

CSE545

빅데이터처리 및 실습

(Big Data Processing and Practice)

Theory 04:

**Hadoop Distributed File Systems
(HDFS)**

담당교수: 전강욱(컴퓨터공학부)

kw.chon@koreatech.ac.kr

빅 데이터

■ 빅 데이터

- 기존의 데이터 처리 응용 SW로 수집, 저장, 분석, 처리하기 어려운 대규모 데이터
- 또는, 대규모 데이터로부터 가치를 추출하고 결과를 분석하는 기술

■ 플랫폼

- 많은 사람들이 쉽게 이용할 수 있고, 다양한 목적의 비즈니스가 이루어지는 공간



기차 플랫폼(출처: <https://www.collinsdictionary.com/ko/dictionary/english/platform>).

빅 데이터 플랫폼

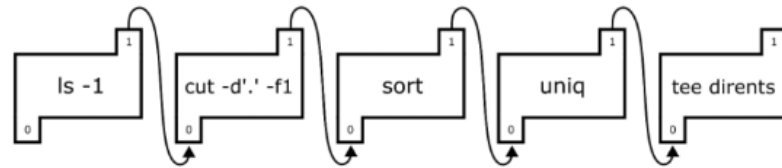
- 플랫폼: 많은 사람들이 쉽게 이용할 수 있고, 다양한 목적의 비즈니스가 이루어지는 공간
 - HW 플랫폼: 동일한 제품을 만드는 프로세스와 그 제품을 만드는 장치
 - SW 플랫폼: 소프트웨어를 실행할 수 있는 기반이나 환경
 - 서비스 플랫폼: 서비스 제공자의 서비스를 다른 서비스들이 쉽게 사용할 수 있게 해주는 환경
- 빅데이터 플랫폼: 기업 내에 많은 사용자들이 데이터를 처리하고 분석을 쉽게 할 수 있는 환경을 제공해주는 시스템
 - 데이터 수집, 처리 및 저장 기능 제공
 - 데이터 검색 및 보안 제공
 - 데이터 분석 기능 제공

데이터 파이프라인

■ 파이프라인이란?

- 하나의 데이터 처리 단계의 출력이 다음 단계의 입력으로 이어지는 형태로 연결된 구조

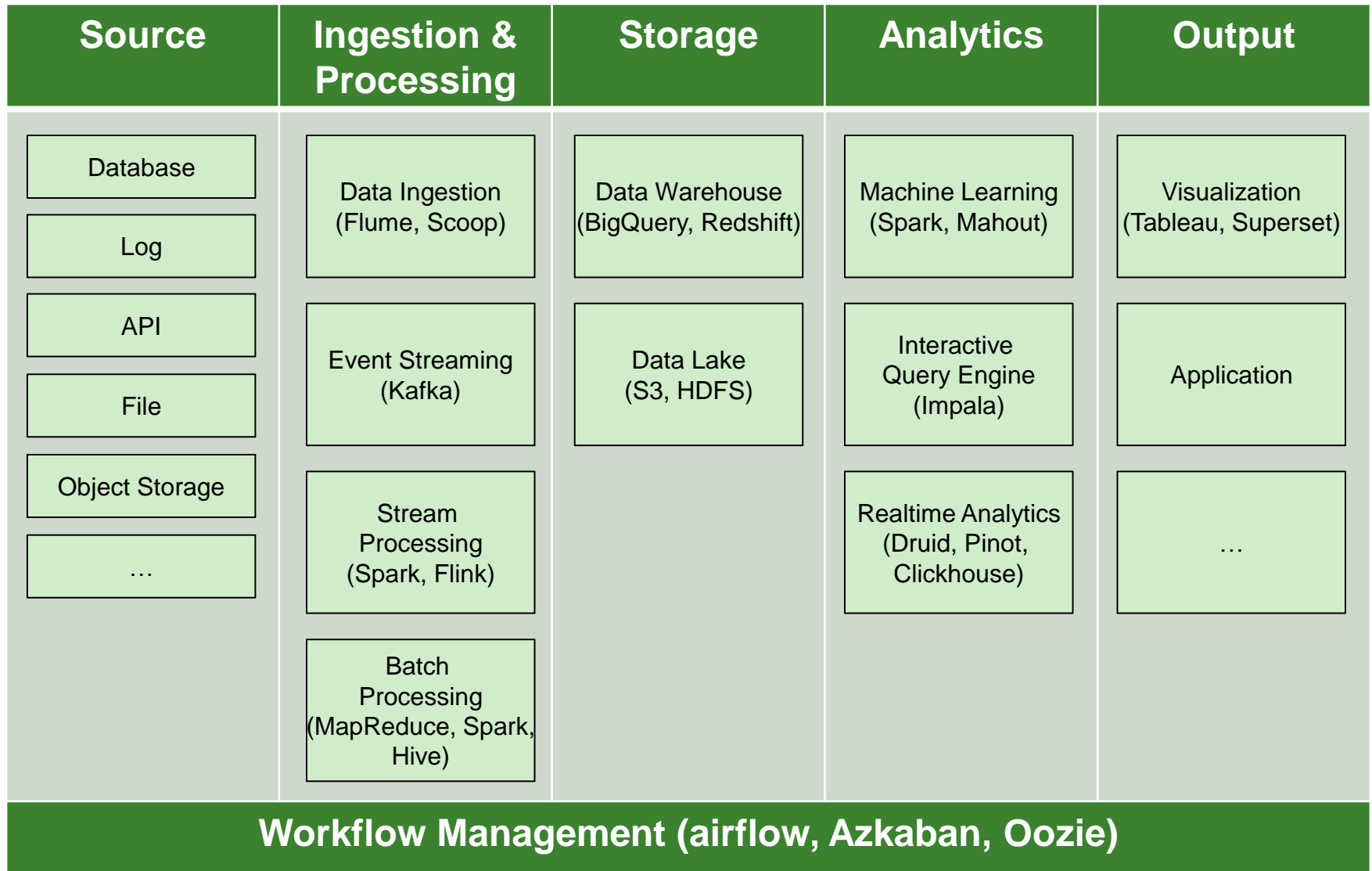
- e.g., Linux pipeline



■ 데이터 파이프라인이란?

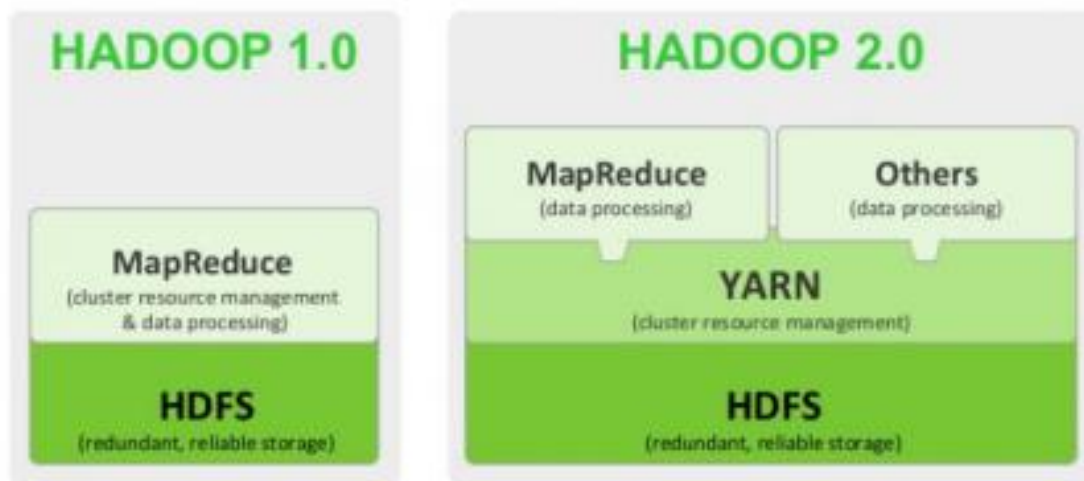
- 다양한 소스에서 데이터를 변환하고 옮기는 과정을 구성한 시스템
 - Source → Processing → Destination
- 데이터 기반 의사 결정, 데이터 기반 응용 프로그램을 위해서 사용
- 데이터 엔지니어들이 주로 수행

