Spark Streaming
+
Kinesis

Chris Fregly
AdvancedAWS Meetup
8/12/2014

Who am I?

Former Netflix’er
(netflix.github.io)

Spark Contributor
(github.com/apache/spark)

Consultant
(fluxcapacitor.com)

Author
(sparkinaction.com)

Quick Poll

- Spark, Spark Streaming?
- Hadoop, Hive, Pig?
- EMR, Redshift?
- Flume, Kafka, Kinesis, Storm?
- Lambda Architecture?

“Streaming”

~~NETFLIX~~

~~hadoop~~

Apache Kafka
A high-throughput distributed messaging system.

Storm

Kinesis

Spark
Streaming

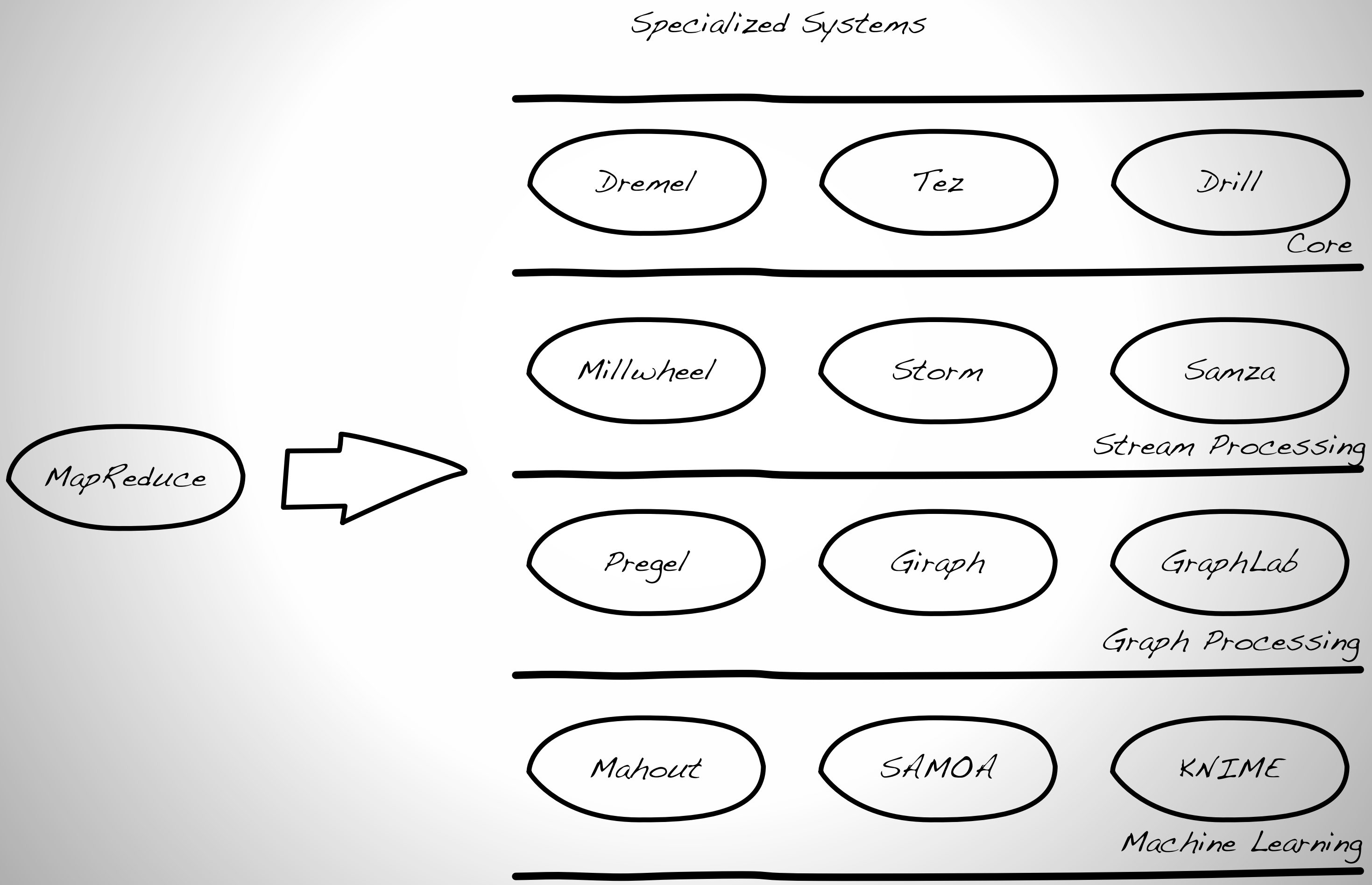
Spark Overview

- Part of Berkeley Data Analytics Stack (“badass”)
- ~2009, Berkeley AMPLab
- Written in Scala
- Supports Java, Python, SQL, and R
- In-memory whenever possible
- Improved efficiency over MapReduce
  - 100x in-memory, **2-10x on-**disk
- Compatible with Hadoop
  - File formats, SerDes, and UDFs
  - Hive (Shark) and Pig (Spork)

Spark API

- Richer, more expressive API than just map() and reduce()
  - filter(), join(), distinct(), groupByKey()
- Supports Java, Python, SQL, and R
- Resilient Distributed Dataset (RDD)
  - Core Spark abstraction
  - Partition across cluster
  - Parallel, fault tolerant, immutable, recomputable
- Unified API across all libraries

