제30회 Open Technet

빅데이터 오픈소스 플랫폼 기술세미나

하둡의 개요 및 생태계

빅데이터 커뮤니티
장형석
chjang1204@nate.com
http://www.bicdata.com (장교수)

1장 - MapReduce

2장 - 하둡분산파일시스템

3장 - 하둡생태계

MapReduce의 설계특성

- 분산컴퓨팅에 적합한 함수형 프로그래밍
- 배치형 데이터 처리 시스템
- 어플리케이션 로직의 주요관심사를 파악,
  많은 요소를 반영

MapReduce의 주요기능

- 자동화된 병렬처리 및 분산처리
- Fault-tolerance(내고장성,결함허용)
- 상태 및 모니터링 툴들
- 프로그래머를 위한 추상클래스

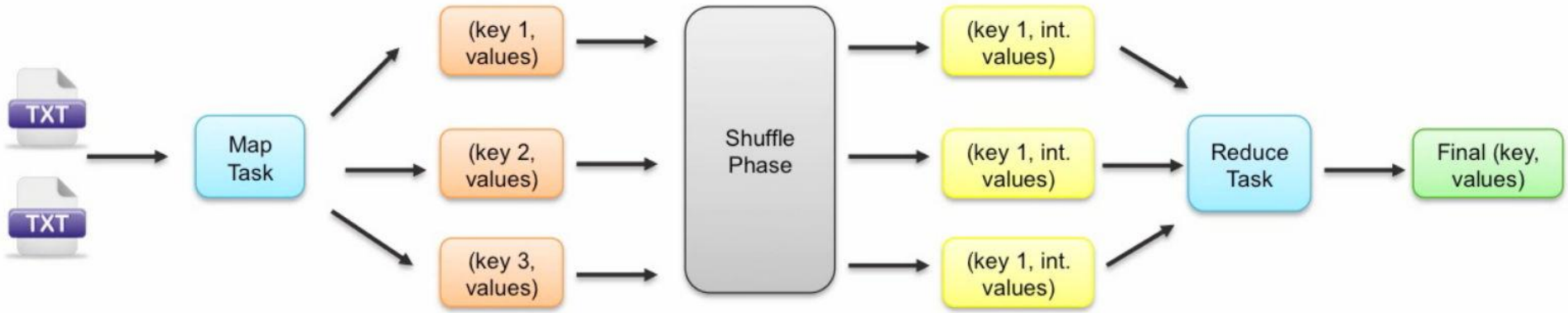
프로그래밍 모델

- 함수형 프로그래밍
- 2가지 함수의 사용자 인터페이스
  - map (in_key, in_value) ->
    (inter_key, inter_value) list
  - reduce (inter_key, inter_value list) ->
    (out_key,out_value) list

1. MapReduce

- 원본데이터(파일,DB레코드) 는
  map 함수에 의해서 <key, value> 쌍으로...

- map() 함수는 입력을 output key와 관련되는
  1..N개의 <key,value>를 만들어줍니다.



예제:Upper-case Mapper

- MapReduce 프레임워크의 map 단계에서 upper case 함수

```
let map( k,v ) =
  emit(k.toUpper(). v.toUpper() )
```

```
('foo', 'bar')      ->  ('FOO', 'BAR')
('Foo', 'other')    ->  ('FOO', 'OTHER')
('key2', 'data')    ->  ('KEY2', 'DATA')
```

예제:Explode Mapper

- 데이타셋을 key,value의 리스트로 변경하는 map() 메소드를 이용한 explode 함수 예시

```
let map( k,v ) =
  foreach char c in v : emit(k, c )

('A', 'cats')   ->  ('A', 'c'), ('A', 'a'),
                    ('A', 't'), ('A', 's')
```

예제:Filter Mapper

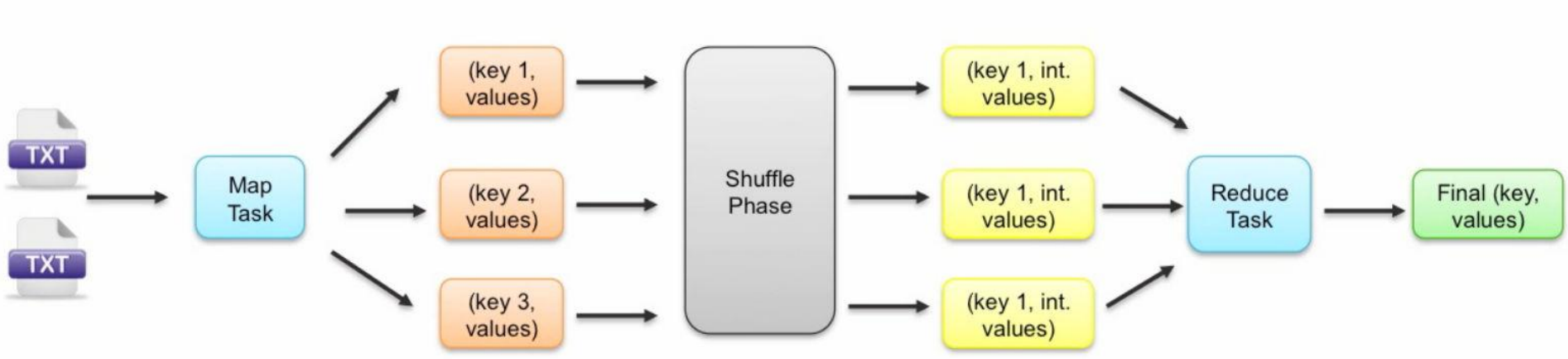
```
let map( k, v ) =
  if( isPrime(v) ) then emit( k,v )

('foo', 7)      ->  ('foo', 7)
('test', 10)    ->  (nothing)
```

- 간단한 데이터의 분석은 map 메소드를 이용해서 standalone 으로 MapReduce 프레임워크 사용가능
- 이제 Reduce 단계로 넘어갑니다.

Reduce

- Map 단계 다음에서 output key의 중간 value 들은 하나의 리스트로 합쳐져야 합니다.

- reduce()는 같은 output key를 가지는 final value로 중간 value 들을 합쳐줍니다.

