

# 빅데이터 배치성 적재

- 대용량 로그 파일 적재

# 빅데이터 적재 소개

---

## 1. 빅데이터 적재 개요

빅데이터 대용량 파일 적재의 기본 개념을 설명한다.



## 2. 빅데이터 적재에 활용하는 기술

빅데이터 적재에서 사용할 두 가지 기술(하둡, 주키퍼)에 대한 소개와 각 기술별 주요 기능 및 아키텍처, 활용 방안을 알아본다.



## 3. 적재 파일럿 실행 1단계 - 적재 아키텍처

스마트카에서 발생하는 로그 파일 적재와 관련한 요구사항을 구체화하고, 적재 요건을 해결하기 위한 파일럿 아키텍처를 설명한다.



## 4. 적재 파일럿 실행 2단계 - 적재 환경 구성

스마트카 로그 파일의 적재 아키텍처를 설치 및 환경을 구성한다



## 5. 적재 파일럿 실행 3단계 - 적재 기능 구현

플럼을 이용해 스마트카의 상태 정보 로그 파일을 하둡에 적재하는 기능을 구현해 본다.

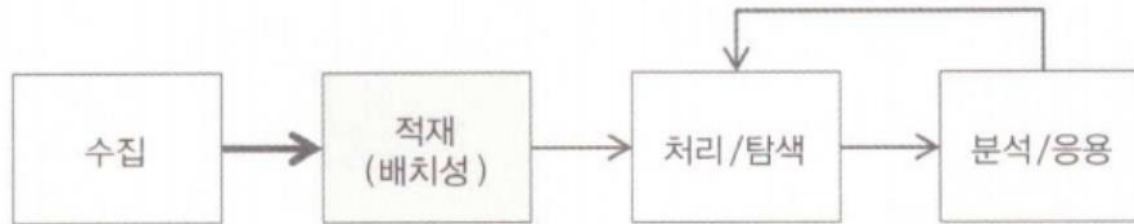


## 6. 적재 파일럿 실행 4단계 - 적재 기능 테스트

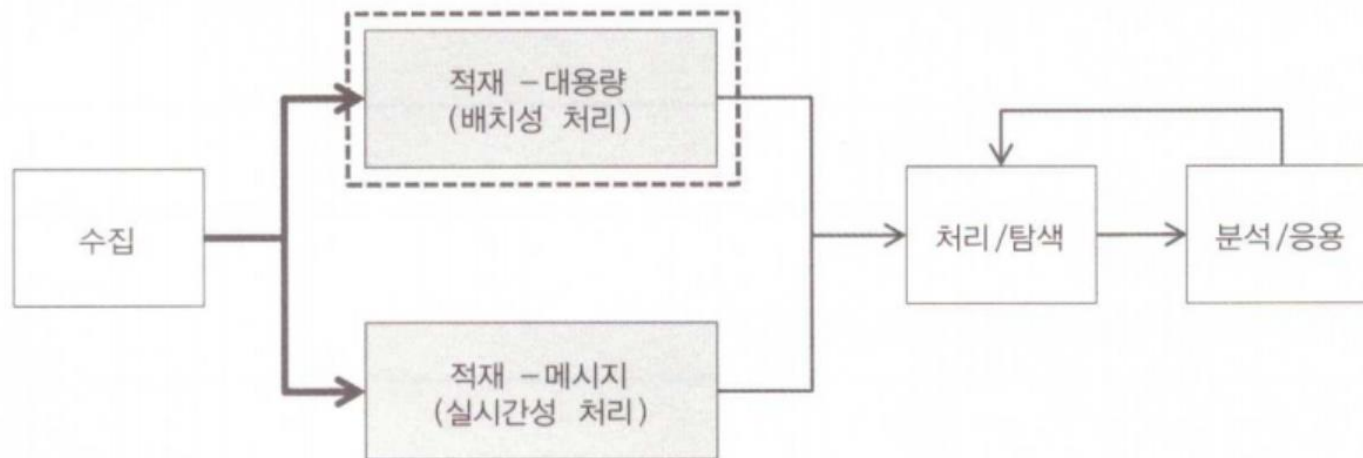
로그 시뮬레이터를 이용해 스마트카의 상태 정보 데이터를 발생시키고, 플럼이 해당 데이터를 HDFS에 정상적으로 적재됐는지 확인한다.

