빅데이터 수집

빅데이터 수집 소개

1. 빅데이터 수집 개요

빅데이터 수집의 개념과 중요성을 설명하고, 일반 수집과의 차이점을 설명한다.

2. 빅데이터 수집에 활용되는 기술

빅데이터 수집에서 사용할 두 가지 기술(플럼, 카프카)을 소개하고 각 기술별 주요 기능과 아키텍처, 활용 방안을 알아본다.

3. 수집 파일럿 실행 1단계 – 수집 아키텍처

스마트카에서 발생하는 로그 파일 수집과 관련된 요구사항을 구체화하고, 수집 요구사항을 해결하기 위한 파일럿 아키텍처를 이해한다.

4. 수집 파일럿 실행 2단계 – 수집 환경 구성

스마트카 로그 파일을 수집하기 위한 아키텍처 설치 및 환경 구성을 진행한다. 플럼, 카프카 순으로 설치를 진행한다.

5. 수집 파일럿 실행 3단계 – 플럼 수집 기능 구현

스마트카 로그 파일을 수집하기 위한 플럼의 환경을 구성하고 관련 에이전트를 생성한다.

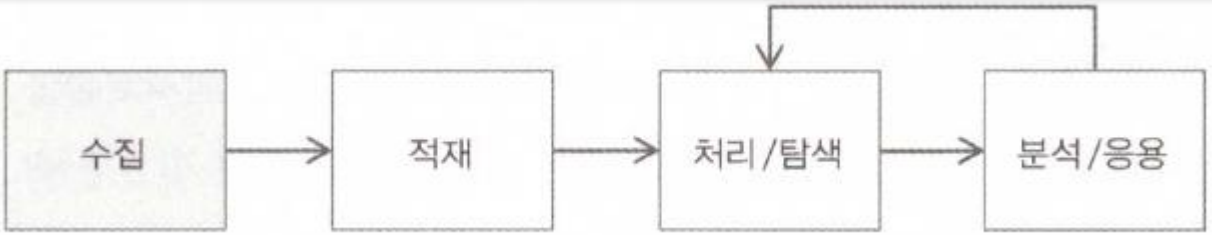
6. 수집 파일럿 실행 4단계 – 카프카 수집 기능 구현

플럼이 수집한 데이터를 카프카 토픽에 전송하는 기능을 구현하고, 카프카의 토픽에 전송된 데이터를 확인하는 방법을 알아본다.

7. 수집 파일럿 실행 5단계 – 수집 기능 테스트

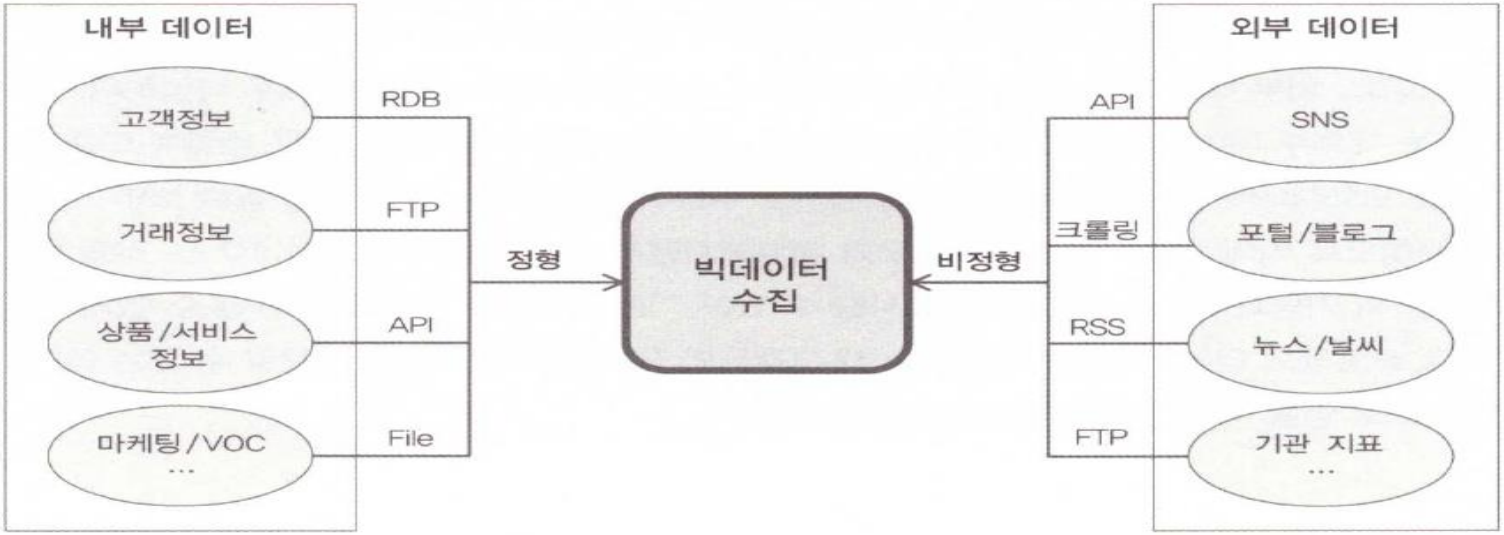
로그 시뮬레이터가 생성한 스마트카의 상태 정보 데이터를 플럼이 수집해서 카프카의 토픽에 전송하는 기능을 점검하고 전송된 데이터를 확인한다.

빅데이터 수집 개요



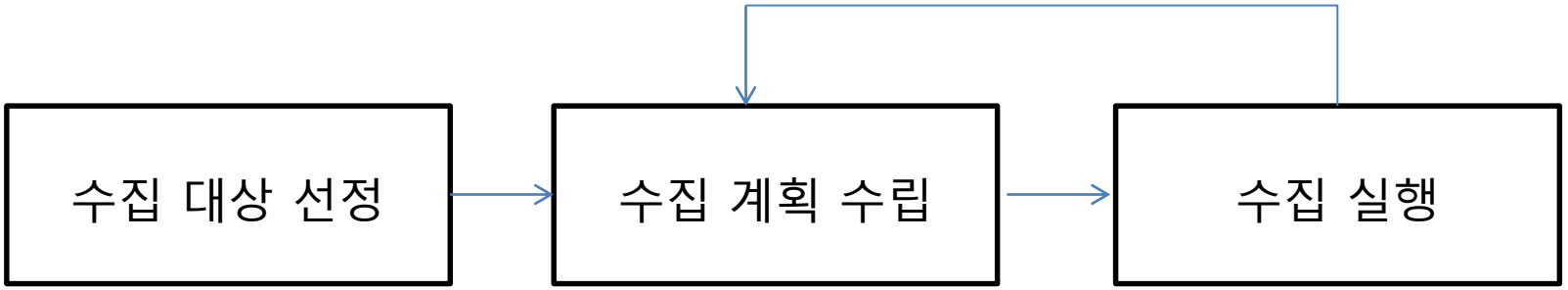
빅데이터 수집 구축 단계

- 빅데이터 시스템 구축은 수집에서부터 시작
- 빅데이터 프로젝트의 여러 공정 단계에서 수집이 전체 공정 과정의 절반 이상을 차지
- 일반적인 수집과의 차이점 : 수집영역이 조직내의 전체시스템에서부터 외부 시스템(SNS, 포털, 정부기관 등)에 이르기까지 매우 광범위하고 다양



