# Toronto Apache Spark (TAS)

## Building Enterprise Data Lake Solution using Spark and SequoiaDB

Tao Wang

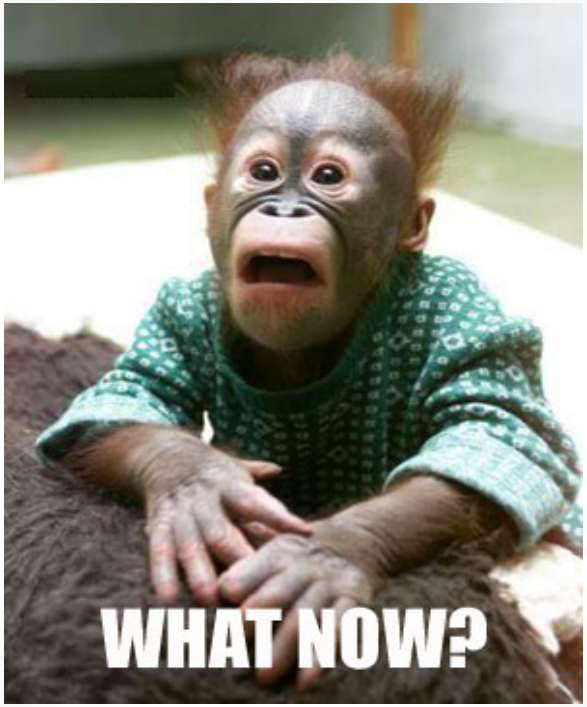
# Outline

What

Why

How

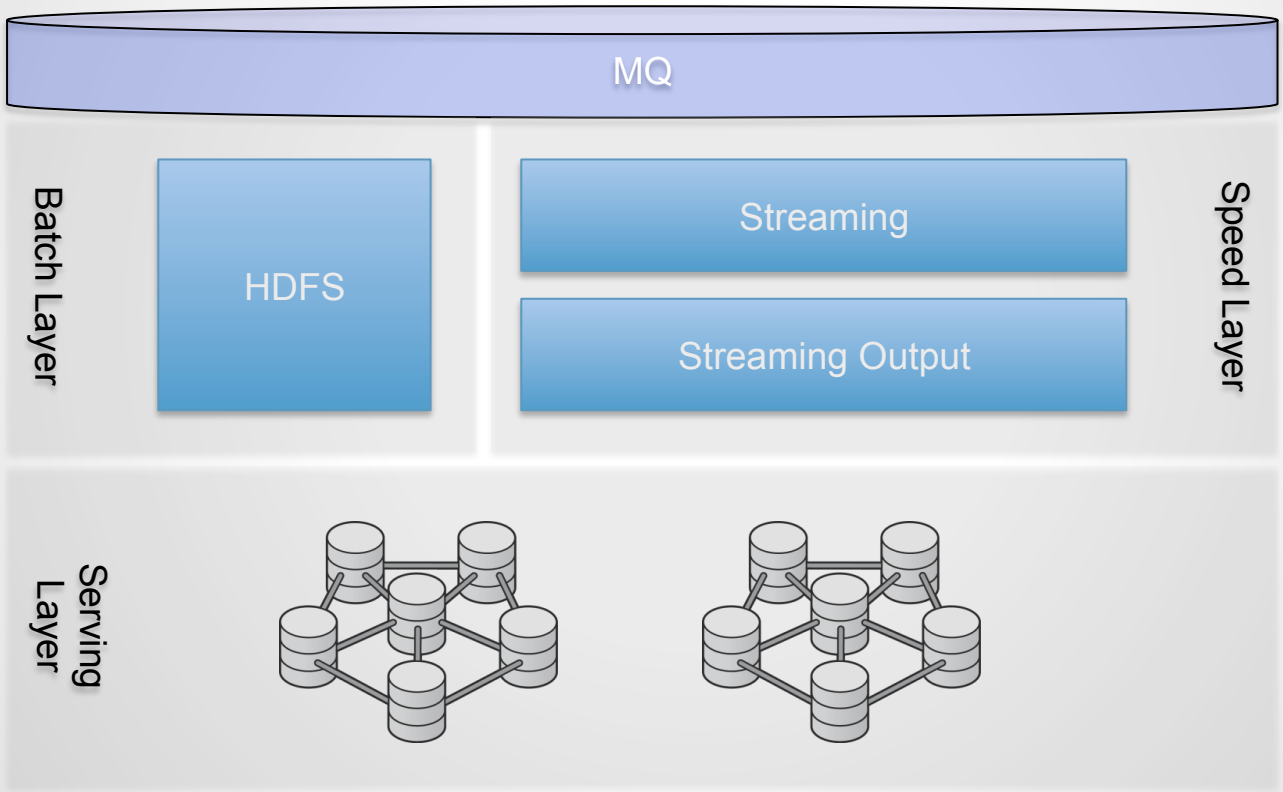
Where

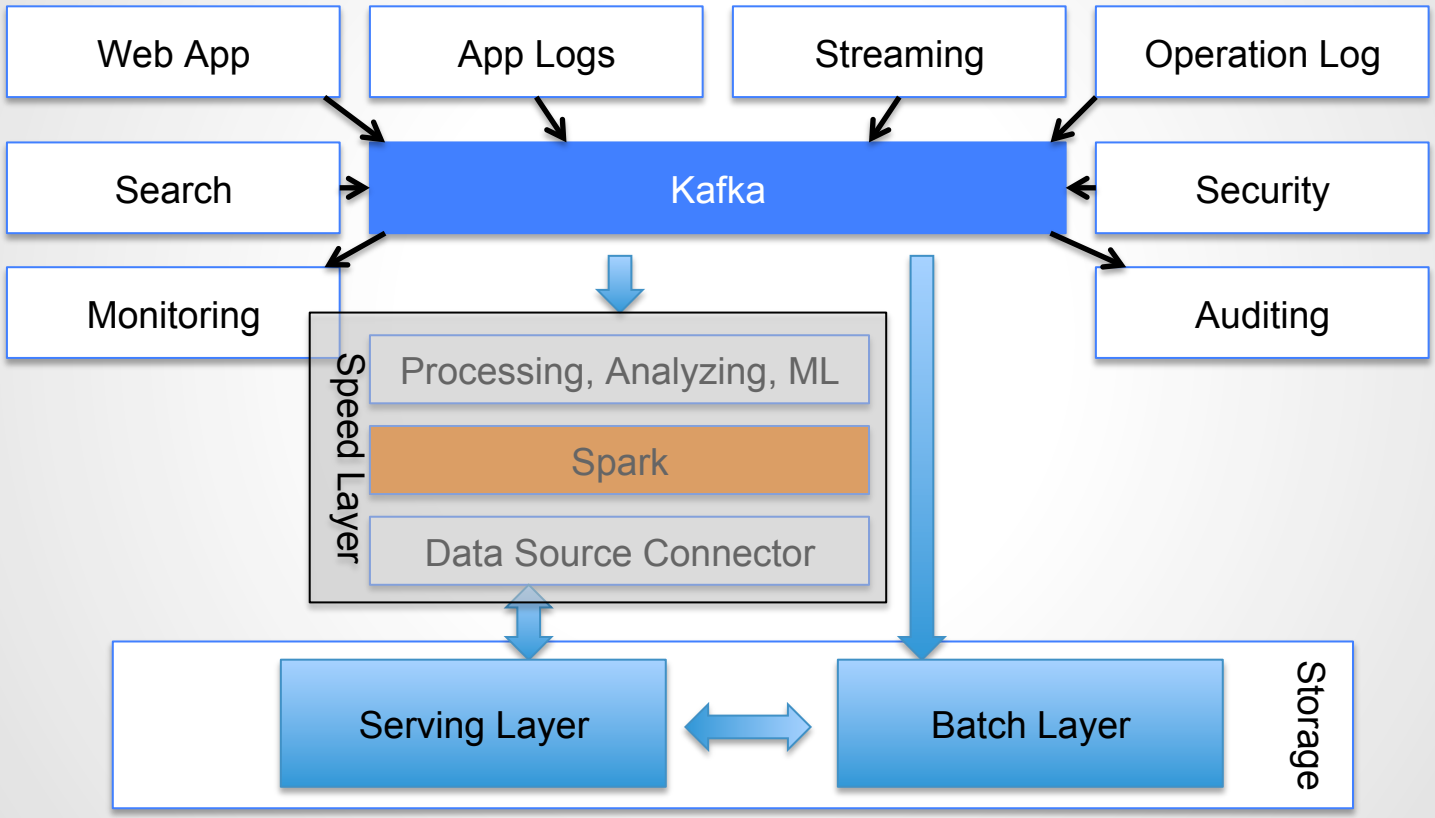

WHAT NOW?

