KOREA UNIVERSITY
DATABASE LAB

Big Data Processing

고려대학교 컴퓨터학과
정 연 돈

ydchung@korea.ac.kr

목차

- Big Data와 Cluster Computing
- NoSQL
- Apache Hadoop
- Apache Spark

Big Data 와 Cluster Computing

빅데이터란 ?

- 빅데이터(Big Data)의 정의
  - 기존 컴퓨팅 기술로는 처리가 불가능했던, 매우 규모가 큰 데이터
  - 빅데이터를 기반으로 이루어지는 기술, 경제/경영, 사회적 작업들

- 빅데이터의 예
  - 웹 페이지, SNS 대화, 이메일, 스마트폰 사용자 위치, 각종 네트워크 및 시스템 로그
  - 인간 유전자 정보, 화학 물질 구조 정보, 우주 전파 수신 기록, 가속기 실험 기록
  - 소비자 구매 기록, 영업/판매 실적

빅데이터 분야들

- 빅데이터 처리
  - Big Data processing platforms
    - data storage; data/stream processing; distributed computing
    - Google PageRank, Google Trends, …
- 빅데이터 분석
  - Big Data mining; Visualization
  - Data Science
    - Personalized Medicine using Genomics, Opinion Mining, Social Recommendations, …
- 빅데이터 인공지능
  - Big Data machine learning
    - Google Translate, …
    - AlphaGo ?

빅데이터 처리의 어려움

단위	크기	의미
Bit (b)	1 or 0	Short for "binary digit, after the binary code (1 or) computers use to store and process data
Byte (B)	8bits	Enough information to create an English letter or number in computer code. It is the basic unit of computing
Kilobyte (KB)	1024, or 2^{10} bytes	From "thousand" in Greek. One page of typed text is around 2KB
Megabyte (MB)	1024KB, or 2^{20} bytes	From "large" in Greek. The complete works of Shakespeare total 5MB. A typical pop song is about 4MB.
Gigabyte (GB)	1024MB, or 2^{30} bytes	From "giant" in Greek. A two-hour film can be compressed into 1-2 GB.
Terabyte (TB)	1024GB, or 2^{40} bytes	From "monster" in Greek. All the catalogued books in America's Library of Congress total 15TB
Petabyte (PB)	1024TB, or 2^{50} bytes	A 1PB MP3 file would take 2000 years to play
Exabyte (EB)	1024PB, or 2^{60} bytes	5 EB is equivalent to all words ever spoken by human beings
Zettabyte (ZB)	1024EB, or 2^{70} bytes	Expected annual global IP traffic is around 2ZB in 2019
Yottabyte (YB)	1024YB, or 2^{80} bytes	Currently, too big to imagine

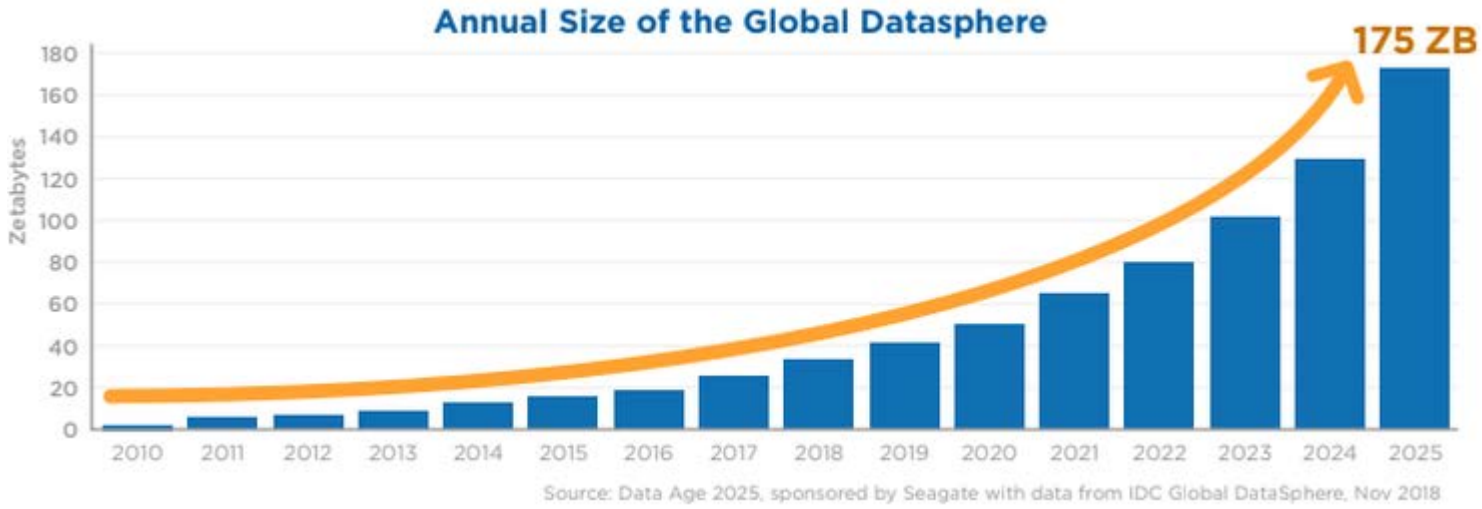
데이터의 단위

빅데이터 처리의 어려움

Seagate Barracuda XT 2TB SATA 6Gb/s
평균 읽기 속도: 111.7MB/s
평균 쓰기 속도: 93.7MB/s

스캔 시간	
데이터 크기	시간
1 Terabytes	2h 36m
1 Petabytes	111 days

빅데이터 처리의 어려움

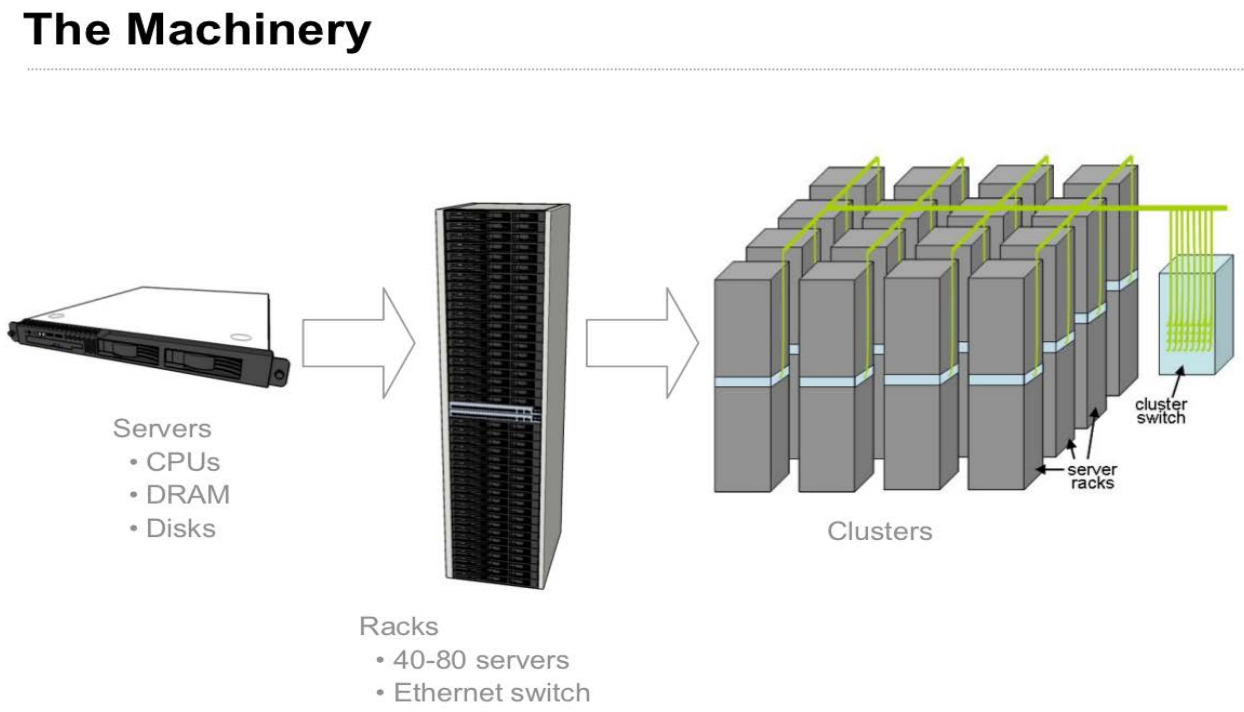


- 데이터의 폭발적 증가

- 복잡한 연산 비용
  - 분석을 위해 단순 스캔보다 훨씬 복잡한 연산을 수행
  - 하나의 기계에서 연산 실행 시 길게는 수 주일의 시간이 걸리는 경우도 있음

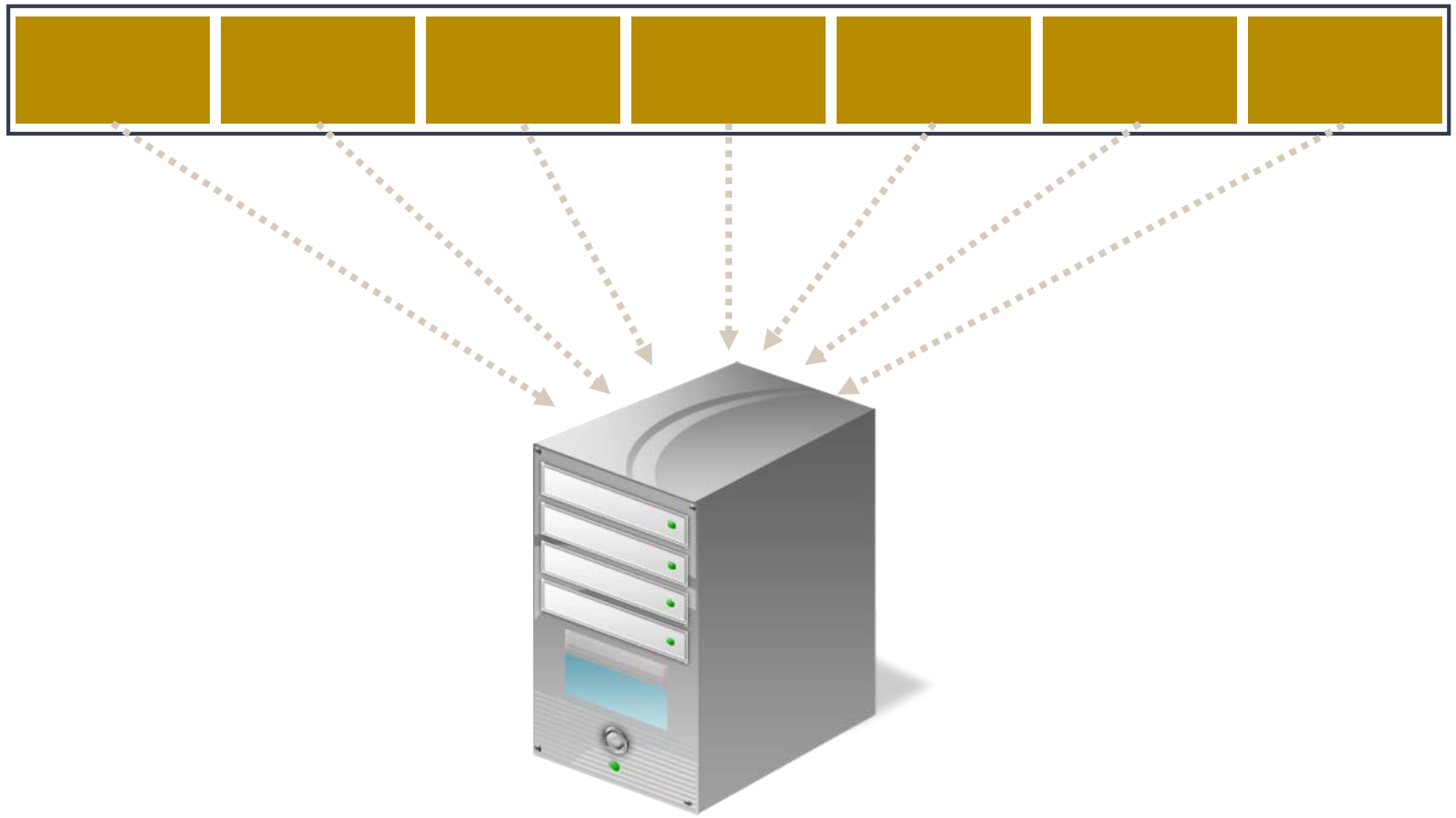
클러스터 컴퓨팅

- 클러스터 컴퓨팅 (Cluster computing)이란 ?
  - 다수의 범용 컴퓨터를 활용하는 분산/병렬 컴퓨팅 기술
    - '비공유(shared-nothing), 분산(distributed) 서버 클러스터'



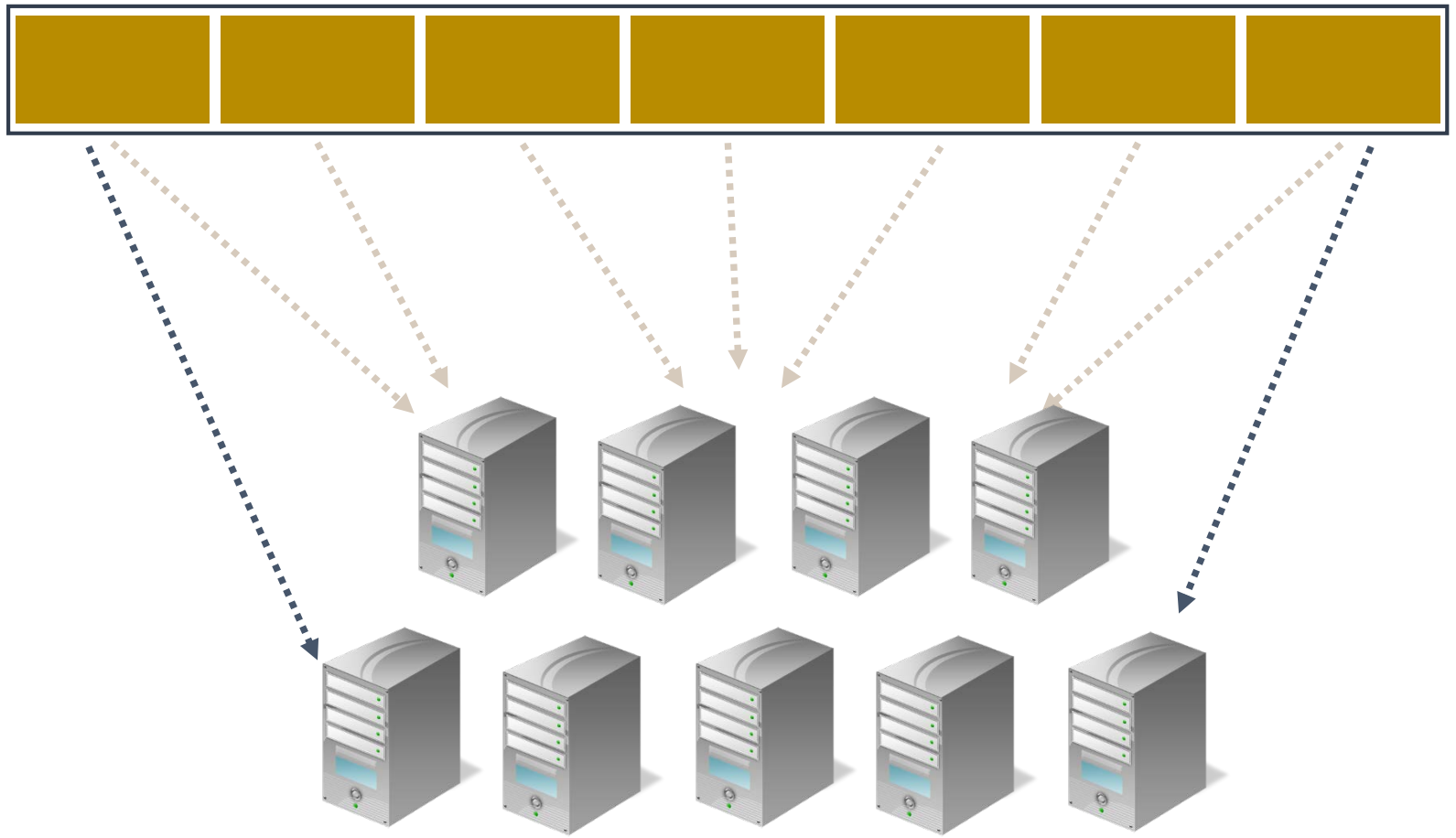
클러스터 컴퓨팅

- 기존의 작업 처리 방식

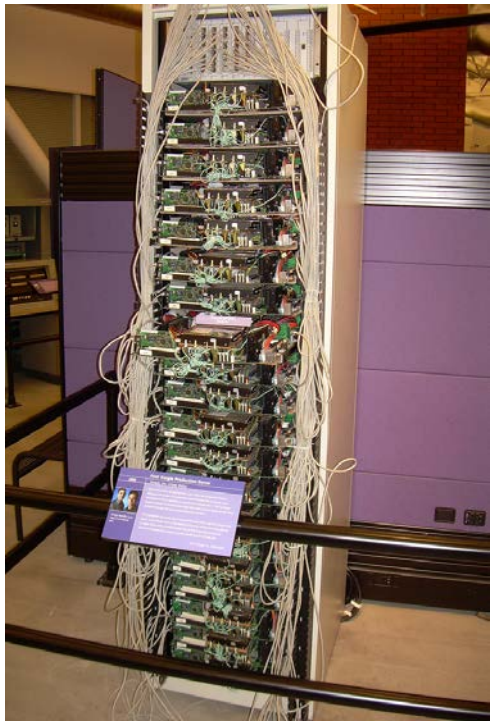


클러스터 컴퓨팅

- 클러스터에서 작업 처리 방식



클러스터 컴퓨팅 vs. 슈퍼컴퓨팅



Commodity Server

Unit Cost: $6,500
CPU: 2.9-GHz Xeon Hexa-core CPU x 2
144G RAM. 12TB disk

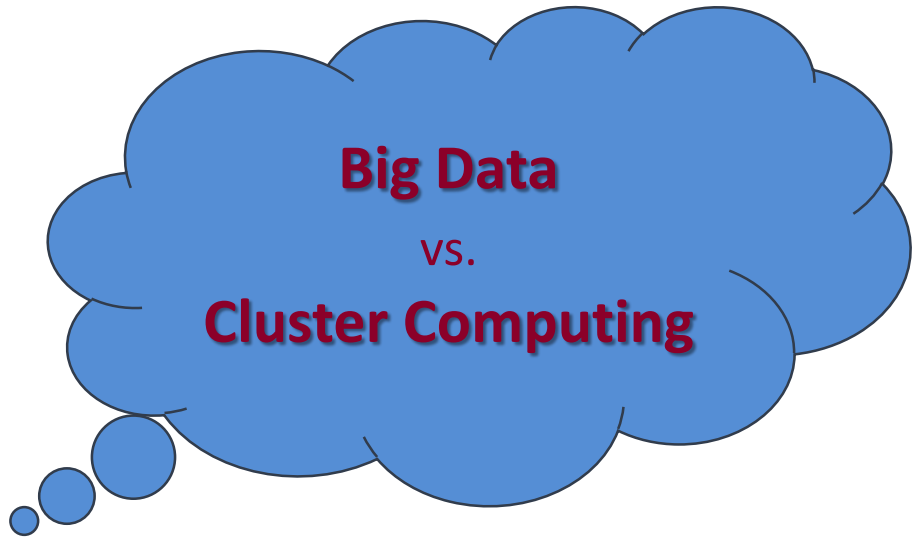
Cost: $6,500 X 100 = $650,000
CPU x 200, 14TB RAM, 1.2PB HDD

High End Server: Cisco UCS C480 M5 Server

Unit Cost: $ 1,000,000
CPU: 2.7-GHz Xeon Scalable CPU x 4
6TB RAM, 14 TB HDD

클러스터 컴퓨팅의 특징

- 확장성 및 유연성
  - 처리 능력의 동적 확장 및 축소가 쉬움
  - 초기 투자 비용의 절감
- 내고장성
  - 분산 저장 및 처리와 복제
  - 단순한 분산/병렬 처리 제어
- 경제성
  - 범용(저가) 서버 사용
  - Open-source SW 시스템 활용
- 데이터 중심 분산/병렬 방식
  - 빅데이터 처리에 적합 !



