

# SMACK 스택으로 본 빅데이터 분석 인프라의 구조

2017. 12. 14

민영근

# 민영근

- <https://github.com/minyk>
- 2016.6 ~ AJ 네트워크 지주부문 IT센터 아키텍처 팀
- 2014.8 ~ 2016.6 kt NexR 개발팀
- 2013.8 ~ 2014.8 kt NexR TA팀
- 2011.8 ~ 2013.7 단국대학교 연구전담 전임강사

# 목차

- 강의 목표
- SMACK 스택이란?
- SMACK 스택: Spark
- SMACK 스택: Mesos
- SMACK 스택: Akka
- SMACK 스택: Cassandra
- SMACK 스택: Kafka
- 다른 스택들
- 마치며

# 강의 목표

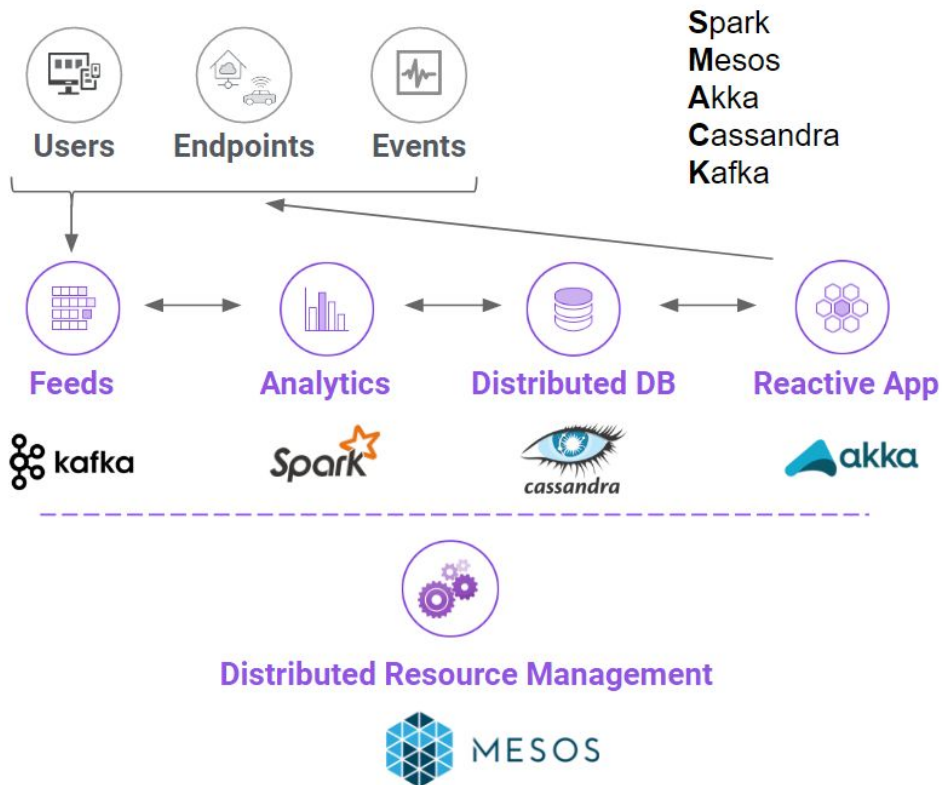
- **SMACK** 스택을 예시로 한 빅데이터 분석 인프라에 대한 정리
  - 필요 기능
  - 구성 요소
- 구성 요소별 특징과 대안들

# 빅데이터 분석 인프라

- 기능 요소
  - 수집
  - 저장
  - 처리
  - 내/외부 연동
  - 관리
- 비기능 요소
  - 고가용성 또는 내고장성
  - 선형 증가 가능

# SMACK 스택

- 빅데이터 분석의 LAMP
  - linux, apache, mysql, php
- 빅데이터 분석 시스템을 쉽게 구축할 수 있음.
- 구성 요소
  - Spark
  - Mesos
  - Akka
  - Cassandra
  - Kafka

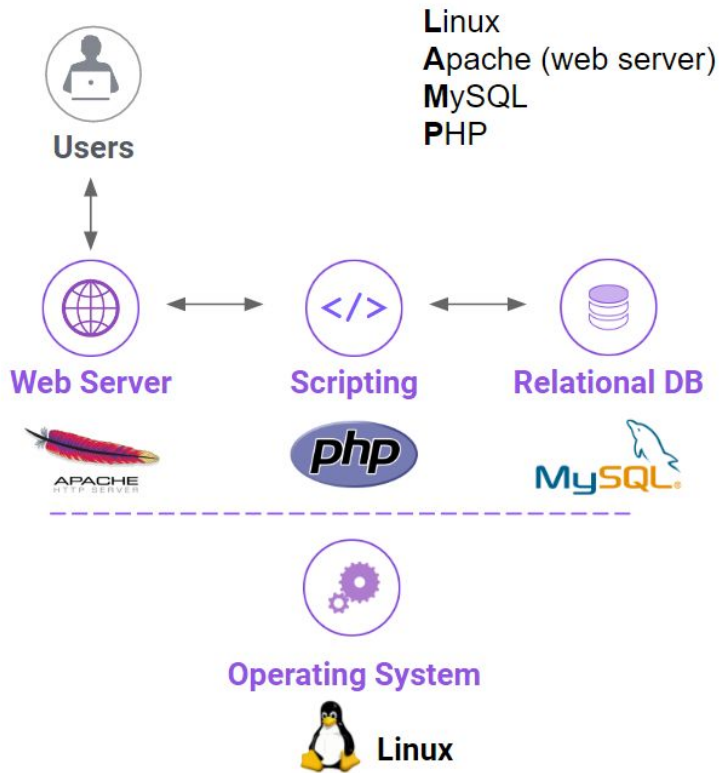


# LAMP ?!

- 2000년대 초반까지 많이 사용된 기본적인 웹 구축 스택
  - Linux: 운영 체제
  - Apache: 웹 서버
  - MySQL: 관계형 데이터베이스
  - PHP: 서버사이드 스크립트 언어
- 현대에도 사용되지만, 웹 서비스의 역할이 커지고 자바스크립트의 발전에 따라 다양한 형태로 구성됨.

