

The background features a large, abstract, wavy shape in shades of green and white, resembling a stylized wave or a flowing ribbon. The shape is composed of several overlapping, curved segments that create a sense of movement and depth. The colors range from a bright, vibrant green to a soft, pale green, with white highlights and shadows that give it a three-dimensional appearance. The overall effect is clean, modern, and visually appealing.

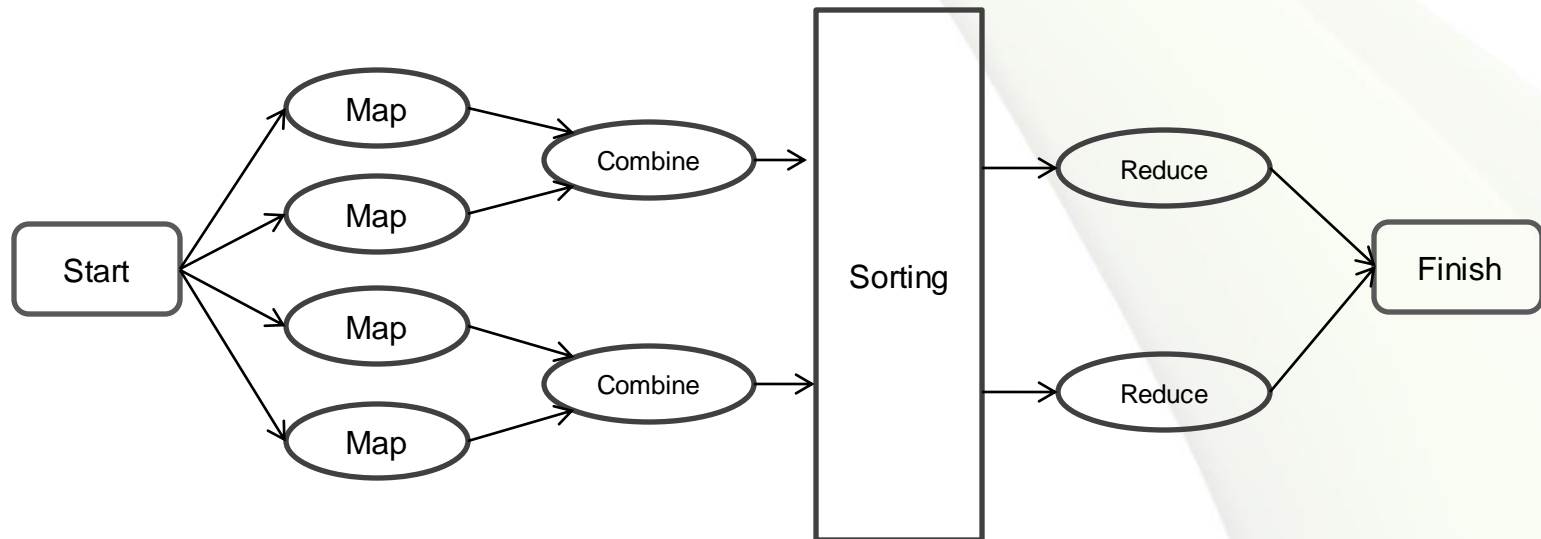
# **MapReduce & Yarn**

# MapReduce

- HDFS에 분산 저장된 데이터에 대한 분산 배치 분석을 지원하는 시스템
- 대규모 분산 컴퓨팅 혹은 단일 컴퓨팅 환경에서 개발자가 대량의 데이터를 병렬로 분석
- 개발자는 맵리듀스 알고리즘에 맞게 분석 프로그램을 개발하고, 데이터의 입출력과 병렬 처리 등 기반 작업은 프레임워크를 통해 처리

# MapReduce

- MapReduce는 Map과 Reduce 두 단계를 통해 데이터 처리
- Map :  $(k1, v1) \rightarrow list(k2, v2)$ 
  - 데이터를 가공해서 분류
- Reduce :  $(k2, list(v2)) \rightarrow list(k3, v3)$ 
  - 분류된 데이터를 통합(집계 연산자)