빅데이터 수집 개요

➢ 빅데이터 수집 절차



- 수집 도메인 도출
- 수집 데이터셋 도출
- 수집 리스트 작성
- 수집 대상 부서 파악

- 데이터 제공 여부 협의
- 데이터 유형/속성 확인
- 수집 환경 및 표준 파악
- 수집 주기/용량 파악
- 수집 연동/포맷 파악
- 수집 기술 선정
- 수집 정의서/계획서 작성

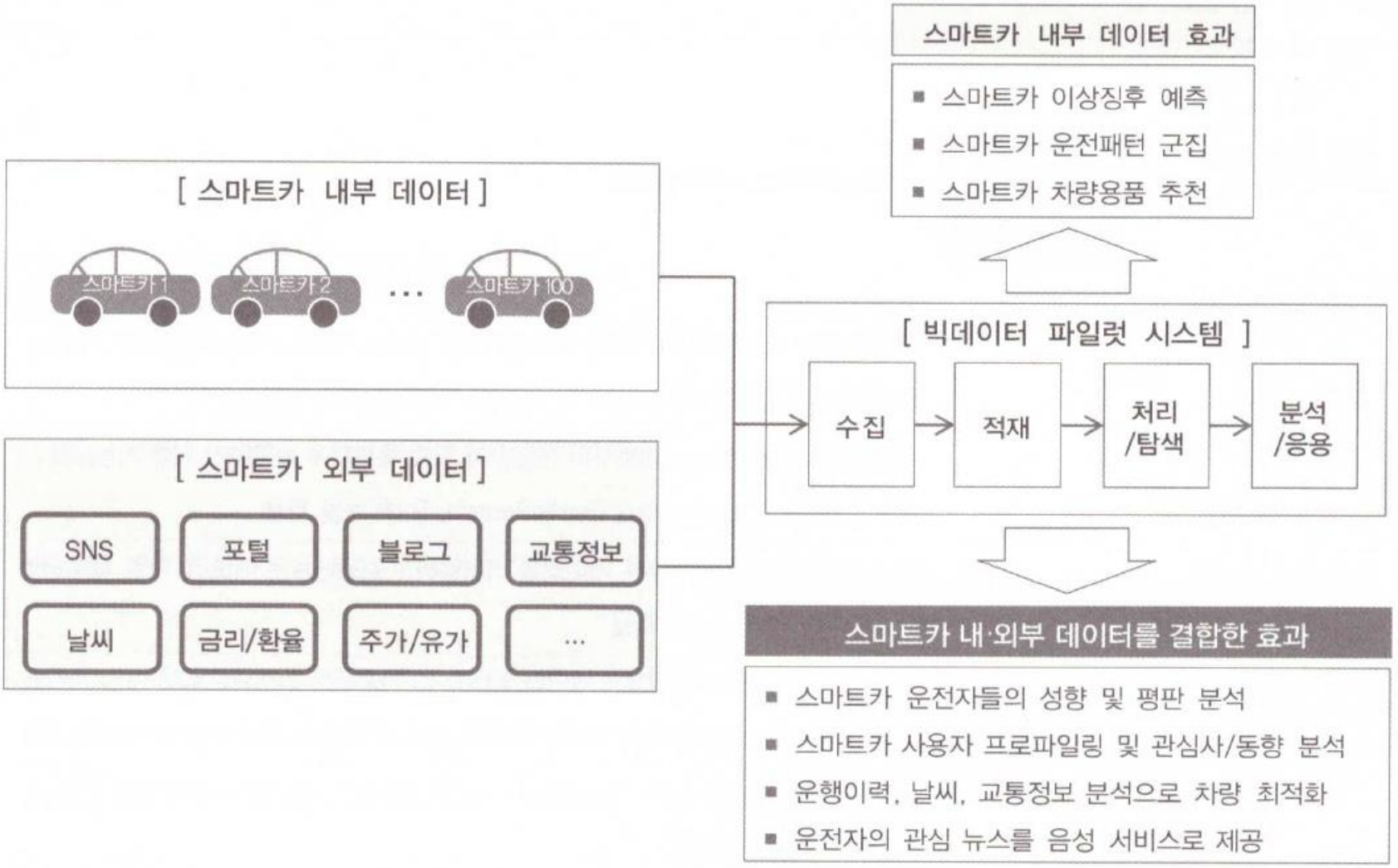
- 단위 테스트 진행
- 연동 테스트 진행
- 데이터 수집 실행
- 데이터 적재 처리

빅데이터 수집 개요

- 빅데이터 수집/적재가 먼저인가? 데이터 분석 영역 도출이 먼저인가?

빅데이터 시스템을 구축할 때 데이터의 수집/적재에 집중해서 시작하는 방법과 분석 영역 도출에 집중해서 시작하는 방법이 있다. 각기 장단점이 있는데, 선수집/후분석의 경우 빅데이터를 우선 쌓아 놓고 지속적인 탐색적 분석으로 마케팅, 상품 개발, 리스크 관리 등의 다양한 관점에서 분석을 시도한다. 이로 인해 조직의 빅데이터 역량을 내재화할 수 있다는 장점도 있으나 우선 수집/적재에 집중하다 보니 불필요한 리소스 낭비가 발생할 수 있고 장시간에 걸쳐 탐색적 분석을 진행해야 하므로 단기간 내에 성과를 달성하기 어려워 타부서의 협조와 경영진의 의사결정을 이루기가 쉽지 않다는 단점이 있다. 선분석/후수집의 경우 조직에서 당장 필요로 하는 분석 주제를 확정 짓고 진행하므로 단기간에 최소 비용으로 성과를 낼 수 있는 장점이 있다. 하지만 빅데이터 분석이 단기적인 노력만으로 좋은 성과를 만드는 케이스가 드물어, 실패 시 빅데이터에 대한 부정적인 인식이 사업 초기에 만들어질 수 있다. 또한 외부 파트너에 대한 기술적 의존도가 높아지고 역량 내재화가 어려워져 결국 사업 확장 시 추가 비용이 크게 발생할 수 있다. 필자의 개인적인 생각으로는 정답은 없다. 빅데이터를 도입하는 주체가 기술 중심의 IT 부서라면 전자가 용이하고, 업무 중심의 BM 부서라면 후자가 더 용이하다. 이보다 더 중요한 것은 빅데이터에 대한 조직의 활용 목적이 뚜렸하고, 빅데이터에 대한 공감대가 만들어져 있다면 어떠한 접근법도 문제 되지 않으리라는 점이다.