# Lambda Architecture

A data-processing architecture designed to handle *massive quantities* of data by taking advantage of both batch and stream processing methods. （wikipedia）

# Lambda Architecture

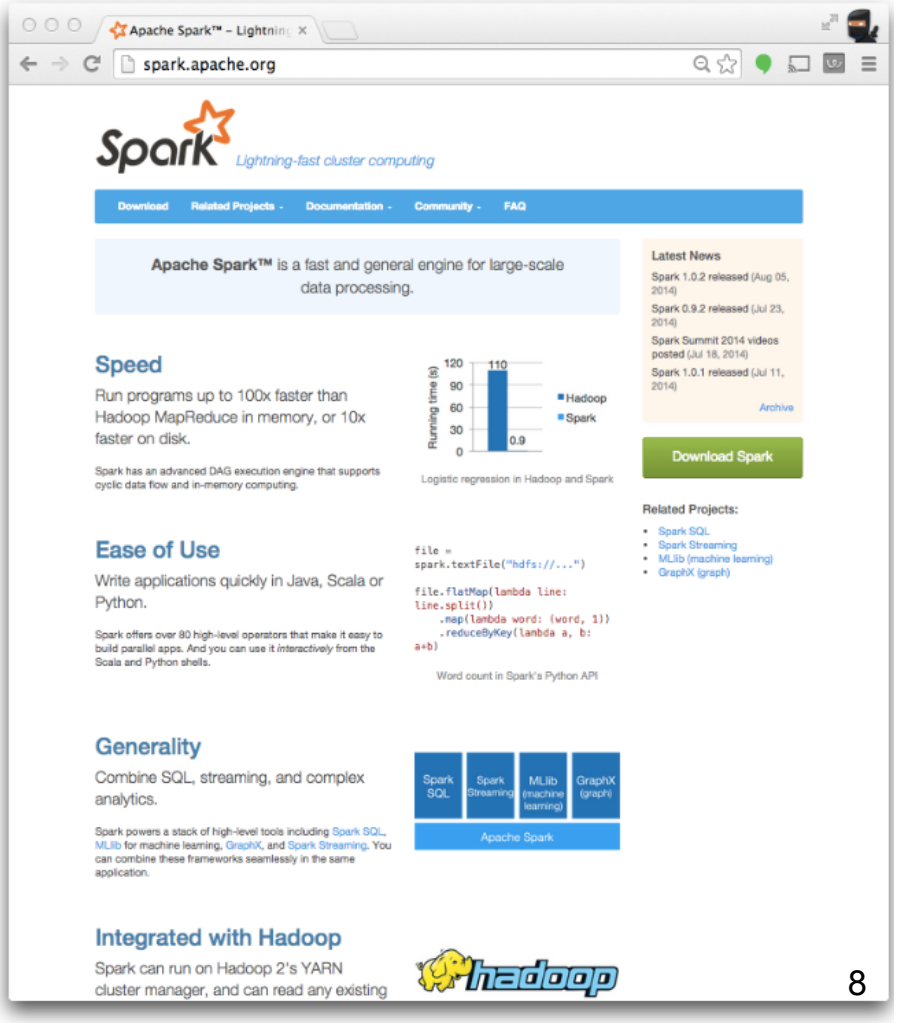



Web App
App Logs
Streaming
Operation Log
Search
Kafka
Security
Monitoring
Auditing
Speed Layer
Processing, Analyzing, ML
Spark
Data Source Connector
Serving Layer
Batch Layer
Storage

# kafka

- High Throughput Distributed Messaging
- Decouples Data Pipelines
- Handles Massive Data Load
- Support Massive Number of Consumers
- Distribution & partitioning across cluster nodes
- Automatic recovery from broker failures

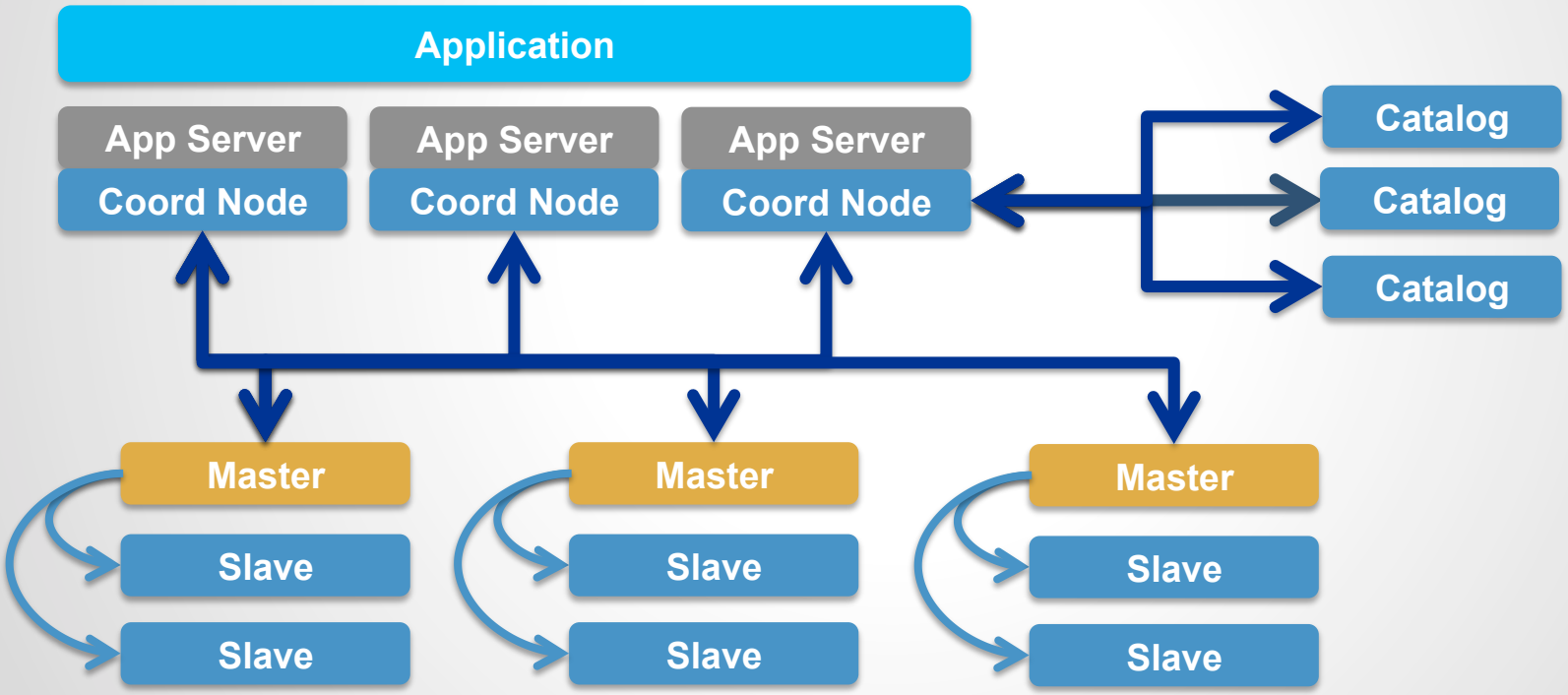

- Fast, distributed, scalable and fault tolerant cluster compute system
- Enables Low-latency with complex analytics
- Developed in 2009 at UC Berkeley AMPLab, open sourced in 2010, and became a top-level Apache project in February, 2014