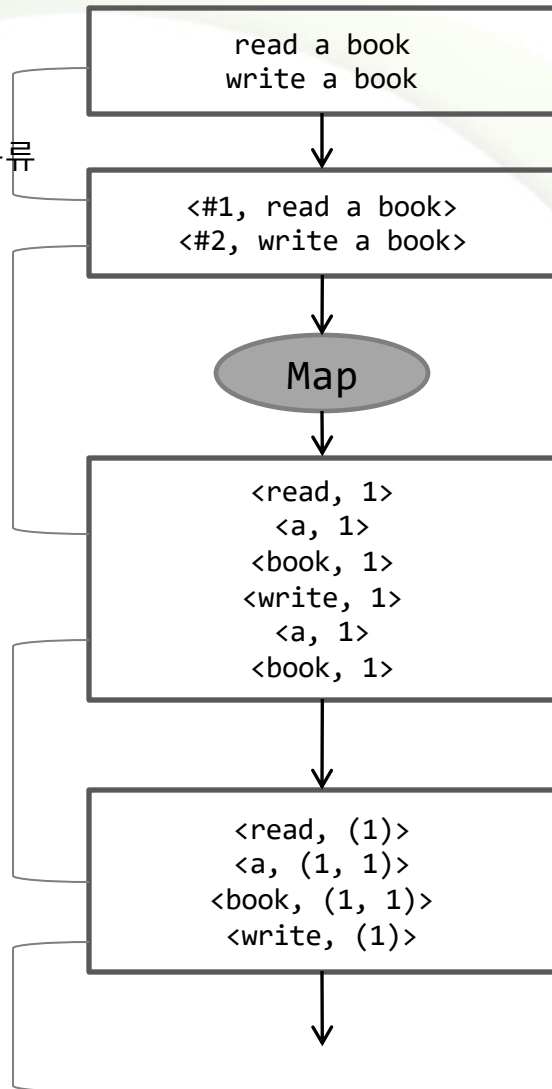


# MapReduce

**입력 데이터 분리**  
: 키와 값으로 분류

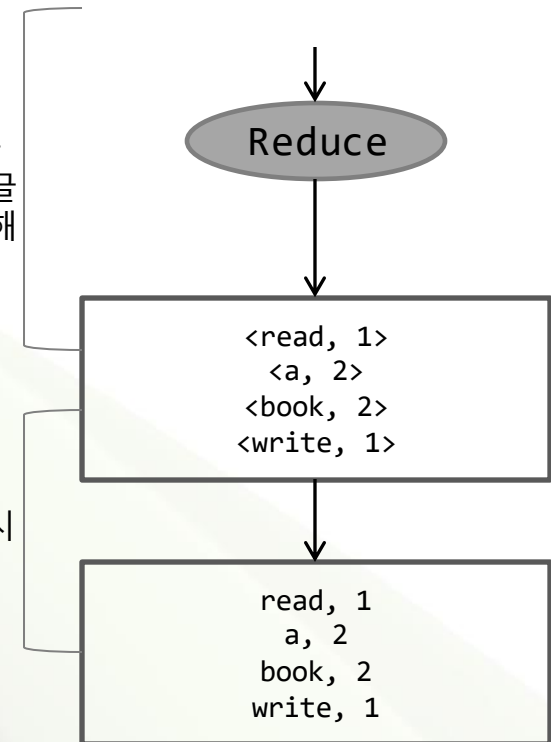
**맵 메서드**  
: 라인별로 문장을 체크, 키에 해당하는 글자별로 글자 수 출력