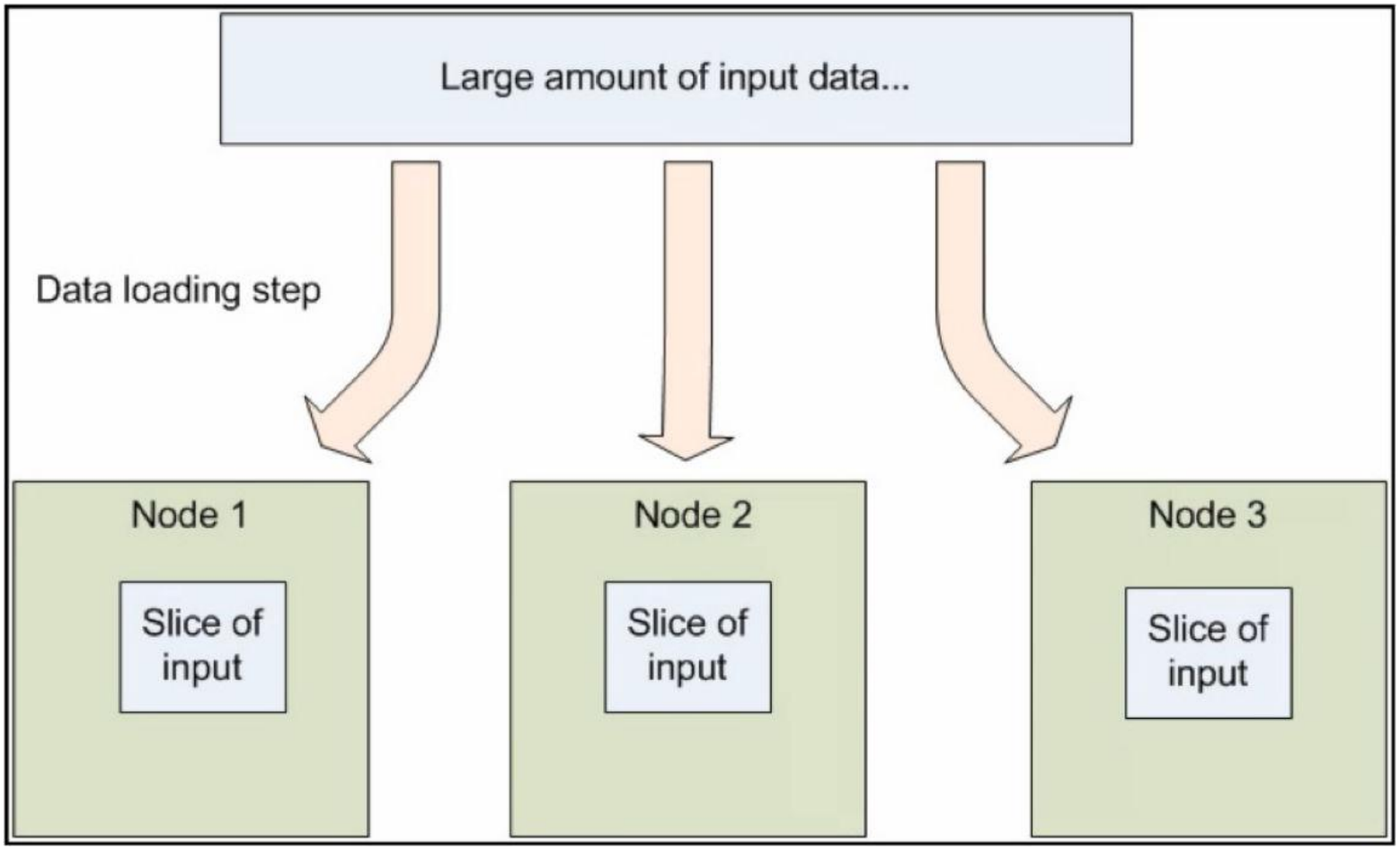


예제:Sum Reducer

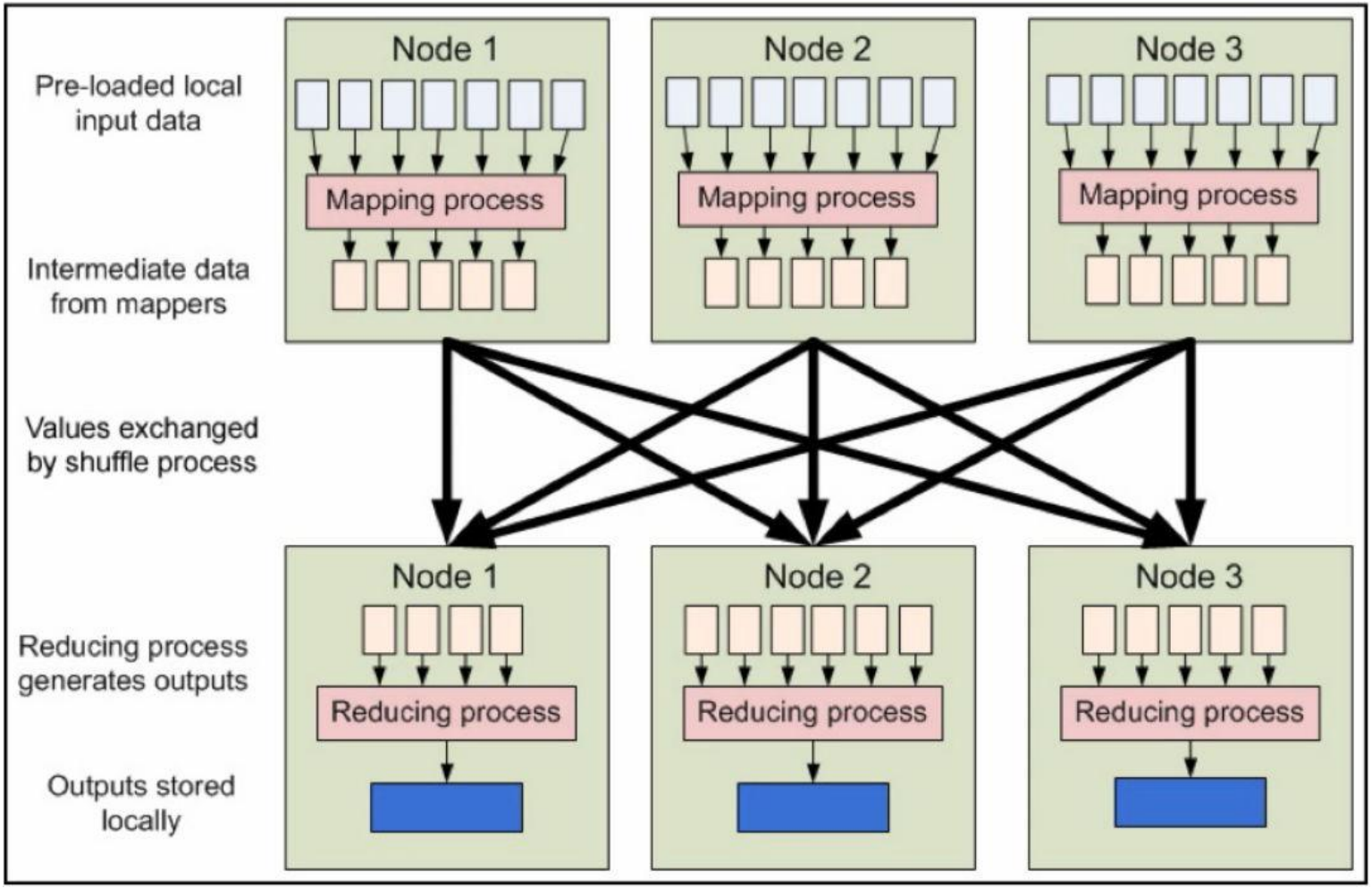
```
let reduce( k,vals ) =
  sum = 0
  foreach int v in vals :
      sum += v
  emit( k, sum )
```

```
('A', [42,100,312])   ->  ('A', 454)
('B', [12,6   ,-2])   ->  ('B', 16)
```

1. MapReduce


Large amount of input data...
Data loading step
Node 1
Slice of input
Node 2
Slice of input
Node 3
Slice of input

1. MapReduce


Pre-loaded local input data
Intermediate data from mappers
Values exchanged by shuffle process
Reducing process generates outputs
Outputs stored locally
Node 1
Mapping process
Node 2
Mapping process
Node 3
Mapping process
Node 1
Reducing process
Node 2
Reducing process
Node 3
Reducing process

MapReduce 요약

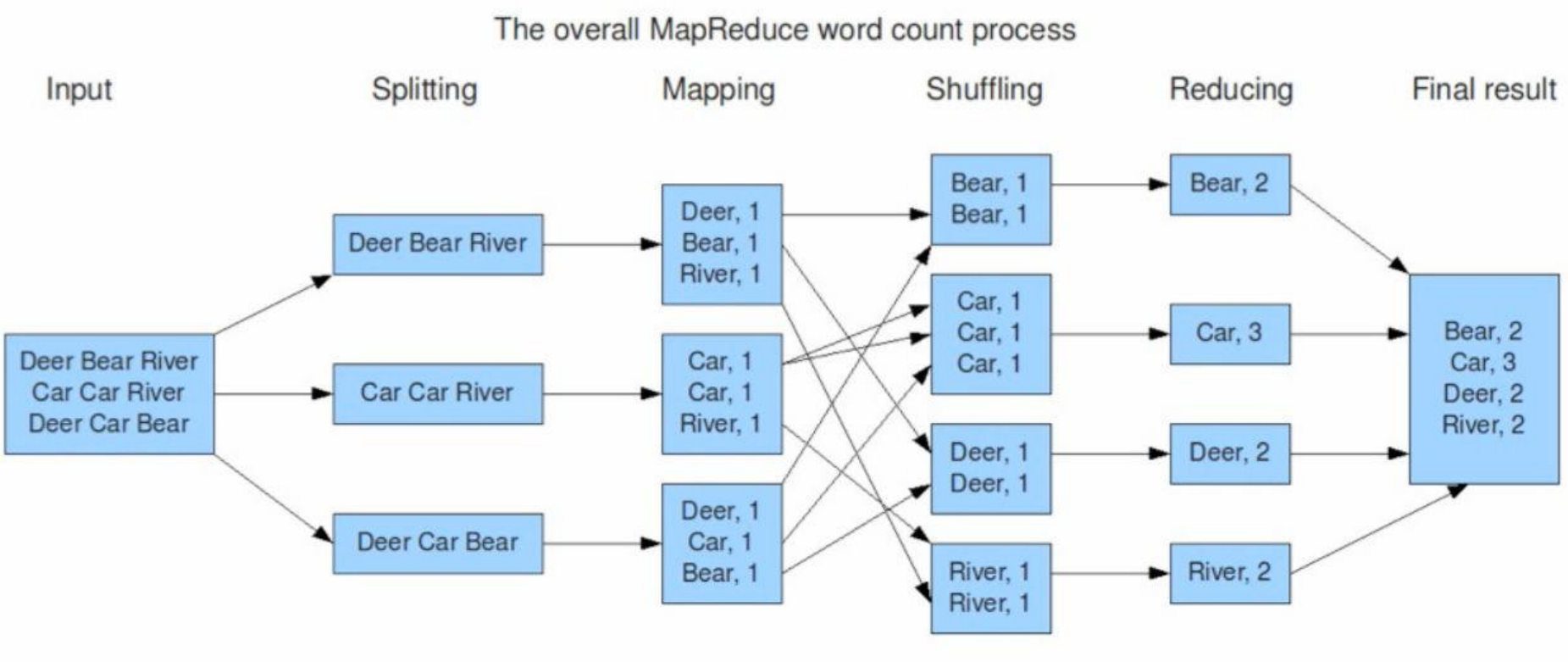
- map() 함수들은 병렬(parallel)로 작동합니다.
  여러 입력 자료셋으로 부터 여러 중간 value 들을 생성합니다.
- reduce() 함수들도 역시 병렬로 작동합니다.
  출력 key를 기준으로 각각 작업을 합니다.

- 모든 value 들은 독립적으로 처리됩니다.
- 병목 : reduce는 map 단계가 완료되지 않으면 시작할 수 없습니다.

1. MapReduce

```
map(String input_key, String input_value):
  // input_key: document name
  // input_value: document contents
  for each word w in input_value:
    emit(w, 1);

reduce(String output_key, Iterator<int> intermediate_values):
  // output_key: a word
  // output_values: a list of counts
  int result = 0;
  for each v in intermediate_values:
    result += v;
  emit(output_key, result);
```

1. MapReduce


The overall MapReduce word count process
Input
Splitting
Mapping
Shuffling
Reducing
Final result
Deer Bear River
Car Car River
Deer Car Bear
Deer Bear River
Car Car River
Deer Car Bear
Deer, 1
Bear, 1
River, 1
Car, 1
Car, 1
River, 1
Deer, 1
Car, 1
Bear, 1
Bear, 1
Bear, 1
Car, 1
Car, 1
Car, 1
Deer, 1
Deer, 1
River, 1
River, 1
Bear, 2
Car, 3
Deer, 2
River, 2
Bear, 2
Car, 3
Deer, 2
River, 2

Combining 단계

- map 단계후의 mapper 노드에서 실행됩니다.
- 로컬에서 map 출력(결과)은 "Mini-reduce"
- Reducer로 자료를 보내는
  대역폭을 줄일 수 있습니다.
- Reducer도 상호연동시에는 Combiner와 같습니다.

MapReduce 결론

- MapReduce는 많은 영역에서 개발자에서 유용한 추상화 기능을 제공합니다.
- 대용량의 계산을 아주 심플하게 만들어줍니다.
- MapReduce의 함수화된 프로그래밍 패러다임은 대용량의 어플리케이션에도 적용될 수 있습니다.
- 현실적인 문제에 초점을 맞출 경우, 세부적인 것은 라이브러리를 다룸으로써 해결될 수 있습니다.