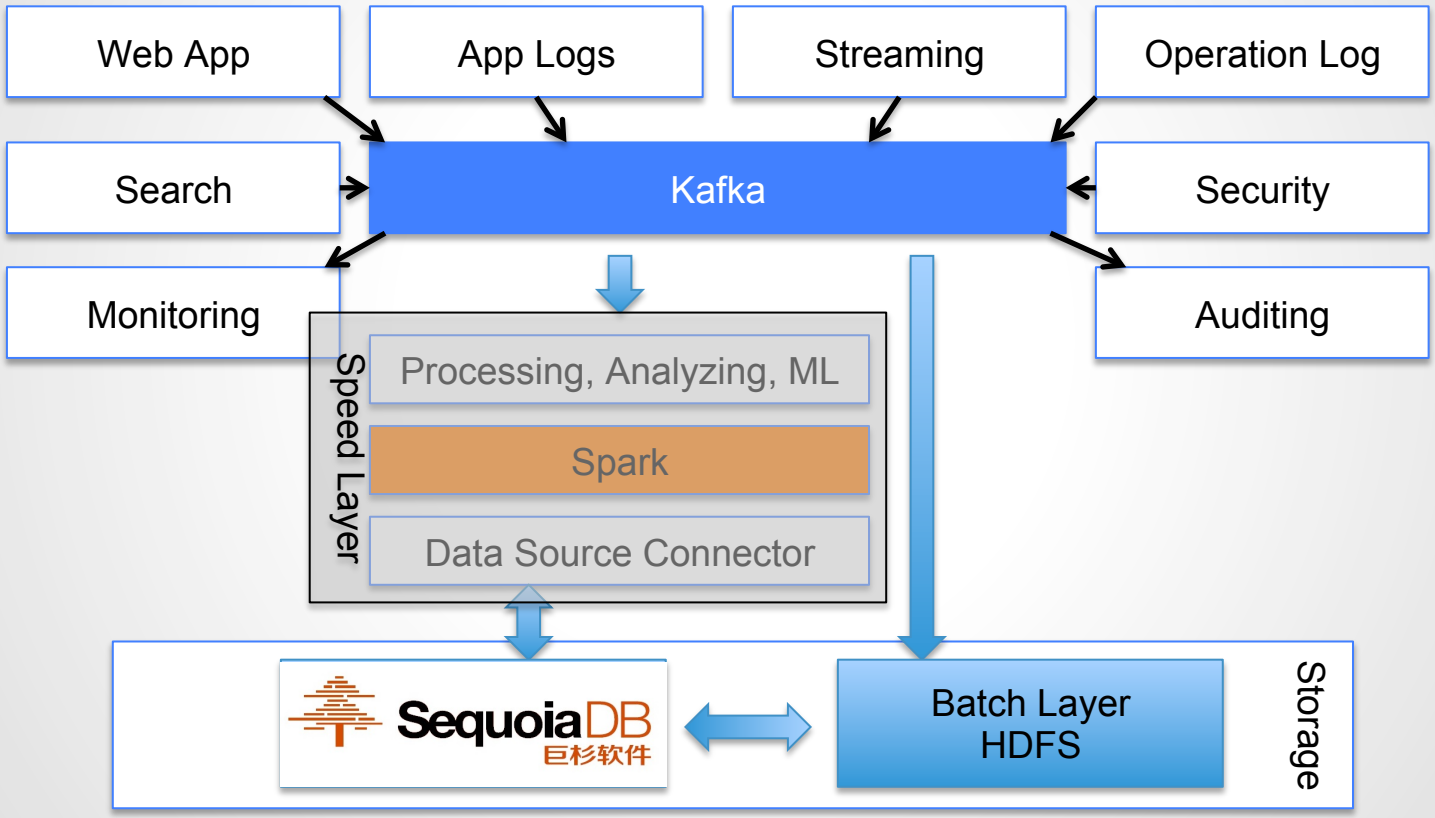


SequoiaDB 巨杉软件

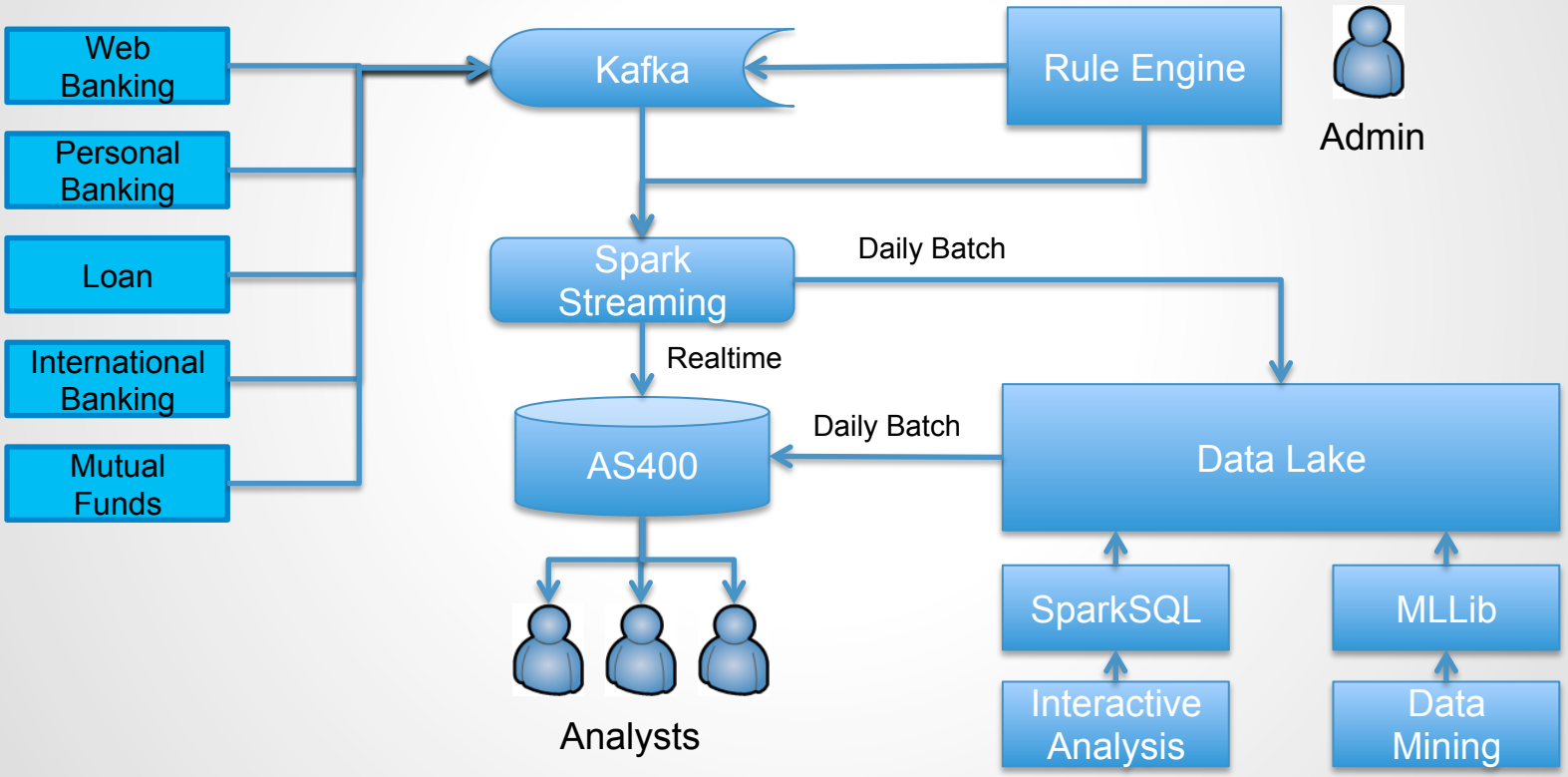
SequoiaDB is a document-oriented NOSQL database that supports JSON transaction processing and SQL. The database is horizontally scalable and provides high performance.

# Storage Layer： SequoiaDB



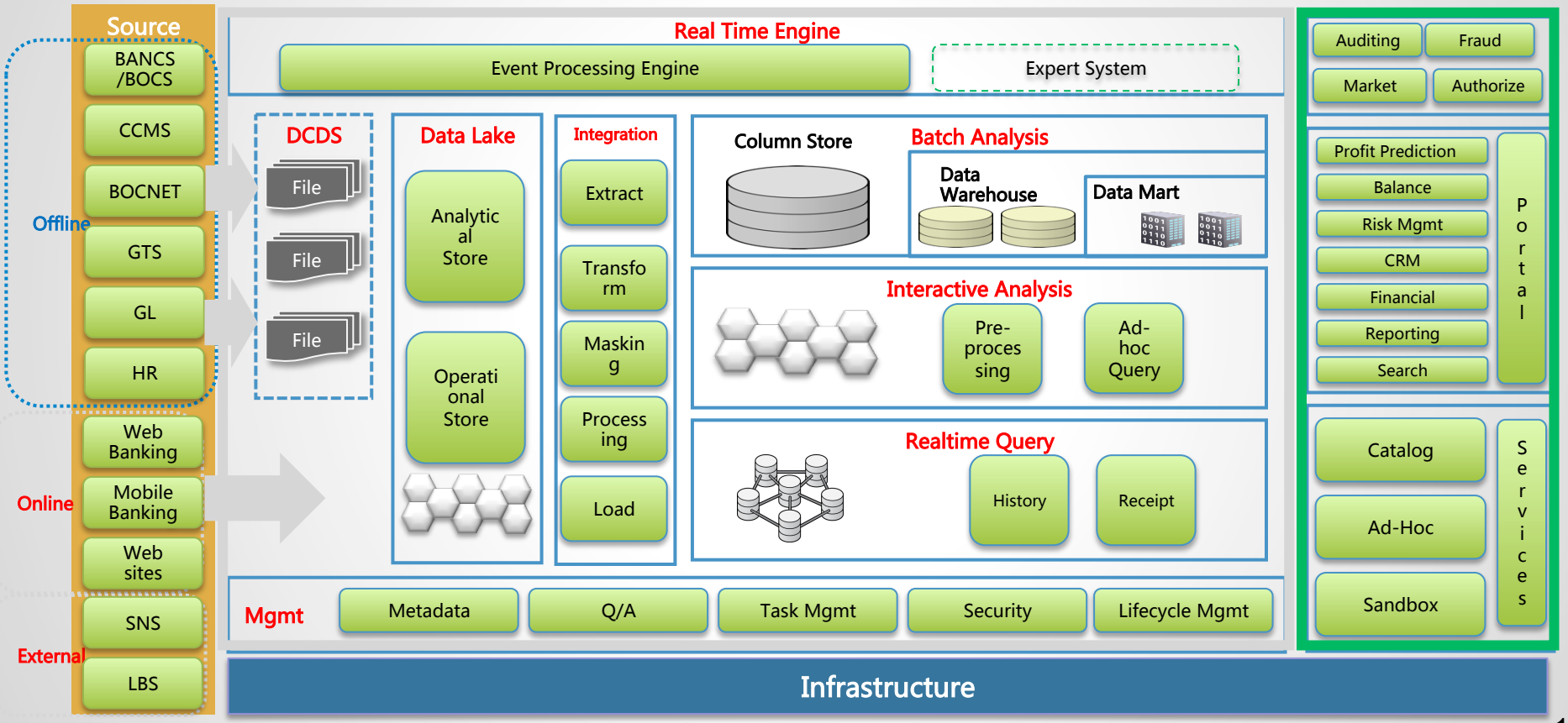

Web App
App Logs
Streaming
Operation Log
Search
Kafka
Security
Monitoring
Auditing
Speed Layer
Processing, Analyzing, ML
Spark
Data Source Connector
SequoiaDB
巨杉软件
Batch Layer
HDFS
Storage

# Real-time Fraud Detection




WHY???
WHY GOD WHY!!!