**정렬과 병합**  
: 맵 메서드의 출력 데이터를 정렬, 병합



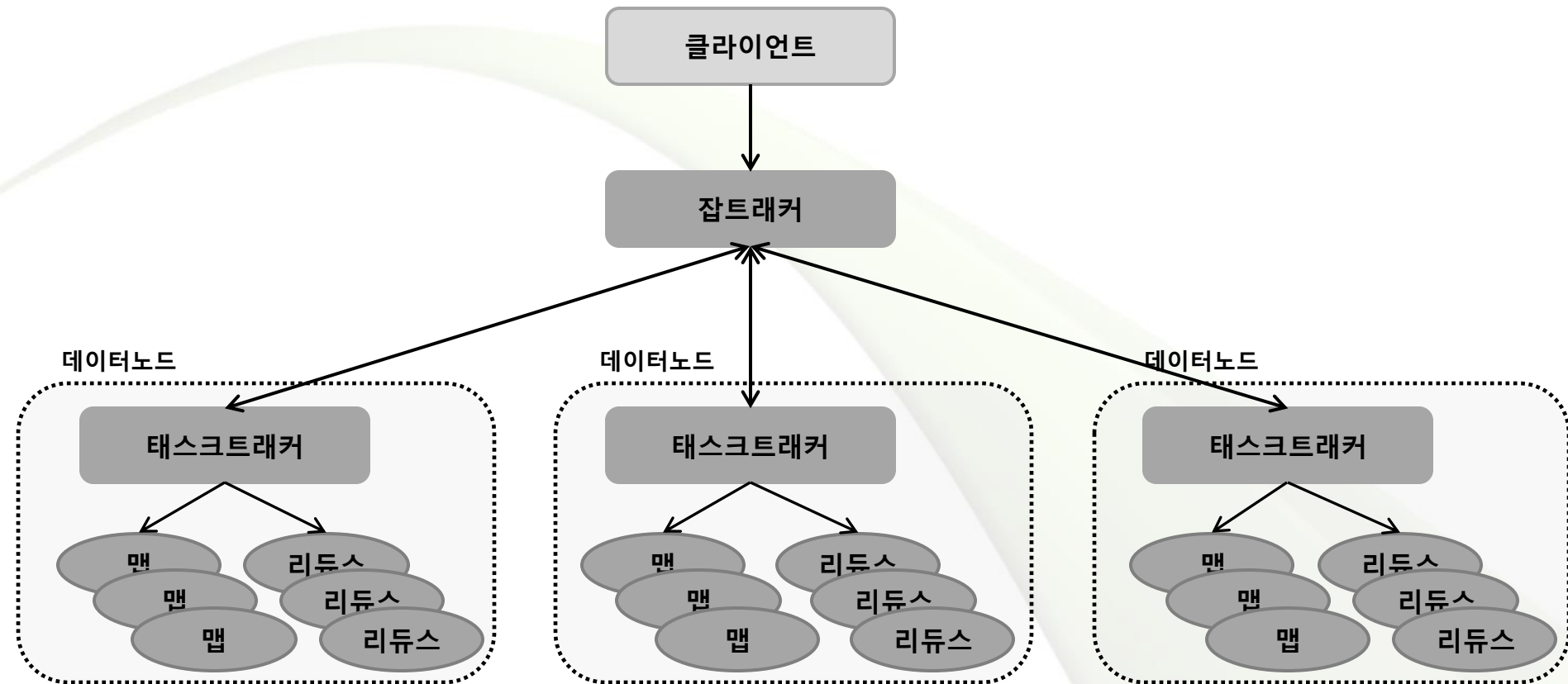
**리듀스 메서드**  
: 키에 해당하는 글자별로 글자 수를 합산해서 출력

**저장**  
: 하둡 파일 시스템에 저장



Map input records=2  
Map output records=6  
Reduce input records=6  
(Reduce input groups=4)  
Reduce output records=4

