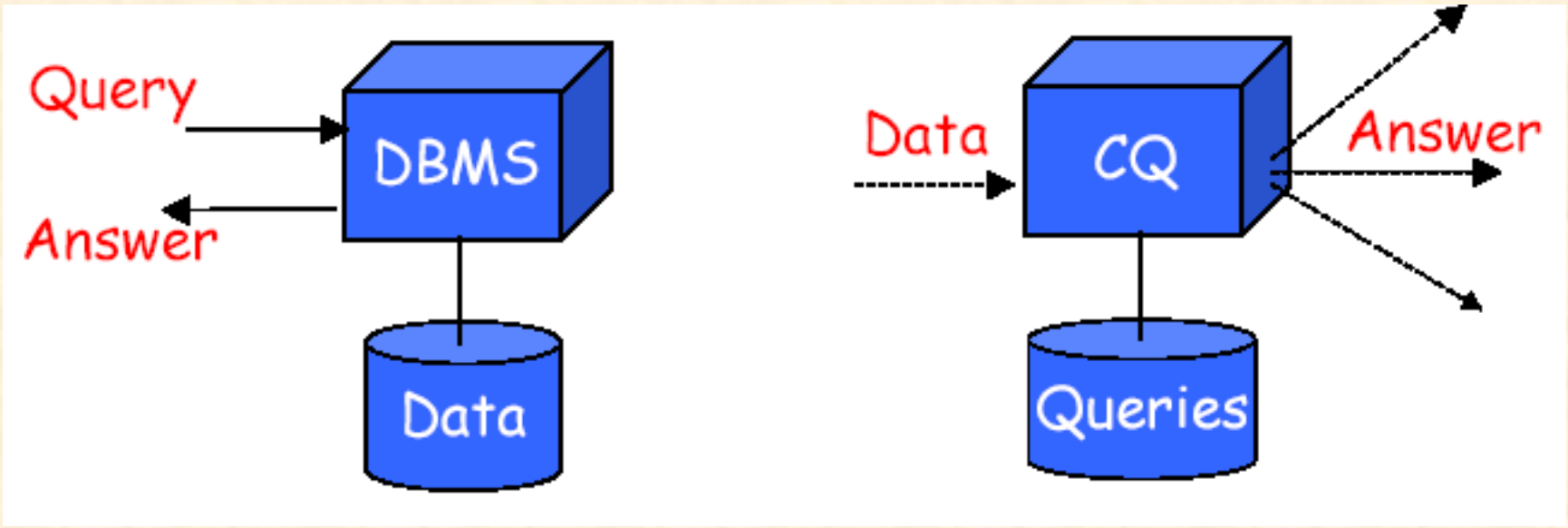
Data Stream Management Systems

- Data in the form of continuous arrival in multiple, rapid, time-varying, possibly unpredictable and unbounded streams

- Traditional DBMS – data stored in finite, persistent data sets
- New Applications – data input as continuous, ordered data streams

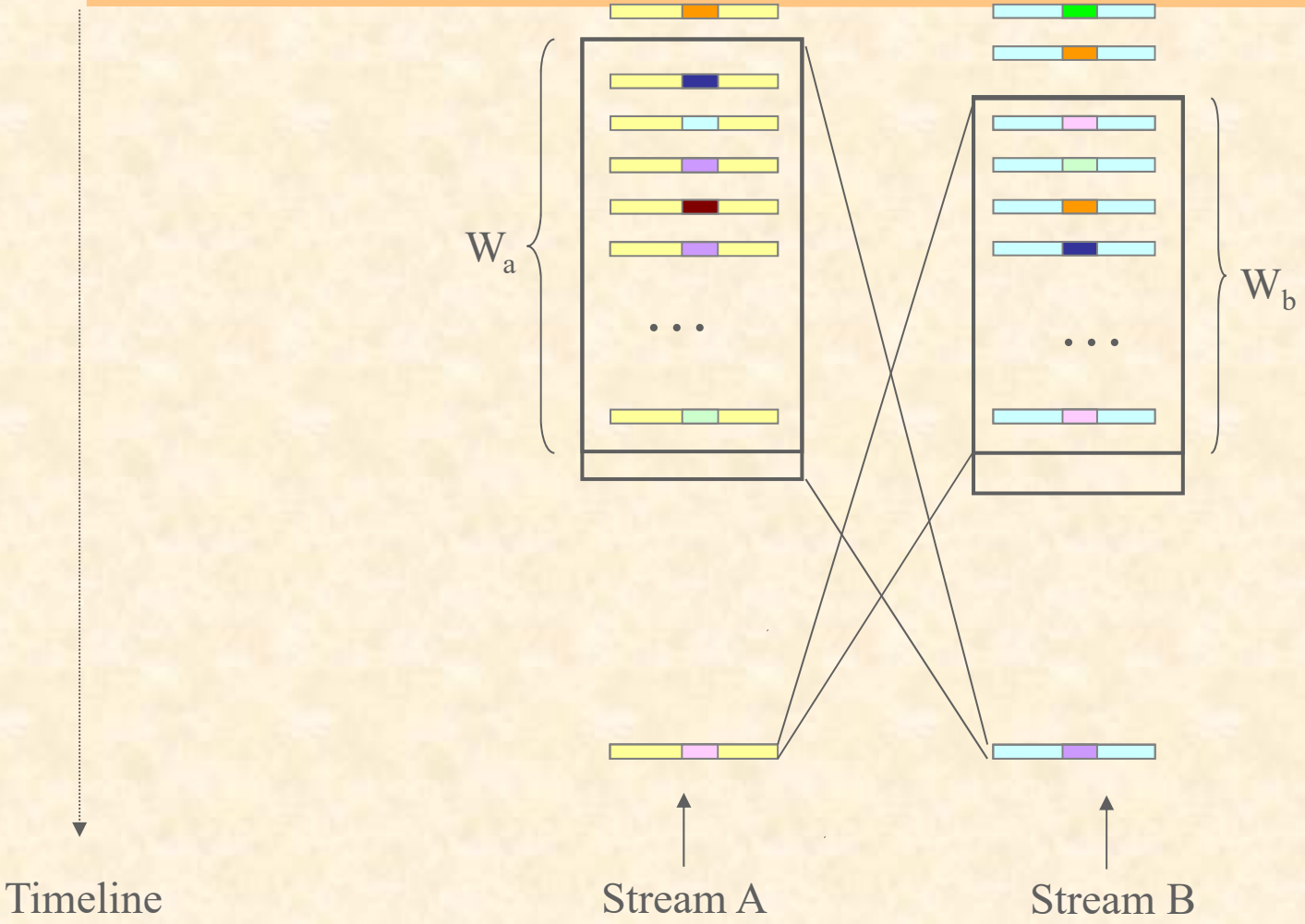
- Many applications
  - Financial applications, network monitoring, security, telecommunications data management, web application, manufacturing, sensor networks, etc.

Continuous Queries



- Conventional approach: a query executes over the current state of the database and terminates.
- Continuous queries are always running, and produce new answers incrementally as the database changes.

Sliding Window Join



Data Stream Management Systems

DBMS	DSMS
Persistent relations	Transient streams
One-time queries	Continuous queries
Random access	Sequential access
"Unbounded" disk store	Bounded main memory
No real-time services	Real-time requirements
Relatively low update rate	Possibly multi-GB arrival rate
Only current state matters	History/arrival-order is critical
Assume precise data	Stale/imprecise Data
Access plan determined by query processor, physical DB design	Unpredictable/variable data arrival and characteristics

