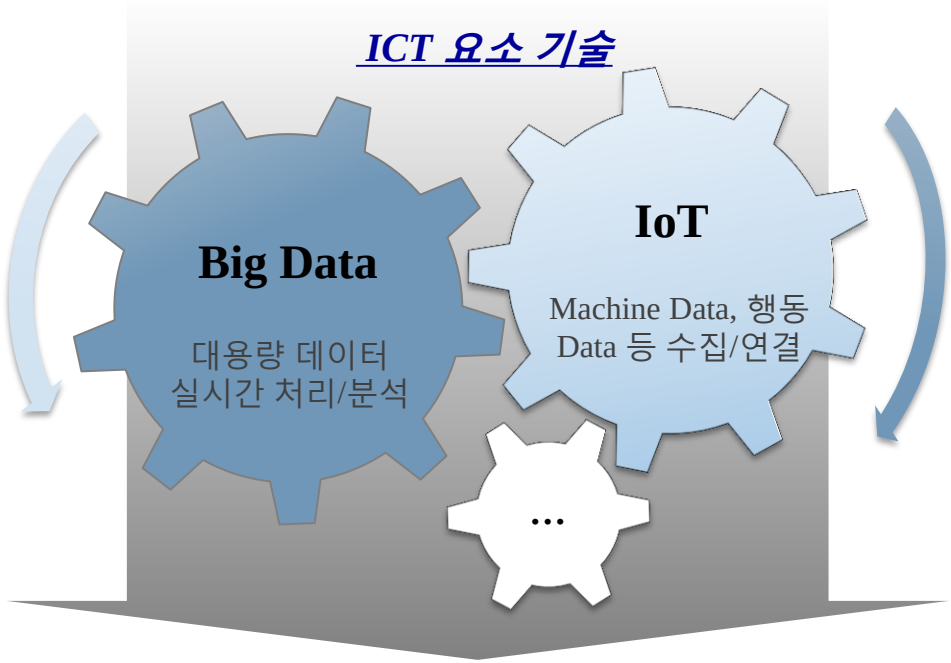
실시간 분산병렬 CEP 플랫폼

2015. 10

목차

【 배경 】

Big Data/IoT를 활용한 IT서비스 Value-up



Global ICT B2B Platform 사업자

필요성

1 Big Data / IoT 기반 기술 패키징

- Big Data/IoT 기술 및 노하우를 자산화하여 패키징 필요
- Big Data/IoT 영역에서의 다양한 Biz 요구에 신속한 반응을 하기 위함

2 Platform 기반 사업 실행

- Platform 기반 사업실행 생태계 조성을 통한 지속적 사업 영역 확대
- B2B 중심 Platform 사업 기회 발굴 이후 서비스 모델 확대 및 Global 진출

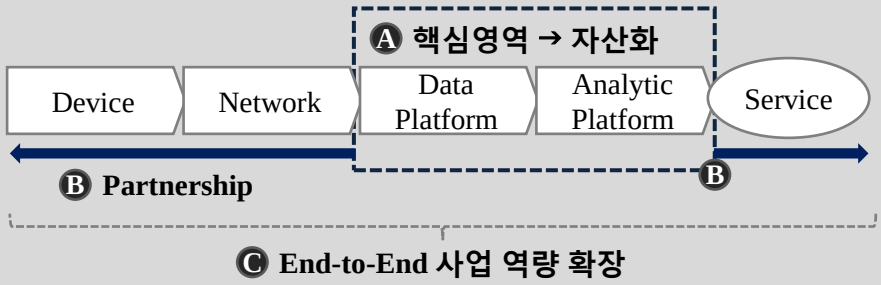
Big Data / IoT 기반기술 및 노하우를 자산화하고 영역별 전문기업들과 Partnership을 구축하고 협력업체와 상생을 통한 확보 추구

1 **Big Data/IoT** 기반 기술 패키징

2 **Platform 기반 사업실행**

Big Data/IoT Eco-system 확보 방안

Big Data/IoT Eco-System



Ⓐ 핵심기술 자산화 (Data 수집/처리 및 Analytic Engine 등)

- Big Data/IoT Core 영역 자사 Solution 확보

Ⓑ 영역별 전문기업 Partnership

- Global ICT 기업 및 영역별 전문 Player 협업 생태계 마련

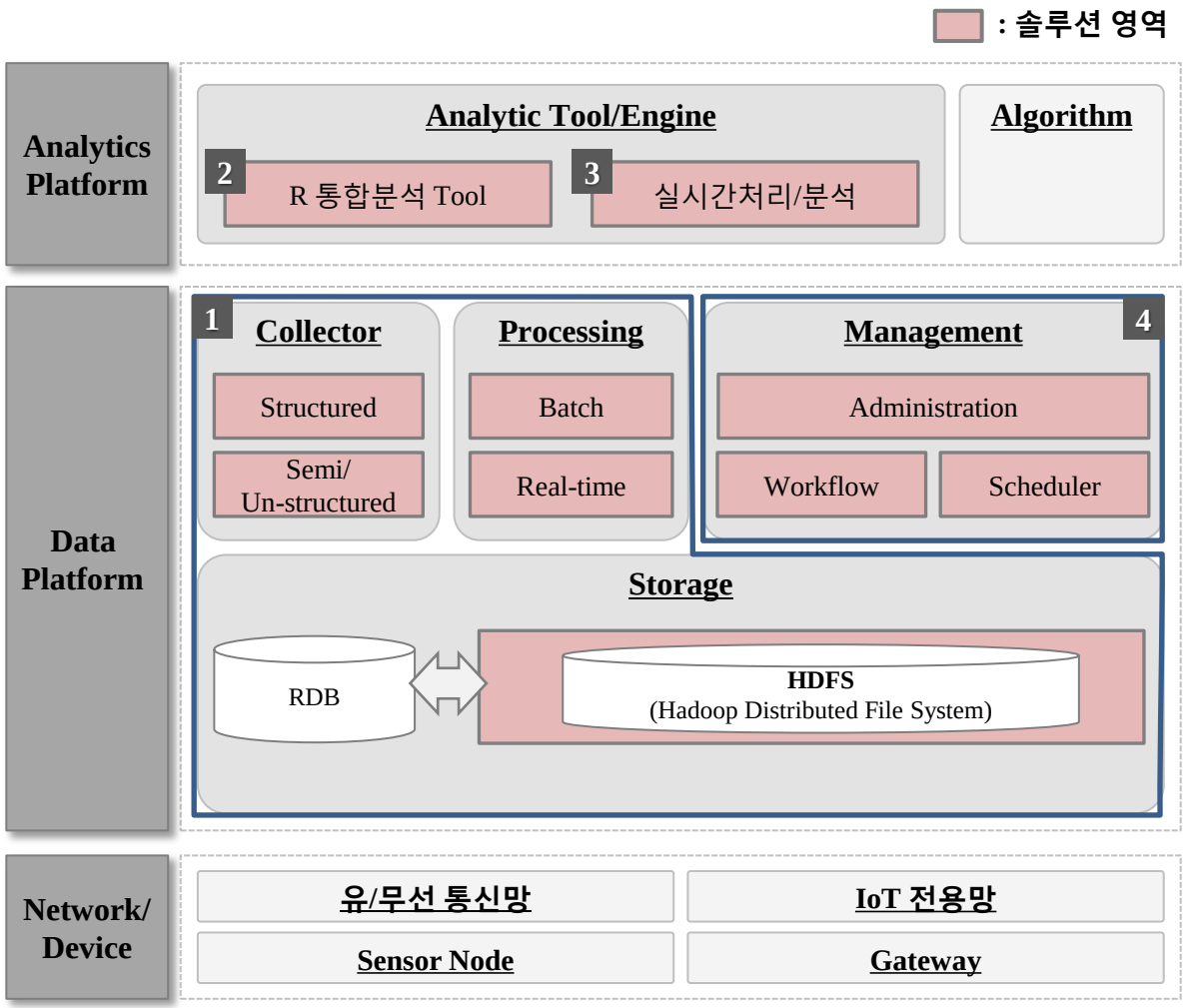
Ⓒ 협력업체와 상생

- 솔루션 개발 유지 보수 Co-Work

Timely/Speedy 한 사업 수행

Big Data 프로젝트를 통해 旣 확보된 R&C를 Leverage하여 핵심기술 4개에 대한 솔루션/자산化를 진행함.