스마트카 내/외부 데이터를 결합한 효과



빅데이터 수집에 활용할 기술 – 플럼(Flume) 소개

- 인공수로, 용수로 등의 사전적 의미를 가짐 – 여러 서비스 제공 서버에 산재해 있는 로그들을 하나의 로그 수집 서버로 모으는 역할을 수행해야 하는 수집기로서 어울리는 이름.
- 다양한 수집 요구 사항들을 해결하기 위한 기능으로 구성된 소프트웨어.
- 다양한 데이터 원천(통신 프로토콜, 메시지 포맷, 발생 주기, 데이터 크기 등)으로 부터 수집 시, 고민을 쉽게 해결할 수 있는 기능과 아키텍처를 제공.
- 스트림 지향의 데이터 플로우를 기반으로 하며 지정된 모든 서버로부터 로그를 수집한 후 하둡 HDFS와 같은 중앙 저장소에 적재하여 분석하는 시스템을 구축해야 할 때 적합.
- 2011년 클라우데라를 통해 처음 소개 – 현재 아파치 프로젝트에 기증되어 최상위 프로젝트로 승격되어 사용 중.
- 0.x 버전은 Flume-OG로 불리며 Agent, Collector, Master로 구분되던 아키텍쳐 구조가, 1.x 이후 버전부터 Flume-NG(Next Generation)로 이름이 바뀌면서 아키텍처가 크게 바뀜.

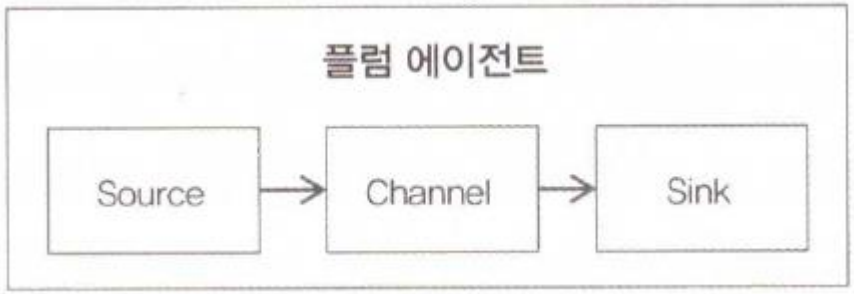
플럼 설계 기준

- 시스템 신뢰성 (Reliability) - 장애가 발생하더라도 로그, 이벤트의 유실 없이 전송할 수 있는 기능
- 시스템 확장성 (Scalability) - Agent의 추가 및 제거가 용이
- 관리 용이성 (Manageability) - 간결한 구조로 관리가 용이
- 기능 확장성 (Extensibility) - 새로운 기능을 쉽게 추가할 수 있음

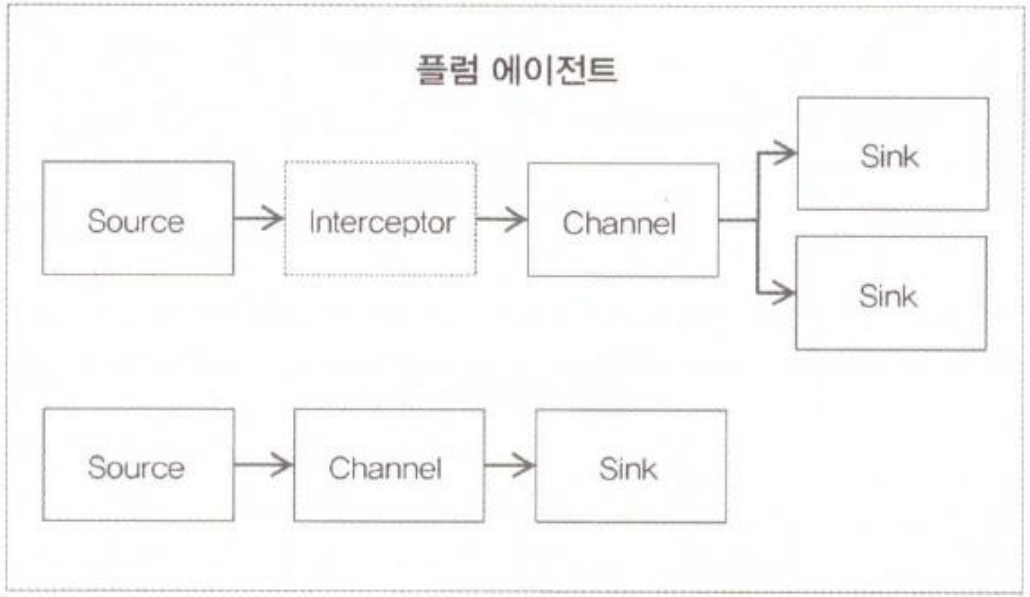
빅데이터 수집에 활용할 기술 – 플럼 기본 요소

공식 홈페이지		http://flume.apache.org
주요 구성 요소	Source	다양한 원천 시스템의 데이터를 수집하기 위해 Avro, Thrift, JMS, Spool Dir, Kafka 등 여러 주요 컴포넌트를 제공하며, 수집한 데이터를 Channel로 전달
	Sink	수집한 데이터를 Channel로부터 전달받아 최종 목적지에 저장하기 위한 기능으로 HDFS, Hive, Logger, Avro, ElasticSearch, Thrift 등을 제공
	Channel	Source와 Sink를 연결하며, 데이터를 버퍼링하는 컴포넌트로 메모리, 파일, 데이터베이스를 채널의 저장소로 활용
	Interceptor	Source와 Channel 사이에서 데이터 필터링 및 가공하는 컴포넌트로서 Timestamp, Host, Regex Filtering 등을 기본 제공하며, 필요 시 사용자 정의 Interceptor를 추가
	Agent	Source → (Interceptor) → Channel → Sink 컴포넌트 순으로 구성된 작업 단위로 독립된 인스턴스로 생성
라이선스	Apache 2.0	
유사 프로젝트	Fluented, Scribe, logstash, Chukwa 등	