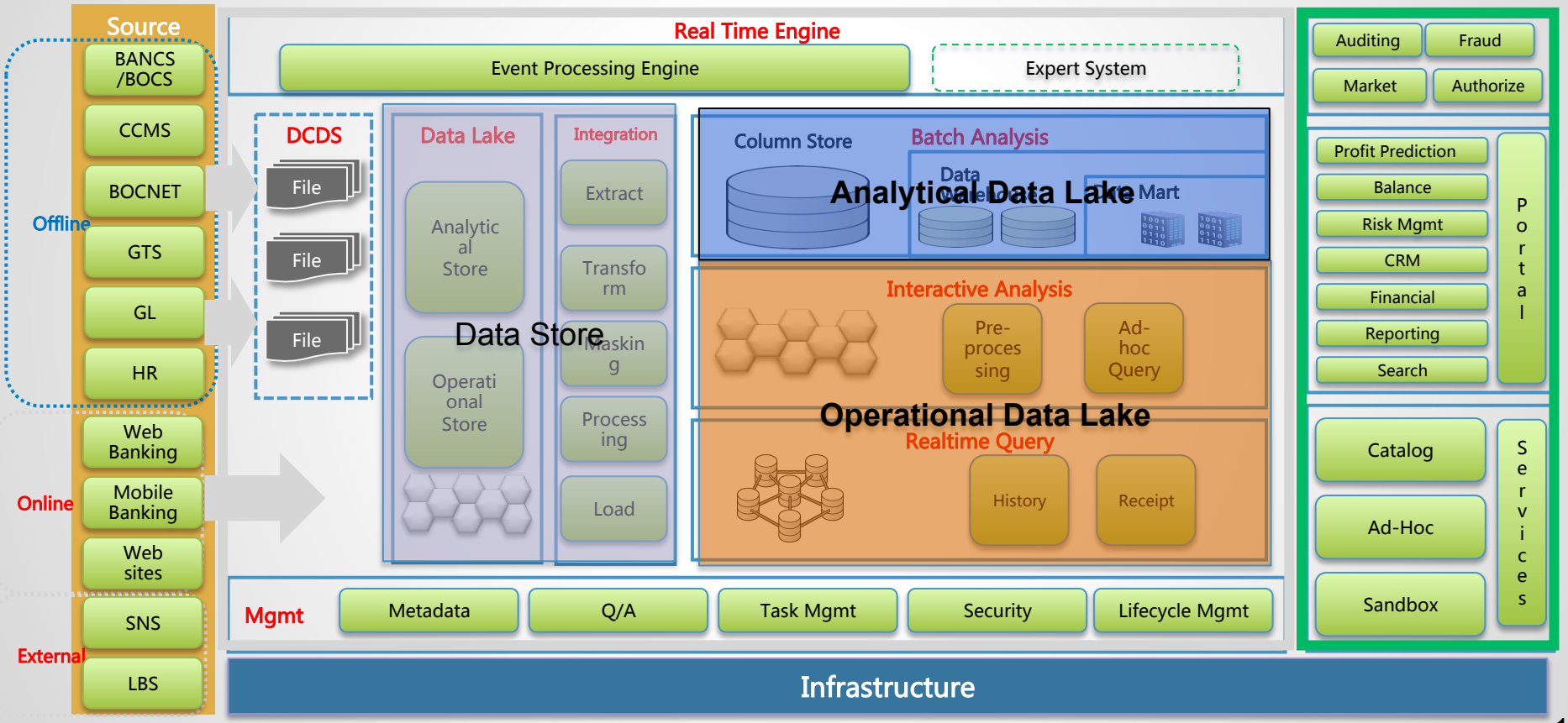
# Enterprise Data Lake Architecture



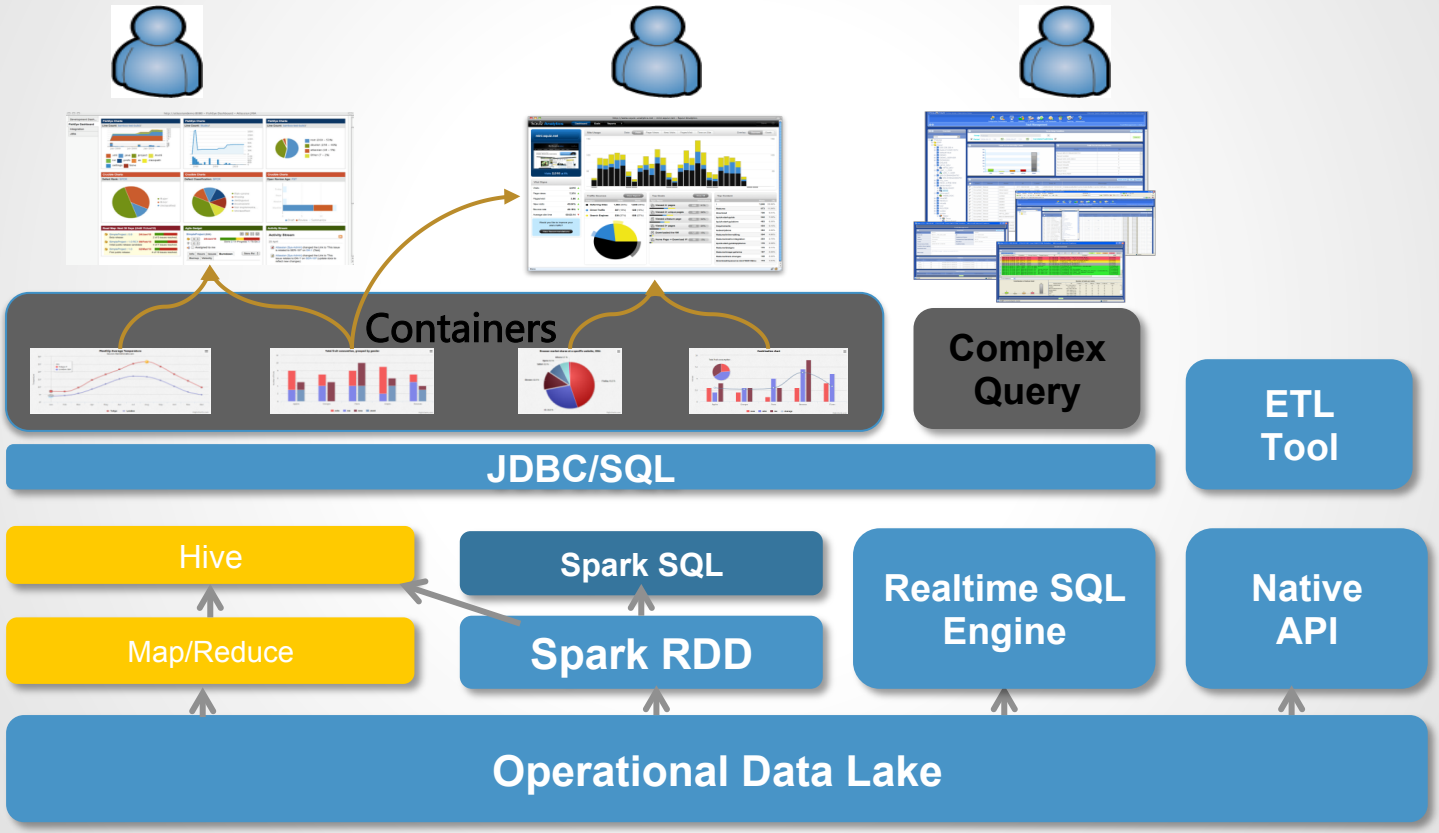
# Data Lake Requirement

- Joint solution of real-time and batch jobs
- Support reporting, OLAP and ad-hoc queries
- HDFS is for reporting system ONLY
- Distributed Database is mandatory for Operational Data Store that supports mixed workload

# Data Lake Architecture



# Mixed Workload Support



# Why Spark

Everybody likes SQL

Streaming is critical for Lambda architecture

Embedded libs/utilities like MLlib and GraphX

Able to handle unstructured data


HOW?

# Spark-SequoiaDB Connector

databricks PRODUCT SPARK SOLUTIONS CUSTOMERS COMPANY BLOG RESOURCES Training Careers

COMPANY
All Posts
Partners
Events
Press Releases

DEVELOPER
All Posts
Spark
Spark SQL
Spark Streaming
MLlib
Spark Summit

Search Blog

Subscribe

## Guest blog: SequoiaDB Connector for Apache Spark

August 3, 2015 | by Tao Wang

This is a guest blog from Tao Wang at SequoiaDB. He is the co-founder and CTO of SequoiaDB, leading its long-term technology vision, and is responsible for the leadership of advanced technology incubations. SequoiaDB is a JSON document-oriented transactional database.