하둡HDFS(하둡분산파일시스템)

HDFS - 하둡분산파일시스템

- 구글의 GFS 기반. 오픈소스
- 저가의 신뢰할 수 없는 컴퓨터에서 대용량의 자료를 저장하는 Redundant 스토리지
- 기존 파일 시스템과의 차이?
  - 기존과 다른 작업량, 설계방법
  - 기존 파일 시스템보다 큰 데이터처럼

HDFS가 필요한 경우

- 찾은 오류가 발생하는 구성품이 많은 경우
  - 저가의 일반적인 시스템에서의 오류

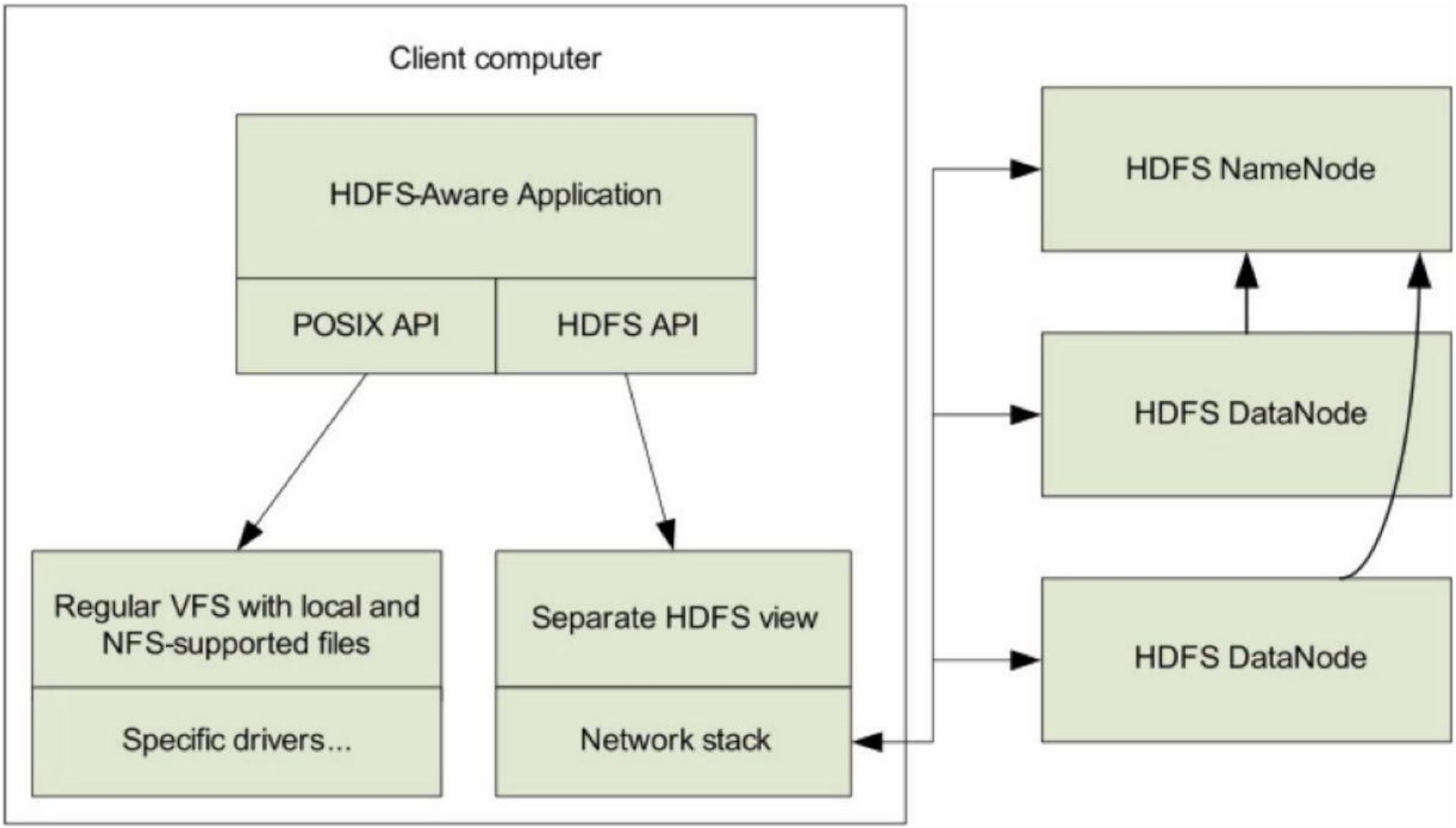
- HUGE 파일
  - 수백만
  - 각각 100MB 이상, 수GB

- 파일은 주로 쓰기와 추가가 대부분
- Full 검색
- 지연시간은 짧으나 많은 처리를 해야 하는 경우

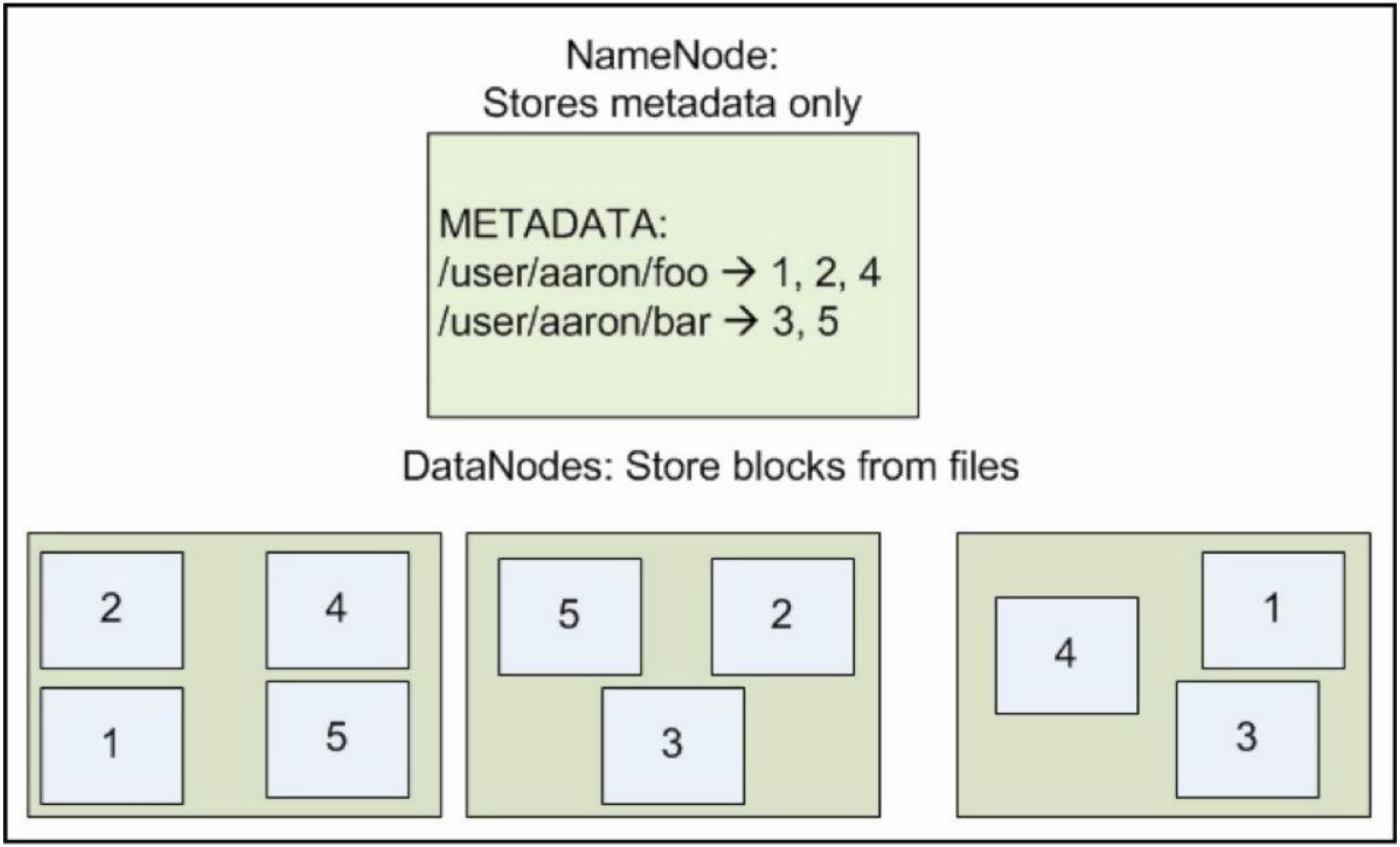
HDFS 개요

- 파일은 블럭단위로 저장
  - 일반적인 파일시스템보다 많은양 (default 64MB)
- Replication에 의한 신뢰성
  - 3개이상의 데이터노드에 복제됨
- 단일마스터(네임노드)에 의한 처리, 메타데이터
  - 간단한 중앙 관리 시스템
- No 데이터 캐싱
  - 많은 양의 풀스캔 데이터에는 유리
- 유사한 인터페이스, API 커스터마이징
  - 쉬운 문제해결 : 분산어플리케이션에 초점

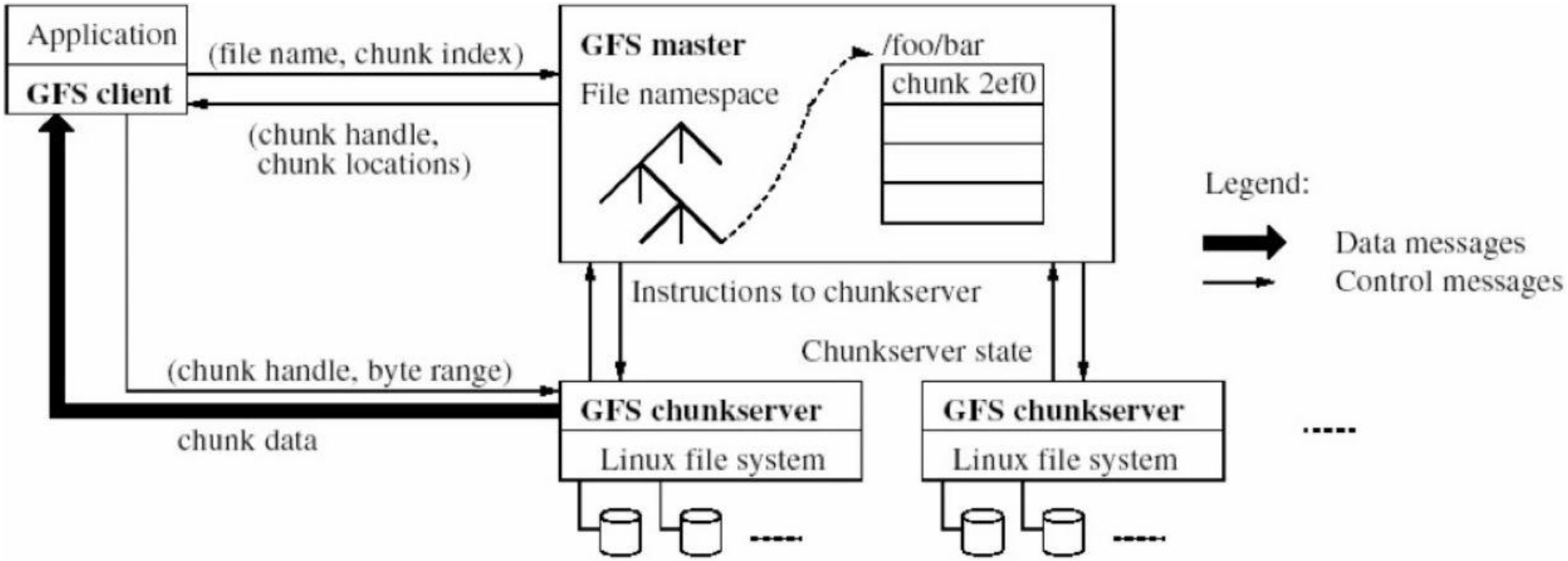
2. HDFS


Client computer
HDFS-Aware Application
POSIX API
HDFS API
Regular VFS with local and NFS-supported files
Specific drivers...
Separate HDFS view
Network stack
HDFS NameNode
HDFS DataNode
HDFS DataNode