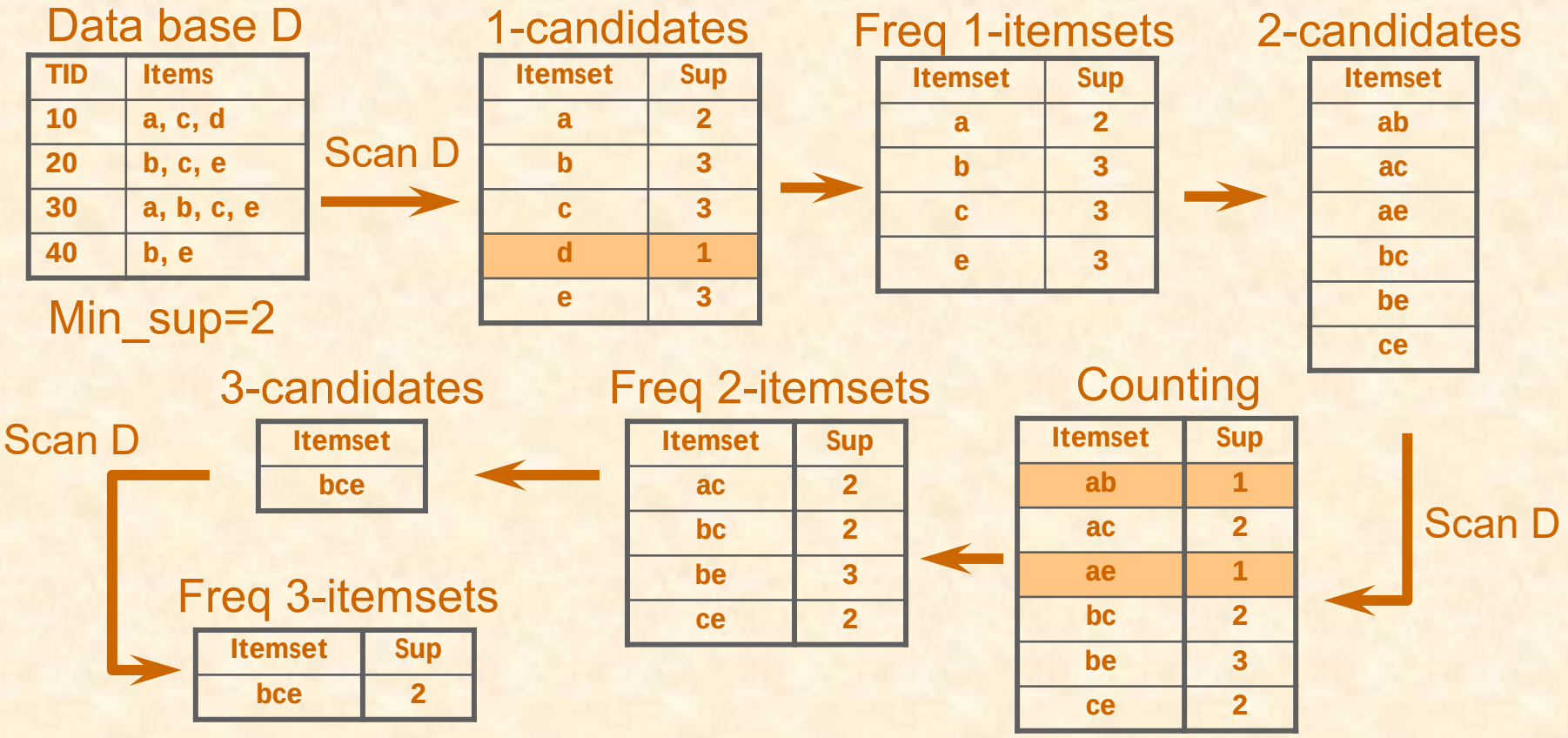
Data Stream Mining– Lossy Counting

- S = 0.1% , ε = 0.01%, N = 100000

1. ALL elements with frequency exceeding 0.1% will be output. (count>=sN=0.001*100000=100)
2. NO element with frequency below 0.09% will be output. (count < 0.0009*100000=90)
   - Elements 0.09% ~ 0.1% may or may not be output. (90 <= count <100) *false positives*
3. All individual frequencies are less than their true frequencies by at most 0.01%

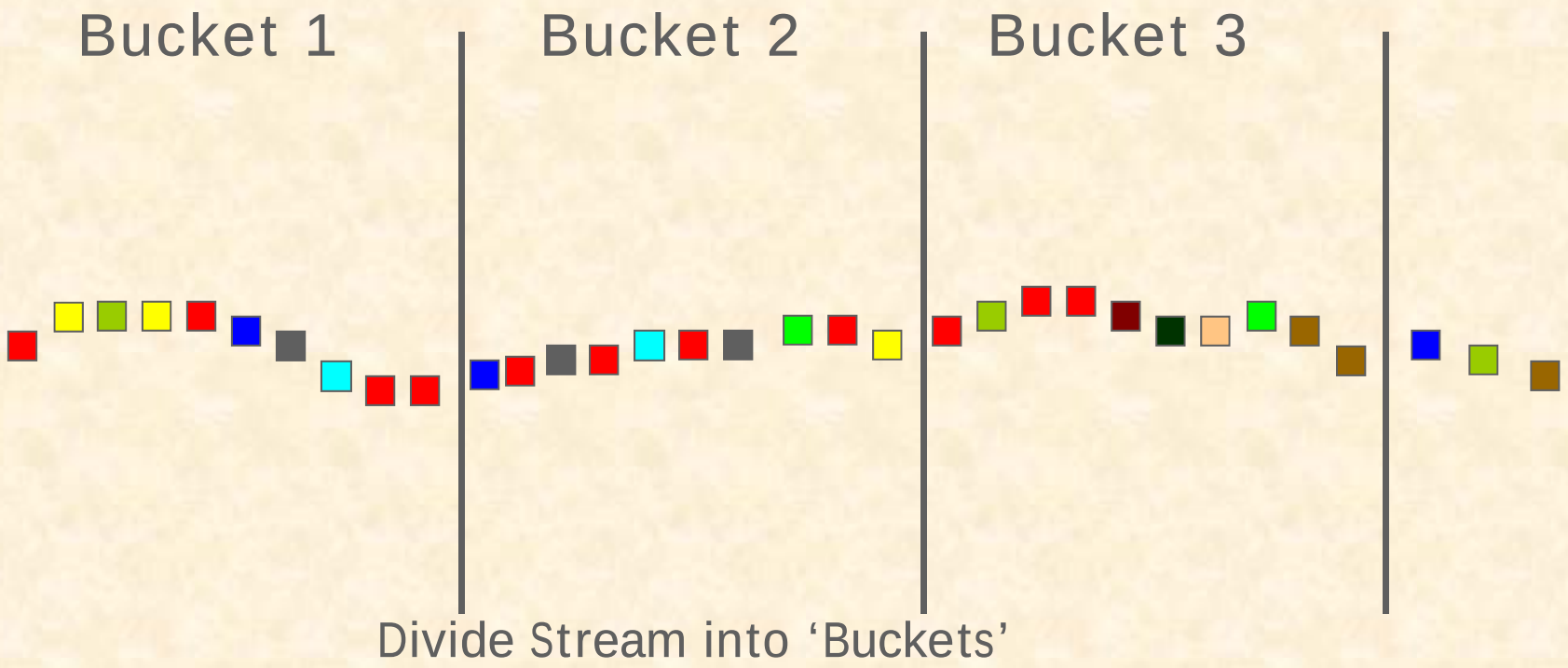
Association Rules – Apriori Algorithm

- A level-wise, candidate-generation&test approach (Agrawal & Srikant 1994)



Data Stream Mining– Lossy Counting

- **S = 3/10 for the stream so far**
- **Feels that 1/10 error is comfortable**



Data Stream Mining– Lossy Counting

First Bucket

Empty
(summary)

At bucket boundary, decrease all counters by 1

Data Stream Mining– Lossy Counting

Next Bucket

- **S = 3/10 for the stream so far**
- **Feels that 1/10 error is comfortable**

Output: items with frequency counts exceeding (3-1)/10*N

At bucket boundary, decrease all counters by 1

K-Means Example

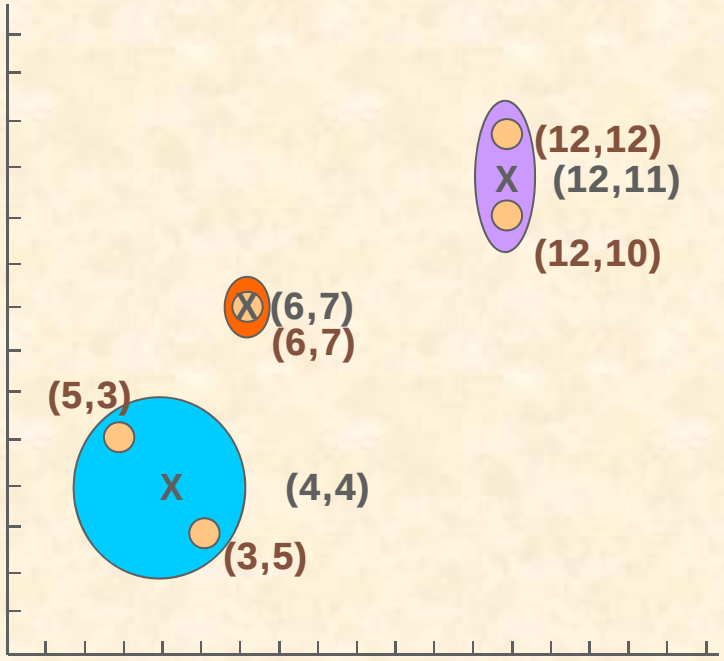
Given: {2,4,10,12,3,20,30,11,25}, k=2
Randomly assign means: m_1=3,m_2=4

- K_1={2,3}, K_2={4,10,12,20,30,11,25}, m_1=2.5,m_2=16
- K_1={2,3,4},K_2={10,12,20,30,11,25}, m_1=3,m_2=18
- K_1={2,3,4,10},K_2={12,20,30,11,25}, m_1=4.75,m_2=19.6
- K_1={2,3,4,10,11,12},K_2={20,30,25}, m_1=7,m_2=25

Data Stream Mining– K-median

1. For a new stream chunk D^1, find k(=3) intermediate centers

(3,5), (5,3), (6,5),(12,10),(12,12)