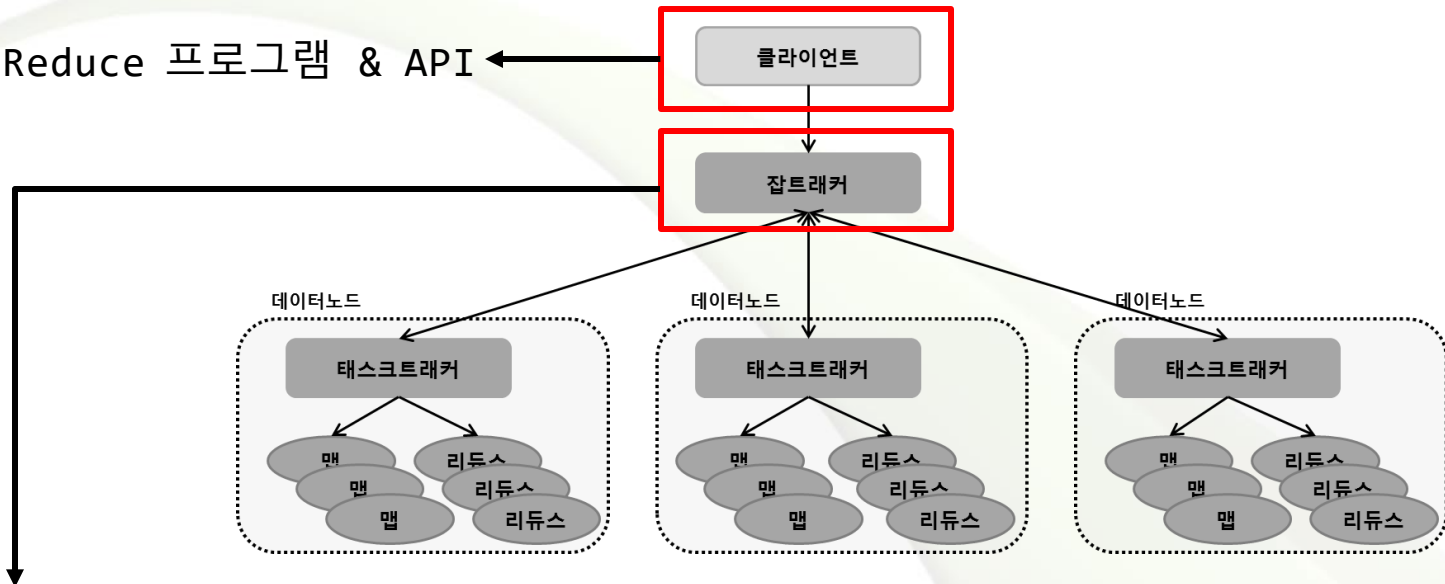
# MapReduce Architecture



# MapReduce 구성 요소

## ■ 클라이언트

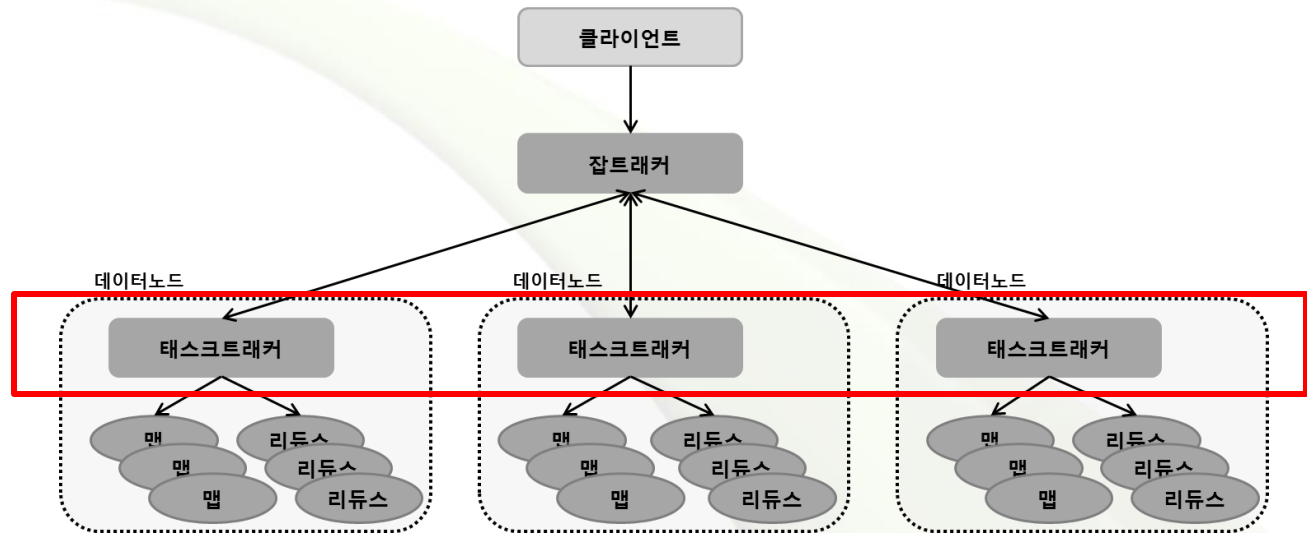
- MapReduce 프로그램 & API



## ■ 잡트래커(JobTracker, 네임노드 서버에서 실행)

- 맵리듀스 프로그램은 잡(Job)이라는 하나의 작업 단위로 관리
- 하둡 클러스터에 등록된 전체 잡의 스케줄링을 관리하고 모니터링
- 사용자가 새로운 잡을 요청하면 잡트래커는 잡을 처리하기 위해 몇 개의 맵과 리듀스를 실행할지 계산 → 어떤 태스크 트래커에서 실행할지 결정 → 잡 할당
- 전체 클러스터에서 하나의 잡 트래커가 실행됨