Non-unified Specialized Systems



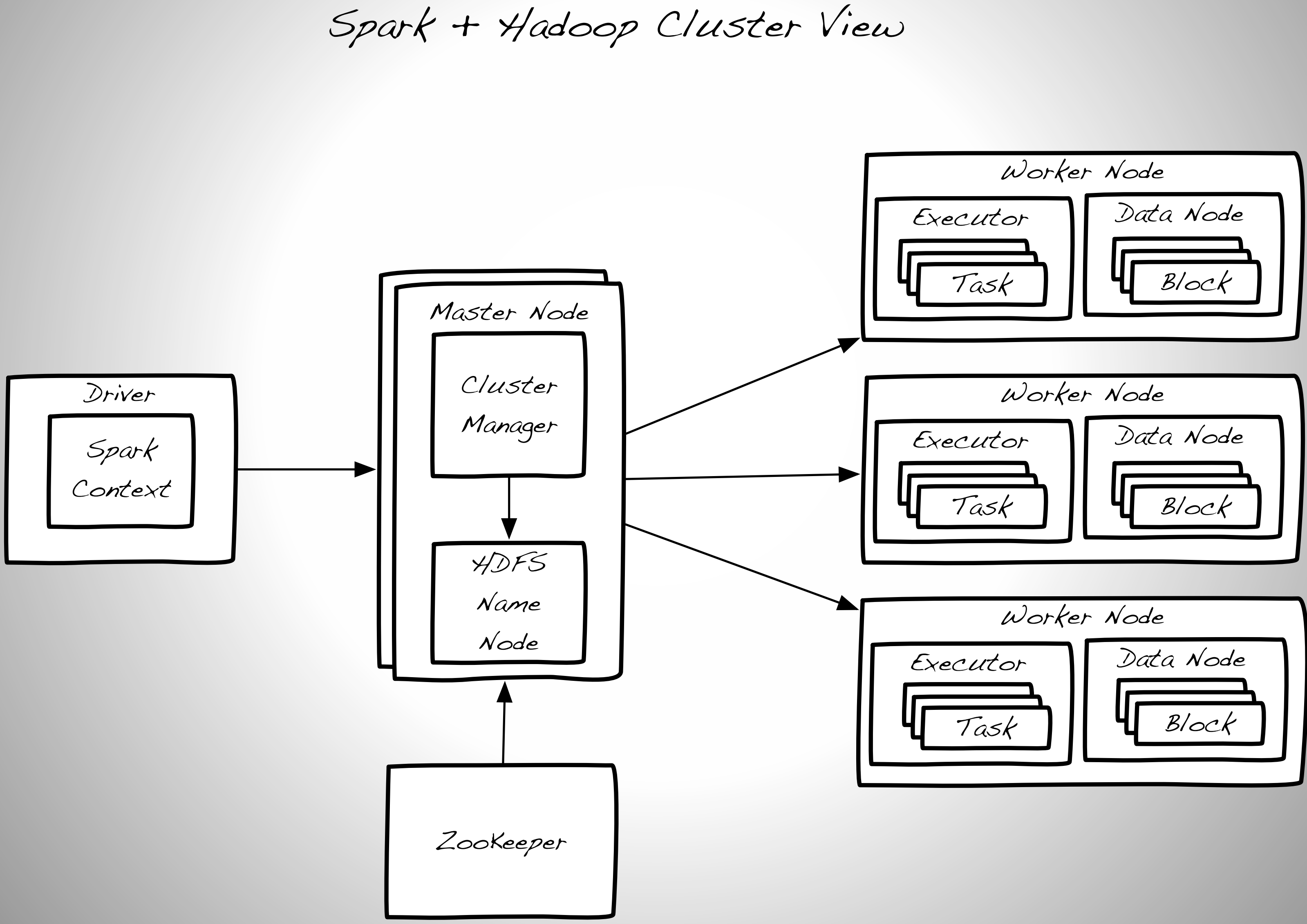
Spark Libraries

- Spark SQL (Data Processing)
- Spark Streaming (Streaming)
- MLlib (Machine Learning)
- GraphX (Graph Processing)
- BlinkDB (Approximate Queries)

Similar Projects

- Spark
  - Microsoft Dryad
  - Apache Tez
  - Impala
  - Google BigQuery
  - Google Cloud Dataflow

Spark + Hadoop Cluster View



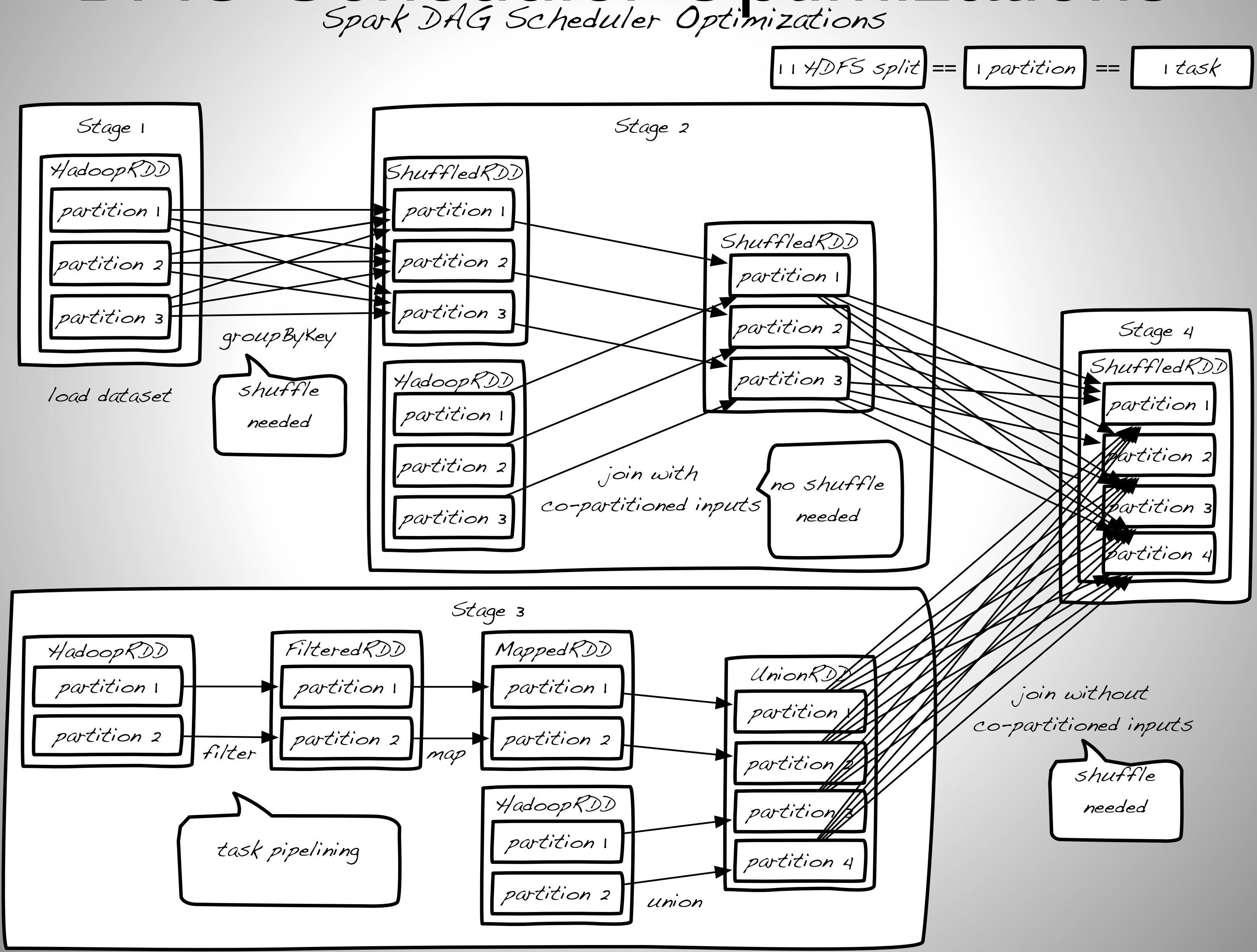
Master High Availability

- Multiple Master Nodes
- ZooKeeper maintains current Master
- Existing applications and workers will be notified of new Master election
- New applications and workers need to explicitly specify current Master
- Alternatives (Not recommended)
  - Local filesystem
  - NFS Mount