2. HDFS


NameNode:
Stores metadata only
METADATA:
/user/aaron/foo → 1, 2, 4
/user/aaron/bar → 3, 5
DataNodes: Store blocks from files
2
4
1
5
5
2
3
4
1
3

2. HDFS


Application
GFS client
(file name, chunk index)
(chunk handle,
chunk locations)
GFS master
File namespace
/foo/bar
chunk 2ef0
Instructions to chunkserver
Chunkserver state
(chunk handle, byte range)
chunk data
GFS chunkserver
Linux file system
GFS chunkserver
Linux file system
Legend:
Data messages
Control messages

메타데이터

- 단일 네임노드는 모든 메타데이터를 저장
  - 파일명, 데이터노드에서의 위치
- 빠른 검색을 위해서 전체가 RAM에서 관리됨
- 데이터노드는 로컬파일시스템에 블록단위로 파일의 내용을 저장
  - 실제저장된 파일의 자료는 비직관적임

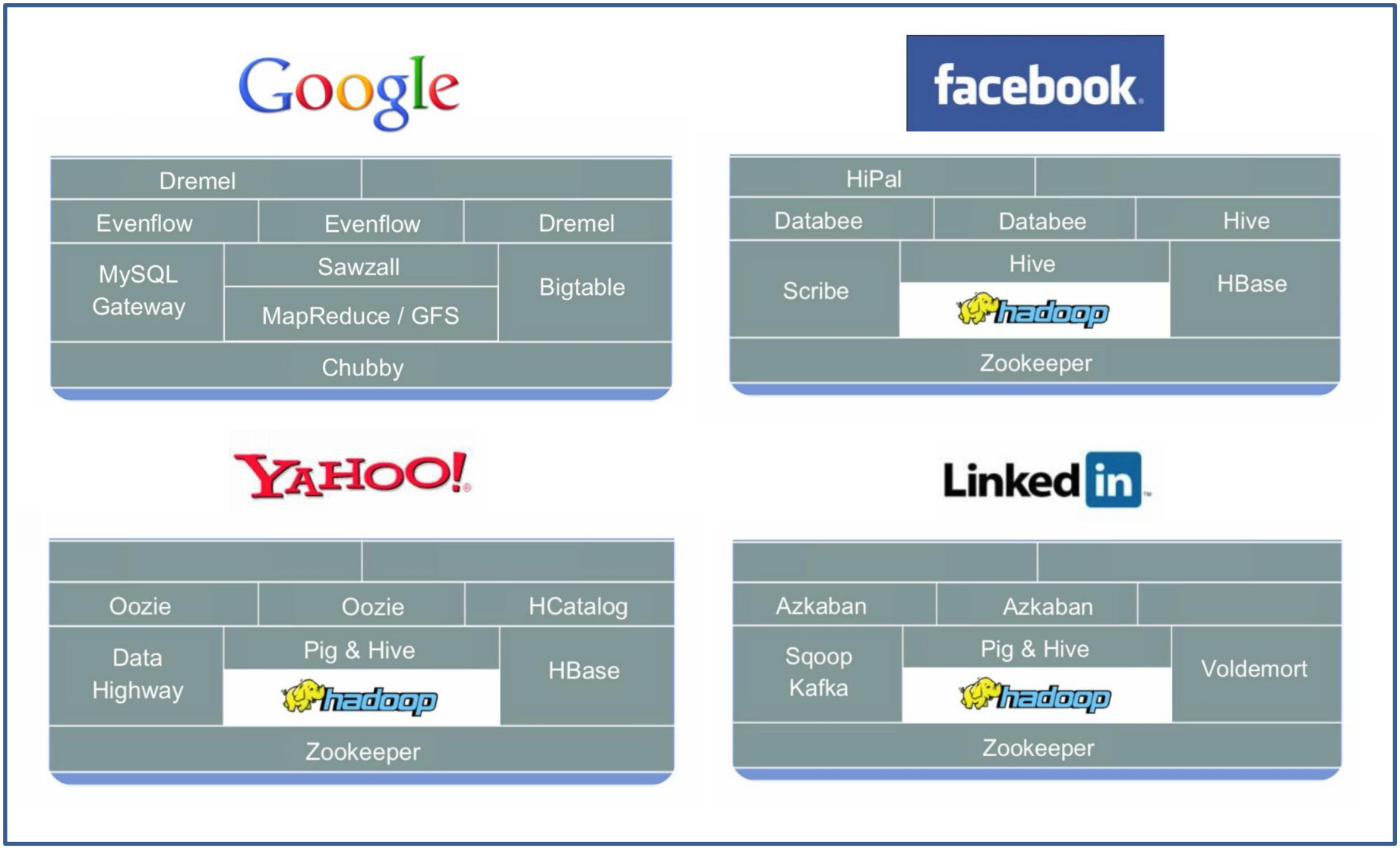
네임노드의 장애 처리

- 네임노드의 이미지와 편집로그는 디스크에
- 하둡은 여러곳에 파일을 저장할 수 있게 되어 있음
  - 로컬디스크 / NFS 마운트
- 백업네임노드는 네임노드이미지와 편집로그를 수집
  - 네임노드의 즉각적인 복구는 아님
  - 복구 상황에서 네임로드로써 임시 사용가능

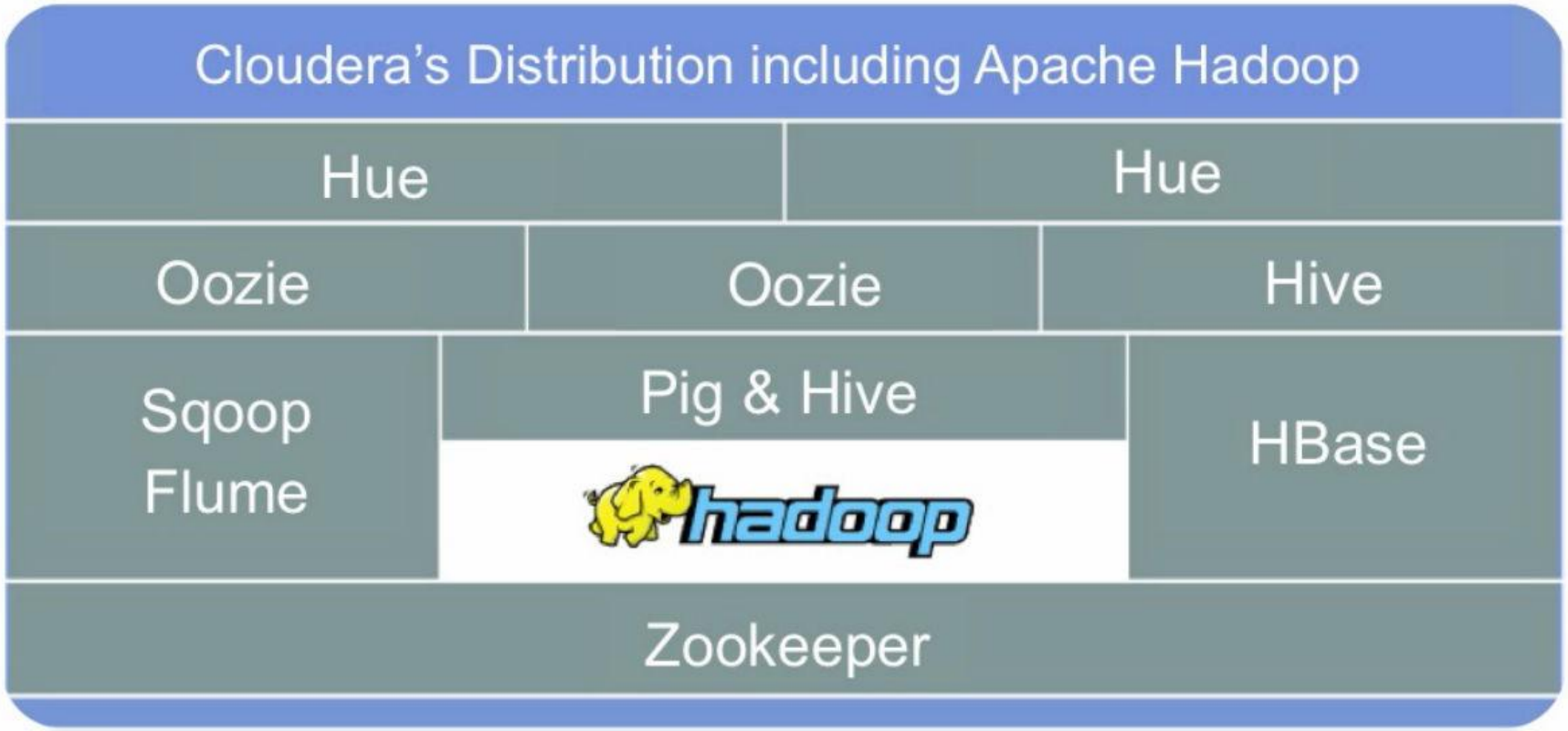
HDFS 특성

- HDFS는 일반하드웨어에서 대용량의 처리 지원
  - 부품의 잦은 오류에 견딜수 있도록 설계
  - 추가되거나 읽기위주의 대용량 파일에 최적화
    (modify 특성의 데이터에는 부적합)
  - 파일시스템의 인터페이스는 해당 작업을 위해서 커스터마이징 가능
    (개발자는 기존에 익숙한 방식으로 처리)
  - 간단한 솔루션으로 활용도 가능(ex.단일 마스터)
- 각각의 클러스터에 수TB 용량의 자료를 신뢰성 있게 분산 저장