# 빅데이터 적재 개요

## ➤ 빅데이터 구축 단계 - 적재(배치성)



## ➤ 빅데이터 적재 유형



# 빅데이터 적재에 활용하는 기술

---

## ➤ 하둡 소개

- 빅데이터의 핵심 소프트웨어
- 빅데이터의 에코시스템들은 대부분 하둡을 위해 존재하고 하둡에 의존해서 발전
- 하둡의 핵심 기능
  - 대용량 데이터를 분산 저장하는 것
    - 분산 병렬 처리 기술
    - 2003년, 구글, Google File System
    - 2004년, 구글, MapReduce : Simplified Data Processing on Large Clusters
    - 더그 커팅(하둡의 창시자) : 2005년 넛치/루씬(Nutch/Lucene) 검색엔진의 서브 프로젝트로 진행하면서 하둡 프로젝트가 시작.
  - 분산 저장된 대용량 데이터를 분석하는 기능
- 하둡의 맵리듀스 : 분산 병렬 처리에서의 기본은 먼저 수많은 컴퓨터가 있는데 어떻게 일을 효율적으로 나눠서 실행 시킬 수 있느냐고(Map), 두 번째는 수많은 컴퓨터들이 나눠서 만든 결과들을 어떻게 하나로 모으냐(Reduce)는 것이다. 이를 하둡의 맵리듀스(MapReduce)가 쉽게 할 수 있도록 지원해 준다.

