Hadoop Tutorial

2013.2

정재화

GRUTER

ABOUT ME

- 현)그루터 책임 개발자 (http://www.gruter.com)
- 전)큐릭스, NHN, 엔씨소프트
- E-mail: jhjung@gruter.com
- Homepage: http://blrunner.com
- Twitter: @blrunner78
- 저서: 시작하세요!하둡 프로그래밍: 기초부터 실무까지 하둡의 모든 것 (2012.10 출간)

GRUTER

목차

1. Hadoop

1.1 하둡 개요

- **하둡은 대용량 데이터를 분산 처리할 수 있는 자바 기반의 오픈소스 프레임워크이다.**

① Distributed File System
  - 분산/병렬처리에 적합한 구조로 파일 저장

② Distributed/Parallel Computing framework
  - Map&Reduce 기반의 컴퓨팅 플랫폼 제공

③ Open Source Project
  - http://hadoop.apache.org
  - Java 기반
  - 4,000 node 이상 단일 클러스터 구성 가능

1.2 하둡 히스토리

- Feb 2003 First MapReduce library by Google
- Dec 2004 Google GFS file system paper
- **Feb 2006 Hadoop becomes Apache Lucene project**
- Apr 2007 Yahoo! runs Hadoop on 1000-node cluster
- Feb 2008 Yahoo! generate production search index with Hadoop
- July 2008 Hadoop Wins Terabyte Sort Benchmark
  - sorted 1 terabyte of data in 209 seconds
- July 2009 Hadoop is getting bigger
  - Hadoop Common, MapReduce, Hadoop Distributed File System(HDFS)
- **Dec 2011 Release 1.0.0**
- May 2012 Release 2.0.0 alpha

1.3 하둡 에코 시스템

프로젝트	설명
Avro	멀티 플랫폼간 데이터 호환 Serialization 도구
Cassandra	DHT기반의 분산 데이터 관리 시스템. Hadoop은 사용하지 않고 로컬 디스크 이용
Chukwa	분산 환경에서 로그를 수집하기 위한 시스템, 저장소로 HDFS를 이용하고 로그 분석을 위해 MapReduce 이용
Hama	Map/Reduce 방식이 아닌 BSP(Bulk Synchronous Parallel) 방식의 컴퓨팅 플랫폼
HBase	HDFS에 데이터 파일을 저장하는 분산 데이터 관리 시스템
Hive	SQL과 비슷한 스크립트 질의를 이용해 HDFS에 저장된 데이터를 MapReduce로 분석하는 도구
Impala	Hive 질의 문법을 지원하는 준 실시간 질의 실행 플랫폼
Mahout	Hadoop 기반 Machine library
Oozie	Hadoop Workflow 엔진
Pig	Hive와 유사하게 스크립트 질의를 이용해 HDFS에 저장된 데이터를 맵리듀스로 분석하는 도구 단순 스크립트가 아닌 반복문, 제어문, 변수 등 사용 가능
ZooKeeper	분산 환경을 관리하는 분산 코디네이터

2. HDFS

2.1 대용량 파일 시스템

파일 시스템	특징
DAS (Direct-attached storage)	- 서버에 직접 연결된 스토리지(storage) - 여러 개의 하드디스크를 장착할 수 있는 외장 케이스를 이용하는 방식
NAS (Network-attached storage)	- 일종의 파일 서버 - 별도의 운영체제를 사용하며, 파일 시스템을 안정적으로 공유할 수 있음. - 주로 첨부 파일이나 이미지 같은 데이터를 저장하는 데 많이 사용
SAN (storage area network)	- 수십에서 수백 대의 SAN 스토리지를 데이터 서버에 연결해 총괄적으로 관리해주는 네트워크를 의미함 - DAS의 단점을 극복하기 위해 개발됐으며, 현재 SAN 기법이 시장의 절반 이상을 차지 - DBMS와 같이 안정적이고 빠른 접근이 필요한 데이터를 저장하는 데 사용

2.2 GFS (Google File System) 개요

① Google에서 개발된 파일 시스템으로 Google의 많은 서비스에서 사용

② 소프트웨어는 공개하지 않고 논문만 공개
- http://www.cs.rochester.edu/meetings/sosp2003/papers/p125-ghemawat.pdf

③ 다음과 같은 설계 원칙
- 저가형 서버로 구성된 환경으로 서버의 고장이 빈번히 발생할 수 있다고 가정
- 대부분의 파일은 대용량 파일로 가정
- 작업 부하는 연속적으로 많은 데이터를 읽는 연산이거나 임의의 영역에서 적은 데이터를 읽는 연산
- 파일에 대한 쓰기 연산은 주로 순차적으로 데이터를 추가하는 연산, 파일에 대한 수정은 드물게 발생
- 여러 클라이언트에서 동시에 동일한 파일에 데이터를 추가하는 환경에서 동기화 오버헤드를 최소화할 수 있는 방법 필요
- 낮은 응답 지연 시간보다 높은 처리율이 좀 더 중요

2.3 HDFS(Hadoop Distributed File System) 개요

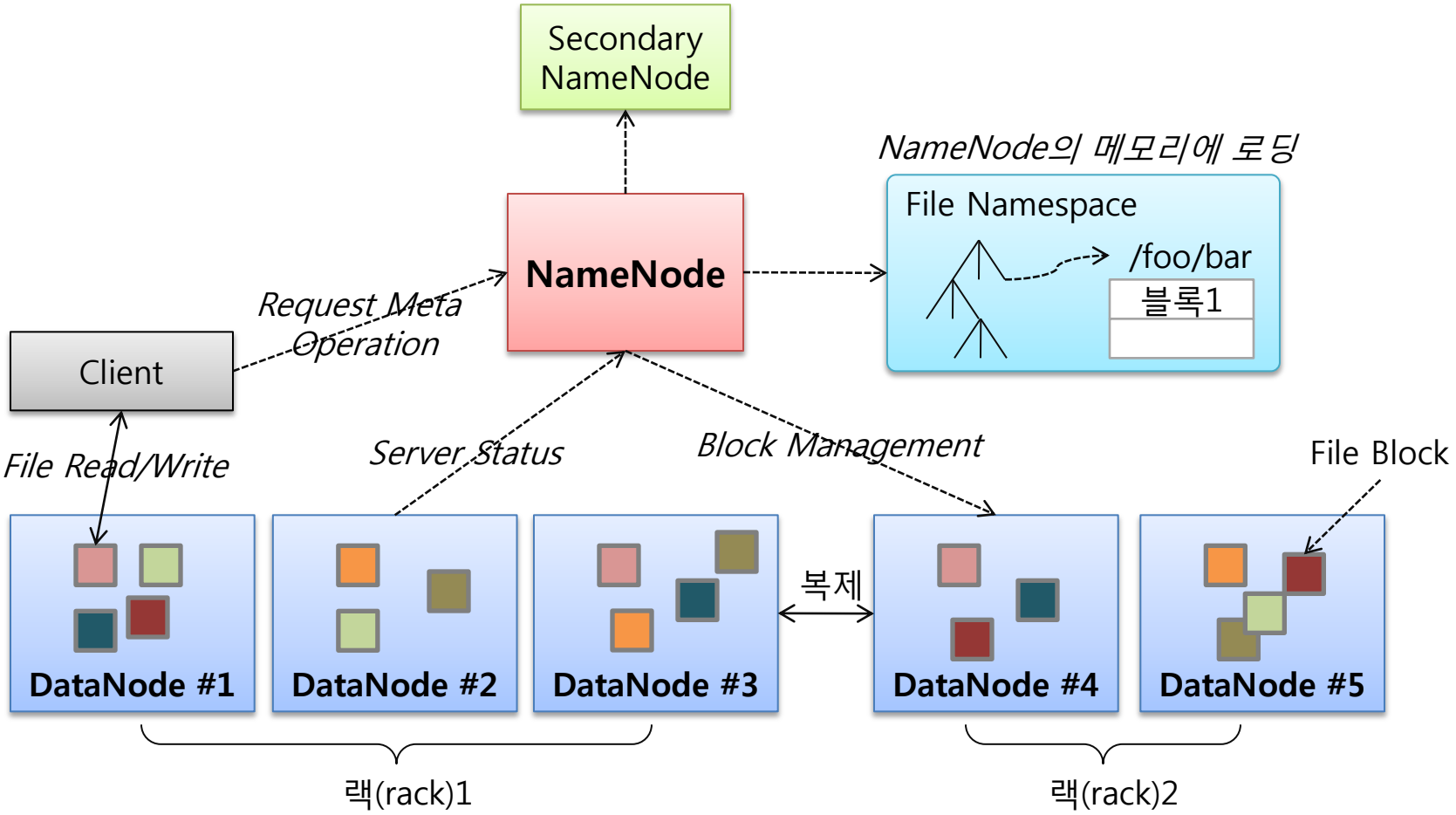
① Very Large Scale Distributed File System
- 10K nodes, 100 million files, 10 PB