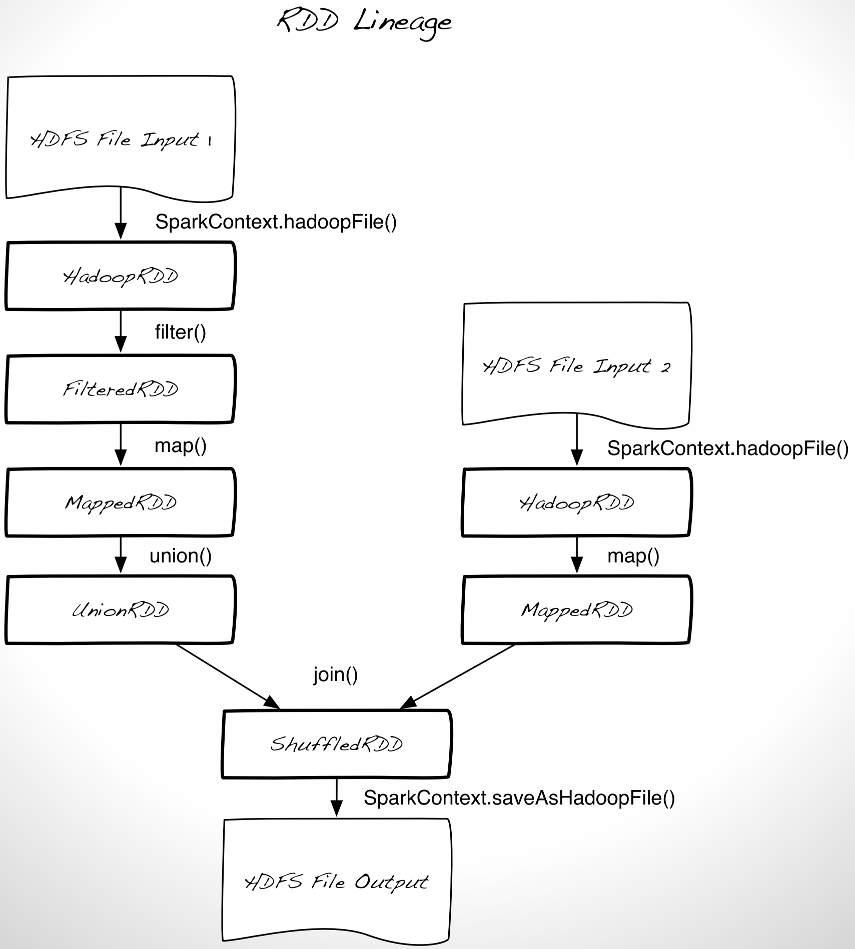
Spark Execution Engine

- General cluster computing engine
- Parallel, distributed, DAG-based
- Lazy evaluation
- Allows optimizations
- Data locality and rack awareness
- Fine-grained fault tolerance using RDD lineage graphs