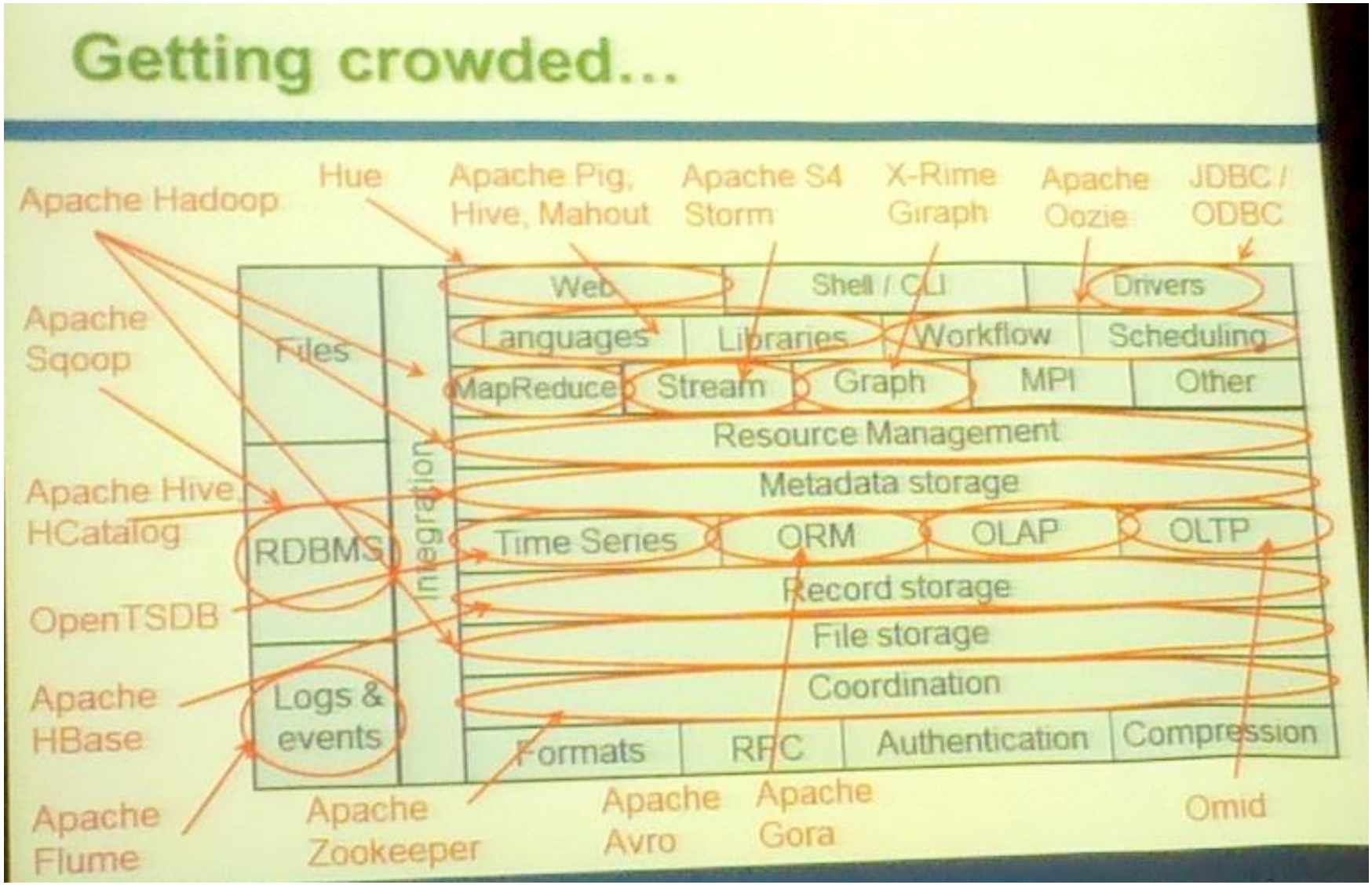
하둡을 사용하는 주요 도메인



Cloudera : 대표적인 하둡 생태계

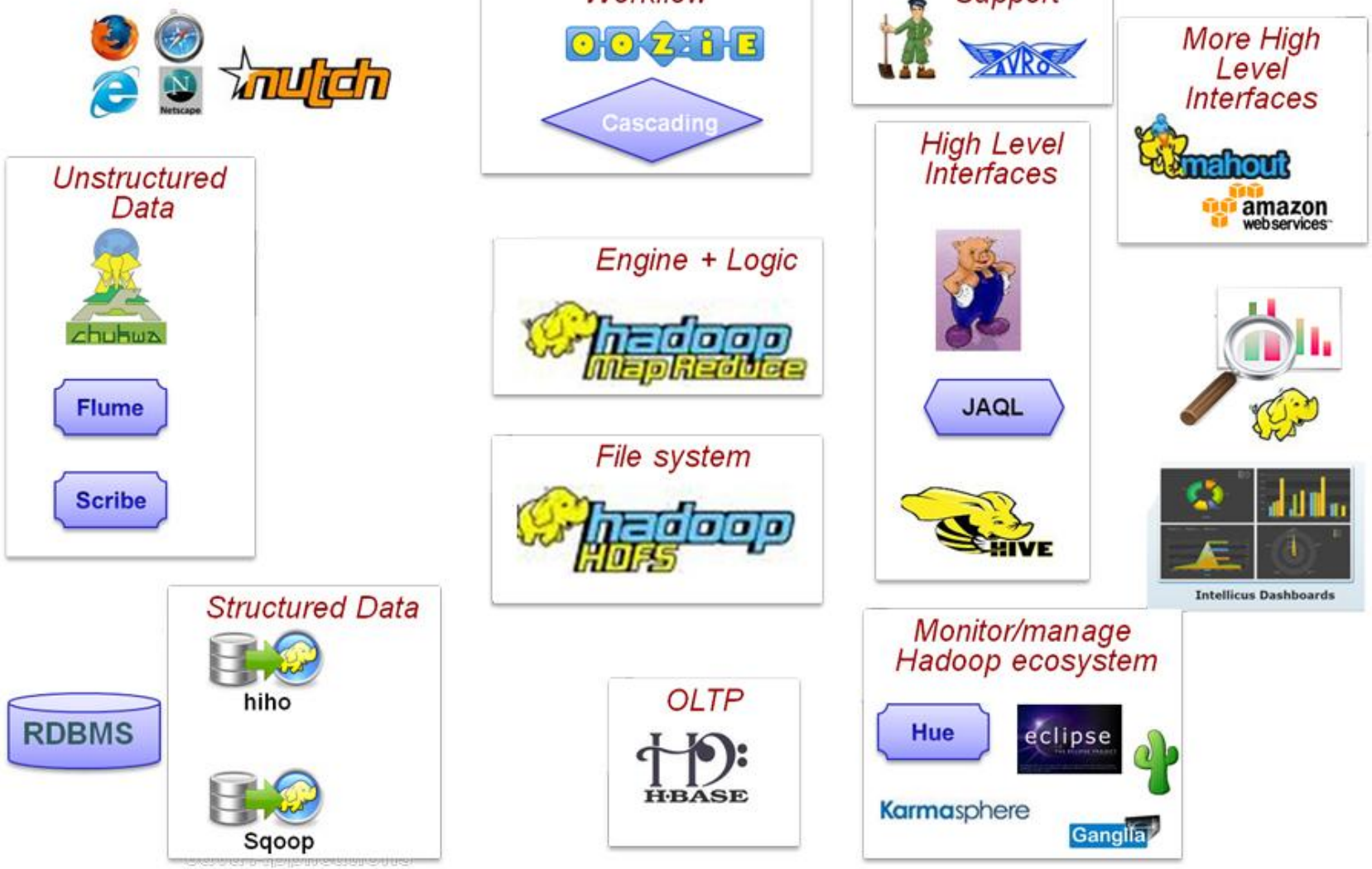


Ecosystem


Getting crowded...
Apache Hadoop
Hue
Apache Pig, Hive, Mahout
Apache S4 Storm
X-Rime Giraph
Apache Oozie
JDBC / ODBC
Apache Sqoop
Apache Hive, HCatalog
OpenTSDB
Apache HBase
Apache Flume
Apache Zookeeper
Apache Avro
Apache Gora
Omid
Files
RDBMS
Logs & events
Integration
Web
Shell / CLI
Drivers
Languages
Libraries
Workflow
Scheduling
MapReduce
Stream
Graph
MPI
Other
Resource Management
Metadata storage
Time Series
ORM
OLAP
OLTP
Record storage
File storage
Coordination
Formats
RPC
Authentication
Compression