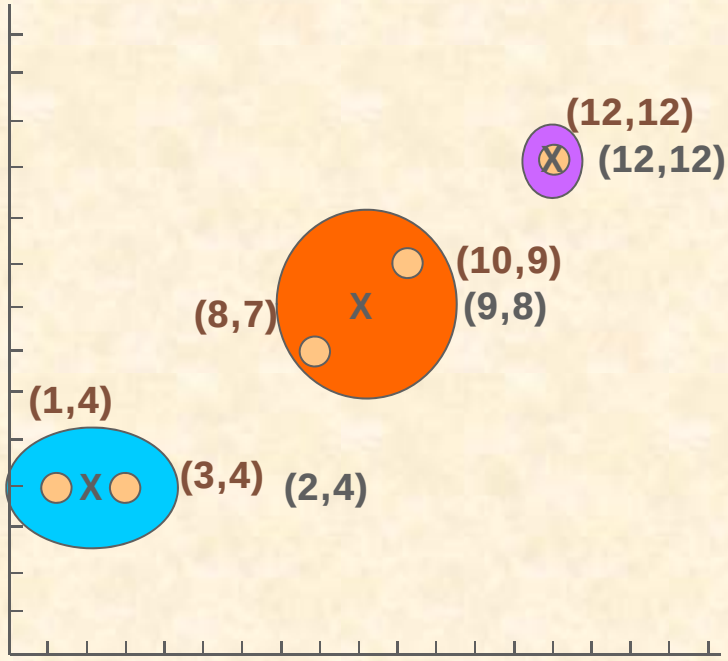


Memory

center	weight
(4,4)	2
(6,7)	1
(12,11)	2

Data Stream Mining– K-median

2. Repeat 1. to find k intermediate centers at each i^{th} stream chuck D^i is created **(1,4), (3,4), (8,7),(10,9), (12,12)**

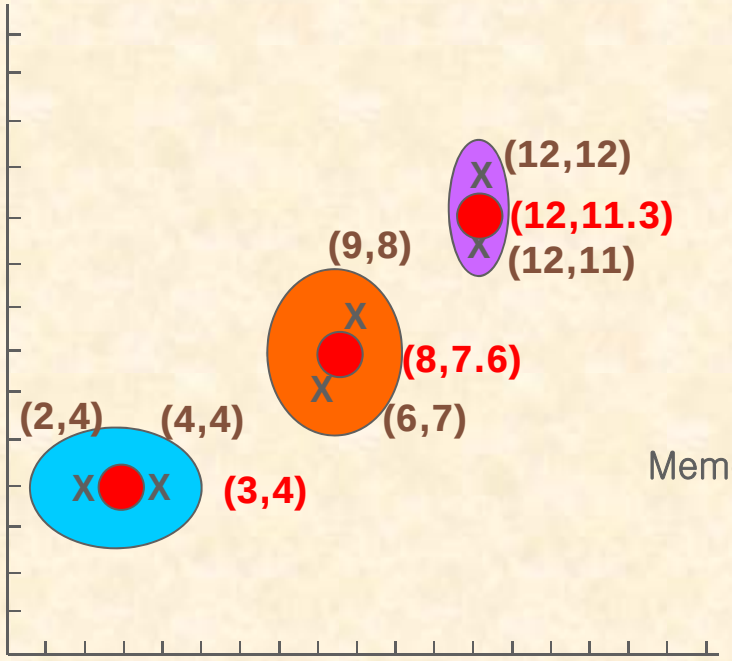


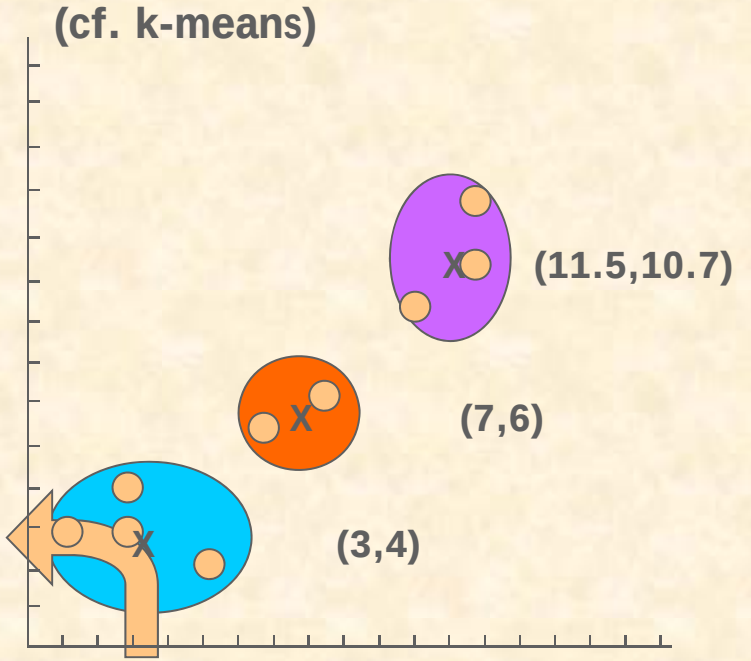
Memory

center	weight
(4,4)	2
(6,7)	1
(12,11)	2
(2,4)	2
(9,8)	2
(12,12)	1

Data Stream Mining– K-median

3. cluster these intermediate centers to find the *k* centers.





Memory

center	weight
(3,4)	4
(8,7.6)	3
(12,11.3)	3
(2,4)	2
(9,8)	2
(12,12)	1

k-means

center	weight
(3,4)	4
(7,6)	2
(11.5,10.7)	4

Storm

- Distributed realtime computation system
  - Creates by Nathan Marz @ BackType/Twitter
  - Free and Open source
  - process unbounded streams of data
  - Similar functionality as a Map-Reduce
- Characteristic
  - Scalability
  - Fault-tolerance
  - Multi language

Storm vs Hadoop

Hadoop

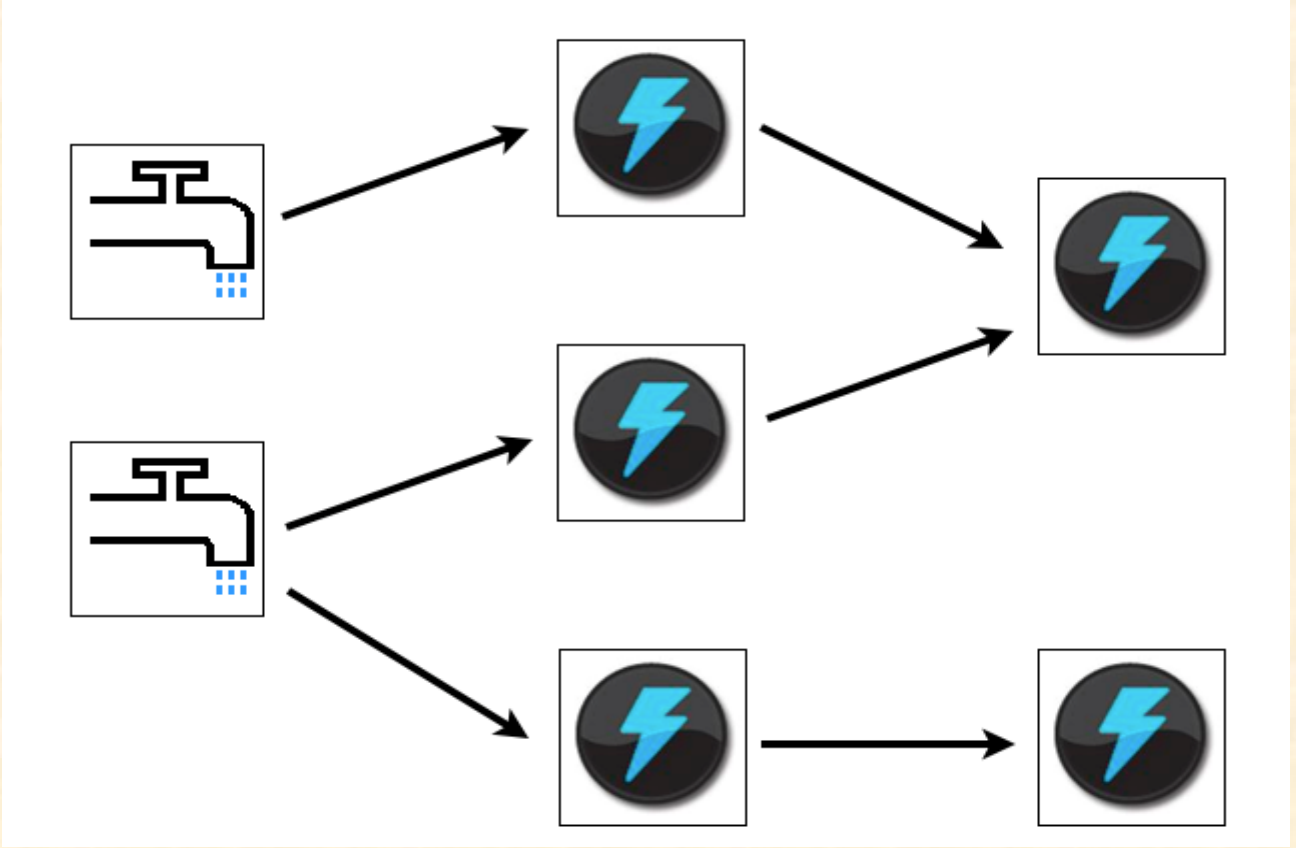
Hadoop is batch computation.

Storm

Storm is real-time computation.