## LAMP Stack

Enabling dynamic web applications



# SMACK 스택: Spark

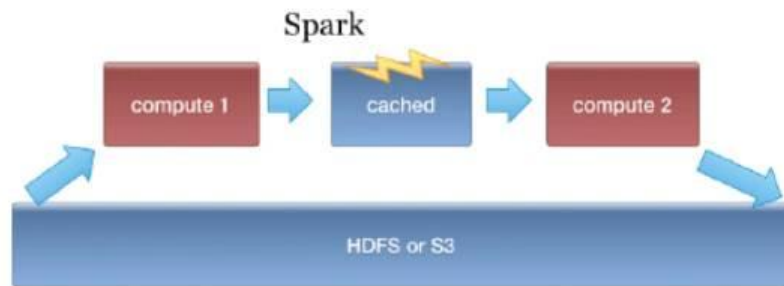
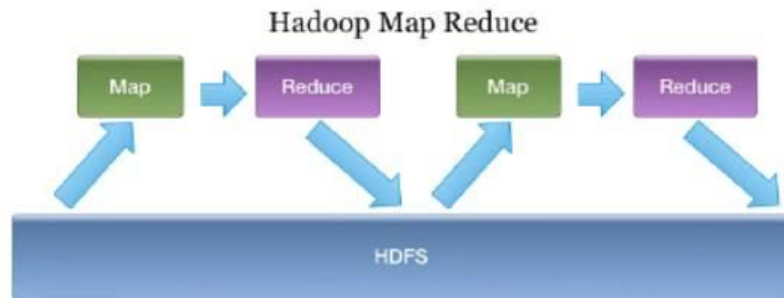
- 미국 버클리 대학 박사과정 수업에서 개발
- 메모리 기반의 처리 기술/ 엔진
- Scala를 기준으로 다양한 언어 지원
  - Python: 개발 전문이 아니더라도 사용할 수 있음.
  - Java: 기존 개발자/ 회사들의 진입 장벽을 낮춤.
- (준)실시간/ 배치/ On-demand 처리까지 범용성 확보
- Hadoop 이후의 빅데이터 영역에서 선두를 달리고 있음.





# SMACK 스택: 고속 처리의 산업 표준 Spark

- 메모리 기반: 디스크 기반인 Hadoop Mapreduce의 단점을 극복
- Mapreduce와 유사한 모델 사용: 논리적인 변경이 어렵지 않음
- Mapreduce의 구조를 변경: 구조적 한계점을 없앴



# SMACK 스택: Spark 2.x과 이후의 방향

- Spark 1.X
  - 핵심 엔진 및 기능 안정화
  - SparkQL 등 사용성 강화를 위한 기능 추가
- Spark 2.0 이후로 실시간 데이터 분석에 초점을 맞춤
  - 실시간 데이터를 직접 질의할 수 있는 기능 추가
  -

# SMACK 스택: Spark를 대체할 수 있는 처리 엔진

- Apache Hadoop Mapreduce
  - 이 계열의 원조: 오래된 만큼 안정적이고 널리 알려져 있음.
  - 디스크 기반이며 프로그래밍 구조적으로 Spark 보다는 느림.
  - 직접 사용되기 보다는 Apache Hive, Apache Pig의 backend로 많이 사용되었음.
- Apache Storm
  - 완전한 실시간 처리 프레임워크
  - bolt라는 작은 단위를 만들고 연결함으로서 실시간 처리 파이프라인을 구성
  - Storm 2.0이 늦어지면서 현재 아주 활발하지는 않은 프로젝트

