Modernizing Infrastructures for Fast Data

Spark, Kafka, Cassandra, Reactive Platform and Mesos

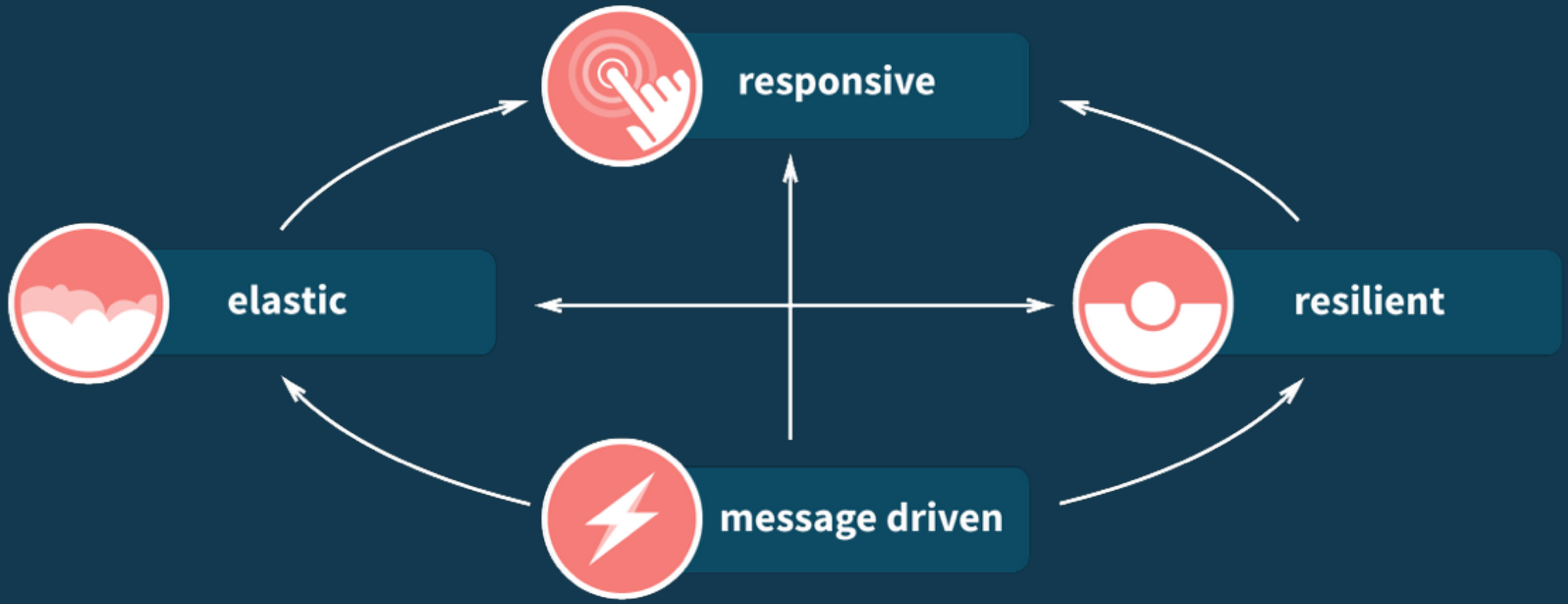
by Dean Wampler, Ph.D. (@deanwampler)

Outline

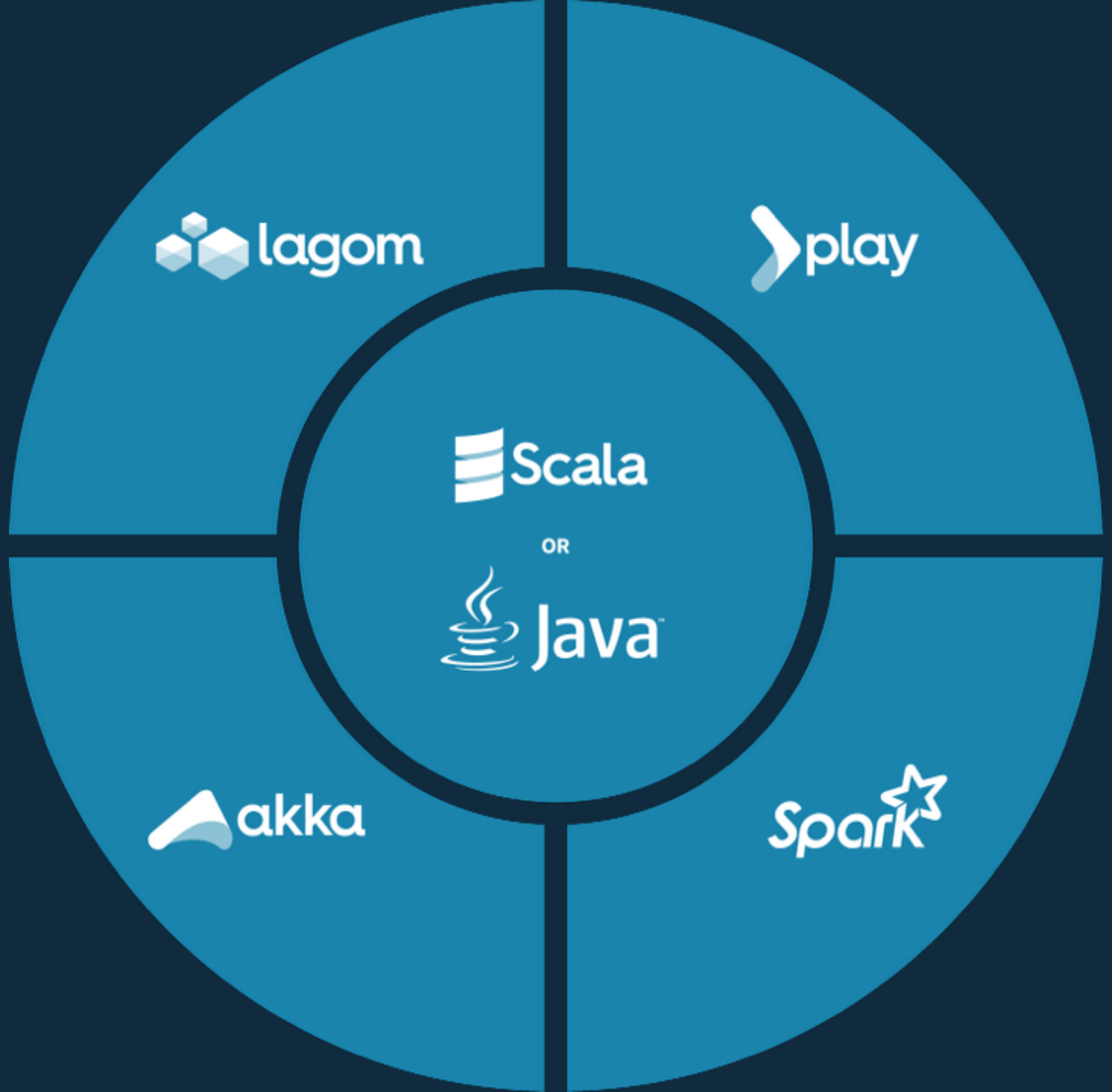
- Reactive Enterprise Architectures: The Lightbend Perspective
- Big Data and the Emergence of Apache Spark
- An Architecture for Fast Data

Reactive Enterprise Applications: The Lightbend Perspective

Reactive Manifesto



The Lightbend Reactive Platform


lagom
play
Scala
OR
Java
akka
Spark


Media
Social
Technology
Education
IoT
Online Services
Retail
Finance

7

Big Data and the Emergence of Apache Spark

Distributed compute frameworks: MapReduce

- Distribution computation over that data.

MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

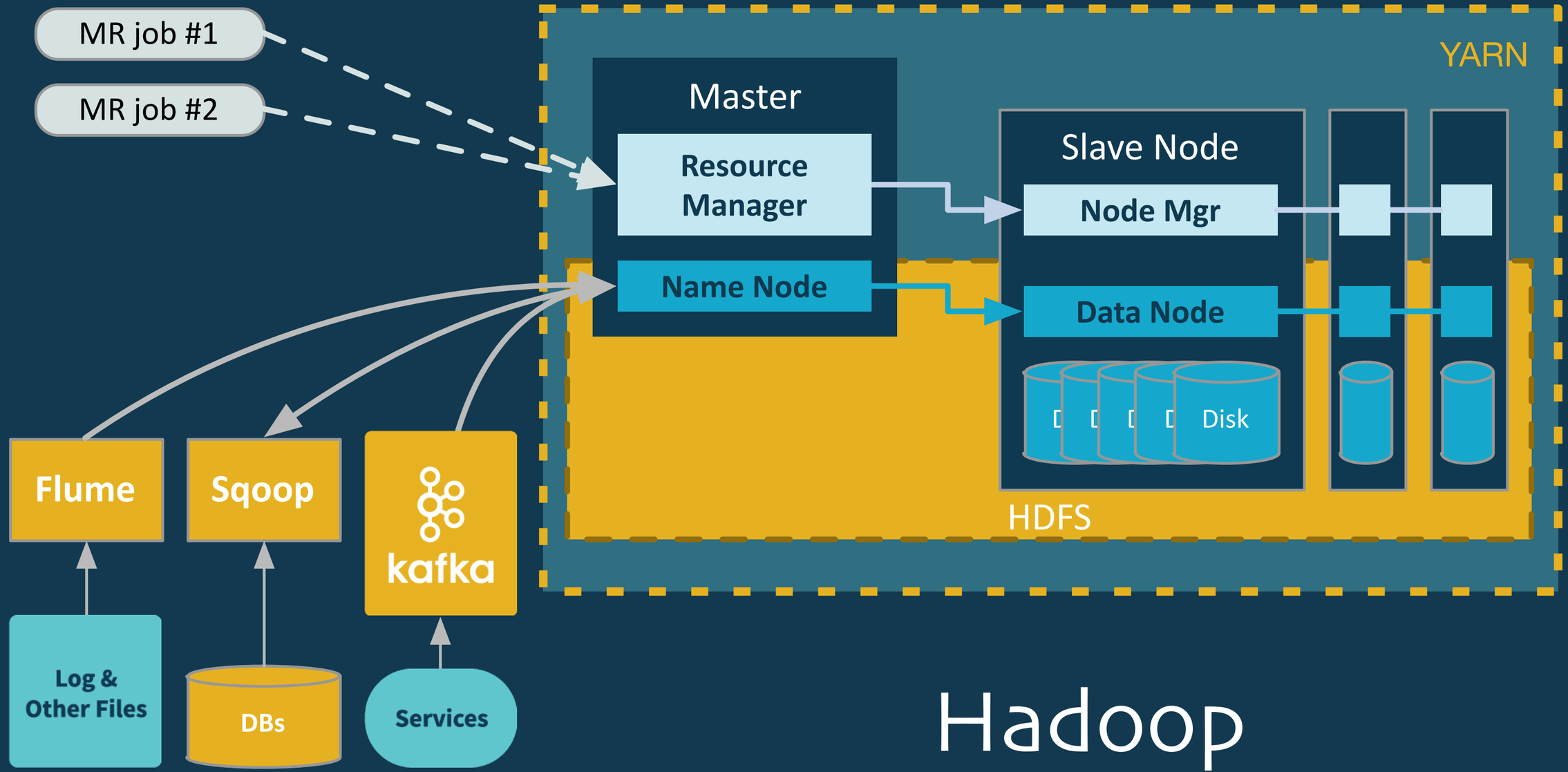
jeff@google.com, sanjay@google.com

Google, Inc.