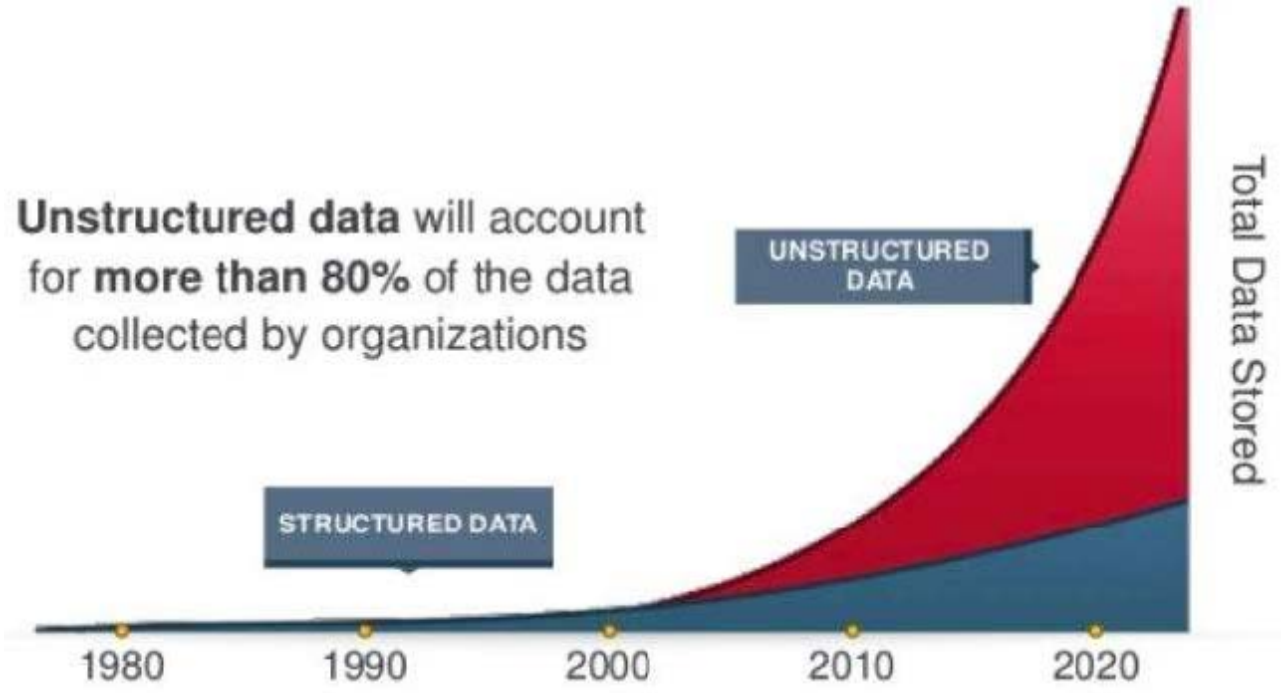
NoSQL

- 배경설명
- NoSQL 과 분산시스템
- 분산시스템 설계 이슈

Big Data 시대의 도래

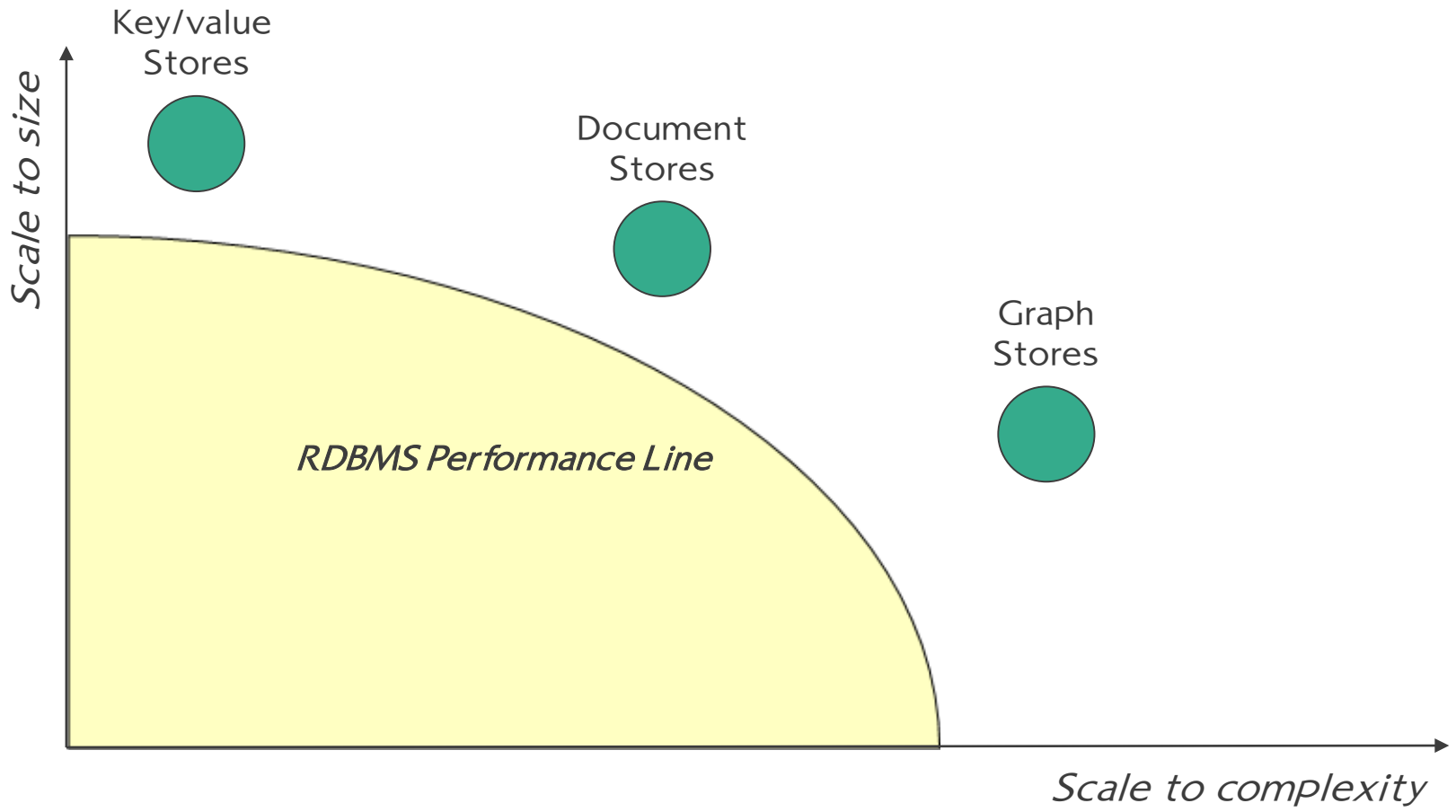
- Really big data
  - Currently, data held by Google is deduced to be about 10-15 Exabytes

- Complex data
  - Structured data
  - Semi-structured data
    - Key/value, document, graph, …
  - Unstructured data



RDBMS의 한계

- Traditional RDBMSs are not sufficient for big data processing.



RDBMS의 한계

- Various advantages of RDBMSs
  - Efficient, reliable, convenient, multi-user support
- But, not every data management/analysis problem is best solved by using traditional DBMSs.
  - Expensive to setup
  - ACID
    - Good, but significantly degrade the performance
  - Excessive functionality
    - Not always useful in every application
- A new system is required.
  - Sufficiently scalable to process large data
  - Sufficiently efficient to process complex data

NoSQL

- **NoSQL**
  - Use the right system for the right job*!*
  - NoRel (No Relational, Not only Relational)
  - SQL (혹은 관계형 모델)이 모든 경우에 대한 정답이 아님*!*

- 응용들의 다양한 환경, 데이터 특성, 처리 기능/목적에 따라 다른 시스템을 사용할 수 있음
  - 비관계형 모델
    - Key/value model, document model, graph model, …
    - Schema-free
    - NOT use SQL anymore
  - 기능적 차별성
    - Quicker/cheaper to set up
    - Massive scalability

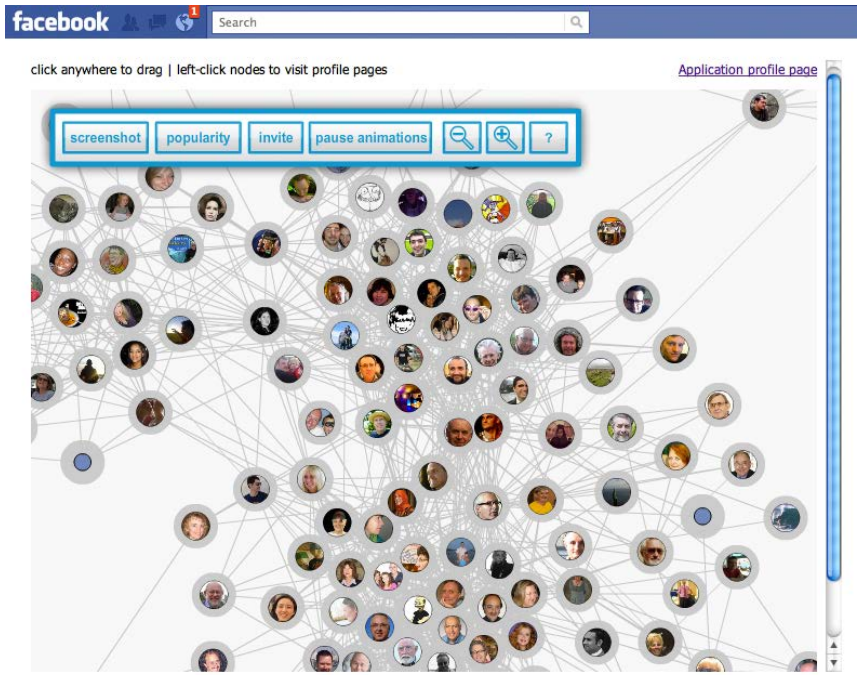
Example #1: Web Log 분석

- Web log format
  - Timestamp, connection time, client IP, cache result, HTTP code, response size, HTTP method, request address, user ID, hierarchical data, host address, content type
- Example query
  - Find average connection time of each user accessing 'youtube.com'.

```
1366211076.021 137 118.68.154.66 TCP_MISS/200 219 GET http://hcvvo.333337.com/menu/Menu_Data.aspx?hidMenuType=0&hidForce=false&hidMenuMainSportKey=444fd2875da0c56b6c698071126a3748&hidMenuOtherSportKey=a7ee4e8eb35253df457d4802180861548&_=1366211072116 - DIRECT/hcvvo.333337.com text/html
1366211076.025 3396 95.104.116.226 TCP_MISS/200 188623 GET http://www.adjarabet.com/skins/adjarabet/images/play-now-big-ka.png - DIRECT/www.adjarabet.com image/png
1366211076.027 91 116.19.167.246 TCP_MISS/200 462 GET http://g.qzone.qq.com/fcg-bin/cgi_emotion_list.fcg?uin=308661432 - DIRECT/g.qzone.qq.com text/html
1366211076.030 0 199.36.73.157 TCP_ERROR/400 3032 GET - - NONE/- -
1366211076.031 6600 171.252.59.3 TCP_REFRESH_HIT/200 64416 GET http://duhm5.i111888.com/commJS/LeftAllInOne.js?v=201301250500 - DIRECT/duhm5.i111888.com application/x-javascript
1366211076.032 0 116.14.113.14 TCP_ERROR/502 3026 GET http://127.0.0.1:10516/gui/pingimg&r=0.8239114470779896 - DIRECT/127.0.0.1 -
1366211076.034 18 58.143.34.100 TCP_MISS/200 495 POST http://terms.naver.com/comments/get_ticket_config.nhn - DIRECT/terms.naver.com text/html
1366211076.036 671 27.73.21.84 TCP_MISS/200 4505 GET http://xz90j.ibet888.net/UnderOver_data.aspx?Market=l&Sport=1&DT=&RT=U&CT=04%252F17%252F2013+11%253A04%253A07&Game=0&OrderBy=0&OddsType=4&k462624908=v462625179&key=Wl5UXV5VQVpEdmhlfwVnGWNwaV1HXUFIWUMcN1ArITBAGgBqGh9bXwoUe14BCAMFX0JeXwwfS1EG%250ASwIWFw1XVldeGgAfSQMDAlpVWVpbWVQ%253DEIIQ&_=1366211074890 - DIRECT/xz90j.ibet888.net text/html
1366211076.036 150 58.186.165.245 TCP_MISS/200 779 POST http://la1k1peng71y.asia.sbo333.com/webroot/restricted/bet-list/bet-list-mini-data.aspx - DIRECT/la1k1peng71y.asia.sbo333.com text/html
1366211076.036 374 27.42.86.172 TCP_MISS/200 2094 GET http://v.youku.com/player/getPlayList/VideoIDS/XNTQzOTY0MDg4 - DIRECT/v.youku.com text/html
```

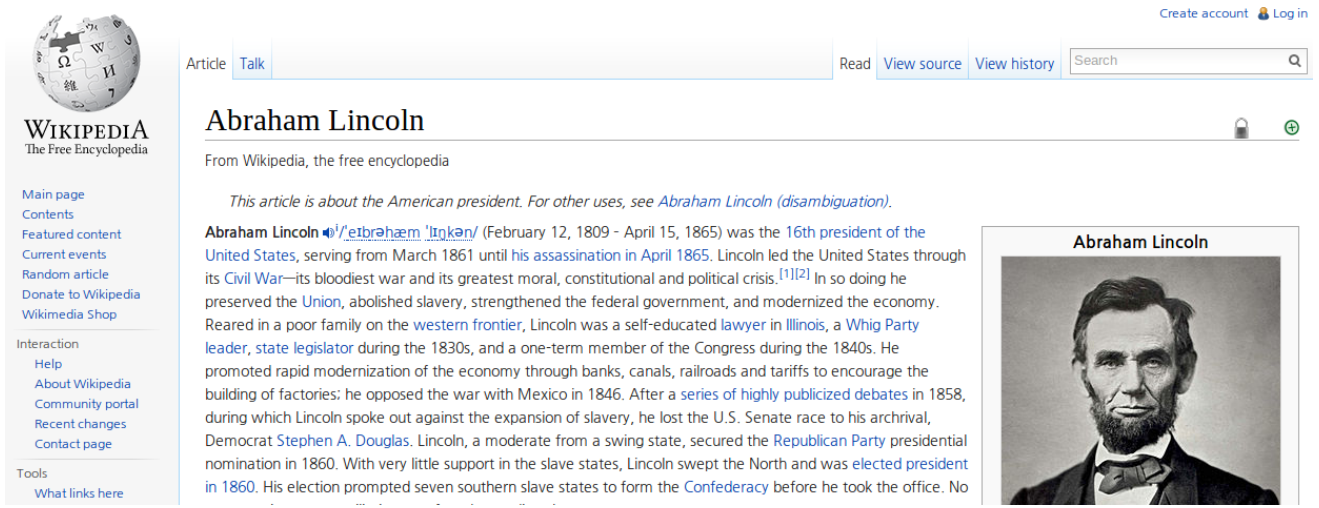
Example #2: Social Network 분석

- Social network data
  - UserID, friend list (UserID, UserID, …)
- Example query
  - Find all friends of friends of friends of … friends of a given user.



Example #3: Wikipedia 페이지 분석

- Wikipedia data
  - Large collection of documents (total 50,101,543 articles, 201,684,815 pages, April 2019)
  - Combination of structured and unstructured data
- Example query
  - Retrieve introductory paragraph of all pages about U.S. presidents before 1900.



NoSQL 시스템

- (일반적으로) NoSQL 시스템은 "빠른 응답시간(low latency)"과 "높은 확장성 (high scalability)"을 지향
  - 클러스터기반 분산 시스템 구조
  - 분산/병렬성으로 확장성과 응답시간 구현

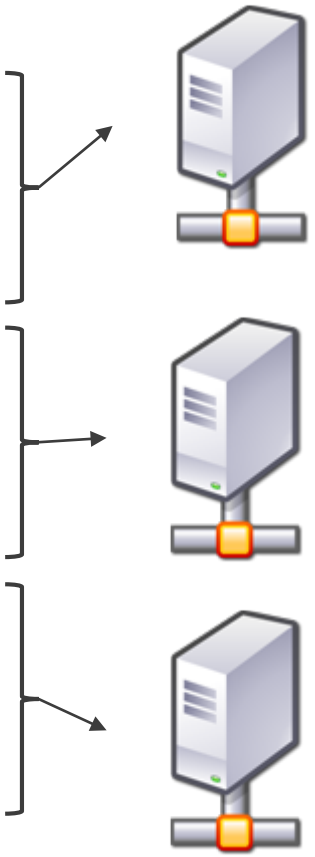
NoSQL == Distributed Systems ?

NoSQL == Big Data Systems ?

분산 시스템 (Distributed Systems)

- 데이터 저장 (Storing data)
  - Data are partitioned by a fixed size of shards, and distributed to multiple machines.

Timestamp	User ID	Host address	Additional Info
1366296049253	1	http://google.com	...
1366296049255	2	http://facebook.com	...
1366296049258	3	http://youtube.com	...
1366296049262	4	http://naver.com	...
1366296049262	5	http://youtube.com	...
1366296049264	6	http://daum.net	...
1366296049265	7	http://youtube.com	...
1366296049268	8	http://google.com	...
1366296049271	9	http://facebook.com	...



분산 시스템 (Distributed Systems)

- 질의 처리 (Query processing)
  - User queries are also distributed and executed in parallel.
  - Ex) Find every user id who visited 'http://youtube.com'.

Timestamp	User ID	Host address	Additional Info
1366296049258	3	http://youtube.com	...
1366296049262	5	http://youtube.com	...
1366296049265	7	http://youtube.com	...

"좋은" 분산 시스템이란 ?

- A system that always provides the high-quality service without errors.
- Design principals
  - **Availability** – 가용성
    - Should always provide the service
  - **Consistency** – 일관성
    - Should provide the service without errors
  - **Latency** – 응답 성능
    - Should provide the high-quality service

Availability (가용성)

- If a user cannot access the system, it is unavailable.
- Formal definition
  - The probability that the system is operating at a specified time t.
  - Ao = (total time – down time) / total time
    - Ao stands for availability, operational.
- Fault-tolerance is important.
  - In distributed systems, more failures occur as clusters become larger.

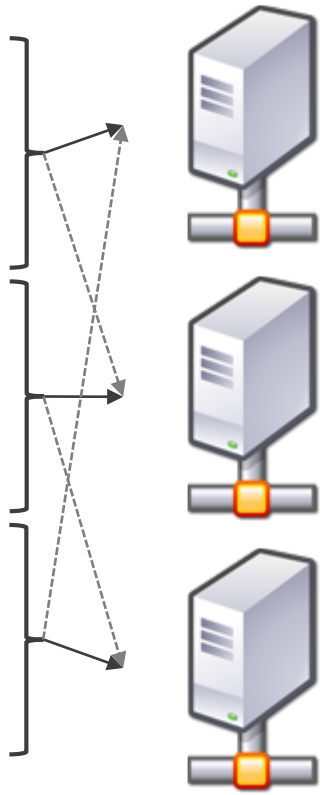
Fault-tolerance (고장허용성)

- To achieve fault-tolerance
  - Elimination of single point of failure
  - Failure detection and recovery
  - Fault containment to prevent propagation of the failure
- Replication
  - Same data are stored on multiple machines.
  - Spare components address the first fundamental characteristic of fault tolerance.

데이터 복제

- Replicating data
  - Store data in two or more different machines

Timestamp	User ID	Host address	Additional Info
1366296049253	1	http://google.com	...
1366296049255	2	http://facebook.com	...
1366296049258	3	http://youtube.com	...
1366296049262	4	http://naver.com	...
1366296049262	5	http://youtube.com	...
1366296049264	6	http://daum.net	...
1366296049265	7	http://youtube.com	...
1366296049268	8	http://google.com	...
1366296049271	9	http://facebook.com	...



Consistency (일관성)

- If all clients see the same data after an update operation, the system is in the consistent state.
- Formal definition
  - The state is consistent if there is no violation of data integrity.
- In distributed systems, preserving consistency requires a high overhead.
  - In general, data are replicated for availability.
  - To preserve consistency, there should be no integrity violation for replicated data.

(대표적인) 일관성 모델들

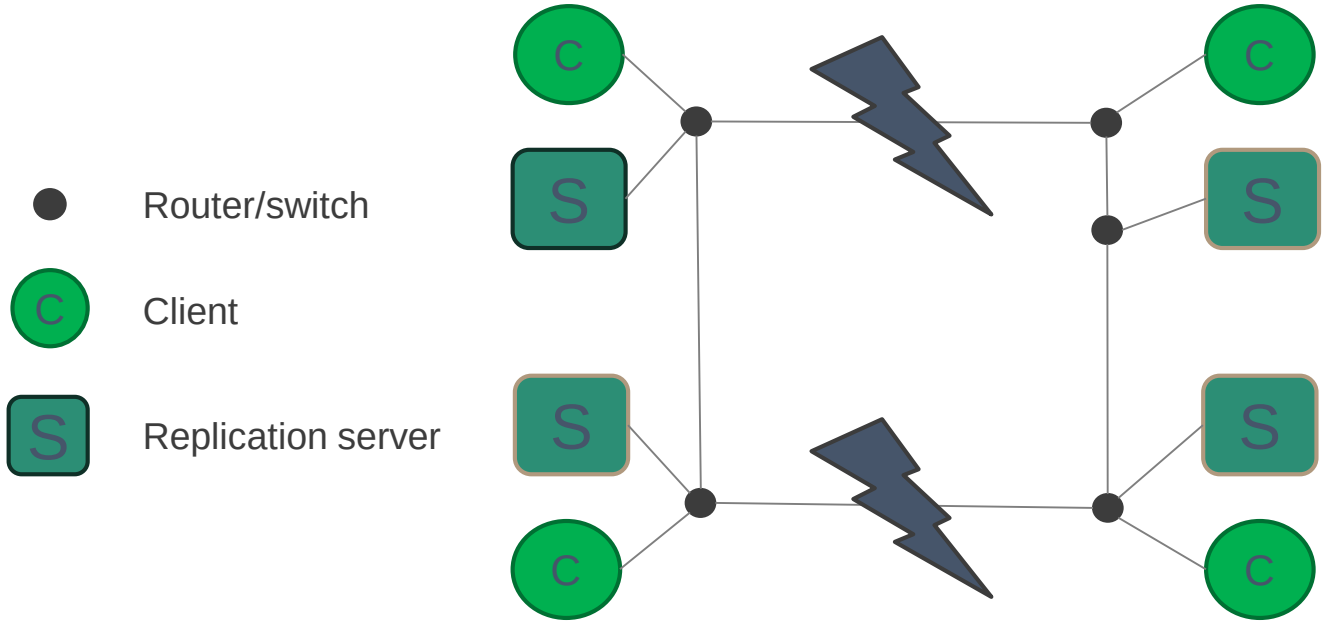
- Strong consistency
  - If a write is successful, any reads than happen after the write would get the same, latest value
- Eventual consistency
  - The system's state can be temporarily inconsistent. But it will become consistent over time, although the system doesn't receive input during that time.
  - Conflict resolution is required.
    - A conflict can occur if two different updates are performed on the replicated data before the system becomes consistent.
    - In order to ensure replica convergence, a system must reconcile differences between multiple copies of distributed data

Latency (응답 성능)

- Response time of a user request
- In distributed systems, there are a lot of factors increasing latency.
  - Message exchange, replication, consensus process, …
  - Failures also increase latency.