플럼 아키텍처

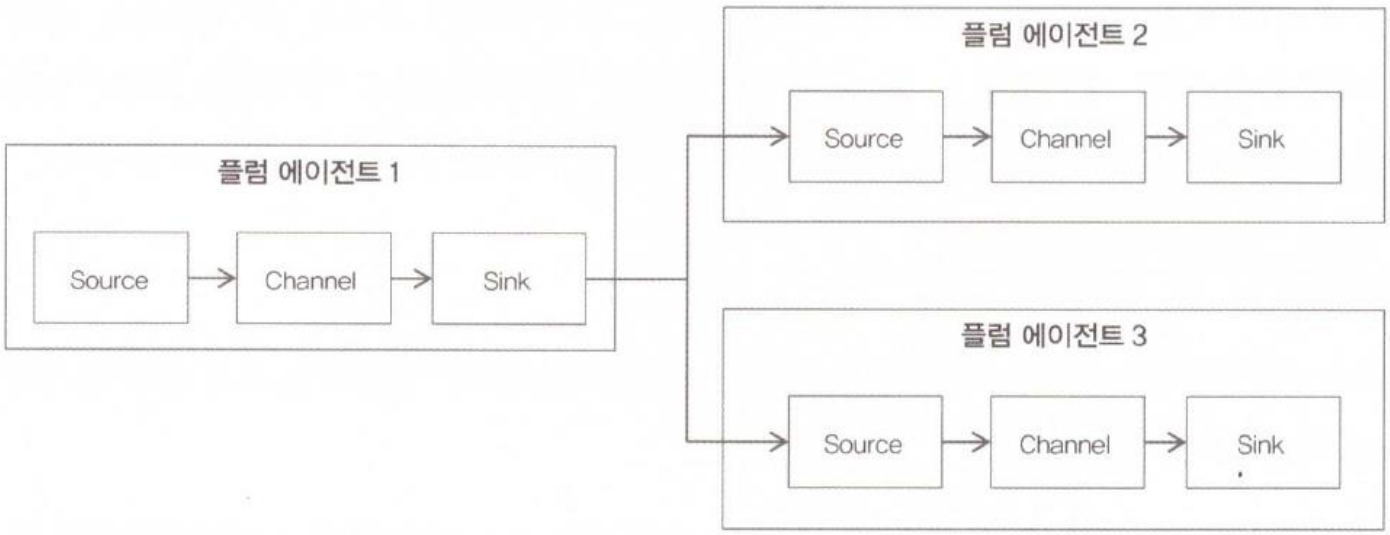


- 가장 단순한 플럼 에이전트 구성
- 원천 데이터를 특별한 요구 사항 없이 단순 수집/적재할 때 주로 활용



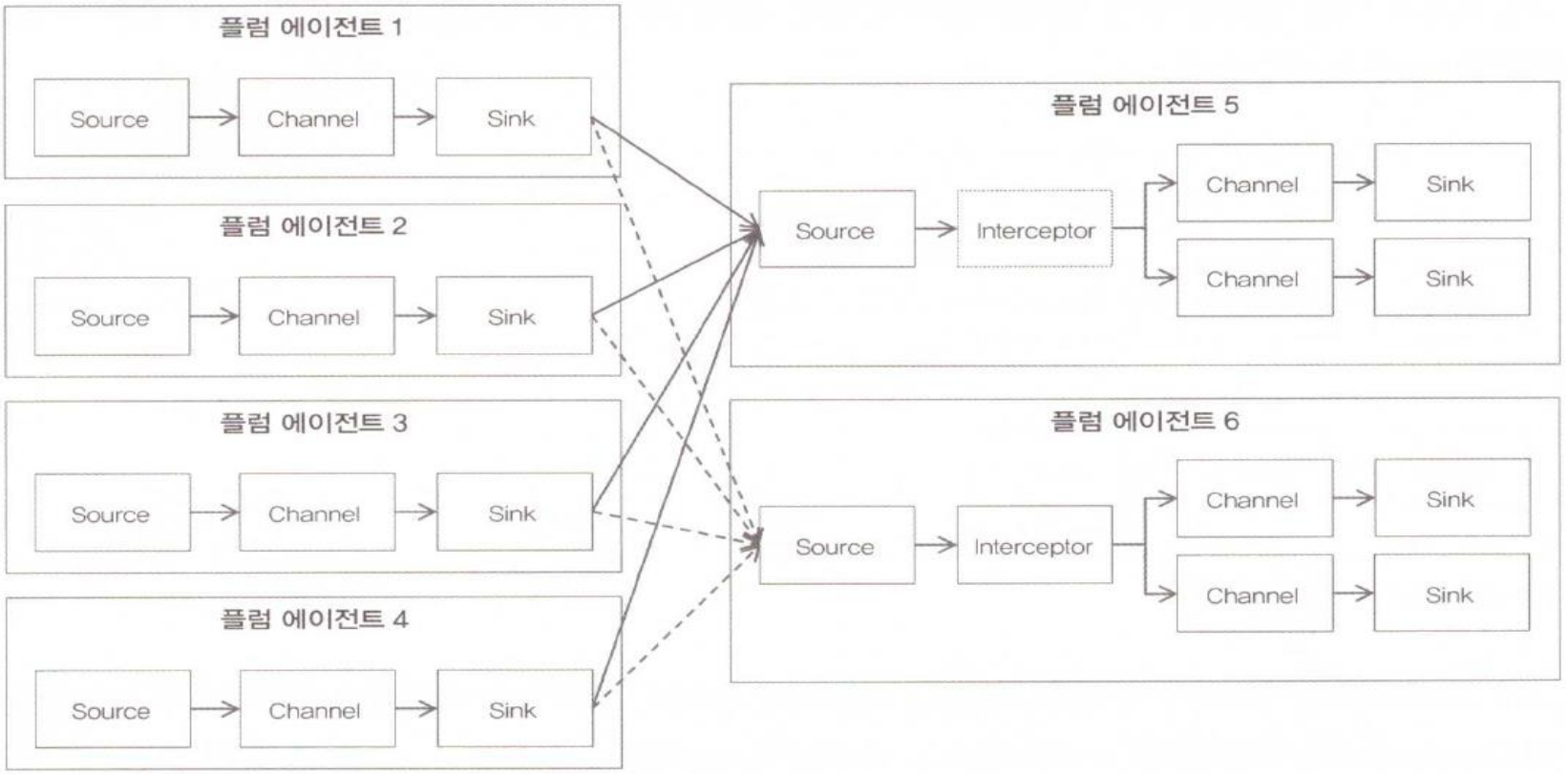
- 원천 데이터 수집 시 Interceptor를 추가해 데이터를 가공하고, 데이터의 특성에 따라 channel에서 다수의 Sink 컴포넌트로 라우팅이 필요할 때 구성
- 한 개의 플럼 에이전트 안에서 두 개 이상의 Source-Channel-Sink 컴포넌트 구성 및 관리 가능