Abstract

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a *map* function that processes a key/value pair to generate a set of intermediate key/value pairs, and a *reduce* function that merges all intermediate

given day, etc. Most such computations are conceptually straightforward. However, the input data is usually large and the computations have to be distributed across hundreds or thousands of machines in order to finish in a reasonable amount of time. The issues of how to parallelize the computation, distribute the data, and handle


MR job #1
MR job #2
YARN
Master
Resource Manager
Name Node
Slave Node
Node Mgr
Data Node
Disk
HDFS
Flume
Sqoop
kafka
Log & Other Files
DBs
Services

Hadoop

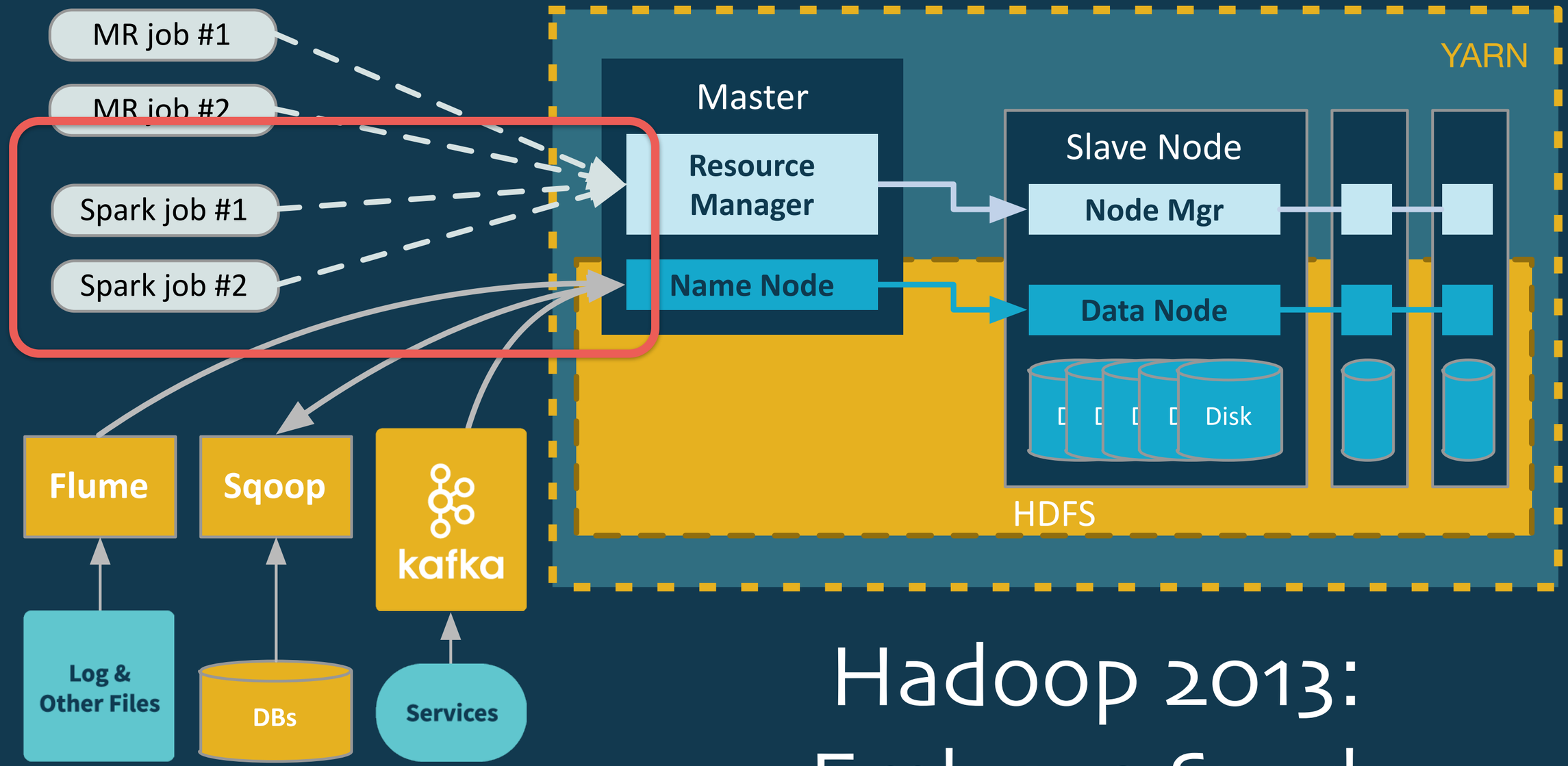
Hadoop Strengths

- Lowest CapEx system for Big Data.
- Excellent for ingesting and integrating diverse datasets.
- Flexible: from classic analytics (aggregations and data warehousing) to machine learning.

Hadoop Weaknesses

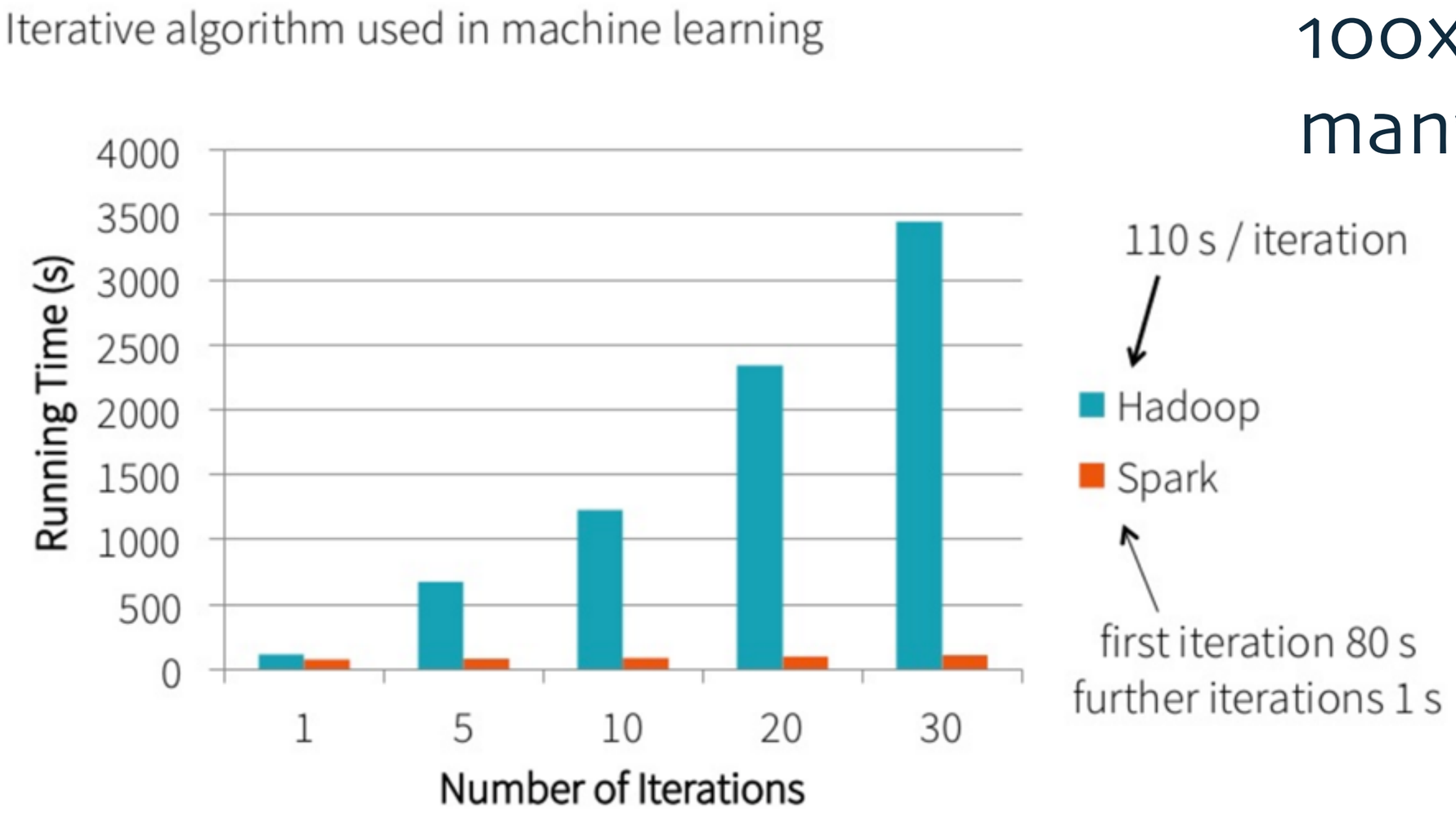
- Complex administration.
- YARN can't manage all distributed services.
- MapReduce:
  - Has poor performance.
  - A difficult programming model.
  - Doesn't support stream processing.

Why Apache Spark?