Big Problem: 네트워크 단절

- System faults are truly common in large distributed systems.
- Sometimes, a cluster is partitioned due to failures.
  - Participants in a partition cannot communicate with other participants in other partitions



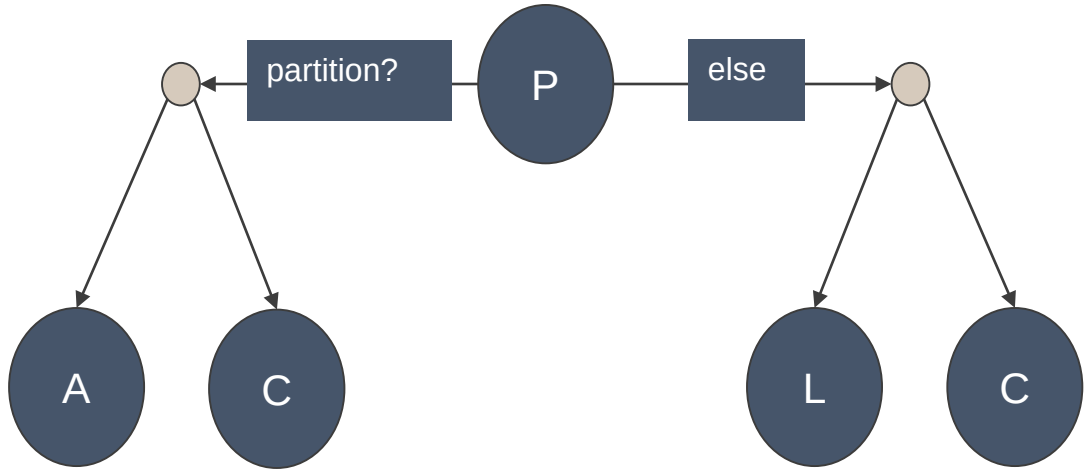
Trade-offs

- Trade-off between availability and consistency
  - In the existence of network partitions, availability is still the key issue.
    - If there are replicas in each partitioned network, consistency can be achieved only after the system is recovered.
    - This makes the system unavailable while recovering from the failure.

- Trade-off between latency and consistency
  - In the absence of network partitions, systems should be always available.
    - Increasing (decreasing) consistency increases (reduces) latency.
    - Note: Data are replicated for the availability.

PACELC Theorem

- Distributed systems should be built with taking the followings into account.
  - If there is a network partition(P), how does the system trade-off availability(A) and consistency(C)?
  - Else(E), how does the system trade off latency(L) and consistency(C)?



"Consistency Tradeoffs in Modern Distributed Database System Design: CAP is Only Part of the Story,"
by Daniel Abadi, *IEEE Computer 45(2): 37-42 (2012)*

BASE

- Emerging design approach for modern systems
- Acronym for
  - **B**asically **A**vailable: the system does guarantee availability.
    - High availability is critical.
  - **S**oft state: the state of the system may change over time, even without input.
    - Because of the eventual consistency model
  - **E**ventually consistent: lower consistency restriction than ACID
- Soft state and eventual consistency are techniques that work well in the presence of partitions.
  - To promote availability

ACID vs. BASE

- High-consistency model vs. Low-consistency model

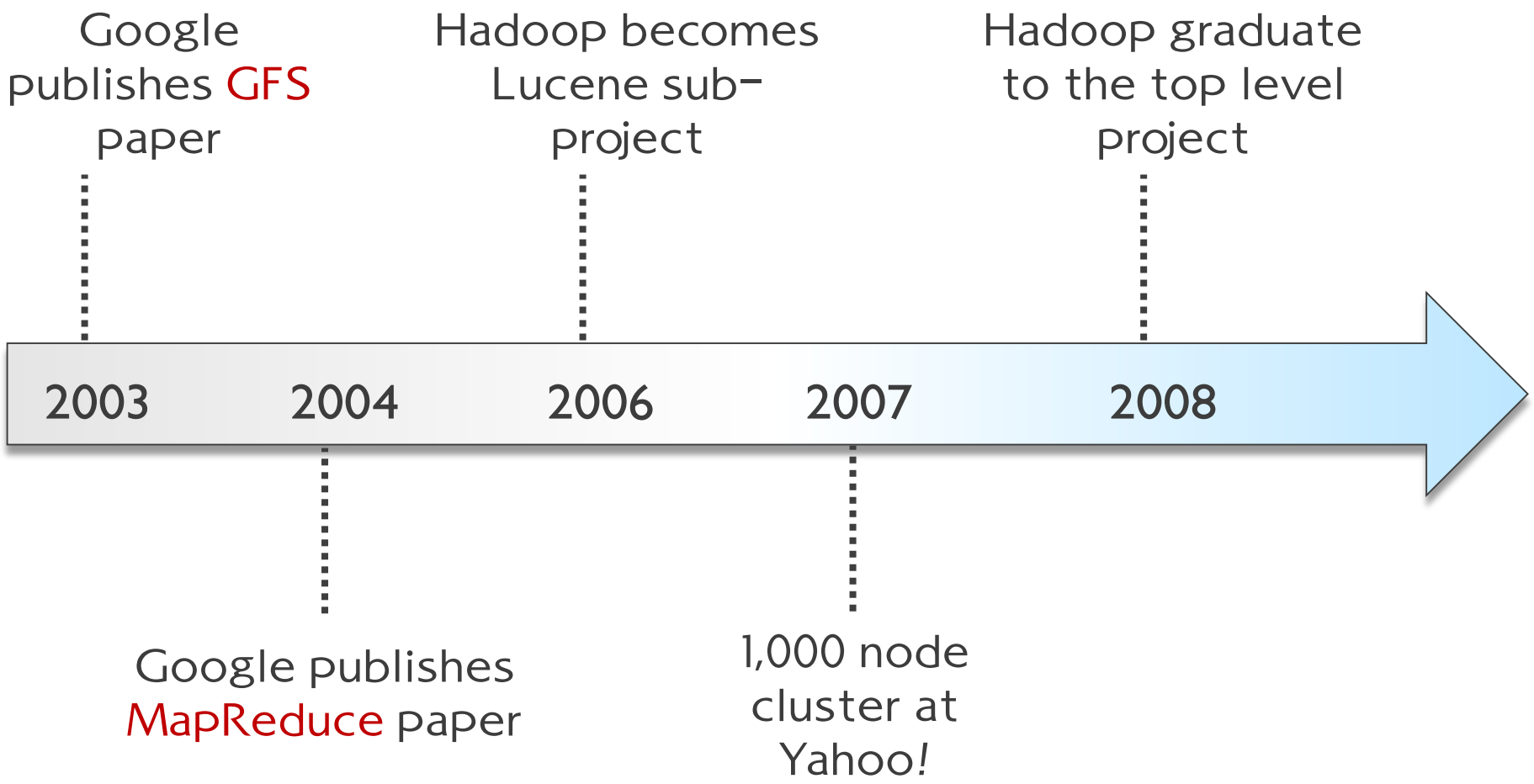
ACID	BASE
Atomicity	Basically Available
Consistency	Soft state
Isolation	Eventual consistency
Durability	

Apache Hadoop

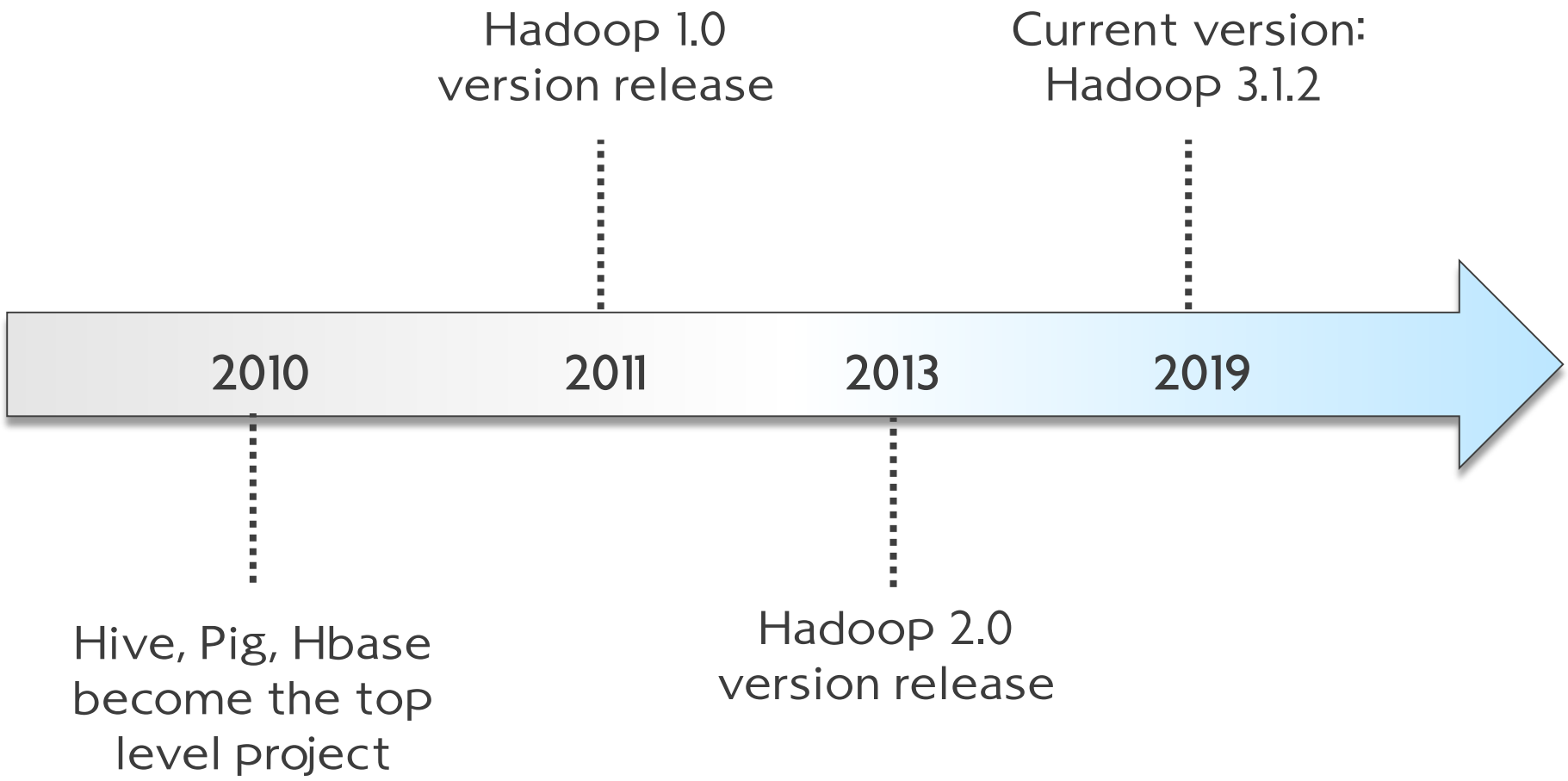
Apache Hadoop

- Open source cloud computing platform
  - The implementation of Google File System(GFS) and MapReduce
    - By Doug Cutting and Mike Cafarella, 2006
- High level language support
  - Java, C++, Python, …
- Scalable, and flexible cluster with commodity servers

The History of Hadoop

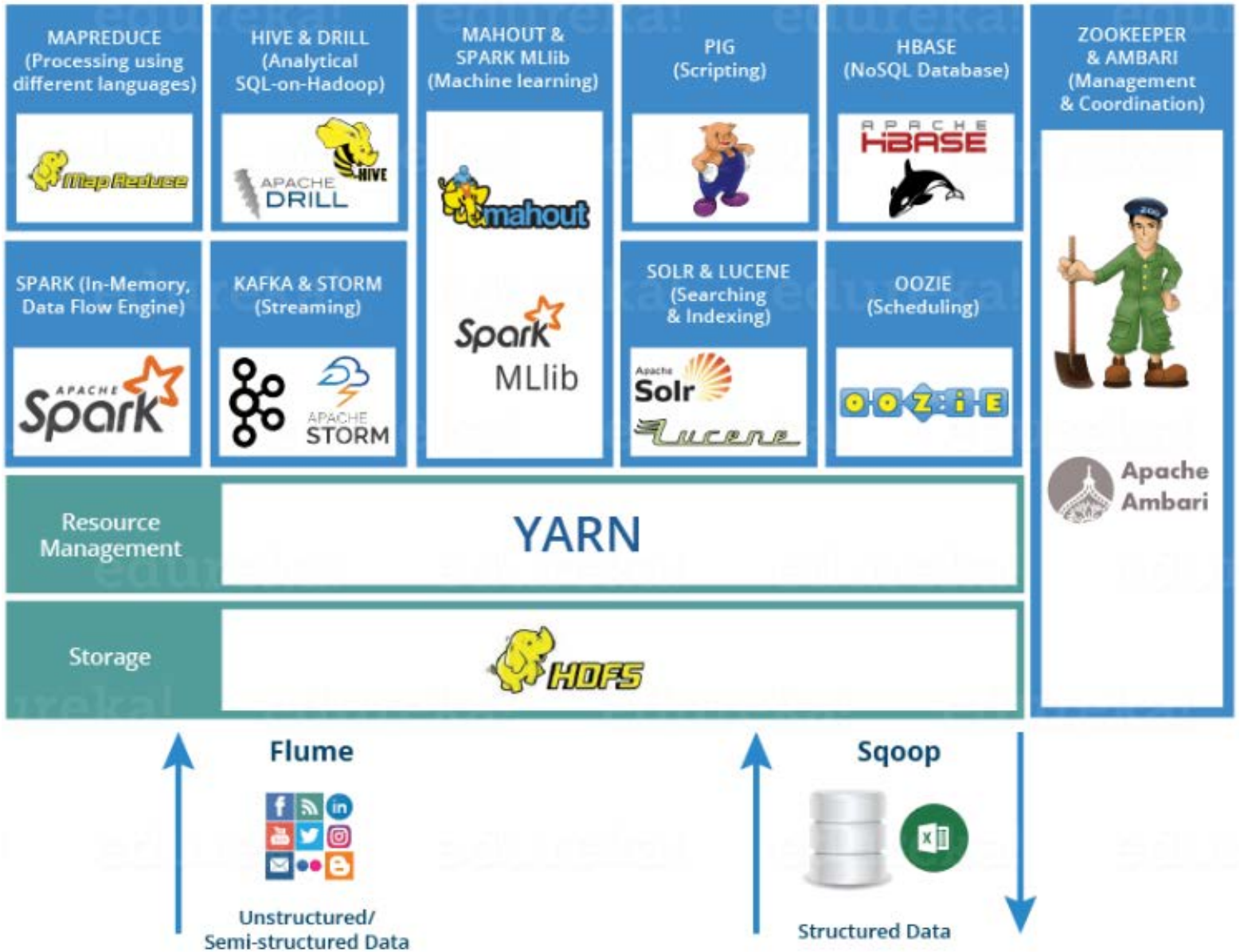


The History of Hadoop



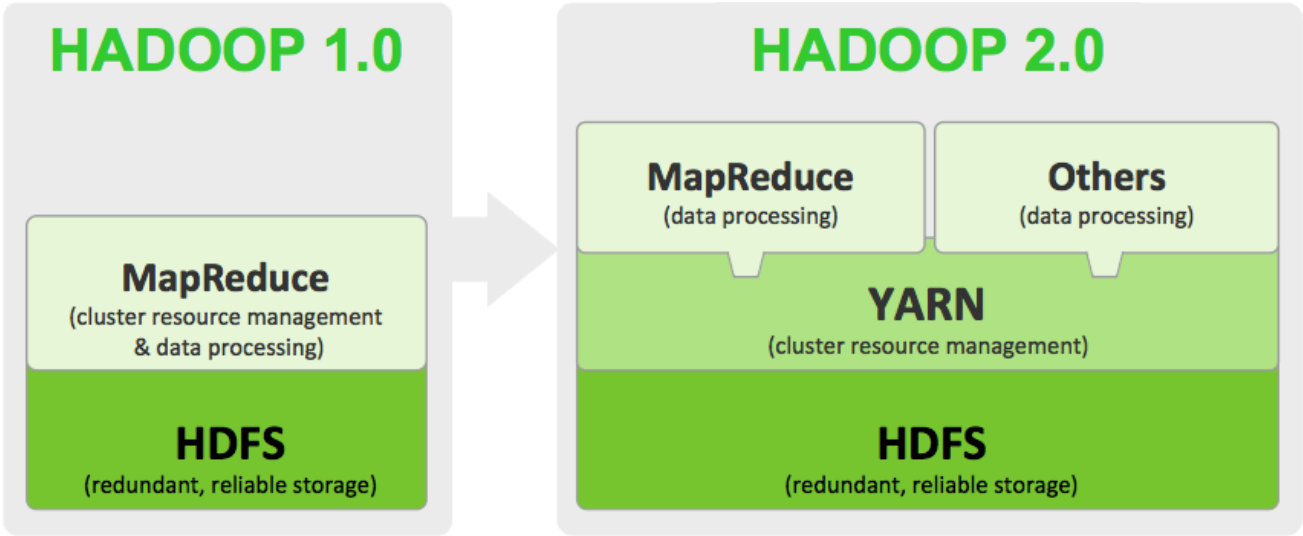
Institutions using Hadoop

Hadoop ecosystems



Hadoop 구조

- Hadoop 1.0
  - HDFS(Hadoop Distributed File System)
  - MapReduce
- Hadoop 2.0
  - HDFS
  - YARN(Yet Another Resource Negotiator)



Hadoop v1.0

HDFS

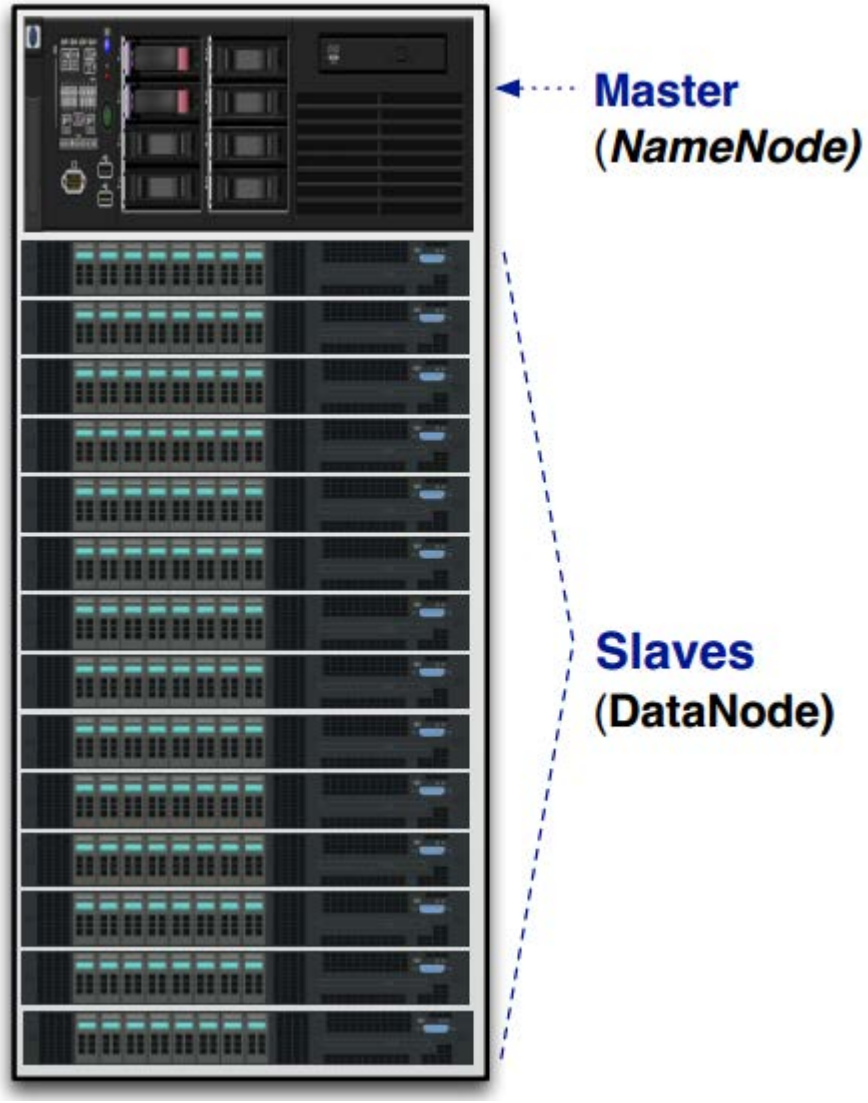
- The implementation of Google File System
  - Written by Java

- HDFS is designed for
  - Storing large files
  - Accessing files with high bandwidth

- Not for
  - Storing small many files
  - Accessing files with low latency

HDFS 구조

- Master-slave
  - Single master - NameNode
    - Provides HDFS meta data
    - Manages status of DataNodes
  - Multiple slaves - DataNode
    - Stores HDFS files