데이터 파이프라인 예시



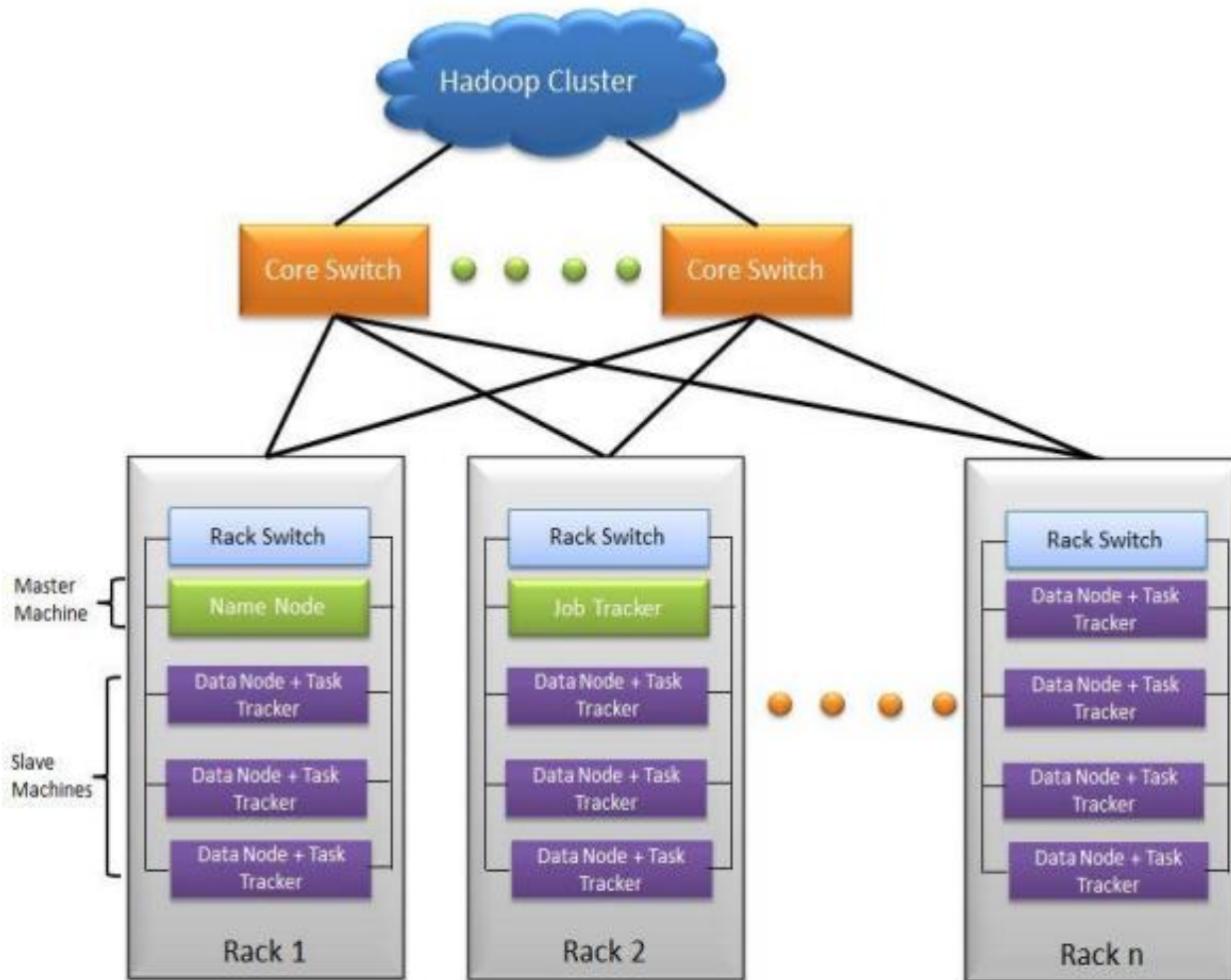
Hadoop

- 오픈소스 자바 소프트웨어 프레임워크
- 아파치 루씬의 하부 프로젝트로부터 2005년 분리
- 2006년 아파치 top 프로젝트로 승격
(<http://hadoop.apache.org>)
- 하둡 분산 파일 시스템(HDFS: Hadoop Distributed File System)과 맵리듀스(MapReduce)로 구성
- 저장 인프라 개념과 개발 SW 개념을 동시에 포함

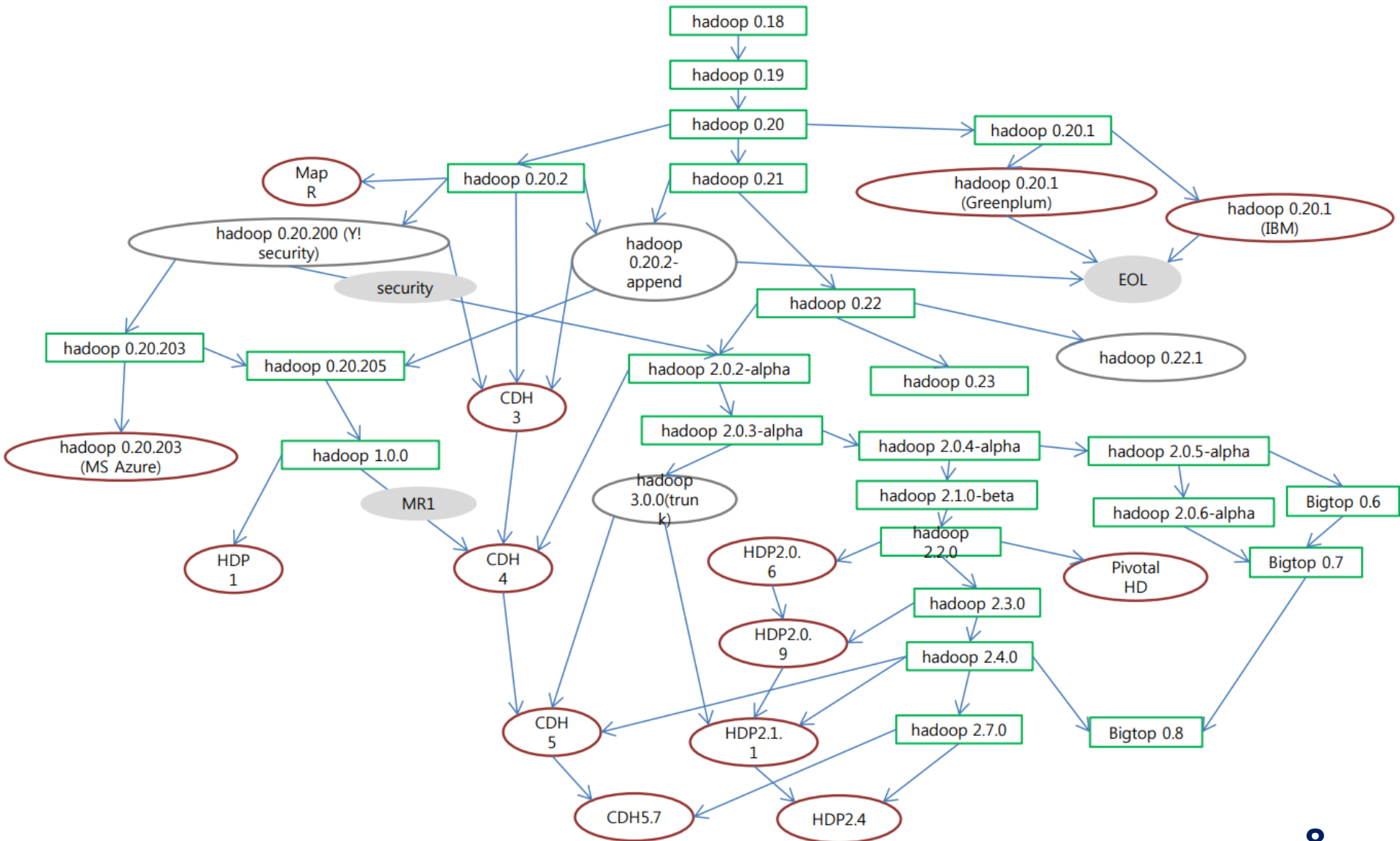


Hadoop 구성




■ 마스터(master)-슬레이브(slave) 구조



Hadoop 변천사



Hadoop 배포판

	HDP 	CDH 	MapR 
Open Source	100% - all code contributed to Apache, ZERO proprietary	Open Core, focus on proprietary, packagers of open	Rewritten core, no longer Apache Hadoop, proprietary
Expertise	19 Hadoop Core 95 Across All Project	8 Hadoop Core ~20 Across All Project	1 Hadoop Core <10 Across All Projects
Focus & Strategic innovation	Hadoop 1, Hadoop 2, Ambari, Hive, Ecosystem	Proprietary Components, Impala and Manager	HBase Rewrite HDFS Rewrite
Hadoop 2 –next gen Hadoop beyond batch	Architected & Built 95%, only company ready to support	Limited involvement, limited knowledge of YARN	Packager, unqualified to support
Use Case Fit	Infrastructure enablement and specific use cases	Infrastructure enablement and specific use cases	Online use cases mostly Not well suited for infrastructure
Platform Support	Windows, Linux	Linux Only	Linux Only
QA/Test Readiness	Tested on thousands of node, production ready release only	Limited test environment, release beta as GA	Limited test environment
Ecosystem Enablement & Interoperability	Enable and empower ALL apps to benefit from Hadoop. Focus on app vendors like Microsoft, Teradata, Splunk	Competitive with Impala, no focus on Hive. Focus on pull through hardware vendors like Dell and HP	Vertical use case focus moves up stack to competitive. Focus on AWS. Unclear app focus.
Leadership	Invented open source model	Traditional enterprise software	Traditional enterprise software