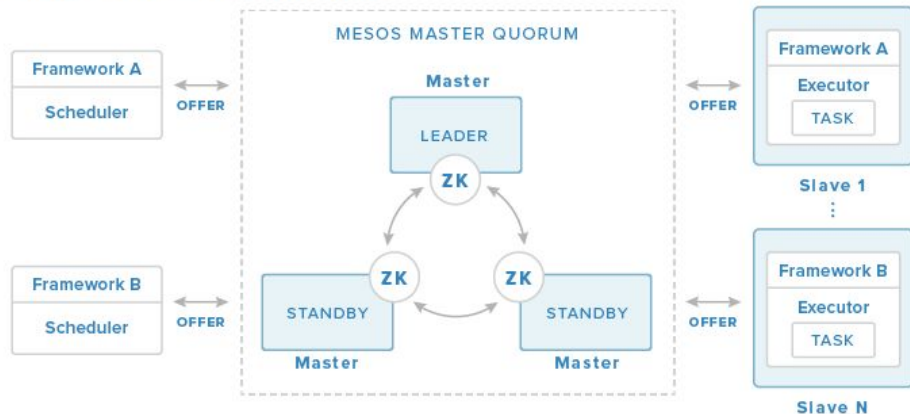
# SMACK 스택: Mesos



# MESOS

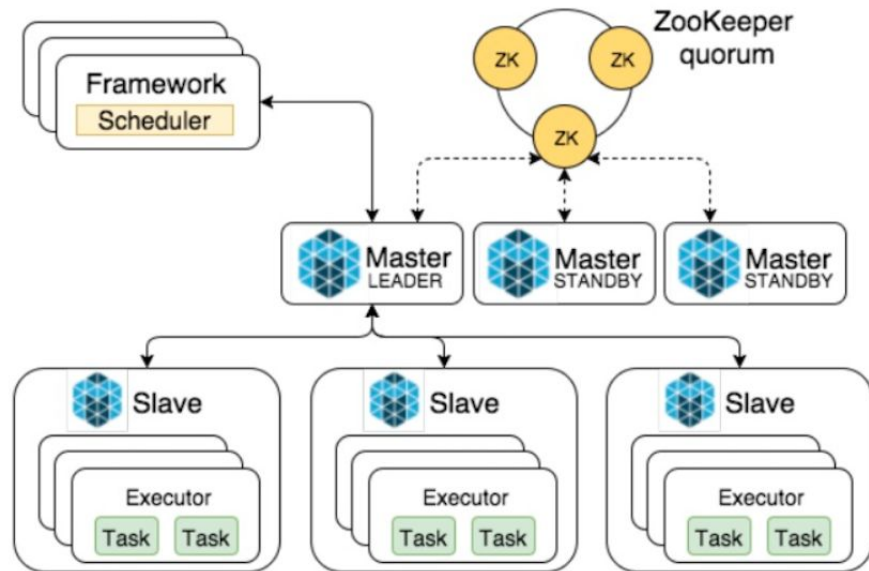
- 미국 버클리 대학 박사과정 수업에서 개발 시작
- 점점 증가하는 컴퓨팅 자원을 통합하여 관리할 수 있는 프레임워크
  - CPU, Memory, Disk, 네트워크의 기초적인 자원부터 현재는 GPU 등의 자원도 포함 됨.
  - Linux Container 기술을 사용하여 격리된 환경으로 제공
- Master/Slave 구조
  - Slave의 자원을 Master가 관리하면서 요청하는 자원을 제공
  - 자원이 필요한 프로그램은 Mesos에

Example Mesos Architecture



# SMACK 스택: 데이터센터의 운영체제 Mesos

- 수십 ~ 수백대의 서버를 하나의 Pool로 관리
  - 자원을 하나의 관점에서 효과적으로 관리
  - 과거의 사용방식과는 다르게 80% ~ 90% 정도까지 사용률을 끌어 올림
- Continuous / Batch 작업을 하나의 클러스터에서 Side-by-Side로 구동
  - 작업별로 별도의 클러스터를 만들 필요 없음



# SMACK 스택: Mesos 자원 관리

Mesos

FrameworksAgentsRolesOffersMaintenance

dcos.alway.kr

Master d6627156-ac6c-4f17-97c1-a62865488e09

Cluster: dcos.alway.kr  
Leader: 172.22.30.45:5050  
Version: 1.4.0  
Built: 2017-09-04T14:39:01+0900 by  
Started: 2017-12-08T16:41:25+0900  
Elected: 2017-12-08T16:04:52+0900

LOG

Agents

Activated24

Deactivated0

Unreachable2

Tasks

Staging0

Starting0

Running131

Unreachable0

Killing0

Finished254

Killed126

Failed608

Lost0

Resources

	CPU	GPU	Mem	Disk
Total	912	0	3.5 TB	109.0 TB
Allocated	172.1	0	424.3 GB	1.7 TB
Offered	0	0	0 B	0 B
Idle	739.9	0	3.1 TB	107.3 TB

Active Tasks

Find...

Framework ID	Task ID	Task Name	Role	State	Started	Host	
ffe3c218-5b90-4b9e-b326-532732ebc0e6-0011	dev_wms-dev.d975b2d4-e059-11e7-b033-9e90e9fd3a19	wms-dev.dev	total	RUNNING	2017-12-14T08:03:36+0900	172.22.30.19	<a href="#">Sandbox</a>
ffe3c218-5b90-4b9e-b326-532732ebc0e6-0013	ajcp_pxc1.999cc30e-dfe2-11e7-bc8d-c6ec0c840608.1	pxc1.ajcp	capital	RUNNING	2017-12-13T17:51:51+0900	172.22.30.36	<a href="#">Sandbox</a>
ffe3c218-5b90-4b9e-b326-532732ebc0e6-0016	rerp_tomcat.a93840e-dfd4-11e7-9070-9e90e9fd3a19	tomcat.rerp	rentacar	RUNNING	2017-12-13T16:10:50+0900	172.22.30.20	<a href="#">Sandbox</a>
ffe3c218-5b90-4b9e-b326-532732ebc0e6-0016	rerp_tomcat.a33cc8bd-dfd4-11e7-9070-9e90e9fd3a19	tomcat.rerp	rentacar	RUNNING	2017-12-13T16:10:30+0900	172.22.30.37	<a href="#">Sandbox</a>
ffe3c218-5b90-4b9e-b326-532732ebc0e6-0013	exhibitor.4b314378-dfd4-11e7-bc8d-c6ec0c840608.1	exhibitor	capital	RUNNING	2017-12-13T16:07:47+0900	172.22.30.34	<a href="#">Sandbox</a>
ffe3c218-5b90-4b9e-b326-532732ebc0e6-0013	exhibitor.4b311c66-dfd4-11e7-bc8d-c6ec0c840608.1	exhibitor	capital	RUNNING	2017-12-13T16:07:46+0900	172.22.30.22	<a href="#">Sandbox</a>
ffe3c218-5b90-4b9e-b326-532732ebc0e6-0013	exhibitor.4b30a734-dfd4-11e7-bc8d-c6ec0c840608.1	exhibitor	capital	RUNNING	2017-12-13T16:07:35+0900	172.22.30.18	<a href="#">Sandbox</a>
ffe3c218-5b90-4b9e-b326-532732ebc0e6-0008	gw_tomcat.ca8ecb0b-da65-11e7-b571-32128adc1913.4	tomcat.gw	ajhq	RUNNING	2017-12-13T15:35:58+0900	172.22.30.18	<a href="#">Sandbox</a>
ffe3c218-5b90-4b9e-b326-532732ebc0e6-0016	rerp_nginx.83c2423c-df0f-11e7-9070-9e90e9fd3a19	nginx.rerp	rentacar	RUNNING	2017-12-13T15:33:40+0900	172.22.30.34	<a href="#">Sandbox</a>
ffe3c218-5b90-4b9e-b326-532732ebc0e6-0013	tadpolehub.c947f1f6-df9e-11e7-bc8d-c6ec0c840608	tadpolehub	capital	RUNNING	2017-12-13T09:45:06+0900	172.22.30.37	<a href="#">Sandbox</a>
ffe3c218-5b90-4b9e-b326-532732ebc0e6-0000	ajpark_marathon-lb.7de04855-df8e-11e7-8183-1efa491721c5	marathon-lb.ajpark	slave_public	RUNNING	2017-12-13T07:48:11+0900	172.22.30.31	<a href="#">Sandbox</a>
ffe3c218-5b90-4b9e-b326-532732ebc0e6-0012	dev_erp-dev.fc42ac17-df8d-11e7-b451-265e3344aa49	erp-dev.dev	park	RUNNING	2017-12-13T07:44:17+0900	172.22.30.19	<a href="#">Sandbox</a>
ffe3c218-5b90-4b9e-b326-532732ebc0e6-0000	ajtotal_marathon-lb.0a433e03-df86-11e7-8183-1efa491721c5	marathon-lb.ajtotal	slave_public	RUNNING	2017-12-13T06:47:41+0900	172.22.30.31	<a href="#">Sandbox</a>
ffe3c218-5b90-4b9e-b326-532732ebc0e6-0009	kibana.79a3fa2e-df1b-11e7-b926-265e3344aa49	kibana	networks	RUNNING	2017-12-12T18:09:35+0900	172.22.30.37	<a href="#">Sandbox</a>
ffe3c218-5b90-4b9e-b326-532732ebc0e6-0013	kibana.129779e4-df1a-11e7-bc8d-c6ec0c840608	kibana	capital	RUNNING	2017-12-12T17:59:54+0900	172.22.30.38	<a href="#">Sandbox</a>
ffe3c218-5b90-4b9e-b326-532732ebc0e6-0000	kibana.d98000c-df0a-11e7-b451-265e3344aa49	kibana	slave	RUNNING	2017-12-12T17:59:54+0900	172.22.30.45	<a href="#">Sandbox</a>