# 빅데이터 적재에 활용하는 기술

## ➤ 하둡의 기본 요소

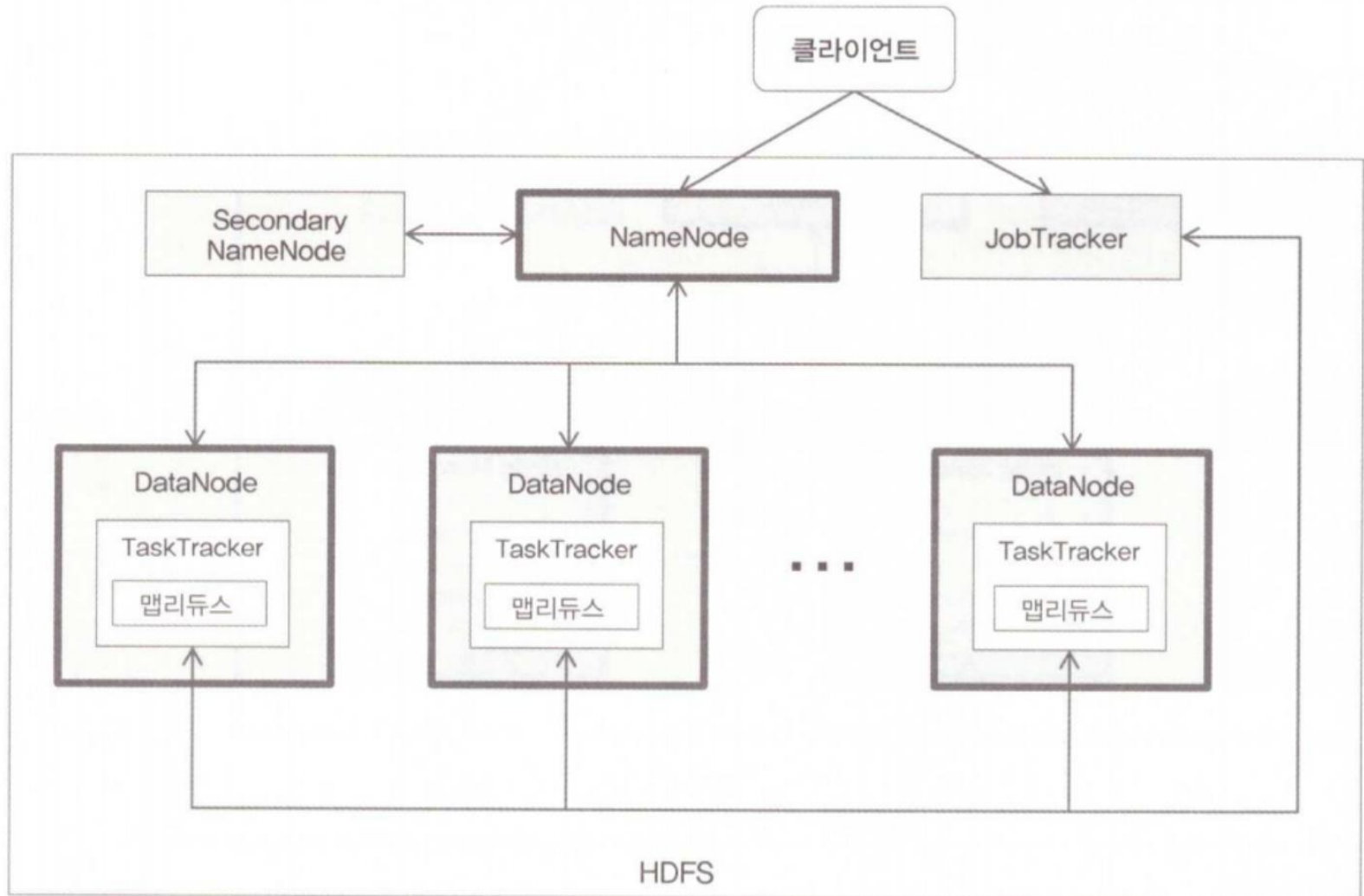
공식 홈페이지			<a href="http://hadoop.apache.org">http://hadoop.apache.org</a>
주요 구성 요소	DataNode	블록(64MB or 128MB 등) 단위로 분할된 대용량 파일들이 DataNode의 디스크에 저장 및 관리	
	NameNode	DataNode에 저장된 파일들의 메타 정보를 메모리상에서 로드 해서 관리	
	EditsLog	파일들의 변경 이력(수정, 삭제 등) 정보가 저장되는 로그 파일	
	FsImage	NameNode의 메모리상에 올라와 있는 메타 정보를 스냅샷 이미지로 만들어 생성한 파일	
Ver. 1.x	SecondaryNameNode	NameNode의 FsImage와 EditsLog 파일을 주기적으로 유지 관리해 주는 체크포인팅 노드	
	MapReduce v1	DataNode에 분산 저장된 파일이 스플릿(Map)되어 다양한 연산(정렬, 그룹핑, 집계 등)을 수행한 뒤 그 결과를 다시 병합(Reduce)하는 분산 프로그래밍 기법	
	JobTracker	맵리듀스의 잡을 실행하면서 태스크에 할당하고, 전체 잡에 대해 리소스 분배 및 스케줄링	
	TaskTracker	JobTracker가 요청한 맵리듀스 프로그램이 실행되는 태스크이며, 이때 맵 태스크와 리듀스 태스크가 생성	