플럼 아키텍처



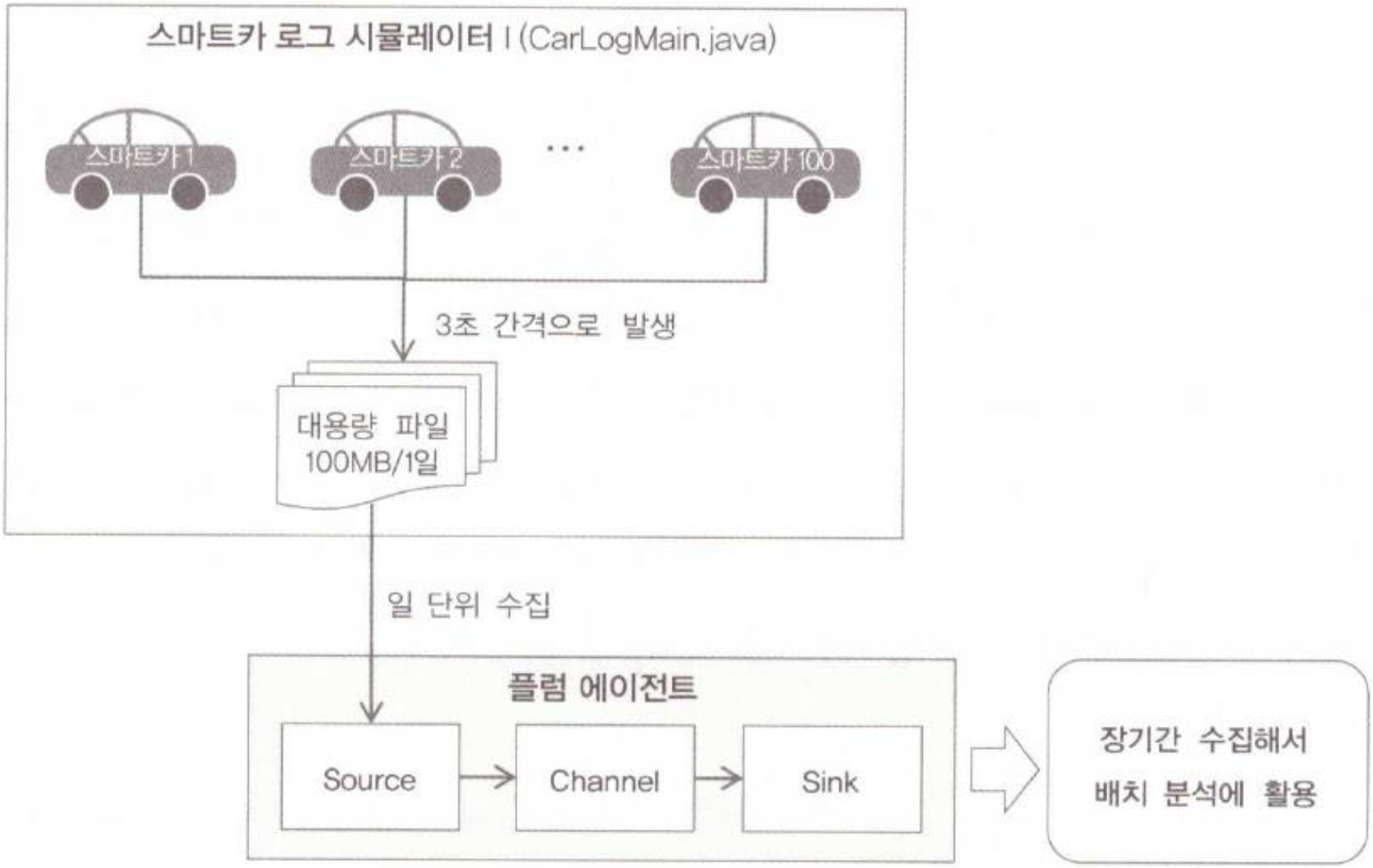
- 수집해야 할 원천 시스템은 한 곳이지만 높은 성능과 안정성이 필요할 때 주로 사용되는 아키텍처

플럼 아키텍처

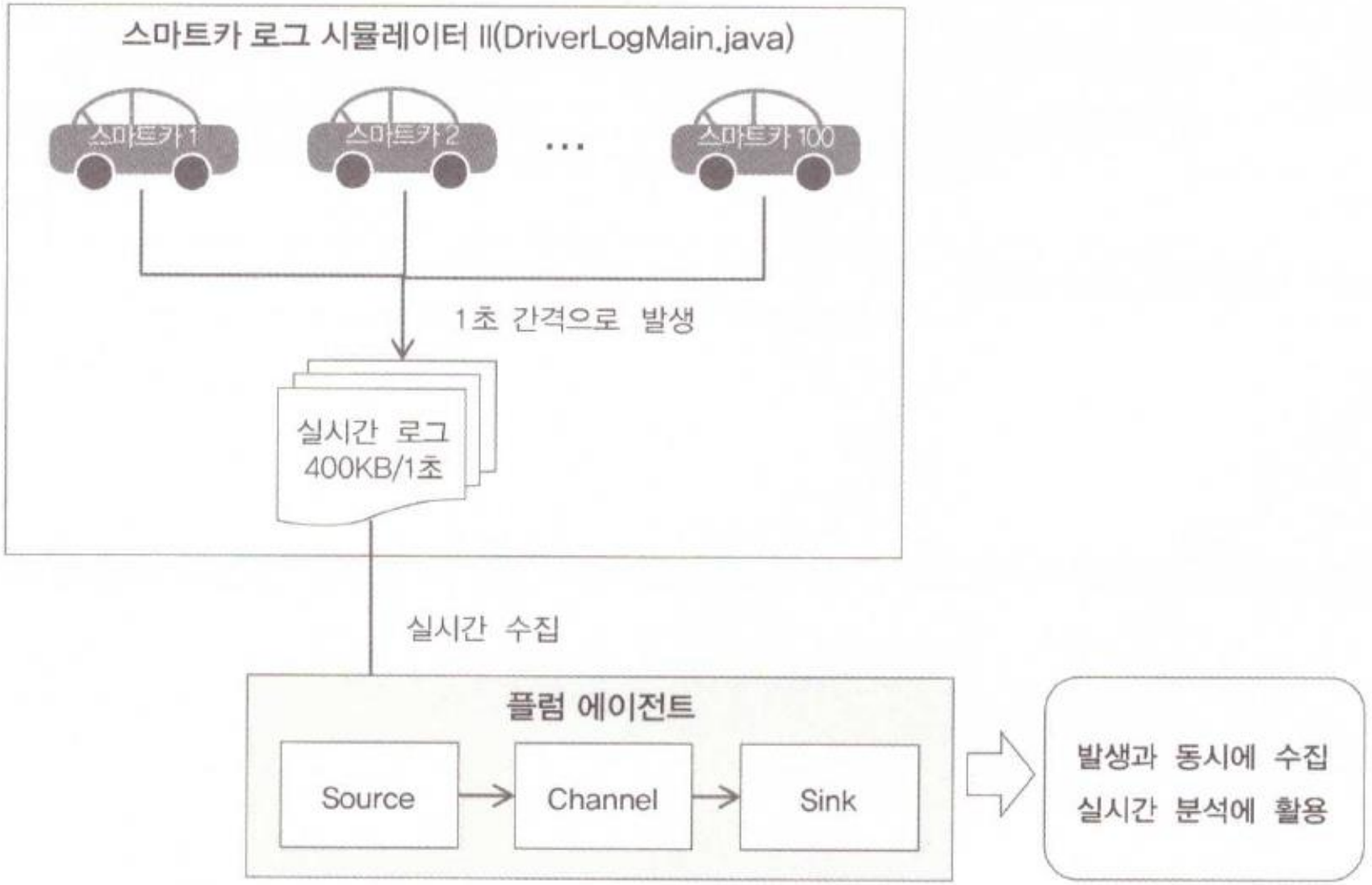


- 수집해야 할 원천 시스템이 다양하고 대규모의 데이터가 유입될 때 사용하는 플럼의 분산 아키텍처
- 플럼 에이전트 1/2/3/4에서 수집한 데이터를 플럼 에이전트 5에서 집계(aggregation)하고, 플럼 에이전트 6으로 이중화해서 성능과 안정성을 보장하는 구성