# SMACK 스택: Mesos 1.x과 이후의 방향

- 1.0 버전에서 GPU 지원 시작
- 최근에는 CNCF의 CNI와 CSI 표준 지원
  - Container Network Interface
  - Container Storage Interface
- Docker를 지원하고 있으나 자체 컨테이너 런타임을 제공함
  - UCR: Universal Container Runtime
  - docker/appc 등 다양한 컨테이너 구동

## Four current CNCF Working Groups

### Continuous Integration

Provides infrastructure to hosted projects.

Looks to offer integration testing between projects.

### Networking

Providing a Container Networking Interface (CNI) specification.

Aims for connectivity and portability in cloud native application networking.

### Storage

Providing a Container Storage Interface (CSI) specification.

Aims for portability across cloud orchestration systems.

### Serverless

Educate cloud native developers on serverless architectures.

Determine what the CNCF should do in this space.

Recommend involvement in specifications and projects.

# SMACK 스택: Mesos를 대체할 수 있는 자원관리 방안

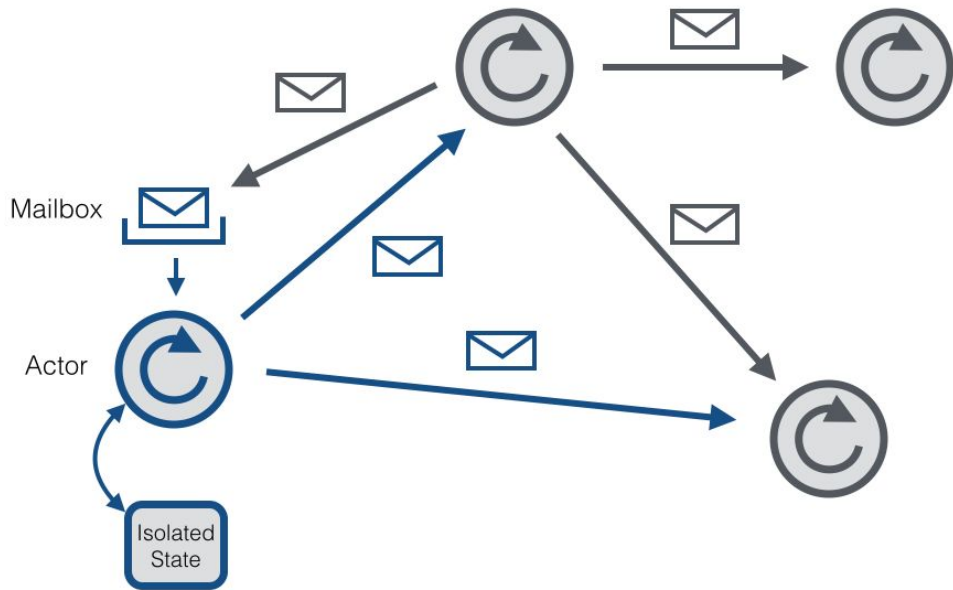
- Apache Hadoop YARN
  - 하둡 2.0에서 도입된 자원 관리 노드
  - 하둡 3.0에서는 도커 컨테이너를 배포하는 기능이 포함 됨
- Kubernetes
  - 구글이 개발한 컨테이너 관리시스템 -> 오픈소스로 변경
  - 도커 컨테이너를 사용하여 마이크로 서비스 아키텍처를 구현하기 위한 환경 제공
- Docker Swarm
  - Docker inc(도커 개발사)에서 개발
  -