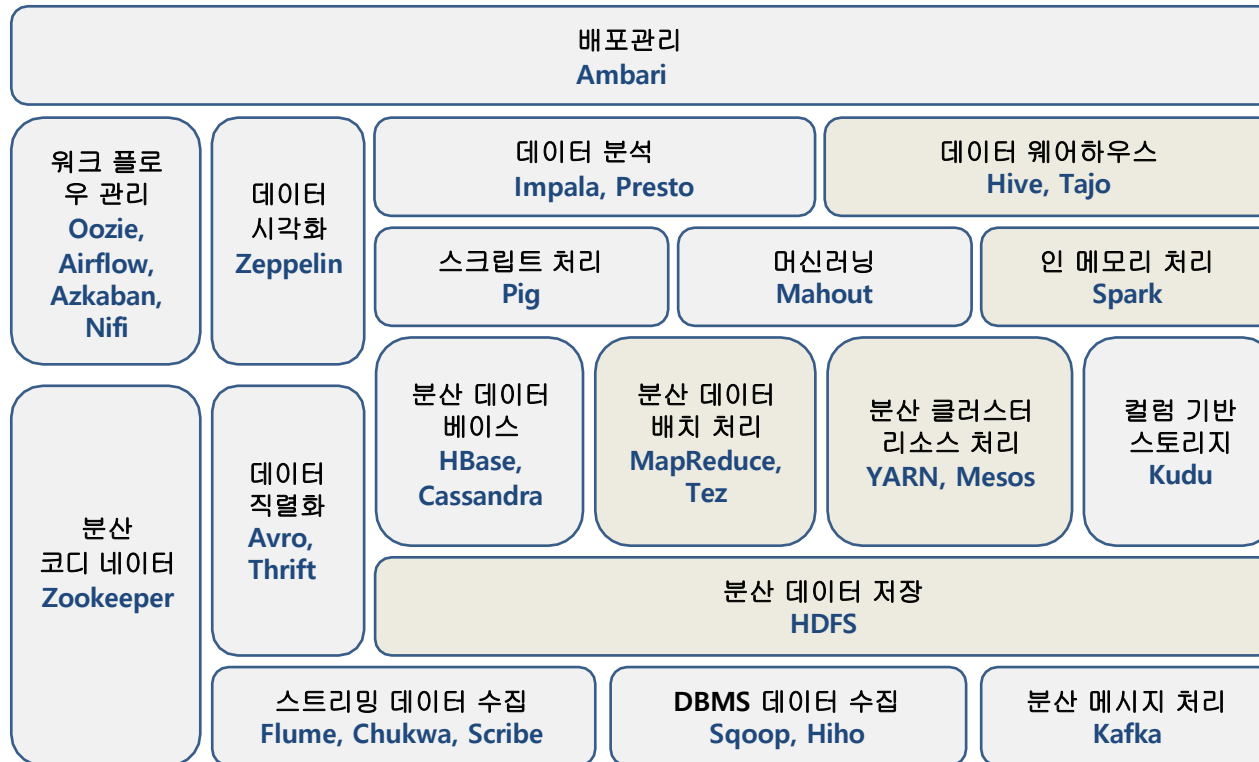
Hadoop Ecosystem

- 하둡 기반의 다양한 서비스 플랫폼을 구성하기 위한 서비스 프로젝트 또는 연관 프로젝트의 조합
 - 오픈 소스 위주 100 개 이상
(<http://hadoopecosystemtable.github.io/>)

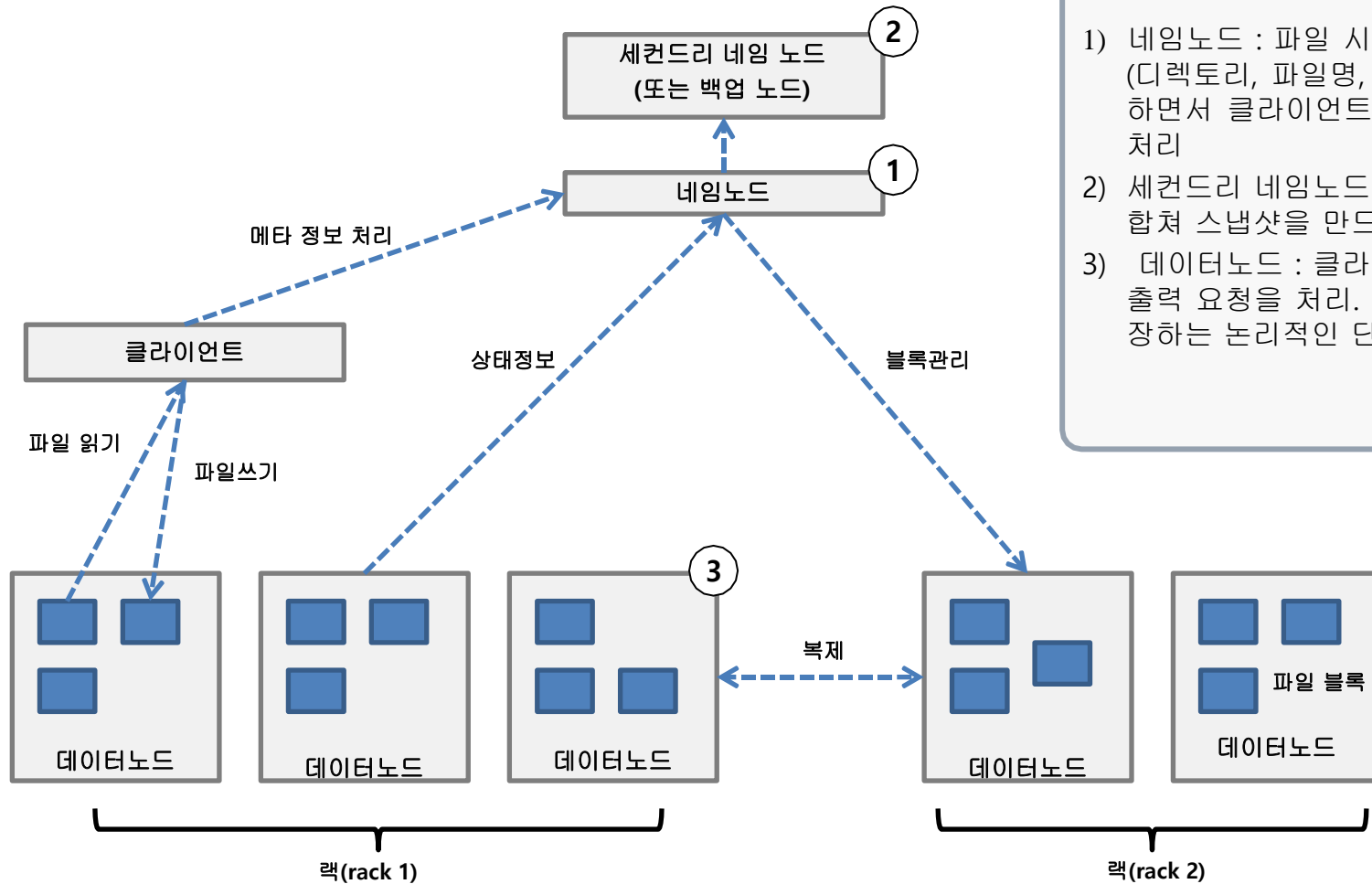
Distributed Filesystem	Distributed Programming		NoSQL Databases			NewSQL	Categorize Pending ...
			Columne	Document	Key-Value		
<ul style="list-style-type: none">• Apache HDFS• GlusterFS• Quantcast File System QFS• Ceph Filesystem• Lustre file system• Tachyon• GridGain	<ul style="list-style-type: none">• MapReduce• Apache Pig• JAQL• Apache Spark• Apache Flink• Netflix PigPen• AMPLab SIMR• Facebook Corona• Apache Twill	<ul style="list-style-type: none">• Damballa Parkour• Apache Hama• Datasalt Pangool• Apache Tez• Apache DataFu• Pydoop• Kangaroo• TinkerPop• Pachyderm	<ul style="list-style-type: none">• Apache HBase• Apache Cassandra• Hypertable• Apache Accumulo	<ul style="list-style-type: none">• MongoDB• RethinkDB• ArangoDB	<ul style="list-style-type: none">• Redis DataBase• Linkedin Voldemort• RocksDB• OpenTSDB	<ul style="list-style-type: none">• TokuDB• HandlerSocket• Akiban Server• Drizzle• Haeinsa• SenseiDB• Sky• BayesDB• InfluxDB	<ul style="list-style-type: none">• Twitter Summingbird• Apache Kiji• S4 Yahoo• Metamarkers Druid• Concurrent Cascading• Concurrent Lingual• Concurrent Pattern• Apache Giraph• Talend• Akka Toolkit• Eclipse BIRT• Spango BI• Jedox Palo• Twitter Finagle• Intel GraphBuilder• Apache Tika
				Graph			
				<ul style="list-style-type: none">• ArangoDB• Neo4j• TitanDB	Stream <ul style="list-style-type: none">• EventStore		
SQL-On-Hadoop	Data Ingestion	Service Programming	Scheduling	Machine Learning	Benchmark	System Deployment	
<ul style="list-style-type: none">• Apache Hive• HCatalog• Trafodion: SQL-on-HBase• Apache Drill• Cloudera Impala• Facebook Presto• Splout SQL• Apache Tajo• Apache Phoenix• Apache MRQL• Kylin	<ul style="list-style-type: none">• Apache Flume• Apache Sqoop• Facebook Scribe• Apache Chukwa• Apache Storm• Apache Kafka• Netflix Suro• Apache Samza• Cloudera Morphline• HIHO• Apache NiFi	<ul style="list-style-type: none">• Apache Thrift• Apache Zookeeper• Apache Avro• Apache Curator• Apache karaf• Twitter Elephant Bird• Linkedin Norbert	<ul style="list-style-type: none">• Oozie• Azkaban• Apache Falcon	<ul style="list-style-type: none">• Apache Mahout• WEKA• Cloudera Oryx• MADlib• H2O• Sparkling Water	<ul style="list-style-type: none">• Apache Hadoop Benchmarking• Yahoo Gridmix3• PUMA Benchmarking• Berkeley SWIM Benchmark• Intel HiBench	<ul style="list-style-type: none">• Ambari• HUE• Whirr• Mesos• Myriad• Marathon• Brooklyn• HOYA• Helix• Bigtop• Buildoop• Deploop	
			Security <ul style="list-style-type: none">• Sentry• Knox Gateway• Ranger				

주요 Hadoop Ecosystem

■ Hadoop 2.0 기준



HDFS Architecture (V1)



- 1) 네임노드 : 파일 시스템의 네임스페이스 (디렉토리, 파일명, 파일블록 등)을 관리 하면서 클라이언트의 파일 접근 요청을 처리
- 2) 세컨드리 네임노드 : 네임노드의 정보를 합쳐 스냅샷을 만드는 기능수행
- 3) 데이터노드 : 클라이언트의 데이터 입출력 요청을 처리. 파일의 데이터를 저장하는 논리적인 단위인 블록을 관리.