【 Big Data 솔루션 영역 】



주요 내용

1 데이터 수집/저장/처리

- Big Data 기술 적용에 필수적으로 필요한 Hadoop 기반 인프라
- Storage 및 Open source 표준 Platform

2 R 통합 분석 Tool

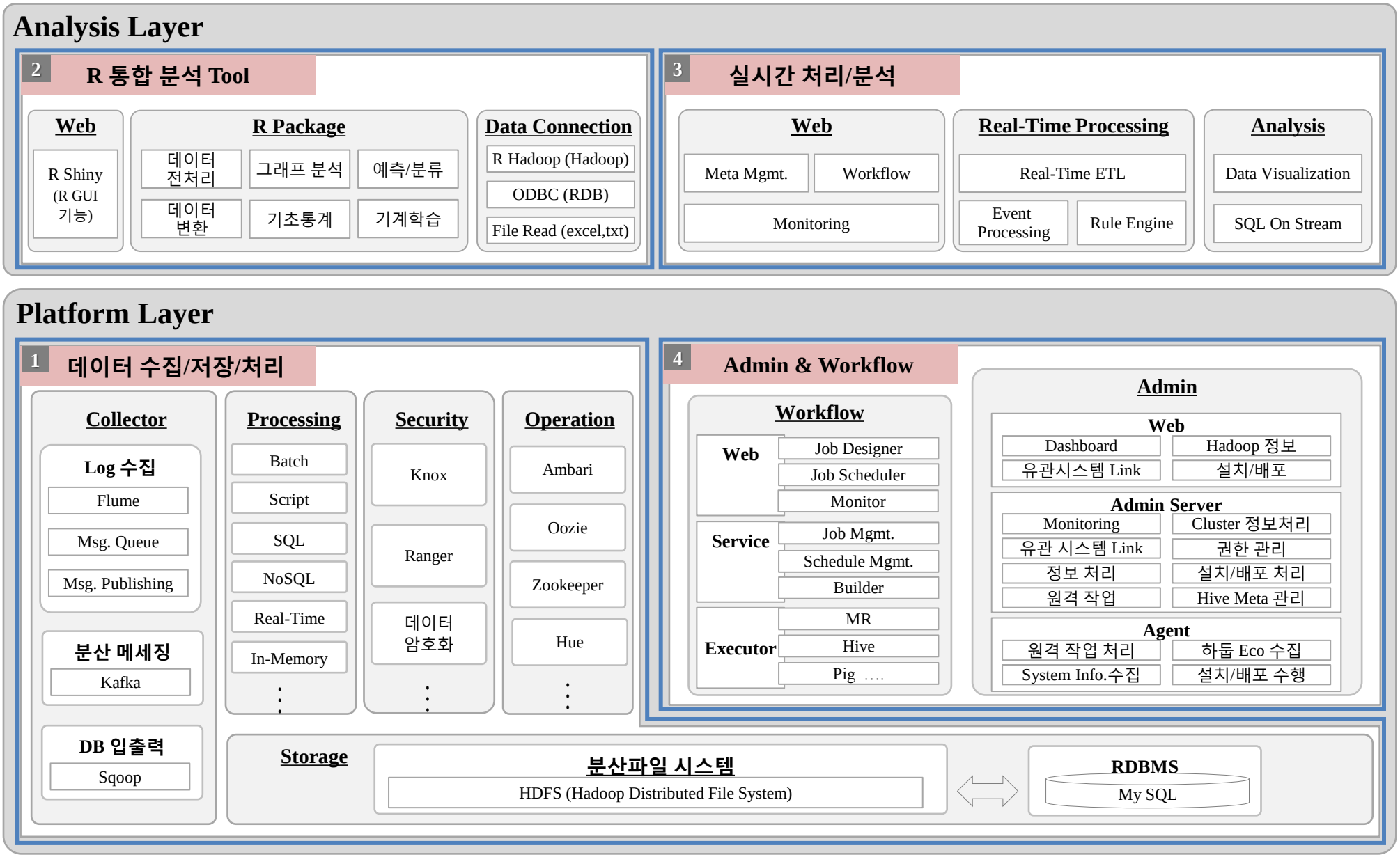
- 통계/마이닝/머신러닝 등 알고리즘을 사용자 Biz.에 맞게 제공하는 분석/시각화 Tool

3 실시간 처리/분석

- 전 산업에서 요구가 증대되고 있는 Event 기반 실시간 데이터 처리/분석 엔진
- 실시간 Streaming, Event Rule 엔진 등

4 Admin & Workflow

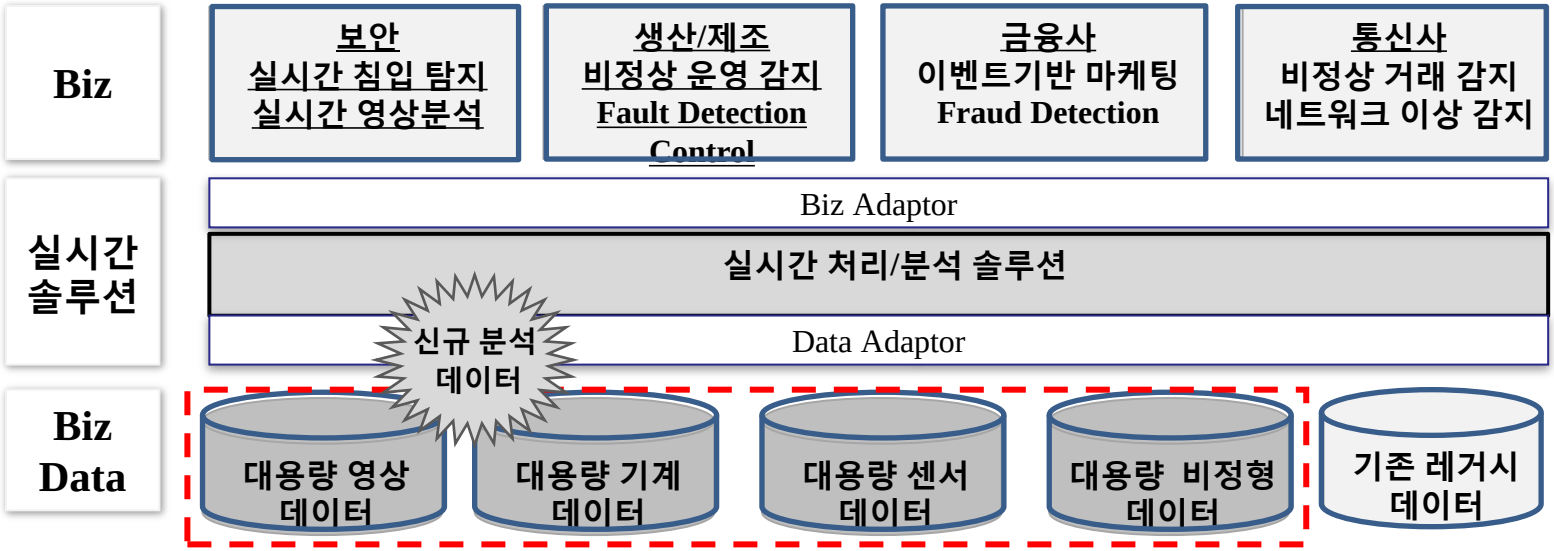
- 구축된 Big Data의 모니터링, 설치/배포
- Big Data 프로세스 Workflow /Designer 및 Job 스케줄러


Analysis Layer
2 R 통합 분석 Tool
Web
R Shiny (R GUI 기능)
R Package
데이터 전처리
그래프 분석
예측/분류
데이터 변환
기초통계
기계학습
Data Connection
R Hadoop (Hadoop)
ODBC (RDB)
File Read (excel,txt)
3 실시간 처리/분석
Web
Meta Mgmt.
Workflow
Monitoring
Real-Time Processing
Real-Time ETL
Event Processing
Rule Engine
Analysis
Data Visualization
SQL On Stream
Platform Layer
1 데이터 수집/저장/처리
Collector
Log 수집
Flume
Msg. Queue
Msg. Publishing
분산 메세징
Kafka
DB 입출력
Sqoop
Processing
Batch
Script
SQL
NoSQL
Real-Time
In-Memory
Security
Knox
Ranger
데이터 암호화
Operation
Ambari
Oozie
Zookeeper
Hue
4 Admin & Workflow
Workflow
Web
Job Designer
Job Scheduler
Monitor
Service
Job Mgmt.
Schedule Mgmt.
Builder
Executor
MR
Hive
Pig ….
Admin
Web
Dashboard
Hadoop 정보
유관시스템 Link
설치/배포
Admin Server
Monitoring
Cluster 정보처리
유관 시스템 Link
권한 관리
정보 처리
설치/배포 처리
원격 작업
Hive Meta 관리
Agent
원격 작업 처리
하둡 Eco 수집
System Info.수집
설치/배포 수행
Storage
분산파일 시스템
HDFS (Hadoop Distributed File System)
RDBMS
My SQL

목차

다양한 분야에서 기존에 처리 할 수 없었던 대용량 데이터에서 실시간으로 다양한 이벤트를 감지하여 Biz 목적에 맞게 반응하려는 요구가 증가

현황

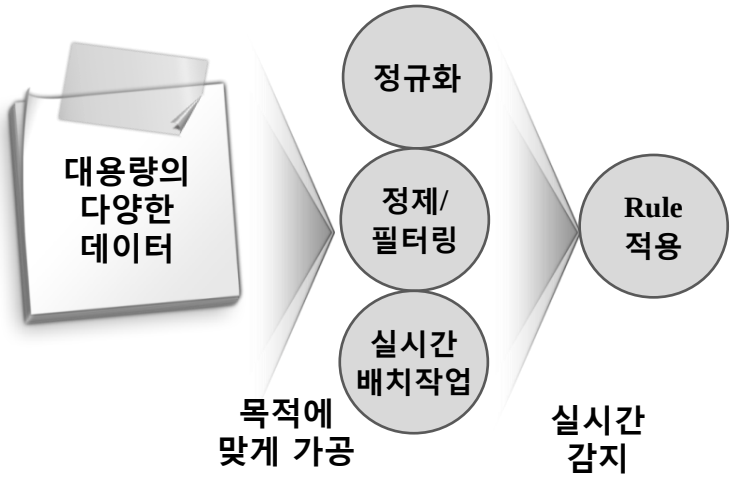


실시간 CEP 이해

- 여러 분야에서 발생하는 대용량의 데이터에서 특정 이벤트들을 적시에 감지하여 목적에 맞게 반응할 수 있는 스마트 시스템이 요구됨
- 대용량의 데이터에서 의미 있는 특정 패턴의 이벤트 필터링이 요구됨
- 하나의 이벤트가 아니라 복수의 이벤트를 상관하여 분석 가능해야 함
- User Component를 활용하여 기존 Biz Solution과도 연동 가능해야 함
- 실시간으로 대용량데이터 처리가 필요하여 기존 CEP 엔진으로는 처리가 불가능함