파일럿 프로젝트에서의 플럼 활용 방안1



파일럿 프로젝트에서의 플럼 활용 방안2



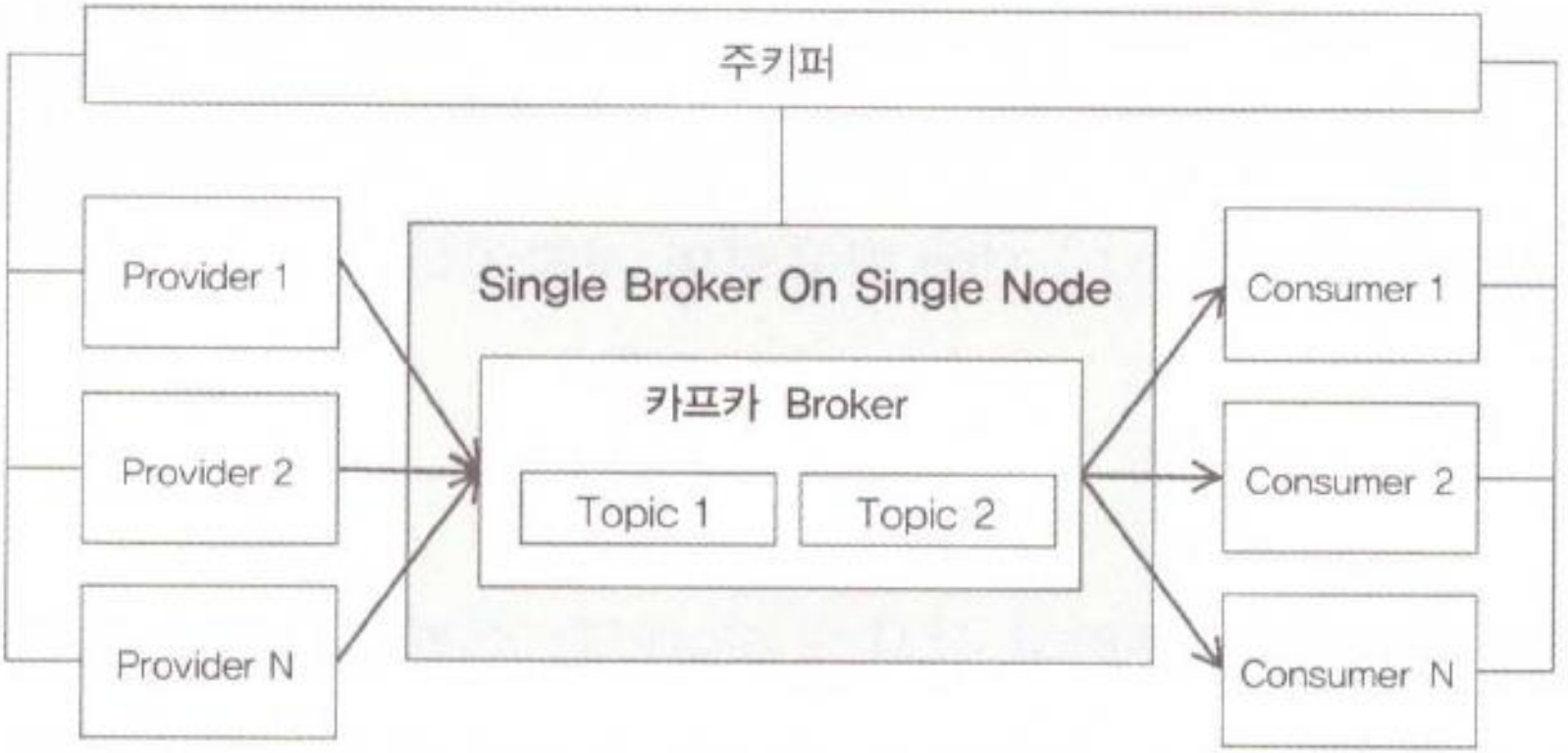
빅데이터 수집에 활용할 기술 – 카프카(Kafka) 소개

- MOM(Message Oriented Middleware) 소프트웨어 중 하나 – 대규모로 발생하는 메시지성 데이터를 비동기 방식으로 중계하는 역할.
- 원천 시스템으로부터 대규모 트랜잭션 데이터가 발생했을 때 중간에 데이터를 버퍼링하면서 타깃 시스템에 안정적으로 전송해 주는 강력한 기능과 아키텍처를 제공하는 중간 시스템.
- 링크드인에서 2011년에 개발되어 그 해 6월에 아파치 인큐베이터에 등록됐고, 2012년 10월에 아파치 최상위 프로젝트로 승격.
- 최근에는 스파크와 연동해서 스트리밍 방식의 데이터 처리 플로우를 구축하는 데 널리 활용.
- 여러 대의 서버에 브로커(Broker)라는 카프카 프로세스를 띄워두고 데이터를 가지고 있는 프로듀서(Producer) 프로세스들이 데이터를 브로커에게 전달하면 컨슈머(Consumer) 프로세스들이 브로커를 통해 데이터를 읽어가는 구조.
- 확장성이 높고 메모리 대신 파일시스템을 이용하면서도 빠르고 안정적인 서비스를 제공.