하둡 분산 파일 시스템 (HDFS) 이해

■ HDFS는 Google File System (GFS)를 오픈소스 버전으로 구현한 것

The Google File System

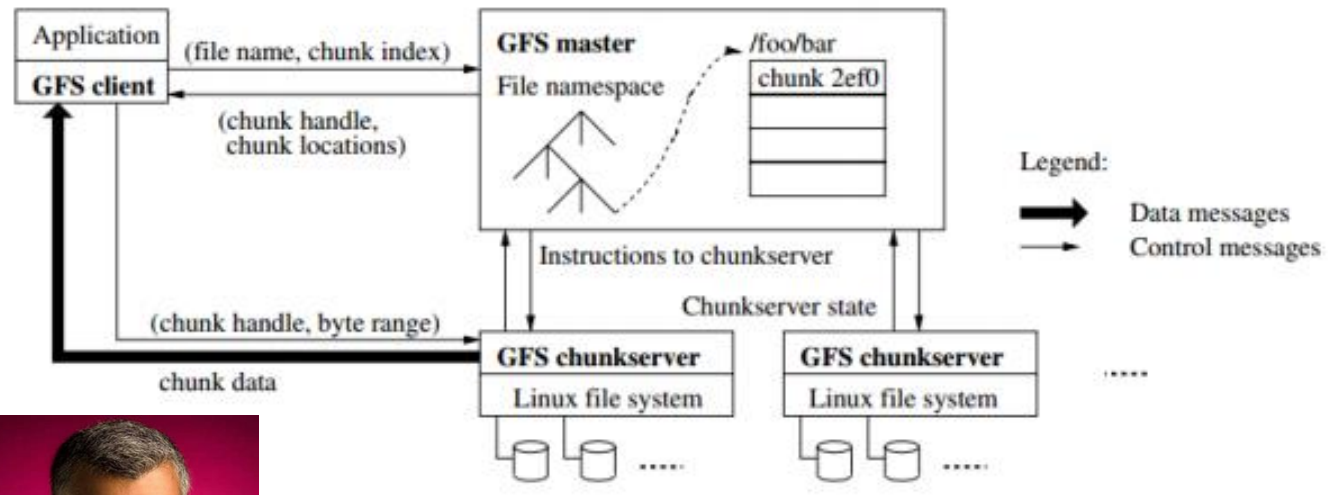
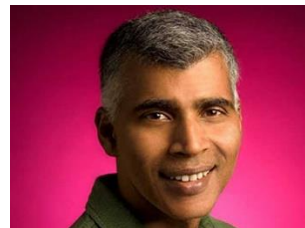


Figure 1: GFS Architecture



- Ghemawat, Sanjay, Howard Gobioff, and Shun-Tak Leung. "The Google file system." Proceedings of the nineteenth ACM symposium on Operating systems principles. 2003.
- # Citation: 10,131회 (Google Scholar 기준)

하둡 분산 파일 시스템 (HDFS) 이해 (계속)

■ Google의 플랫폼 철학

- Scale-out 지향
- 쉬운 시스템 확장 지향
- 결함허용성(Fault Tolerance) 보장
 - 다수의 이유로 H/W는 정상 동작 하지 않을 수 있음
- 데이터 분산 저장 지향

하둡 분산 파일 시스템 (HDFS) 이해 (계속)

■ 하둡의 특성

- 수천대 이상의 Linux 기반 범용 서버(commodity machines)들을 하나의 클러스터로 엮어서 활용
- 마스터-슬레이브(master-slave) 구조
- 파일은 블록(block) 단위로 저장
- 신뢰성 향상
 - 블록 데이터의 복제본(기본적으로 3개의 복제본) 유지
- 데이터 처리의 지역성 보장

하둡 분산 파일 시스템 (HDFS) 이해 (계속)

■ 하둡에서 **하나의 파일을 여러 개의 블록들로 저장**

- 주로 128MB 또는 64MB 등의 큰 크기로 설정하며, 설정을 통해서 블록의 크기를 조정 가능함

■ 하둡에서 **블록의 크기가 큰 이유**

- 탐색 비용을 최소화
- HDD에서 블록의 시작점을 탐색하는 데 걸리는 시간을 최소화
- 블록 전송 시 네트워크 대역폭을 최대한 활용

하둡 분산 파일 시스템 (HDFS) 이해 (계속)

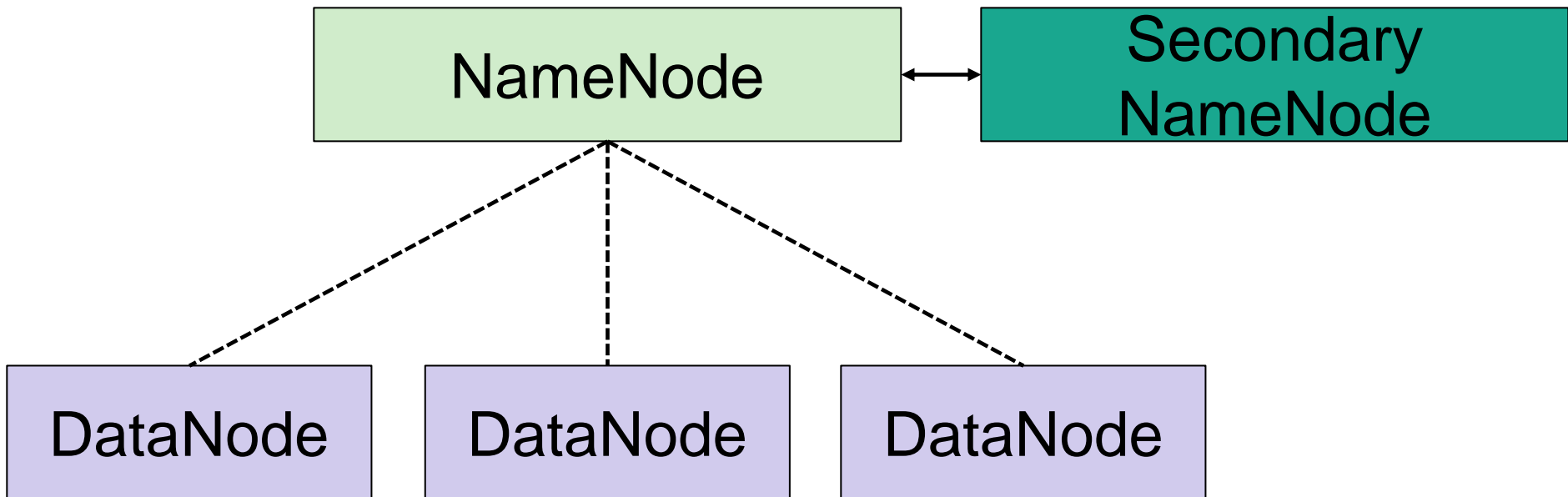
■ 블록 단위 파일 저장의 장점

- 디스크 사이즈보다 더 큰 파일을 저장 가능함
 - 예: $800\text{GB} * 2 = 1.6\text{ TB}$ 의 HDFS에 1TB의 파일 저장 가능함
- 블록 단위의 추상화를 통해 스토리지의 서브 시스템을 단순하게 만들 수 있음
 - 파일 탐색 지점이나 메타 정보를 저장할 때 사이즈가 고정되어 있어, 구현이 용이함
- 복제(Replication)를 통한 내고장성(Fault Tolerance) 구현할 때 용이함
 - 같은 노드에 같은 블록이 존재하지 않도록 복제 후, 노드가 고장일 경우 다른 노드의 블록을 이용하여 복구
- I/O 속도를 높일 수 있음
 - 같은 파일을 분산 처리하여 데이터 처리 성능을 개선 할 수 있음

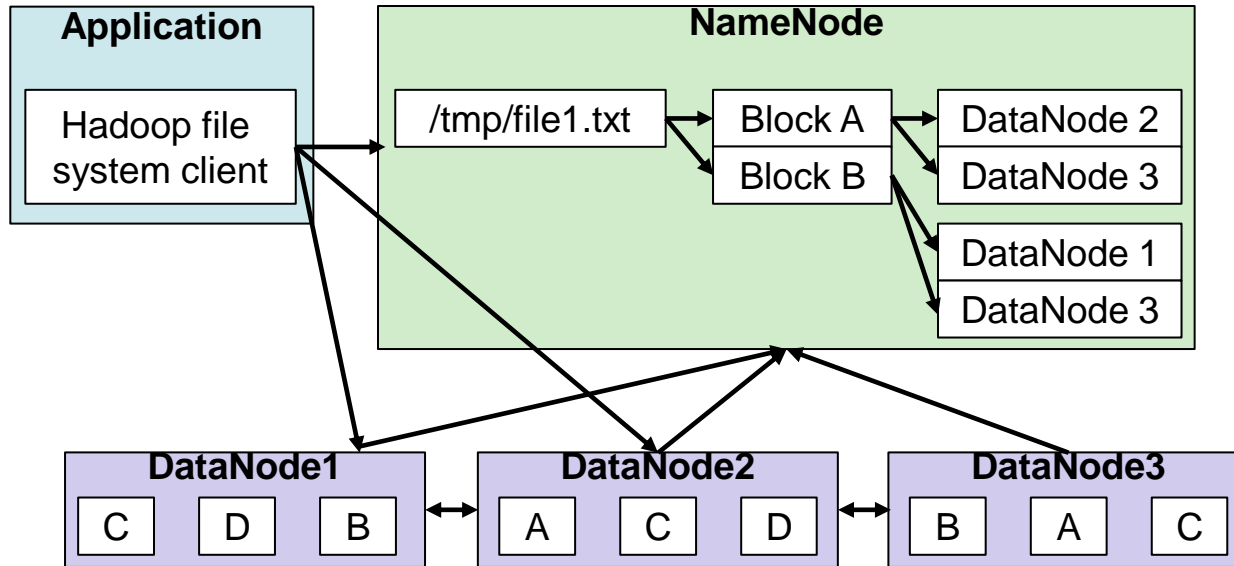
하둡 분산 파일 시스템 (HDFS) 이해 (계속)