# MapReduce Architecture



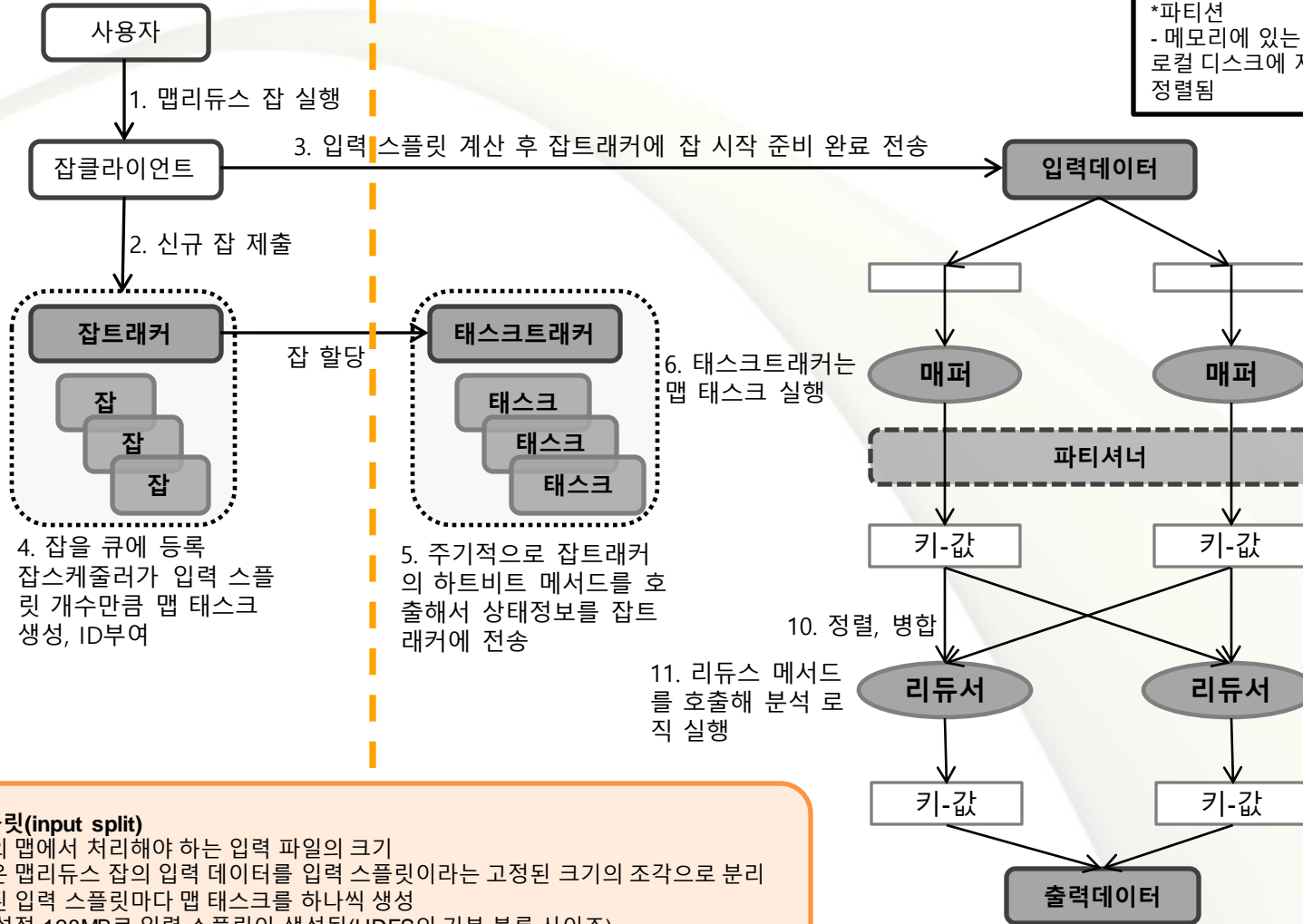
- 태스크 트래커(TaskTracker) (데이터 노드에서 실행)
  - 잡 트래커의 작업 수행 요청을 받아 맵리듀스 프로그램 실행
  - 잡 트래커가 요청한 맵과 리듀스 개수만큼 맵, 리듀스 태스크를 생성
  - 새로운 JVM을 구동해 맵, 리듀스 태스크 실행
  - 이 때, JVM은 재사용할 수 있게 설정 가능. 데이터 노드가 하나라도 여러 개의 JVM을 실행해서 데이터를 동시에 분석함(병렬처리 가능)