# SMACK 스택: Akka

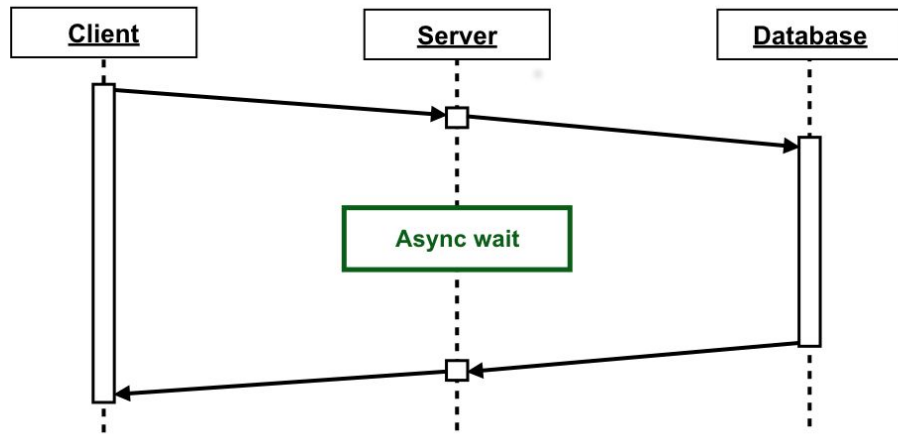
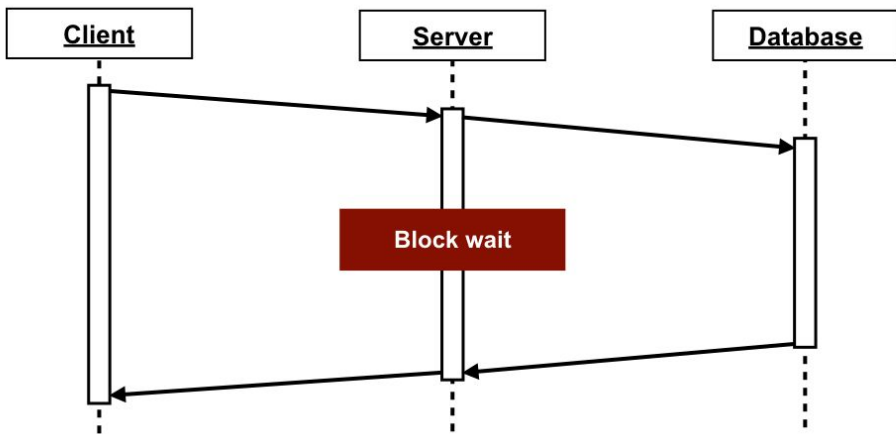


- Actor 모델의 구현체
  - 메시지 기반의 통신
  - 동시성 (concurrency)
  - 내고장성 (fault-tolerant)
  - 고성능
  - Non blocking API
- 하나의 작은 일을 하는 actor가 네트워크를 이루어서 복잡한 일을 수행



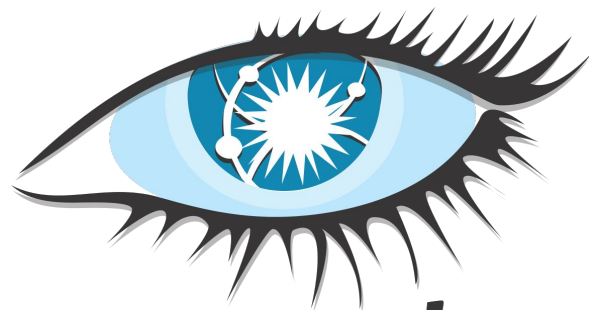
# SMACK 스택: Akka의 대안

- Quasar: 또 다른 Actor 모델 구현체
  - 경량의 thread로 구성
  - JVM 언어 지원(scalar 제외)
  - blocking API: 충분히 가볍고 빠르기 때문에 가능

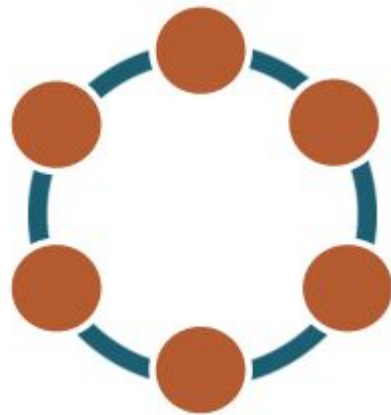


# SMACK 스택: Cassandra

- Facebook에서 개발 시작
- 대용량 분산 데이터베이스
- 높은 수준의 내고장성(fault-tolerant)을 제공
  - SPOF(single point of failure)가 없음
  - Master같은 특별한 기능을 하는 노드가 없음
- Scale out이 가능한 구조
  - 노드를 늘리면 저장 공간과 성능이 선형적으로 증가
  - Hash Ring 구조
- 기존 시스템과 연계가 쉬움
  - SQL과 유사한 CQL 제공
  - 비정형, 반정형, 정형 데이터를 모두 수용할 수 있음

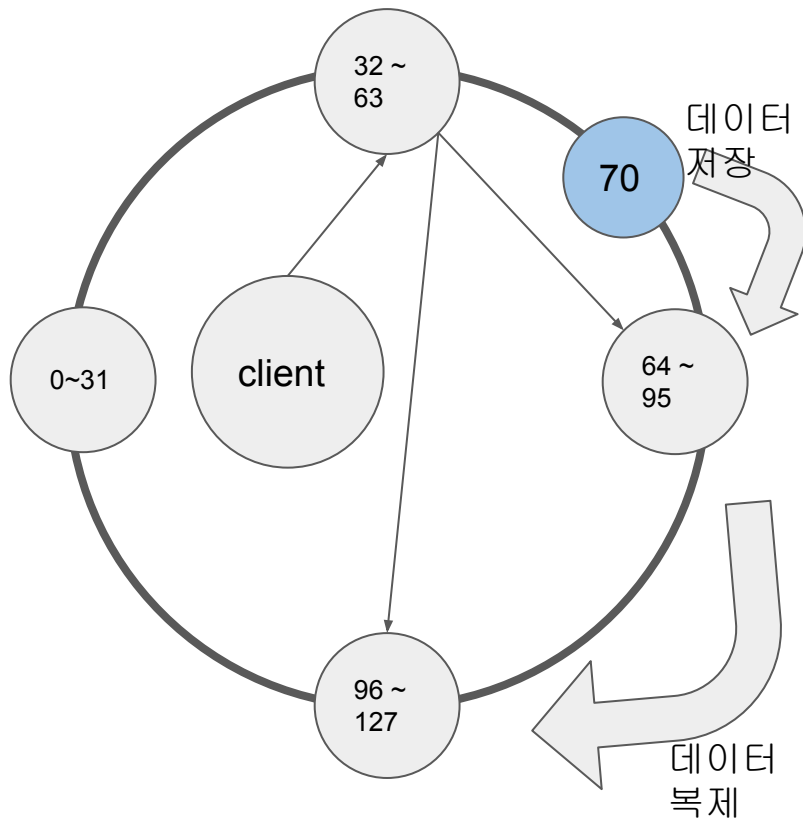


**cassandra**



# SMACK 스택: Cassandra의 데이터 저장 모델

- Hash ring 구조
  - 키를 hash 하여 저장될 노드를 선택
  - 각 노드는 hash 범위의 일정 부분을 저장
  - 각 노드는 어느 노드가 어느 범위의 hash 를 담당하는지 알고 있음
  - Murmurhash 64bit
- Hash값은 노드를 결정하는 것에만 사용
  - 데이터는 실제 key로 저장
  - hash collision을 방지
  - 최악의 경우에도 클러스터내의 데이터 쓸림 현상 정도 발생