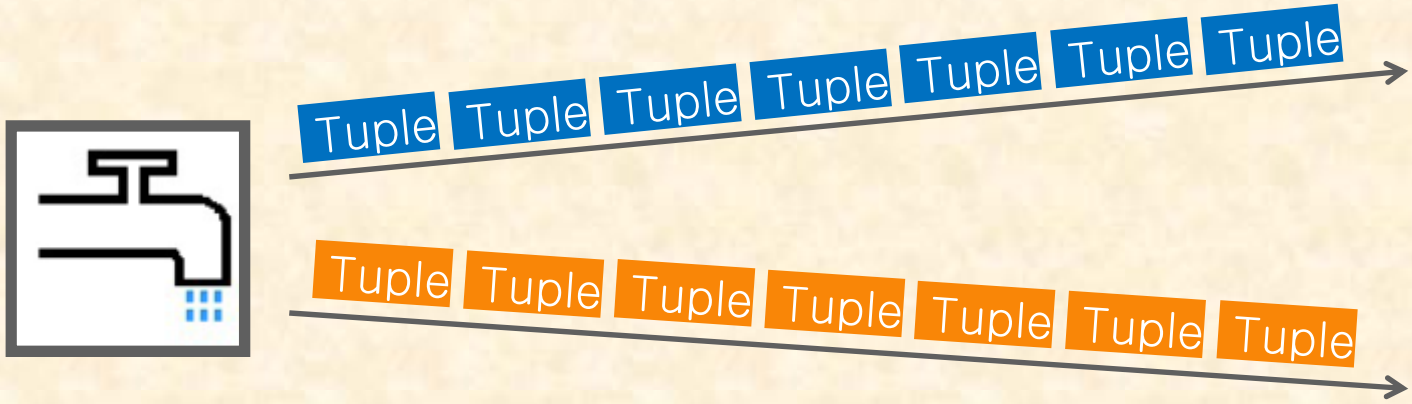
Storm | Component

- Topology
  - A network of spouts and bolts



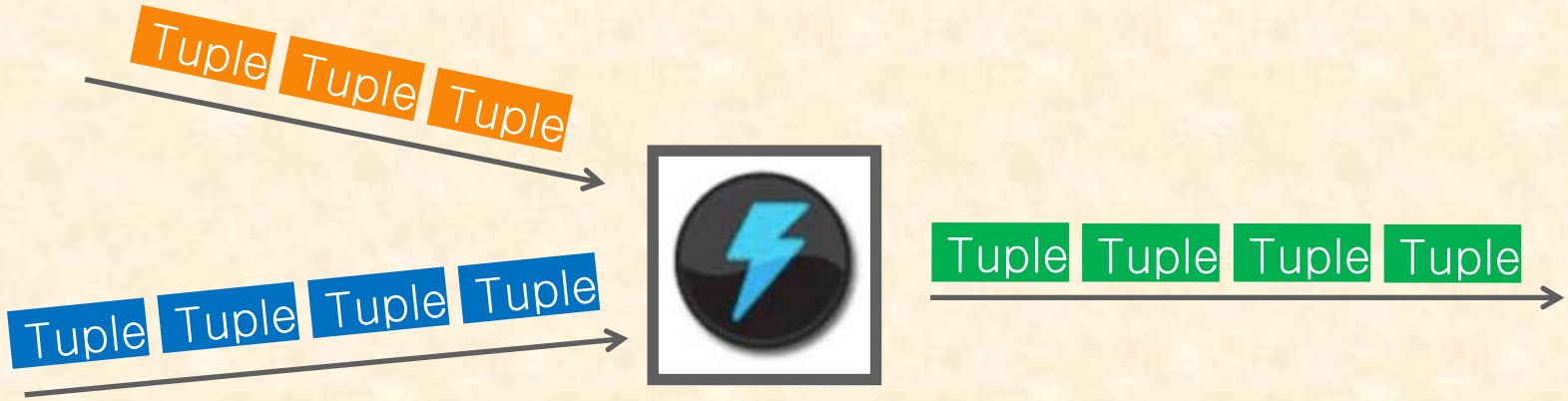
Storm | Component

- Spout
  - Source of streams in a computation.
  - Read from a queueing broker
  - generate its own stream or read from somewhere like the twitter



Storm | Component

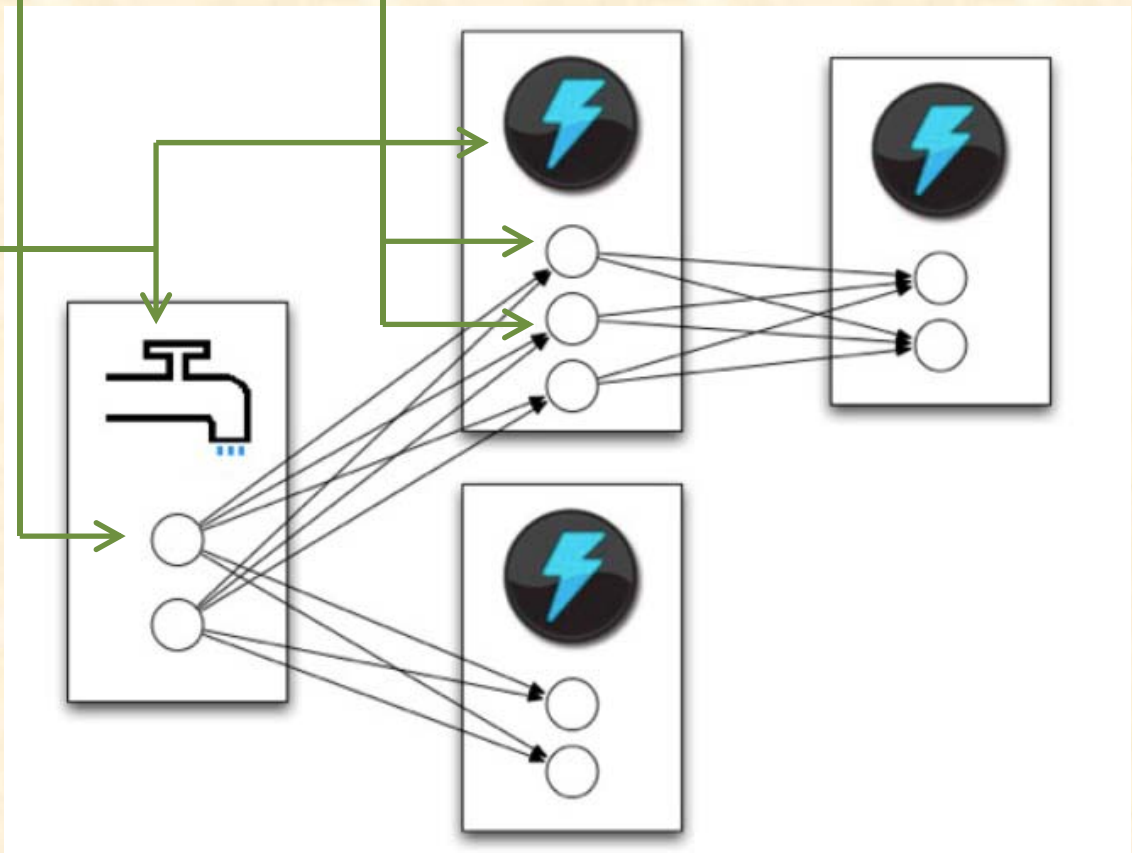
- Bolt
  - Processes any number of input streams
  - Produces any number of output streams



  - Filters
  - Aggregation
  - Joins
  - Connect to DB
  - User defined function

Storm | Task & Worker

- Task
  - Each spout or bolt executes many tasks across the cluster.
- Worker
  - Topologies execute across one or more worker processes



Storm vs Hadoop

Hadoop	Storm
• Batch	• RealTime
• JobTracker	• Nimbus
• TaskTracker	• Supervisor
• Task	• Worker(spout, bolt)
• Only two stages in processing pipeline (map and reduce)	• Multiple stages in processing pipeline
• Necessary storage(HDFS)	• Unnecessary storage
• One-time processing	• Continuous processing