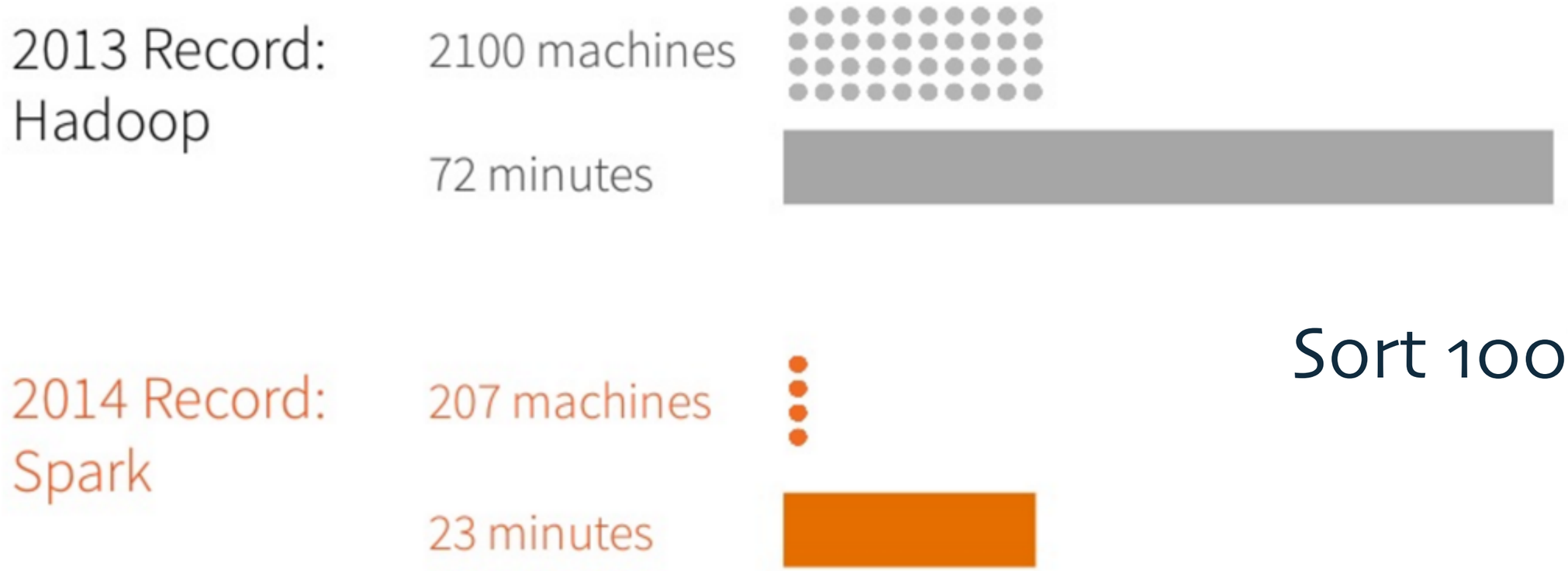
Hadoop 2013: Embrace Spark

Spark vs. MapReduce Performance



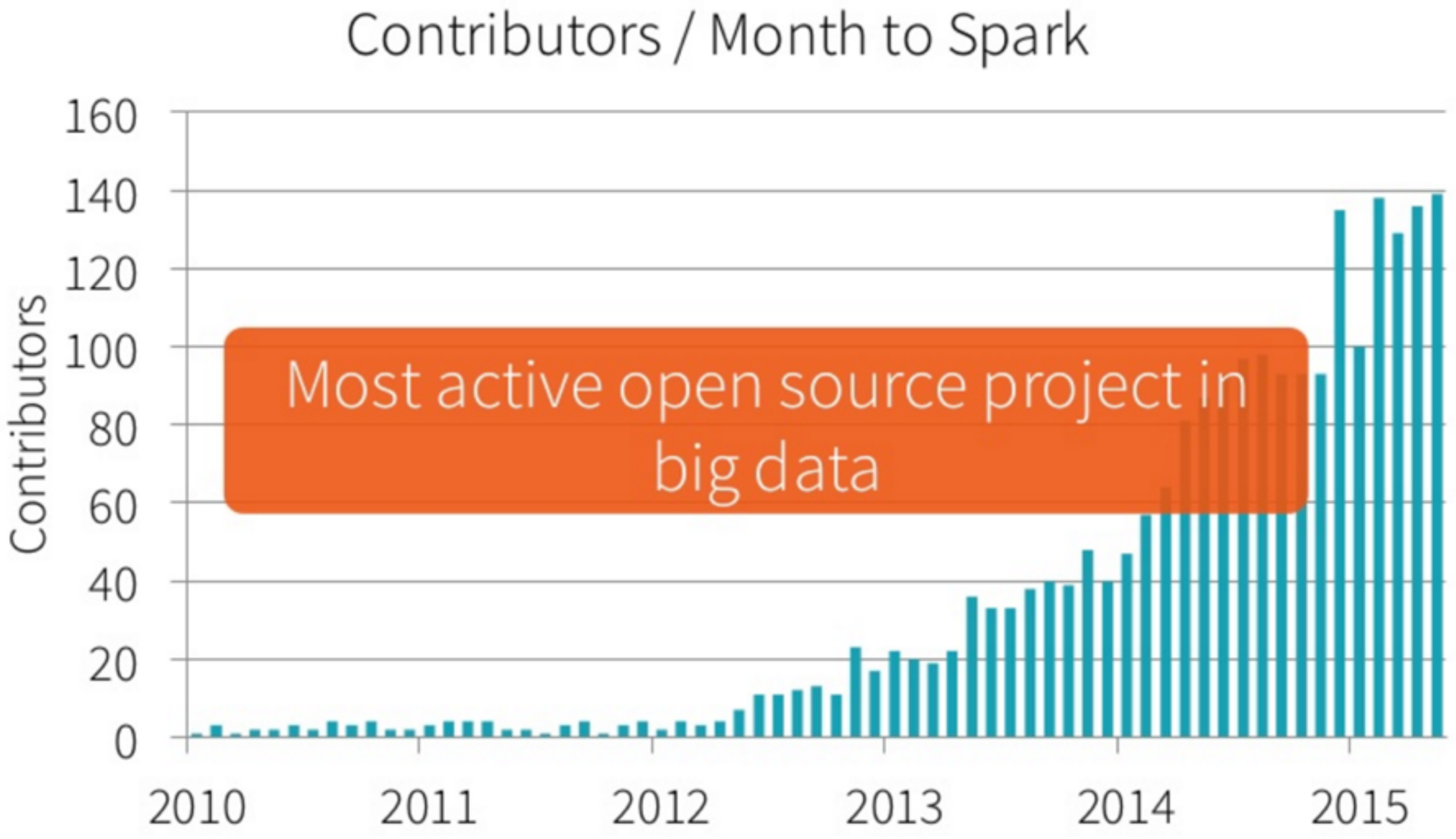
100x better for many algorithms.

Spark: Major Performance Improvements



Sort 100TB

One of the Fastest Growing OS Projects



Modules

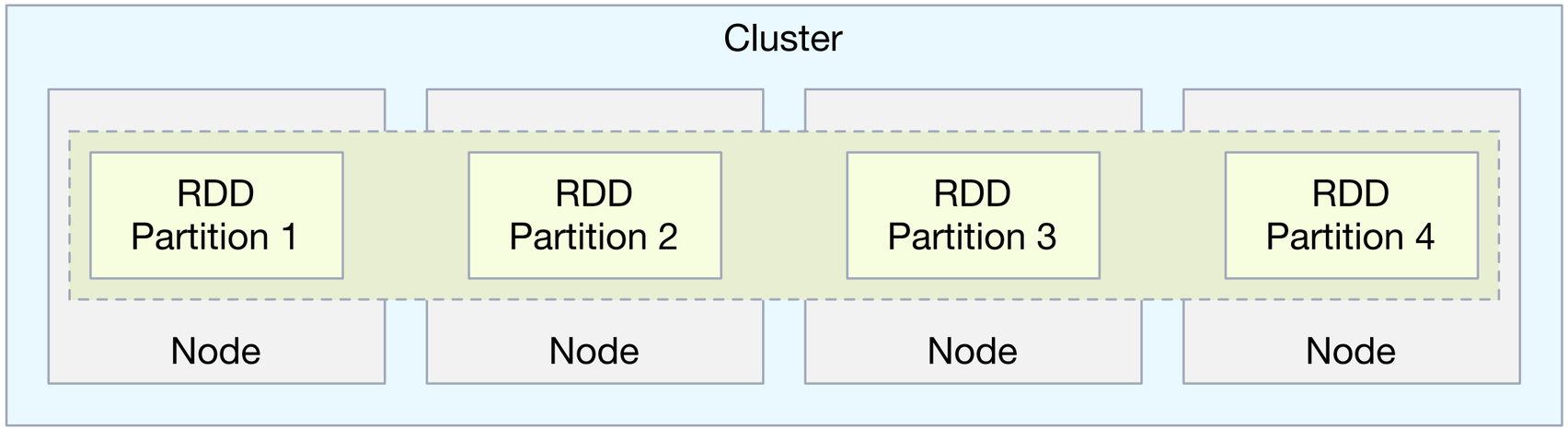
SQL/DataFrames (Structured Data)	**Spark Streaming** (~Real Time)	**MLlib** (Machine Learning)	**GraphX** (Graphs)

Spark RDD
(Core)

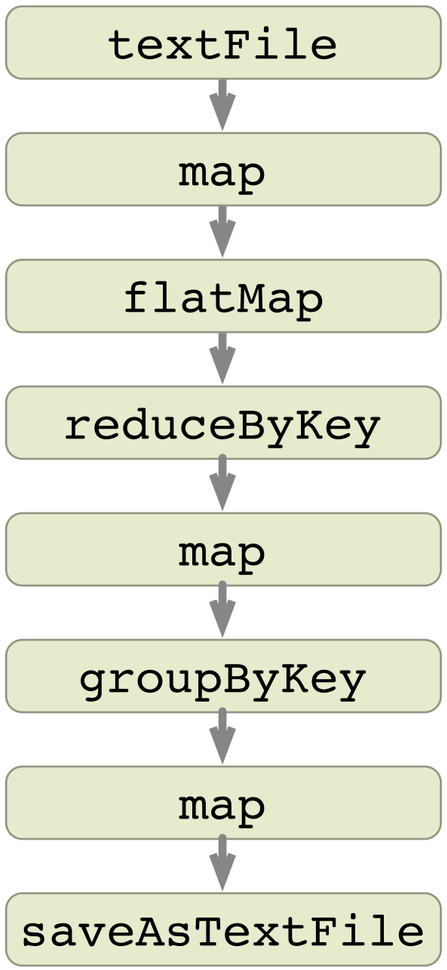
The Core - Resilient Distributed Datasets

SQL/DataFrames (Structured Data)	**Spark Streaming** (~Real Time)	**MLlib** (Machine Learning)	**GraphX** (Graphs)

Spark RDD
(Core)