실시간으로 처리할 데이터의 특성을 파악하여 비즈니스 목표에 맞게 적절한 분산 병렬 시스템을 설계 구축 해야 함

데이터 특성



- 다용한 데이터의 구조 파악
  - 데이터 구조와 처리할 데이터 양 파악
- 필요한 필드 추출할 정규화 작업 정의 (필드 정의)
- 필터링 정제 로직 설계
- 실시간 데이터 처리의 최종 목표와 Rule 정의
- 실시간(마이크로) 배치 작업의 수준 정의
  - Map / Filter / MapReduce / Repartition / Sliding Time Window / Join / Group By

분산병렬 처리

- 대용량데이터를 실시간으로 수용 가능한 아키텍처 필요
  - 수집-> 처리-> 저장->감지 I/F간 병목 현상이 없어야 함
- 실시간 감지와 반응을 비즈니스 목적에 맞게 적시에 수행
  - Milli-Seconds / Seconds / Minutes
- 목적에 맞는 분산병렬 Tool 선택
  - ElasticSearch / HBase / In-Memory DB
- 감지 속도 향상, Scale Out 방식의 유연한 확장(x86기반)

병렬처리 가능한 Hadoop기반 CEP영역은 '14년 부터 글로벌 대형 기업(Tibco, SAS, Oracle) 중심으로 시장 본격화, 국내는 최근 일부 벤처기업에서 솔루션 개발 중

구분	회사	제품명	특징	비용	비고
외산	TIBCO	StreamBase	• 실시간 처리 성능(응답시간, 처리량)면에서 우수 • 실시간 복합 이벤트 Rule 처리 기능 제공	유료	• 시각화 솔루션 별도 구매 (LiveView)
	Oracle	Stream Explorer	• Web기반으로 사용자가 쉽게 데이터흐름을 제어 • 실시간 데이터의 시각화 기능 포함	유료	• 메모리DB제품과 Bundle로만 판매
	SAS	EventStream Processing	• Stream 이벤트를 사전정의 Rule기반으로 탐지하거나 데이터를 분석하여 패턴을 파악하는 기능 제공	유료	• 시각화 솔루션 별도 구매(Visual Analysis)
국산	Raonbit	Raonbit	• 대용량 실시간 데이터와 배치 데이터의 분석 플랫폼 • Open Source를 활용한 제품 구성	유료 (License 정책 수립 중)	• 현재 솔루션 개발 진행단계

실시간 솔루션 필요성

- **Open Source 기반으로, 상용 Tool의 비용 부담(서버 증설 시, 비용 부담 高)을 해결**
- **실시간 처리/분석에 필요한 시각화 기능을 포함하여 독립 솔루션化 (번들/옵션 방식이 아님)**

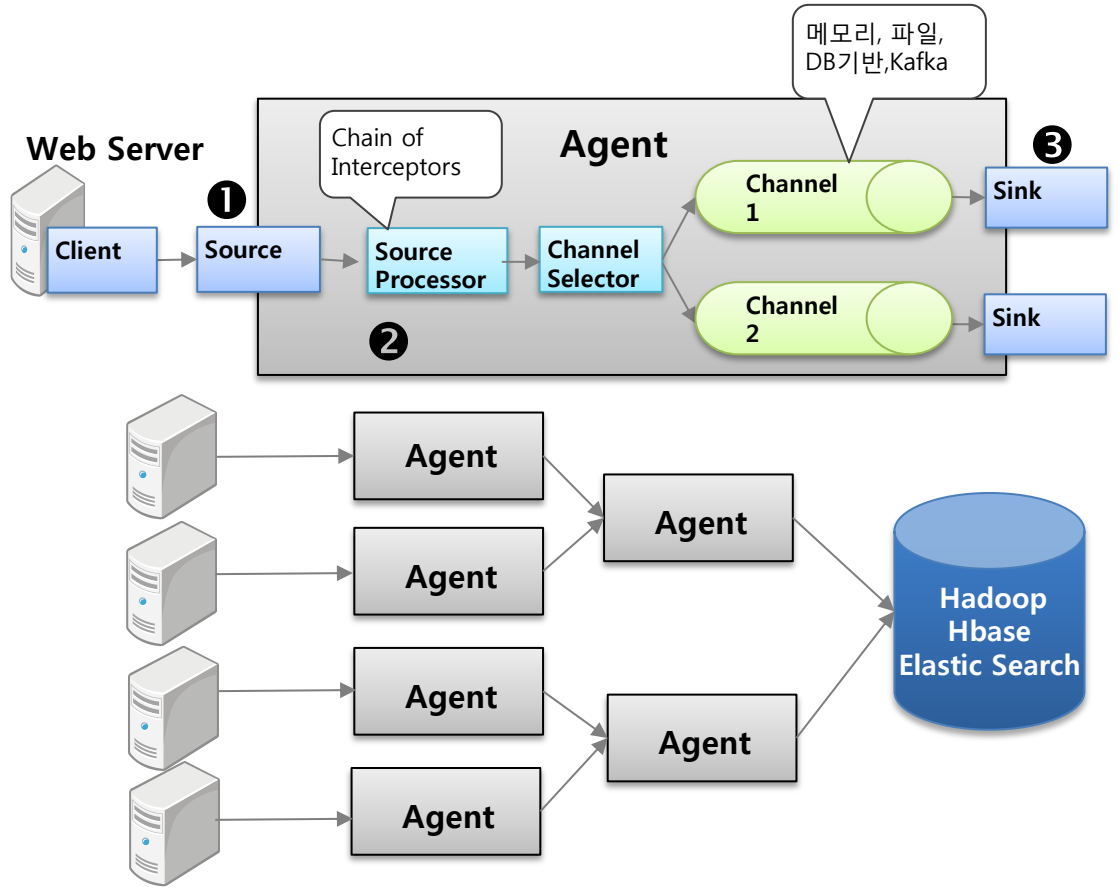
오픈 소스 기반의 솔루션중 최근 Spark Streaming이 다른 기술과 연동하여 대용량 데이터 처리뿐만 아니라 기계학습/R 분석도 가능하여 활발히 사용되고 있음

	Spark Streaming	Storm	Storm Trident
Processing Model	Micro Batches	Tuple	Micro Batches
성능	++++	++	++++
Latency	Second	Sub-Second	Second
Reliability Model	Exactly Once	At least once	Exactly once
Embedded Hadoop Distro	HDP, CDH, MapR	HDP,MapR	HDP
기술지원 업체	Databricks		
Community 활성	++++	++	++
연동기술 범위	Batch, Streaming, Graph, Machin Learning, SQL (RDBMS, Hive), R	Streaming Only	

Spark 선택이유

가장 많이 사용되는 빅데이터용 수집도구인 Flume은 다양한 데이터 소스와 데이터 정제와 다양한 형태로 대용량 데이터를 저장함

Flume 아키텍처



Flume 특장점

1 다양한 Data Source

- 빅데이터 수집 영역에서 가장 많이 활용되는 도구
- 다양한 Data Source (Avro, Thrift, Files, Http, JMS, EXEC, Syslog등)

2 데이터 정제 가능

- Source와 연동하는 Interceptor를 이용하여 Tagging Filtering 가능 (정규표현식 이용)
- Channel Selector로 Routing 기능 제공

3 데이터 다양한 형태로 전송

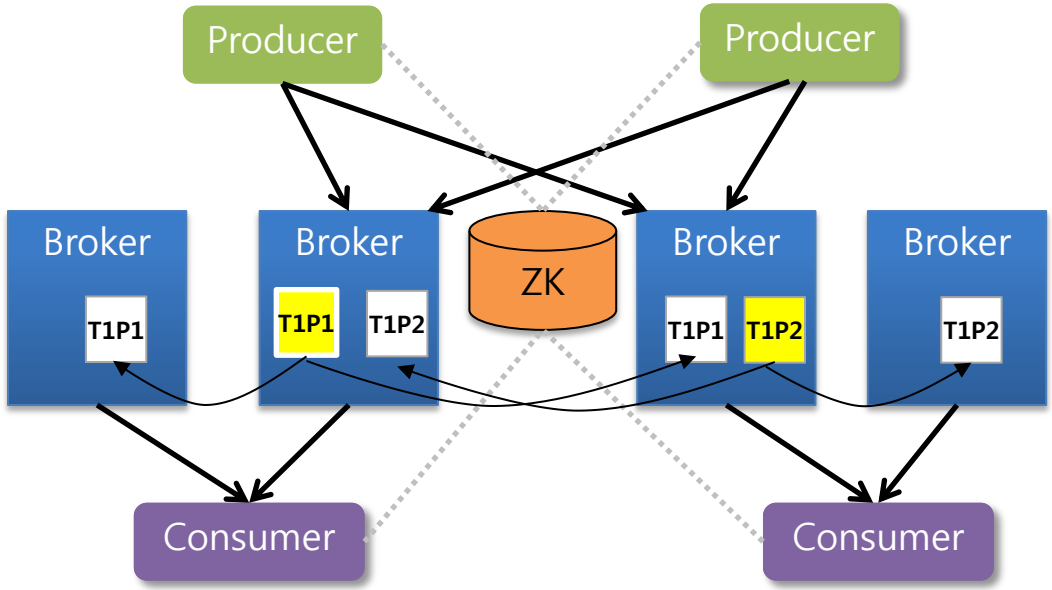
- Avro,Http, Hbase, Hadoop, File, Elastic Search 등다양한 형태로 저장 가능

3 기타 장점

- HA기능 제공
- 다양한 수집도구와도 Plugin 기능제공(Fluentd, Scribe등)

실시간으로 대용량 데이터를 CEP엔진으로 안정적으로 유입하는 버퍼로 사용되고 있으며 메시지 Broker로도 활용되고 있다

Kafka 아키텍처



- 각 노드당 Broker가 토픽들을 관리
- 각 토픽은 멀티 파티션으로 구성 가능
- 복제도 가능함
- 보통 장애 복구를 복제는 2F+1로 하는데 Kafka는 F+1를 보장