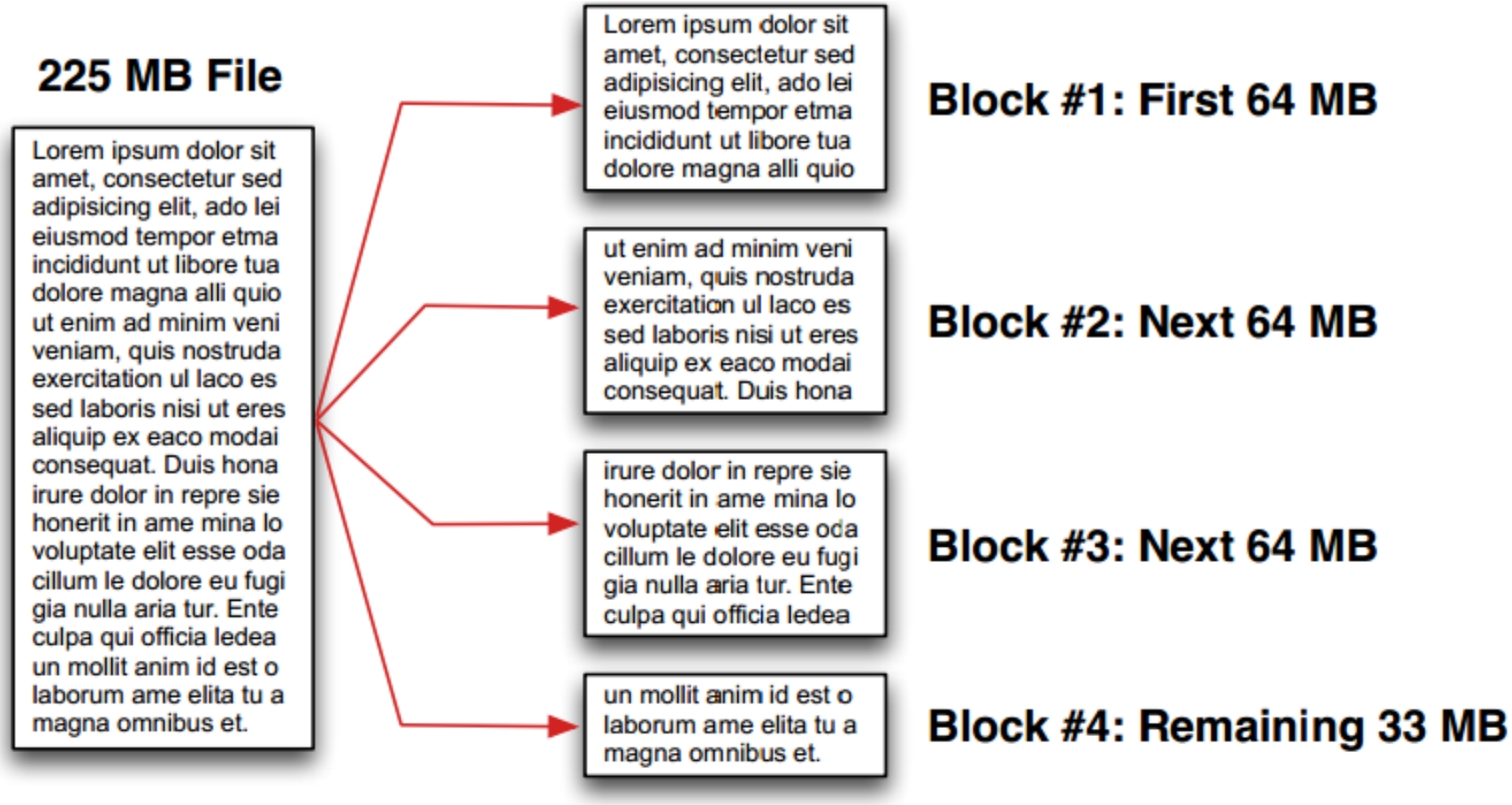


HDFS 파일 저장

- HDFS file is split into HDFS blocks
  - Basic unit of reading/writing access
  - Fixed size (default 128MB) and configurable
  - Making HDFS good for large file and high throughput

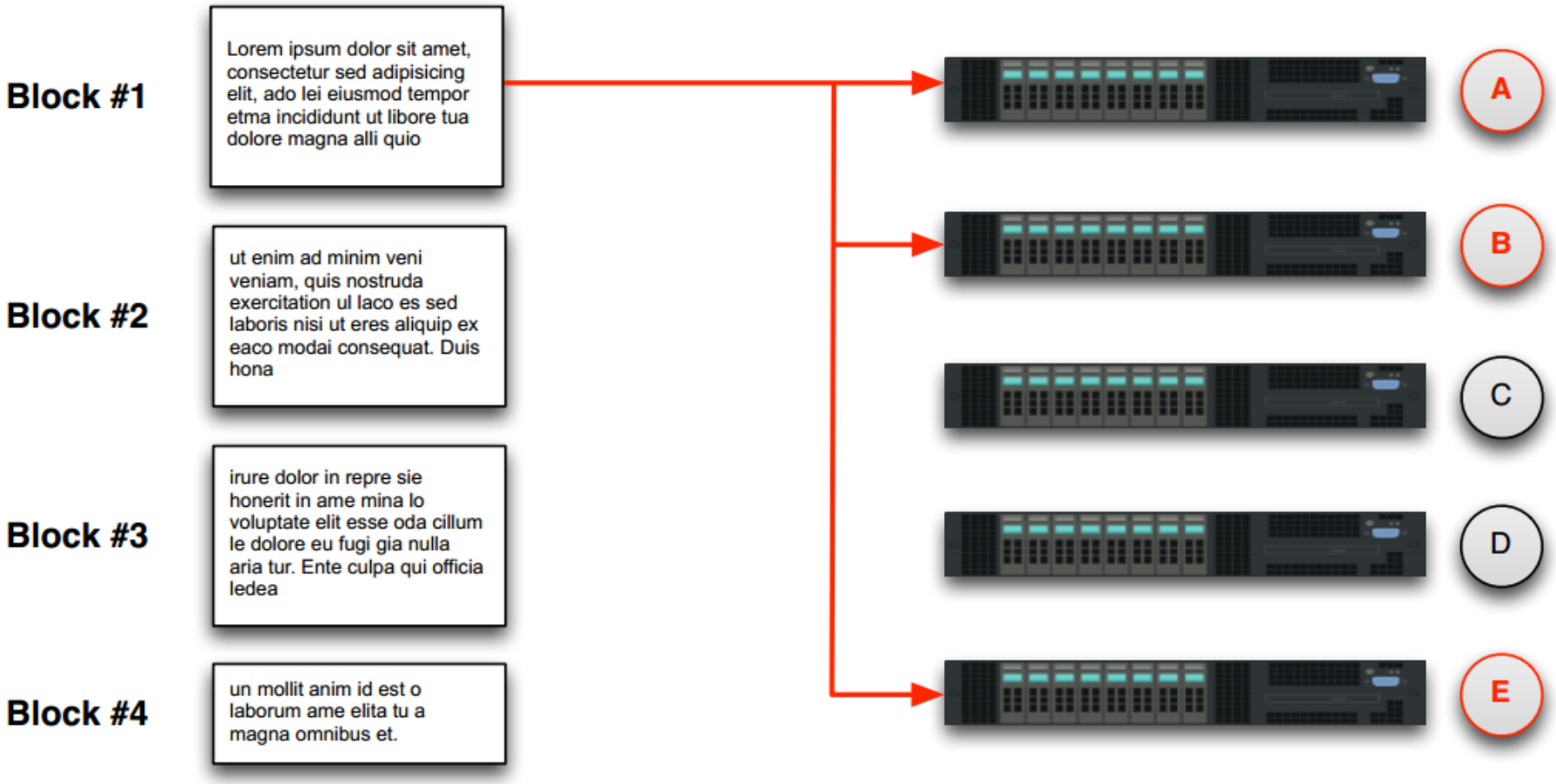
- Block may have multiple replicas
  - One block can be stored over multiple DataNodes
  - Making HDFS storage fault tolerant

HDFS 파일 저장



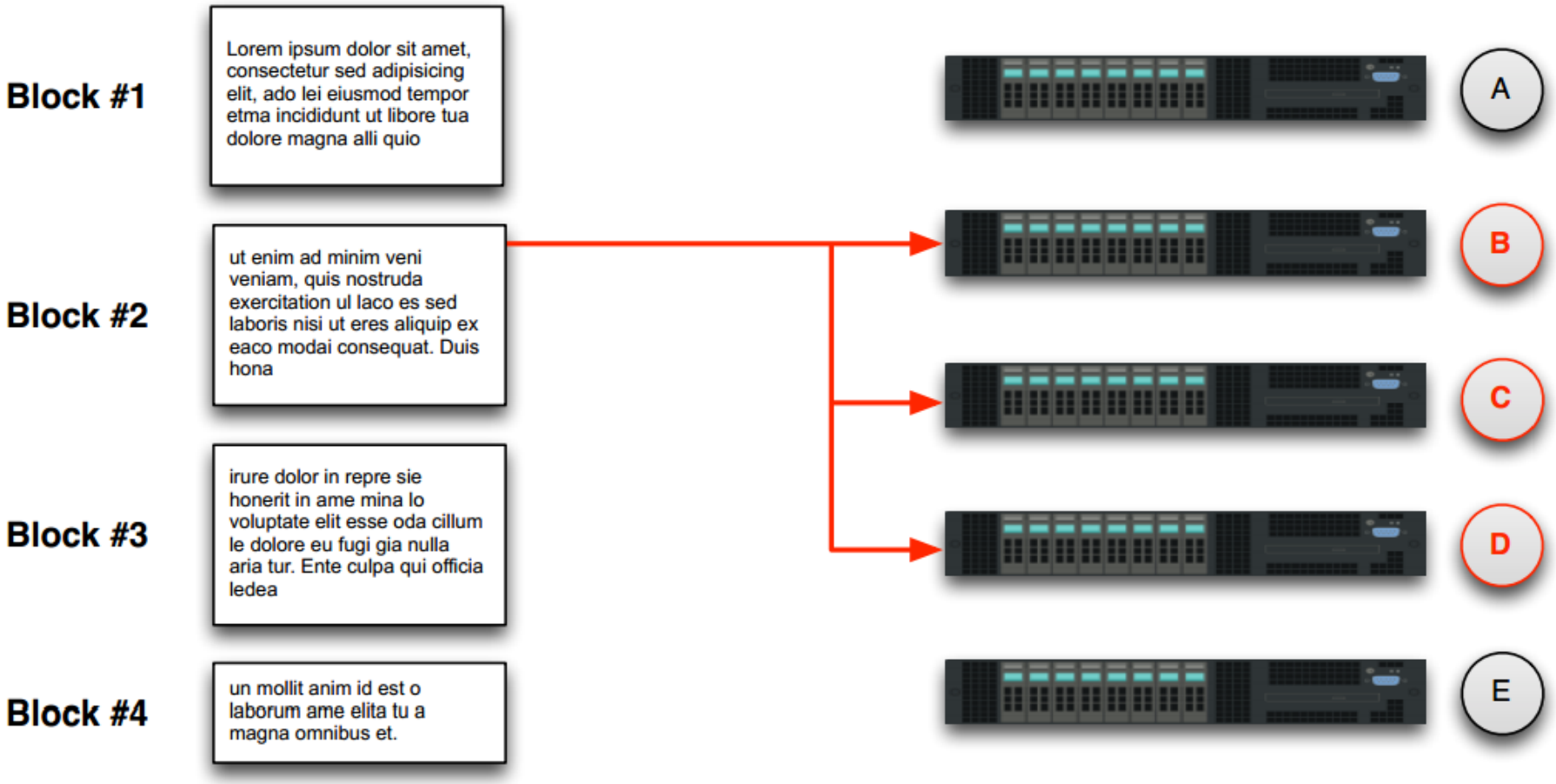
가정: 64MB HDFS block을 사용하는 경우

HDFS 파일 저장



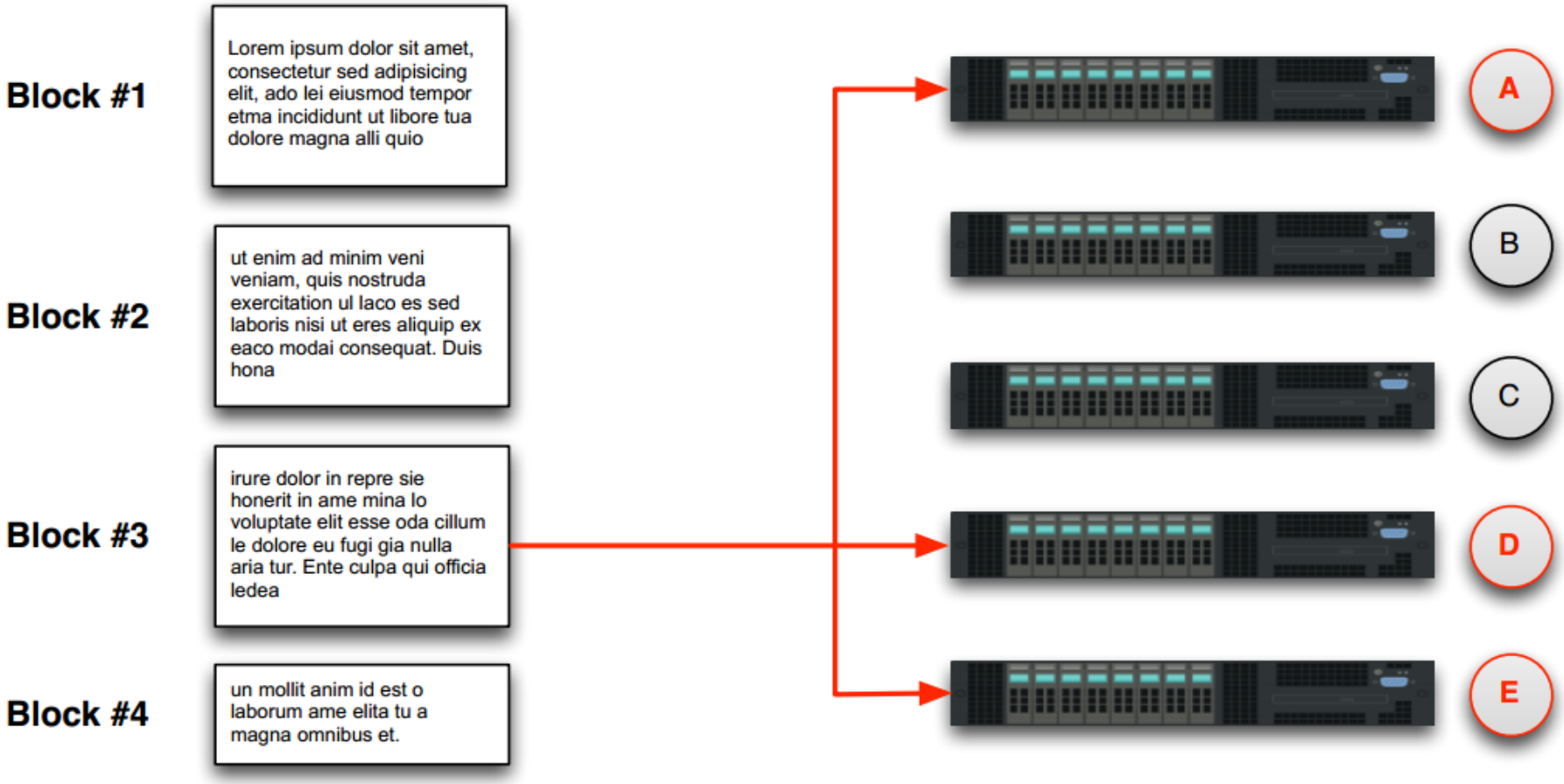
가정: 복제본 개수 3인 경우

HDFS 파일 저장



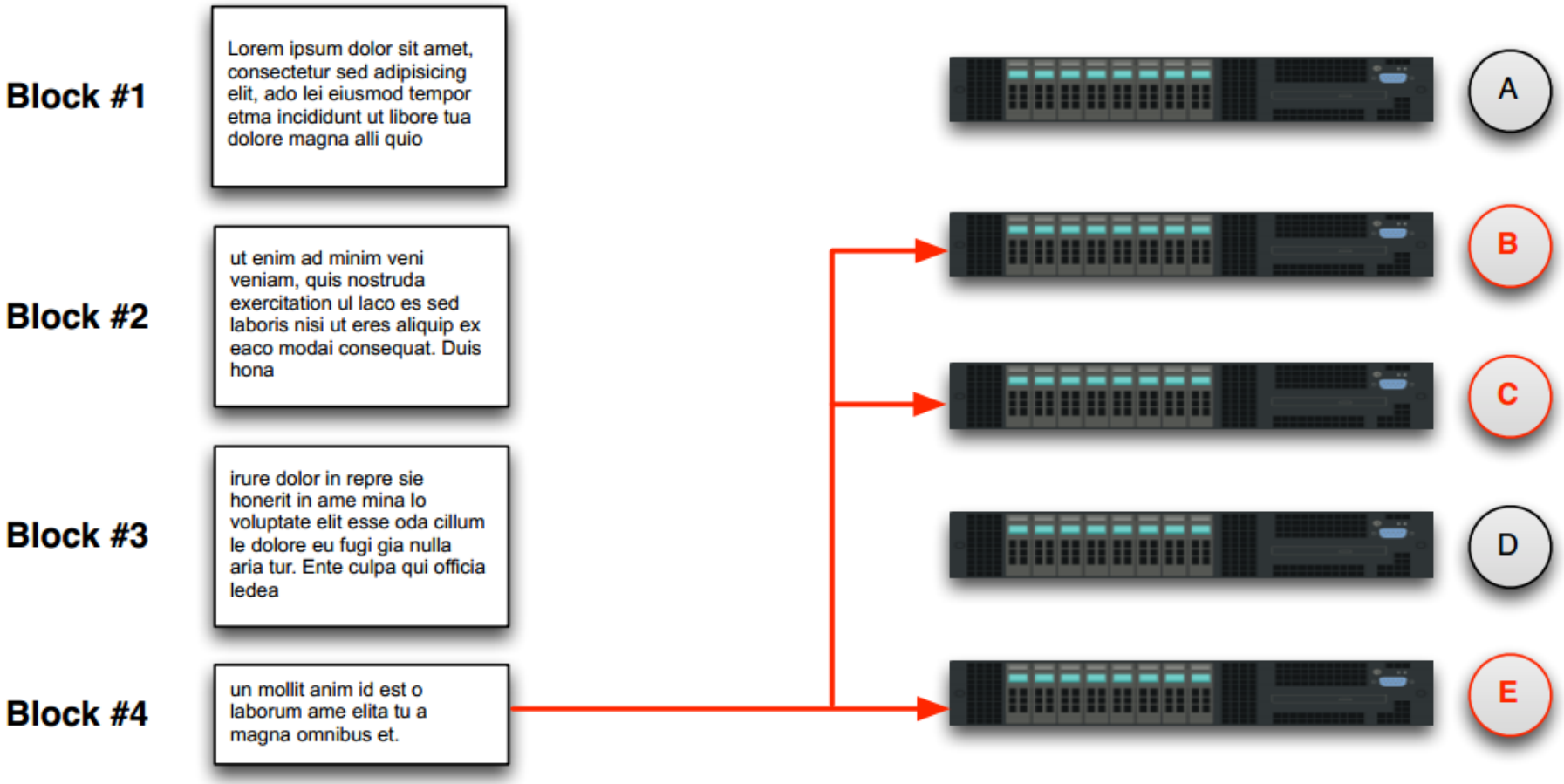
가정: 복제본 개수 3인 경우

HDFS 파일 저장



가정: 복제본 개수 3인 경우

HDFS 파일 저장



가정: 복제본 개수 3인 경우

HDFS 사용 예

- "hadoop fs" command
  - HDFS commands are similar to corresponding UNIX commands

```
$ hadoop fs -ls /user/tomwheeler

$ hadoop fs -cat /customers.csv

$ hadoop fs -rm /webdata/access.log

$ hadoop fs -mkdir /reports/marketing
```

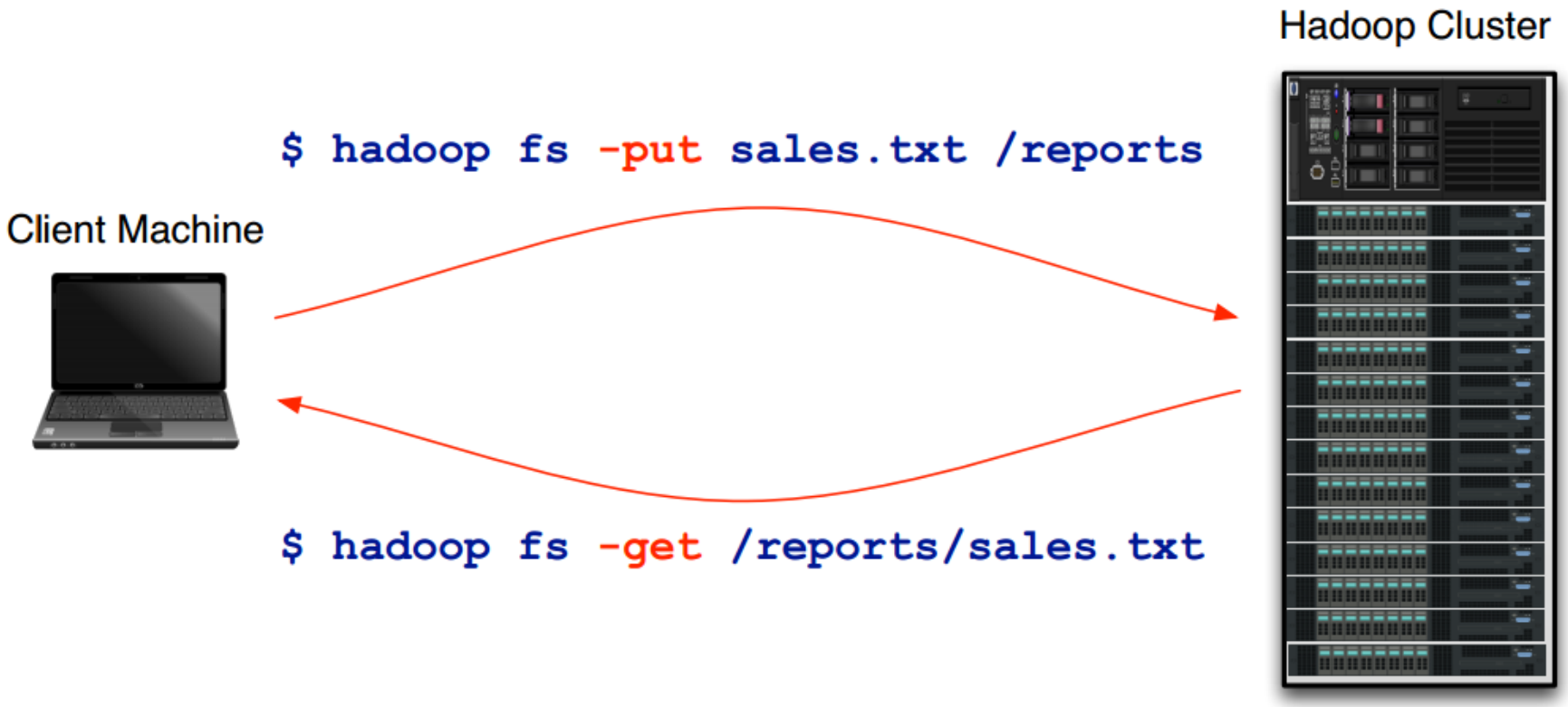
http://hadop.apache.org/docs/r2.4.1/hadoop-project-dist/hadoop-common/FileSystemShell.html