■ 네임노드(NameNode)와 데이터노드(DataNode)

- 마스터-슬레이브(master-slave) 구조



하둡 분산 파일 시스템 (HDFS) 이해 (계속)



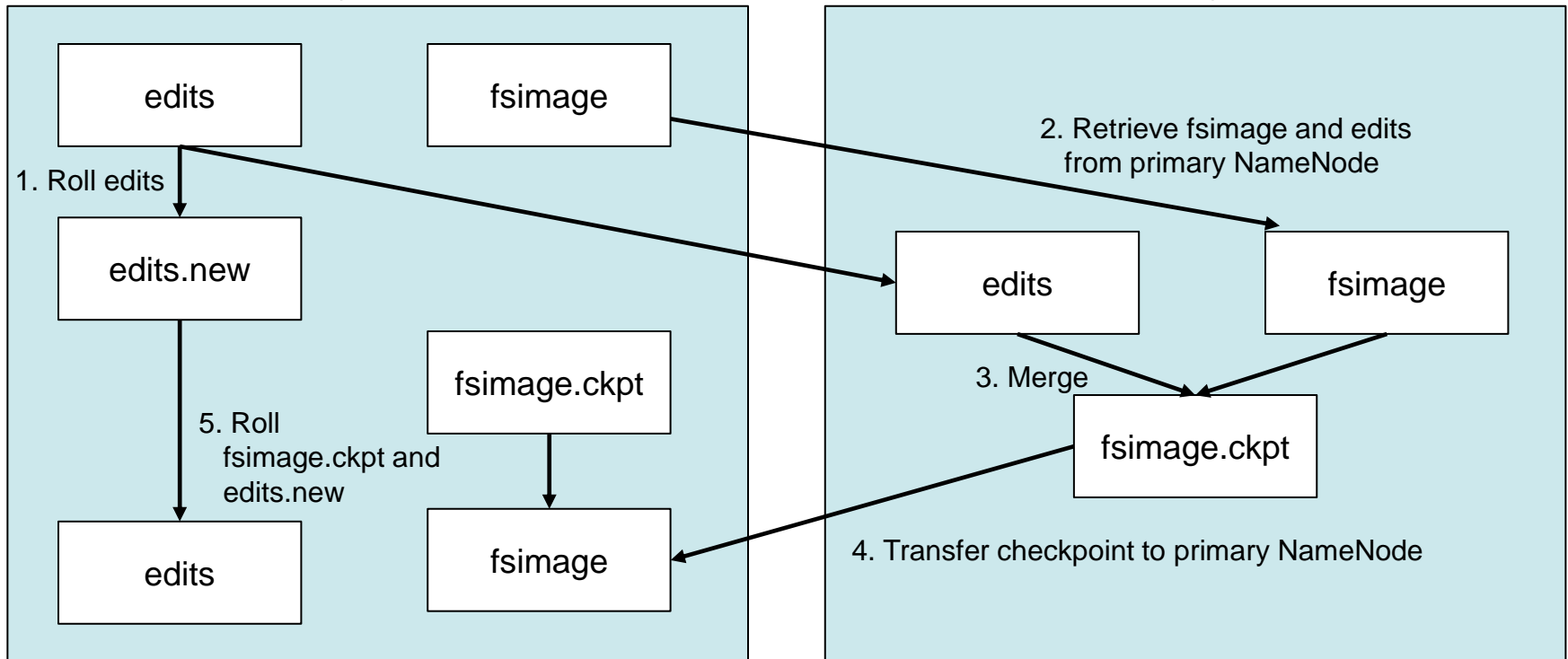
■ 네임노드(NameNode)

- 메타데이터 관리
 - 전체 HDFS에 대한 Name Space를 관리함
 - FsImage(파일 시스템 이미지): 네임스페이스를 포함한 데이터의 모든 정보임
 - EditLog: 데이터 노드에서 발생한 데이터 변환 내역임
- DataNode 관리
 - DataNode로부터 Block을 리포트 받음

하둡 분산 파일 시스템 (HDFS) 이해 (계속)

Primary NameNode

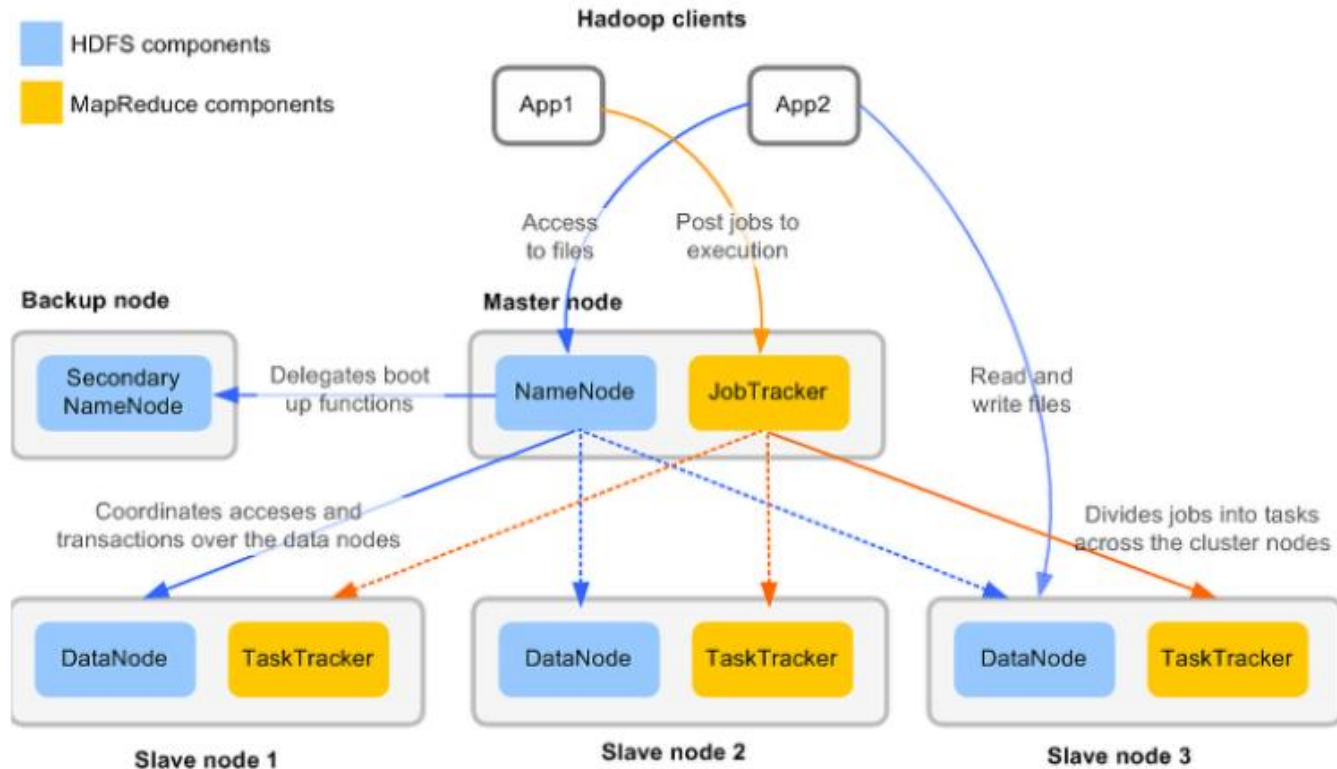
Secondary NameNode



■ 보조 네임노드(Secondary NameNode)

- 네임노드와 보조네임노드는 **Active/Standby 구조가 아님**
- FsImage와 EditLog를 주기적으로 병합함
- 1시간 주기 또는 EditLog가 일정 사이즈 이상이면 체크 포인트 수행
- 단일 장애점 (Single Point of Failure, SPOF) 문제가 발생 가능

하둡 분산 파일 시스템 (HDFS) 이해 (계속)



■ 데이터노드(DataNode)

- 물리적으로 로컬 파일시스템에 HDFS 데이터를 저장함
- 데이터노드는 HDFS에 대한 정보가 없음
- 데이터노드는 네임노드에게 블록 리포트를 함
 - 네임노드가 시작될 때, 주기적으로 로컬 파일 시스템에 있는 모든 HDFS 블록들을 검사한 후 정상적인 블록의 목록을 만들어 네임노드에 전송

하둡 분산 파일 시스템 (HDFS) 이해 (계속)