Flume 특장점

1 Pub – Sub 구조

- Producer가 메시지를 Kafka 토픽에 publish하고
- 특정 Topic을 미리 구독한 Consumer가 등록된 메시지를 받아 처리

2 Persistency

- 메시지를 파일로 저장하여 Replay가 가능하여 안정성을 높일 수 있음
- 메시지 큐 역할과 Integration 역할도 함

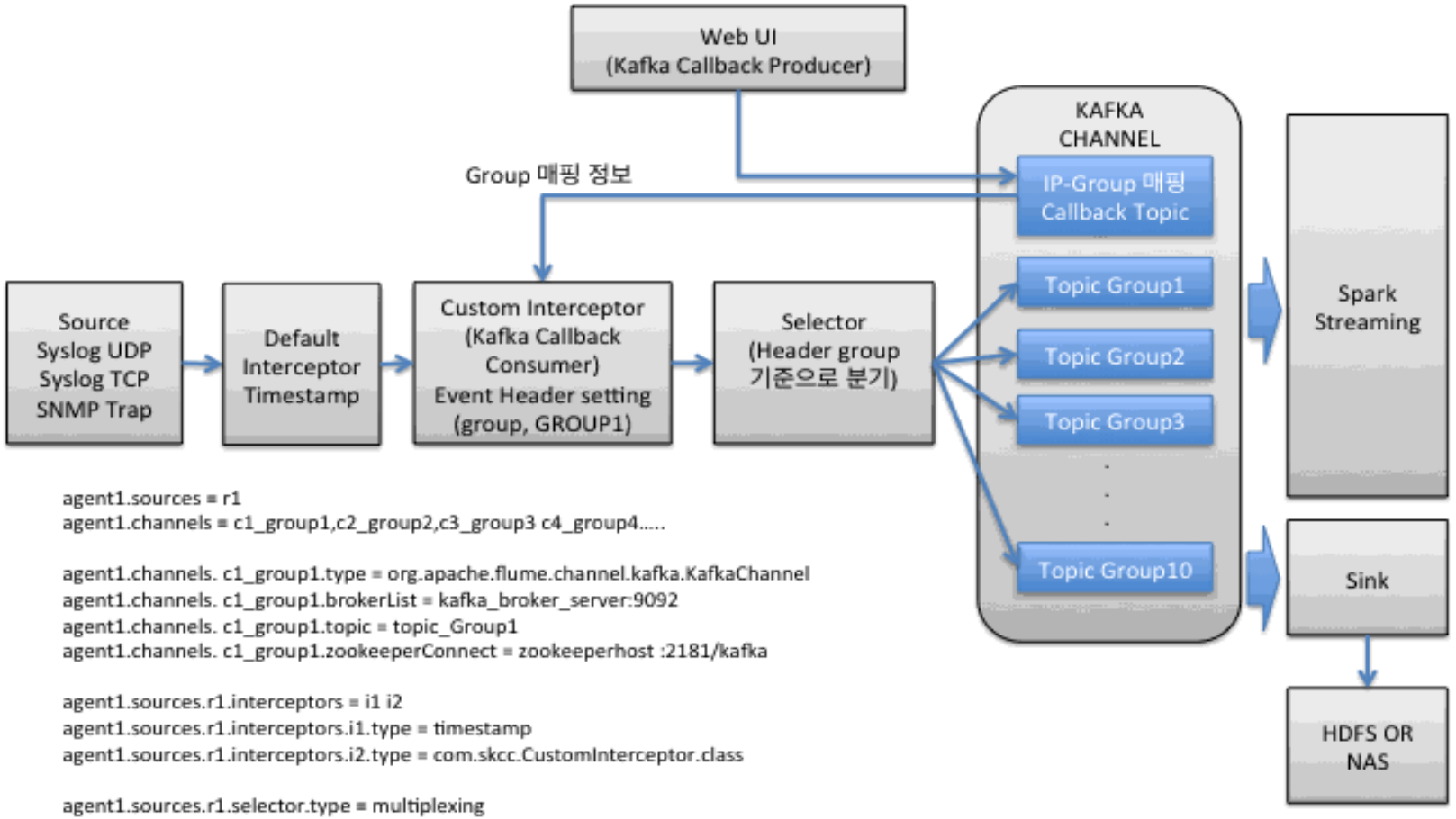
3 Zero-Copy

- OS 레벨에서 파일을 NIC카드와 직접 연동하여 신속한 데이터 처리 가능

3 기타 장점

- 확장 가능한 High Throughput
- 파티션을 활용한 분산병렬 처리로 응답속도 빠름
- 배치처리와 압축처리 지원하여 처리속도 향상

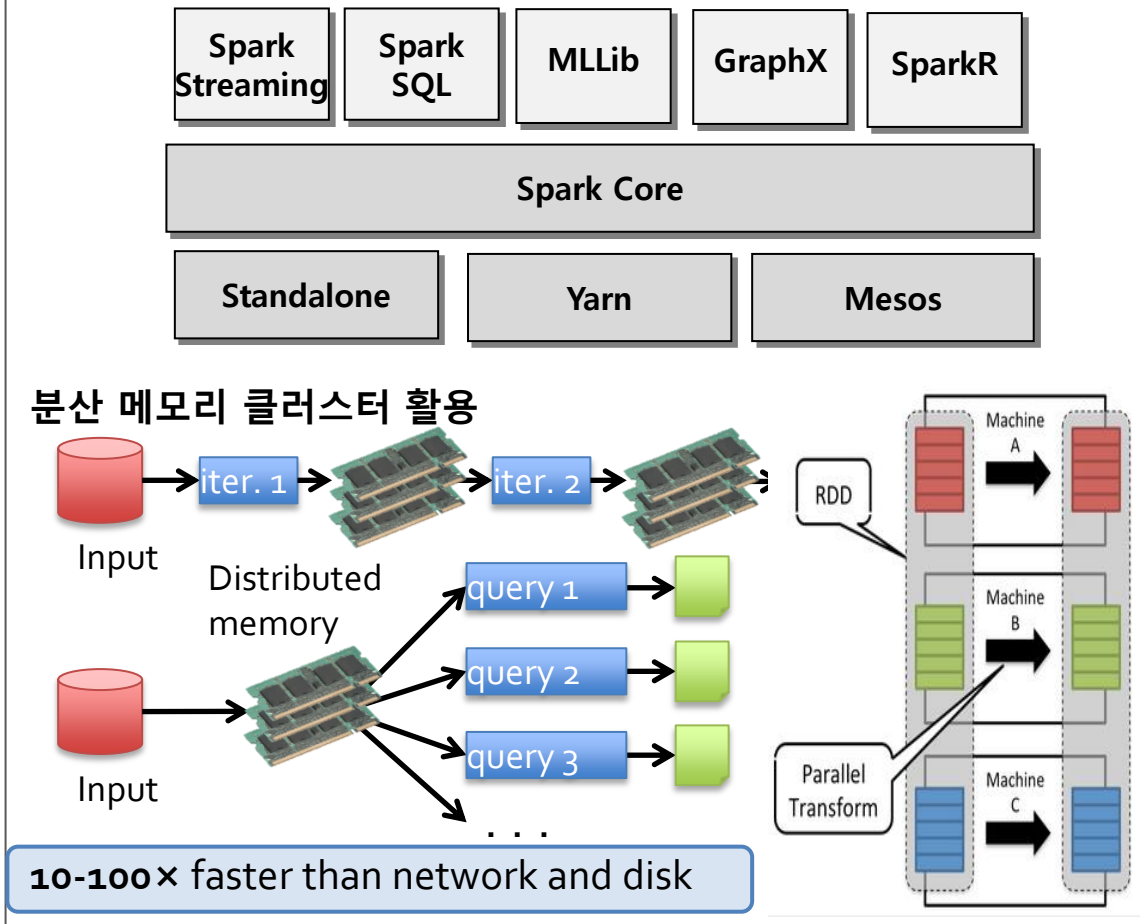
Kafka를 Flume의 채널로 사용하여 데이터 수집을 안정적으로 처리 할 수 있다




```
agent1.sources = r1
agent1.channels = c1_group1,c2_group2,c3_group3 c4_group4.....

agent1.channels. c1_group1.type = org.apache.flume.channel.kafka.KafkaChannel
agent1.channels. c1_group1.brokerList = kafka_broker_server:9092
agent1.channels. c1_group1.topic = topic_Group1
agent1.channels. c1_group1.zookeeperConnect = zookeeperhost :2181/kafka

agent1.sources.r1.interceptors = i1 i2
agent1.sources.r1.interceptors.i1.type = timestamp
agent1.sources.r1.interceptors.i2.type = com.skcc.CustomInterceptor.class

agent1.sources.r1.selector.type = multiplexing
a1.sources.r1.selector.header = group
a1.sources.r1.selector.mapping.GROUP1= c1_group1
a1.sources.r1.selector.mapping. GROUP2= = c2_group2
a1.sources.r1.selector.mapping. GROUP3= = c3_group3
                                                .
                                                .
                                                .
```

데이터 처리 방식이 Hadoop 기반의 File 기반에서 메모리 기반의 Spark로 진화되면서 Hadoop상의 Application들의 성능이 급성장하고 있음