빅데이터 수집에 활용할 기술 – 카프카 기본 요소

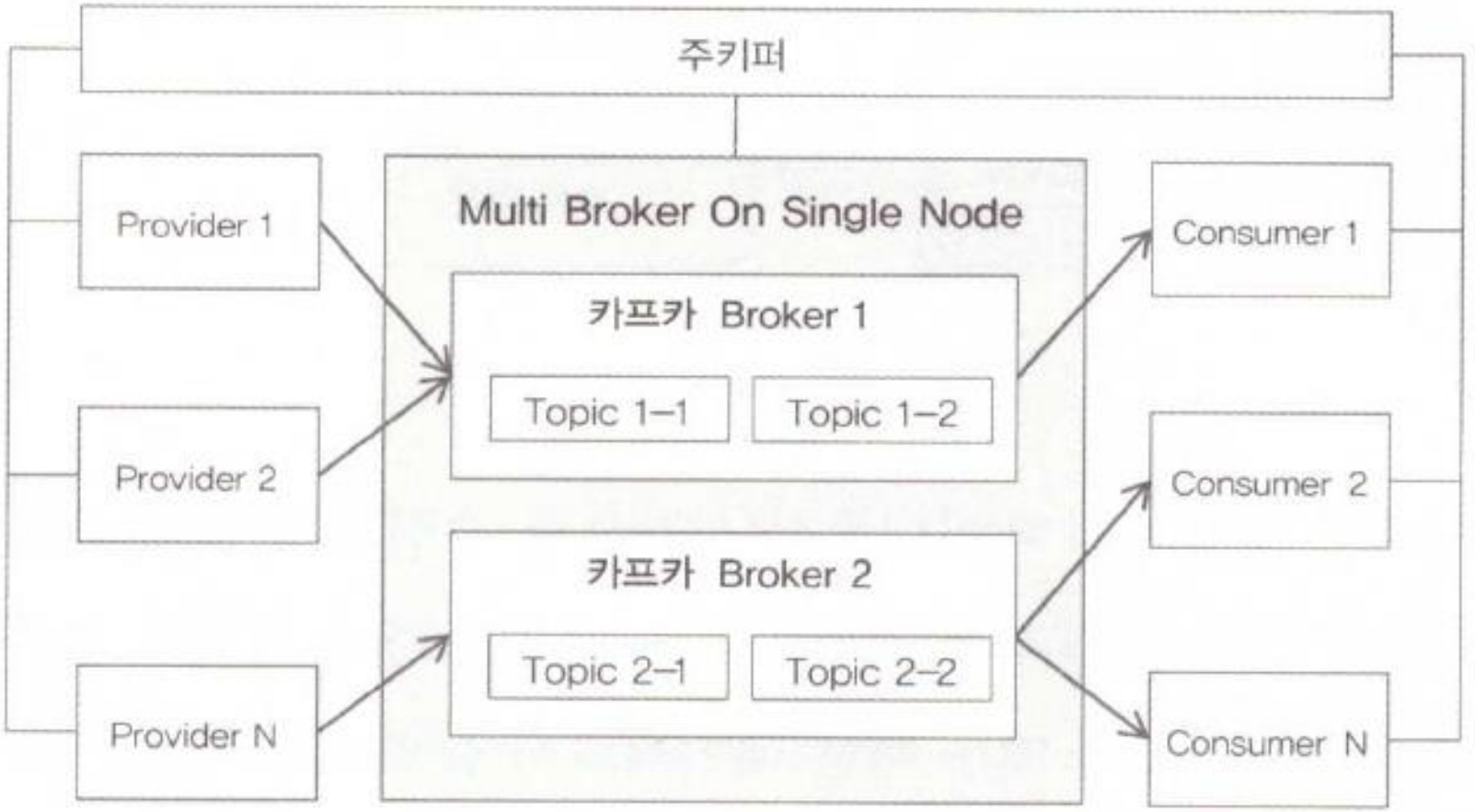
공식 홈페이지	Apache Kafka http://kafka.apache.org	
주요 구성 요소	Broker	카프카의 서비스 인스턴스로서, 다수의 Broker를 클러스터로 구성하고 Topic이 생성되는 물리적 서버
	Topic	Broker에서 데이터의 발행/소비 처리를 위한 저장소
주요 구성 요소	Provider	Broker의 특정 Topic에 데이터를 전송(발행)하는 역할로서 애플리케이션에서 카프카 라이브러리를 이용해 구현
	Consumer	Broker의 특정 Topic에서 데이터를 수신(소비)하는 역할로서 애플리케이션에서 카프카 라이브러리를 이용해 구현
라이선스	Apache	
유사 프로젝트	ActiveMQ, RabbitMQ, HonnetQ 등	

카프카 아키텍처 유형 – 싱글 브로커/싱글 노드



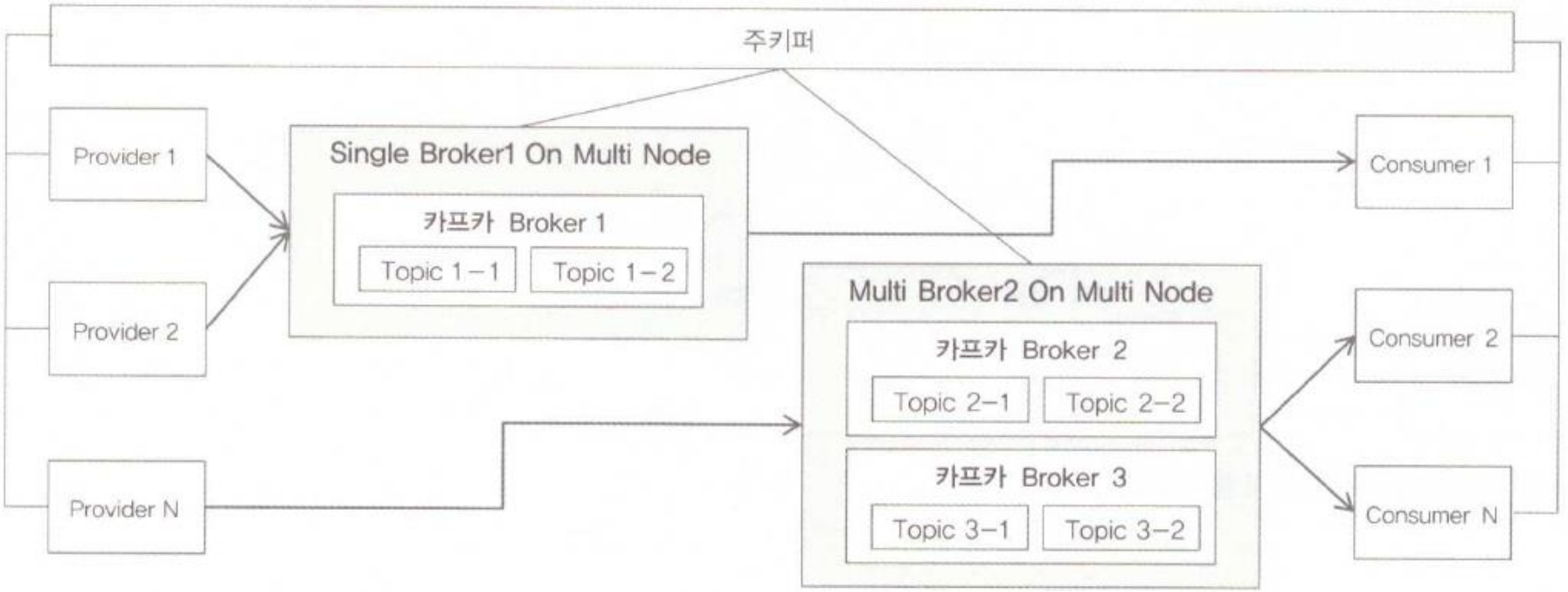
- 1대의 카프카 서버만 설치하고, 1개의 Broker만 구성한 아키텍처
- 대량의 발행/소비 요건이 없고, 단순한 업무 도메인에 이용

카프카 아키텍처 유형 – 멀티 브로커/싱글 노드



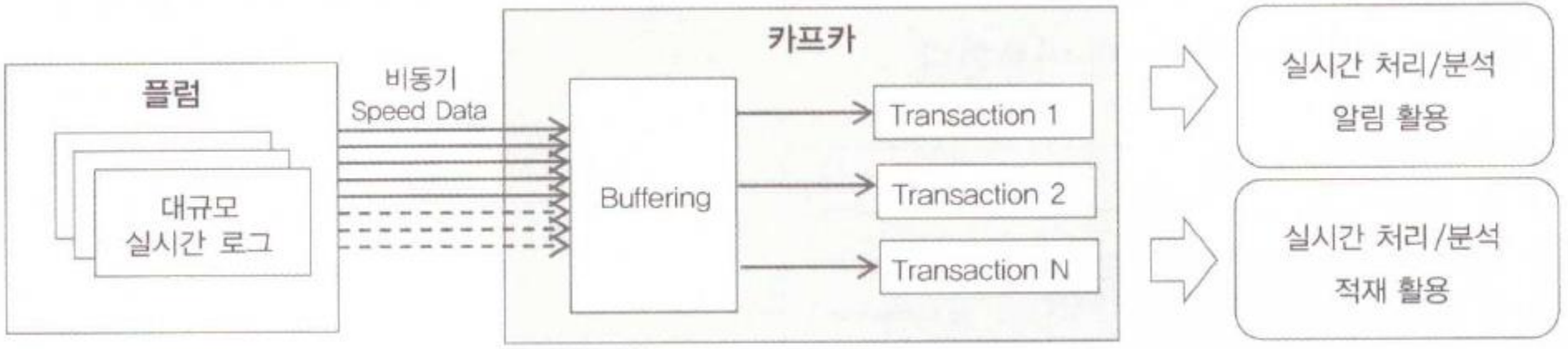
- 1대의 카프카 서버에 2개의 Broker를 구성한 아키텍처
- 대량의 발행/소비 요건에는 사용하기 어렵지만 업무 도메인이 복잡해서 메시지 처리를 분리 관리해야 할 때 이용