DAG Scheduler Optimizations



Lineage-based Fault Tolerance



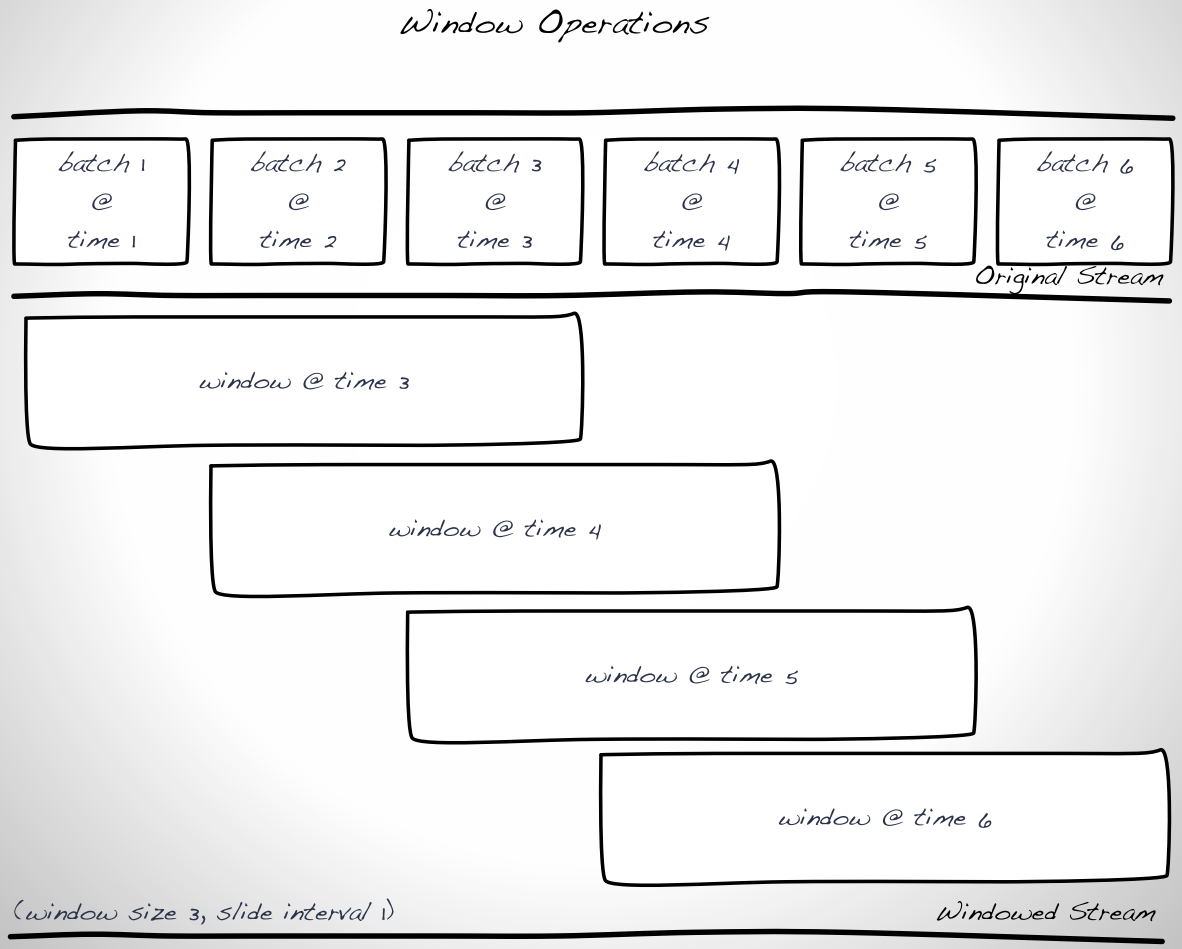
Spark Streaming Overview

- Low latency, high throughput, fault-tolerant
- DStream: Micro-batches of RDDs
  - Operations are similar to RDD
  - Lineage for fault-tolerance
- Supports Flume, Kafka, Twitter, Kinesis, etc.
- Built on Spark Core Execution Engine and API
- Long-running Spark Application

Spark Streaming API

- Rich, expressive API based on core Spark API
  - filter(), join(), distinct(), groupByKey()
- Maintain State
  - updateStateByKey()
- Window Operations
  - Window size & slide interval
- Checkpointing
- Register DStream as a SQL table

Window Operations



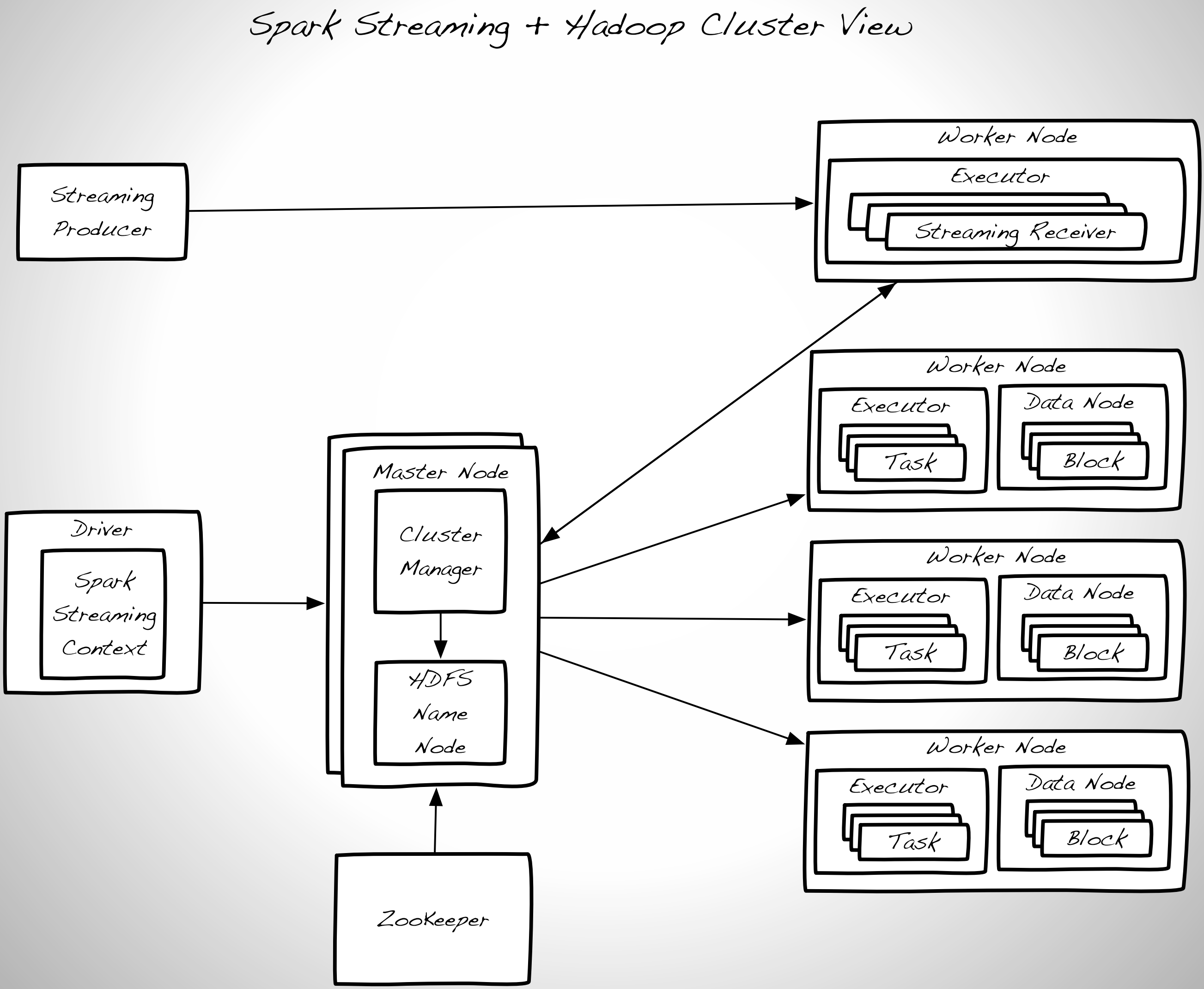
Similar Projects

- Spark Streaming
  - Twitter Storm
  - Yahoo! S4
  - LinkedIn Samsa
  - Google Millwheel