Spark 아키텍처

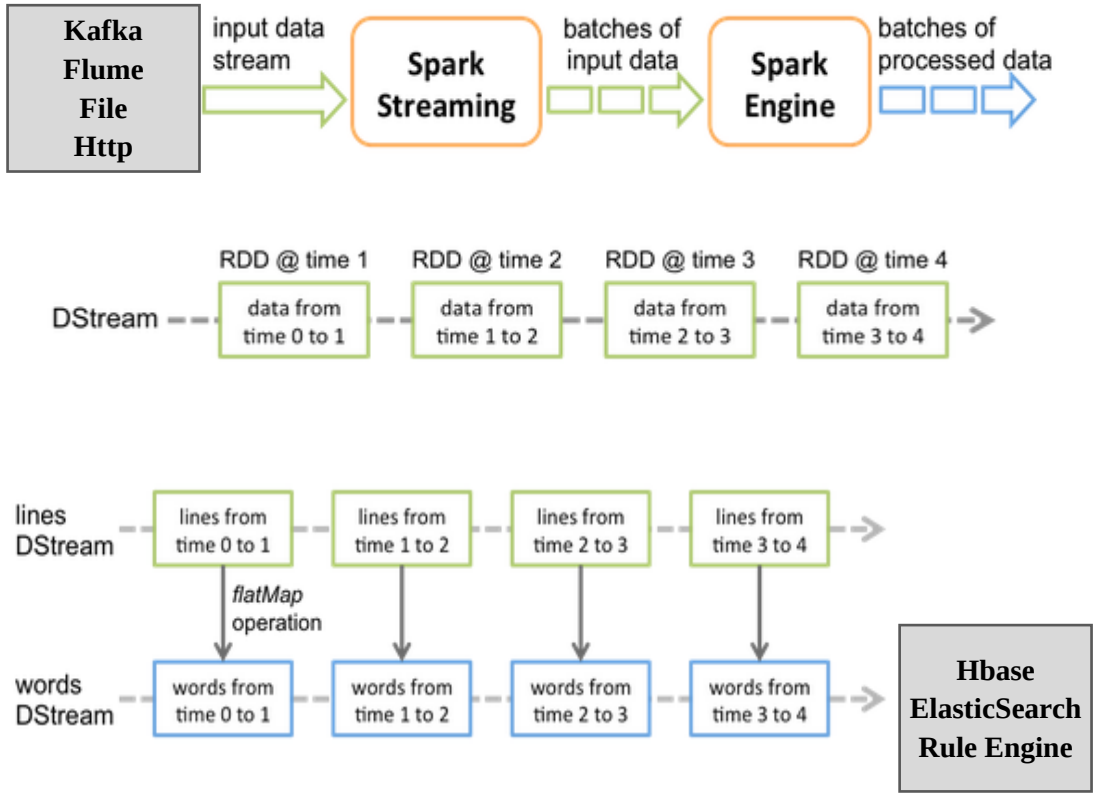


Spark 특장점

- Spark Core기반으로 기계학습 (MLLIb,), 그래프분석, R, SQL, 및 스트리밍등 목적에 맞게 다양한 라이브러리 연동하여 사용 가능
- Standalone, Yarn, Mesos기반으로 데이터 처리
- Resilient Distributed Data (RDD)기반으로 데이터를 메모리 클러스터에 분산시켜 처리하여 우수한 성능이 보장
- RDD는 병렬로 처리될 수 있는 Immutable (read-only) , partitioned 된 elements의 집합
- RDD Operation은 변환작업을 하는Transformation과 결과 처리하는 Action으로 나뉨
- Spark SQL은 표준SQL과 Hive-QL 지원
- Spark Streaming은 초단위로 Micro-Bath작업을 하여 실시간으로 데이터 신속히 처리 가능 (초당 수십만 건 처리 가능)

Spark Streaming은 Micro Batch 기술을 활용하여 Streaming 데이터를 실시간 배치 처리 하는 기술로 실시간 솔루션으로 최근 많이 활용되고 있다

Spark Streaming 아키텍처

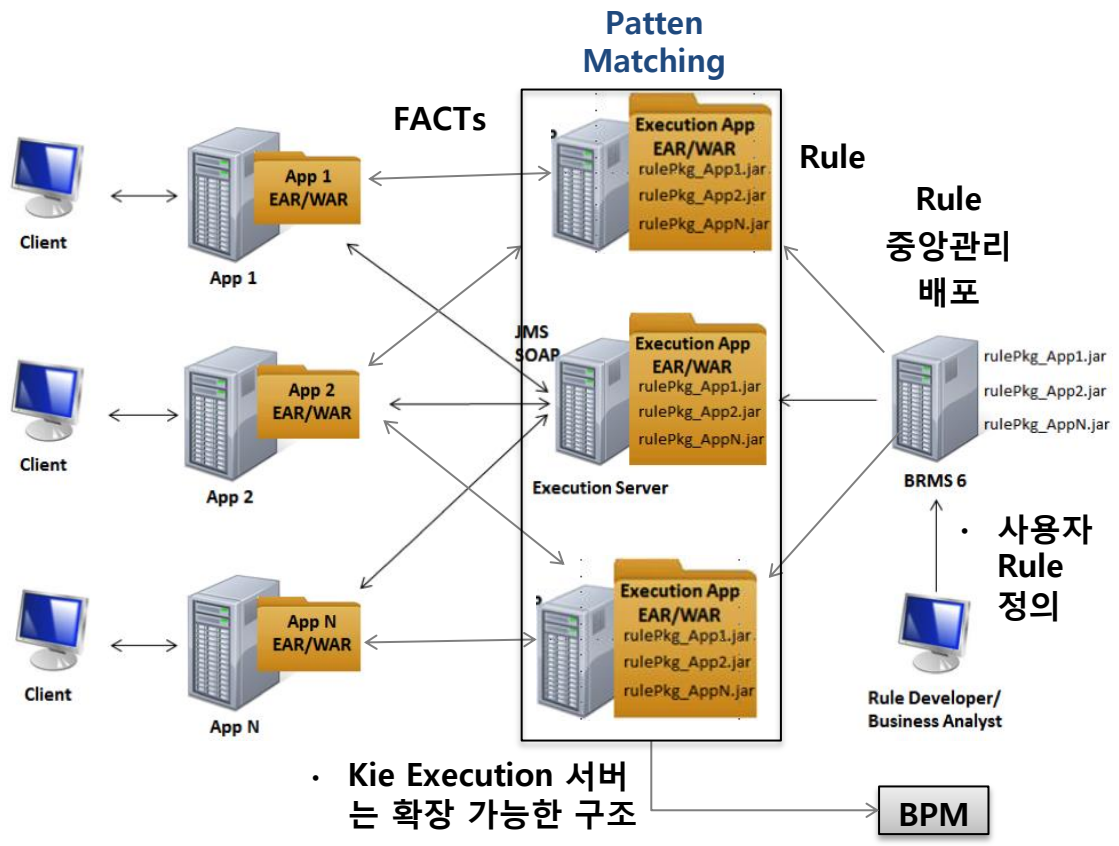


Spark Streaming 특장점

- 실시간으로 들어오는 data stream은 batch 단위로 나뉘어 지고 나뉘어진 batch 단위의 data는 Spark 엔진에 의해서 processing 된 뒤에 최종 final stream이 생성
- Spark streaming은 Discretized stream 혹은 Dstream이라고 하는 High-level abstraction을 제공
- DStream은 여러 input 소스에서부터 생성될 수 있음 (Kafka, Flume, File 등)
- DStream은 연속적인 RDD의 집합
- DStream 내 RDD는 일정한 인터벌 시간 내 존재하는 Data가 들어있음
- Dstream은 다른 Dstream의 변환되었다가 최종적으로 Output으로 다른 저장 또는 처리 시스템에 전송된다

오픈소스 기반의 Rule 엔진인 Drools를 활용하여 실시간 CEP 엔진을 구성 가능함

Drools 아키텍처



Drools 특장점

- 성숙된 오픈소스 기반의 Rule 엔진
- 애플리케이션에서 작업 흐름을 제어하는 조건들을 별도 분리
- Rule 변경으로 인해 전체 어플리케이션을 다시 빌드할 필요가 없음
  => Rule 변경 동적으로 적용 가능
- Rule을 별도의 파일에 저장
- Web UI를 Workbench로 제공
- 사용자 그룹에 의한 Rule 권한 설정 가능
- 모든 Rule들이 단일 저장소(repository)에 저장되어 통합된 Rule 관리 가능
- 읽기 쉽고 편한 Rule 포맷 제공
- 분산 처리가 안됨

목차

동기

- 인포섹 Big Data 기반 실시간 보안관제 솔루션 개발 => 신속한 탐지와 대응 + 무중단 서비스 제공
- 대용량 보안 로그 데이터를 실시간으로 분산 처리 할 수 있는 기술력 확보
- 향후 Machine Learning 및 Data Mining 등과 융합하여, 보안 위협 예측과 침해 패턴 추출 등을 자동화하는 솔루션으로 진화가 최종 목표