# SMACK 스택: Cassandra를 대체할 있는 저장소

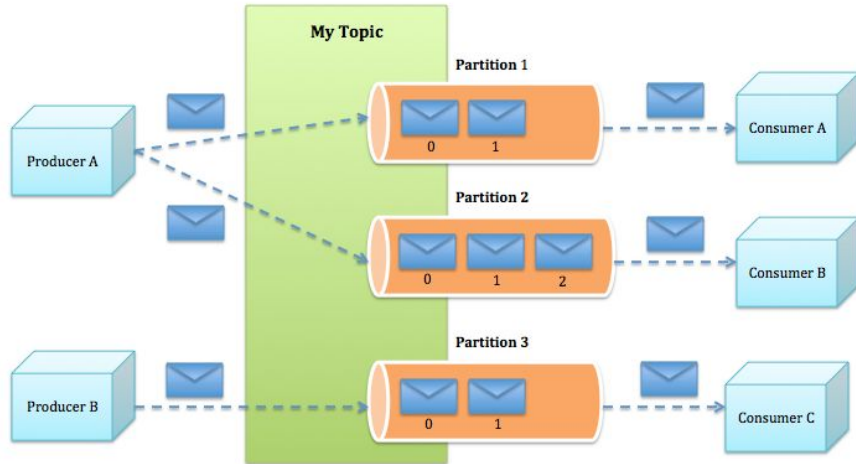
- Apache HBase
  - HDFS 등의 분산 파일시스템을 기반으로 하여 고속의 레코드 질의 기능 제공
- Apache Hadoop HDFS
  - 대용량의 분산 파일 시스템
  - 직접 사용되기보다는 다른 저장소의 backend 혹은 cold 백업으로 사용
- Druid
  - 시계열 데이터의 집계 연산에 특화된 저장소
  - 집계 연산을 자체 수행하기 때문에 처리기(spark 등)의 부하를 줄일 수 있음
- RDB 들
  - MySQL, PostgreSQL, Oracle Exadata 등.
  - 널리 알려진 SQL을 사용하는 장점이 있으나 빅데이터 처리가 힘들거나(mysql) 가격이 비쌈(oracle).

# SMACK 스택: Cassandra의 이후 방향

- 오래된(2008~) 프로젝트로 충분히 성숙됨
- 새로운 기술의 도입보다는 안정적인 운영을 위한 개선들이 이루어 질 것

# SMACK 스택: Kafka

- LinkedIn에서 개발 시작
- 분산 대용량 메세지 큐
  - 대용량/ 빠름, 토픽
  - 데이터를 보관하는 용도로도 사용할 수 있음: 영구 보관은 어려움.
- Publish/Subscribe 기반의 동작
  - publisher: 데이터를 발행하는 프로세스
  - subscriber: 데이터를 구독하는 프로세스
- 특이하게 디스크 기반
  - 디스크를 잘 쓰면 충분한 성능을 낼 수 있음: 저비용 고효율
  - 리눅스의 **kernel/user** 공간 컨텍스트 스위칭을 줄여서 시간을 절약: **zero copy** 기법



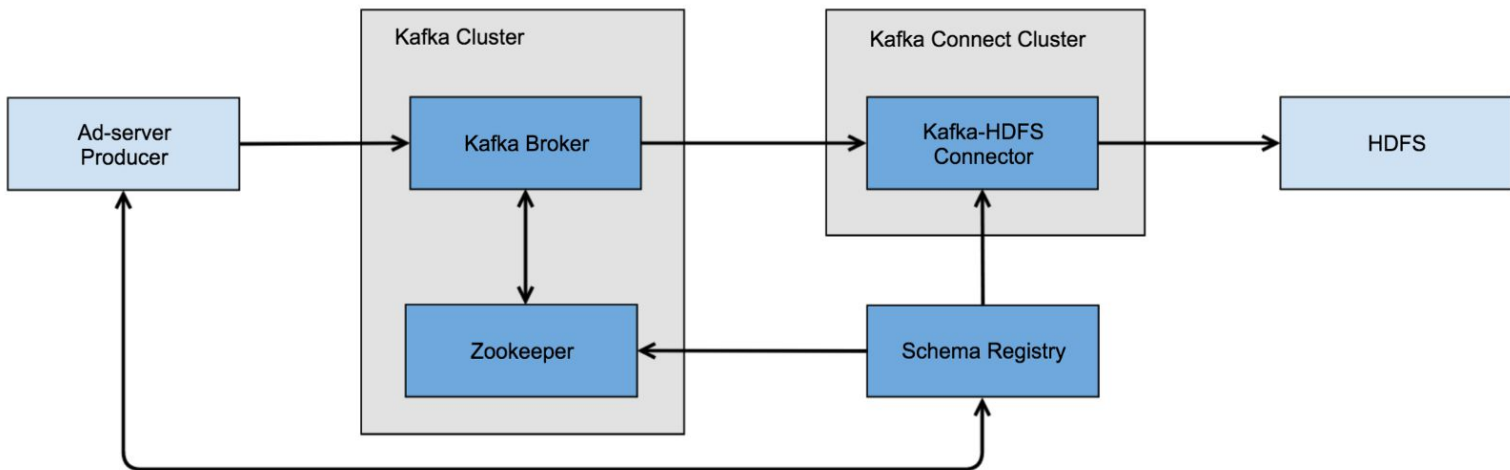
# SMACK 스택: 입/출력 버퍼로서의 Kafka

- 외부에서 실시간 입력을 모두 **Kafka**로 전달
  - 내부 처리 시스템의 부하에 따라서 데이터의 손실 없이 사용할 수 있음
  - 데이터가 일부 유실되더라도 **Kafka** 보관되어 있는 데이터로 **replay** 할 수 있음.
- 내부의 출력을 모두 **Kafka**로 전달
  - 내부의 다양한 시스템이 복잡한 외부 연결을 각각 구현할 필요 없음
  - 외부로 데이터를 전달하는 프로그램은 **Kafka**에서 데이터를 가져와서 외부 형식에 맞게 가공하여 전달