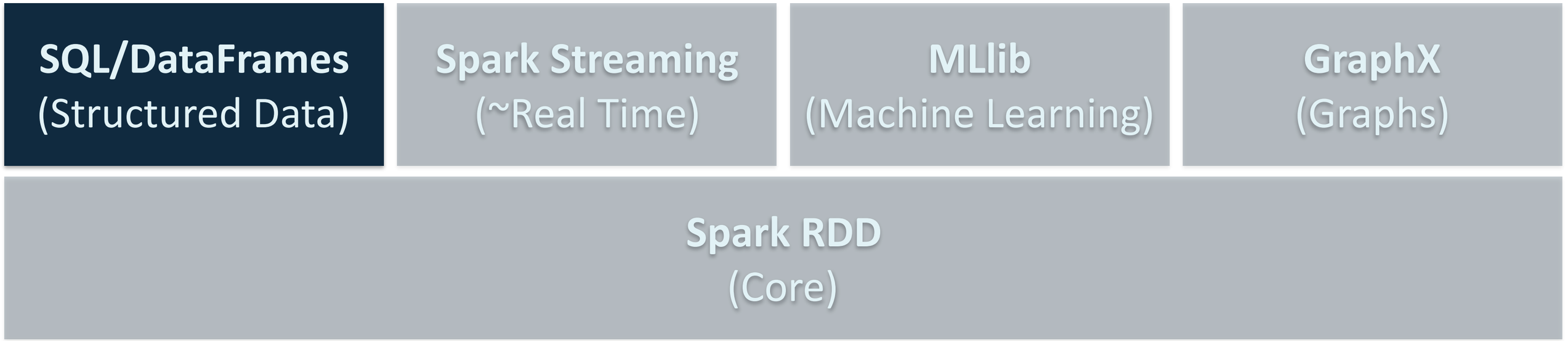


“Inverted Index” in Spark




```
sparkContext.textFile("/path/to/input")
.map { line =>
  val array = line.split(",", 2)
  (array(0), array(1))
}.flatMap {
  case (id, contents) => toWords(contents).map(w=>((w,id),1))
}.reduceByKey {
  (count1, count2) => count1 + count2
}.map {
  case ((word, path), n) => (word, (path, n))}
.groupByKey
.map {
  case (word, list) => (word, sortByCount(list))
}.saveAsTextFile("/path/to/output")
```

SQL queries and a "DataFrame" DSL



- For data with a fixed schema…
  - Write SQL queries (currently a subset of HiveQL).
  - Use equivalent Python-inspired `DataFrame` API.

Use SQL or the Idiomatic DataFrame API

```
# SQL:
sqlContext.sql("""
  SELECT state, age, COUNT(*) AS cnt
  FROM people
  GROUP BY state, age
  ORDER BY cnt DESC, state ASC, age ASC
  """)

// DataFrame (Scala):
people.state($"state", $"age")
  .groupBy($"state", $"age").count()
  .orderBy($"count".desc, $"state".asc, $"age".asc)
```

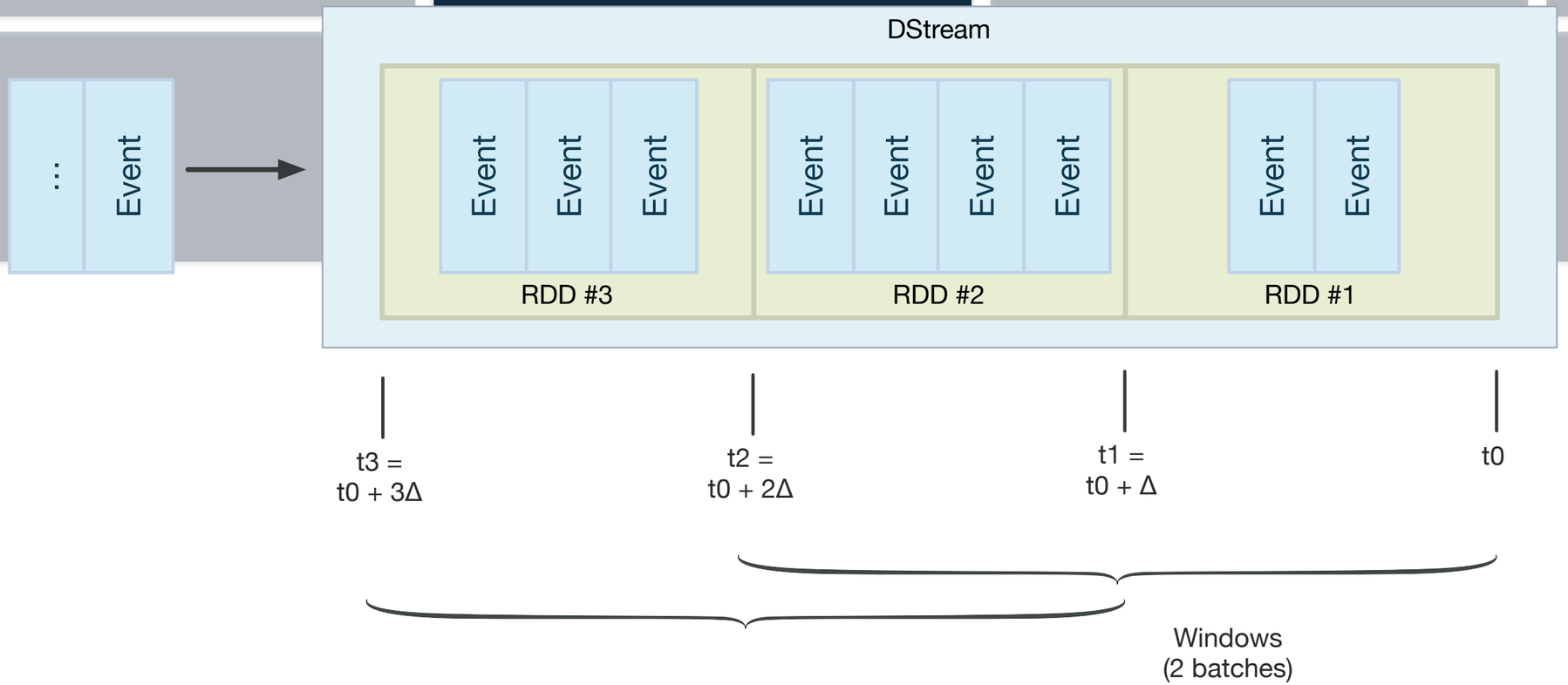
Spark Streaming: "Mini-batch" Processing

SQL/DataFrames (Structured Data)

Spark Streaming (~Real Time)

MLlib (Machine Learning)

GraphX (Graphs)



Streaming Inverted Index

```
val kafkaBrokers = "host1:port1,host2:port2,..."
val kafkaTopics  = Set("topic1", "topic2", ...)

val sparkConf = new SparkConf().setAppName("...")
val ssc = new StreamingContext(sparkConf, Seconds(2))

// Create direct kafka stream with kafkaBrokers and kafkaTopics
val kafkaParams = Map[String, String]("metadata.broker.list" -> kafkaBrokers)
val messages =
  KafkaUtils.createDirectStream[String,String,StringDecoder,StringDecoder](
    ssc, kafkaParams, kafkaTopics)

messages.flatMap {case (topic,text) => toWords(text).map(w=>((w,topic),1L))}
.reduceByKey (_ + _)
.map {case ((word, topic), n) => (word, (path, n))}
.groupByKey
.map {case (word, list) => (word, sortByCount(list))}
```

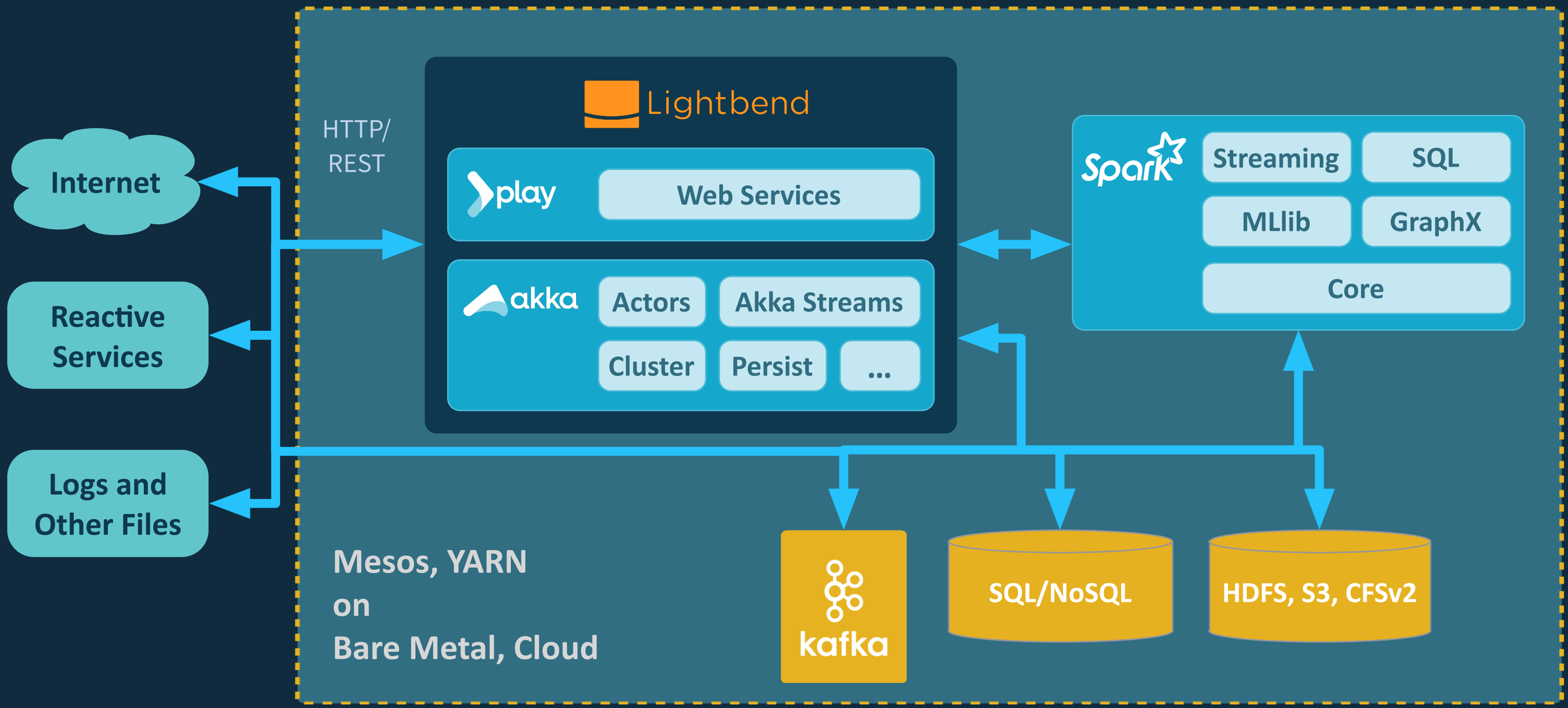
```
val ssc = new StreamingContext(sparkConf, Seconds(2))

// Create direct kafka stream with kafkaBrokers and kafkaTopics
val kafkaParams = Map[String, String]("metadata.broker.list" -> kafkaBrokers)
val messages =
  KafkaUtils.createDirectStream[String,String,StringDecoder,StringDecoder](
    ssc, kafkaParams, kafkaTopics)

messages.flatMap {case (topic,text) => toWords(text).map(w=>((w,topic),1L))}
.reduceByKey (_ + _)
.map {case ((word, topic), n) => (word, (path, n))}
.groupByKey
.map {case (word, list) => (word, sortByCount(list))}
.saveAsTextFiles("/path/to/output")

ssc.start()
ssc.awaitTermination()
```

An Architecture for Fast Data

Fast as in Streaming. Why?