공격 탐지 logic

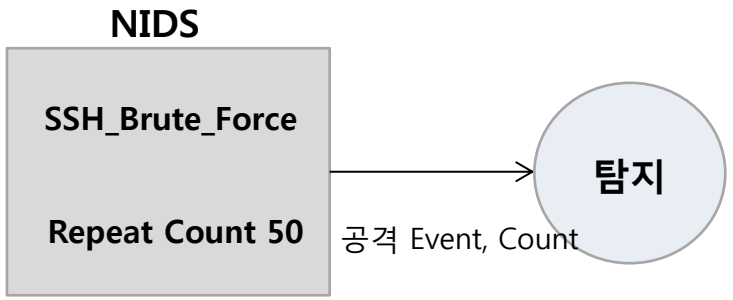
AS-IS

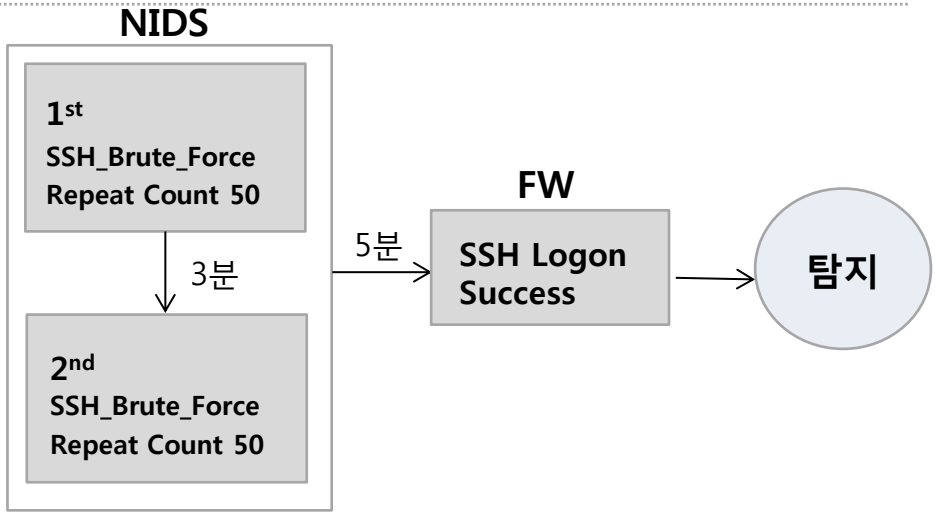
- **단일 탐지**
  - 개별 솔루션의 단일 Event 중심 탐지
- **탐지 Field 조건**
  - 10개 Field->Count 임계치, 공격 문자열, 내외부 등
- **상관 탐지**
  - 상관 탐지 분석 수행하지 않음
- **배치 작업으로만 탐지 (1시간이상 소요)**

TO-BE

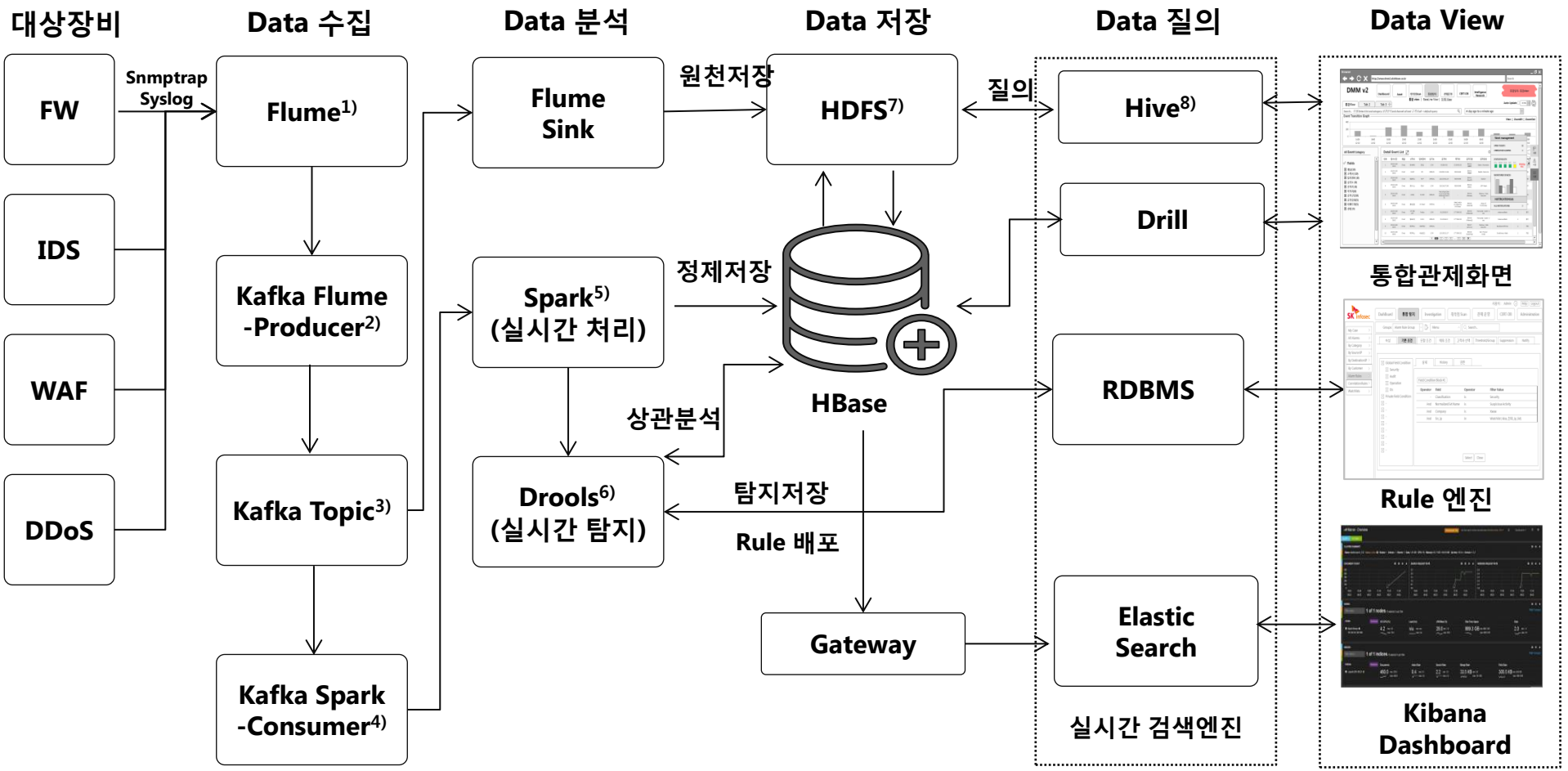
- **단일 탐지**
  - 단일 Event의 조건 Field 추가
  - Time 기준 탐지 조건 추가
  - 전체 공격 유형 표준화
- **탐지 Field 조건**
  - 50개 Field 조건 확장
- **상관 탐지**
  - Multi-Device 상관 분석 수행하여 정확도 향상
- **실시간 분산 병렬 시스템 구축 (초단위)**

공격 탐지 예시



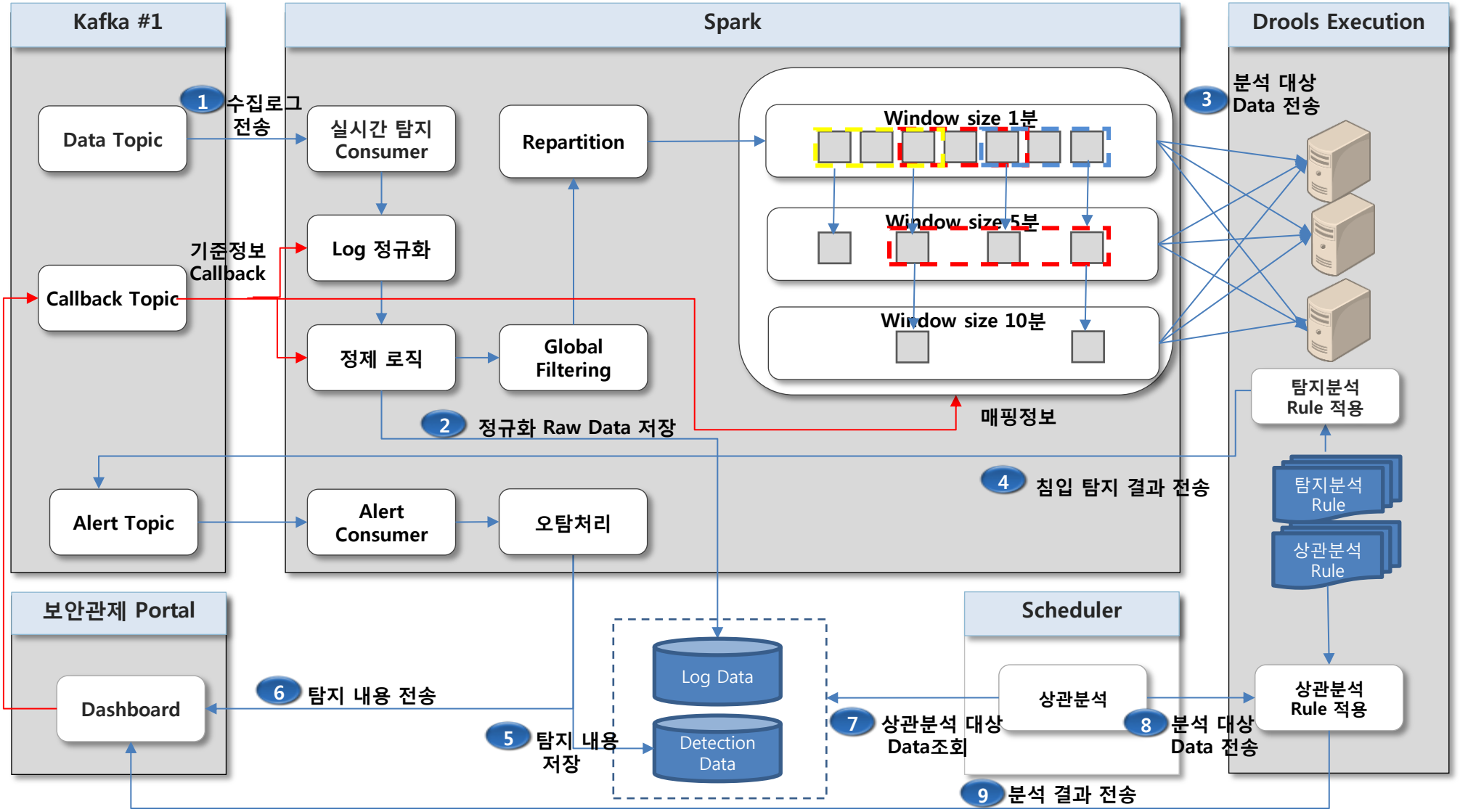


Platform의 전체 구성은 Data 수집/분석/저장/질의 하는 4단계로 구성되는 엔진과, 탐지 및 질의 결과를 화면에 View 하는 UI로 구성됨.

