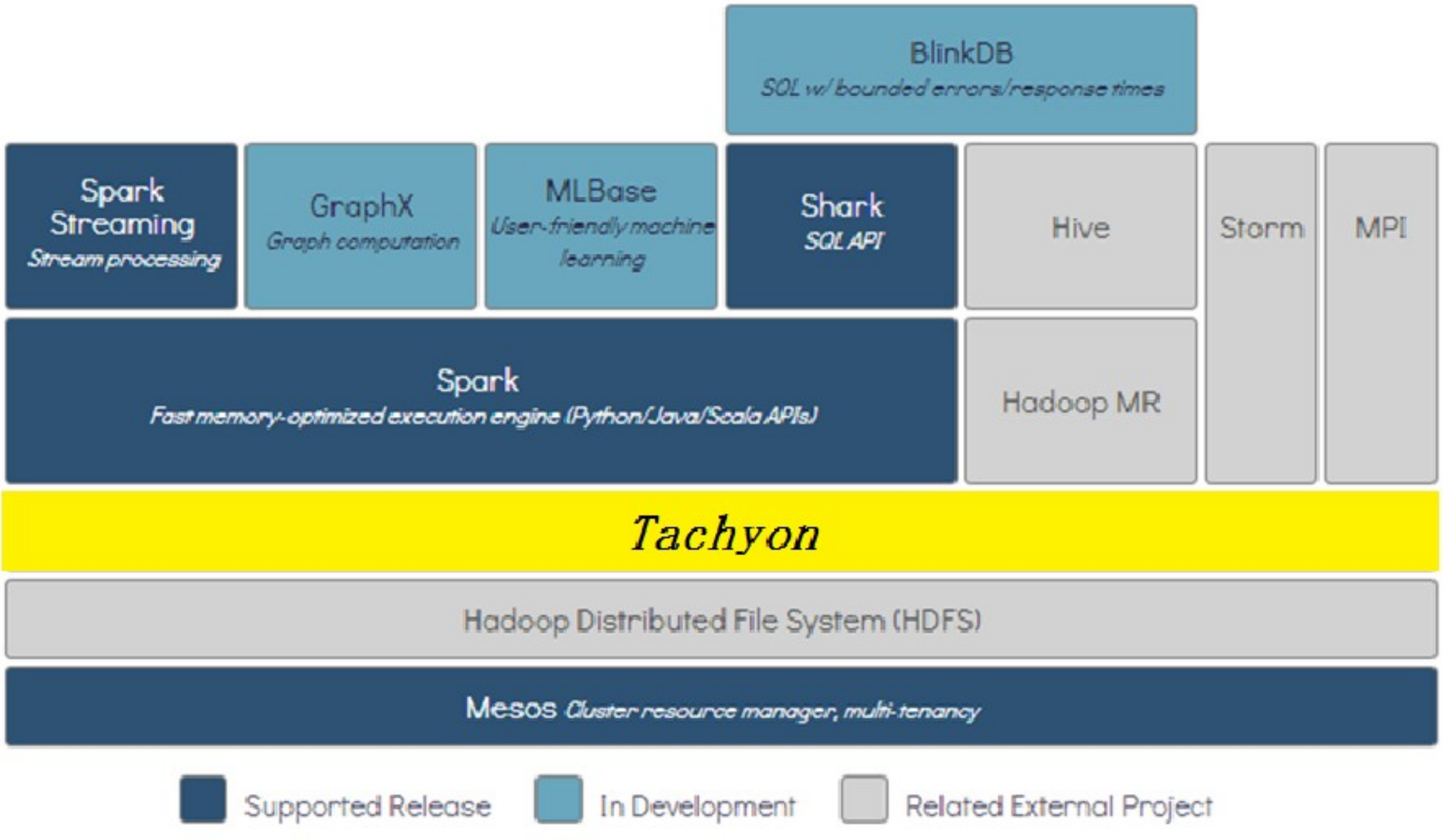
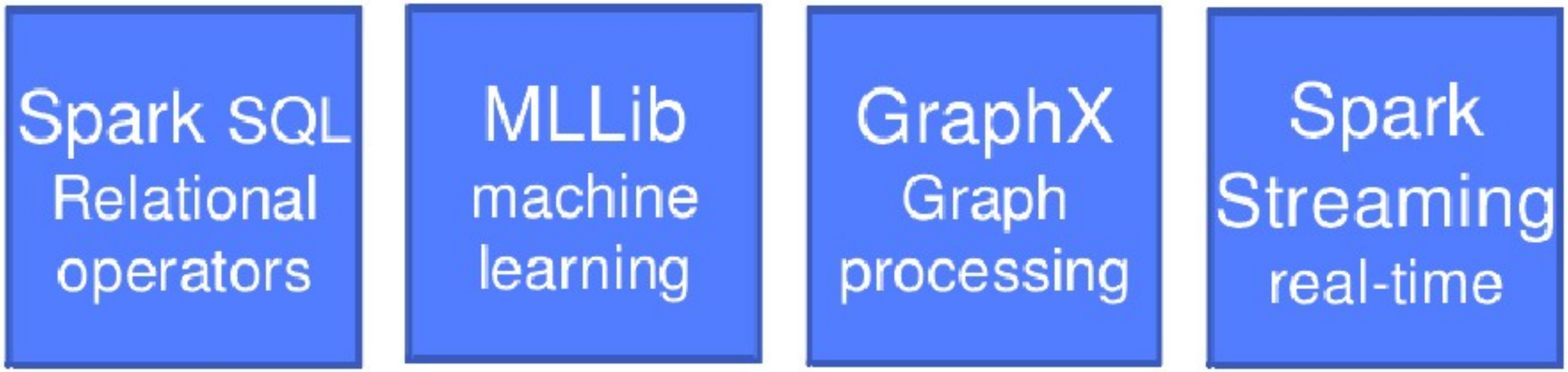
Spark知识分享