# SMACK 스택: 데이터 교환 표준으로서의 Kafka

- 큰 회사/ 조직의 경우 내부 데이터 교환을 **Kafka**를 기반으로 구성하기도 함.
- 조직은 데이터를 주고받는 방법을 정하기 위해서 싸울 필요가 없음
  - 웹서비스 팀과 분석 팀
  - 웹서비스 팀이 **kafka**로 데이터를 발행하면 분석 팀은 구독하면 끝.
  - 최근에는 **schema registry**를 도입하여 데이터의 형식까지 공유: 자동으로 **deserialize** 됨.



# SMACK 스택: Kafka 1.x과 향후 방향

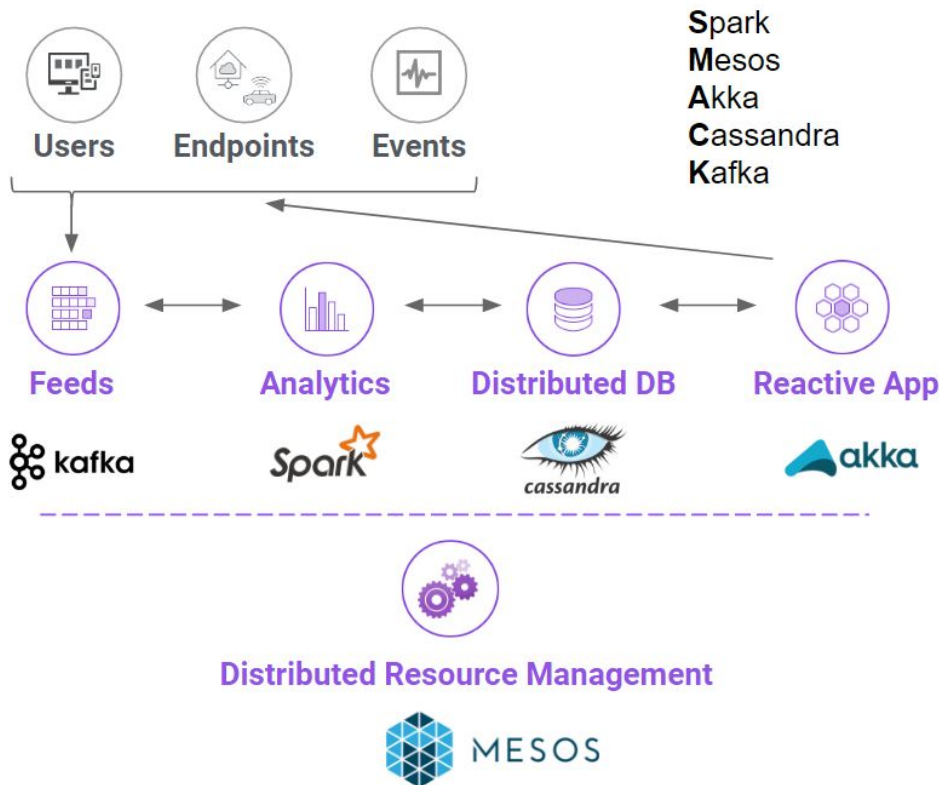
- 1.0 버전 직전에 Kafka Stream과 Kafka Connector를 추가
  - Kafka Stream: 토픽에 저장된 데이터에서 바로 실시간 처리/분석을 하는 기능
  - Kafka Connector: 데이터의 입/출력을 통합한 기능
  - 최초에는 작은 규모에서 도입될 것
- KSQL이라는 스트리밍 SQL을 [confluent.io](https://confluent.io)에서 제공
  - Kafka의 상업용 버전을 판매하는 회사
- 메시지 큐에서 벗어나 스트리밍 플랫폼으로 영역 확장 시도