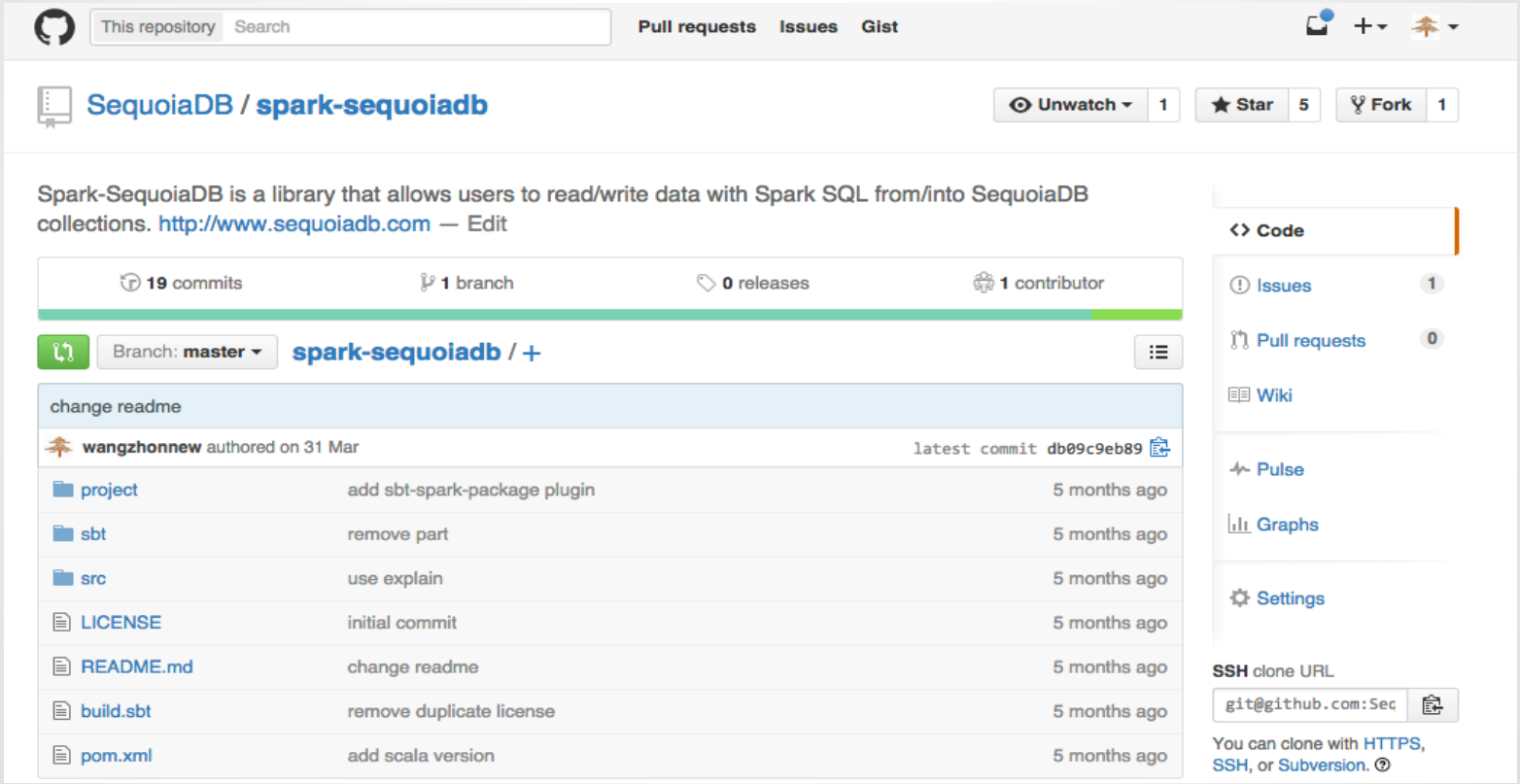
### Why We Chose Spark

SequoiaDB is a NoSQL database that has the capability to replicate data on different physical nodes and allows users to specify which "copy of data" that the application should access. It is capable of running analytical and operational workloads simultaneously on the same cluster with minimal I/O or CPU contention.

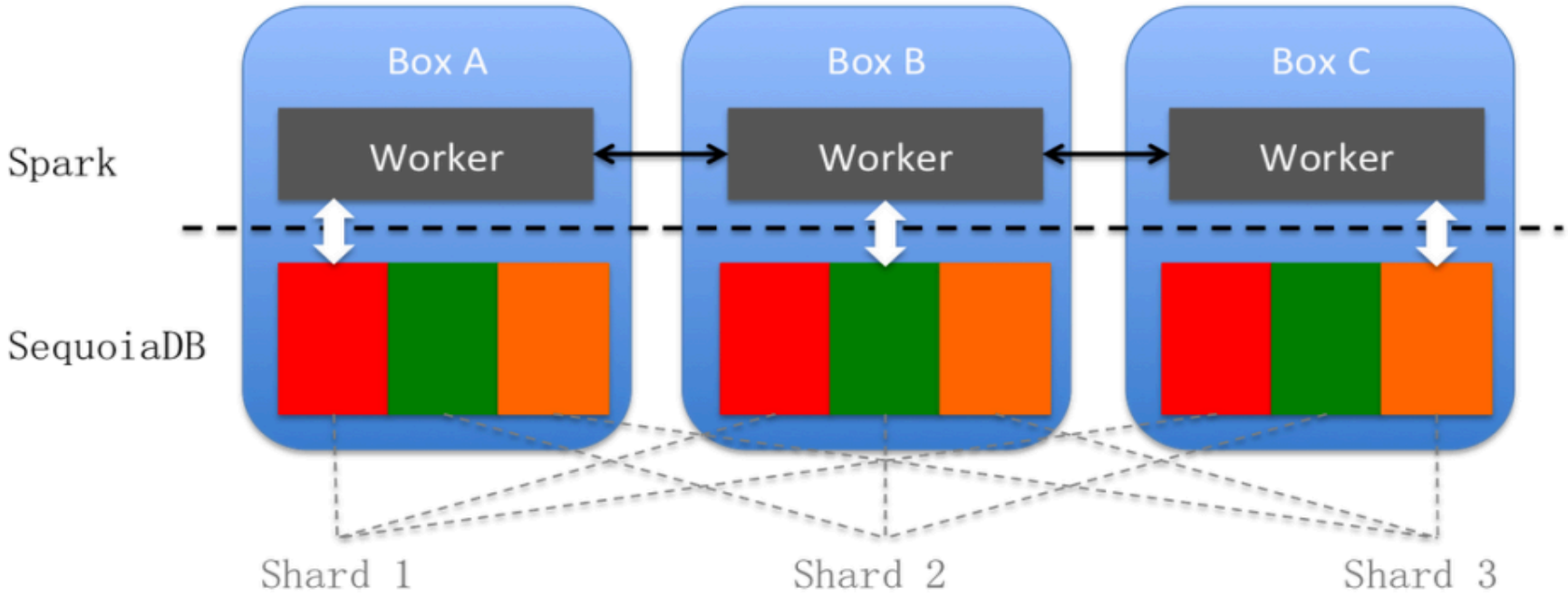
The joint solution of Apache Spark and SequoiaDB allows users to build a single platform such that a wide variety of workloads (e.g., interactive SQL and streaming) can run together on the same physical cluster.