# 빅데이터 적재에 활용하는 기술

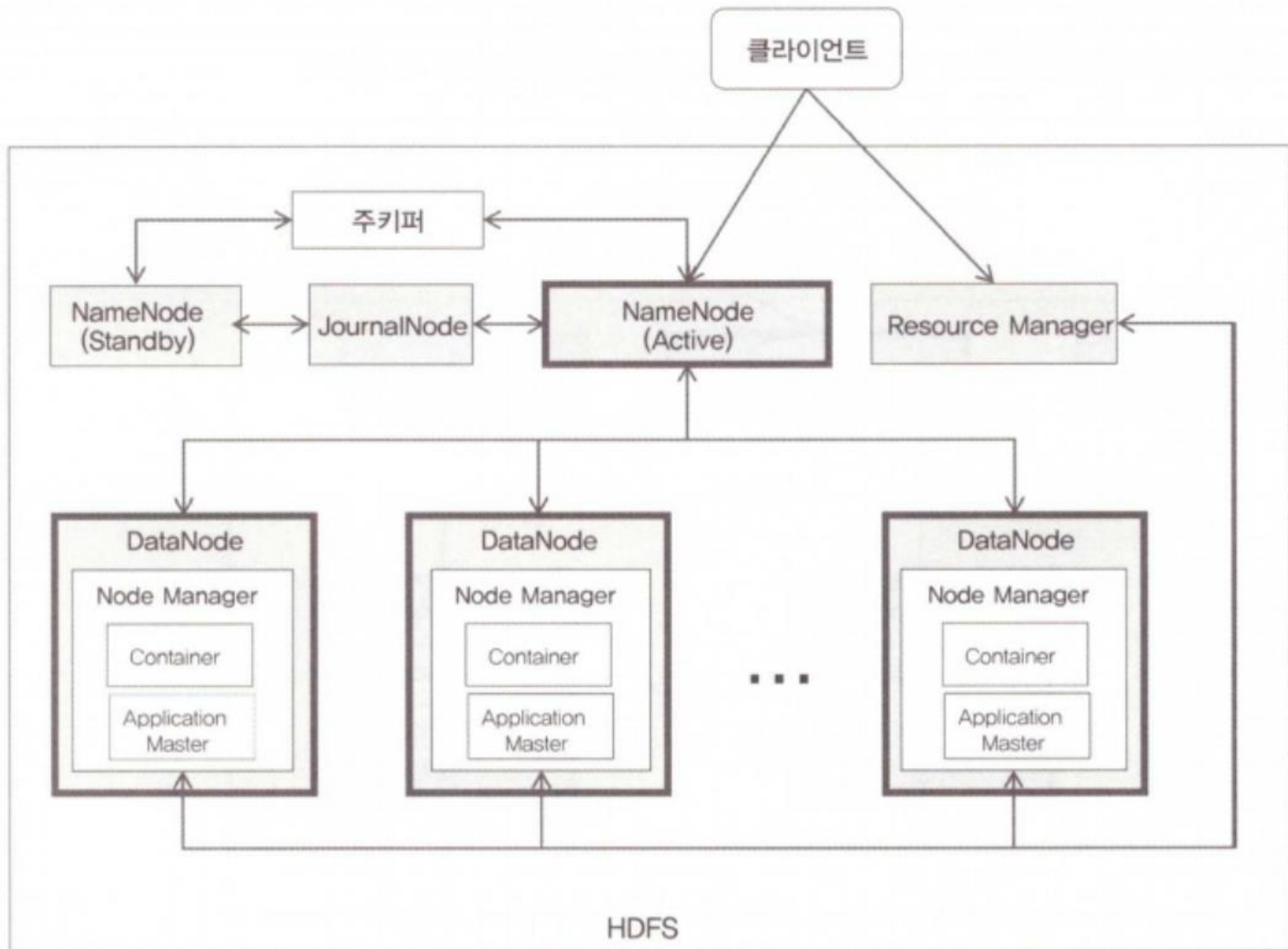
## ➤ 하둡의 기본 요소

주요 구성 요소	Ver. 2.x	Active/Stand-By NameNode	NameNode를 이중화해서 서비스 중인 Active NameNode와 실패 처리를 대비한 Standby NameNode로 구성
		MapReduce v2 / YARN	하둡 클러스터 내의 자원을 중앙 관리하고, 그 위에 다양한 애플리케이션을 실행 및 관리가 가능하도록 확장성과 호환성을 높인 하둡 2.x의 플랫폼
		ResourceManager	하둡 클러스터 내의 자원을 중앙 관리하면서, 작업 요청 시 스케줄링 정책에 따라 자원을 분배해서 실행시키고 모니터링
		NodeManager	하둡 클러스터의 DataNode마다 실행되면서 Container를 실행시키고 라이프 사이클을 관리
		Container	DataNode의 사용 가능한 리소스(CPU, 메모리, 디스크 등)를 Container 단위로 할당해서 구성
		ApplicationMaster	애플리케이션 실행되면 ApplicationMaster가 생성되며 ApplicationMaster는 NodeManager에게 애플리케이션이 실행될 Container를 요청하고, 그 위에서 애플리케이션을 실행 및 관리
		JournalNode	3개 이상의 노드로 구성되어 EditsLog를 각 노드에 복제 관리하며 Active NameNode는 EditsLog에 쓰기만을 수행하고 Standby NameNode는 읽기만을 실행
라이선스	Apache		
유사 프로젝트	GFS(Google File System), Gluster, MogileFS, GridFS, Lustre		