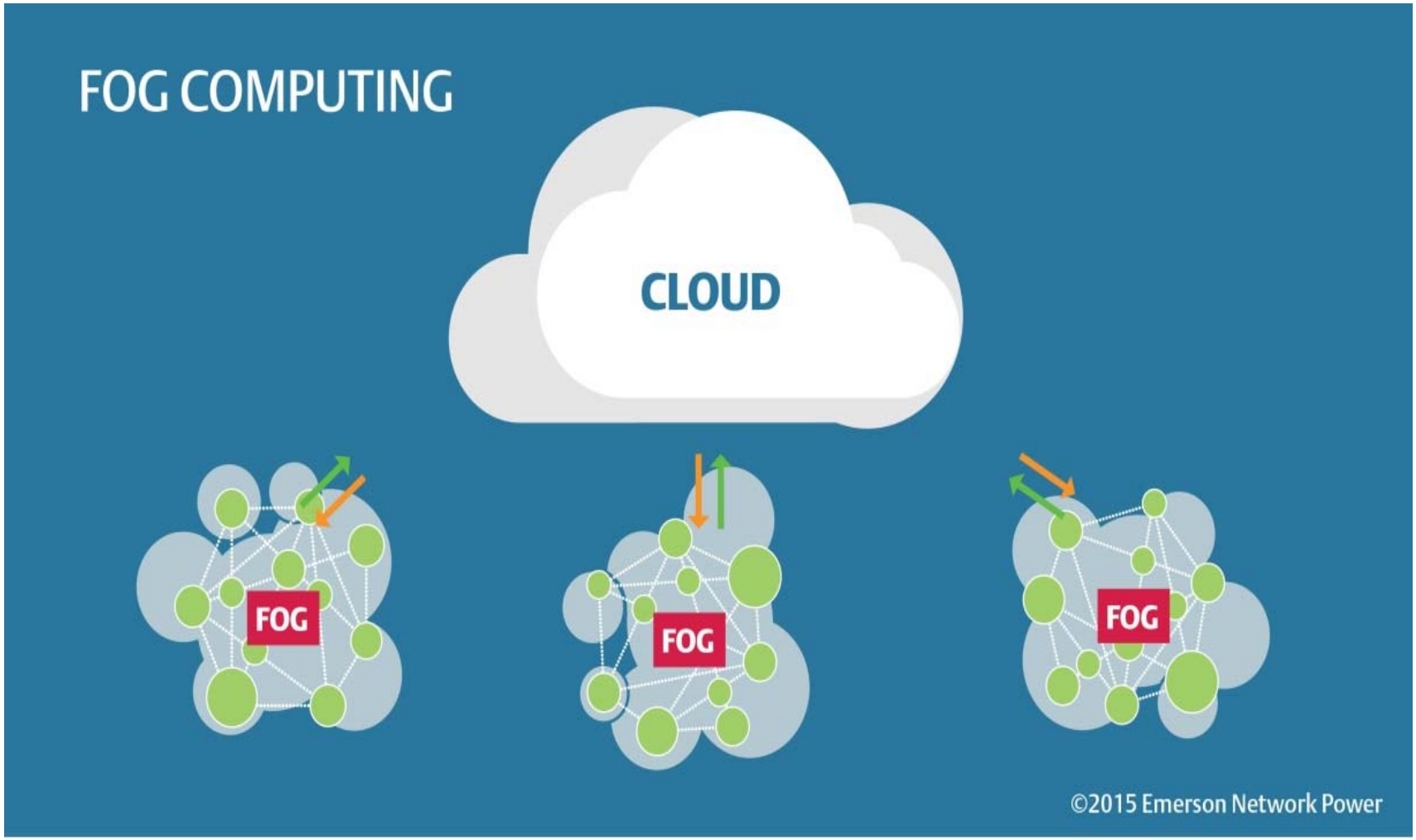
실시간 빅데이터 기술 – Fog computing

Fog Computing/Edge Computing

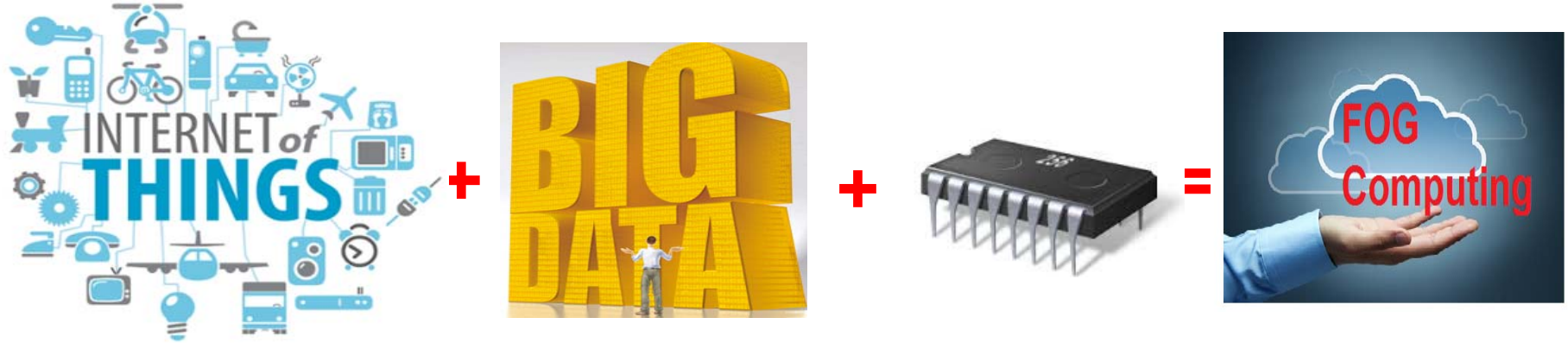
구름(Cloud)

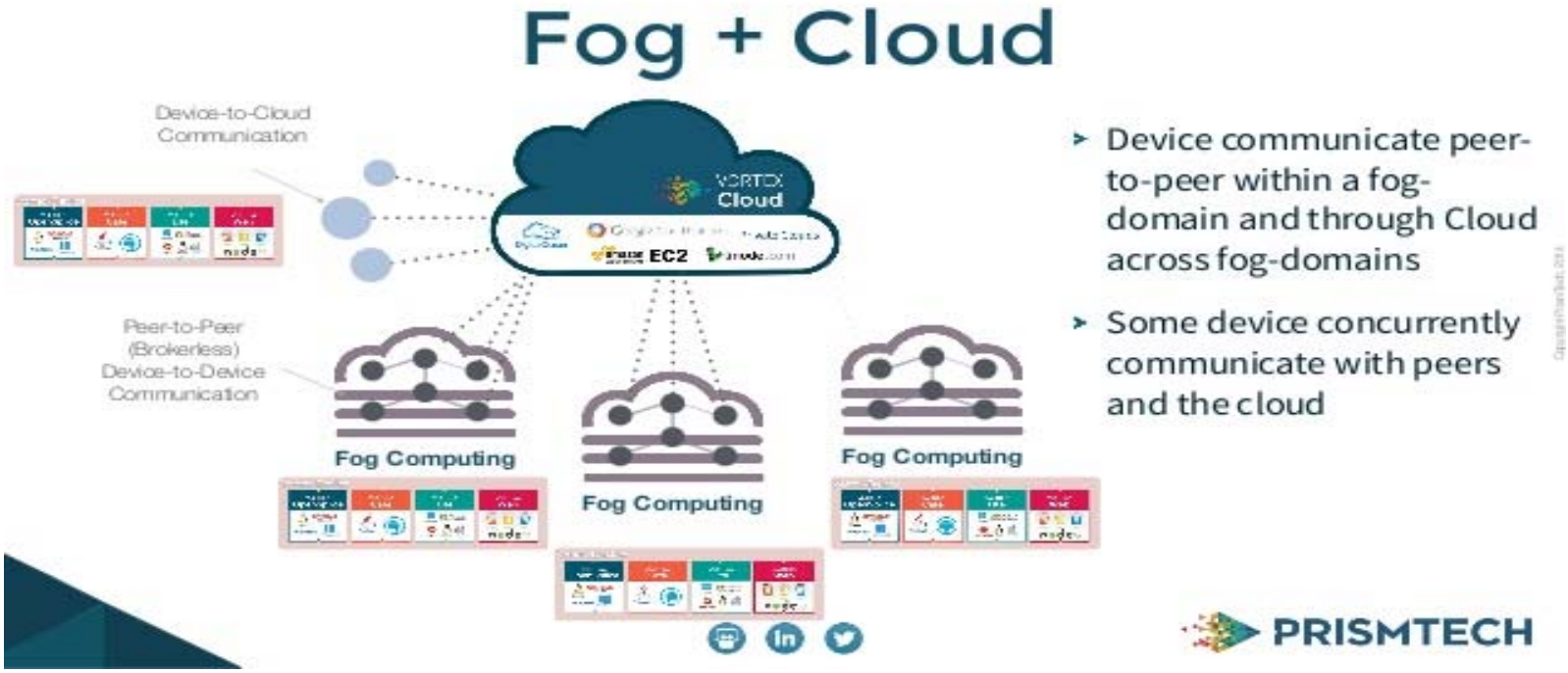
안개(Fog)

실시간 빅데이터 기술 – Fog computing



실시간 빅데이터 기술 – Fog computing





실시간 빅데이터 기술 – Fog computing

+

=

Fog Computing/
Edge Computing

+

Fog Computing/
Edge Computing

=

시나리오 #3: Smartphone Mining

❖ 예제 1 : 방향 센서를 이용한 사진 촬영 후 MMS 첨부 (학습 완료)

❖ 예제 2 : 방향(Orientation) 센서를 이용한 앱 실행 (학습 완료)

❖ 실시간 의미적 SNS 분석 기술

- 초기 Seed 단어 사전 제공 (자살, 투신 등)
- 지속적 자기 학습을 통한 연관 단어 자율적 인지 (빈발한 자살 관련 단어)

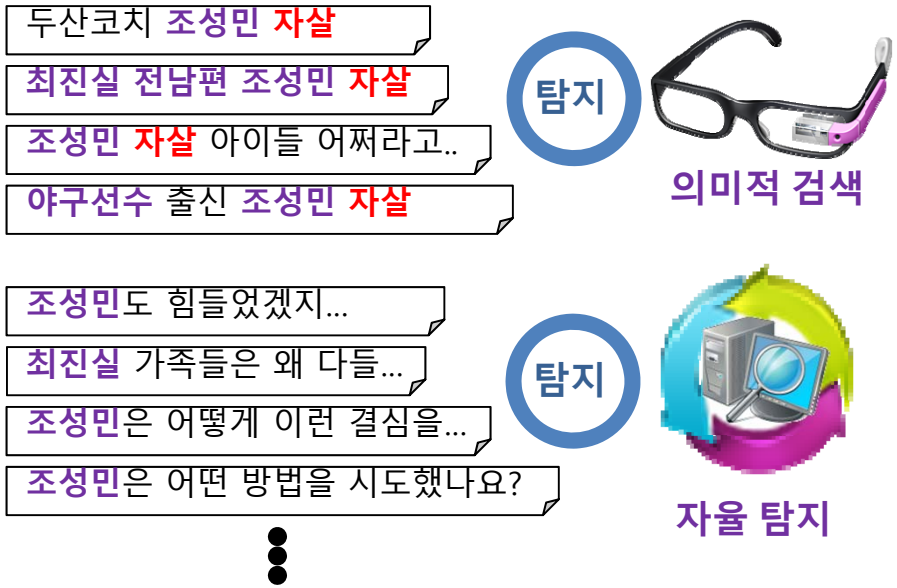
실시간 빅데이터 기술 - 데이터스트림

기존의 SNS 분석 기술

- **키워드 검색** – 주제 관련 트윗에 직접적인 키워드가 없으면 검출하지 못함
- **수동 학습 사전** – 관련 키워드를 수동적으로 데이터를 정제 및 처리하여 학습
- **단순 질의** – 키워드와의 단순 매칭을 통한 트윗 검색