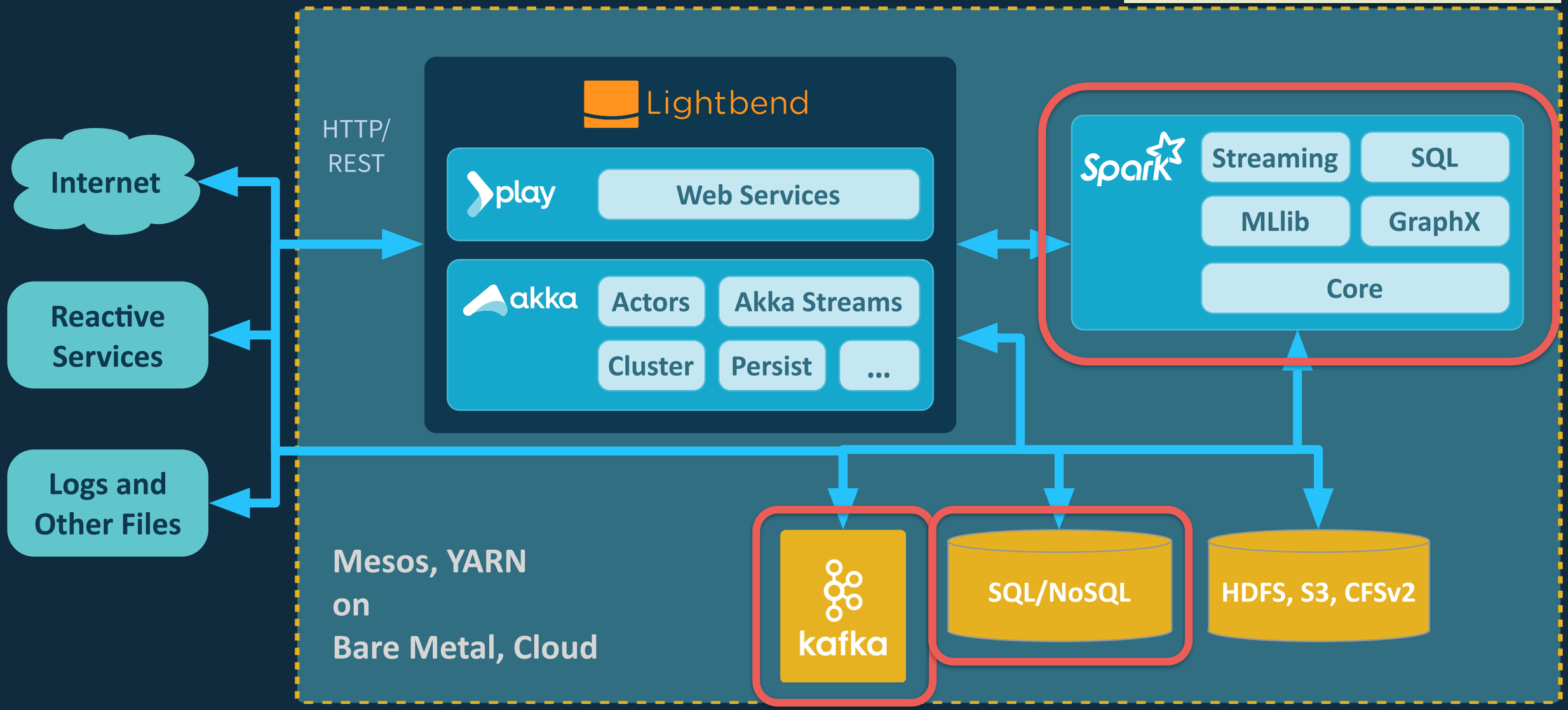
- Update a search engine in real time as web page or documents change.
- Train a SPAM filter with every email.
- Detect anomalies as they happen through processing of logs and monitoring data.