② Use Commodity Hardware
- self-healing: failover, recovery, backup
- 서버 장애를 일반적인 상황이라고 가정

③ Optimized for batch processing
- 주로 저장 후 읽기 위주의 데이터 저장
- 1 file = n개의 64MB size block으로 split
- 각 block은 서로 다른 node에 분산 저장

④ POSIX 표준 API는 지원하지 않음
- 자체 API 지원(Java, C)
- File lock, Random write 미 지원
- append는 제한된 기능으로 제공
- 일반적인 응용 애플리케이션(DB 등)의 저장소로 활용 불가

2.4 HDFS 시스템 구성



2.5 HDFS 컴포넌트

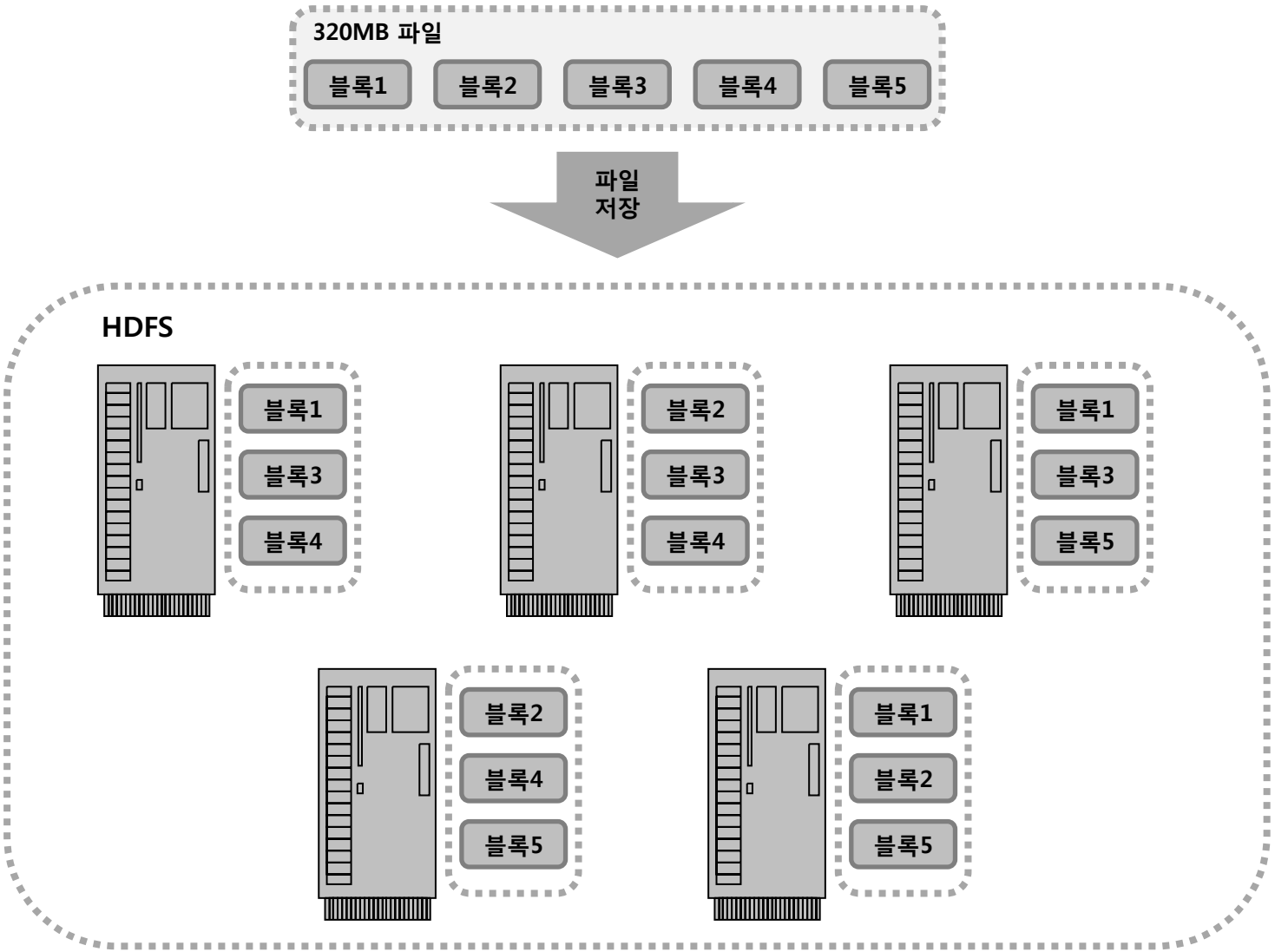
① NameNode
- 파일 시스템의 inode namespace 관리(메모리)
- DataNode 클러스터 멤버쉽 관리 및 장애 시 장애 복구
- File Block에 대한 복제 수, 볼륨 밸런싱 등에 대한 제어

② Secondary NameNode
- NameNode의 namespace 정보를 주기적(1시간)으로 rolling 하여 다시 NameNode로 전달하는 기능