3. 하둡 생태계

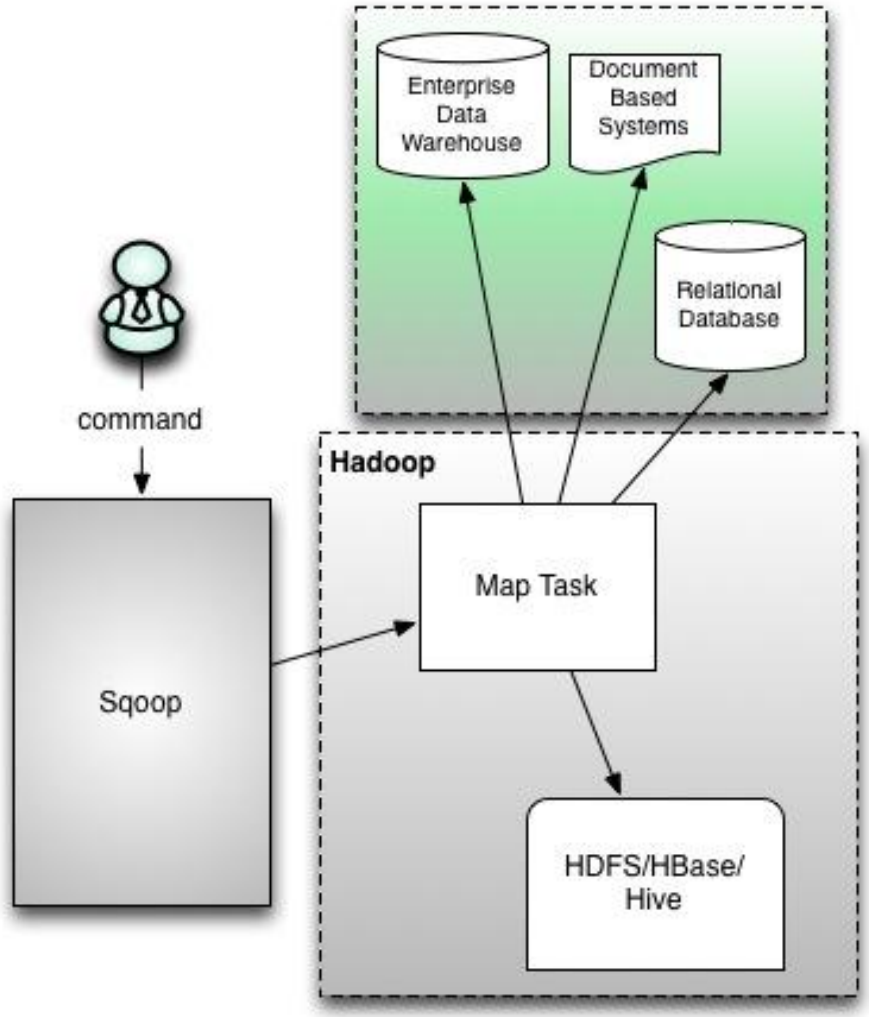
Ecosystem Map

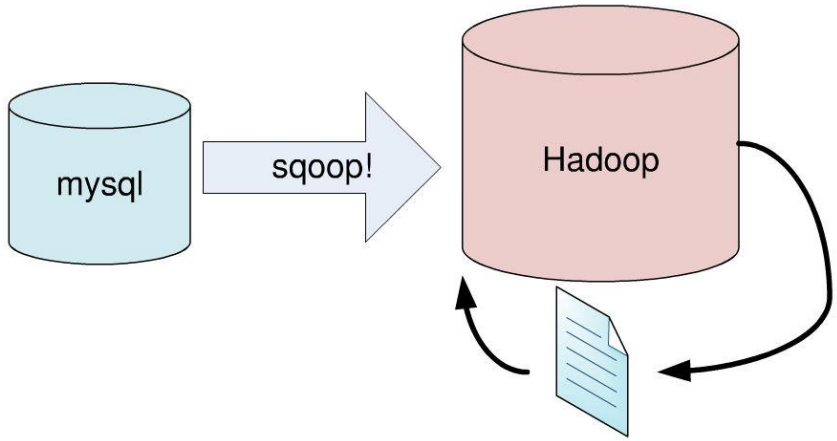


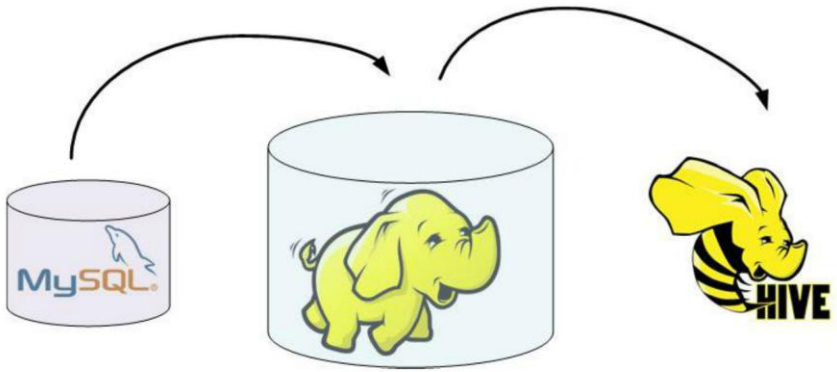
Sqoop

- 다양한 DBMS의 자료를 HDFS로 Import/Export
  - Command line 인터페이스
  - JDBC 지원하는 모든 DBMS
  - RDBMS(Mysql, Oracle)
    + Data Warehouse + NoSQL Datastore
- MapReduce 를 위한 프로그램 코드의 생성
- 하둡기반 시스템과의 통합
  - Hive, HBase, Oozie
- RDBMS와의 고성능 커넥터 지원
- Cloudera에서 개발

Sqoop


Enterprise Data Warehouse
Document Based Systems
Relational Database
command
Hadoop
Map Task
Sqoop
HDFS/HBase/ Hive

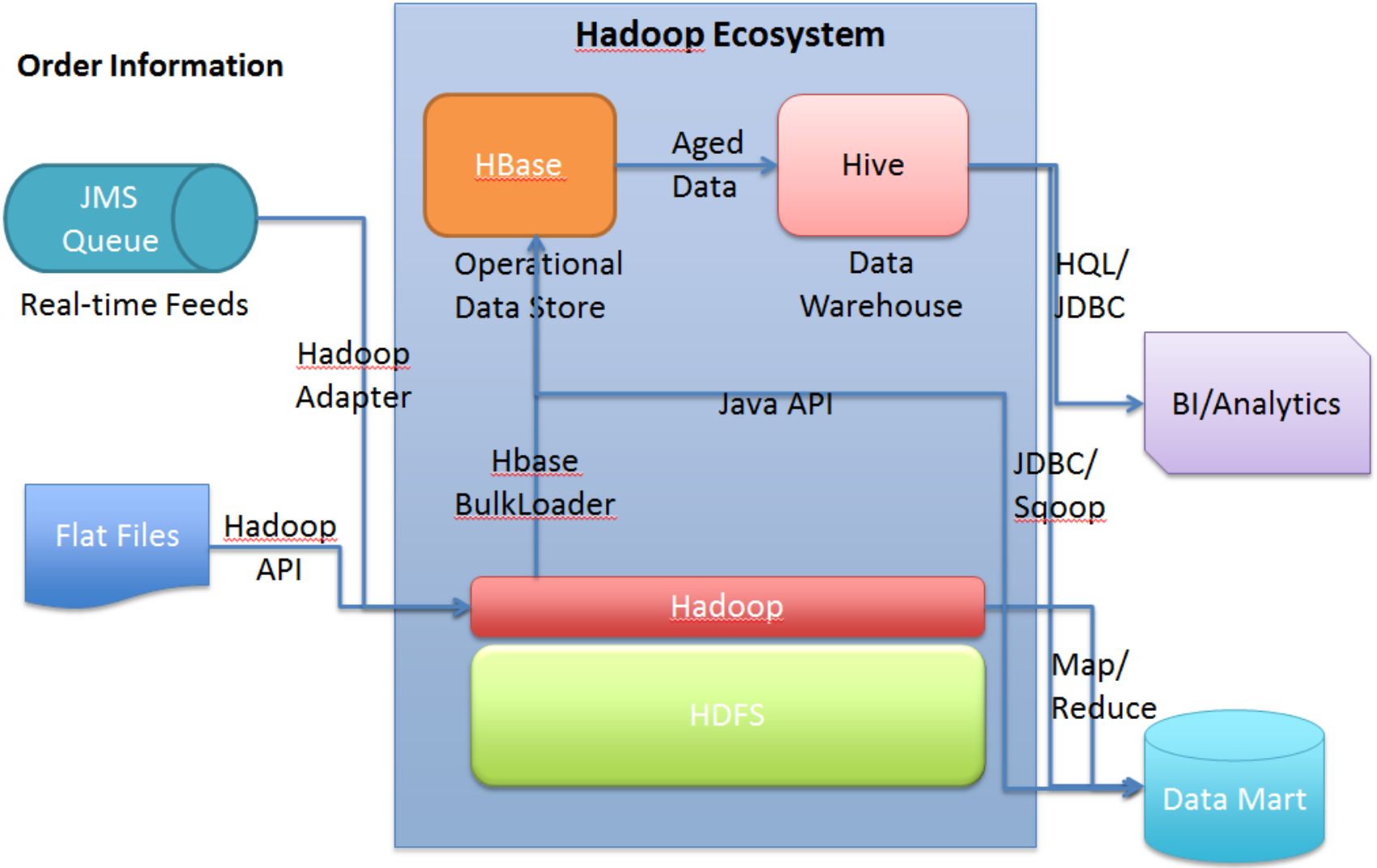

mysql
sqoop!
Hadoop


MySQL
HIVE

HBase

- 컬럼 구조의 저장소
- 구글 빅테이블의 설계를 기반으로 개발
- 인터페이스 제공
- 대용량 데이터를 안정적으로 처리
- 접근 모델을 유지함
- (key,value) 룩업
- 전송제한