Fast Data Architecture



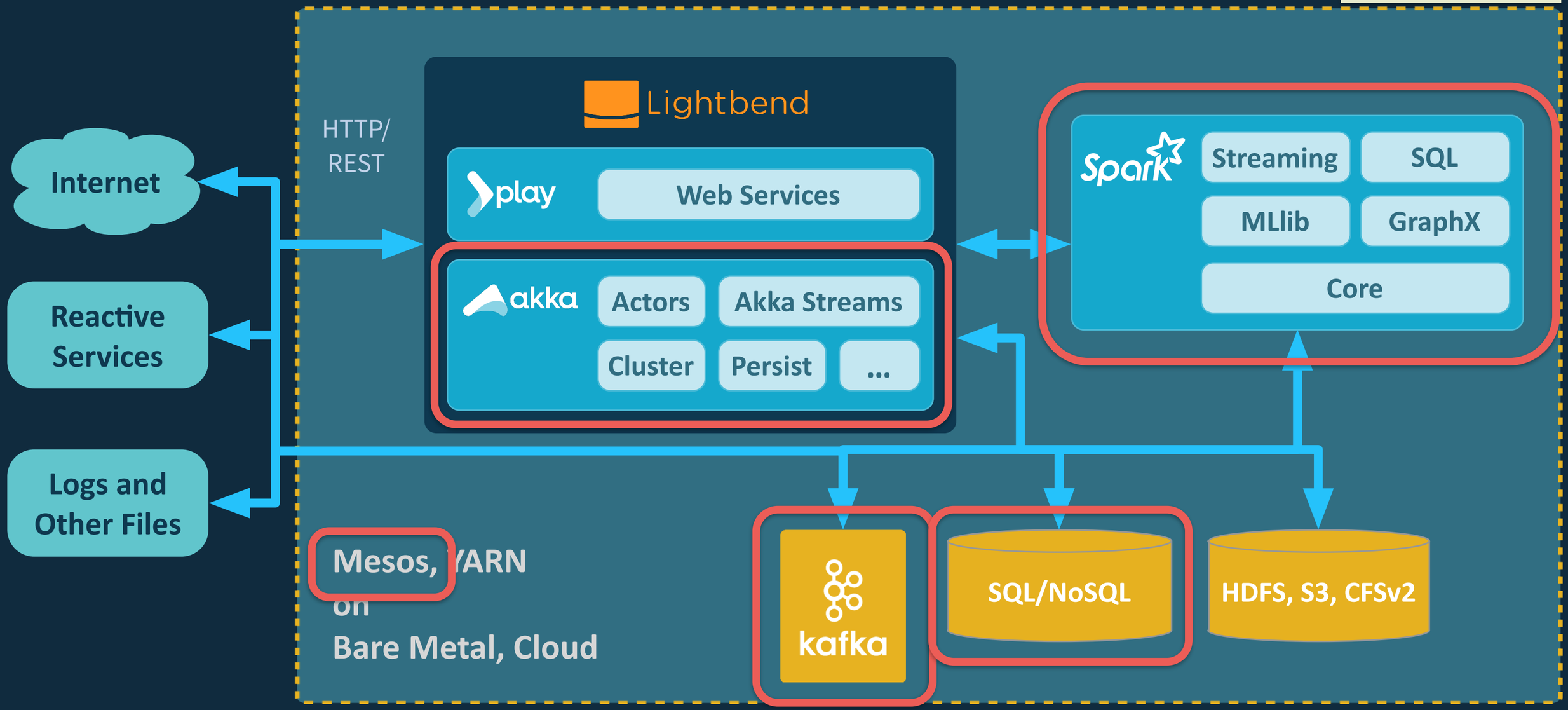
Fast Data Architecture

Core of Spark, Kafka, and Cassandra

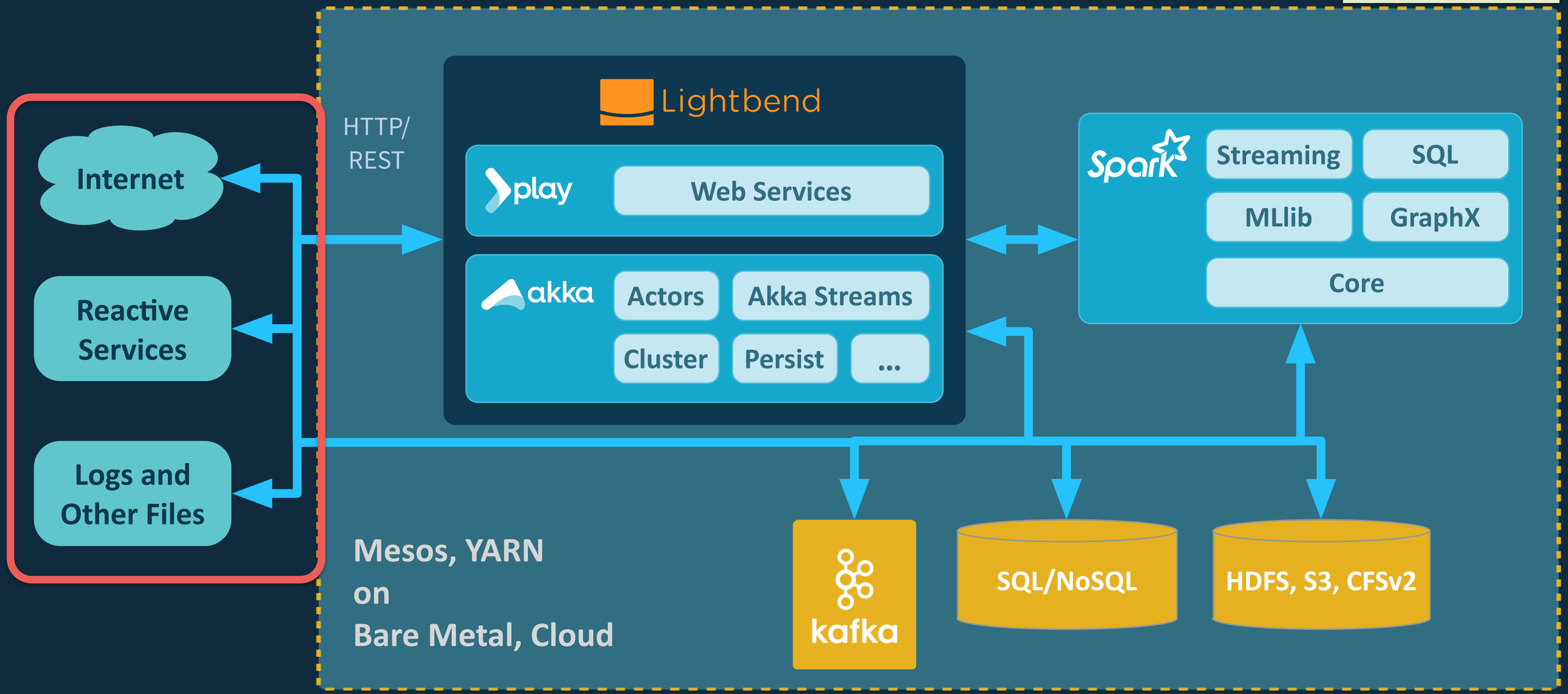


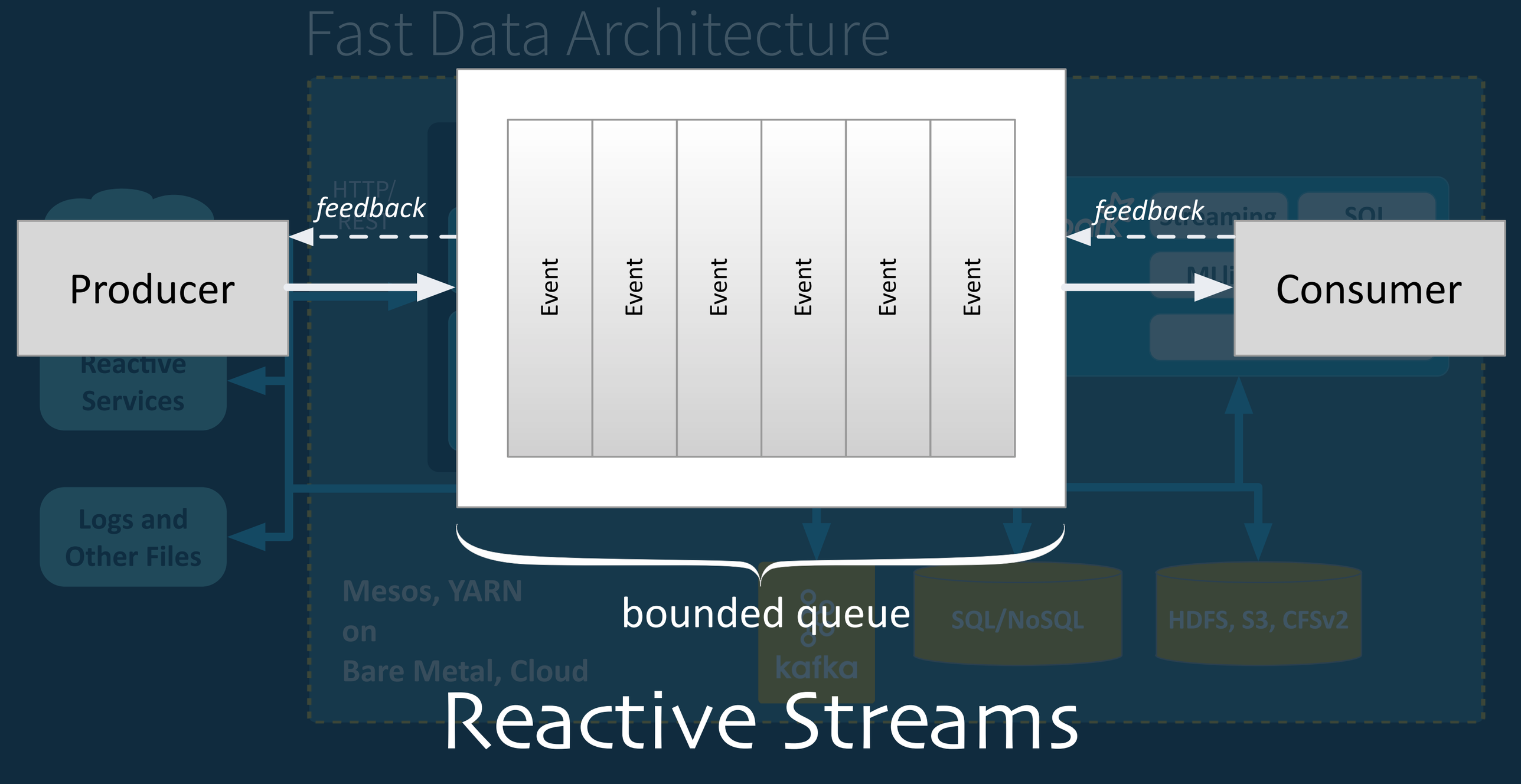
Fast Data Architecture

"SMACK" Stack



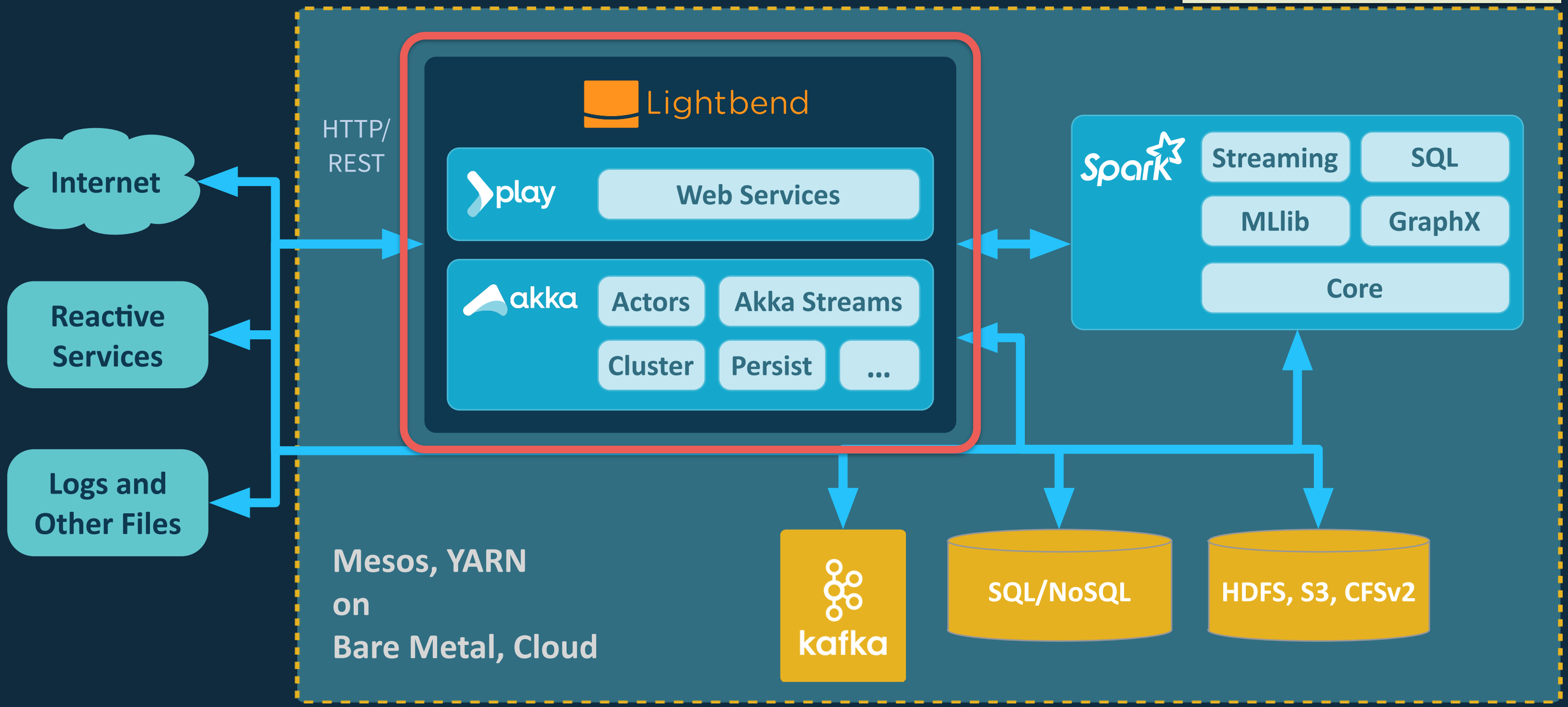
Fast Data Architecture




Fast Data Architecture
HTTP/
REST
feedback
feedback
Producer
Consumer
Event
Event
Event
Event
Event
Event
Reactive
Services
Logs and
Other Files
Mesos, YARN
on
Bare Metal, Cloud
bounded queue
kafka
SQL/NoSQL
HDFS, S3, CFSv2
Reactive Streams