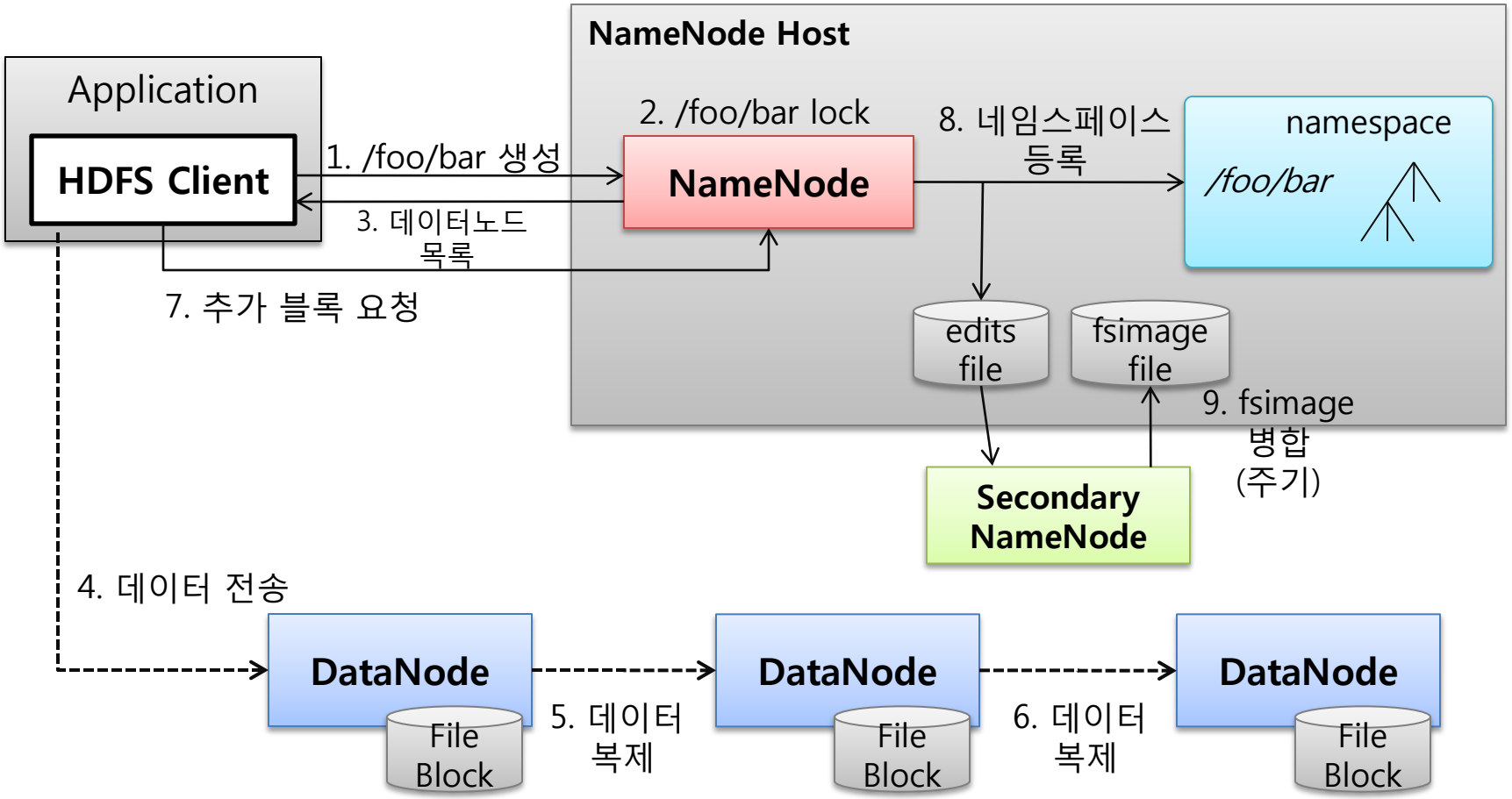
③ DataNode
- 실제 File Block을 저장
- File Block은 운영체제(Linux)의 파일시스템내에 하나의 파일로 저장

④ Client Library
- 파일 연산을 위한 API 제공
- 파일 저장 시 NameNode, DataNode와 연동

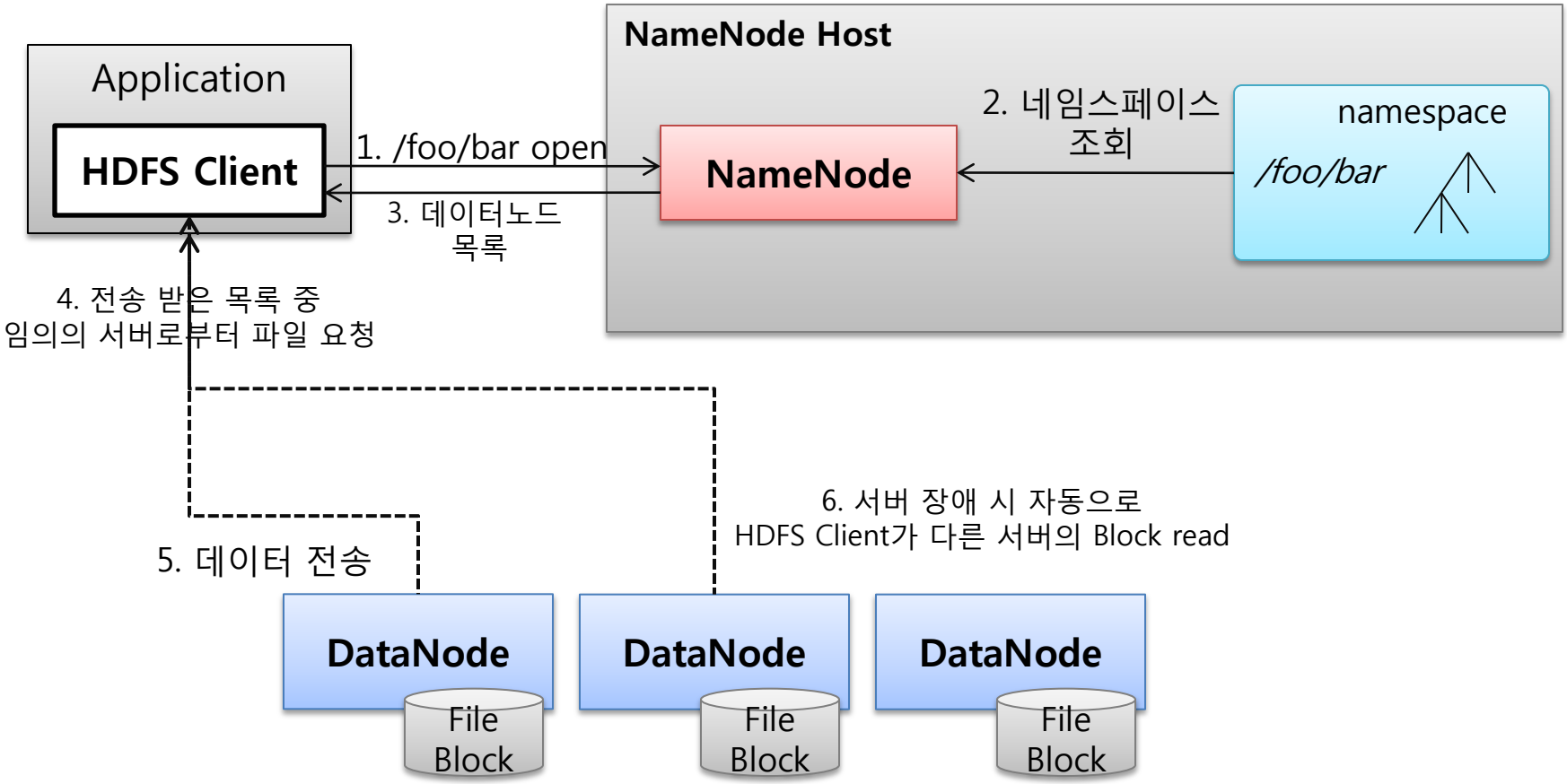
2.6 블록 기반 구조


320MB 파일
블록1
블록2
블록3
블록4
블록5
파일
저장
HDFS
블록1
블록3
블록4
블록2
블록3
블록4
블록1
블록3
블록5
블록2
블록4
블록5
블록1
블록2
블록5

2.7 파일 저장 과정


NameNode Host
Application
HDFS Client
1. /foo/bar 생성
2. /foo/bar lock
NameNode
3. 데이터노드 목록
7. 추가 블록 요청
8. 네임스페이스 등록
namespace
/foo/bar
edits file
fsimage file
9. fsimage 병합 (주기)
Secondary NameNode
4. 데이터 전송
DataNode
File Block
5. 데이터 복제
DataNode
File Block
6. 데이터 복제
DataNode
File Block