# MapReduce 작동방식

네임노드

데이터노드

- \*입력스플릿 결과
  - 입력 스플릿 정보, 설정 파일, 맵리듀스 JAR 파일을 HDFS에 저장
- \*하트비트 메시지를 통해 상태정보 전송
  - CPU, 메모리, 서버가용 리소스정보, 현재 실행 중 태스크 개수, 실행 가능한 최대 태스크 개수, 신규 태스크 실행 가능 여부 등
- \*파티션
  - 메모리에 있는 데이터를 키에 따라 정렬 후, 로컬 디스크에 저장, 하나의 출력 파일로 병합, 정렬됨



## 입력스플릿(input split)

- 하나의 맵에서 처리해야 하는 입력 파일의 크기
- 하둡은 맵리듀스 잡의 입력 데이터를 입력스플릿이라는 고정된 크기의 조각으로 분리
- 생성된 입력스플릿마다 맵태스크를 하나씩 생성
- 기본 설정 128MB로 입력스플릿이 생성됨(HDFS의 기본 블록 사이즈)



# MapReduce 프로그래밍 요소

- 데이터 타입
- InputFormat
- 매퍼 (Mapper)
- 파티셔너 (Partitioner)
- 리듀서 (Reducer)
- 콤바이너
- OutputFormat

# 데이터 타입

- 맵리듀스 API는 자주 쓰는 데이터타입에 대한 WritableComparable 인터페이스를 구현한 Wrapper 클래스 제공

클래스명	대상 데이터 타입
BooleanWritable	Boolean
ByteWritable	단일 byte
DoubleWritable	Double
FloatWritable	Float
IntWritable	Integer
LongWritable	Long
Text Wrapper	UTF8 형식의 문자열
NullWritable	데이터값이 필요 없을 경우에 사용

- WritableComparable 인터페이스
  - Writable & Comparable 인터페이스를 다중 구현하고 있음
  - Comparable 인터페이스
    - 정렬을 처리하기 위해 compareTo 메서드 제공
  - Writable 인터페이스
    - write 메서드 → 데이터값을 직렬화함
    - readFields 메서드 → 직렬화된 데이터값을 해제해서 읽는 역할

# InputFormat

- 입력 스플릿(Input Split)을 맵 메서드의 입력 파라미터로 사용할 수 있게 InputFormat 추상화 클래스 제공
- InputFormat 클래스
  - getSplits 메서드 → 입력 스플릿을 맵 메서드가 사용할 수 있도록 함
  - createRecordReader 메서드 → 맵 메서드가 입력 스플릿을 키와 목록의 형태로 사용할 수 있게 RecordReader 객체 생성함
  - 맵 메서드는 RecordReader 객체에 포함된 키와 값을 읽어들이어 분석 로직 수행