카프카 아키텍처 유형 – 멀티 브로커/멀티 노드



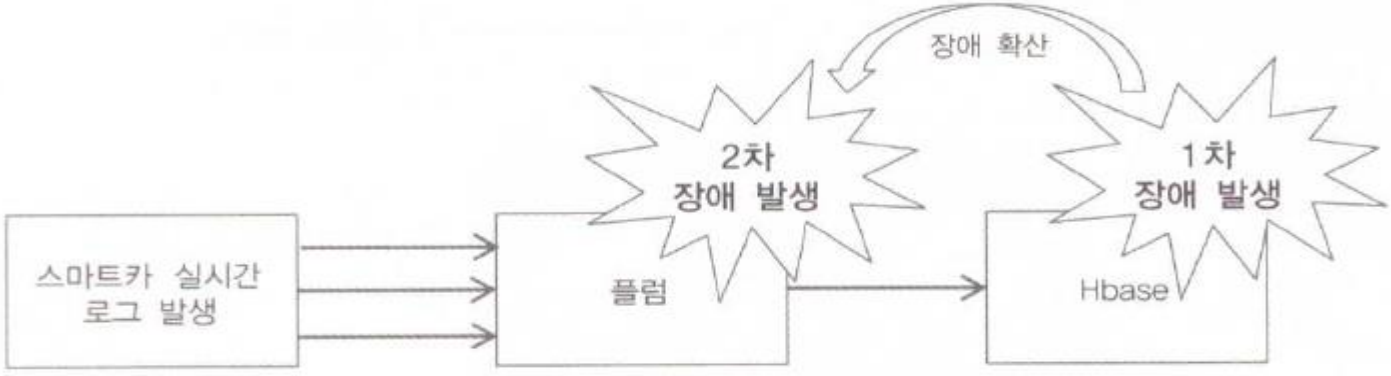
- 2대 이상의 카프카 서버로 멀티 Broker를 구성한 아키텍처
- 대량의 발행/소비 데이터 처리에 적합하며, 물리적으로 나눠진 브로커 간의 데이터 복제가 가능해 안정성이 높다.
- 업무 도메인별 메시지 그룹을 분류할 수 있어 복잡한 메시지 송/수신에 활용하면 적합

카프카 활용 방안 1

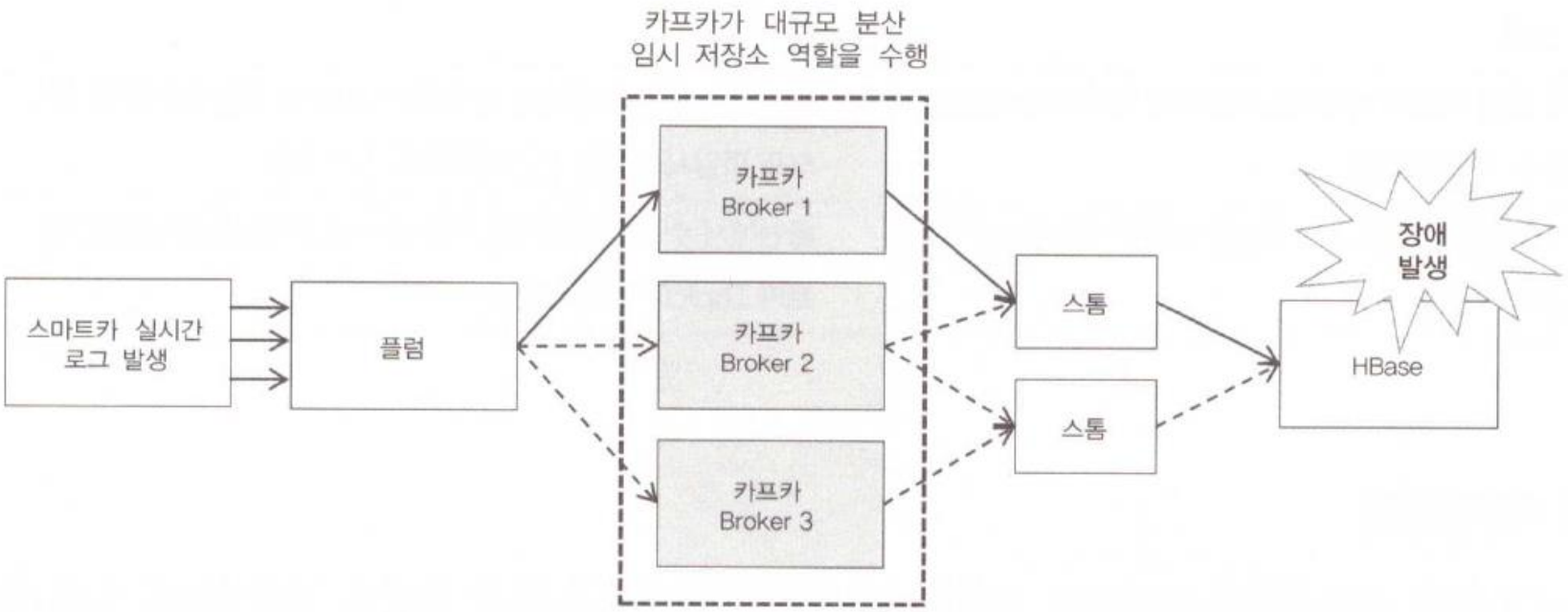


- 역할: 플럼이 실시간 데이터를 수집해 카프카 토픽에 전송하면 카프카는 전송받은 데이터를 토픽에 임시로 저장하고 있다가 컨슈머 프로그램이 작동해 토픽에서 데이터를 가져감.
- 목적 : 플럼이 아주 빠르게 발생하는 데이터를 실시간으로 수집하게 되면 이를 최종 목적지에 전달하기 전 중간에서 안정적인 버퍼링 처리가 필요하기 때문.

- 플럼이 수집한 데이터를 카프카를 거치지 않고 곧바로 타깃 저장소인 Hbase에 전송할 때 문제점



카프카 활용 방안 2



- 카프카를 대규모 분산 환경으로 구성
- HBase 장애가 발생해도 카프카에서 데이터를 저장해 놓았다가 HBase가 복귀되면 곧바로 재처리가 가능
- 플럼이 수집한 데이터를 카프카의 토픽에 비동기로 전송함으로써 수집 속도가 빨라지는 장점