2.8 파일 조회 과정


NameNode Host
Application
HDFS Client
1. /foo/bar open
3. 데이터노드 목록
NameNode
2. 네임스페이스 조회
namespace
/foo/bar
4. 전송 받은 목록 중 임의의 서버로부터 파일 요청
5. 데이터 전송
6. 서버 장애 시 자동으로 HDFS Client가 다른 서버의 Block read
DataNode
File Block
DataNode
File Block
DataNode
File Block

2.9 HDFS 특징

① 하나의 파일을 여러 개의 block으로 분리하여 분산 저장
② 별도의 외부 스토리지가 아닌 x86 장비 내부의 로컬 디스크 이용
③ 특정 서버 장애 발생 시 자동 감지 및 복구
④ 서버 추가/제거 시 별도의 작업 불필요
⑤ 저장 가능한 파일 수 한계
⑥ NameNode가 SPOF(Single Point Of Failure)
⑦ 범용 스토리지로 사용하기는 기능적 제약이 있음

3. MapReduce

3.1 대용량 데이터 처리의 어려움

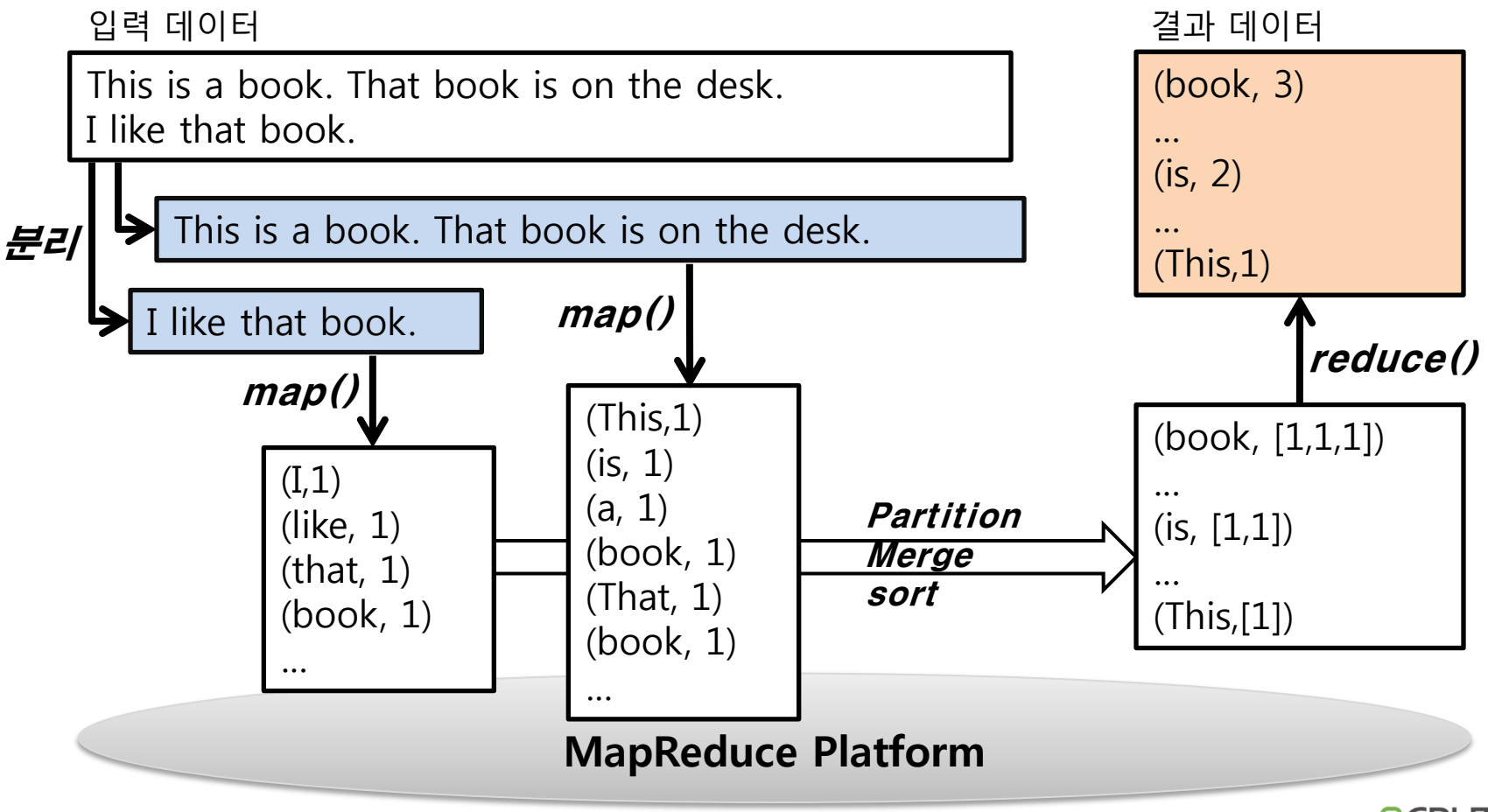
① 1대 장비에서의 자원의 제약
- CPU 갯수, 메모리 용량
- Scale Up 방식의 확장은 비용 증가 및 최대 확장 가능 자원 제약

② 데이터 읽기 속도
- 디스크 읽기 속도가 100MB/sec 라고 가정하면
- 1TB = 10,485 sec = 7일 소요

③ 성능을 높이기 위해서는 분산 처리가 필수
- 프로그램 복잡도 증가
- 컴퓨팅 도중 일부 장비, 네트워크 장애 발생 가능성이 높아짐
- 대부분의 분산 컴퓨팅 플랫폼은 컴퓨팅만 분산

3.2 MapReduce 개념

- map (k1,v1) → list(k2,v2)
- reduce (k2, list (v2)) → list(v2)



3.3 Google MapReduce

① MapReduce 개념을 대규모 분산 환경에 구현
② 논문으로만 외부에 공개
  - 2004년
  - http://research.google.com/archive/mapreduce.html

3.4 Hadoop MapReduce

① MapReduce를 쉽게 구현, 실행할 수 있는 프레임워크
- MapReduce 관련 라이브러리
- MapReduce 프로그램 실행 환경 제공