Spark Streaming Use Cases

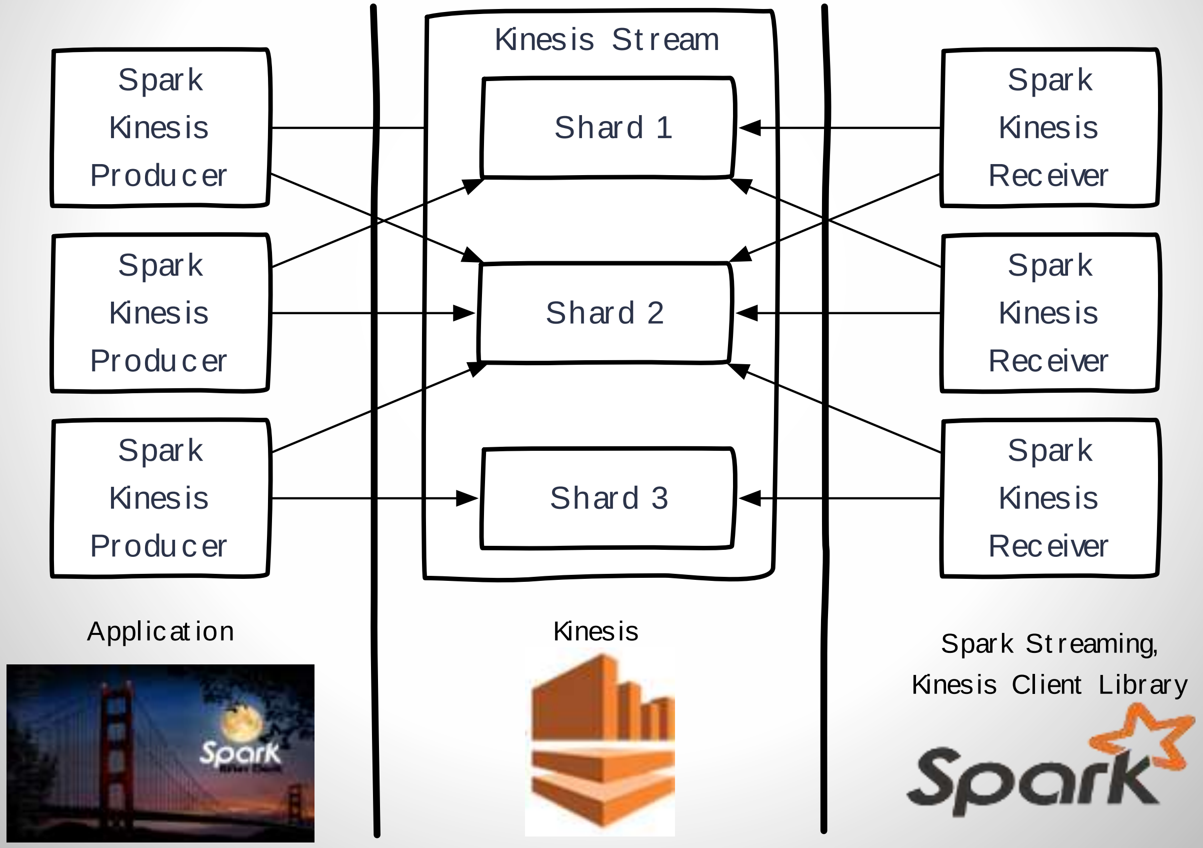
- Operational dashboards
- ETL on streaming data ingestion
- Anomaly, malware, and fraud detection
- Predictive maintenance
  - – Sensors
- NLP analysis
  - – Twitter Firehose
- Lambda architecture
  - – Unified batch and streaming
  - – ie. Different machine learning models for different time frames

Spark Streaming Cluster View

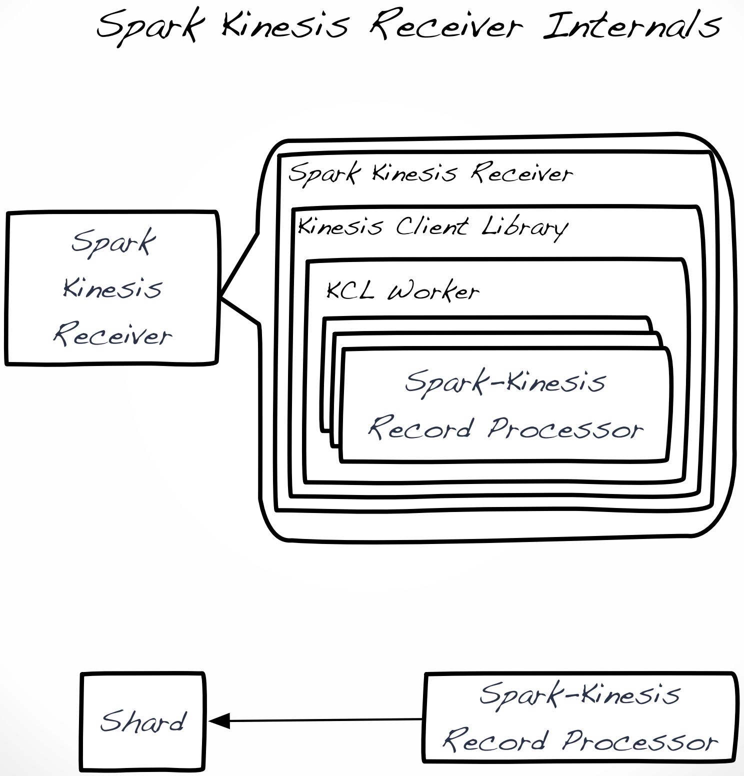


Spark Streaming + Kinesis

Spark Streaming Kinesis Architecture



Spark Kinesis Receiver Internals


Spark Kinesis Receiver Internals
Spark Kinesis Receiver
Spark Kinesis Receiver
Kinesis Client Library
KCL Worker
Spark-Kinesis Record Processor
Shard
Spark-Kinesis Record Processor

Scaling

- Horizontally scale by adding more Kinesis Receivers
- Kinesis Client Library within each Kinesis Receiver will negotiate and rebalance shard processing
- Never need more Kinesis Receivers than the number of stream shards
- Each Kinesis Receiver can support multiple shards
- Supports Kinesis shard splitting/merging
- Recommendation: over provision shards and avoid splitting/merging

Demo!

Kinesis

Streaming

https://github.com/apache/spark/blob/master/extras/kinesis-asl/src/main/…

Scala: scala/org/apache/spark/examples/streaming/KinesisWordCountASL.scala

Java: java/org/apache/spark/examples/streaming/JavaKinesisWordCountASL.java

Streaming Receiver Failure

- Upon failure, backup receiver takes over
- Checkpoint sources like Kafka and kinesis allow multiple receivers to pull from the same stream (ie. during a failover)
  - De-duping is handled by Spark
- Supports graceful shutdown to allow in-flight message draining
- Recommendation: choose buffered sources like Flume, Kafka and Kinesis

Streaming Driver Failure

- Streaming Driver app is long-running
  - Monitor driver, receiver worker nodes, and streams
  - Alert upon failure or unusually high latency
- Driver failure interrupts stream processing
- Enable checkpointing for backup Driver to take over
- Use StreamingContext.getOrCreate(…) in Driver app