### Making SequoiaDB Work with Spark: The Spark-SequoiaDB Connector

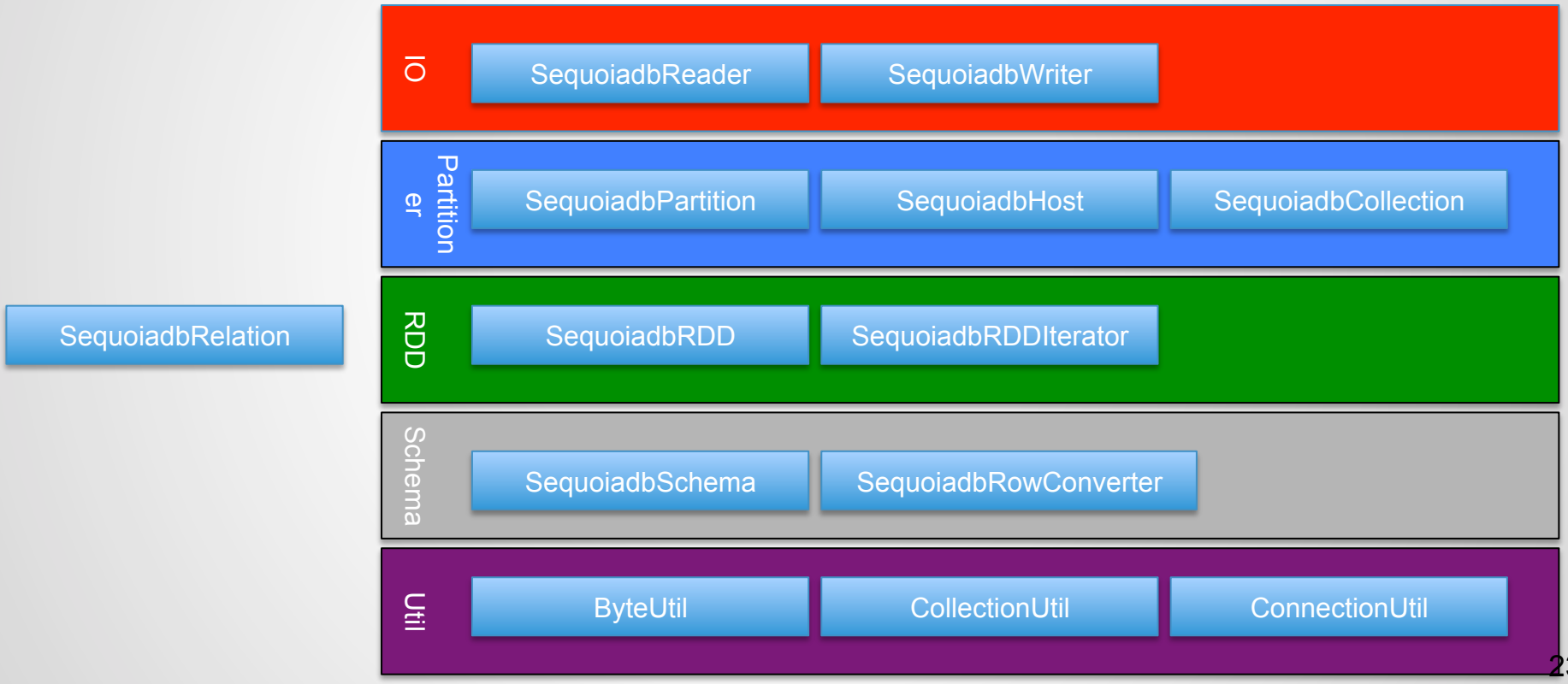
# Spark-SequoiaDB Connector



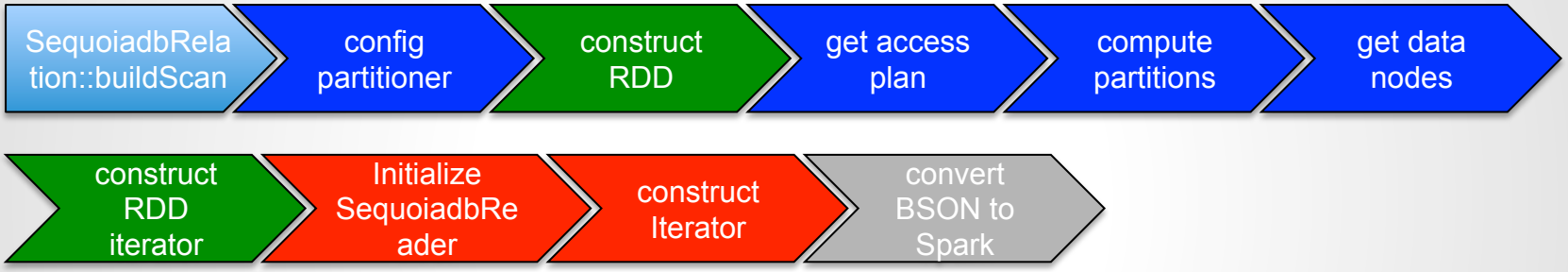
# Spark/SequoiaDB Deployment



# Connector Architecture



# Query Process



# SparkSQL + Connector

```
sqlContext.sql("CREATE temporary table org_department ( deptno string, deptname s
tring, mgrno string, admrdept string, location string ) using com.sequoiadb.spark
 OPTIONS ( host 'host-60-0-16-2:50000', collectionspace 'org', collection 'depart
ment', username 'sdbreader', password 'sdb_reader_pwd')")
res2: org.apache.spark.sql.DataFrame = []

sqlContext.sql("CREATE temporary table org_employee ( empno int, firstnme string,
 midinit string, lastname string, workdept string, phoneno string, hiredate date,
 job string, edlevel int, sex string, birthdate date, salary int, bonus int, comm
 int ) using com.sequoiadb.spark OPTIONS ( host 'host-60-0-16-2:50000', collectio
nspace 'org', collection 'employee', username 'sdb_reader', password 'sdb_reader_
pwd')")
res3: org.apache.spark.sql.DataFrame = []

sqlContext.sql("select * from org_department a, org_employee b where a.deptno='D1
1'").collect().take(3).foreach(println)
[D11,MANUFACTURING SYSTEMS,000060,D01,null,10,CHRISTINE,I,HAAS,A00,3978,null,PRES
,18,F,null,152750,1000,4220]
[D11,MANUFACTURING SYSTEMS,000060,D01,null,20,MICHAEL,L,THOMPSON,B01,3476,null,MA
NAGER,18,M,null,94250,800,3300]
[D11,MANUFACTURING SYSTEMS,000060,D01,null,30,SALLY,A,KWAN,C01,4738,null,MANAGER,
20,F,null,98250,800,3060]
```


WHERE?

# Real-time fraud detection

## Before

- Hard coded data processing architecture
- Difficult to manage
- Lack of ability to horizontally scale

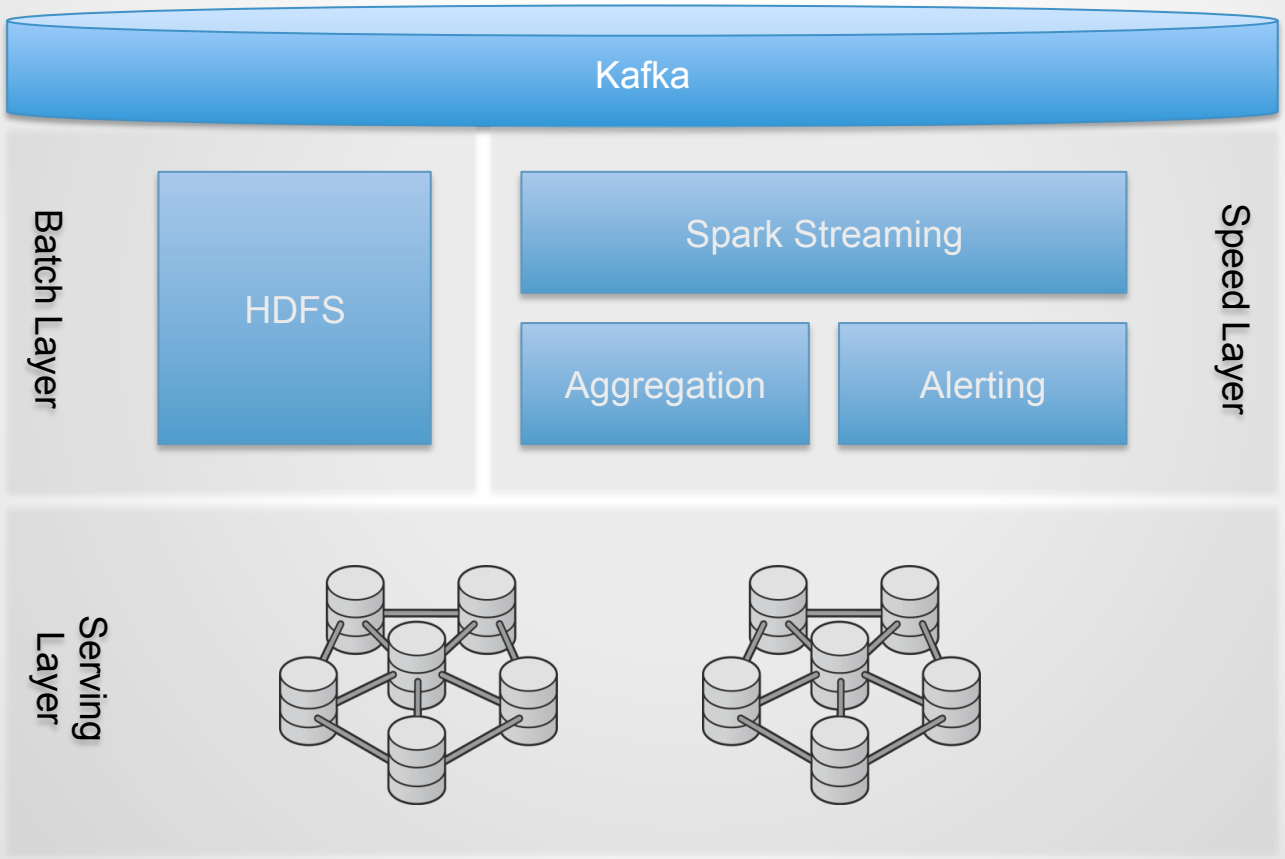
## Now

- Horizontally scale out
- Dynamic schema for data model changes
- Support ad-hoc and real time queries
- Streaming input

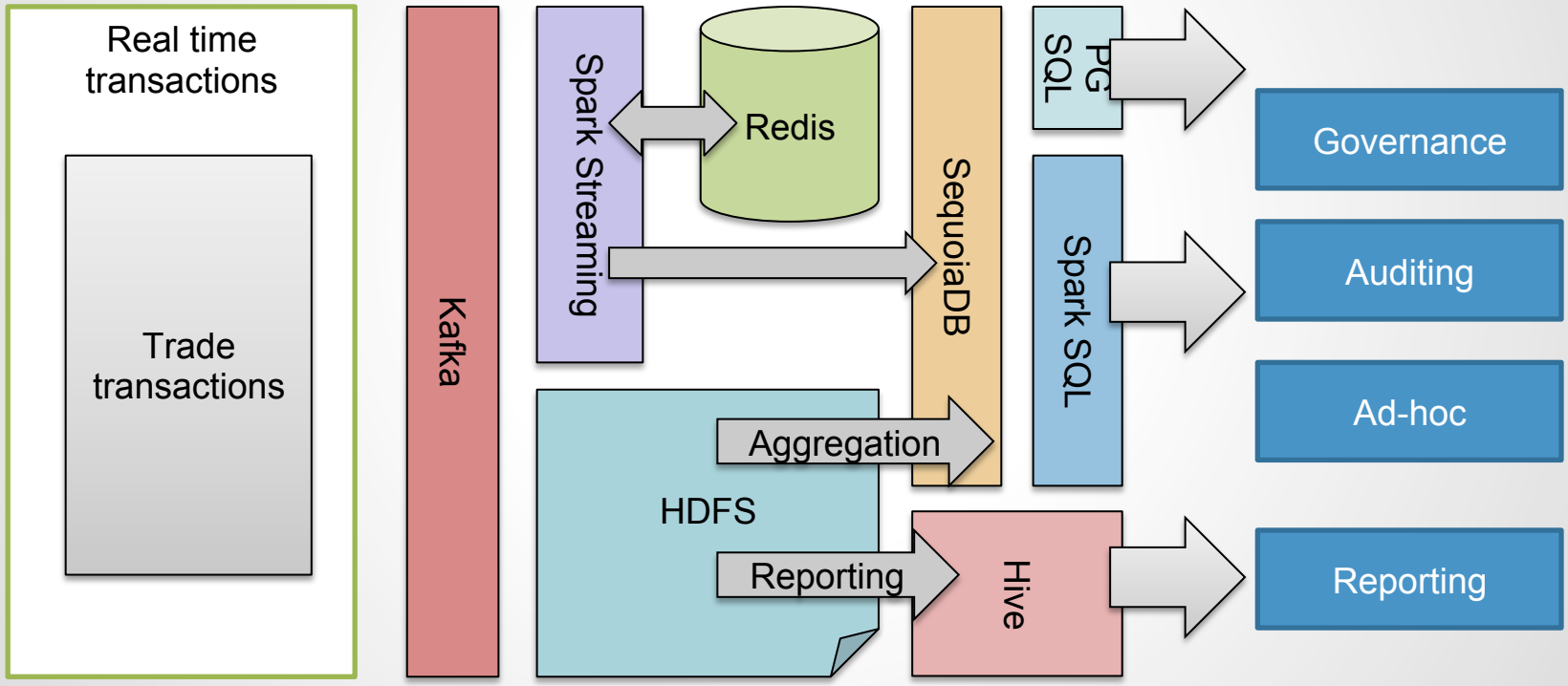
# Requirement

# of stocks: 50k
# of accounts: 300m
# of active accounts / day: 50m
# of transactions / day: 400m
# of trading office: 10k
# of transactions / sec during spike: 1m
fraud alarm interval: 10 seconds
detection window: 3 min