실시간 의미적 SNS 분석 기술

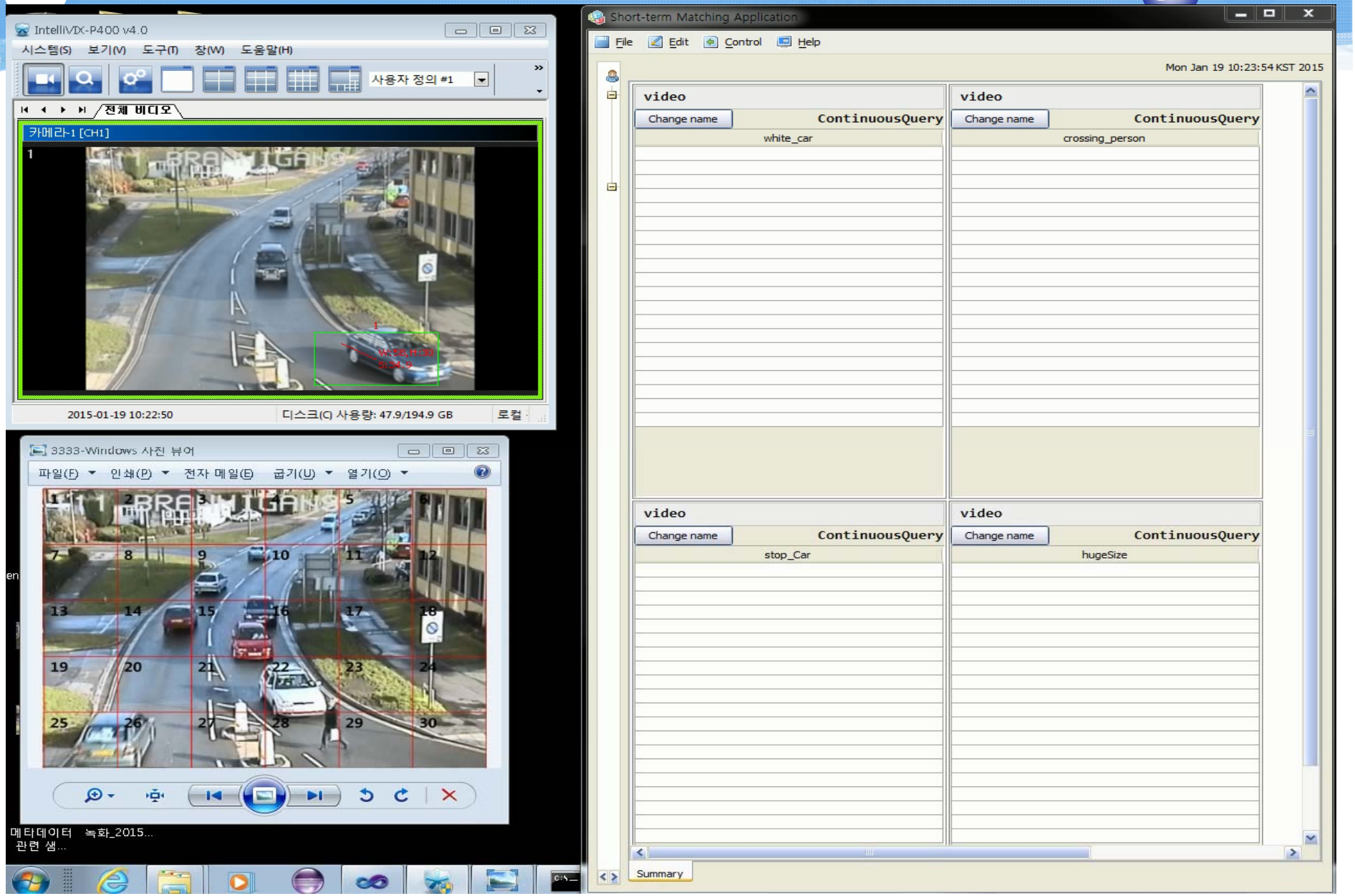
- **의미적 검색** – 주제 관련 트윗에 의미적으로 접근하여 트윗을 감지
- **자동 학습 사전** – 초기 데이터만 등록하면 지속적으로 관련 키워드 자동 학습
- **복합질의** – 연관성, 매칭 개수 등을 SQL 집계를 활용하여 트윗 검색 조건의 다양화