# InputFormat

## ■ InputFormat 클래스 종류

InputFormat	기능
TextInputFormat	텍스트 파일을 분석할 때 사용. 키는 LongWritable 타입, 값은 Text타입을 사용
KeyValueTextInputFormat	텍스트 파일을 입력 파일로 사용할 때 라인 번호가 아닌 임의의 키값을 지정해서 키와 값의 목록으로 읽음
NLineInputFormat	맵 태스크가 입력 받은 텍스트 파일의 라인 수를 제한하고 싶을 때 사용
DelegatingInputFormat	여러 개의 서로 다른 입력 포맷을 사용하는 경우에 각 경로에 대한 작업을 위임
CombineFileInputFormat	여러 개의 파일을 스플릿으로 묶어서 아용(다른 InputFormat은 파일당 스플릿을 생성)
SequenceFileInputFormat	시퀀스 파일(바이너리 형태의 키와 값의 목록으로 구성된 텍스트 파일)을 입력 데이터로 쓸 때 사용.
SequenceFileAsBinaryInputFormat	시퀀스 파일의 키와 값을 임의의 바이너리 객체로 변환해서 사용
SequenceFileAsTextInputFormat	시퀀스 파일의 키와 값을 Text 객체로 변환해서 사용

# 매퍼(Mapper)

- 매퍼듀스 프레임워크는 입력 스플릿마다 하나의 맵 태스크(매퍼 클래스)를 생성함.
- 대부분 이 매퍼 클래스를 상속해서 매퍼 클래스를 새롭게 구현하여 사용함
- Context 객체를 이용해 job에 대한 정보를 얻어오고, 입력 스플릿을 레코드 단위로 읽을 수 있음
  - RecordReader로 맵 메서드가 키와 값의 형태로 데이터를 읽을 수 있음
- map 메서드 → 실제 데이터 처리 내용을 구현하기 위해 재정의
- run 메서드 → Context 객체에 있는 키를 순회하면서 맵 메서드를 호출함

# 파티셔너 (Partitioner)

- 맵 태스크의 출력 데이터가 어떤 리듀스 태스크로 전달될지 결정
- 지정하지 않을 경우 HashPartitioner가 사용됨
  - `getPartition` 메서드에서 맵태스크 출력키의 Hash 값 % 전체 리듀스 태스크 개수로 파티션 할당

# 리듀서(Reducer)

- 맵 태스크의 출력 데이터를 입력 데이터로 전달받아 집계 연산을 수행
- reduce 메서드를 재정의해서 실제 데이터 처리 내용 구현
- run 메서드에서 데이터 개수 만큼 reduce를 반복해서 호출

# 콤바이너 클래스(Combiner)

- 셔플(Shuffle)
  - 맵 태스크와 리듀스 태스크 사이의 데이터 전달 과정
  - 맵 태스크의 출력 데이터는 네트워크를 통해 리듀스 태스크로 전달됨
- 콤바이너 클래스
  - 셔플할 데이터의 크기를 줄이는 데 도움을 줌
  - 매퍼의 출력 데이터를 입력 데이터로 전달받아 연산을 수행
  - 로컬 노드에서 로컬에 생성된 매퍼의 출력 데이터를 이용하기 때문에 네트워크 비용이 발생하지 않음



# OutputFormat

- 맵리듀스 잡의 출력 포맷은 OutputFormat 추상 클래스를 상속해서 구현
- 자주 사용되는 형식은 Hadoop이 미리 만들어서 제공
- 사용자가 명시적으로 지정하지 않을 경우 TextOutputFormat을 기본 형식으로 사용

OutputFormat	기능
TextOutputFormat	텍스트 파일에 레코드를 출력할 때 사용. 키, 값의 구분자는 탭을 사용
SequenceFileOutputFormat	시퀀스 파일을 출력물로 쓸 때 사용
SequenceFileAsBinaryOutputFormat	위에 있는 것을 상속받아 구현되었음. 바이너리 포맷의 키와값을 SequenceFile 컨테이너에 씀
FilterOutputFormat	OutputFormat 클래스를 편리하게 사용할 수 있는 메서드 제공 (OutputFormat 클래스의 Wrapper 클래스)
LazyOutputFormat	FileOutputFormat을 상속받은 클래스는 출력할 내용이 없을때도 part-nnnnn을 생성함. 이 포맷을 사용하면 첫 번째 레코드가 해당 파티션(part-nnnnn)으로 보내질 때만 출력 파일을 생성함
NullOutputFormat	출력 데이터가 없을 때 사용