Types of Checkpoints

Spark

1. Spark checkpointing of StreamingContext DStreams and metadata
2. Lineage of state and window DStream operations

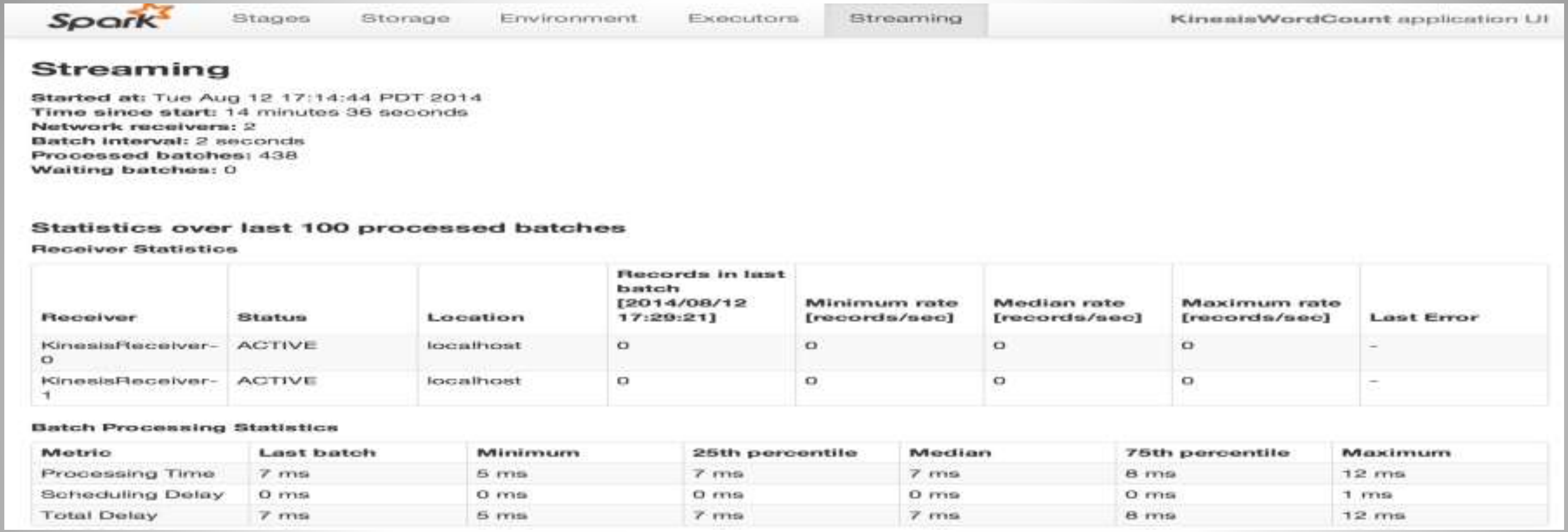
Kinesis

3. Kinesis Client Library (KCL) checkpoints current position within shard
   - Checkpoint info is stored in DynamoDB per Kinesis Application keyed by shard

Monitoring

- Streaming tab in Spark Web UI
- CloudWatch
- StreamingListener callback

Web UI Monitoring



Streaming

Started at: Tue Aug 12 17:14:44 PDT 2014
Time since start: 14 minutes 36 seconds
Network receivers: 2
Batch interval: 2 seconds
Processed batches: 438
Waiting batches: 0

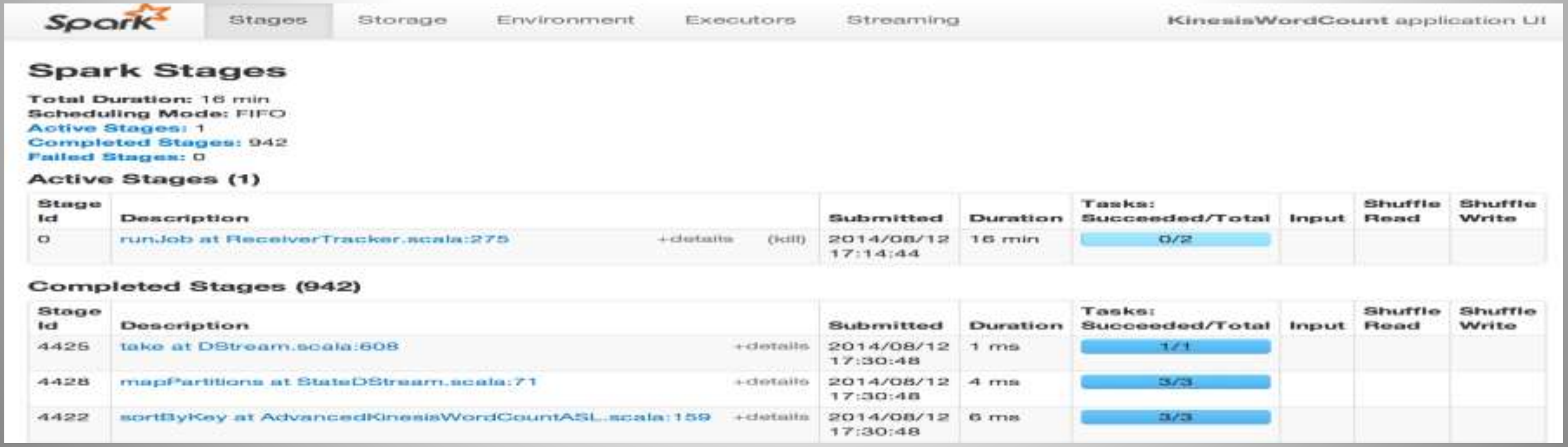
Statistics over last 100 processed batches

Receiver Statistics

Receiver	Status	Location	Records in last batch [2014/08/12 17:29:21]	Minimum rate [records/sec]	Median rate [records/sec]	Maximum rate [records/sec]	Last Error
KinesisReceiver-0	ACTIVE	localhost	0	0	0	0	-
KinesisReceiver-1	ACTIVE	localhost	0	0	0	0	-

Batch Processing Statistics

Metric	Last batch	Minimum	25th percentile	Median	75th percentile	Maximum
Processing Time	7 ms	5 ms	7 ms	7 ms	8 ms	12 ms
Scheduling Delay	0 ms	0 ms	0 ms	0 ms	0 ms	1 ms
Total Delay	7 ms	5 ms	7 ms	7 ms	8 ms	12 ms



Spark Stages

Total Duration: 16 min
Scheduling Mode: FIFO
Active Stages: 1
Completed Stages: 942
Failed Stages: 0

Active Stages (1)

Stage Id	Description			Submitted	Duration	Tasks: Succeeded/Total	Input	Shuffle Read	Shuffle Write
0	runJob at ReceiverTracker.scala:275	+details	(kill)	2014/08/12 17:14:44	16 min	0/2			

Completed Stages (942)