■ 데이터노드 블록스캐너는 주기적으로 블록 리포팅을 수행

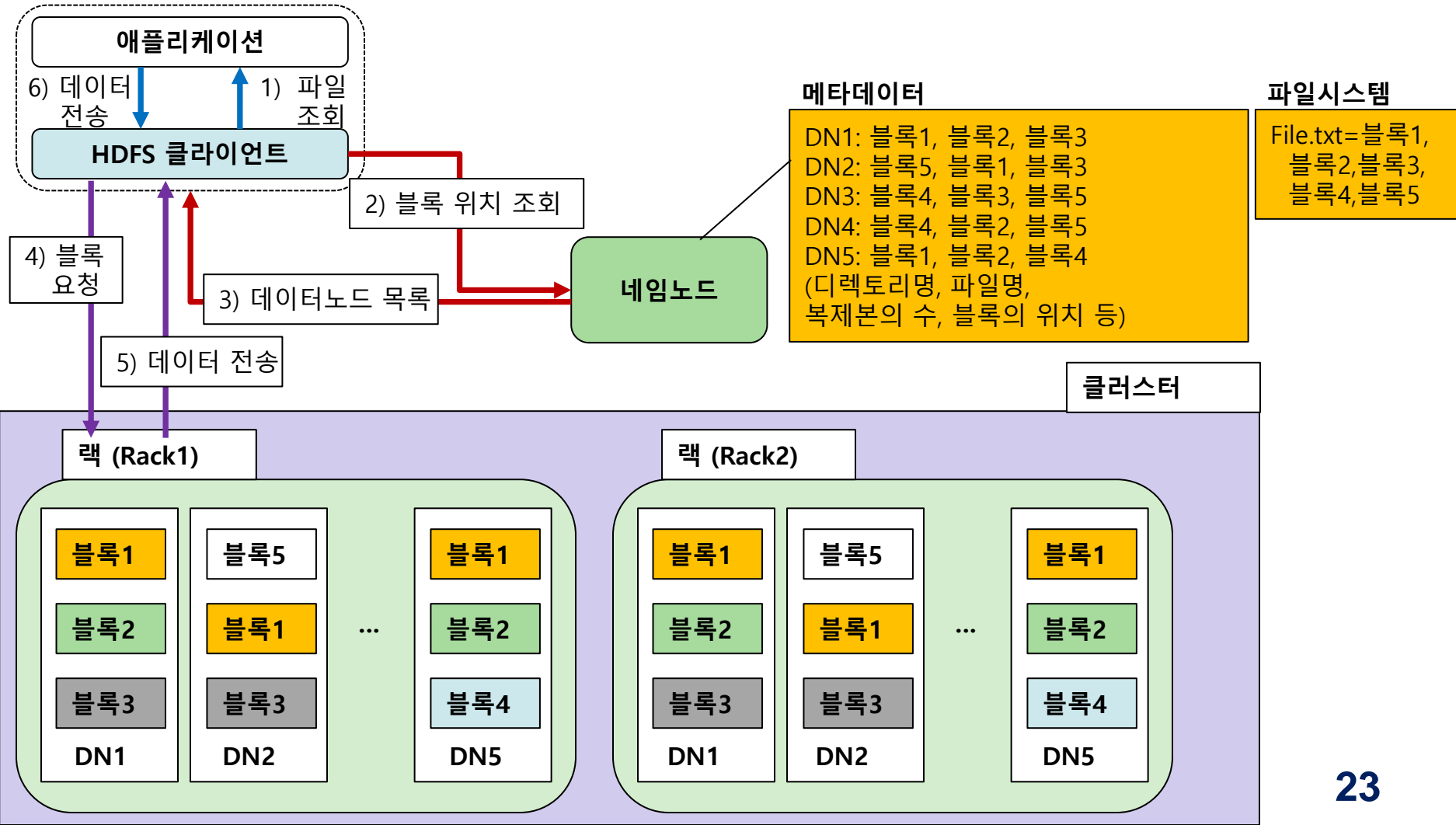
□ <http://datanode:50075/blockScannerReport>

Block report for block pool: BP-2125152513-172.31.45.216-1410037307133

Total Blocks	:	33254
Verified in last hour	:	0
Verified in last day	:	0
Verified in last week	:	33254
Verified in last four weeks	:	33254
Verified in SCAN_PERIOD	:	33254
Not yet verified	:	0
Verified since restart	:	370457
Scans since restart	:	370457
Scan errors since restart	:	0
Transient scan errors	:	0
Current scan rate limit Kbps	:	1024
Progress this period	:	198%
Time left in cur period	:	86.90%

하둡 분산 파일 시스템 (HDFS) 이해 (계속)

■ HDFS 읽기 연산



하둡 분산 파일 시스템 (HDFS) 이해 (계속)

■ HDFS 읽기 연산

1. 사용자 어플리케이션이 클라이언트에게 파일 읽기를 요청함
2. 클라이언트는 네임노드에게 어플리케이션이 요청한 파일이 어떤 블록에 저장돼 있는지 블록의 위치 정보를 요청함
3. 네임노드는 요청된 파일의 복제 블록이 저장된 위치를 메타데이터를 통해 확인한 후 해당 데이터노드의 목록을 반환함. 이 때 데이터노드의 목록은 요청 중인 클라이언트와 가까운 순서대로 정렬해서 반환됨
4. 클라이언트는 데이터노드에게 블록을 조회할 것을 요청함
5. 데이터노드는 클라이언트에게 요청한 데이터를 전송함
6. 클라이언트는 어플리케이션에게 전송 받은 데이터를 전달함

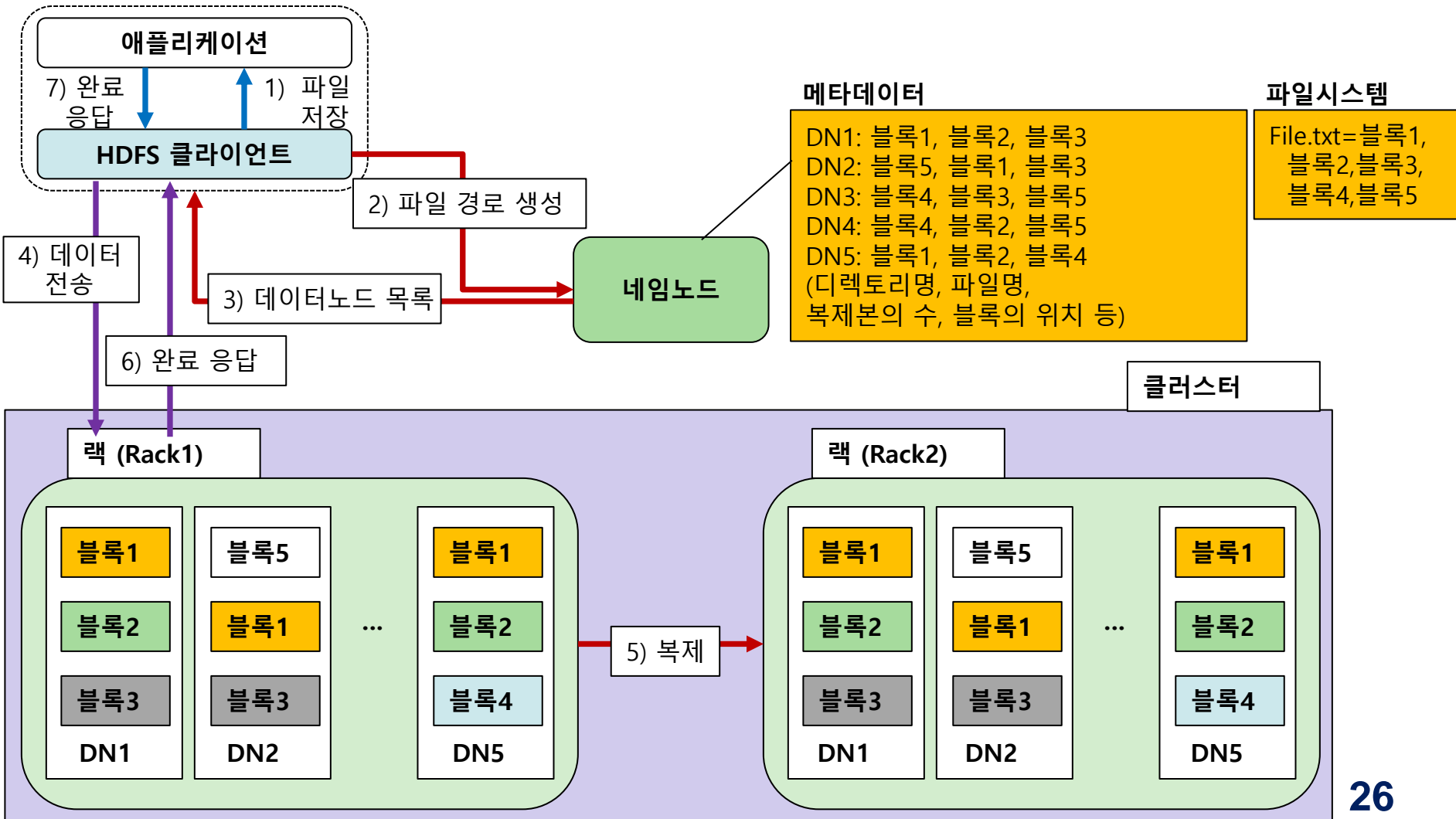
하둡 분산 파일 시스템 (HDFS) 이해 (계속)

■ Java활용 HDFS 파일 읽기 샘플 코드

```
FileSystem fileSystem = FileSystem.get(conf);
Path path = new Path("/path/to/file.ext");
if (!fileSystem.exists(path)) {
    System.out.println("File does not exists");
    return;
}
FSDataInputStream in = fileSystem.open(path);
int numBytes = 0;
while ((numBytes = in.read(b)) > 0) {
    System.out.println((char)numBytes);
    // code to manipulate the data which is read
}
in.close();
out.close();
fileSystem.close();
```