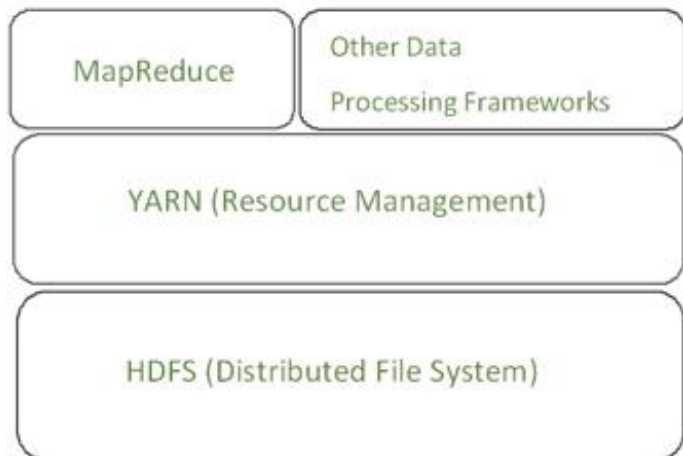
# MapReduce의 문제

- 단일 고장점 (Single Point of Failure)
  - 모든 작업이 잡트래커를 통해 이루어지기 때문에 잡트래커에 문제가 생기면 전체 시스템이 동작하지 않는 문제 발생
- 메모리 효율성
  - 메모리에 전체 잡의 실행 정보를 유지하기 때문에 메모리 수요가 매우 높음
- 맵 또는 리듀스 중 하나의 처리가 실행되는 경우 나머지 자원을 사용할 수 없게 되는 문제
- 클러스터 확장성
  - 최대 동시 실행 태스크 제약
- 버전 동일성
  - 클라이언트와 클러스터의 버전이 일치해야 동작하는 문제

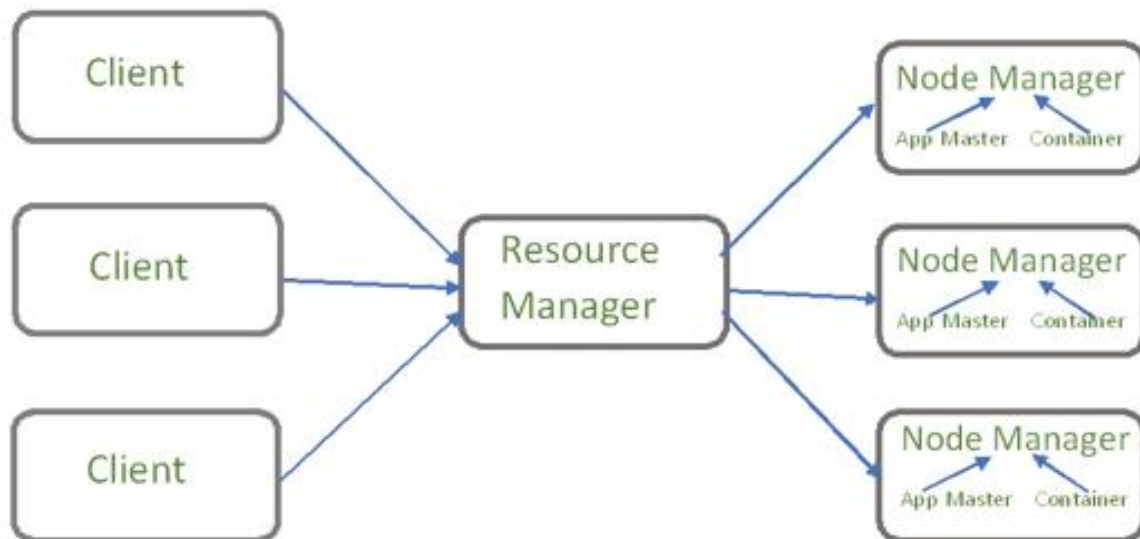
# Yarn (Yet Another Resource Negotiator)

- MapReduce의 문제 해결을 위해 Hadoop2부터 적용된 분산 데이터 처리 프레임워크
- 특징
  - 잡트래커의 주요 기능 추상화
  - 클러스터 규모 확장
  - 클러스터 활용 개선
  - 수용 가능 작업 확장
  - 맵리듀스 호환성

# Yarn 실행 구조



Hadoop 2.0



Hadoop Yarn architecture