HDFS 쉘 명령어

Command	What It Does	Usage
chmod	Changes the permissions of files. With -R, makes the change recursively by way of the directory structure. The user must be the file owner or the superuser.	-chmod [-R] <MODE[,MODE]... | OCTAL MODE> URI [URI …]
chown	Changes the owner of files. With -R, makes the change recursively by way of the directory structure. The user must be the superuser.	-chown [-R] [OWNER][:[GROUP]] URI [URI]
copyFromLocal	Works similarly to the put command, except that the source is restricted to a local file reference.	-copyFromLocal <localsrc> URI
copyToLocal	Works similarly to the get command, except that the destination is restricted to a local file reference.	-copyToLocal [-ignorecrc] [-crc] URI <localdst>
cp	Copies one or more files from a specified source to a specified destination. If you specify multiple sources, the specified destination must be a directory.	-cp URI [URI …] <dest>
du	Displays the size of the specified file, or the sizes of files and directories that are contained in the specified directory. If you specify the -s option, displays an aggregate summary of file sizes rather than individual file sizes. If you specify the -hoption, formats the file sizes in a "human-readable" way.	-du [-s] [-h] URI [URI …]
get	Copies files to the local file system. Files that fail a cyclic redundancy check (CRC) can still be copied if you specify the -ignorecrc option. The CRC is a common technique for detecting data transmission errors. CRC checksum files have the .crc extension and are used to verify the data integrity of another file. These files are copied if you specify the -crcoption.	-get [-ignorecrc] [-crc] <src> <localdst>
ls	Returns statistics for the specified files or directories.	-ls <args>
mv	Moves one or more files from a specified source to a specified destination. If you specify multiple sources, the specified destination must be a directory. Moving files across file systems isn’t permitted.	-mv URI [URI …] <dest>
put	Copies files from the local file system to the destination file system. This command can also read input from stdin and write to the destination file system.	-put <localsrc> ... <dest>
rm	Deletes one or more specified files. This command doesn’t delete empty directories or files. To bypass the trash (if it’s enabled) and delete the specified files immediately, specify the -skip Trashoption.	-rm [-skipTrash] URI [URI …]
stat	Displays information about the specified path.	-stat URI [URI …]
tail	Displays the last kilobyte of a specified file to stdout. The syntax supports the Unix -f option, which enables the specified file to be monitored. As new lines are added to the file by another process, tailupdates the display.	-tail [-f] URI

HDFS 사용법

- File copy local data to and from HDFS
  - hadoop fs –put "local file path" "HDFS file path"
  - hadoop fs –get "HDFS file path" "local file path"



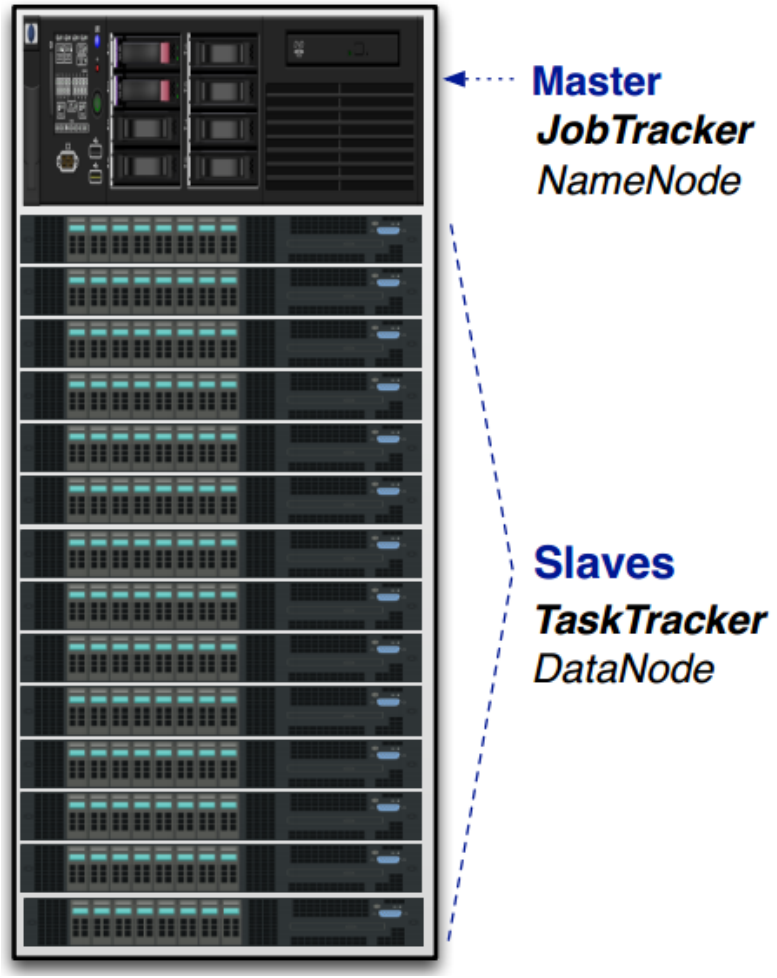
Hadoop MapReduce (맵리듀스)

- The implementation of MapReduce paper
  - Written by Java
- MapReduce is designed for
  - Simplified distributed data processing
  - Batch processing
- Not for
  - Interactive data processing, real-time data processing

MapReduce: 구조

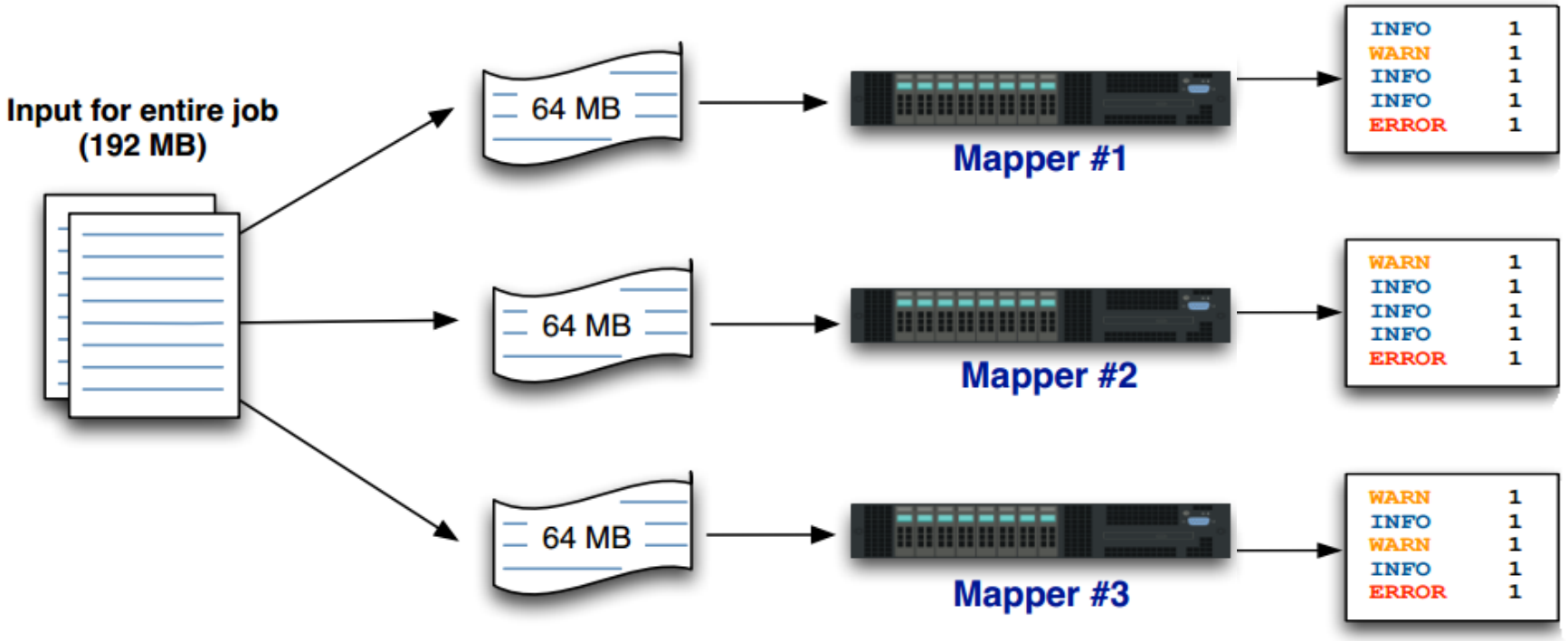
- Provides user functions
  - Map
  - Reduce

- Master-slave architecture
  - Single master - JobTracker
    - Schedules MR job
  - Multiple slaves - TaskTracker
    - Executes map, reduce functions



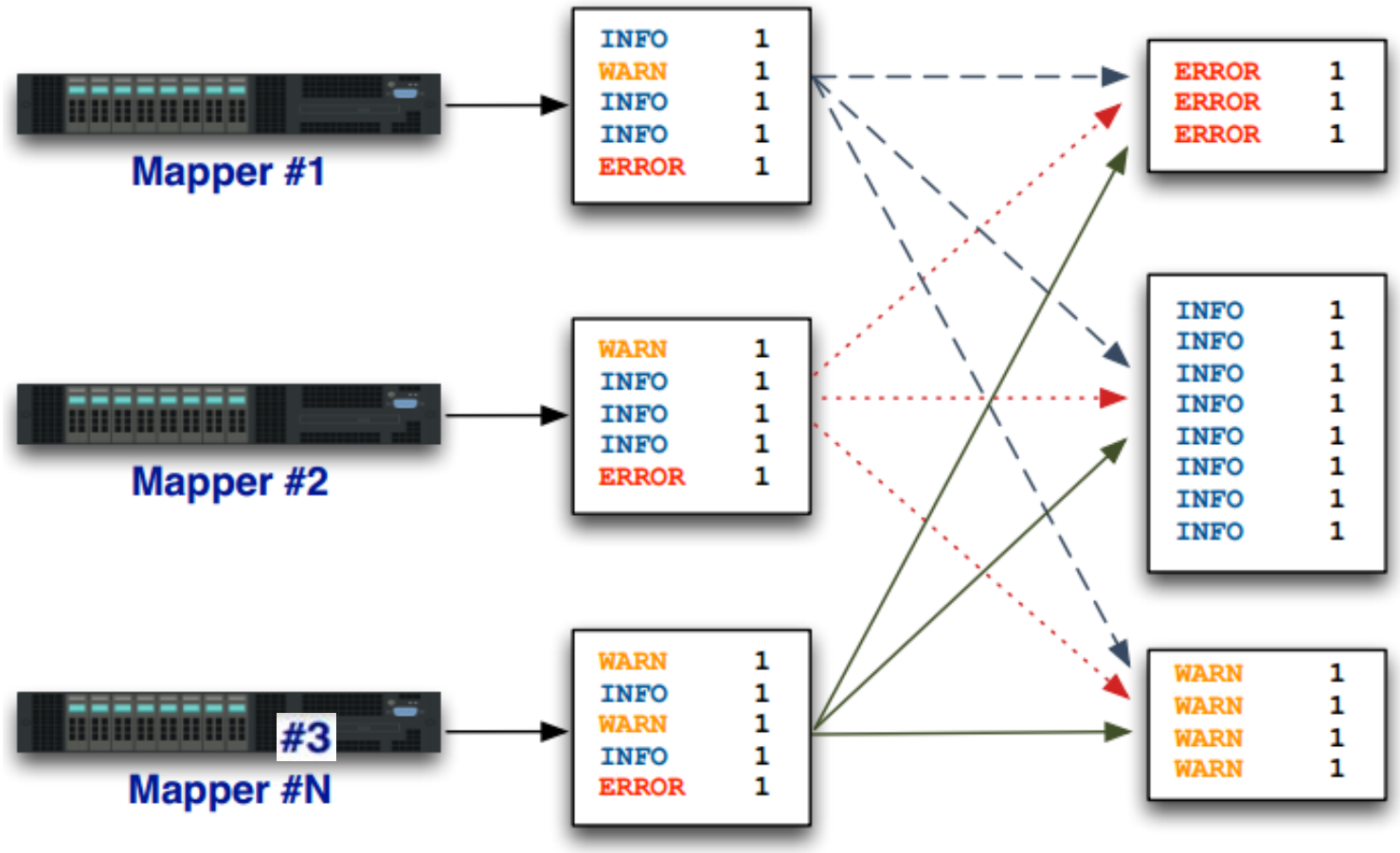
MapReduce: 함수/클래스들

- Map
  - Each mapper reads HDFS block as their input data
  - Produces <key,value> pairs as output
  - Mainly used to transformation, parsing, filtering



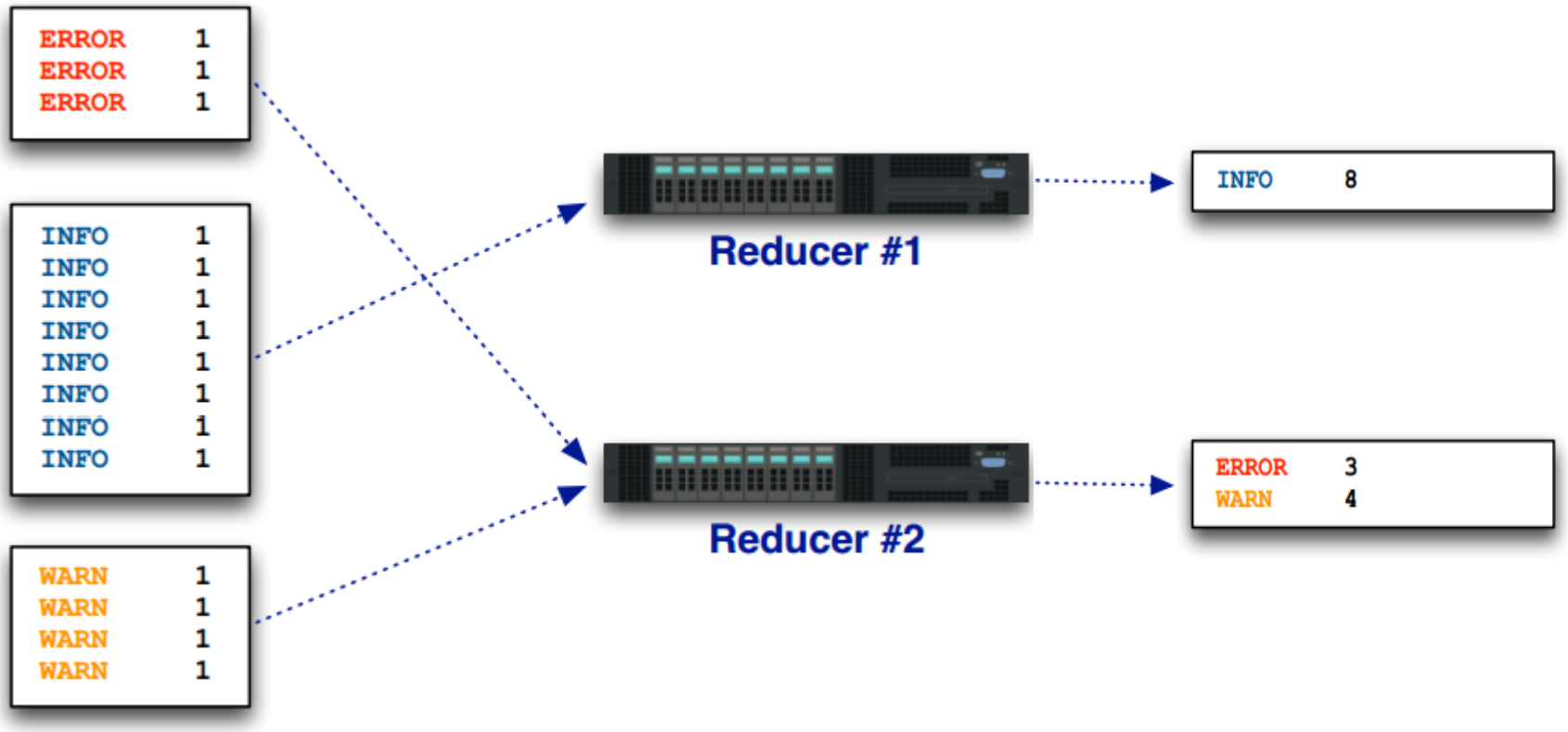
MapReduce: 함수/클래스들

- (Shuffle)
  - groups the mapper output values by the output key

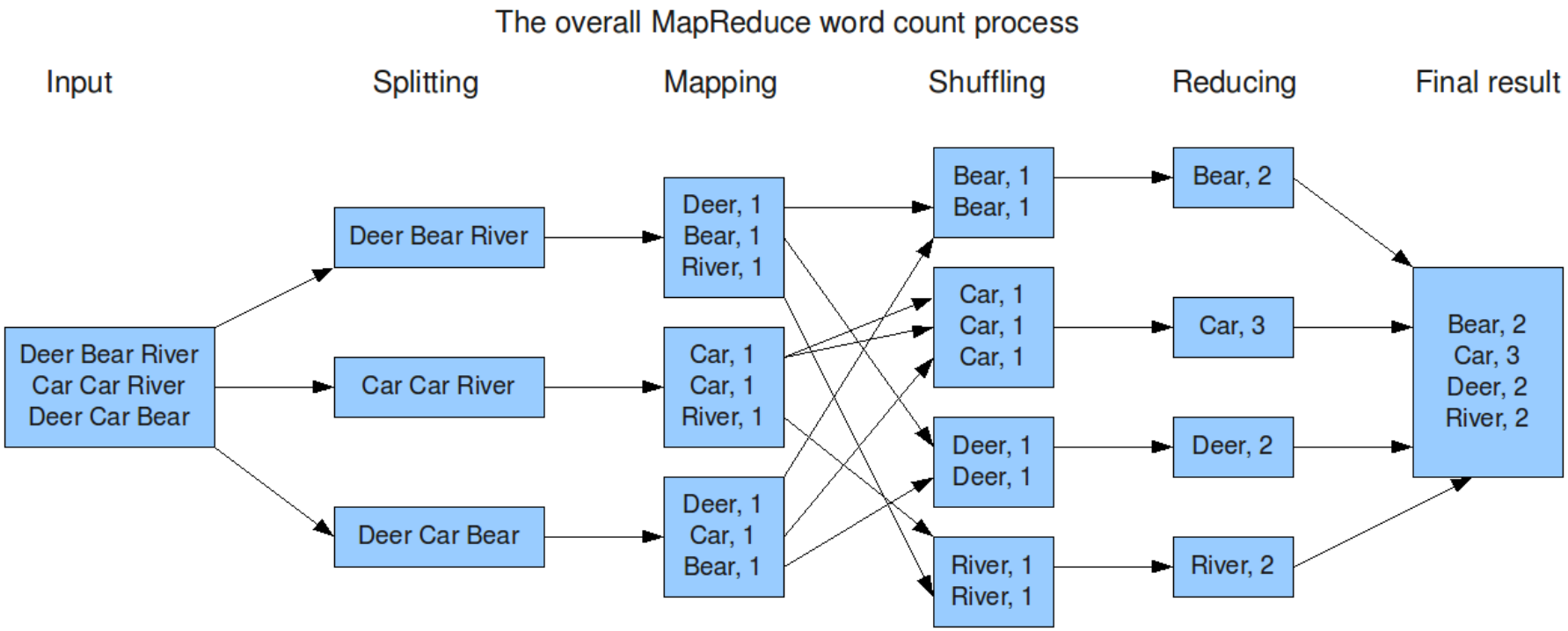


MapReduce: 함수/클래스들

- Reduce
  - Receives a key and all values for that key



MapReduce 응용 예: WordCount



MapReduce 응용 예: WordCount

- WordCount Map function

```
public static class TokenizerMapper
     extends Mapper<Object, Text, Text, IntWritable>{

  private final static IntWritable one = new IntWritable(1);
  private Text word = new Text();

  public void map(Object key, Text value, Context context
                  ) throws IOException, InterruptedException {
    StringTokenizer itr = new StringTokenizer(value.toString());
    while (itr.hasMoreTokens()) {
      word.set(itr.nextToken());
      context.write(word, one);
    }
  }
}
```

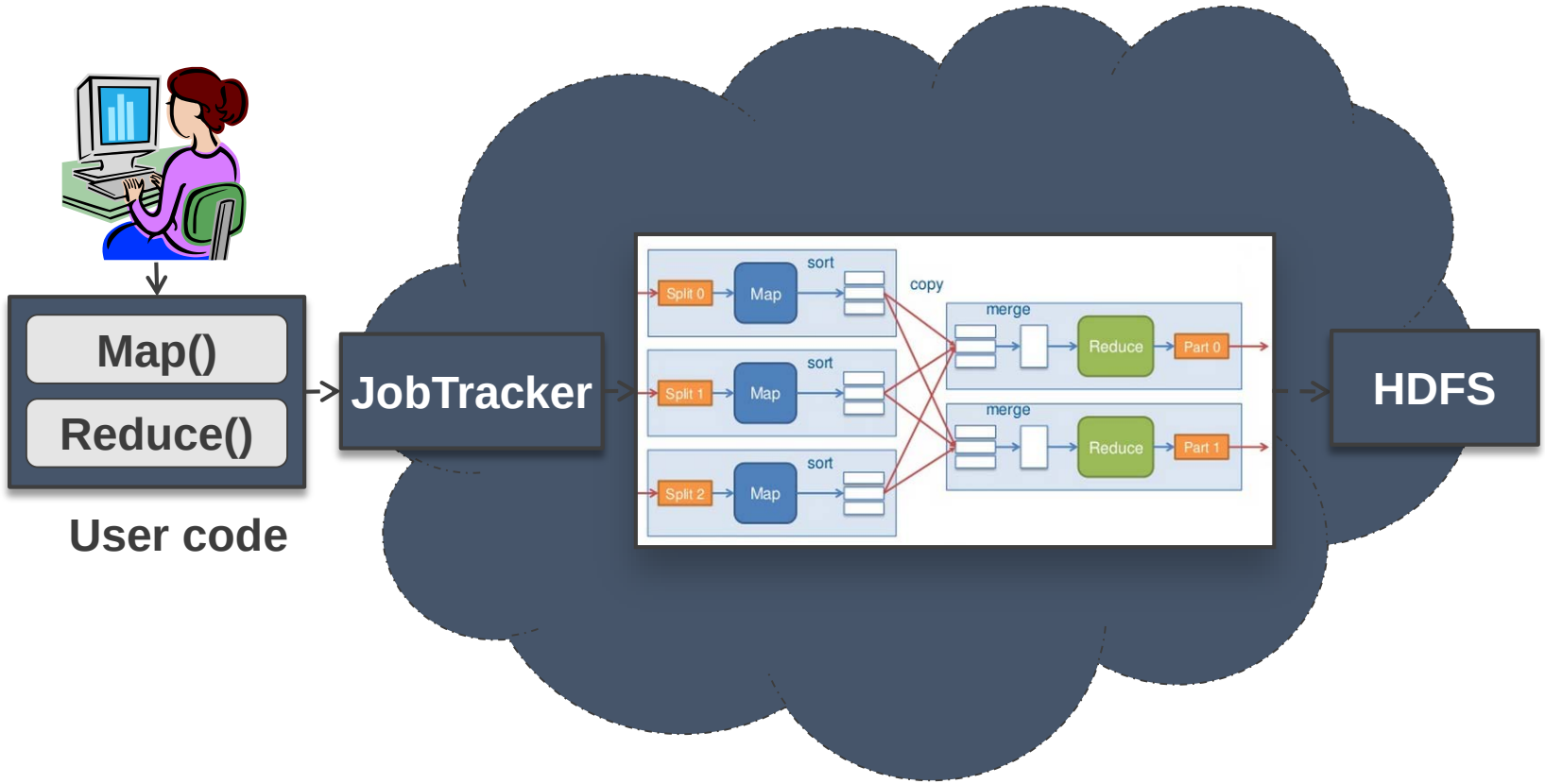
MapReduce 응용 예: WordCount

- WordCount reduce function

```
public static class IntSumReducer
     extends Reducer<Text,IntWritable,Text,IntWritable> {
  private IntWritable result = new IntWritable();

  public void reduce(Text key, Iterable<IntWritable> values,
                     Context context
                     ) throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values) {
      sum += val.get();
    }
    result.set(sum);
    context.write(key, result);
  }
}
```

MapReduce 특징

- Simple and easy
  - Users only make map and reduce function
    - The MR framework automatically controls the distributed task execution, data flow, fault handling and etc.