하둡 분산 파일 시스템 (HDFS) 이해 (계속)

■ HDFS 쓰기 연산



하둡 분산 파일 시스템 (HDFS) 이해 (계속)

■ HDFS 쓰기 연산

1. 사용자 어플리케이션이 클라이언트에게 파일 저장을 요청함
2. 클라이언트는 네임노드에게 사용자가 요청한 파일 경로를 생성할 것을 요청함
3. 네임노드는 해당 파일 경로가 기존에 존재할 경우 에러 처리를 함. 기존에 존재하지 않는다면 메모리에 파일 경로를 생성한 후, 다른 클라이언트가 해당 경로를 수정하지 못하게 락을 걸
4. 네임노드는 클라이언트에게 해당 파일을 저장할 데이터노드의 목록을 반환함 (환경설정 파일 내의 “블록의 복제 개수”만큼)
5. 클라이언트는 첫 번째 네임노드에게 데이터를 전송함

하둡 분산 파일 시스템 (HDFS) 이해 (계속)

■ HDFS 쓰기 연산

6. 첫 번째 데이터노드는 전송받은 데이터를 로컬 디스크에 저장한 후, 두 번째 데이터노드로 데이터를 전송함
7. 두 번째 데이터노드도 전송 받은 데이터를 로컬에 저장한 후, 세 번째 데이터노드로 데이터를 전송함
8. 세 번째 데이터도 전송 받은 데이터를 로컬에 저장함
9. 두 번째와 세 번째 데이터노드는 첫 번째 데이터노드에게 로컬 저장이 완료됐다는 사실을 알려줌
10. 첫 번째 데이터노드는 클라이언트에게 파일 저장이 완료됐음을 응답함
11. 클라이언트는 어플리케이션에게 파일 저장이 완료됐음을 응답함

하둡 분산 파일 시스템 (HDFS) 이해 (계속)

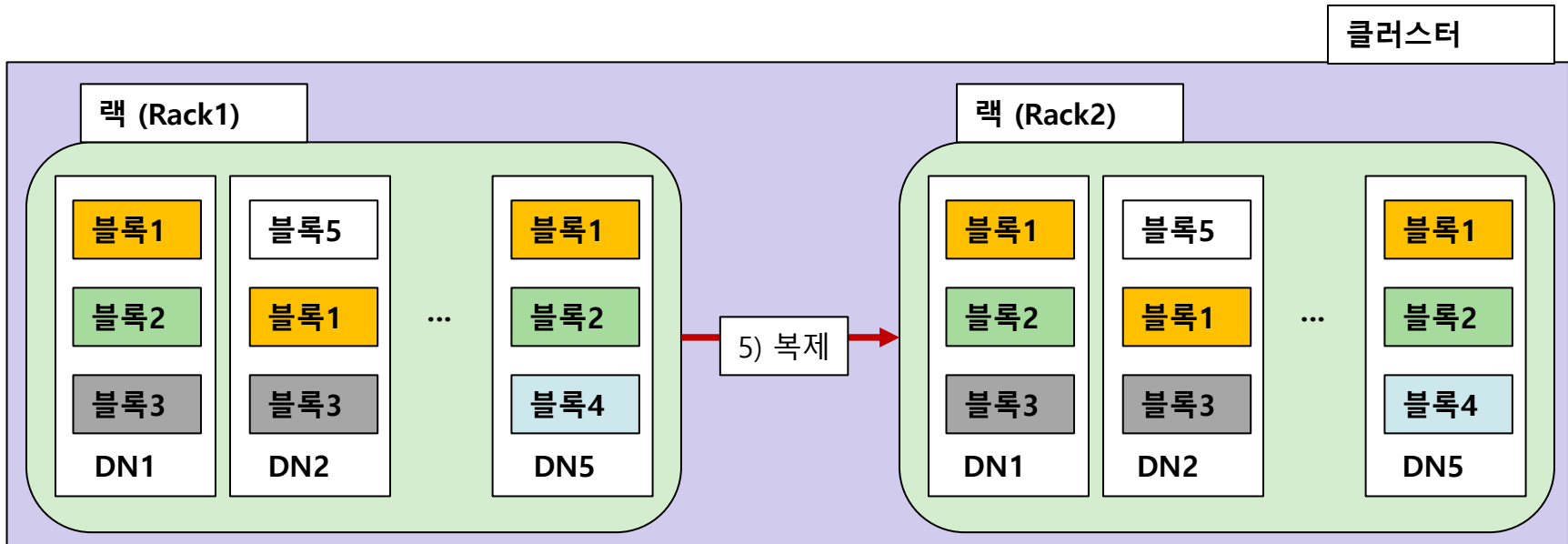
■ Java활용 HDFS 파일 쓰기 샘플 코드

```
FileSystem fileSystem = FileSystem.get(conf);
// Check if the file already exists
Path path = new Path("/path/to/file.ext");
if (fileSystem.exists(path)) {
    System.out.println("File " + dest + " already exists");
    return;
}
// Create a new file and write data to it.
FSDataOutputStream out = fileSystem.create(path);
InputStream in = new BufferedInputStream(new FileInputStream(
    new File(source)));
byte[] b = new byte[1024];
int numBytes = 0;
while ((numBytes = in.read(b)) > 0) {
    out.write(b, 0, numBytes);
}
// Close all the file descriptors
in.close();
out.close();
fileSystem.close();
```

하둡 분산 파일 시스템 (HDFS) 이해 (계속)

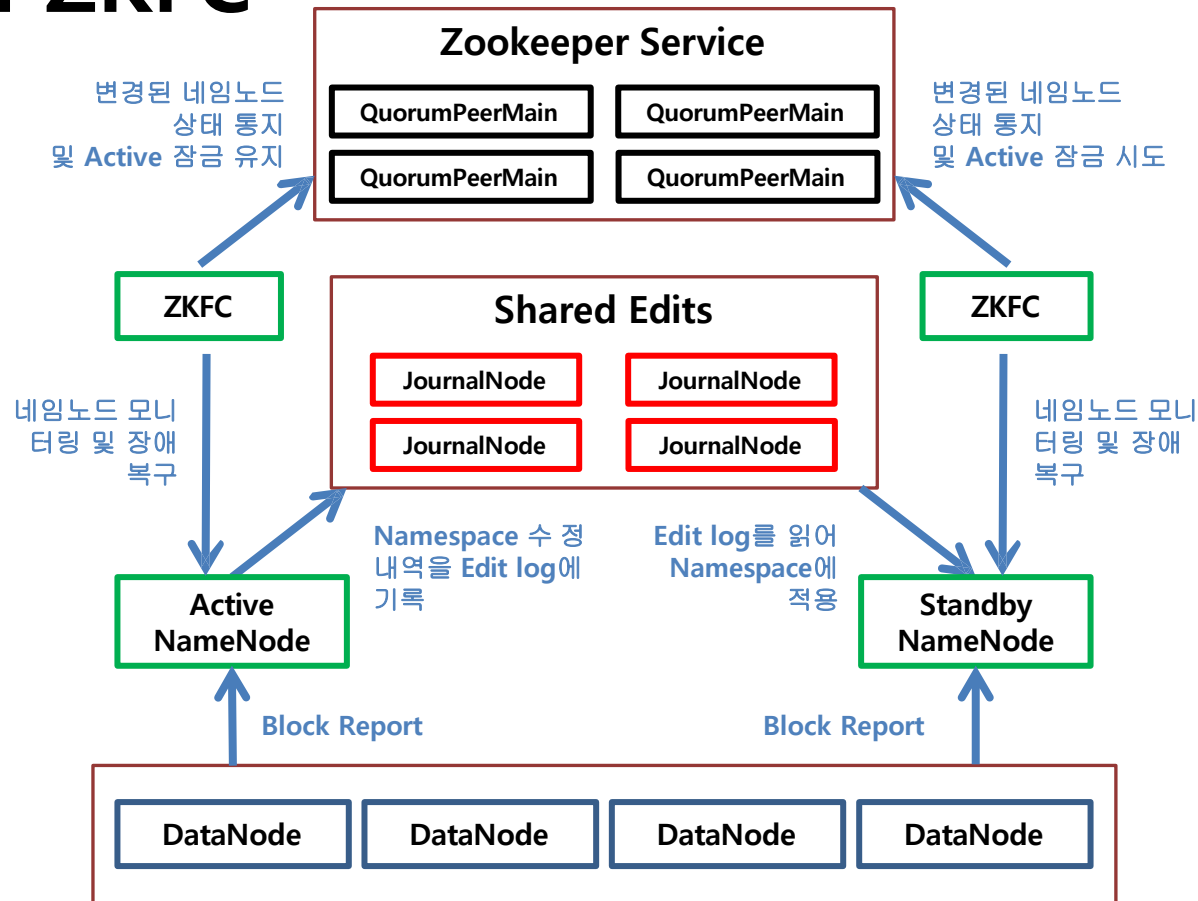
■ Rack Awareness: 랙 단위 장애(전원, 스위치 장애) 발생에도 전체 블록이 유실되지 않도록 구성