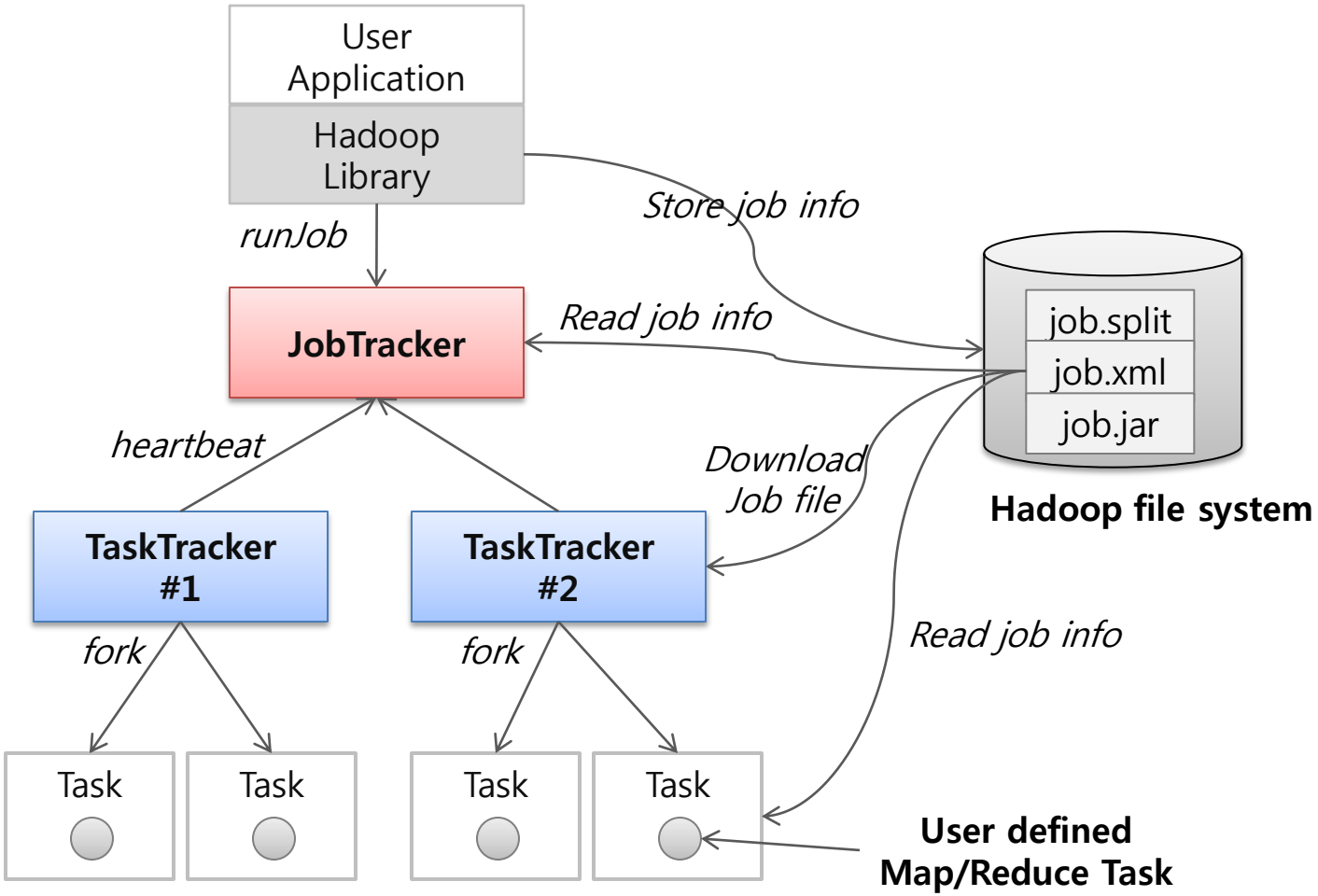
② HDFS와 연동
- 입력/출력 데이터 파일 저장소로 사용
- 작업 수행 시 데이터 파일을 네트워크를 거쳐 읽는 것이 아니라 입력 블록이 있는 서버에서 수행되도록 스케줄링
- 프로그램 수행에 필요한 설정 파일, 사용자 프로그램 등의 저장소로 활용

③ 스케줄러
- 기본은 FIFO, 추가로 Fair, Capacity 스케줄러 제공
- 작업 실행 도중 특정 태스크에 장애 또는 서버에 장애 발생 시 다른 서버로 재할당

④ 작업 단위
- Job: 사용자가 실행 시키는 작업
- Task: 하나의 작업은 n개의 Task(Map or Reduce)로 분리되어 각 노드에서 분산되어 실행

⑤ 다양한 실행/프로그램 옵션 제공
- MapReduce 프로그램은 기본은 자바로 개발해야 하지만 다양한 프로그램 언어로 구현 가능
- 인터페이스 기반으로 다양한 사용자 정의 기능 구현 가능

3.5 Hadoop MapReduce 아키텍처



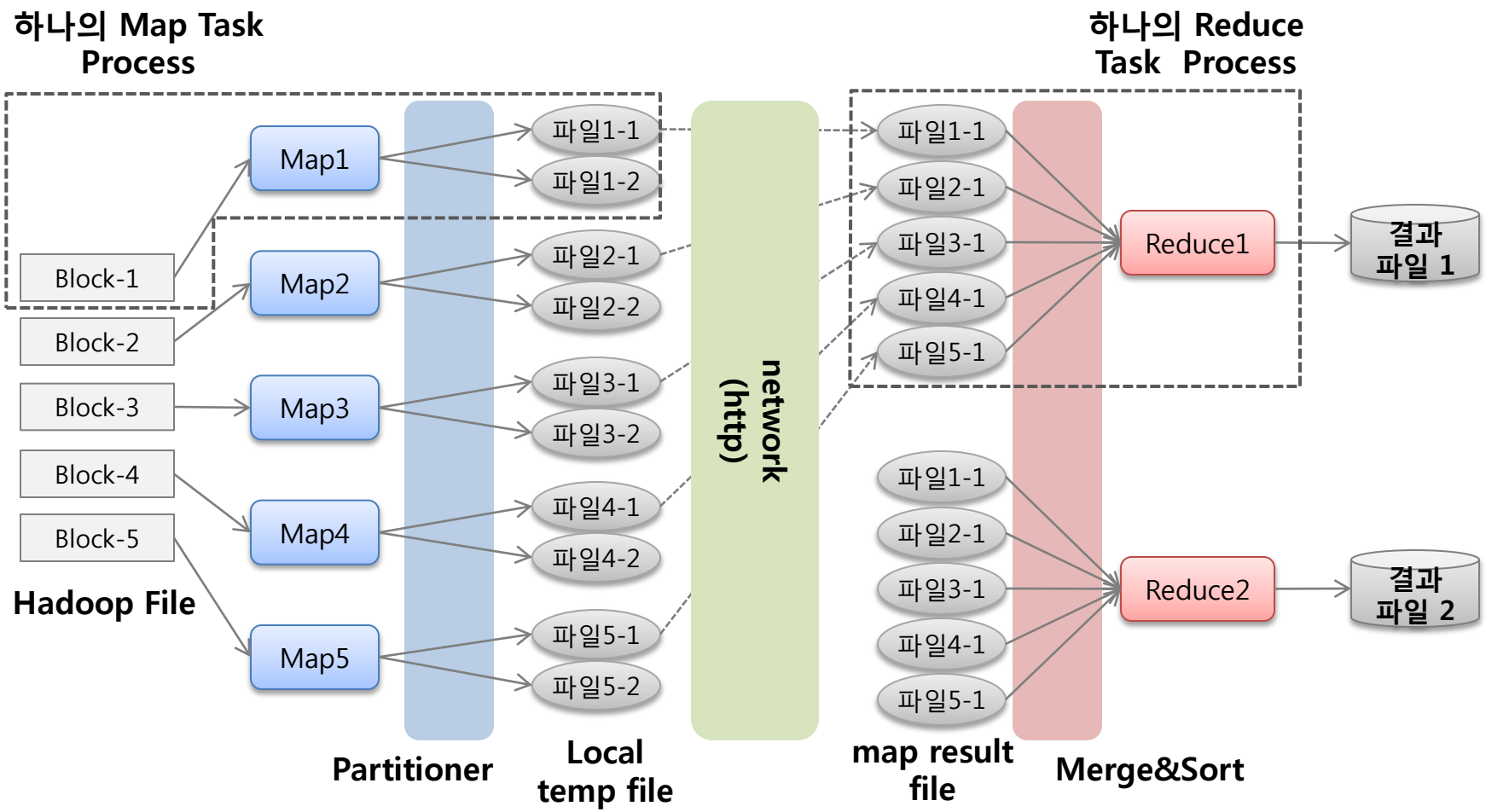
3.6 MapReduce 시스템 콤포넌트

① JobTracker
- 작업을 관리하는 마스터 서버
- 주로 NameNode와 동일한 서버 사용
- 사용자의 작업은 JobTracker로 요청됨
- 작업에 대한 스케줄링 관리
- Single Point Of Failure

② TaskTracker
- Task가 수행하는 서버
- 주로 DataNode와 동일한 서버 사용
- 하나의 TaskTracker 서버에는 동시에 N 개의 Task 수행 가능