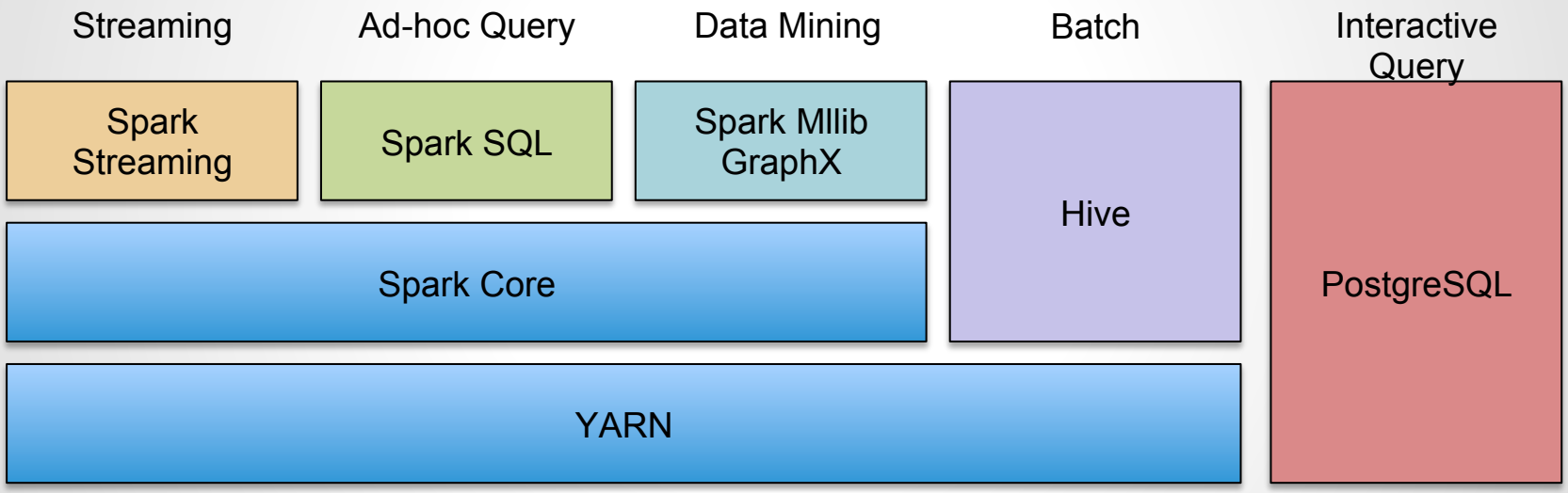
# Architecture View



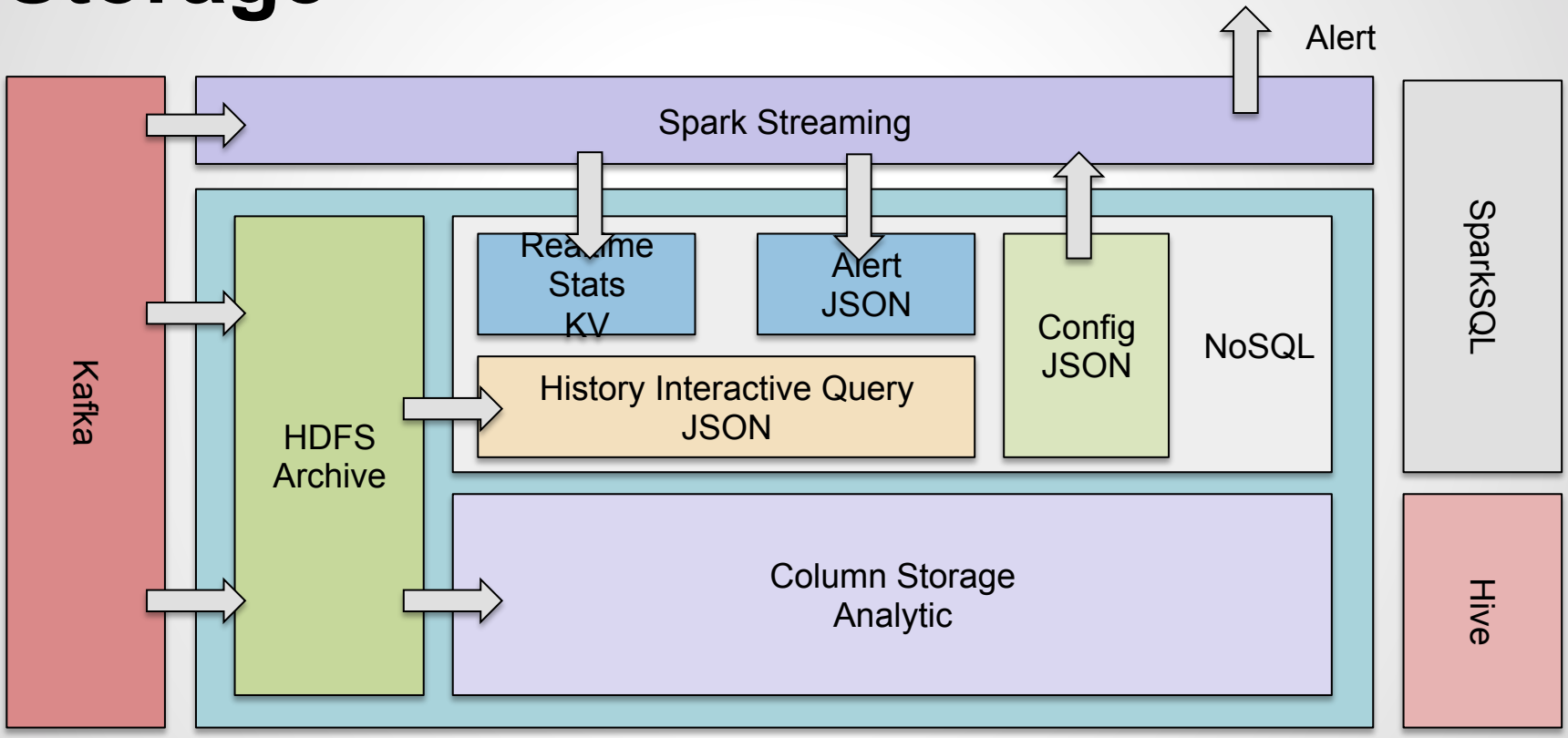
# Architecture



# Computing

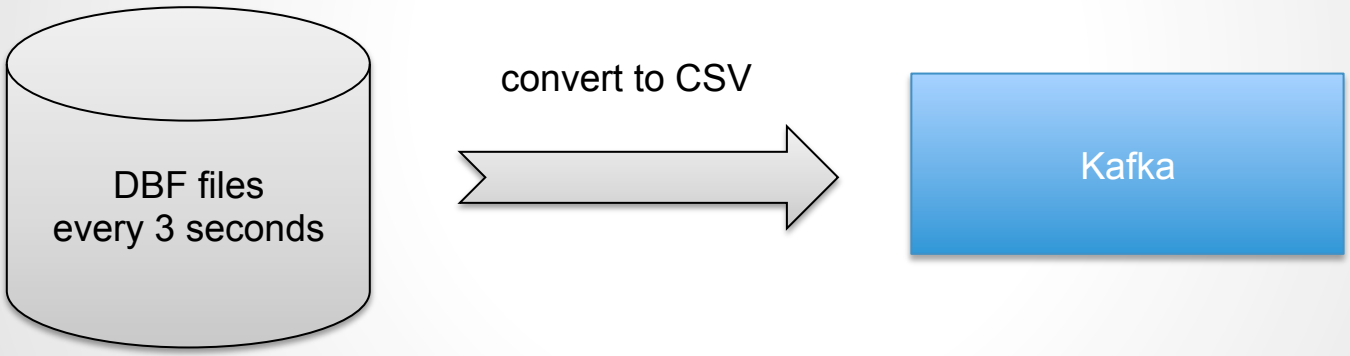

Streaming
Ad-hoc Query
Data Mining
Batch
Interactive Query
Spark Streaming
Spark SQL
Spark Mllib GraphX
Hive
PostgreSQL
Spark Core
YARN