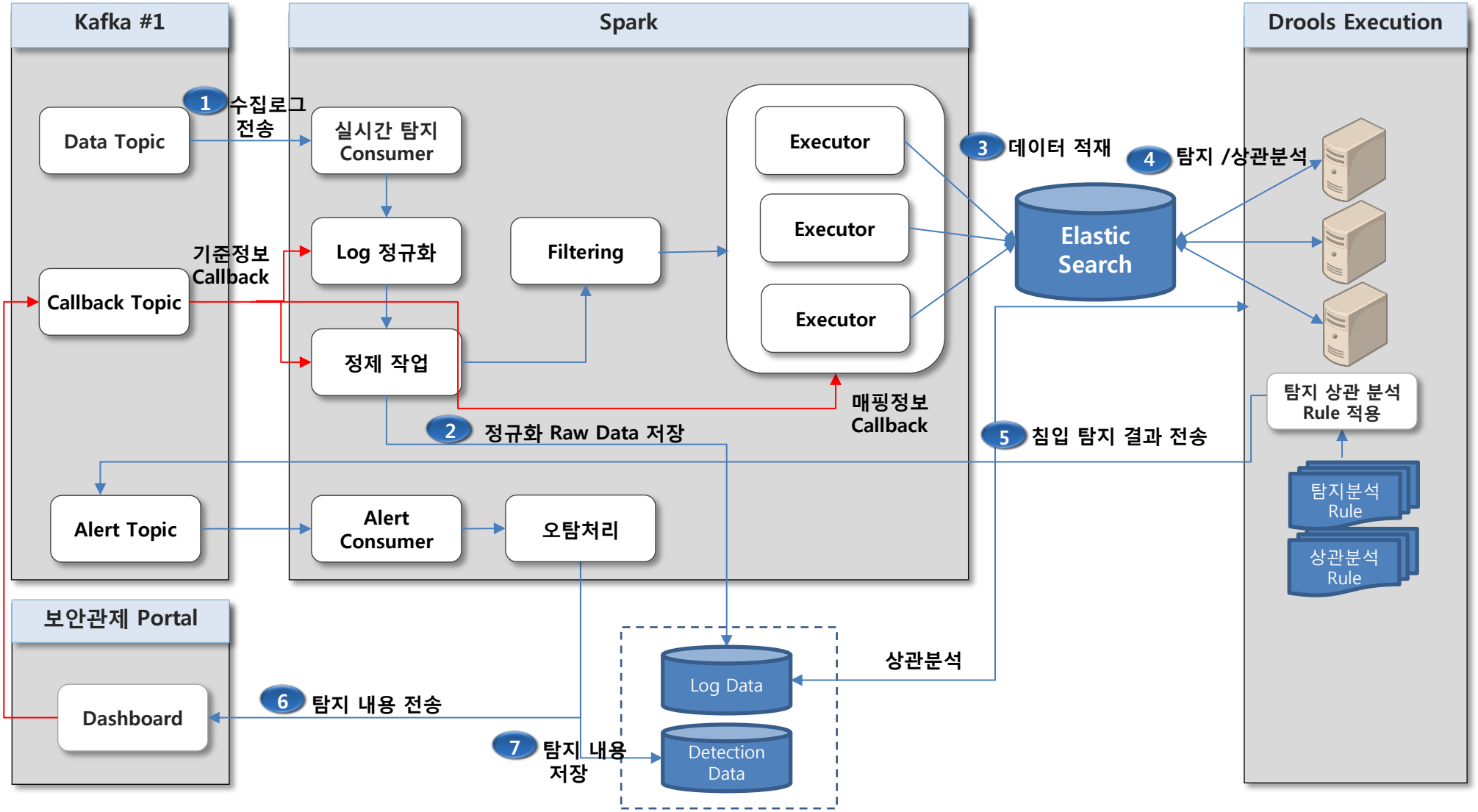


	내용	장단점
Spark Streaming Time Window 활용 방안	▪ Drools의 Time Window 기능 대신 Spark Streaming의 Time Window 기능을 활용 ▪ Spark Streaming에서 같은 데이터를 여러 개의 Time Windows를 생성 ▪ 각 레코드는 특정 Window의 Summary 데이터를 매핑 정보를 통하여 해당 Drools 서버로 전송	▪ Spark Streaming의 자원 사용률이 높아 적절한 부하 분산이 필요 ▪ Rule을 동적으로 Update시에도 영향이 없음 ▪ Drools Rule 서버의 부하를 덜어줌 ▪ 실시간 탐지분석과 상관분석을 별도로 수행 ▪ 다양한 사이즈의 Time Window 사용이 제한적임
Elastic Search 활용 방안	▪ Spark Streaming에서 데이터 정규화 정제 후 바로 Elastic Search에 적재 ▪ Drools에서 Rule에 따라서 Elastic Search에 조회하여 탐지 분석 수행 ▪ Elastic Search 시간을 제외한 상관 분석은 Hbase나 Hive를 병행하여 활용하여 함	▪ Elastic Search의 검색 기능 활용 가능 ▪ 실시간 탐지분석과 상관분석을 한번에 수행하여 프로세스가 단순 해짐 ▪ 실시간 데이터 처리 성능이 Elastic Search의 데이터 처리 용량에 제약을 받음 => 수십 TB 분석에 적용하기는 어려움
Kafka 활용방안	▪ 1차 Spark Streaming에서 정규화 / 정재 후 매핑 정보를 활용하여 각 데이터를 해당 파티션에 Grouping하여 배분 ▪ 2차 Spark Streaming에서 Grouping된 데이터를 각 파티션 별로 Drools로 전송 ▪ Drools 는 Rule 적용하고 분석 수행	▪ Spark Streaming에서는 간단히 분류 작업만 수행하여 부하가 마니 줄어듦 ▪ Kafka를 활용하여 Group By를 수행하게 됨 ▪ Drools Rule 서버는 Time Windows 기능을 사용하여 Rule 분석하여 서버 부하 증가

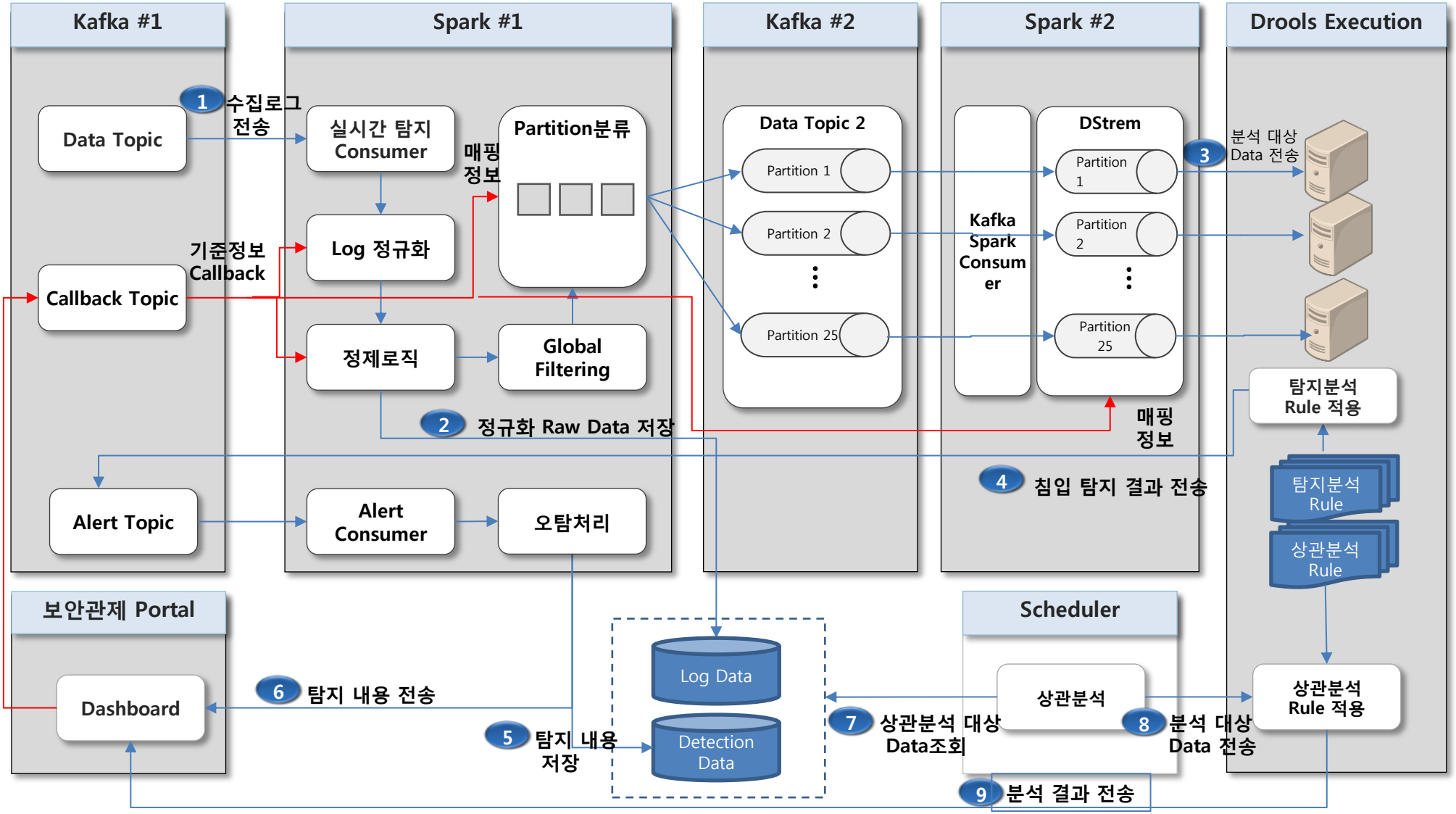
Spark Streaming의 sliding time windows를 활용하여 이벤트 카운트하여 Rule서버에 적용