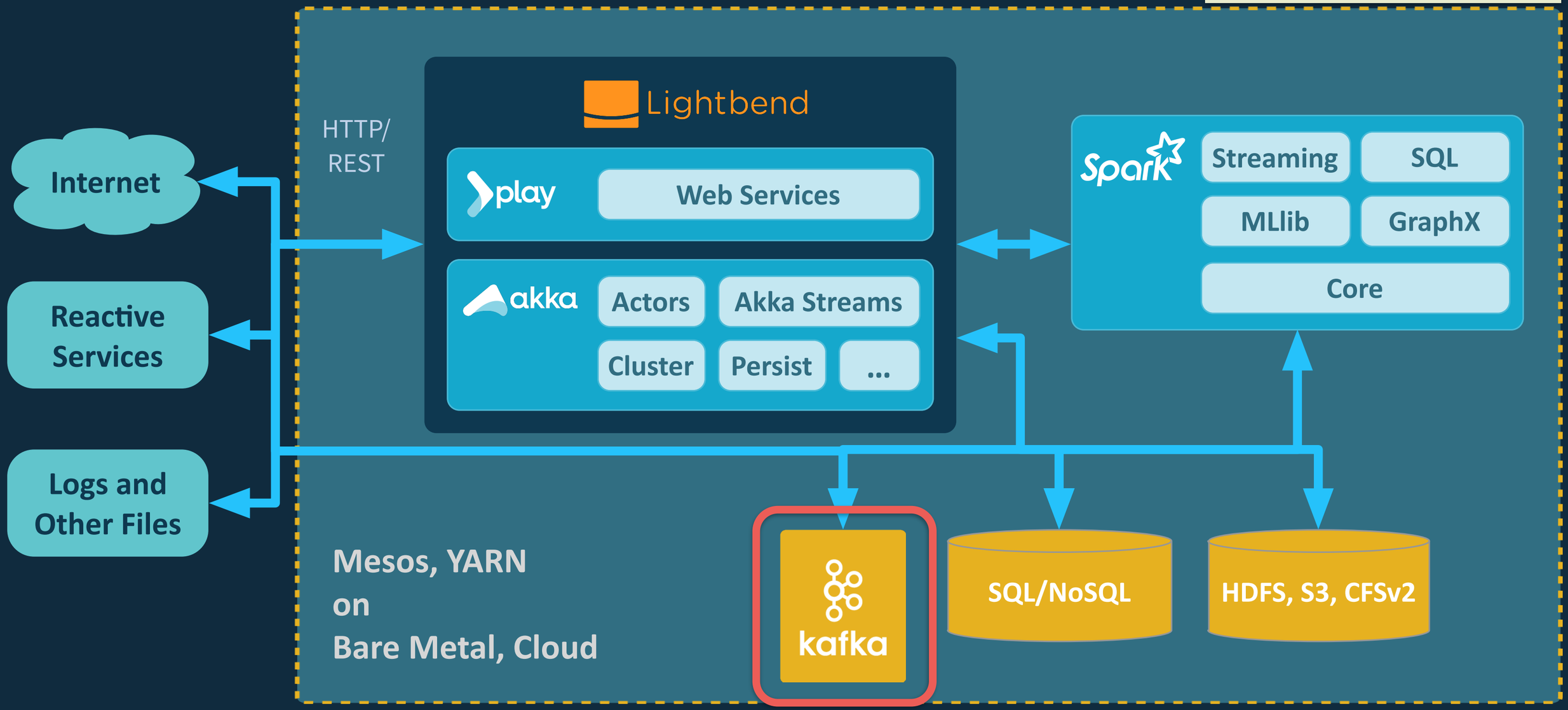
Fast Data Architecture

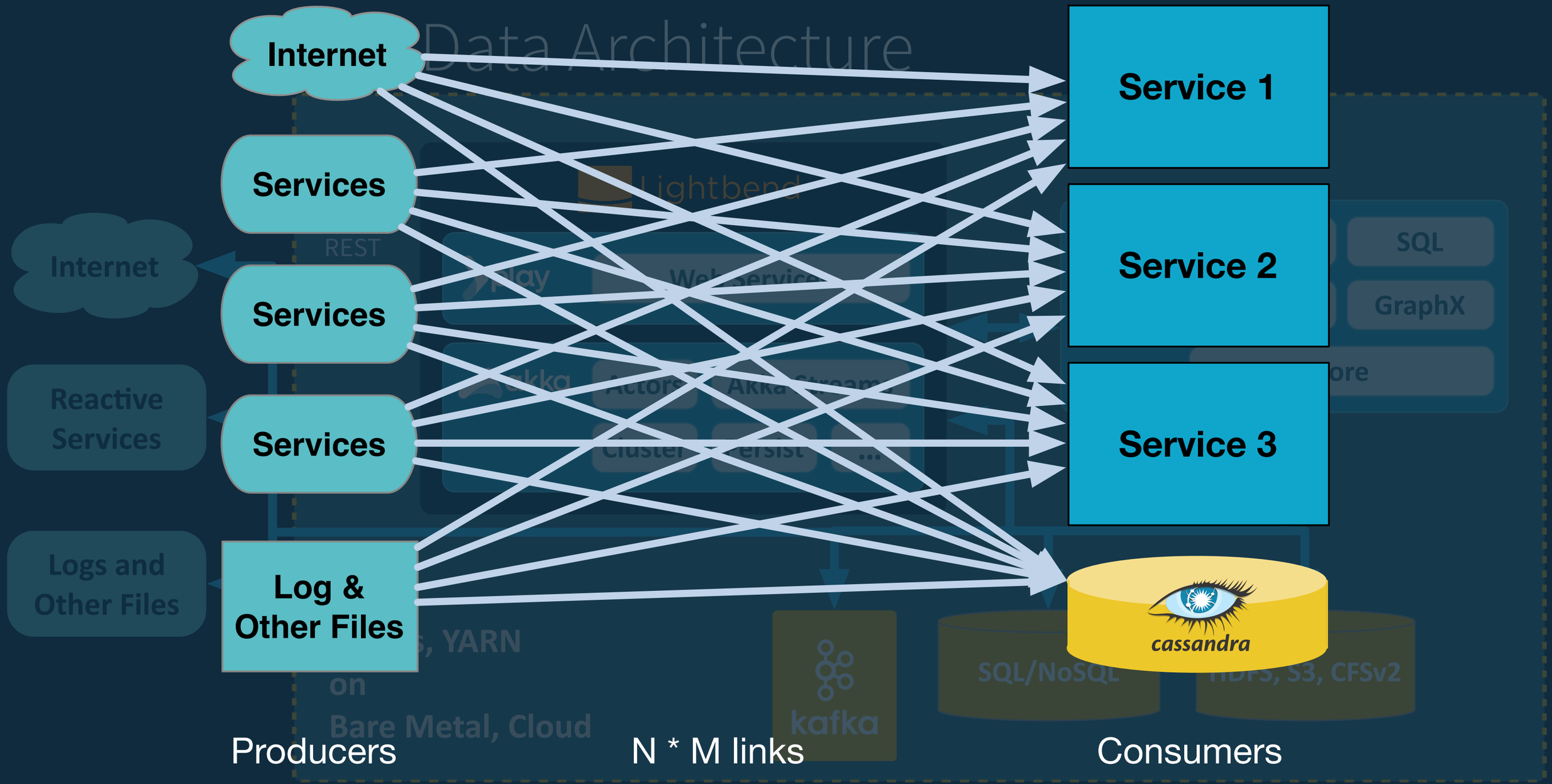
Lightbend Reactive Platform

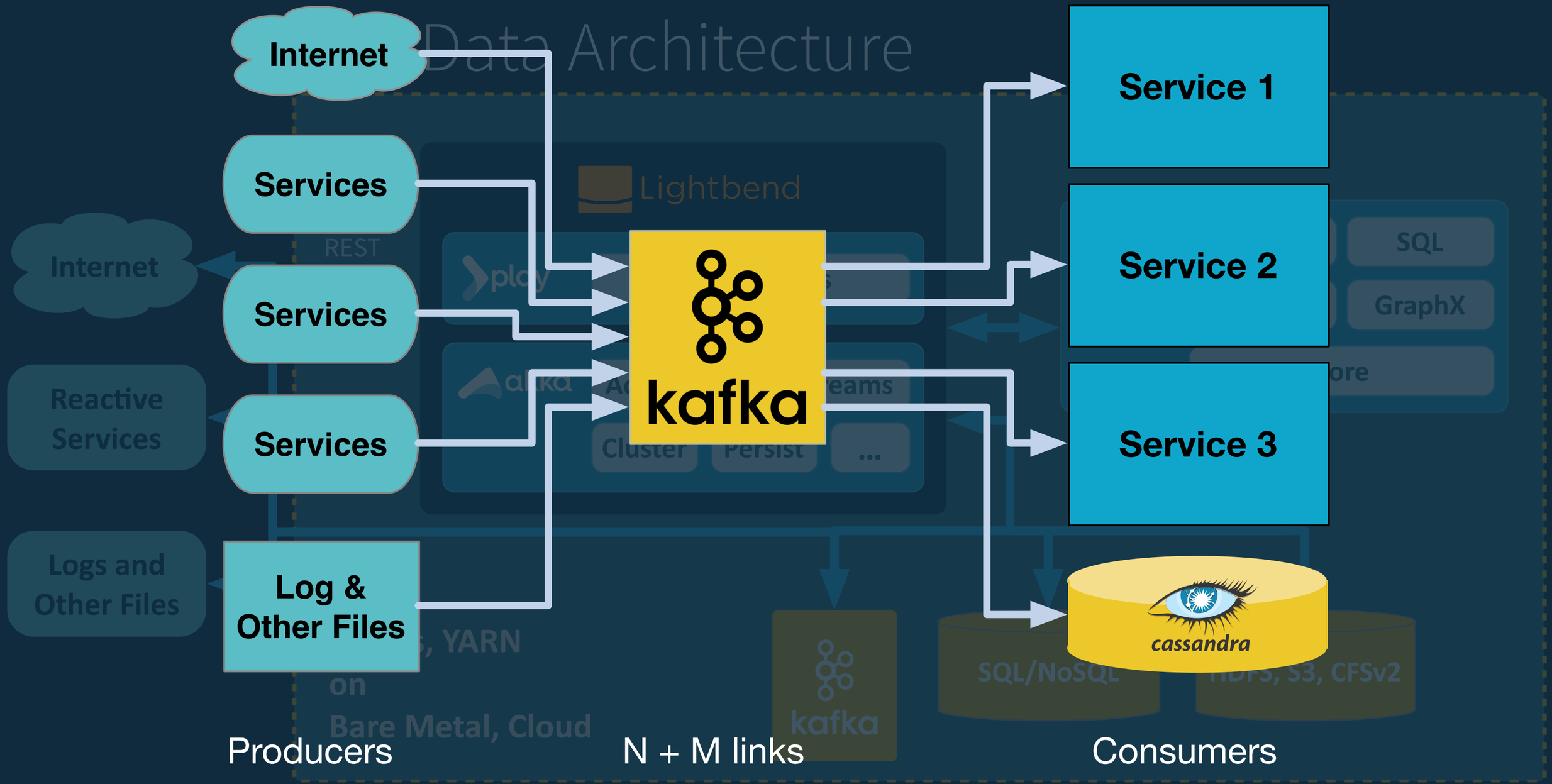


Fast Data Architecture

Kafka for Stream Storage

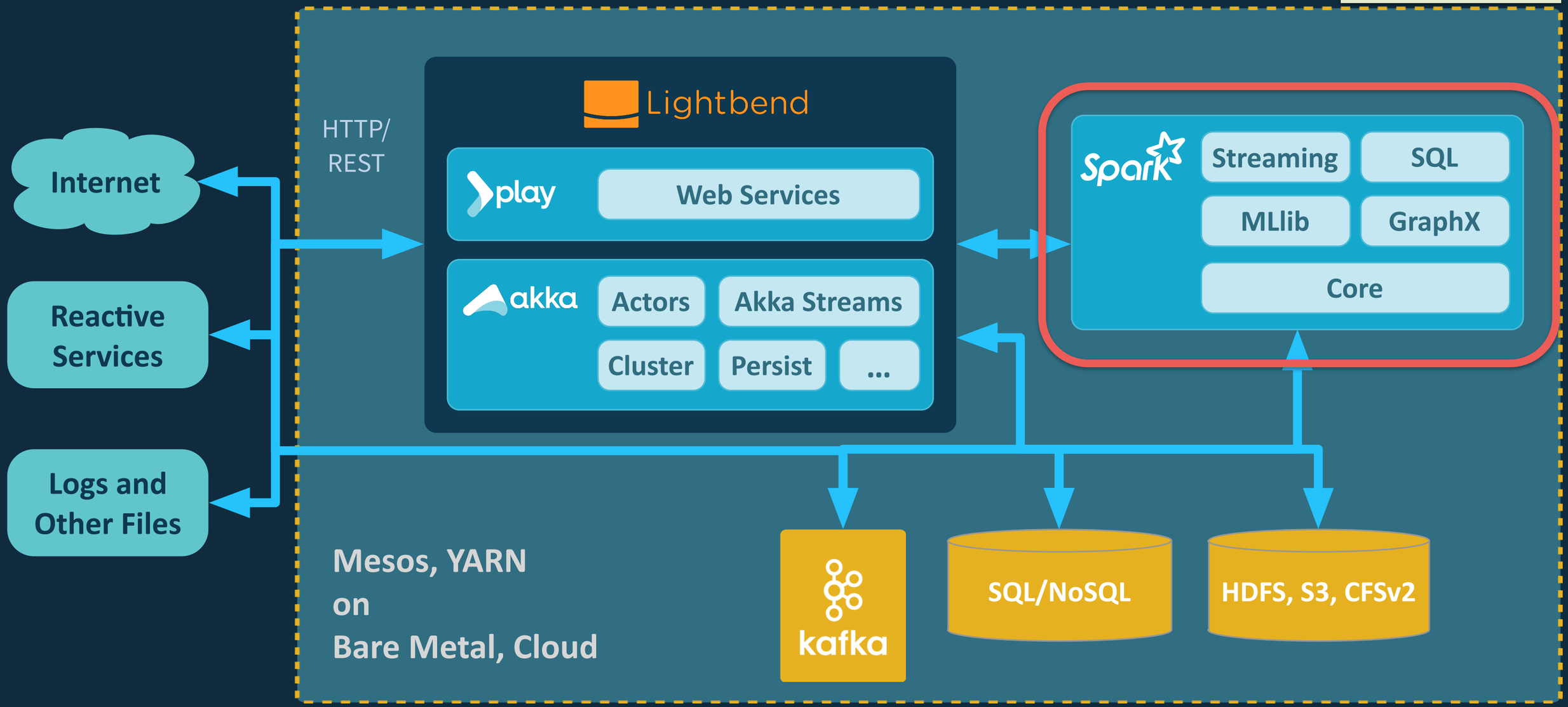



Data Architecture
Internet
Services
Services
Services
Log &
Other Files
Internet
Reactive
Services
Logs and
Other Files
Service 1
Service 2
Service 3
cassandra
kafka
SQL
GraphX
SQL/NoSQL
Producers
N * M links
Consumers

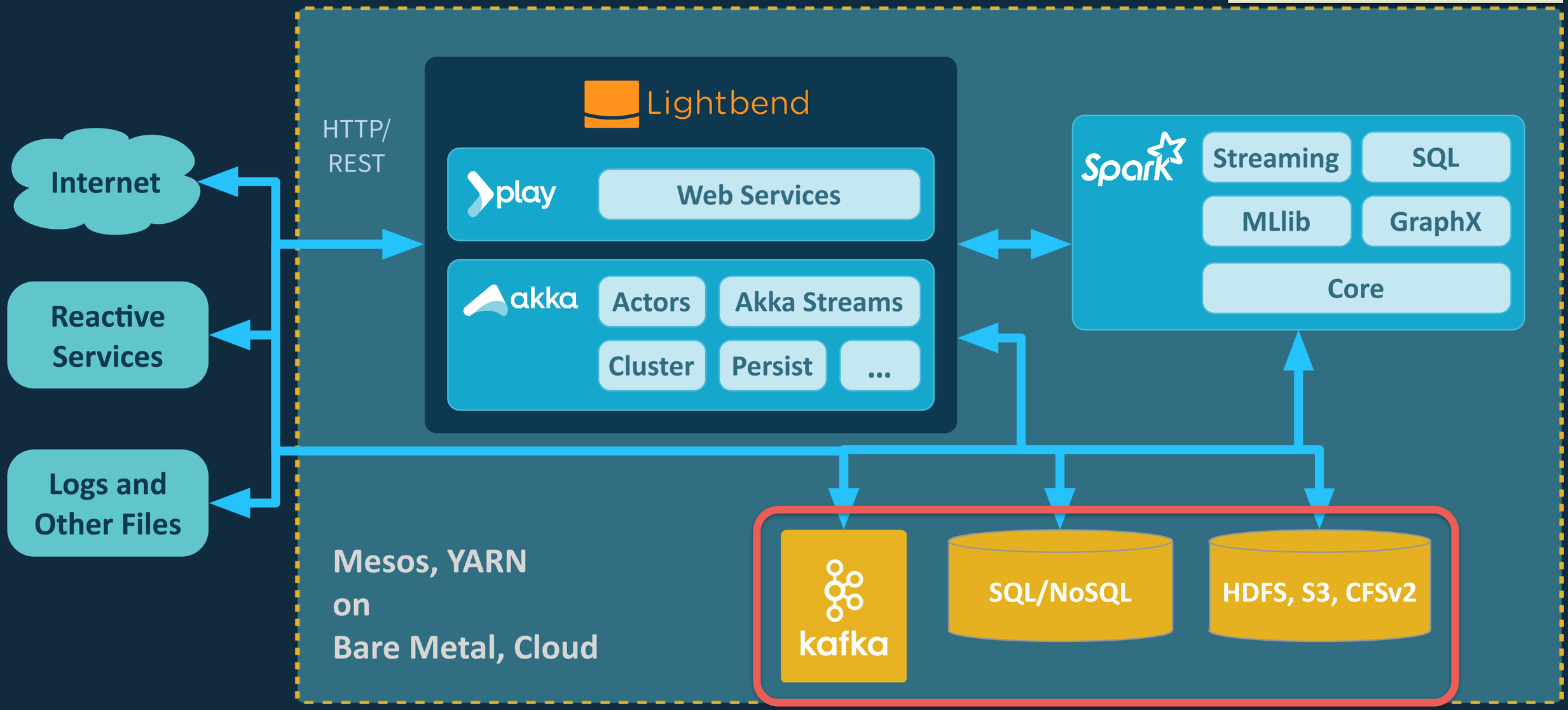

Data Architecture
Internet
Services
Services
Services
Log &
Other Files