- 블록을 저장할 때, 2개의 블록은 같은 랙에 저장
- 나머지 하나의 블록은 다른 랙에 저장되도록 구성



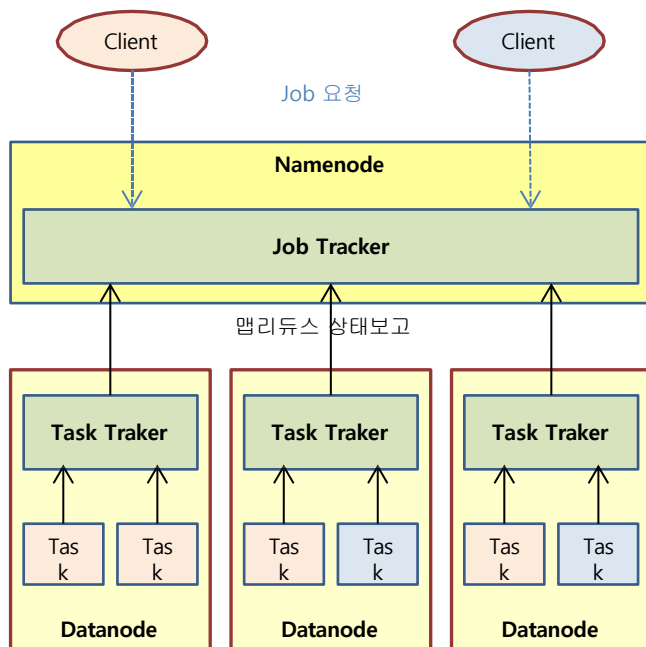
HDFS Architecture (V2)

■ Hadoop High Availability Architecture with ZKFC

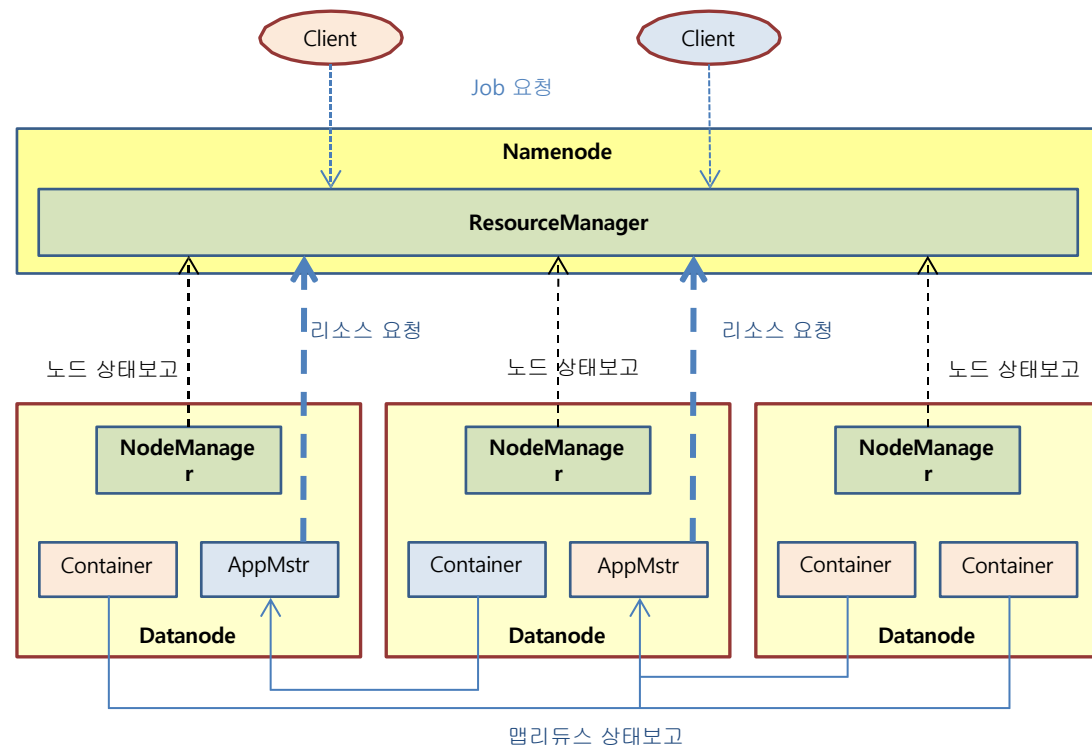


MapReduce Architecture (V2)

맵리듀스 Architecture



안(YARN) Architecture



네임노드 고가용성 (High Availability)

- HDFS 고가용성은 이중화된 두대의 서버인 액티브(active) 네임노드와 스탠바이(standby) 네임노드를 이용하여 지원
- 액티브 네임노드와 스탠바이 네임노드는 데이터 노드로부터 블록 리포트와 하트비트를 모두 받아서 동일한 메타데이터를 유지하고, 공유 스토리지를 이용하여 에디트파일을 공유
- 액티브 네임노드는 네임노드의 역할을 수행하고, 스탠바이 네임노드는 액티브 네임노드와 동일한 메타데이터 정보를 유지하다가, 액티브 네임노드에 문제가 발생하면 스탠바이 네임노드가 액티브 네임노드로 동작
- 액티브 네임노드에 문제가 발생하는 것을 자동으로 확인하는 것이 어렵기 때문에 보통 주키퍼를 이용하여 장애 발생시 자동으로 변경될 수 있도록 구성함

HDFS 세이프 모드(Safemode)

- HDFS의 세이프 모드(Safemode)는 데이터노드를 수정할 수 없는 상태임
 - 읽기 전용 상태가 됨
- 네임노드에 문제가 생겨서 정상적인 동작을 할 수 없는 경우 자동으로 세이프모드로 전환
- 세이프 모드 상태일 경우 파일 복사도 불가능 (아래와 같은 에러 발생)
 - `$ hdfs fs -put sample.txt /user/sample.txt`
 - *put: Cannot create file /user/sample2.txt._COPYING_. Name node is in safe mode.*

HDFS 세이프 모드(계속)

- HDFS 운영 중 세이프 모드가 발생한 경우 문제 파악이 필요(네임노드의 문제? 데이터노드의 문제?)
 - fsck 명령어: HDFS의 무결성 확인
 - hdfs dfsadmin -report: 각 데이터 노드의 상태를 확인
 - 문제 해결 후 세이프 모드를 해제
- 세이프 모드 상태 확인
 - `$ hdfs dfsadmin -safemode get`
 - Safemode is OFF
- 세이프 모드 진입
 - `$ hdfs dfsadmin -safemode enter`
 - Safe mode is ON
- 세이프 모드 해제
 - `$ hdfs dfsadmin -safemode leave`
 - Safe mode is OFF

HDFS 세이프 모드(계속)

- HDFS 운영 중 세이프 모드가 발생한 경우 문제 파악이 필요(네임노드의 문제? 데이터노드의 문제?)
 - fsck 명령어: HDFS의 무결성 확인
 - hdfs dfsadmin -report: 각 데이터 노드의 상태를 확인
 - 문제 해결 후 세이프 모드를 해제
- 세이프 모드 상태 확인
 - \$ hdfs dfsadmin -safemode get
 - Safemode is OFF
- 세이프 모드 진입
 - \$ hdfs dfsadmin -safemode enter
 - Safe mode is ON
- 세이프 모드 해제
 - \$ hdfs dfsadmin -safemode leave
 - Safe mode is OFF

Corrupt Block

- HDFS는 Heartbeat를 통해 데이터 블록에 문제가 생기는 것을 감지하고 자동으로 복구 가능함
- 보통 다른 데이터노드에 복제된 데이터를 가져와 복구를 진행하지만, 모든 블록(Replication)에 문제가 생겨서 복구하지 못하면 Corrupt 상태가 됨
- Corrupt 상태의 파일들은 삭제 후 HDFS에 재적재 해주어야 함

■ "hadoop fs" 명령어를 통해

"hadoop fs" 명령

Options

<Path>

-move

-delete

-openforwrite

-files

-blocks

-locations

-racks

```
Status: HEALTHY
Number of data-nodes: 1
Number of racks: 1
Total dirs: 21
Total symlinks: 0

Replicated Blocks:
Total size: 9696971 B
Total files: 4
Total blocks (validated): 4 (avg. block size 2424242 B)
Minimally replicated blocks: 4 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 0 (0.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 1
Average block replication: 1.0
Missing blocks: 0
Corrupt blocks: 0
Missing replicas: 0 (0.0 %)
Blocks queued for replication: 0

Erasure Coded Block Groups:
Total size: 0 B
Total files: 0
Total block groups (validated): 0
Minimally erasure-coded block groups: 0
Over-erasure-coded block groups: 0
Under-erasure-coded block groups: 0
Unsatisfactory placement block groups: 0
Average block group size: 0.0
Missing block groups: 0
Corrupt block groups: 0
Missing internal blocks: 0
Blocks queued for replication: 0
FSCK ended at Sun Sep 20 22:31:56 KST 2020 in 4 milliseconds

The filesystem under path '/' is HEALTHY
```

ere

lirectory

FS

checking

while checking

ode locations

Corrupt 블록 상태 확인 및 조치

■ Corrupt 블록 상태 확인

- ❑ `$ hdfs fsck /user/Hadoop`
- ❑ The filesystem under path '/user/Hadoop' is
CORRUPT **Corrupt 상태가 발생한 것 확인**
- ❑ `$ hdfs fsck -delete` **Corrupt 파일 삭제**
- ❑ `$ hadoop fs -setrep 5 -R /user/Hadoop`
/user/Hadoop 의 하위의 모든 파일의 복제 수를 조정

감사합니다!

담당교수: 전강욱(컴퓨터공학부)

kw.chon@koreatech.ac.kr

하둡 분산 파일 시스템 (HDFS) 이해 (계속)

■ HDFS 쓰기 연산

- (1) 클라이언트는 네임노드에 접속하여 원하는 파일이 저장된 블록위치를 조회
- (2) 해당 블록이 저장된 데이터 노드에서 직접 데이터를 조회