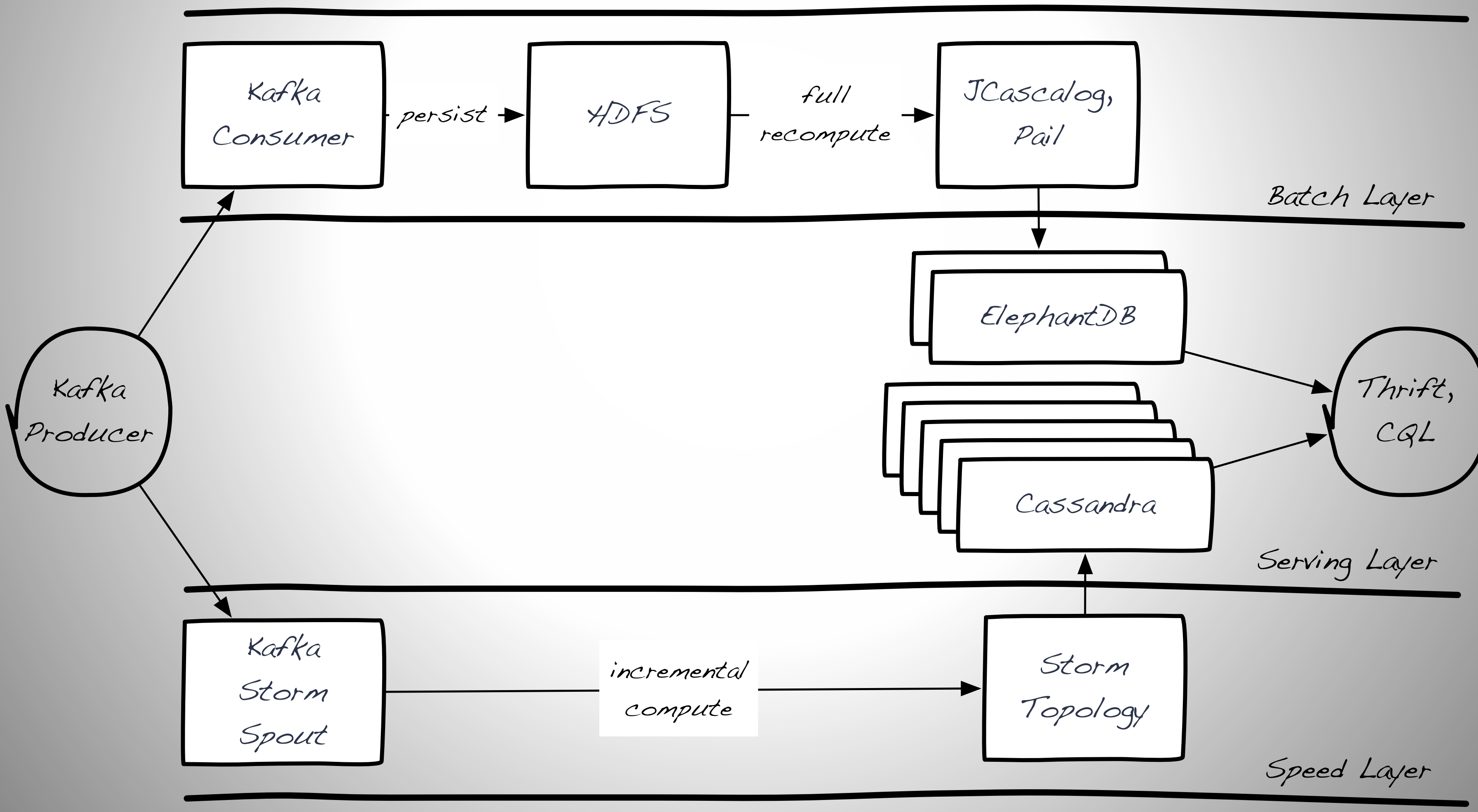
Stage Id	Description		Submitted	Duration	Tasks: Succeeded/Total	Input	Shuffle Read	Shuffle Write
4425	take at DStream.scala:608	+details	2014/08/12 17:30:48	1 ms	1/1			
4428	mapPartitions at StateDStream.scala:71	+details	2014/08/12 17:30:48	4 ms	3/3			
4422	sortByKey at AdvancedKinesisWordCountASL.scala:159	+details	2014/08/12 17:30:48	6 ms	3/3			

Tuning

- Batch interval
  - High: reduce overhead of submitting new tasks for each batch
  - Low: keeps latencies low
  - Recommendation: test & find the sweet spot
- Checkpoint interval
  - High: reduce load on checkpoint overhead
  - Low: reduce amount of data loss on failure
  - Recommendation: 5-10x sliding window interval
- Explicitly uncache DStreams when no longer needed
- Use CMS GC for consistent processing times
- Use Kryo serialization
- DStreams are already serialized as byte arrays (versus Java objects) to minimize GC