# 하둡 1.x 아키텍처

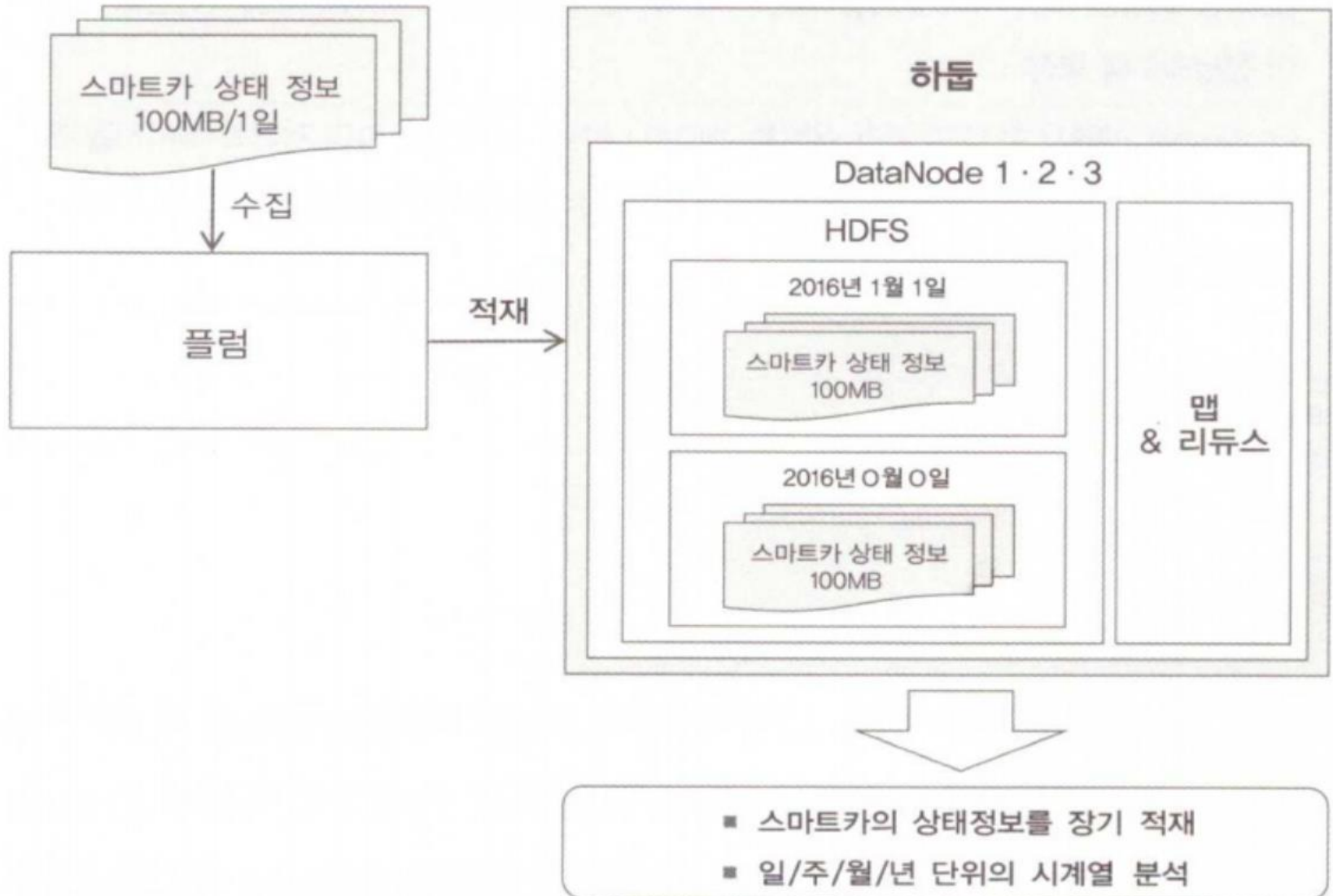


## 하둡 2.x 아키텍처





# 하둡 활용 방안




# 주키퍼

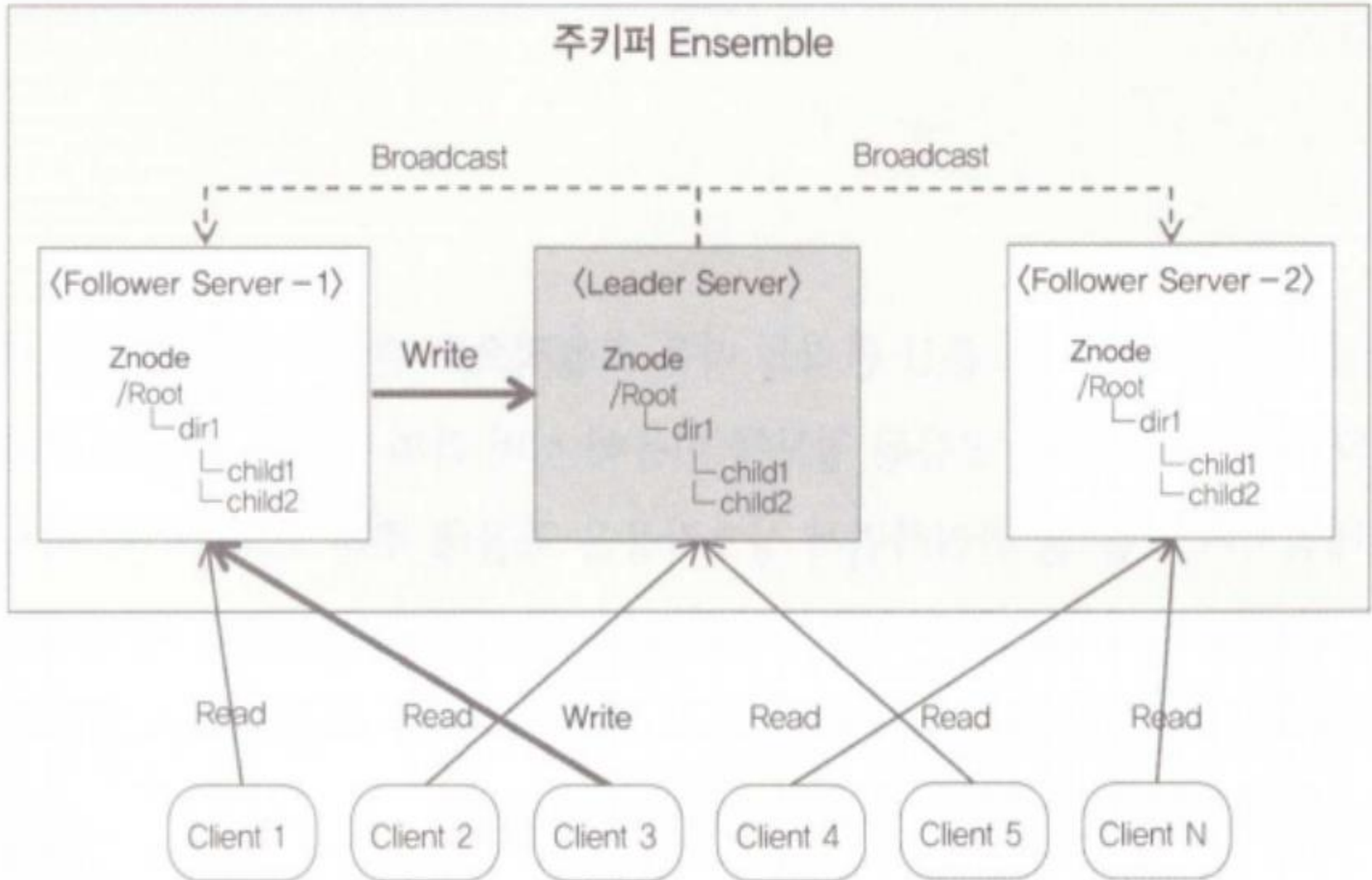
---

- 서버 간의 정보를 쉽고 안전하게 공유할 필요성
  - 수십 ~ 수천 대의 서버에 설치돼 있는 빅데이터 분산 환경을 더욱 효율적으로 관리하기 위함.
- 공유된 정보를 이용해 서버 간의 중요한 이벤트를 관리하면서 상호 작용을 조율해 주는 코디네이터 시스템이 필요.
- 분산 코디네이터.
- 하둡, HBase, 카프카, 스톰 등의 분산 노드 관리에 사용.