Internet
Reactive
Services
Logs and
Other Files
Lightbend
REST
play
akka
Streams
Cluster
Persist
...
kafka
Service 1
Service 2
Service 3
SQL
GraphX
ore
cassandra
SQL/NoSQL
HDFS, S3, CFSv2
kafka
, YARN
on
Bare Metal, Cloud
Producers
N + M links
Consumers

Fast Data Architecture

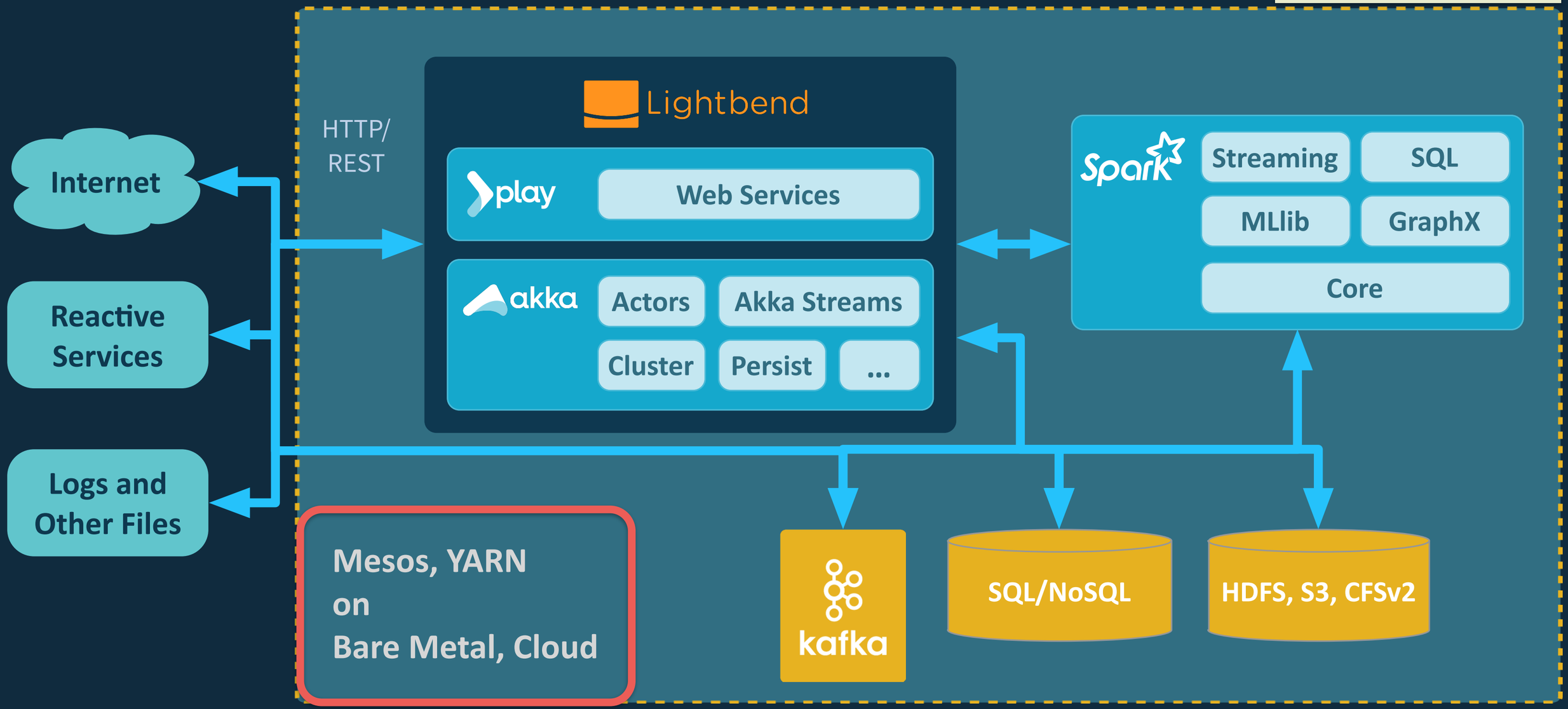
Minibatch Processing



Fast Data Architecture



Fast Data Architecture



- Next Steps
  - Learn - Fast Data: Big Data Evolved
  - Watch - Using Spark, Kafka, Cassandra and Akka on Mesos for Real-Time Personalization
  - Review - Spark success stories by Lightbend clients


Lightbend