③ ClientAPI
- Hadoop의 MapReduce 프로그램 모델 및 프레임워크
- 클러스터 상황에 대한 간단한 API

3.7 데이터 흐름


하나의 Map Task Process
하나의 Reduce Task Process
Block-1
Block-2
Block-3
Block-4
Block-5
Hadoop File
Map1
Map2
Map3
Map4
Map5
파일1-1
파일1-2
파일2-1
파일2-2
파일3-1
파일3-2
파일4-1
파일4-2
파일5-1
파일5-2
Partitioner
Local temp file
network (http)
파일1-1
파일2-1
파일3-1
파일4-1
파일5-1
파일1-1
파일2-1
파일3-1
파일4-1
파일5-1
map result file
Merge&Sort
Reduce1
Reduce2
결과 파일 1
결과 파일 2

4. Hive

4.1 Hive 개요

- 배경
  - ✓ Facebook이 만들기 시작했으며 현재 Top level ASF project
  - ✓ Facebook 은 2008년 0.2T(per day) 에서 2010년 12+TB(per day)으로 데이터가 폭발적으로 증가
  - ✓ Hive 질의를 수행하는 cron job 으로 Oracle DB 에 저장
  - ✓ Python code 로 ETL 처리

- MapReduce
  - ✓ MapReduce 프로그램의 어려움
  - ✓ Pig/Cascading
  - ✓ Metadata 관리문제(Hcatalog 는 2012년 incubator)

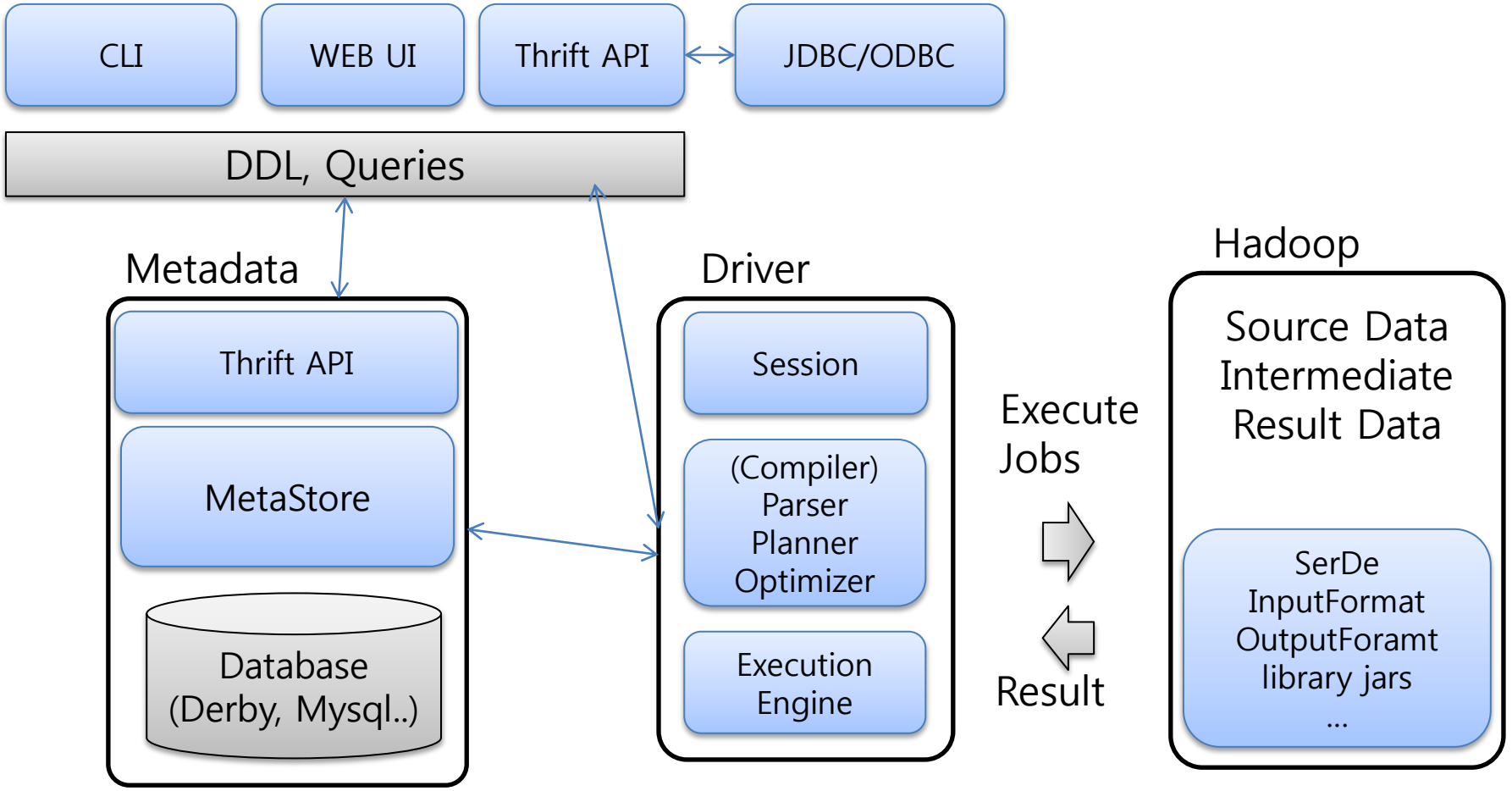
4.1 Hive 개요

- Hadoop 기반의 Data warehouse system
  - Hive 는 데이터를 구조화하고 복잡한 MR 프로그램을 HQL을 사용하여 쉽게 분석할 수 있다.
    - ✓ SQL-Like Query Language
    - ✓ HDFS 를 스토리지로 사용
    - ✓ MapReduce 사용으로 분산처리
    - ✓ RDB 를 이용한 Metadata 관리
    - ✓ 확장성(format, function, scripts..)
  - Hive 는 Hadoop 기반에서 MapReduce를 실행하기 때문에 batch 성 데이터를 처리하기에 적합
    - ✓ 작고 간단한 테이블 조회 시에도 많은 단계를 거치기 때문에 온라인 트랜잭션 작업이나 row 단위 업데이트는 불가능.

4.2 Hive 컴포넌트

① Driver: session, fetch, execute
② Compiler: parse, plan, optimize
③ Metastore: schema, location in HDFS, serializers and deserializers
④ Execution engine: Compiler에서 생성된 plan 을 실행하며, stages간의 dependency 관리
⑤ 관리도구
  - Interactive console, hadoop streaming 형태로 실행됨
  - 간단한 WEB UI 제공

4.3 Hive 아키텍처



4.4 데이터 모델 : 테이블

① Location
- 테이블을 만들면 DFS 의 /user/hive/warehouse/{db}/{tablename} 으로 directory 가 생성된다.

  e.g) CREATE TABLE invites (foo INT, bar STRING);
- External Table : 다른 위치에 있는 data로 생성가능, drop 시 지워지지 않는다.

  e.g) CREATE EXTERNAL TABLE invites (foo INT, bar STRING) location '/tmp/hive/data'

② Delimiter
- Column 과 row를 구분하기 위해 사용한다. 그러나 row 는 hadoop 이 결정하여 변경할 수 없다.

③ Data Types
- Primitive, complex 등을 지원한다.
- https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Types

④ File format: SEQUENCEFILE, TEXTFILE, RCFILE, INPUTFORMAT/OUTPUTFORMAT

4.5 HiveQL: 데이터 로드

① Loading files into tables

- LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO TABLE tablename [PARTITION (partcol1=val1, partcol2=val2 ...)]

  e.g) CREATE TABLE invites (foo INT, bar STRING);

  e.g) LOAD DATA LOCAL INPATH 'examples/files/kv1.txt' OVERWRITE INTO TAB LE invites;

② Inserting data into Hive Tables from queries(multiple inserts)

- INSERT OVERWRITE | INTO TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...) [IF NOT EXISTS]] select_statement1 FROM from_statement;

  e.g) CREATE TABLE invites1 like invites;

  e.g) INSERT OVERWRITE TABLE invites1 select * from invites;

③ Writing data into filesystem from queries(multiple inserts)

- INSERT OVERWRITE [LOCAL] DIRECTORY directory1 SELECT ... FROM ...