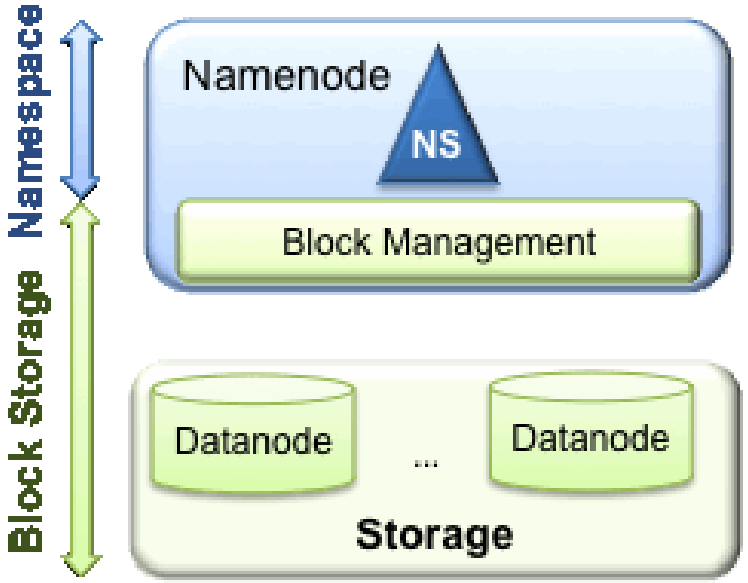
MapReduce 응용들

- Text search, indexing
- Web link-graph generation
- Machine learning
- Analytics, statistics
- Document clustering
- Data filtering, transformation
- Web access log processing
- …

Hadoop v2.0

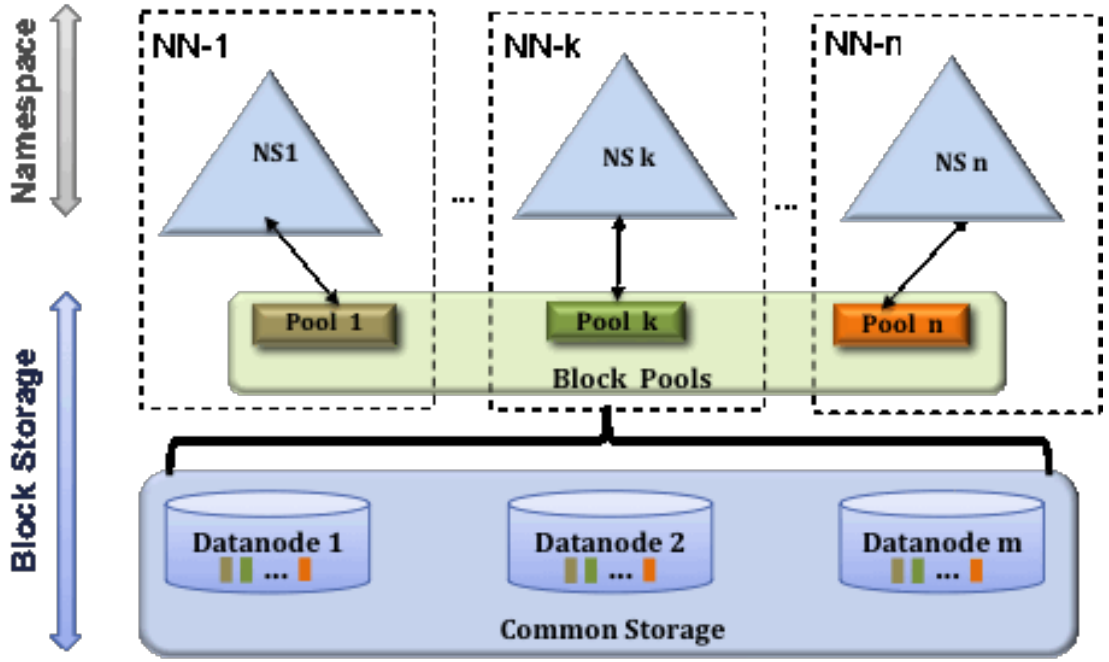
HDFS v1의 문제점들

- 단일 NameNode 사용 !
  - Single Namespace management
  - Single block management
- 문제점들
  - 확장성 (Scalability) 제약
    - Limited number of files, blocks
  - 성능 (Performance) 제약
    - Limited HDFS operation throughput
  - 독립성 제약 (Poor isolation)
    - All the users share a single namespace



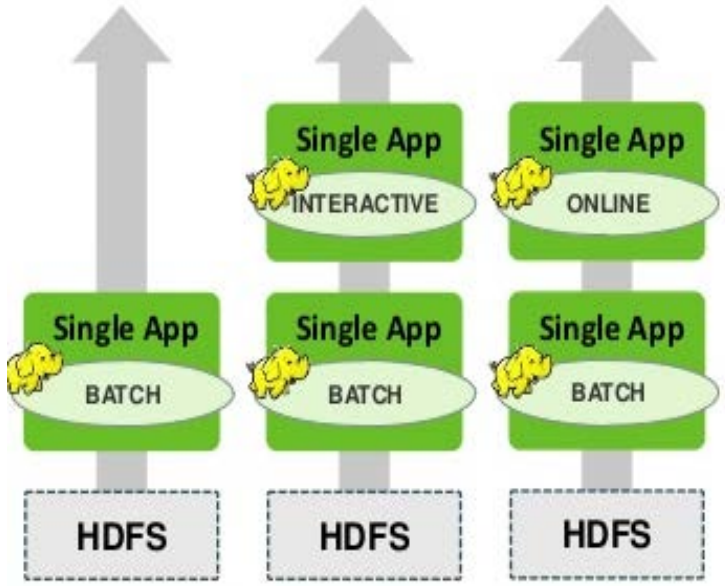
HDFS v2

- 다중 NameNode 사용
  - Multiple independent namespaces
  - Multiple independent block managements



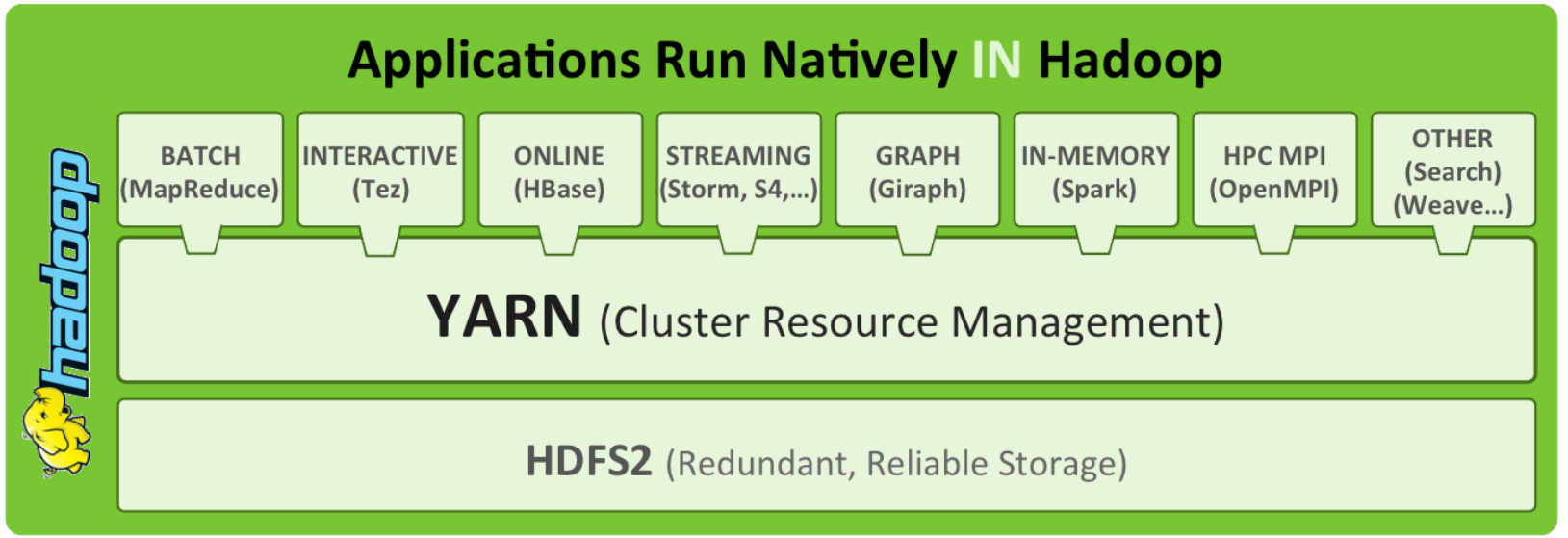
MapReduce v1의 문제점

- 문제점들
  - MR이외의 다른 패러다임 사용불가
    - MR위에서 사용 ?
  - 확장성 제약
    - JobTracker가 하나
  - 자원 관리가 비효율적
    - Map/Reduce 태스크로 일관 관리



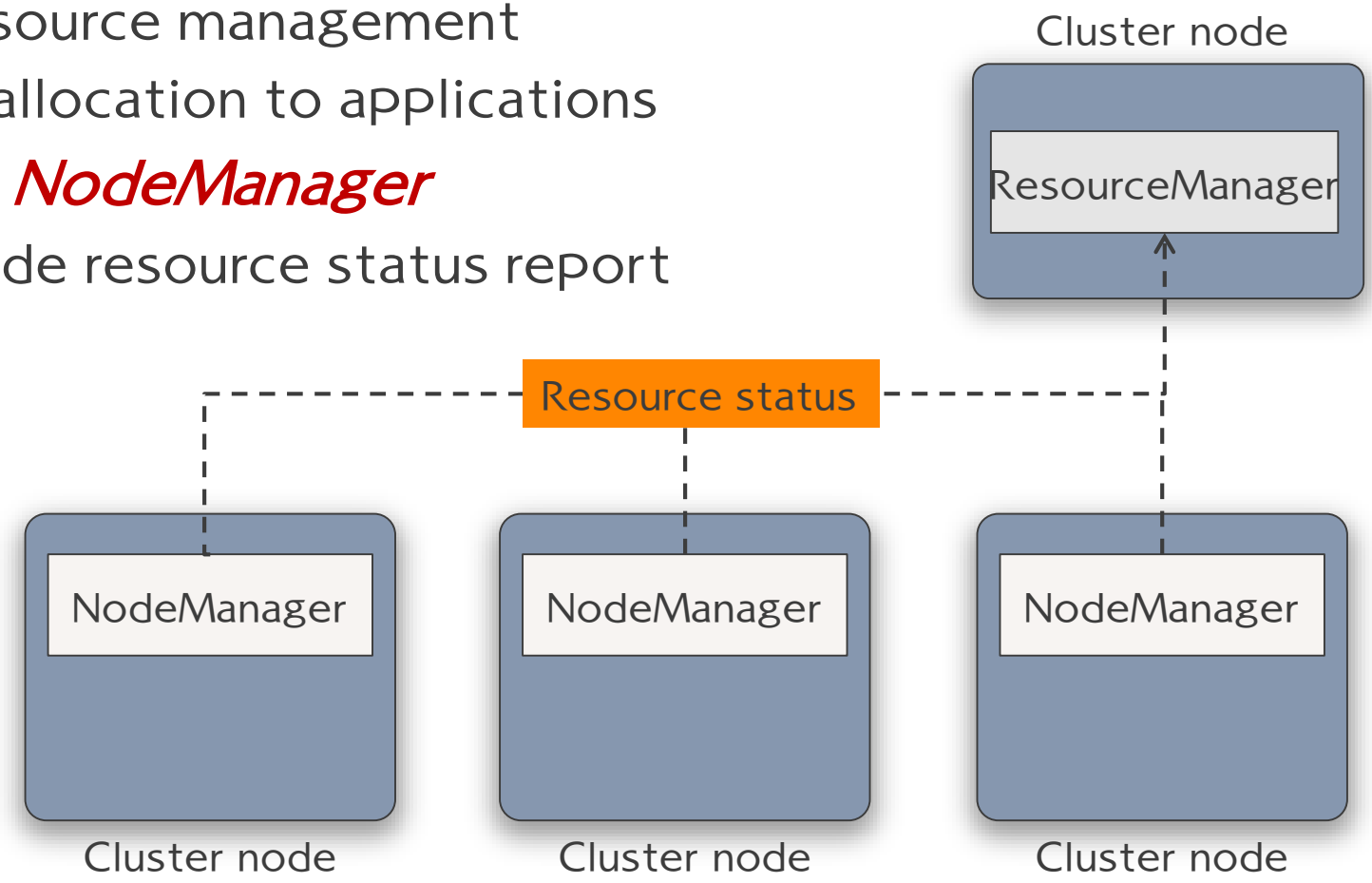
YARN(Yet Another Resource Negotiator)

- Cluster resource management
  - New process: *ResourceManager* and *NodeManager*
- New Hadoop appilcations including MapReduce
  - User defined Hadoop applications like MapReduce
  - Scheduled by *ApplicationMaster*



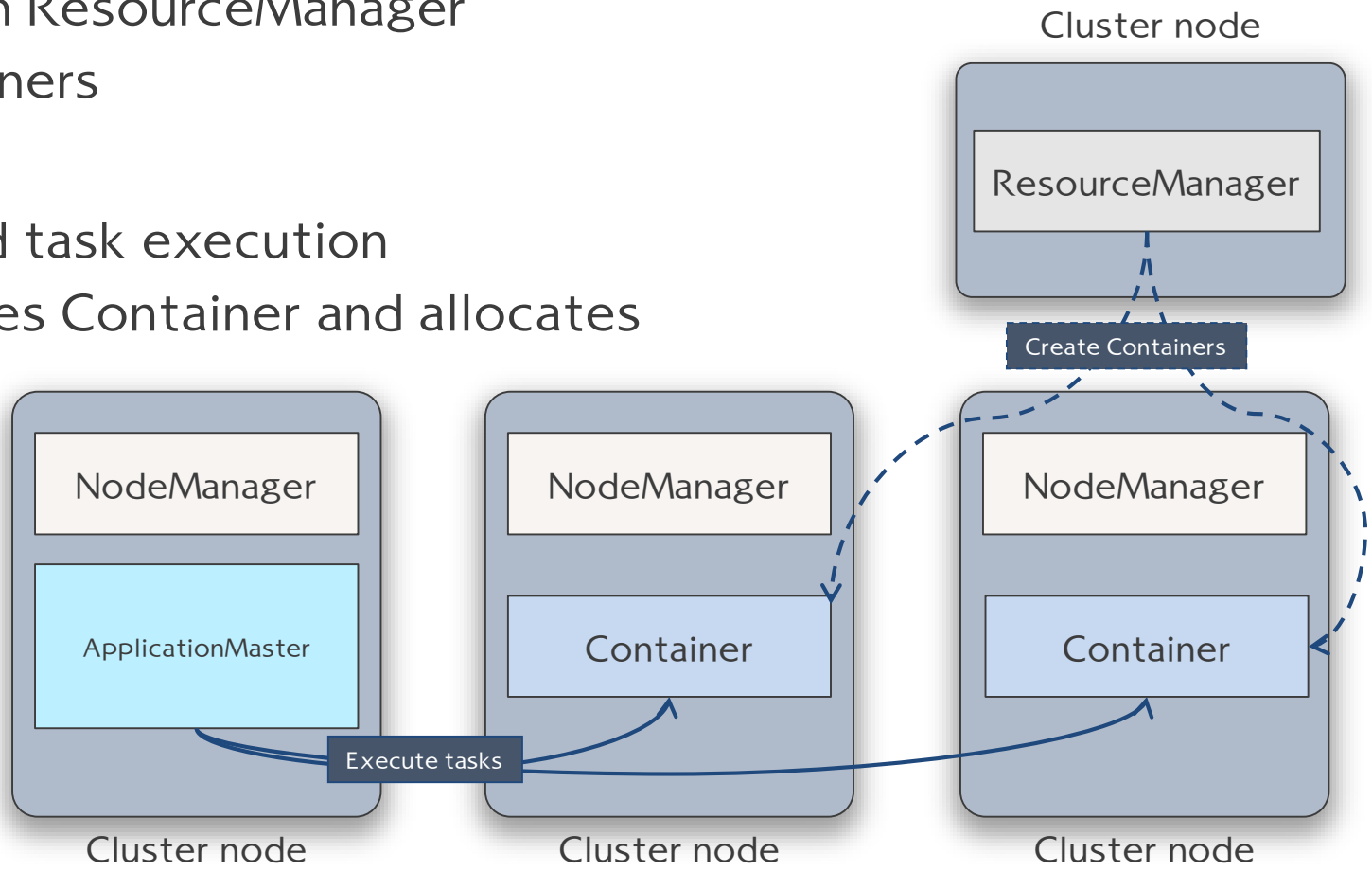
YARN: 구조

- Master/Slave
  - One master: ***ResourceManager***
    - Cluster resource management
    - Resource allocation to applications
  - Many slaves: ***NodeManager***
    - Cluster node resource status report