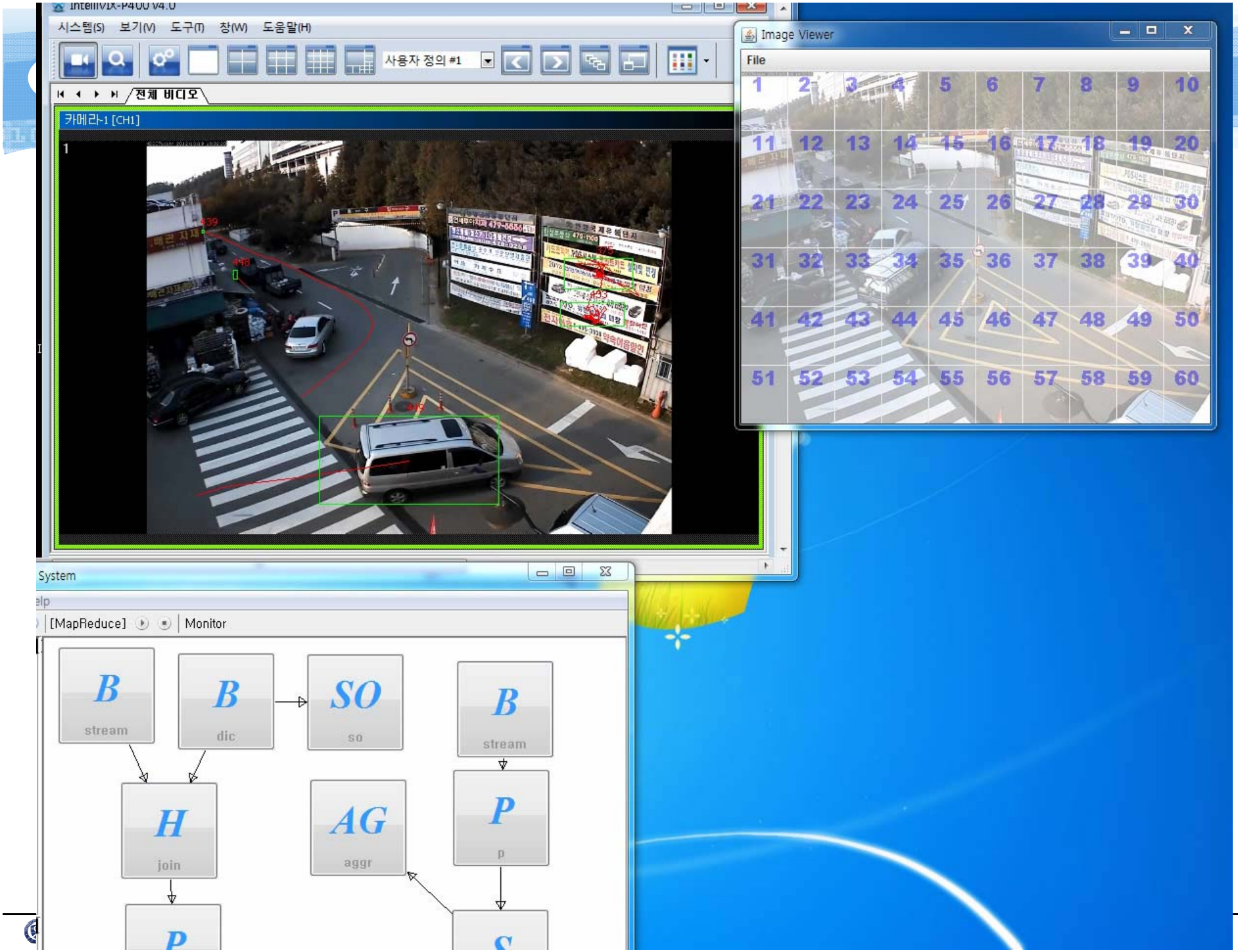


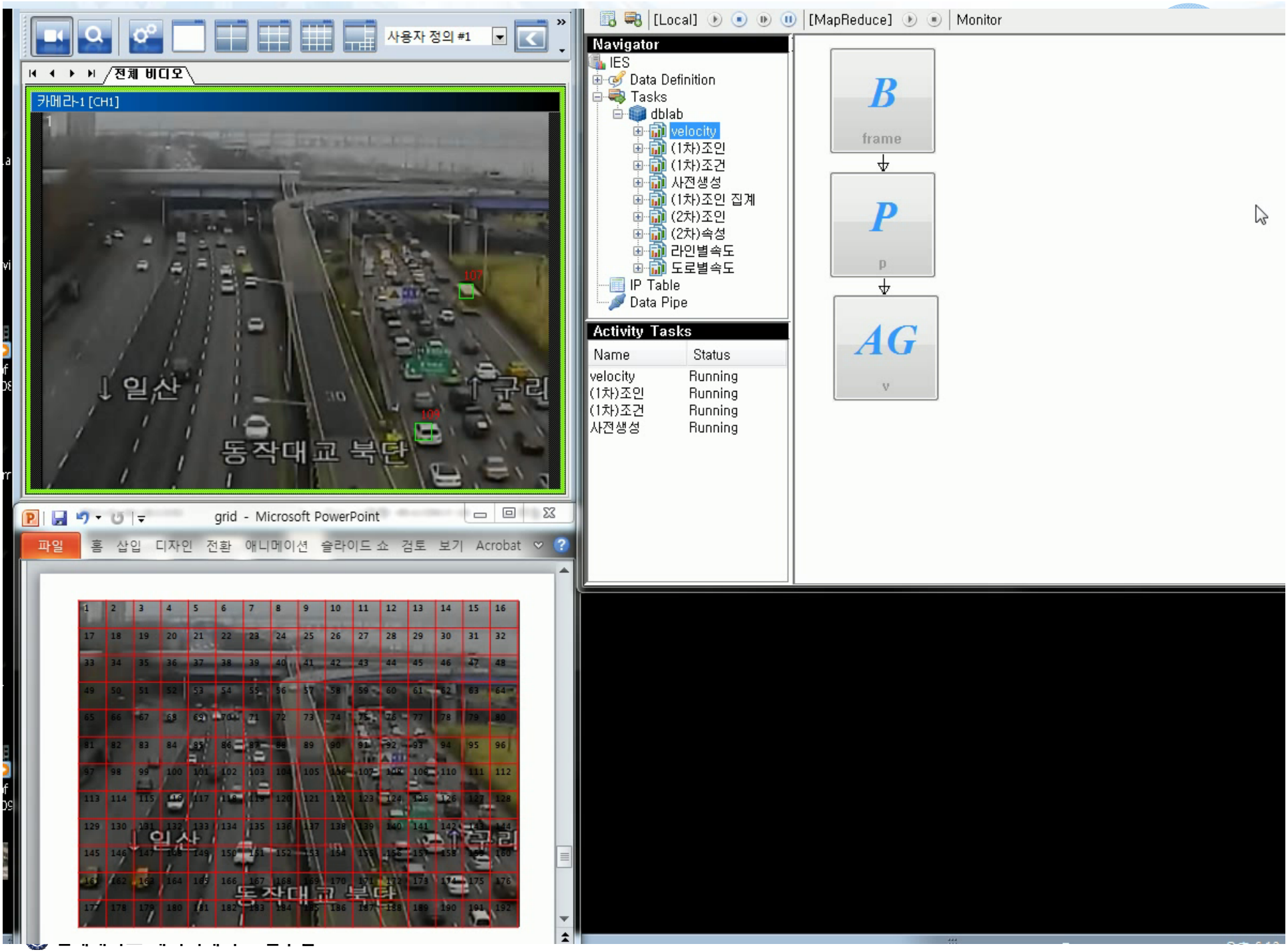
2012년도 학습 일반 단어

단어	지지도
자살	0.899831
중학생	0.443077
부모	0.37
광주	0.332
3명	0.325
대구	0.324286
유치장	0.315
학교폭력	0.29
학생	0.288
폭력	0.256
가해자	0.246667
처벌	0.235714
생활	0.231667
죽음	0.228571
...	...

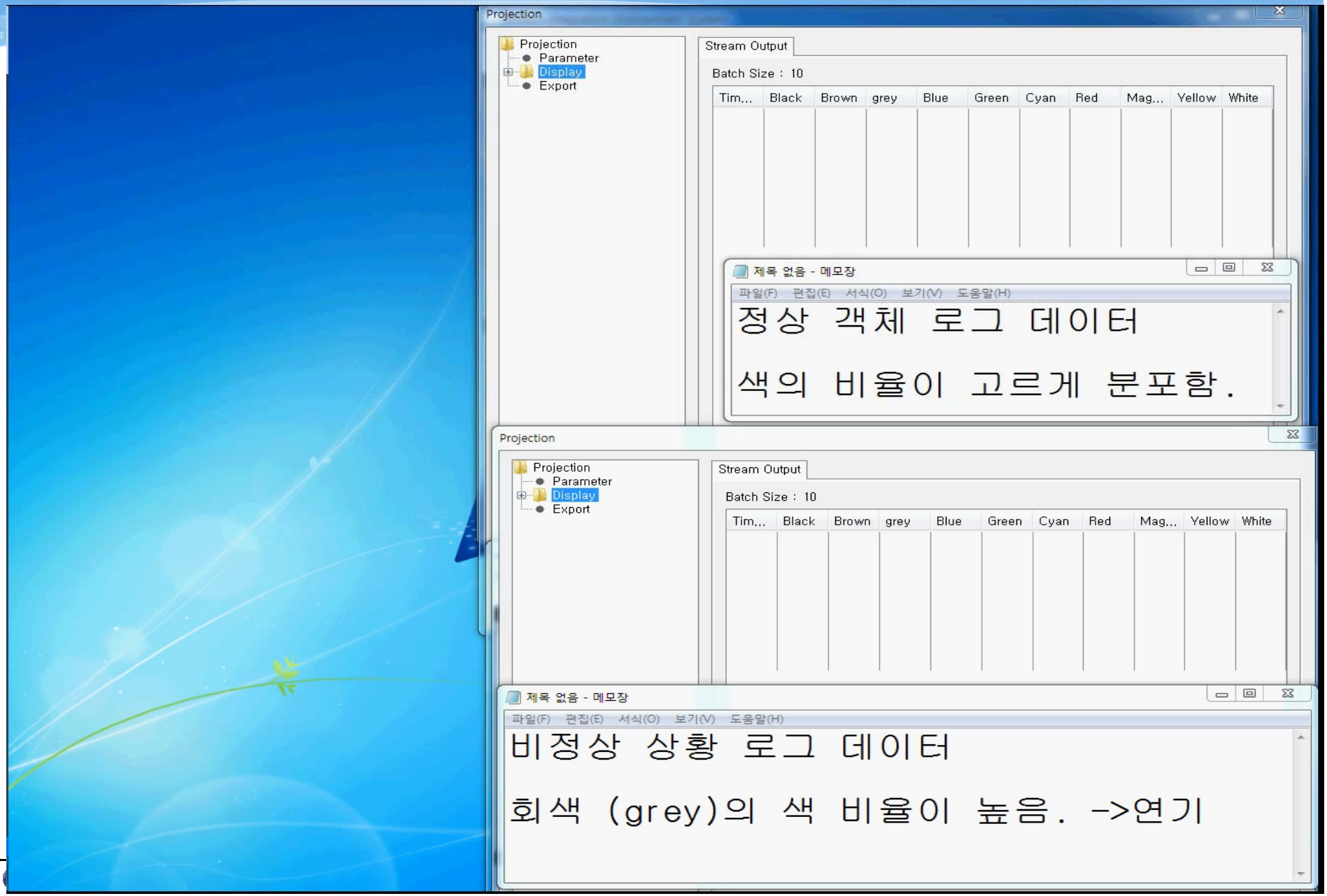
실시간 빅데이터 기술 – 데이터스트림




시스템(S) 보기(V) 도구(T) 창(W) 도움말(H)
사용자 정의 #1
전체 비디오
카메라-1 [CH1]
Image Viewer
File
System
[MapReduce]
Monitor
B
stream
B
dic
SO
so
B
stream
H
join
AG
aggr
P
p

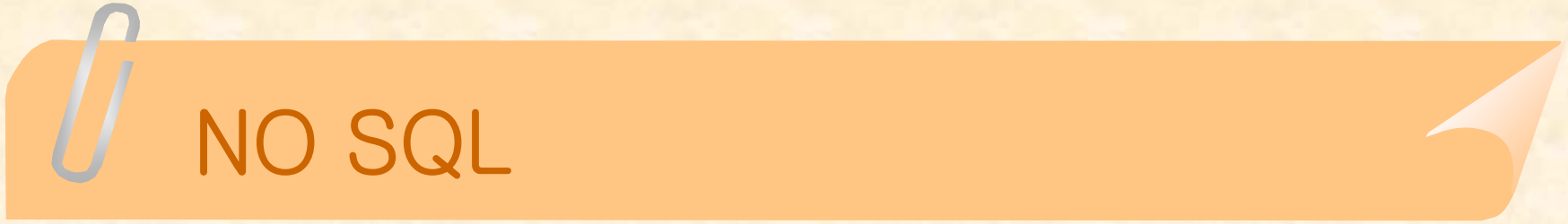

사용자 정의 #1
전체 비디오
카메라-1 [CH1]
일산
금릉
동작대교 북단
[Local]
[MapReduce]
Monitor
Navigator
IES
Data Definition
Tasks
dblab
velocity
(1차)조인
(1차)조건
사전생성
(1차)조인 집계
(2차)조인
(2차)속성
라인별속도
도로별속도
IP Table
Data Pipe
Activity Tasks
Name
Status
velocity Running
(1차)조인 Running
(1차)조건 Running
사전생성 Running
B
frame
P
p
AG
v
grid - Microsoft PowerPoint
파일
홈
삽입
디자인
전환
애니메이션
슬라이드 쇼
검토
보기
Acrobat