4.6 HiveQL: 데이터 조회

① Select Syntax
e.g) SELECT * FROM t1
② WHERE Clause
e.g) SELECT * FROM sales WHERE amount > 10 AND region = "US"
③ DISTINCT Clauses
e.g) SELECT DISTINCT col1 FROM t1
④ Partition Based Queries
⑤ HAVING Clause
e.g) SELECT col1 FROM t1 GROUP BY col1 HAVING SUM(col2) > 10
⑥ LIMIT Clause
e.g) SET mapred.reduce.tasks = 1
e.g) SELECT * FROM sales SORT BY amount DESC LIMIT 5 //Top 5
⑦ REGEX Column Specification
e.g) SELECT `(ds|hr)?+.+` FROM sales

c.f) https://cwiki.apache.org/confluence/display/Hive/LanguageManual

5. Demo

5.1 Hadoop 실행 모드

모드	내용
Standalone	- 하둡의 기본 실행 모드. - 하둡 환경 설정 파일에 아무런 설정을 하지 않고, 실행을 하게 되면 로컬 장비에서만 실행되기 때문에 로컬 모드라고도 함 - 하둡에서 제공하는 데몬을 구동하지 않기 때문에 분산된 환경을 고려한 테스트는 불가능. - 단순하게 맵리듀스 프로그램을 개발하고, 해당 맵리듀스를 디버깅하는 용도에만 적합한 모드.
Pseudo-distributed	- 가상 분산모드 - 하나의 장비에 모든 하둡 환경 설정을 하고, 하둡 서비스도 이 장비에서만 제공하는 방식 - HDFS와 맵리듀스와 관련된 데몬을 하나의 장비에서만 실행함 - 하둡 스터디 및 개발 테스트 용도에 적합
Fully distributed	- 완전 분산 모드 - 여러 대의 장비에 하둡이 설치된 경우

5.1 Hadoop 실행 모드

호스트	설치 데몬
server01	NameNode, JobTracker
server02	Secondary NameNode, DataNode, TaskTracker
server03	DataNode, TaskTracker
server04	DataNode, TaskTracker

5.2 설치 환경 구성

① 리눅스 서버 준비
- 윈도우 계열의 경우 VirtualBox, VMWare와 같은 가상 호스트에 리눅스를 설치 할 것

② 호스트 설정
- /etc/hosts 에 하둡 설치 대상 호스트 등록
- e.g) 192.168.56.101 server01

③ JDK 설치
- 다운로드: http://www.oracle.com/technetwork/java/javase/downloads/jdk6u38-downloads-1877406.html
- su - root
- cd ${download_dir}
- chmod 755 jdk-6u38-linux-i586-rpm.bin
- ./jdk-6u38-linux-i586-rpm.bin
- /etc/profile에 JAVA_HOME 추가
  - export JAVA_HOME=JDK설치 경로
  - export PATH=$PATH:$JAVA_HOME/bin
- source /etc/profile

5.2 설치 환경 구성

④ 사용자 생성
- su – root
- adduser hadoop
- passwd hadoop

⑤ ssh 키 생성 및 복사
- su – hadoop
- ssh-keygen –t rsa
- ssh-copy-id –i ~/.ssh/id_rsa.pub 데이터 노드 호스트명
- ssh-copy-id –i ~/.ssh/id_rsa.pub 보조 네임 노드 호스트명
- 모든 Slave서버로 복사할 것

5.3 설치 파일 다운로드

stable 버전으로 다운로드 (2013년2월 현재 stable 버전은 1.0.4 버전임)

5.4 Hadoop 설치

① 하둡 설치 파일을/home/hadoop에 업로드

② 압축 해제

- su - hadoop
- cd /home/hadoop
- tar xzf hadoop-1.0.4.tar.gz

③ 디렉터리 생성

- namespace 저장 디렉토리(namenode)
  - ✓ mkdir /home/hadoop/filesystem
- data 디렉토리(datanode)
  - ✓ mkdir /home/hadoop/data

④ conf/hadoop-env.sh

- export HADOOP_HEAPSIZE=256
- export HADOOP_PID_DIR=/home/hadoop/pids

⑤ conf/core-site.xml

```
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://server01:9000</value>
  </property>
</configuration>
```

5.4 Hadoop 설치

⑥ conf/hdfs-site.xml

```
<property>
  <name>dfs.name.dir</name>
  <value>/home/hadoop/filesystem</value>
</property>
<property>
  <name>dfs.replication</name>
  <value>3</value>
</property>
<property>
  <name>dfs.http.address</name>
  <value>server01:50070</value>
</property>
<property>
  <name>dfs.secondary.http.address</name>
  <value>server02:50090</value>
</property>
```

```
<property>
  <name>dfs.data.dir</name>
  <value>/home/hadoop/data/hdfs</value>
</property>
<property>
  <name>dfs.support.append</name>
  <value>true</value>
</property>
```

5.4 Hadoop 설치

⑦ conf/mapred-site.xml

```
<configuration>
  <property>
    <name>mapred.job.tracker</name>
    <value>server01:9001</value>
  </property>
</configuration>
```

⑧ conf/masters

```
server02
```

⑨ conf/slaves

```
server02
server03
server04
```

5.5 Slave 서버 설치

① NameNode에 설치된 내용을 DataNode로 복사

② NameNode 설치 내용 tar
- cd /home/hadoop
- tar cf hadoop.tar hadoop-1.0.4

③ DataNode로 복사 및 압축 해제
- 모든 DataNode에 대해 내용 실행
- scp hadoop.tar <데이터노드 호스트명>:/home/hadoop
- ssh <데이터노드 호스트명> "cd /home/hadoop; tar xf hadoop.tar"
- ssh <데이터노드 호스트명> "mkdir /home/hadoop/data"
- 운영환경에서는 스크립트 작성하여 활용

5.6 NameNode 포맷

① FileSystem Namespace 정보 저장 디렉토리를 구성하는 과정
- NameNode에서만 실행
- Format 명령 실행 시 기존 파일 정보가 있으면 모두 삭제됨
- conf/hdfs.site.xml 파일에 있는 "dfs.name.dir" 설정 값에 생성

② 실행 명령
- bin/hadoop namenode –format
- ls –al /home/hadoop/filesystem

5.7 Hadoop 실행 및 중지

① 실행 관리는 NameNode가 설치된 서버에서 스크립트 실행
② Hadoop 전체 시작 및 중지
- cd ${HADOOP_HOME}
- bin/start-all.sh
- bin/stop-all.sh

③ HDFS만 시작 및 중지
- bin/start-dfs.sh
- bin/stop-dfs.sh

④ 특정 데몬 시작 및 중지
- NameNode
  - ✓ NameNode 설치 서버
  - ✓ bin/hadoop-daemon.sh start namenode
  - ✓ bin/hadoop-daemon.sh stop namenode
- DataNode
  - ✓ 실행 할 DataNode 서버에서 실행
  - ✓ bin/hadoop-daemon.sh start datanode
  - ✓ bin/hadoop-daemon.sh stop datanode

5.8 MapReduce 플랫폼 실행

① 전체 실행: JobTracker 서버(NameNode 서버)에서 실행
- su – hadoop
- cd /home/hadoop/hadoop-1.0.4
- bin/start-mapred.sh
- bin/stop-mapred.sh