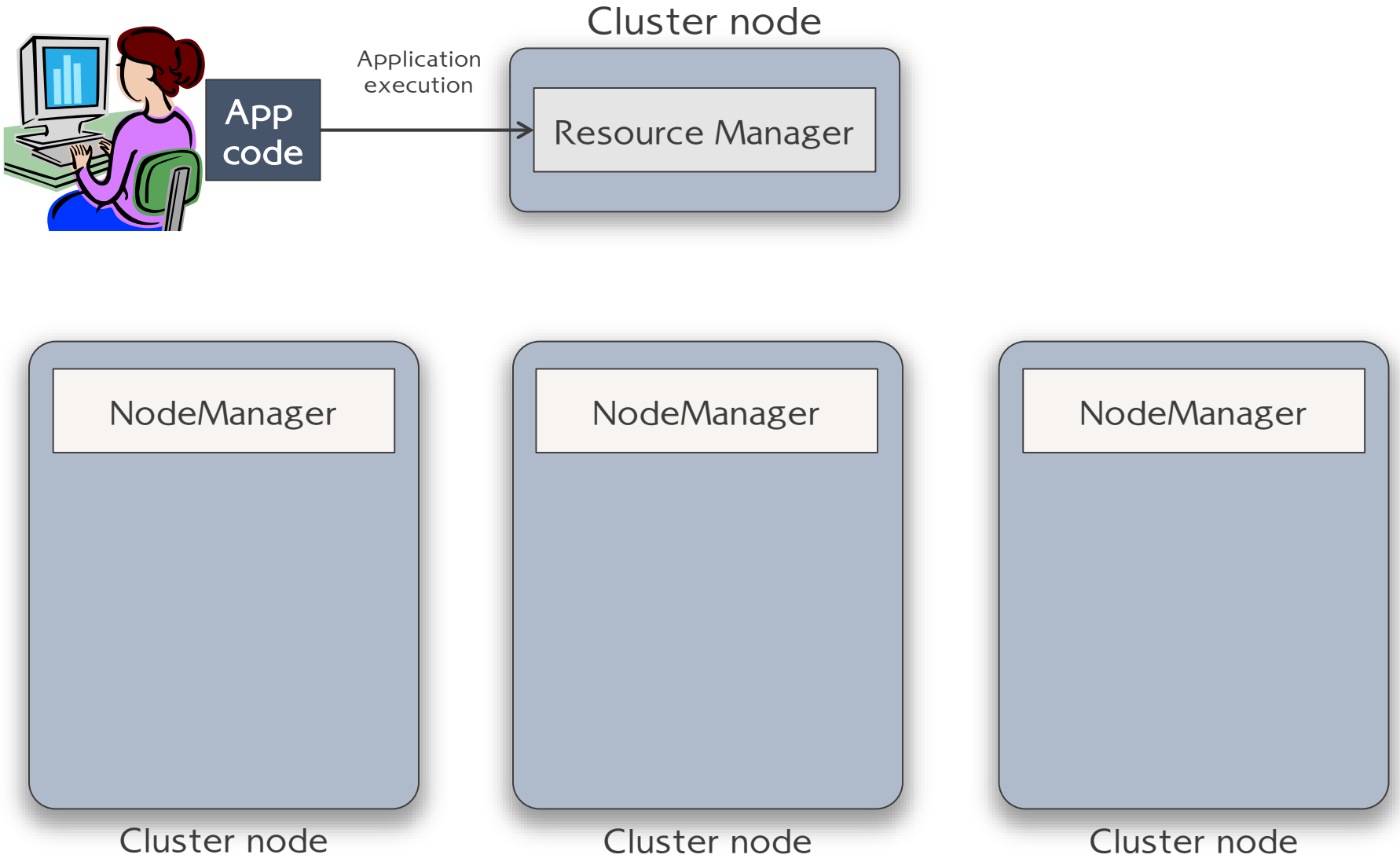


YARN: 구조

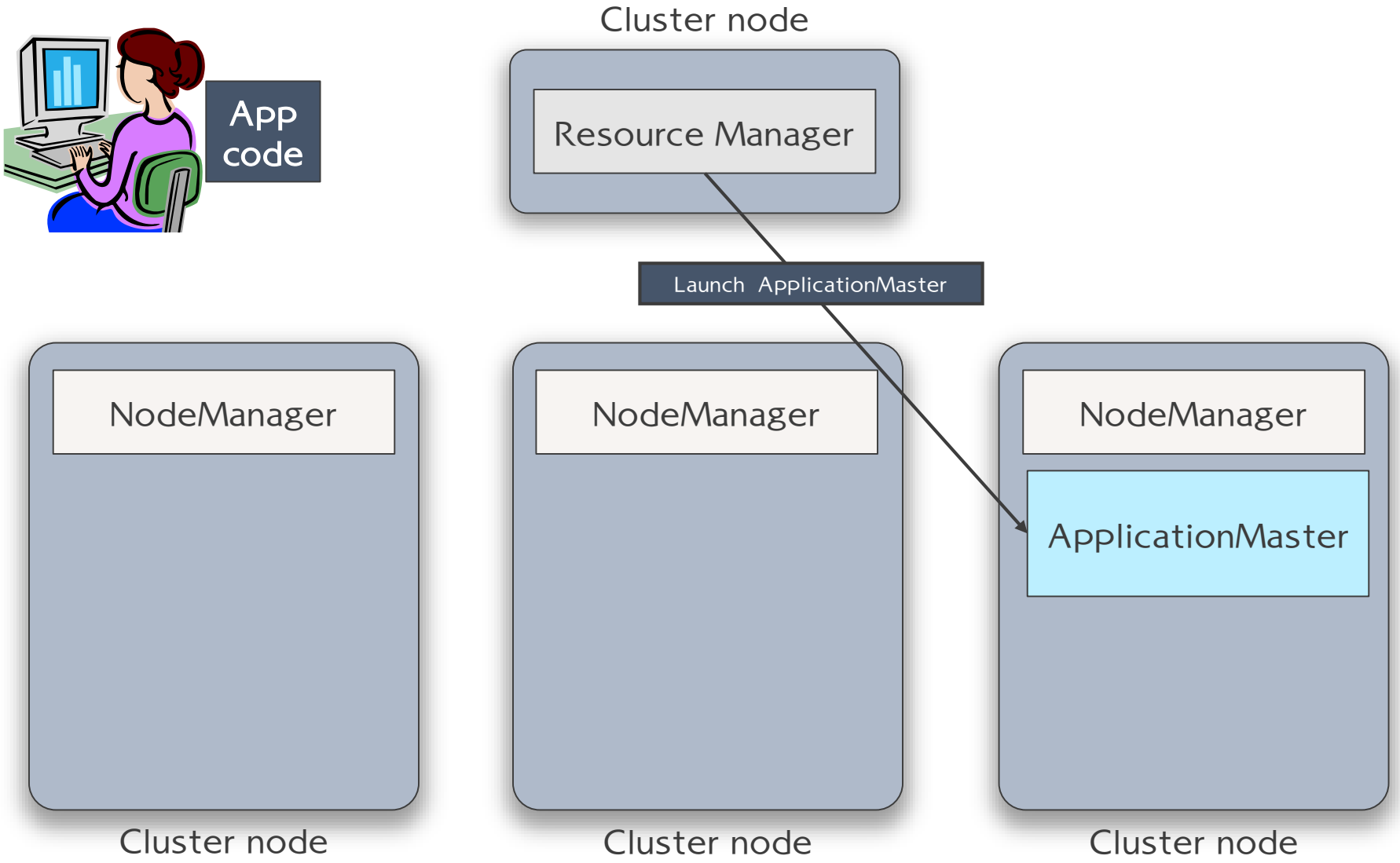
- ApplicationMaster
  - Schedules Hadoop application tasks
  - Obtains Containers from ResourceManager
  - Execute tasks on containers
- Container
  - The unit of resource and task execution
  - ResourceManager creates Container and allocates to ApplicationMaster



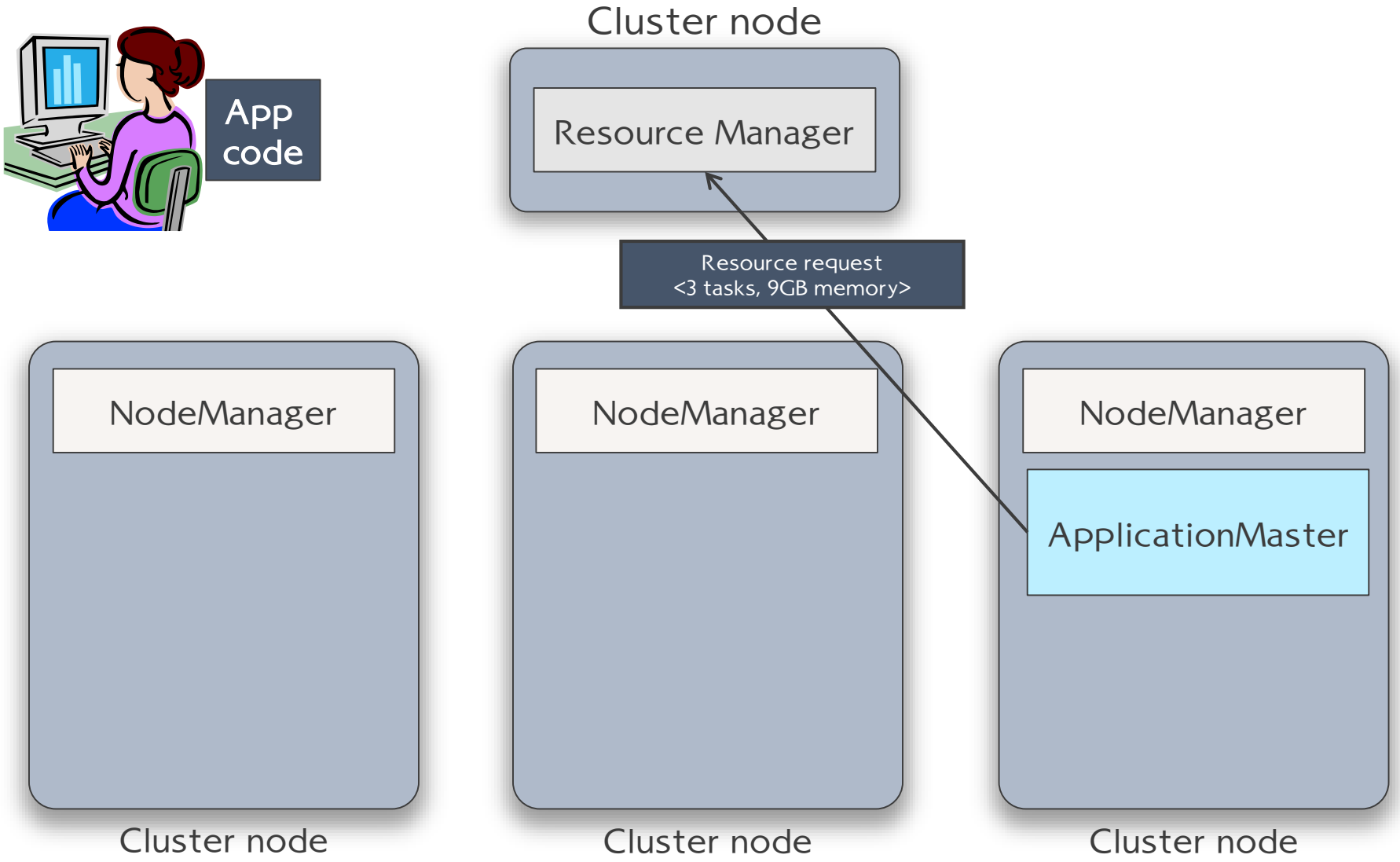
YARN 사용 예제


Cluster node
Application execution
App code
Resource Manager
NodeManager
NodeManager
NodeManager
Cluster node
Cluster node
Cluster node

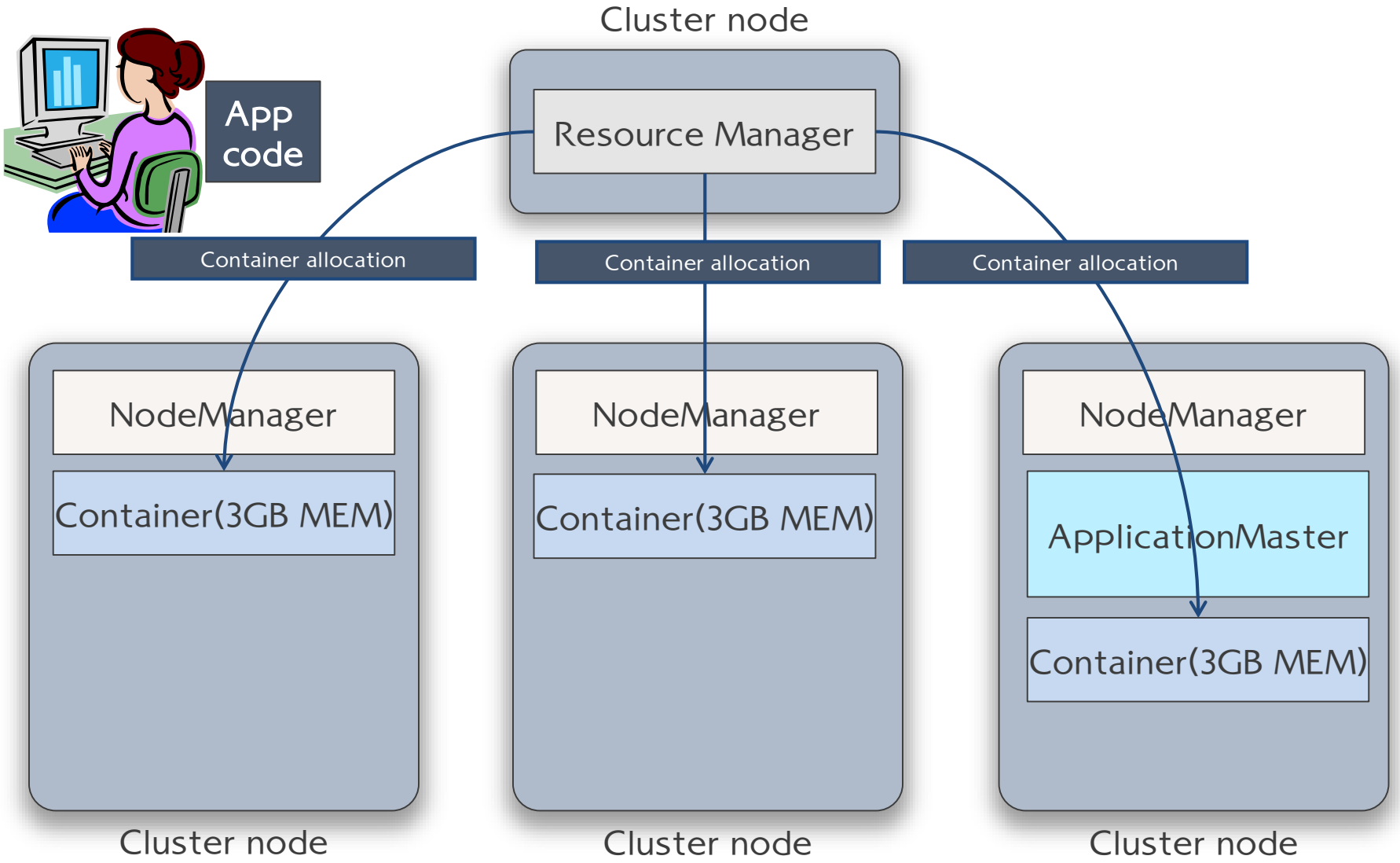
YARN 사용 예제


App code
Cluster node
Resource Manager
Launch ApplicationMaster
NodeManager
NodeManager
NodeManager
ApplicationMaster
Cluster node
Cluster node
Cluster node

YARN 사용 예제


App code
Cluster node
Resource Manager
Resource request
<3 tasks, 9GB memory>
NodeManager
NodeManager
NodeManager
ApplicationMaster
Cluster node
Cluster node
Cluster node

YARN 사용 예제


Cluster node
App code
Resource Manager
Container allocation
Container allocation
Container allocation
NodeManager
Container(3GB MEM)
NodeManager
Container(3GB MEM)
NodeManager
ApplicationMaster
Container(3GB MEM)
Cluster node
Cluster node
Cluster node

YARN 사용 예제

Cluster node

Resource Manager

Container informations

NodeManager

Container(3GB MEM)

Cluster node

NodeManager

Container(3GB MEM)

Cluster node

NodeManager

ApplicationMaster

Container(3GB MEM)

Cluster node

YARN 사용 예제

Cluster node

Resource Manager

NodeManager

Container(3GB MEM)

Cluster node

NodeManager

Container(3GB MEM)

Task allocation

Cluster node

NodeManager

ApplicationMaster

Container(3GB MEM)

Cluster node

YARN 사용 예제

Cluster node

Resource Manager

NodeManager

Task execution

Cluster node

NodeManager

Task execution

Cluster node

NodeManager

ApplicationMaster

Task execution

Cluster node

YARN 사용 예제

Cluster node

Resource Manager

NodeManager

Completed task

NodeManager

Completed task

complete report

NodeManager

ApplicationMaster

Completed task

Cluster node

Cluster node

Cluster node

YARN 사용 예제

Cluster node

Resource Manager

complete report

NodeManager

NodeManager

NodeManager

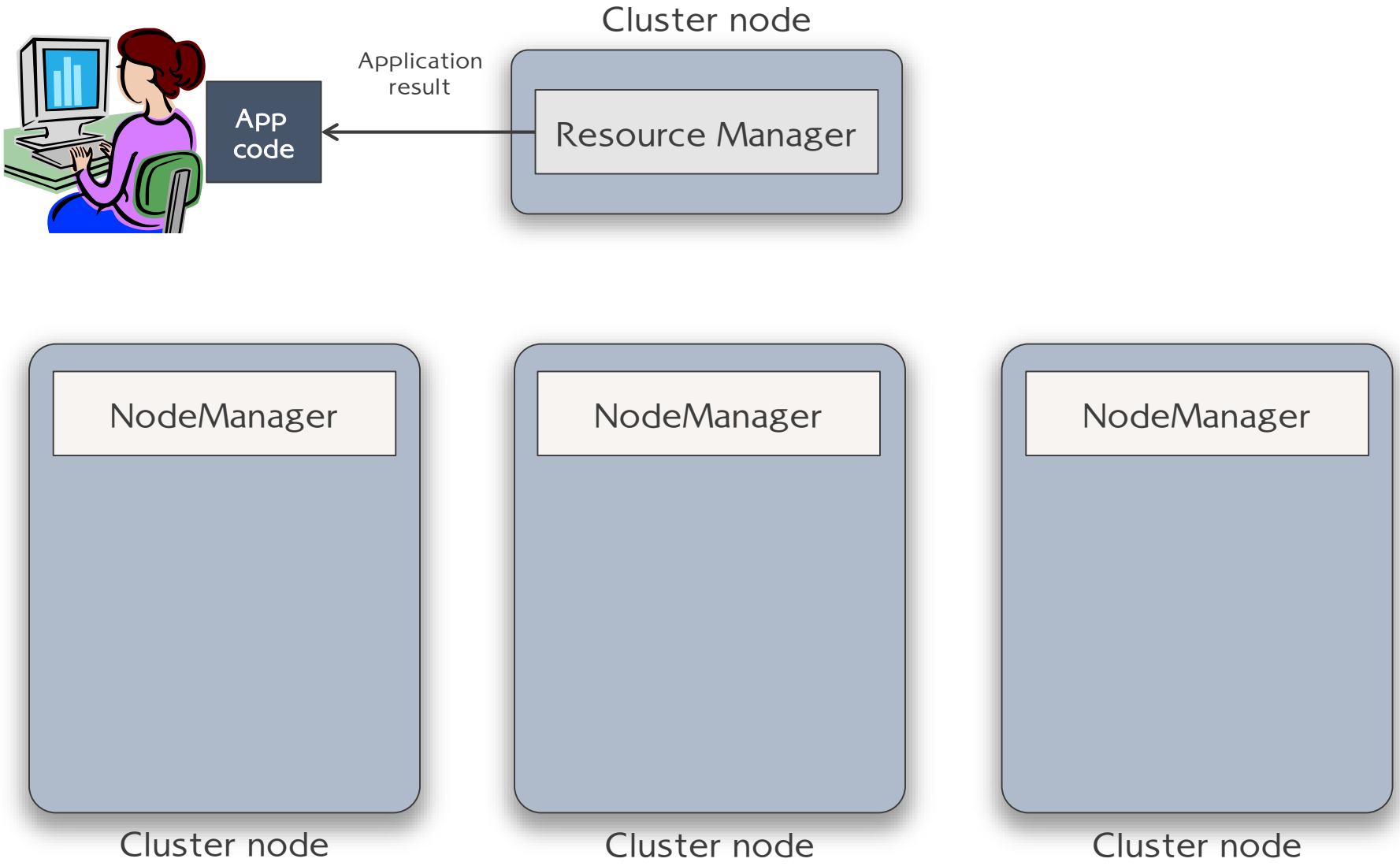
ApplicationMaster

Cluster node

Cluster node

Cluster node

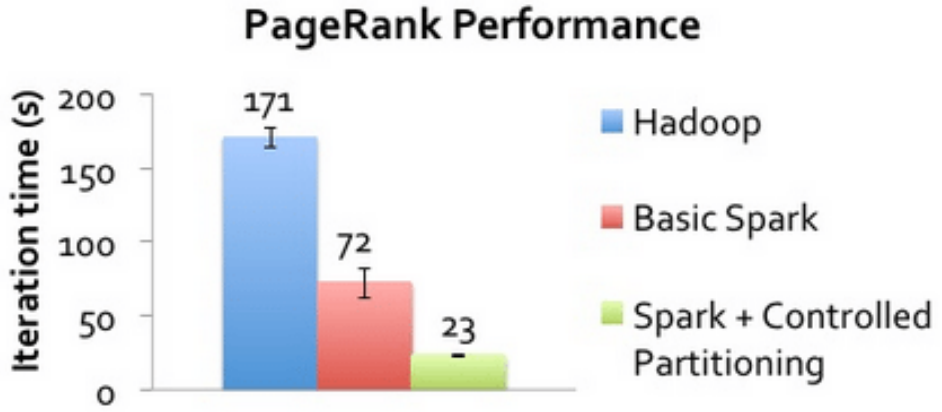
YARN 사용 예제


Cluster node
Application result
App code
Resource Manager
NodeManager
NodeManager
NodeManager
Cluster node
Cluster node
Cluster node

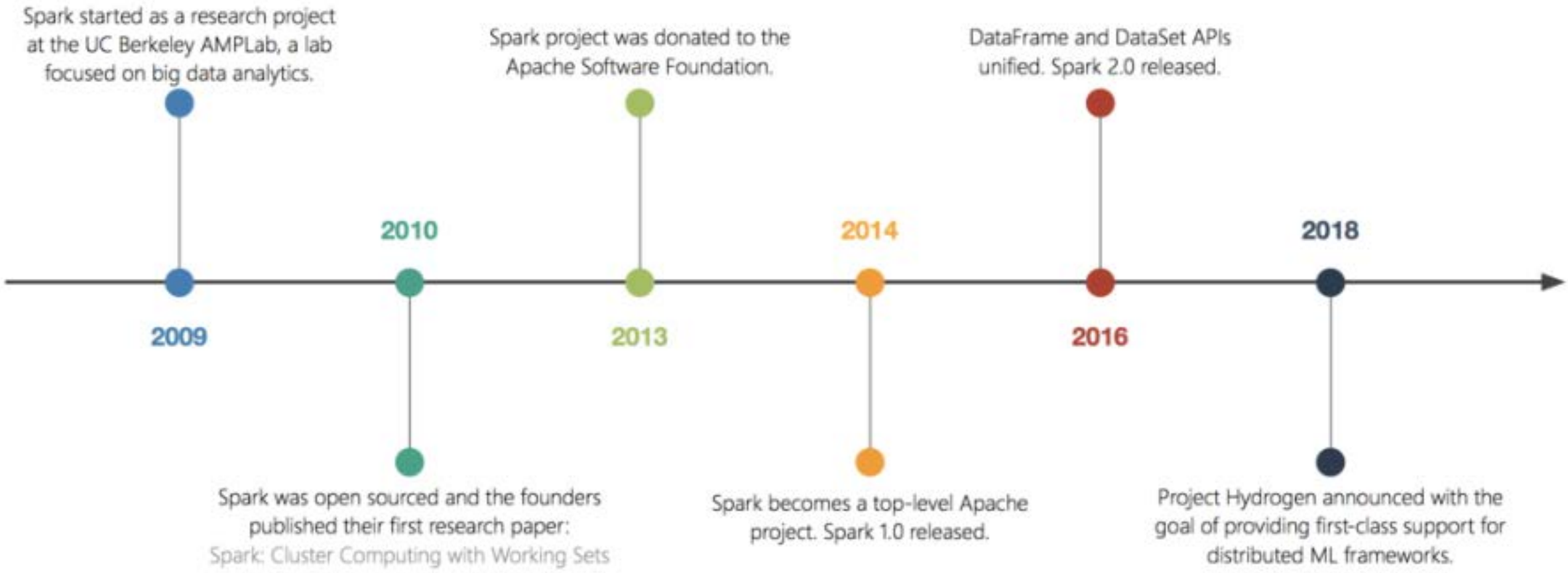
Apache Spark

Spark란?