실시간 빅데이터 기술 – Fog computing



인물 없는 경우

인물 있는 경우


NO SQL

Cloud computing, cloud databases

- Cloud computing
  - data intensive applications on hundreds of thousands of commodity servers and storage devices
  - basic features:
    - elasticity,
    - fault-tolerance
    - automatic provisioning
- Cloud databases: traditional scaling up (adding new expensive big servers) is not possible
  - requires higher level of skills
  - is not reliable in some cases
- Architectural principle: scaling out (or horizontal scaling) based on data partitioning, i.e. dividing the database across many (inexpensive) machines

Cloud computing, cloud databases

- Technique: data sharding, i.e. horizontal partitioning of data
  (e.g. hash or range partitioning)

- Consequences:
  - manage parallel access in the application
  - scales well for both reads and writes
  - not transparent, application needs to be partition-aware

Relaxing ACID properties

- Cloud computing: ACID is hard to achieve, moreover, it is not always required, e.g. for blogs, status updates, product listings, etc.
- Availability
  - Traditionally, server/process available 99.999 % of time
  - For a large-scale node system, there is a high probability that a node is either down or that there is a network partitioning
- Partition tolerance
  - Ensures that write and read operations are redirected to available replicas when segments of the network become disconnected

Eventual Consistency

- Eventual Consistency
  - When no updates occur for a long period of time, eventually all updates will propagate through the system and all the nodes will be consistent
  - For a given accepted update and a given node, eventually either the update reaches the node or the node is removed from service
- BASE (Basically Available, Soft state, Eventual consistency) properties, as opposed to ACID
  - Soft state: copies of a data item may be inconsistent
  - Eventually Consistent – copies becomes consistent at some later time if there are no more updates to that data item
  - Basically Available – possibilities of faults but not a fault of the whole system

NoSQL databases

- Not Only SQL
- Common features:
  - non-relational
  - usually do not require a fixed table schema
  - horizontal scalable
  - mostly open sources
- More characteristics
  - relax one or more of the ACID properties (CAP theorem)
  - replication support
  - easy API (if SQL, then only its very restricted variant)
- Do not fully support relational features
  - no join operations (except within partitions),
  - no referential integrity constraints across partitions.

What kinds of NoSQL

- NoSQL solutions fall into two major areas:
  - Key/Value or 'the big hash table'.
    - Amazon S3 (Dynamo)
    - Voldemort
    - Scalaris
  - Schema-less which comes in multiple flavors, column-based, document-based or graph-based.
    - Cassandra (column-based)
    - CouchDB (document-based)
    - Neo4J (graph-based)
    - HBase (column-based)

Common Advantages

- Cheap, easy to implement (open source)
- Data are replicated to multiple nodes (therefore identical and fault-tolerant) and can be partitioned
  - Down nodes easily replaced
  - No single point of failure
- Easy to distribute
- Don't require a schema
- Can scale up and down
- Relax the data consistency requirement (CAP)

Less Relation Operations

- joins
- group by
- order by
- ACID transactions
- SQL as a sometimes frustrating but still powerful query language
- easy integration with other applications that support SQL

Key/Value

Pros:

- very fast
- very scalable
- simple model
- able to distribute horizontally

Cons:

- many data structures (objects) can't be easily modeled as key value pairs

Schema-Less

Pros:

- Schema-less data model is richer than key/value pairs
- eventual consistency
- many are distributed
- still provide excellent performance and scalability

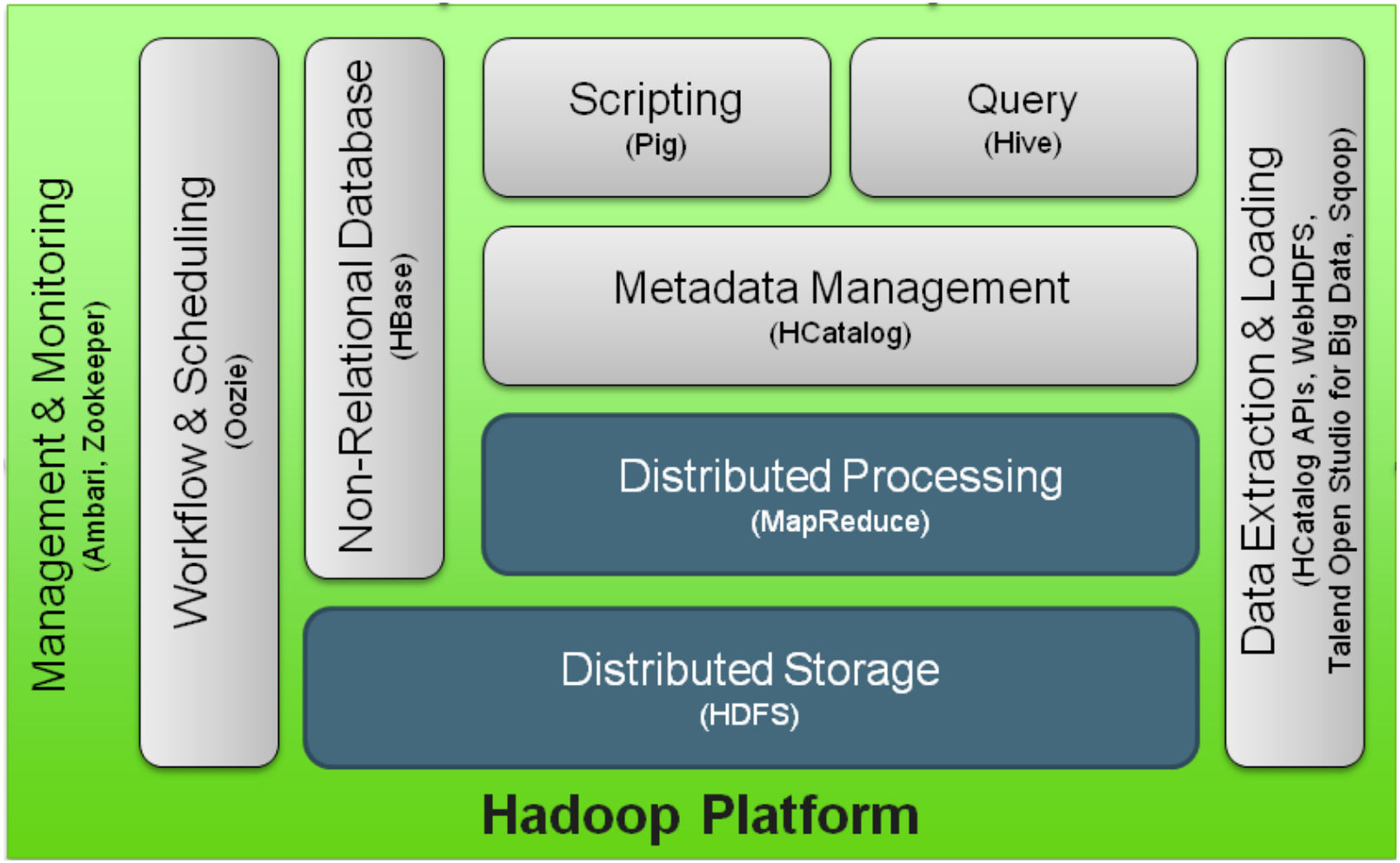
Cons:

- typically no ACID transactions or joins

빅데이터 기술 – NoSQL 솔루션

데이터 모델	특징	솔루션
키-값	• 키와 바이너리 타입의 값을 저장소에 저장하는 구조 • 가장 단순한 데이터 모델	Dynamo, Voldemort, Tokyo Cabinet, Redis
컬럼	• 관계형 데이터베이스와 유사 • 스키마가 존재 • 컬럼에 데이터 저장	Google Bigtable, HBase, Cassandra, Hypertable, Cloudata
문서	• 문서 단위 데이터 저장 • 문서 내에 여러 개의 필드와 대응 값 존재	MongoDB, CouchDB
그래프	• 그래프의 노드와 엣지 저장 • 그래프 탐색 모델 제공	Neo4j, FlockDB, InfiniteGraph

HADOOP ECOSYSTEM



HADOOP - ECOSYSTEM

- **Apache Spark (2010-)**
  - Matei Zaharia et al. Spark: Cluster Computing with Working Sets,. HotCloud 2010.
  - Matei Zaharia et al. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. NSDI 2012.
  - http://spark.apache.org/
  - Resilient Distributed Dataset (RDD)
  - RDD operations
    - MapReduce-like parallel operations
  - DAG of execution stages and pipelined transformations
  - Simple collectives: broadcasting and aggregation

Conclusions (1)

- NoSQL database cover only a part of data-intensive cloud applications (mainly Web applications).
- Problems with cloud computing:
  - SaaS applications require enterprise-level functionality, including ACID transactions, security, and other features associated with commercial RDBMS technology, i.e. NoSQL should not be the only option in the cloud.
  - Hybrid solutions:
    - Voldemort with MySQL as one of storage backend
    - deal with NoSQL data as semistructured data

      ⇒ integrating RDBMS and NoSQL via SQL/XML

Conclusions (2)

- Next generation of highly scalable and elastic RDBMS: NewSQL databases they are designed to scale out horizontally on shared nothing machines,
  - still provide ACID guarantees,
  - applications interact with the database primarily using SQL,
  - the system employs a lock-free concurrency control scheme to avoid user shut down,
  - the system provides higher performance than available from the traditional systems.
- Examples: MySQL Cluster (most mature solution), VoltDB, Clustrix, ScalArc, …