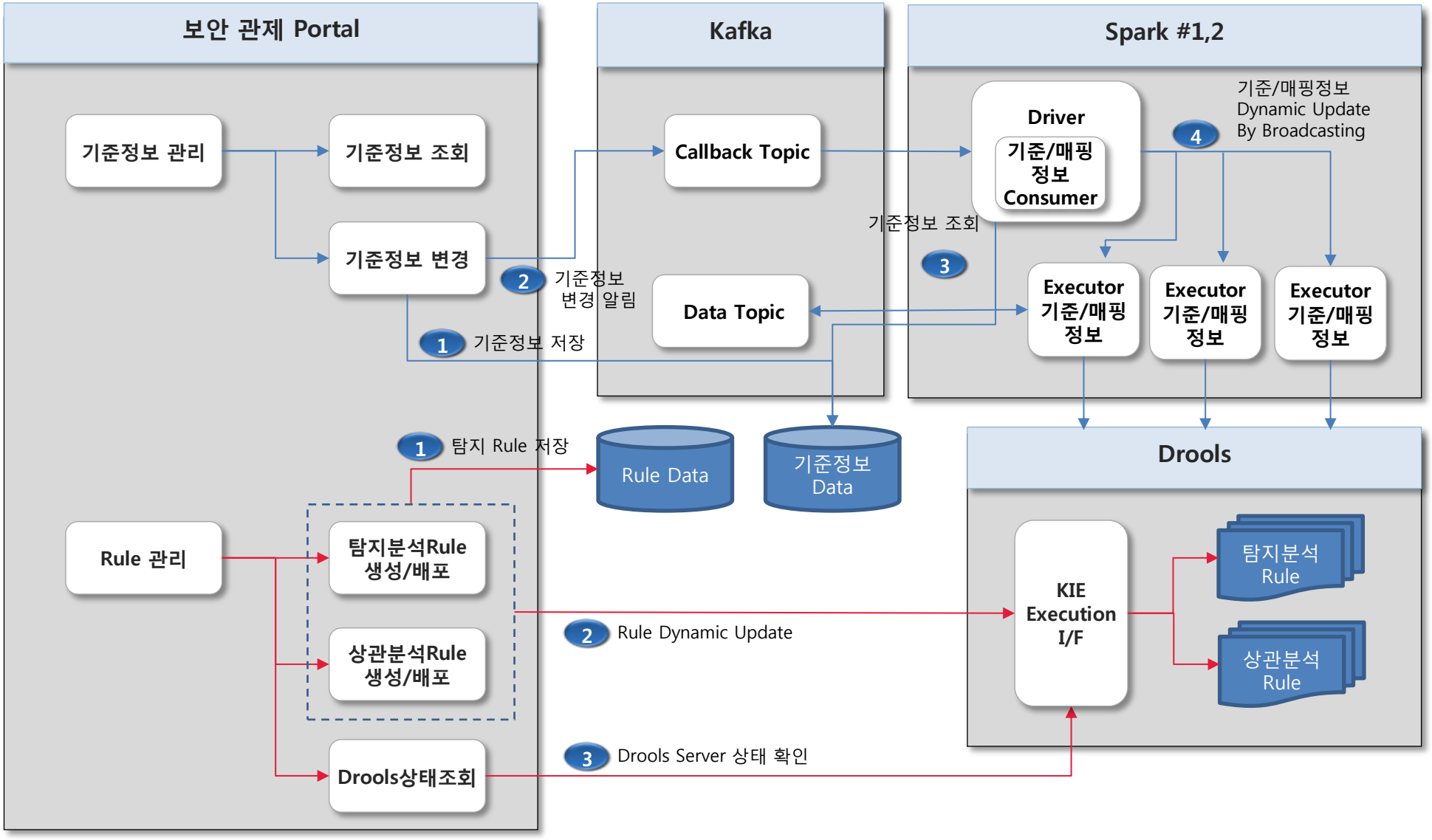
분산 검색엔진인 Elastic Search를 활용하여 대용량 데이터를 신속하게 검색하여 실시간 탐지에 적용



Kafka를 활용하여 Grouping하여 Spark Streaming의 부하 분산

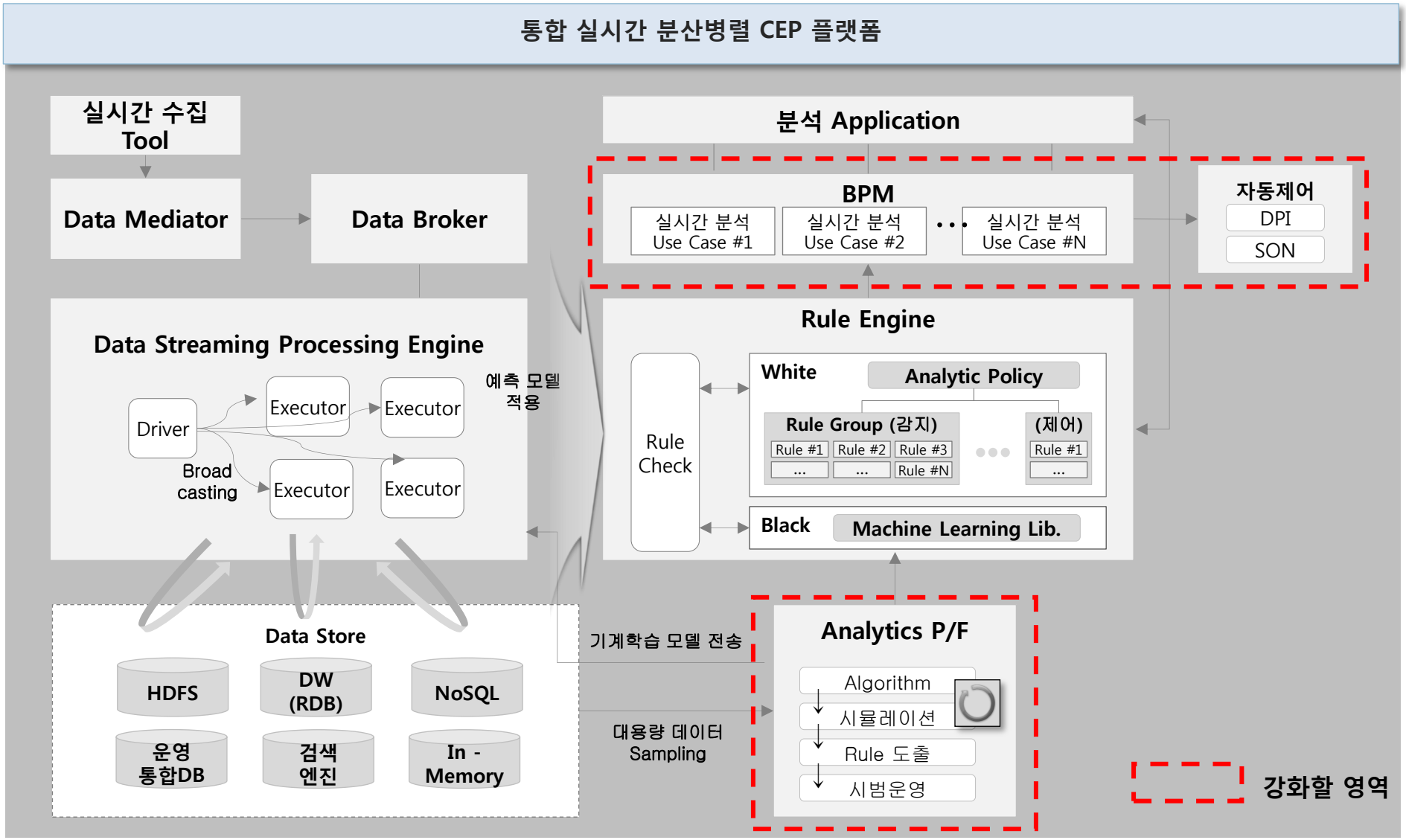


Rule이나 기준정보 변경 시 서비스의 중단 없이 동적으로 변경사항 적용하는 방안 필요



목차

대용량 데이터를 기계 학습하여 데이터 기반의 분석모델을 자동 생성하여 실시간 동적으로 적용하여 비즈니스 프로세스까지 자동화 할 수 있는 플랫폼 필요



- **데이터의 특성 파악이 가장 중요**
  - 데이터 구조와 처리할 데이터 양과 보관주기를 파악
  - 데이터 정제 로직 파악 (정규화, 필터링, 기준데이터와 연동방안)
  - 실시간으로 처리할 로직의 수준 결정
    (간단한 map/filtering 작업, 적당한 배치작업, 무거운 작업)
- **비즈니스 목표에 맞는 시스템 구현**
  - 실시간의 수준 정의 (초당 몇 건 처리가 목표)
  - 처리 속도에 필요한 SW 선택 => 목표에 맞는 시스템 구현
  - Streaming 처리에 병목현상 없게 해야 함
    - 효율적인 코딩 필요 (예: repartition 수, cache, recoverable, sliding windows)
    - 적절한 Kafka Partition 수나 Rule 서버 대수
- **PoC 해봐야 안다** => 목표에 맞는 처리 방안을 찾아야

  SW변경, 아키텍처 변경, 자원증설, 그래도 안되면 목표 수정도 고려
- **다양한 분석 방법 / BPM과 연동 고려**
  - R, Machine Learning, Graph, 추천, 형태소분석…

Q & A