# SMACK 스택: Kafka를 대체할 수 있는 메시지 큐

- 빅데이터 영역에서는 사실상 없음.
  - 웹, 캐싱 등의 영역에서는 redis, memcached 및 상용 제품들 존재

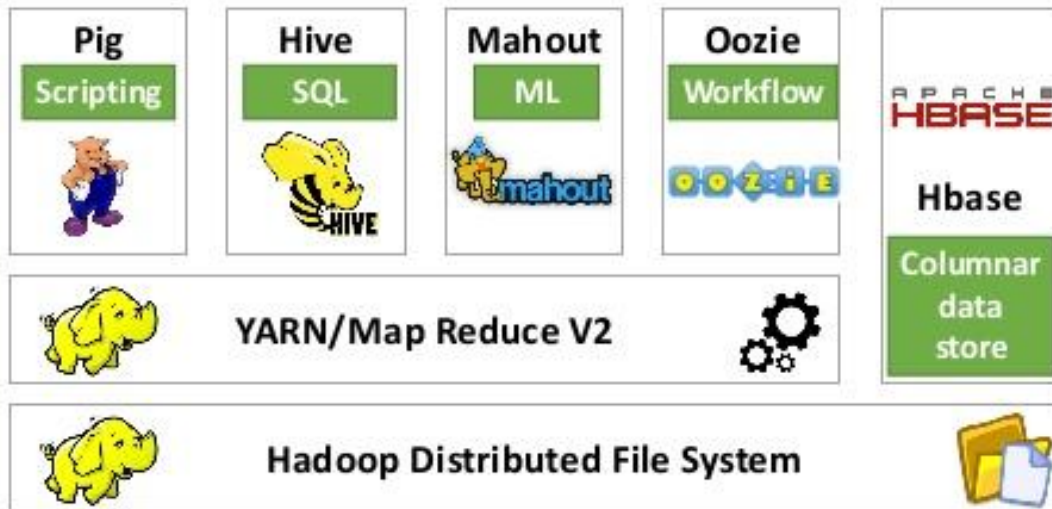
# SMACK 스택: Recap

- 자원관리: Mesos
- 메시지 큐: Kafka
- 처리: Spark
- 저장: Cassandra
- 연결: Akka



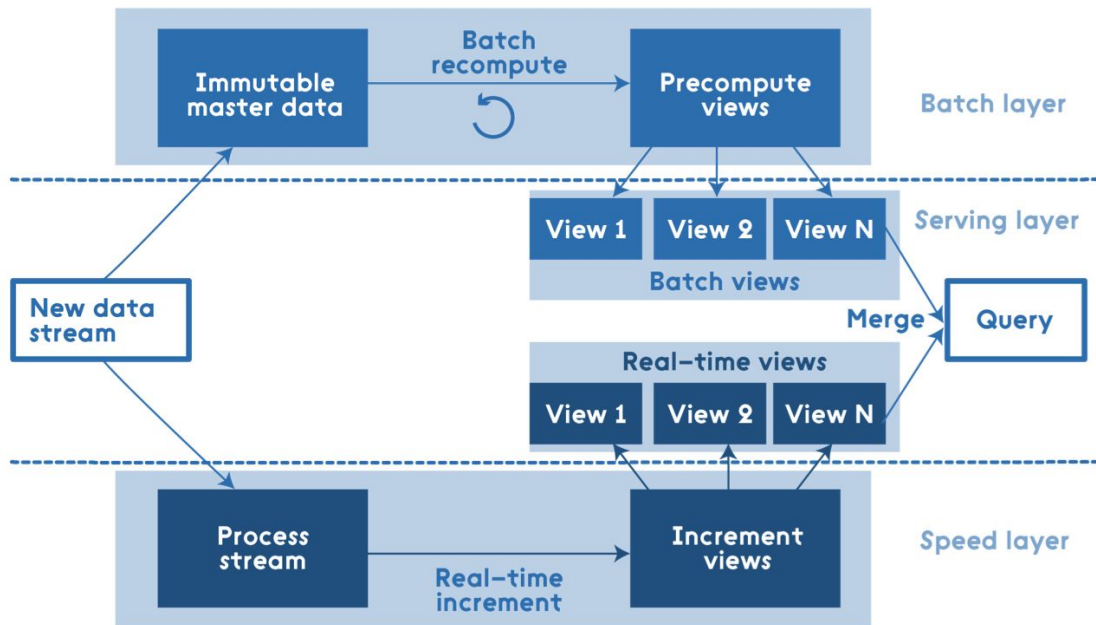
# 초기 빅데이터 분석 인프라

- 자원관리: Hadoop YARN
- 메시지 큐: 없음
- 처리: Hive(MR) 또는 Pig(MR)
- 저장
  - 파일: Hadoop HDFS
  - 실시간 질의: HBase
- 연결: 없음



# Lambda 스택:

- 자원관리: Hadoop YARN
- 메시지 큐: Kafka
- 처리
  - 배치: Hive(MR)
  - 실시간: Spark
- 저장
  - 파일: Hadoop HDFS
  - 실시간 질의: Cassandra
- 연결: 없음



# 빠진 퍼즐 조각들

- 데이터 수집 방법/ 프레임워크
  - Fluentd, logstash,
- 데이터 시각화 방법/ 프레임워크
  - Apache Superset
  - Metabase
- 데이터 관리(기간, 크기, 형식, 접근 권한 등) 방법
  - Apache Ranger

# Big Data Landscape

## Log Data Apps

splunk>

loggly

sumologic

## Vertical Apps

PREDICTIVE  
POLICING

bloomreach  
GET FOUND.

## Business Intelligence

ORACLE | Hyperion

SAP Business Objects

Microsoft | Business Intelligence

IBM COGNOS  
sas MicroStrategy

GoodData

## Analytics and Visualization

+ a b l e a u Palantir

METAMARKETS

TERADATA ASTER

visual.ly KARMASPHERE

EMC GREENPLUM.

platfora ClearStory  
New You See It

## Data Providers

GNIP

DATASIFT

SPACE  
CURVE

INRIX

## Analytics Infrastructure

Hortonworks

cloudera

MAPR TECHNOLOGIES VERTICA  
An HP Company

## Operational Infrastructure

Couchbase

TERADATA

10gen | the MongoDB company

HADAPT

## Infrastructure As A Service

amazon  
web services

infochimps

Windows Azure

## Structured Databases

ORACLE

MySQL

Microsoft SQL Server

PostgreSQL

memsql

## Technologies

hadoop

Map Reduce

APACHE  
HBASE

Cassandra



## BIG DATA LANDSCAPE 2017



# Opensource

OPEN

## FRAMEWORK



## QUERY / DATA FLOW



## DATA ACCESS



## COORDINATION



## STREAMING



## STAT TOOLS



SOURCE

## AI / MACHINE LEARNING / DEEP LEARNING



## SEARCH



## LOG ANALYSIS



## VISUALIZATION



## COLLABORATION



## SECURITY



# 마치며.

- **SMACK** 스택은 빅데이터 분석 인프라의 기본적인 구성을 가지고 있다.
  - 일부 구성 요소는 대안이 있지만, 그렇지 않은 것도 있음.
  - 자원 관리는 중요한 문제: 사용률을 얼마나 높일 것인가.
  - 기반이 되는 **SW**일수록 다른 영역으로 확장 시도: 특히 **Kafka**
- 정해진 스택 사용 **vs** 상황에 따라 스택을 결정
  - 범용적인 스택은 낭비가 있을 여지가 있음
  - 상황에 알맞는 스택은 비용이 문제(학습 비용, 개발자 구하기, 운영 비용 등)