# Storage


Alert
Spark Streaming
Kafka
HDFS
Archive
Realtime
Stats
KV
Alert
JSON
Config
JSON
NoSQL
History Interactive Query
JSON
Column Storage
Analytic
SparkSQL
Hive

# Deployment

# 1： Data Input



# 2：Cluster Planning

- Storage: Sizing, Schema, Logical Design
- Computing: Sizing, # of zones, # of workers

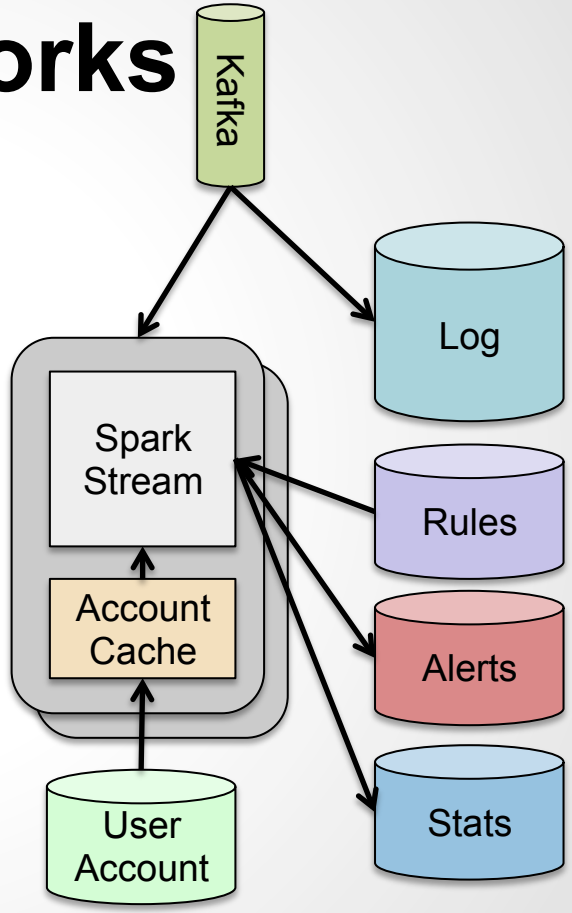
# 3： How Streaming Works

Kafka feeds data to Spark Streaming and Log files in HDFS

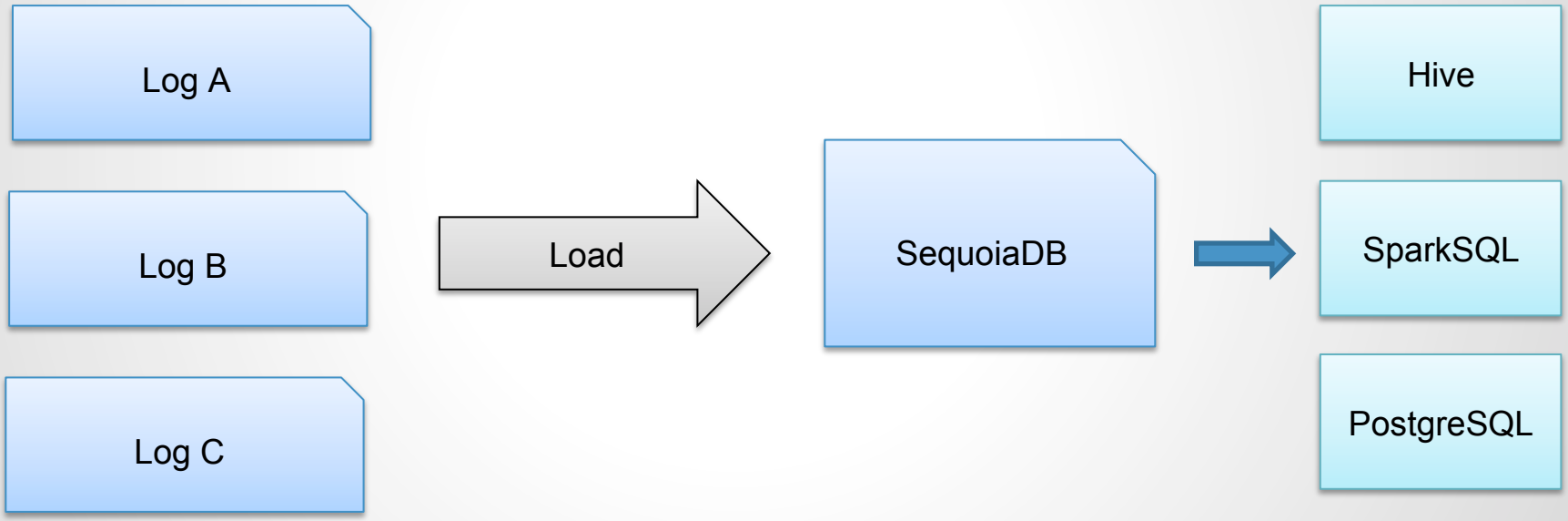
Cache user accounts in every box

Write alerts and aggregated stats to NoSQL

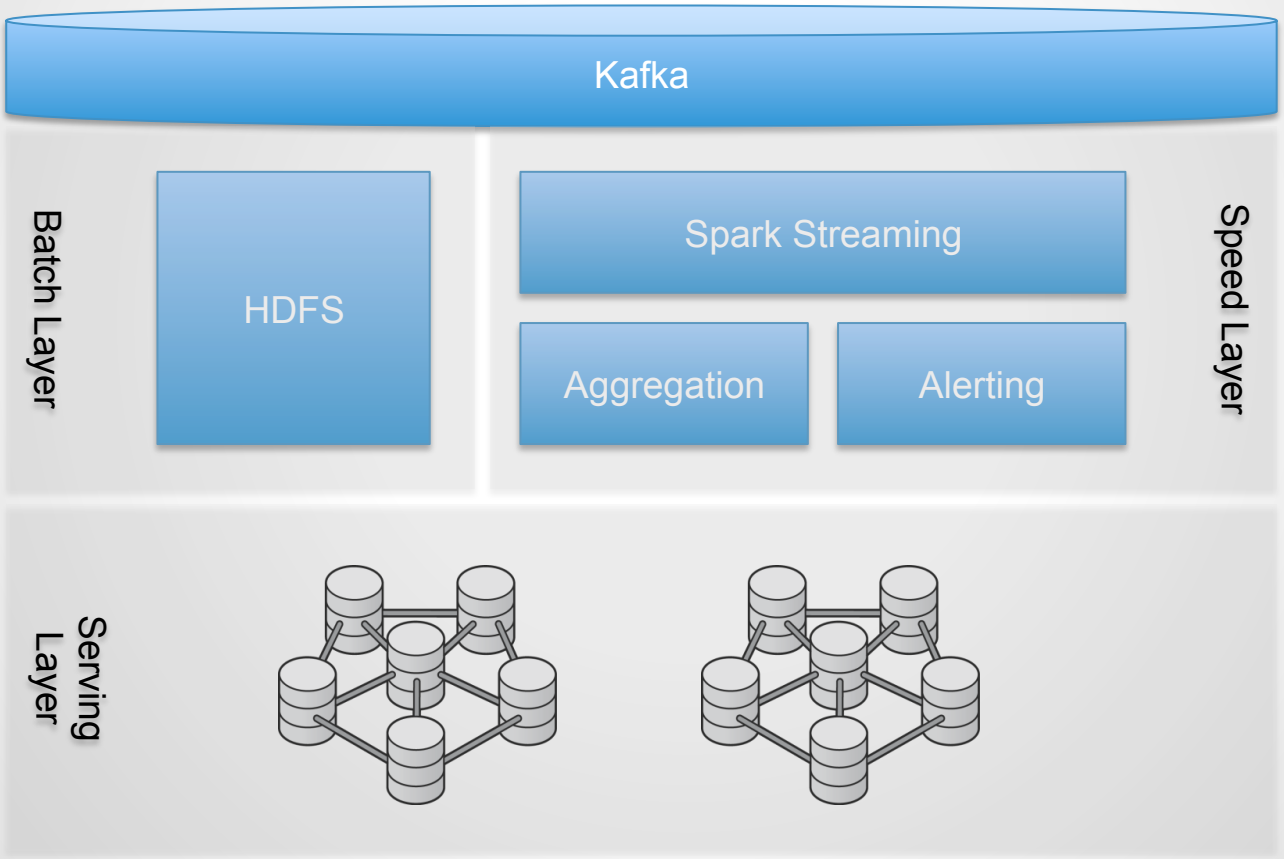
Applications read from aggregated results



# 4：Batch Job



# 5. Overall


Kafka
Batch Layer
HDFS
Spark Streaming
Aggregation
Alerting
Speed Layer
Serving Layer

# Pitfalls

# Scheduler

Cannot terminate tasks that created by ThriftServer

Resolved in Spark 1.5

Tasks run in serialized mode in Standalone mode

SET spark.sql.thriftserver.scheduler.pool=accounting;

Need to create many resource pools for parallel jobs

# Functions

OOM caused by beeline java heap size

Script was not able to pass arguments to java process, resolved in Spark 1.5

Hostname doesn’t work in --master

Use IP address instead

# Monitoring

- Limited monitoring methods, web only
- Number of workers/executors are critical to performance
- SparkSQL is 10 times faster than Hive for big table joins

# Spark/SequoiaDB Connector

- Predicate Pushdown is critical to performance
- Milliseconds query if predicates hit index
- Database design is critical for real-time queries
- Specify multiple coordinator nodes in connection string

# We Are Hiring!

# Thanks!

My Contact Info:

- taoewang@sequoiadb.com
- www.sequoiadb.com