HBase

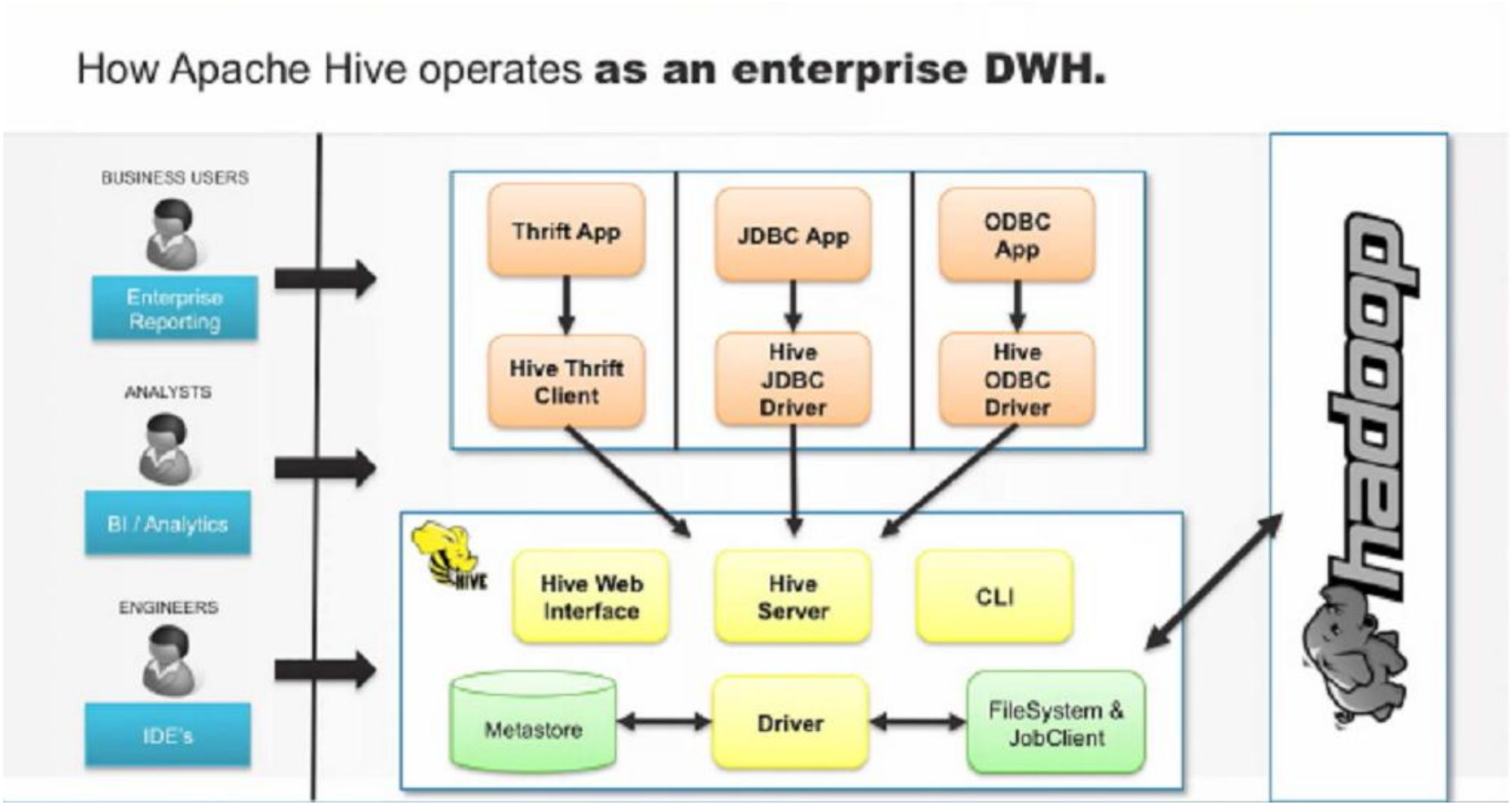

Order Information
Hadoop Ecosystem
JMS Queue
Real-time Feeds
HBase
Aged Data
Hive
Operational Data Store
Data Warehouse
HQL/ JDBC
Hadoop Adapter
Java API
BI/Analytics
Hbase BulkLoader
JDBC/ Sqoop
Flat Files
Hadoop API
Hadoop
HDFS
Map/ Reduce
Data Mart

Hive

- 하둡의 상위에 위치한 SQL 기반 Data Warehouse
  - SQL과 유사한 문법
  - Select, Join, Group by, Limit 지원
- 대용량 데이터의 분석에 접합
  - 테이블
  - 파티션 컬럼
  - 버킷(샘플링)
- Shell : 인터렉티브 쿼리 지원
  - Web & JDBC 클라이언트 제공
- 페이스북이 개발

3. 하둡 생태계

Hive


How Apache Hive operates as an enterprise DWH.
BUSINESS USERS
Enterprise Reporting
ANALYSTS
BI / Analytics
ENGINEERS
IDE's
Thrift App
JDBC App
ODBC App
Hive Thrift Client
Hive JDBC Driver
Hive ODBC Driver
Hive Web Interface
Hive Server
CLI
Metastore
Driver
FileSystem & JobClient
hadoop

Pig

- 데이터-흐름 기반의 스크립트 프로그래밍 언어
  - "Pig Latin"
  - 집합, 연관배열, Tuple등의 데이터타입을 포함
  - 데이터를 처리하는 하이레벨 언어
    자바 프로그램으로 복잡한 태스크를 쉽게 처리가능
- 야후가 개발

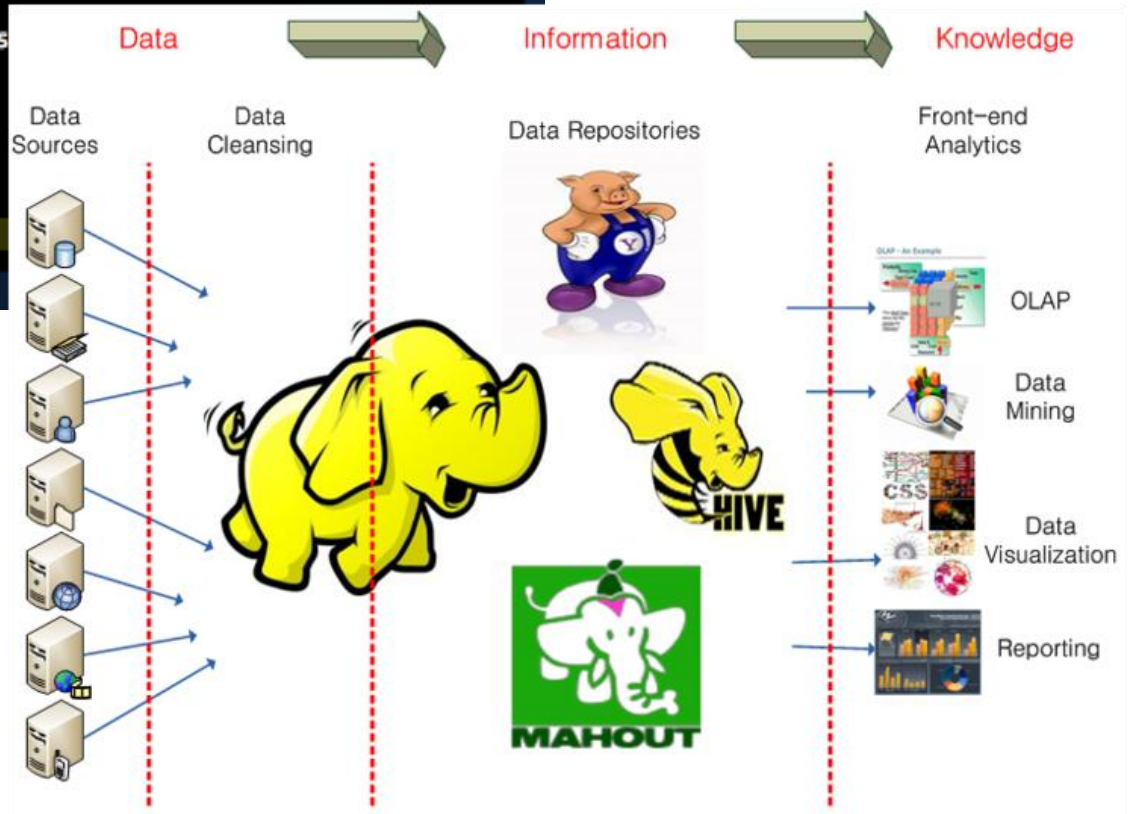
3. 하둡 생태계

Pig

A Real Pig Script

top_5.pig

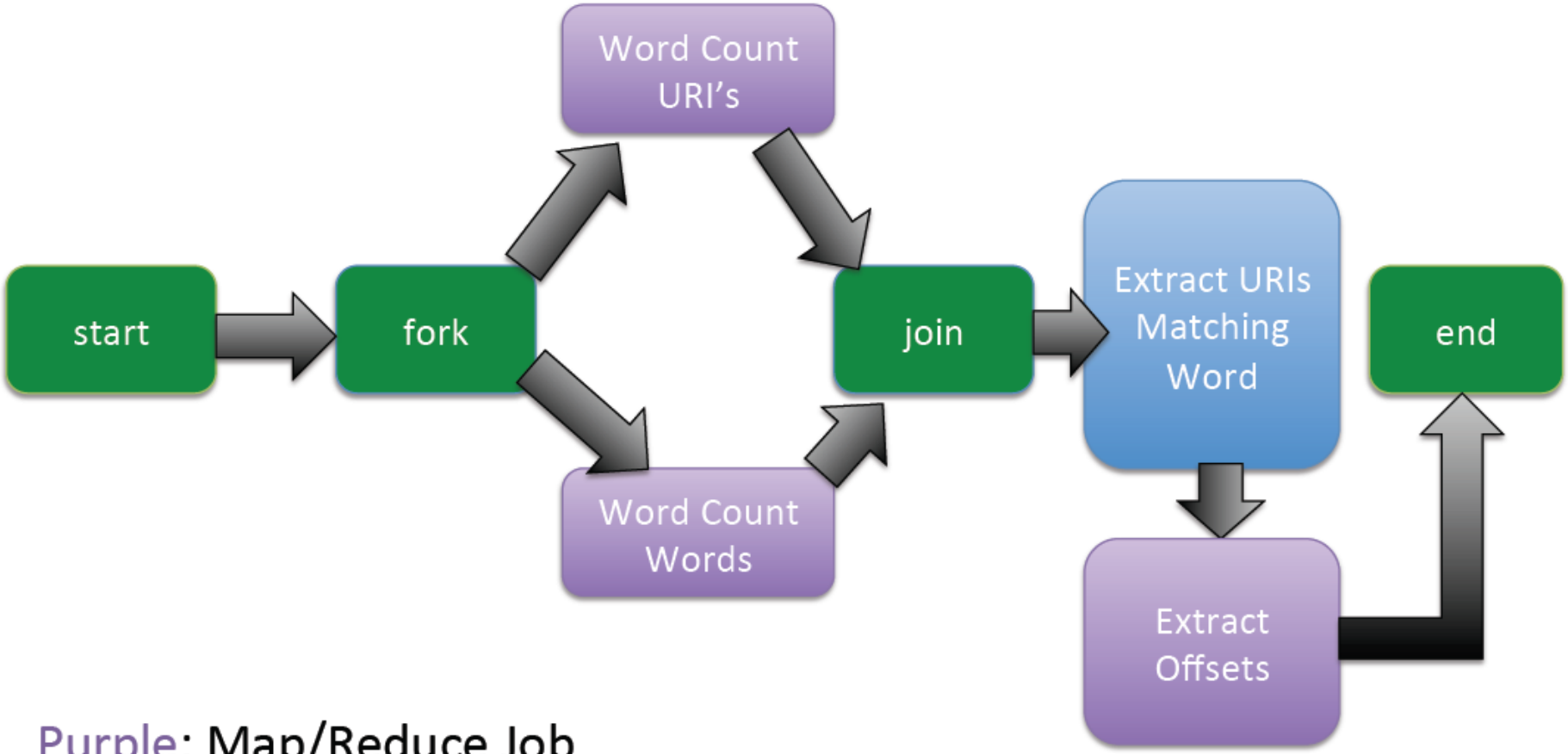
```
users = load 'users.csv' as (username: chararray, age: int);
users_1825 = filter users by age >= 18 and age <= 25;

pages = load 'pages.csv' as (username: chararray, url: chararray);

joined = join users_1825 by username, pages
grouped = group joined by url;
summed = foreach grouped generate group as
sorted = order summed by views desc;
top_5 = limit sorted 5;

store top_5 into 'top_5_sites.csv';
```