② 데몬별 실행: 각 서버에서 실행
- JobTracker
  - ✓ bin/hadoop-daemon.sh start jobtracker
  - ✓ bin/hadoop-daemon.sh stop jobtracker
- TaskTracker
  - ✓ bin/hadoop-daemon.sh start tasktracker
  - ✓ bin/hadoop-daemon.sh stop tasktracker

5.9 예제 프로그램 실행

① 기본 제공되는 예제 프로그램
- hadoop-examples-1.x.x.jar

② HDFS에 샘플 데이터 업로드
- bin/hadoop fs –mkdir conf
- bin/hadoop fs -put conf/hadoop-env.sh conf/hadoop-env.sh

③ WordCount 예제 실행
- bin/hadoop jar hadoop-examples-*.jar wordcount conf/hadoop-env.sh wordcount_output

④ 예제 실행 결과 확인
- bin/hadoop fs -cat wordcount_output/part-r-00000

5.10 Hive 설치

① 요구사항
- Hadoop
- RDBMS
  - ✓ Default: derby
  - ✓ MySql Server, mysql-connector-java(GPL)
- Hive-0.9.0
- ANT-LIB (OPTION hive web interface)

② 압축 해제
- cp ${download}/hive-0.9.0.tar.gz /home/hadoop
- cd /home/hadoop
- tar xzf hive-0.9.0.tar.gz

③ 환경 설정
- cd hive-0.9.0
- cp conf/hive-env.sh.template conf/hive-env.sh
- vi conf/hive-env.sh
  - ✓ HADOOP_HOME=/home/hadoop/hadoop-1.0.4

5.10 Hive 설치

④ vi conf/hive-site.xml

```
<configuration>
<property>
  <name>hive.metastore.warehouse.dir</name>
  <value>/user/hive/warehouse</value>
</property>
</configuration>
```

⑤ HDFS에 데이터 저장소 디렉터리 생성

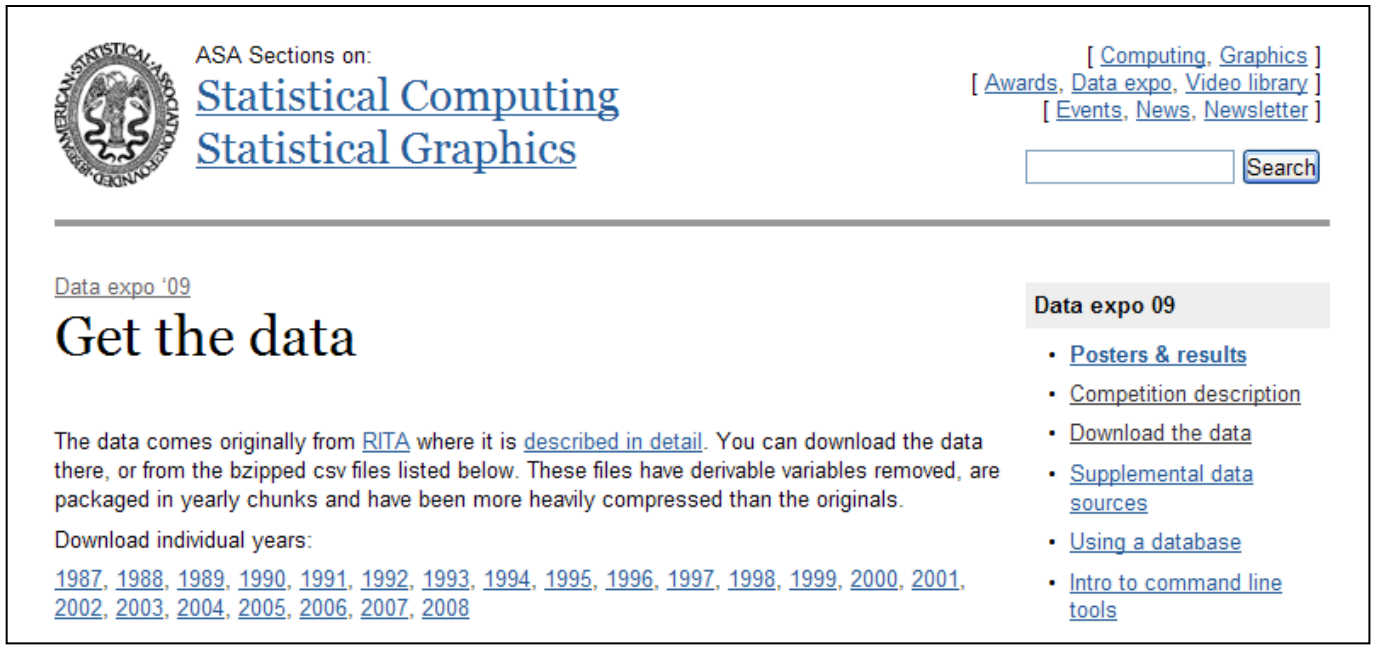
- cd /home/hadoop
- bin/hadoop fs –mkdir /tmp
- bin/hadoop fs -mkdir /user/hive/warehouse
- bin/hadoop fs -chmod g+w /tmp
- bin/hadoop fs -chmod g+w /user/hive/warehouse

⑥ 실행

- bin/hive
- show tables;

5.11 Hive 활용 사례

① 미국 항공운항 통계 데이터 다운로드: http://stat-computing.org/dataexpo/2009/the-data.html



② Hive 질의 시 편의를 위해 헤더 정보를 삭제함
- sed -e '1d' 2008.csv > 2008_new.csv

③ HDFS에 데이터 업로드
- bin/hadoop fs –mkdir input
- bin/hadoop fs -put 다운로드경로/2008_new.csv input

5.11 Hive 활용 사례

④ Hive 테이블 생성하기

```
CREATE TABLE airline_delay(Year INT, Month INT,
    DayofMonth INT, DayOfWeek INT,
    DepTime INT, CRSDepTime INT,
    ArrTime INT, CRSArrTime INT,
    UniqueCarrier STRING, FlightNum INT,
    TailNum STRING, ActualElapsedTime INT,
    CRSElapsedTime INT, AirTime INT,
    ArrDelay INT, DepDelay INT,
    Origin STRING, Dest STRING,
    Distance INT, TaxiIn INT,
    TaxiOut INT, Cancelled INT,
    CancellationCode STRING
    COMMENT 'A = carrier, B = weather, C = NAS, D = security',
    Diverted INT COMMENT '1 = yes, 0 = no',
    CarrierDelay STRING, WeatherDelay STRING,
    NASDelay STRING, SecurityDelay STRING,
    LateAircraftDelay STRING)
    PARTITIONED BY (delayYear INT)
    ROW FORMAT DELIMITED
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '₩n'
    STORED AS TEXTFILE;
```

5.11 Hive 활용 사례

⑤ Hive 테이블에 데이터 업로드

```
LOAD DATA INPATH '/user/hadoop/input/2008_new.csv'
OVERWRITE INTO TABLE airline_delay
PARTITION (delayYear='2008');
```

⑥ 데이터 업로드 확인

```
SELECT year, month, deptime, arrtime, uniquecarrier, flightnum
FROM airline_delay
WHERE delayYear = '2008'
LIMIT 10;
```

⑦ 집계 함수 활용

```
SELECT COUNT(1)
FROM airline_delay
WHERE delayYear = 2008;
```

```
SELECT Year, Month, AVG(ArrDelay) AS avg_arrive_delay_time, AVG(DepDelay) AS
avg_departure_delay_time
FROM airline_delay
WHERE delayYear = 2008
AND ArrDelay > 0
GROUP BY Year, Month;
```