Lambda Architecture Overview

Lambda Architecture as Initially Described



Kafka Consumer

persist

HDFS

full recompute

JCascalog, Pail

Batch Layer

ElephantDB

Kafka Producer

Thrift, CQL

Cassandra

Serving Layer

Kafka Storm Spout

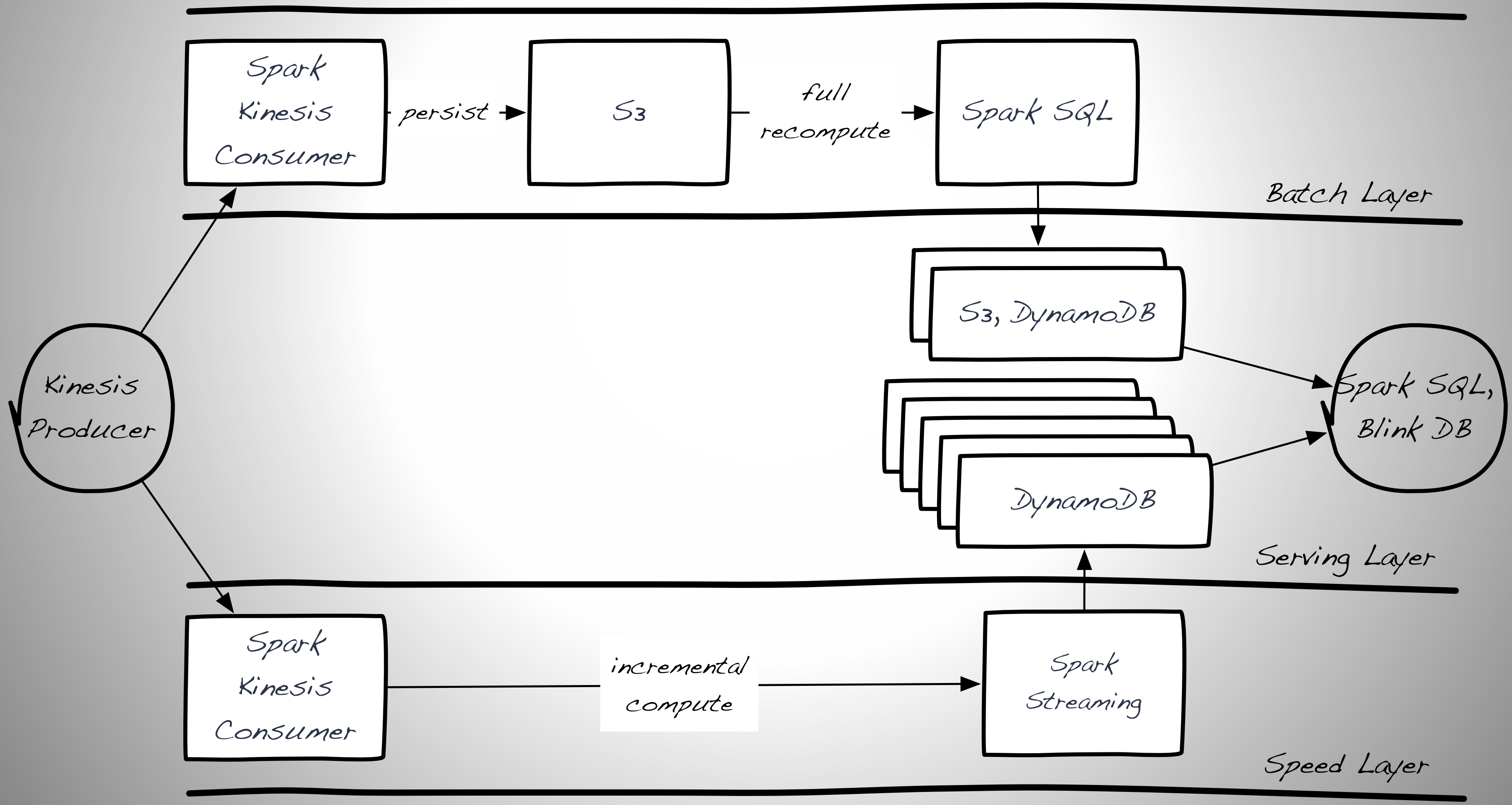
incremental compute

Storm Topology

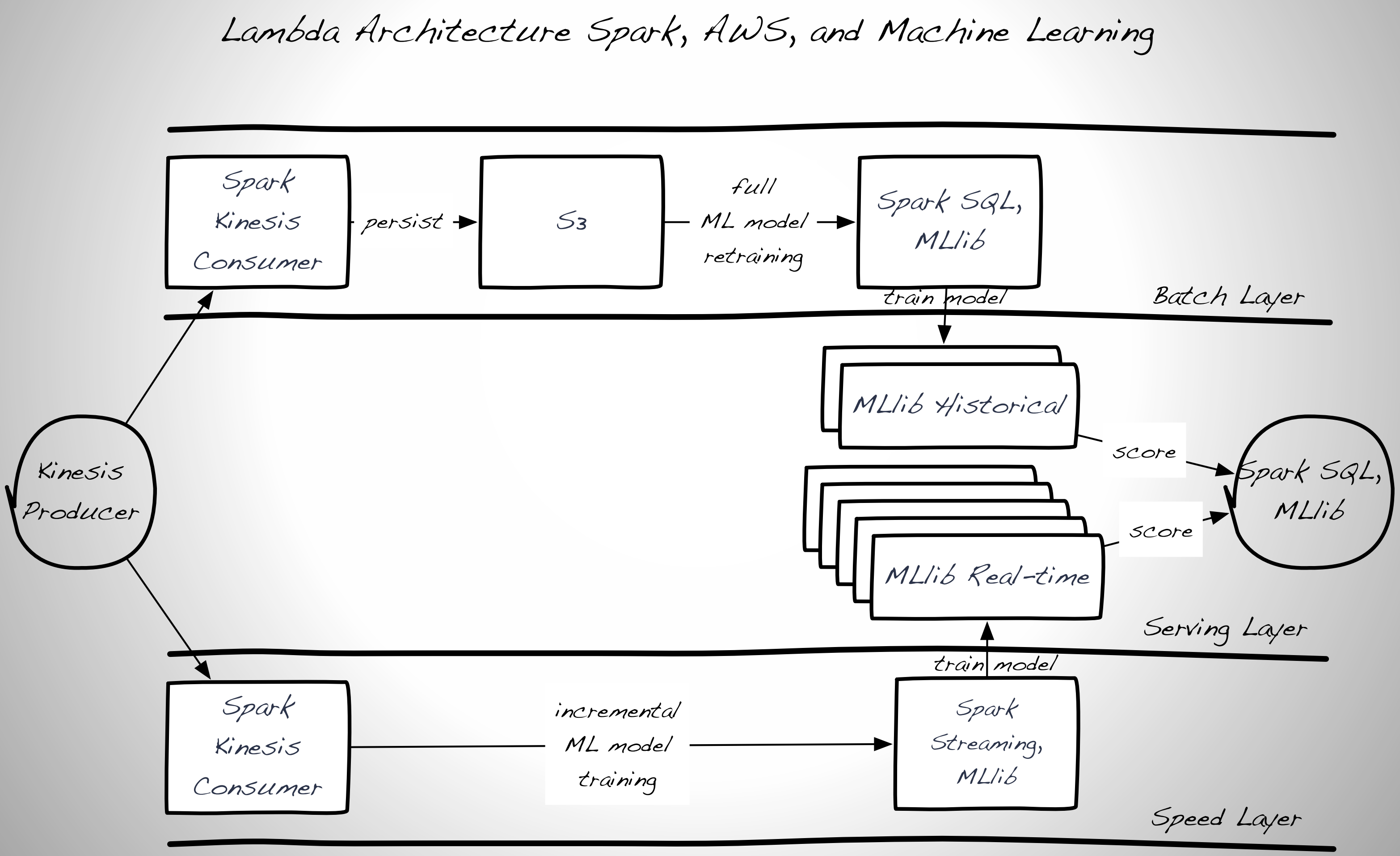
Speed Layer

Spark + AWS + Lambda

Lambda Architecture Spark and AWS



Spark + AWS + Lambda + ML



Summary

- Spark
- Spark Streaming + Kinesis
- Scaling
- Fault Tolerance
- Monitoring
- Tuning
- Lambda Architecture
- Spark in Action
  - Oct 2014 MEAP
  - Early access: http://sparkinaction.com