Oozie

- 하둡의 Workflow 스케쥴러
  - XML 기반, DAG
- 코디네이터 지원 : 스케쥴링/모니터링
- HTTP 인터페이스
  + Command Line 인터페이스 + Web 콘솔
- 다양한 엑션노드(어플)를 지원하고 제어함
- - MapReduce in HDFS - Pig, Hive...
- - 일반 자바 프로그램
- 야후에서 복잡한 검색엔진의 Workflow를 위해 개발
- 오픈소스 아파치 재단 프로젝트

Oozie



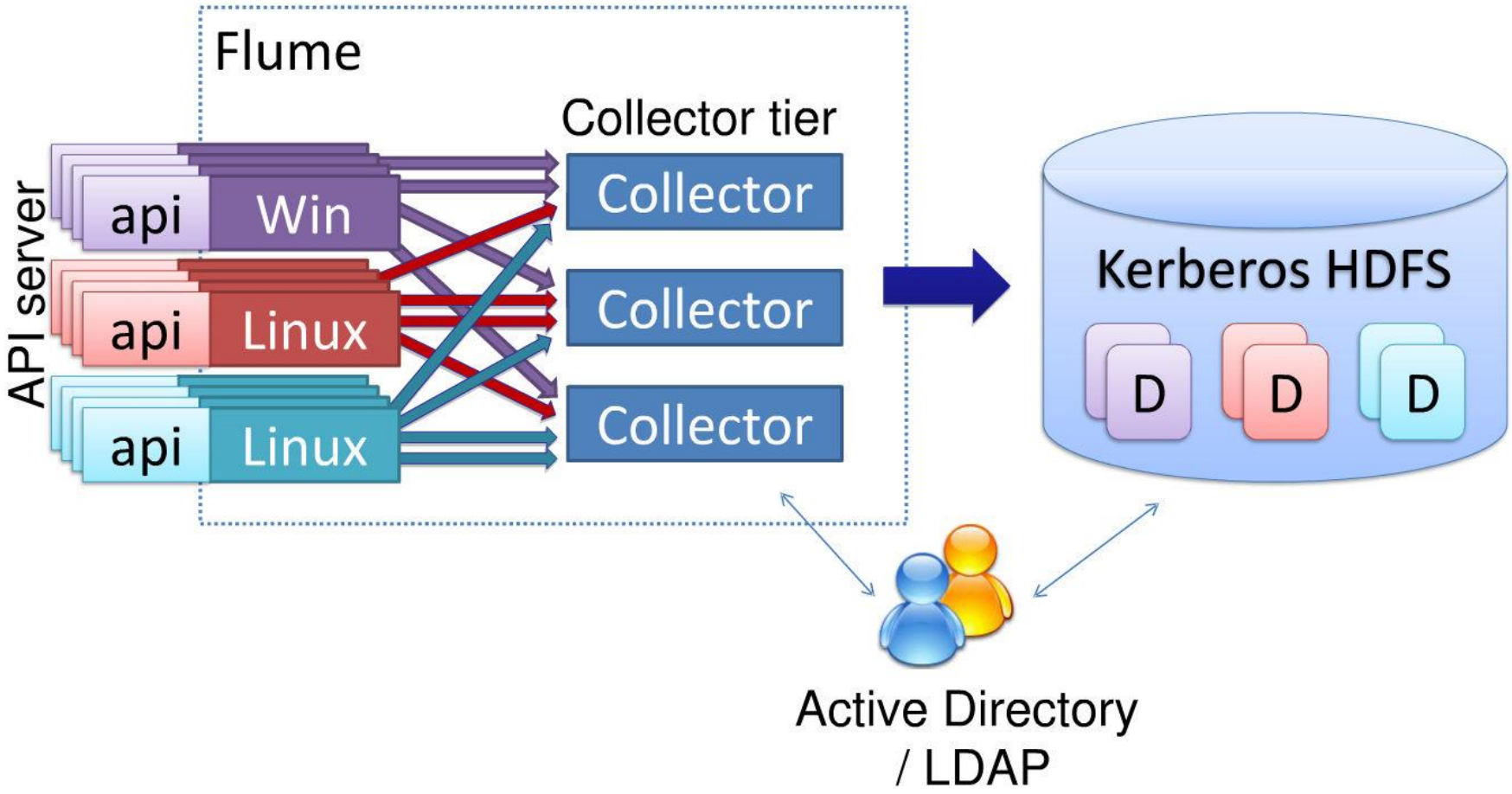
Purple: Map/Reduce Job

Blue: Java Job

Flume

- 다양한 다수의 서버로 부터 데이터를 안정적으로 수집해서 HDFS에 저장하는 프레임워크
  - 로그등 대용량 실시간 데이터 수집
  - 에이젼트와 콜렉터 다중 구성
  - 데이터를 수비해서 HDFS에 저장 및 통합관리 지원
- Flume의 특성
  - 확장성 : 모든 노드와 마스터가 수평적으로 확장
  - 신뢰성 : Fault-tolerance(내고장성), 고성능
  - 다양성 : Unix 기본지원, 모든 종류의 데이터와 시스템
  - 관리성 : 동적 재구성이 가능한 통합 관리 지원
  - 개방성 : Apache v2.0 라이센스, 오픈소스데이터

Flume

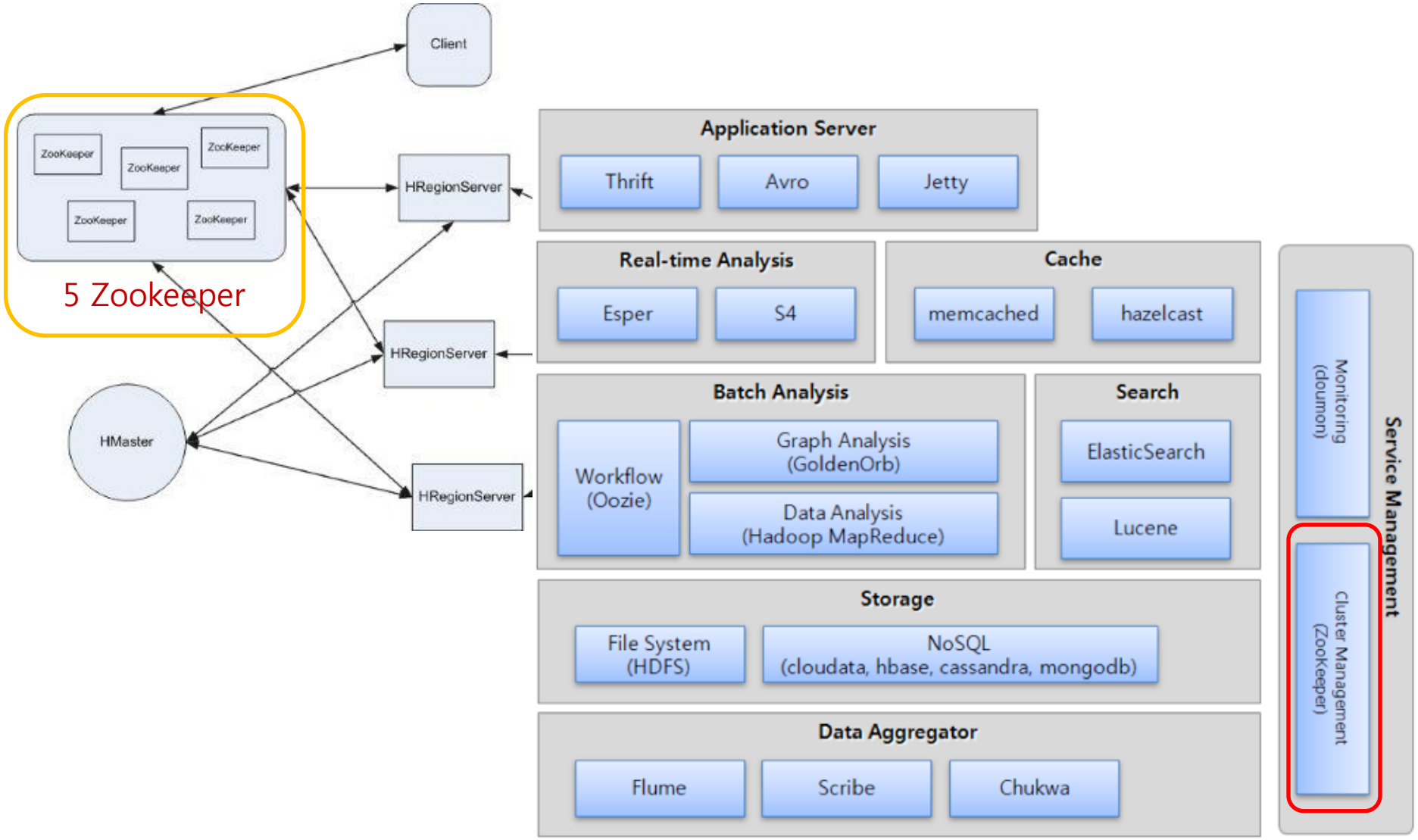

Flume
Collector tier
API server
api
Win
api
Linux
api
Linux
Collector
Collector
Collector
Kerberos HDFS
D
D
D
Active Directory
/ LDAP

Zookeeper

- 분산 락+공유 합의체 엔진
  - 다수의 zookeeper 서버는 모두 동등하고(임시리더)
    모든 자원을 공유(SPOF 방지)
- 주요 용도 및 특징
  - 분산 클러스터 머신과 자원의 상태관리
  - 리더 선출
  - 분산 Lock
  - 상호배제 : Deadlock 방지
  - 이벤트 처리

3. 하둡 생태계

Zookeeper



MR : Pipe와 Streaming

- MapReduce를 위한 다양한 언어 커넥터 라이브러리
- Pipe – C++ 언어를 위한 Wrapper 라이브러리
- Streaming – Python,Shell 등의 스크립트언어를 활용하는 Command Line 인터페이스

HDFS : FUSE

- 리눅스의 FUSE 파일시스템에 HDFS를 Mount

  - HDFS는 범용의 파일시스템에는 부적합
    (사이즈가 작은 다수의 이미지 데이터 등)

  - 다른 시스템과의 데이터의 입력/출력을 쉽게 지원

기타 프로젝트

- Avro : 직렬화, RPC 프레임워크
- Mahout : 머신러닝 라이브러리
- Dumbo – Streaming을 위한 Python 라이브러리
- Chukwa – 로그 수집 분석 및 HICC

감사합니다