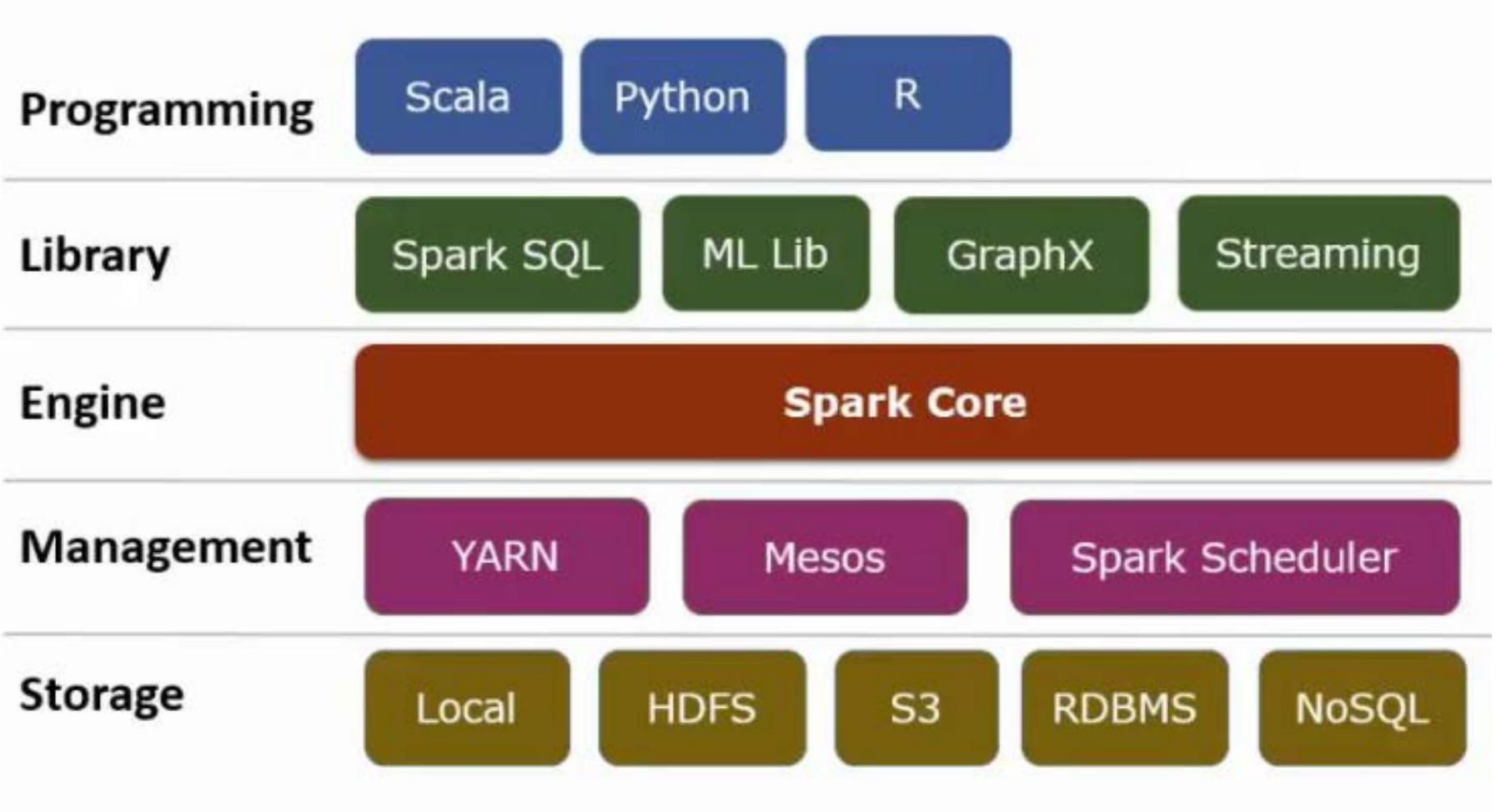
- 오픈소스 (클러스터 기반) 분산 데이터 처리 엔진
- 하둡 맵리듀스 대비 빠른 처리 속도 제공
  - 하둡은 매번 중간 결과를 디스크에 저장하는 데에 비해 스파크는 중간 결과를 메모리에서 처리
- Map->Reduce로 고정된 Dataflow만 가능한 하둡 맵리듀스에 비해 유연한 처리 기능 제공
  - PageRank나 머신러닝 알고리즘과 같이 반복계산이 많은 경우 특히 좋은 성능을 보임



History of Spark



스파크 구조


Programming
Scala
Python
R
Library
Spark SQL
ML Lib
GraphX
Streaming
Engine
Spark Core
Management
YARN
Mesos
Spark Scheduler
Storage
Local
HDFS
S3
RDBMS
NoSQL

Spark 특징

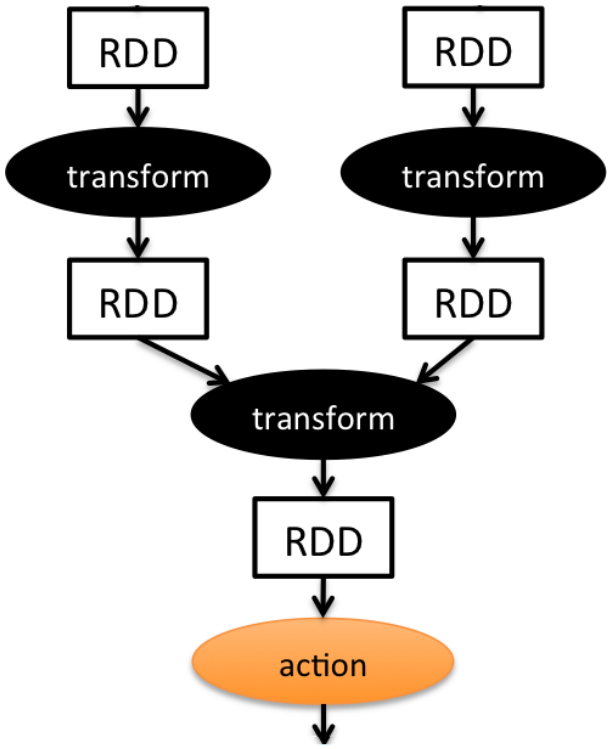
- 메모리에 데이터 캐싱 (caching)을 통해 반복적 작업의 인-메모리 처리가 가능
- 클러스터를 활용한 분산 처리를 크게 신경 쓰지 않아도 되는 High-level API 제공
  - Scala, Java ,Python, R 지원
- 클러스터 관리를 위한 다양한 자원 관리 프레임워크 활용 가능
  - YARN, Apache Mesos 등
- 다양한 저장소 활용 지원
  - HDFS, Amazon S3 등
  - 즉, 기존 구축된 클러스터 환경에서 쉽게 활용 가능

RDD (Spark v1.0)

- Resilient Distributed Dataset
  - Spark의 추상화된 데이터 처리 단위
    - Resilient
      - 오류 복구가 가능한
    - Distributed
      - 분산 저장/처리를 위한
    - Immutable
      - 변경 불가한, 읽기 전용의
  - 물리적으로 RDD는 여러 개의 Partition으로 구성
  - 메모리상에 Cache 가능

RDD: 지원 연산

- Transformations (deriving a new RDD)
  - RDD에 변환/처리 함수를 적용
    - map, filter, flatMap, sample, groupByKey, reduceByKey, sort …
- Actions (getting computed values from RDD)
  - 처리 결과를 저장하거나 표시
    - count, collect, reduce, lookup, save



RDD: 예제

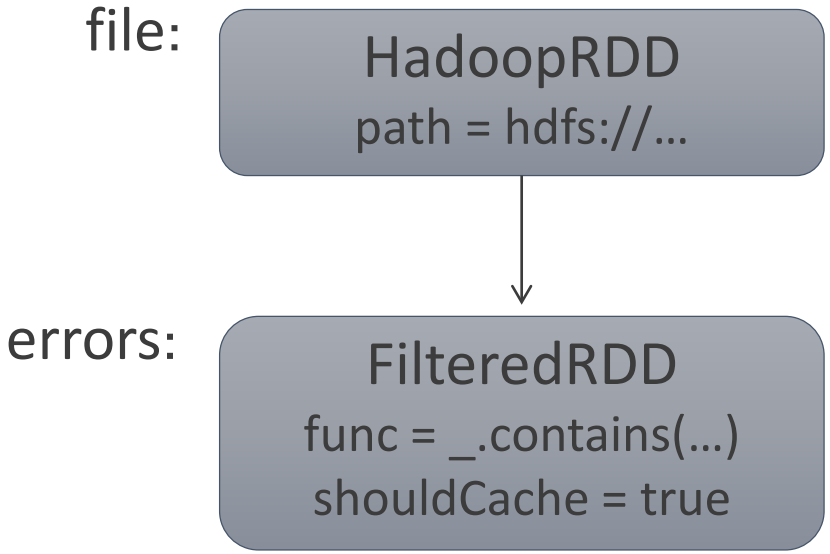
```
val sc = new SparkContext(
  “spark://...”, “MyJob”, home, jars)

val file = sc.textFile(“hdfs://...”)

val errors = file.filter(_.contains(“ERROR”))

errors.cache()

errors.count()
```

Resilient distributed datasets (RDDs), **Transformation**

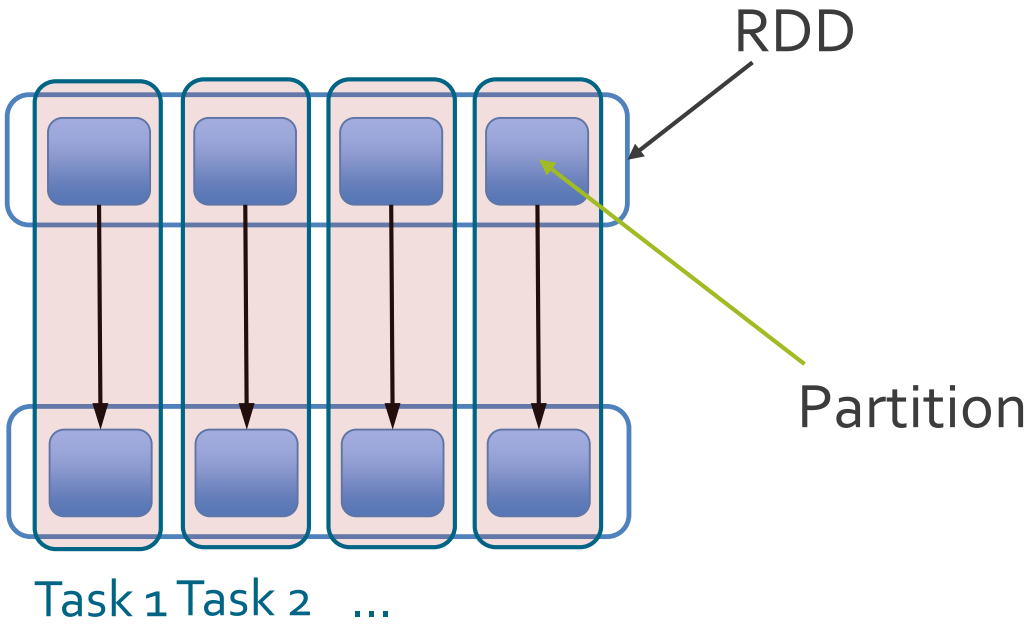
Action

RDD: 예제

Dataset-level view:

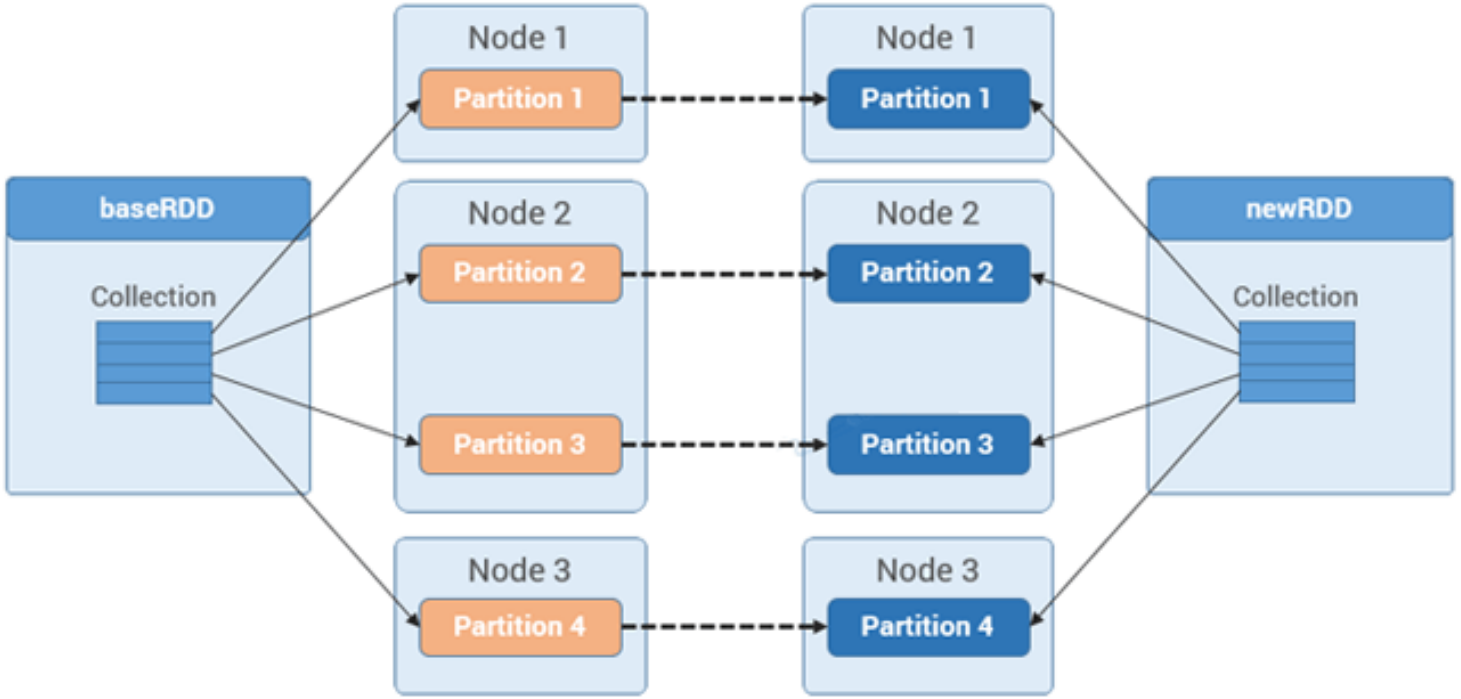


Partition-level view:



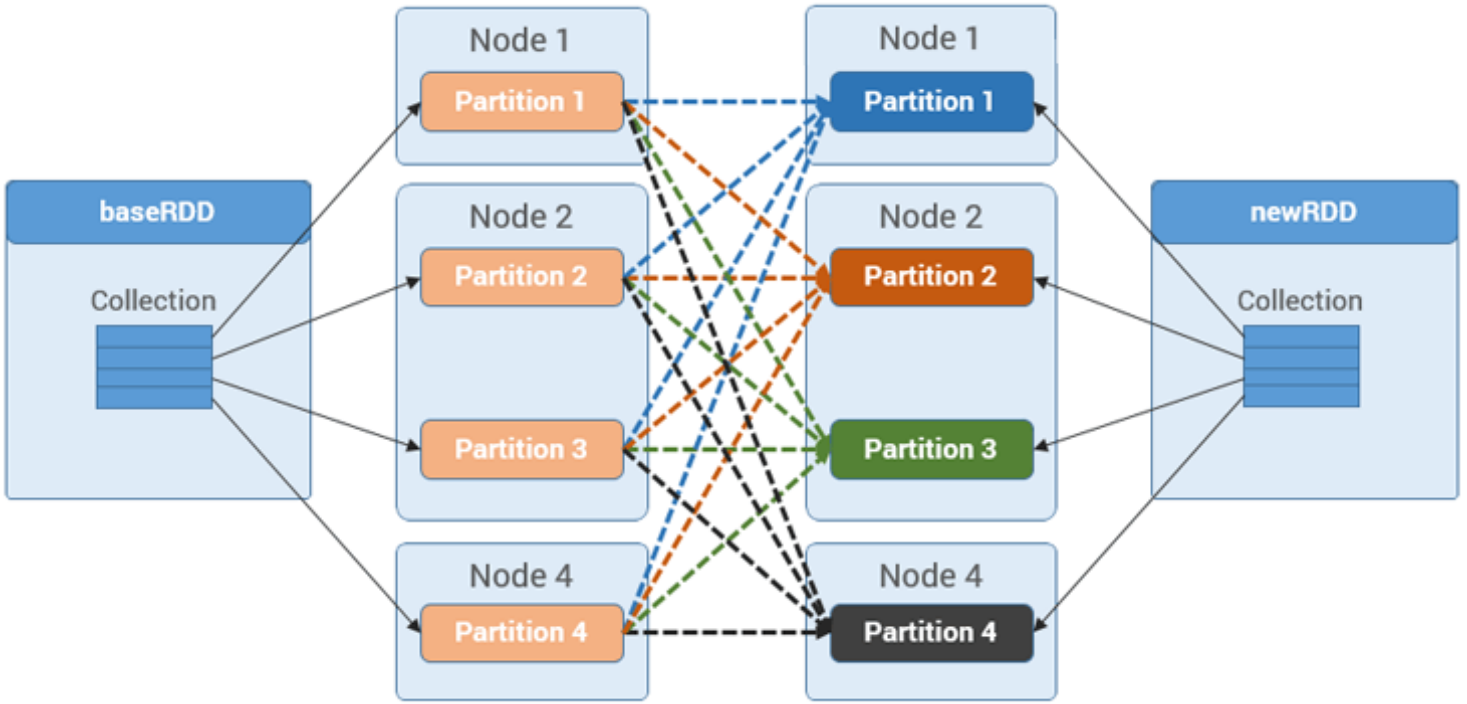
연산자의 Dependency

- Narrow operations
  - map, union, filter 등
  - No data transfer between cluster nodes



연산자의 Dependency

- Wide operations
  - groupByKey, distinct, join 등
  - Cause shuffle
  - Operator is split into stages of tasks and tasks are manipulated by the



RDD: 오류 복구

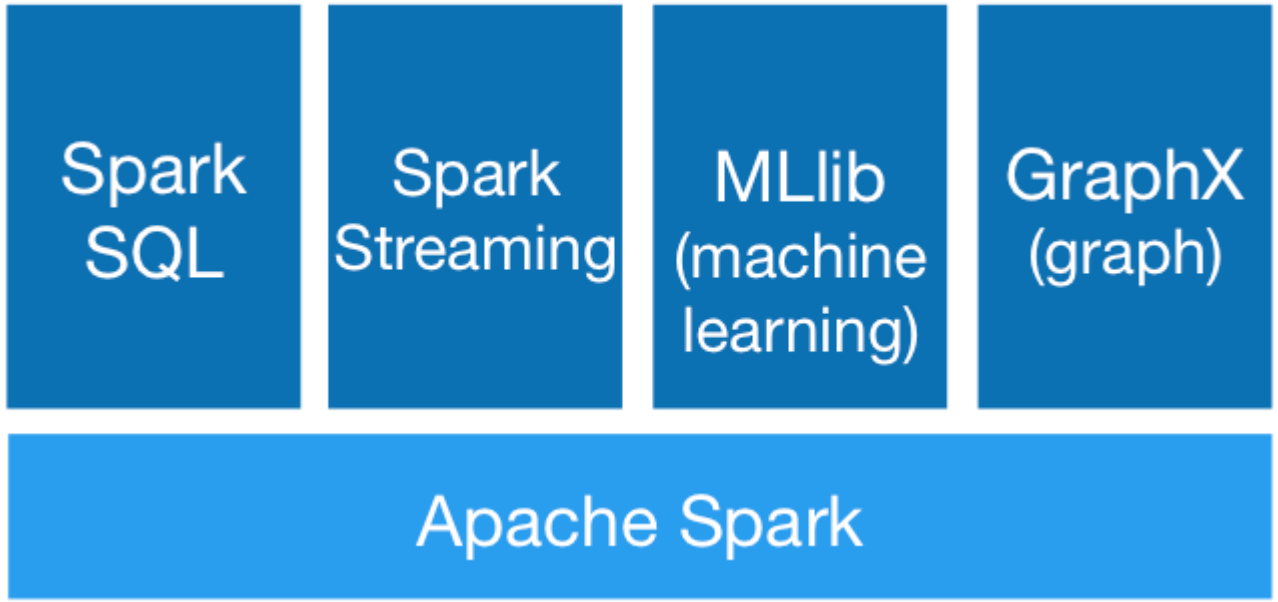
- Lineage
  - Transformation 선언 시 데이터가 바로 처리 되는 것이 아니라 어떤 연산이 선언 되었는지에 대한 정보를 기록
  - Action이 선언 될 때 비로소 기록한 Lineage를 통해 새로운 RDD를 도출
    - Lazy evaluation

- 클러스터의 고장이나 연산의 오류 등의 이유로 작업이 실패해도 Lineage를 통해 데이터를 복구 가능

RDD의 진화

- DataFrame (Spark v1.3)
  - 데이터를 테이블 형태로 추상화하여 처리
  - 다양한 최적화 기법을 통해 성능 향상 도모
    - SQL optimization 적용
    - 메모리와 CPU의 병목 완화
      - Serialization into off-heap memory (Structured binary)
- DataSet (Spark v1.6)
  - DataFrame 의 확장
  - OOP 스타일의 프로그래밍과 compile time type checking 지원
  - RDD와 DataFrame의 장점 모두 지원

Spark Library



- Spark SQL: SQL을 활용하여 데이터 분석
- Spark Streaming: 스파크를 활용하여 스트림 데이터 처리
- MLlib: 머신러닝 라이브러리
- GraphX: 그래프 분석

Spark Use Cases

- Spark runs on a Hadoop cluster in the range of 2000 nodes managed with YARN

- Spark is used for personalizing its news web page and for targeted advertising
- 15,000 lines of C++ code (before Spark) -> 120 lines of Scala code (after Spark)

- Uses Spark for real-time stream processing to provide online recommendations
- Over 450 billion events are processed per day

Q & A

https://database.korea.ac.kr

KOREA UNIVERSITY
DATABASE LAB