# 주키퍼의 기본 요소

공식 홈페이지	 Apache Zookeeper	<a href="http://zookeeper.apache.org">http://zookeeper.apache.org</a>
주요 구성 요소	Client	주키퍼의 ZNode에 담긴 데이터에 대한 쓰기, 읽기, 삭제 등의 작업을 요청하는 클라이언트
	ZNode	주키퍼 서버에 생성되는 파일시스템의 디렉터리 개념으로, 클라이언트의 요청 정보를 계층적으로 관리(버전, 접근 권한, 상태, 모니터링 객체 관리 등의 기능 지원)
	Ensemble	3대 이상의 주키퍼 서버를 하나의 클러스터로 구성한 HA 아키텍처
	Leader Server	Ensemble 안에는 유일한 리더 서버가 선출되어 존재하며, 클라이언트의 요청을 받은 서버는 해당 요청을 리더 서버에 전달하고, 리더 서버는 모든 팔로워 서버에게 클라이언트 요청이 전달되도록 보장
	Follower Server	Ensemble 안에서 한 대의 리더 서버를 제외한 나머지 서버로서, 리더 서버와 메시지를 주고받으면서 ZNode의 데이터를 동기화하고 리더 서버에 문제가 발생할 경우 내부적으로 새로운 리더를 선출하는 역할을 수행
라이선스	Apache	
유사 프로젝트	Chubby, Doozerd, Consul	

# 주키퍼 아키텍처



# 주키퍼 활용 방안

---

- 스마트카 파일럿 프로젝트에서는 주키퍼를 직접적으로 활용하지는 않음.
- 파일럿 프로젝트에서 하둡, HBase, 카프카, 스톰의 내부에서 주키퍼를 활용해 클러스터 멤버십 기능 및 환경 설정의 동기화 등을 사용하고 있어, 없어서는 안 될 중요 소프트웨어.