Spark知识分享

- Spark 简介
- Spark SQL 简介
- Spark Streaming 简介
- Spark应用场景

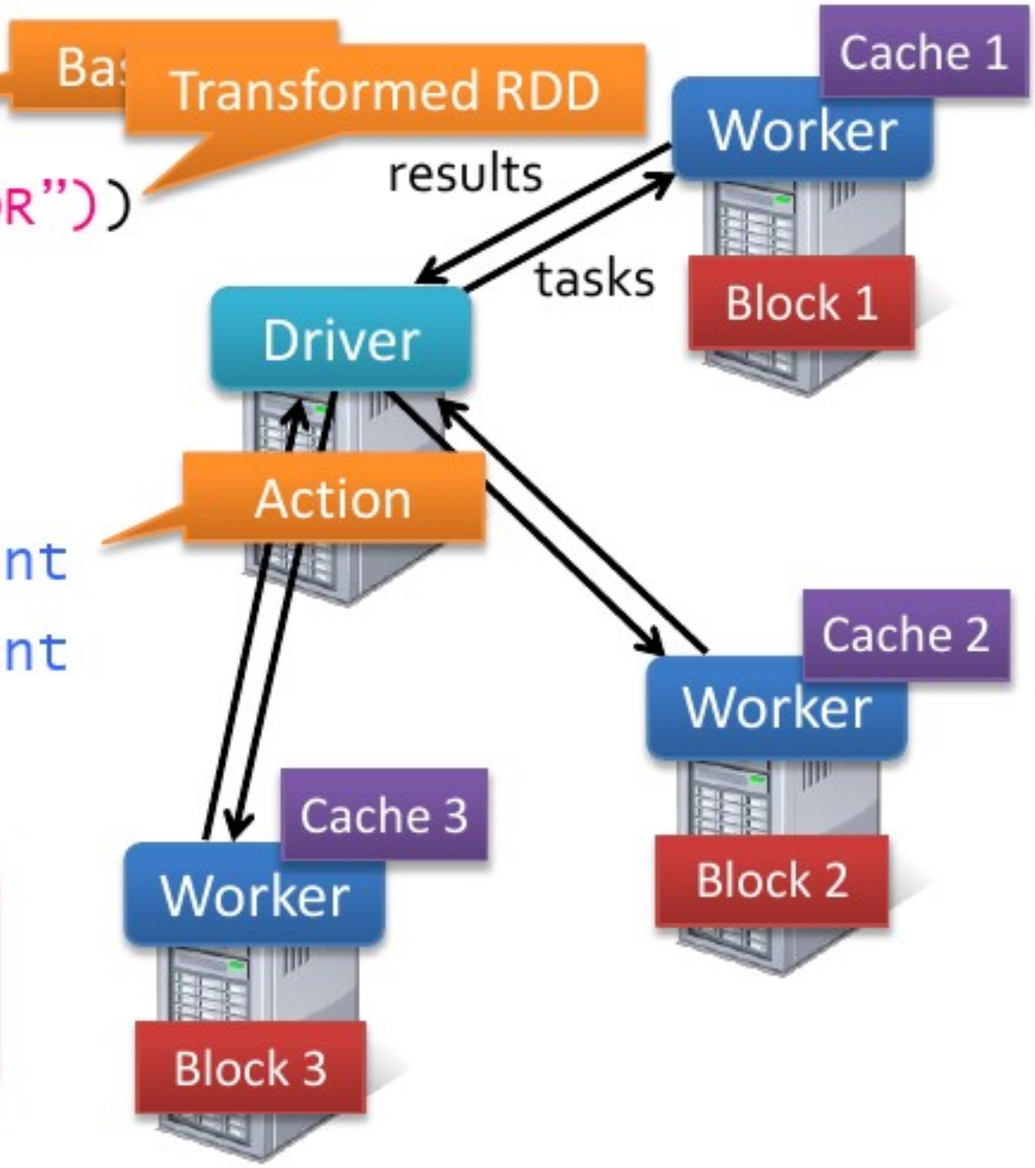
- Hadoop在2003年从Nutch发展到Lucene，在Yahoo成长，进入Apache孵化，2008年获得大量使用。但一直存在**MR算法少、每次Reduce都需要磁盘读写、MR需要成对出现、Master节点调度慢、单节点**等等问题。
- Spark2007年在Yahoo起步，用于改善MR算法。2009年独立为一个项目，2010年开源，2013年进入Apache孵化。被称为以下一代计算平台。
- Berkeley大学成为大数据技术中心，Berkeley Data Analysis Stack(BDAS)逐步形成大数据平台。




Spark SQL
Relational operators
MLLib
machine learning
GraphX
Graph processing
Spark Streaming
real-time
Spark Runtime
Cluster Managers
YARN, Mesos, AWS
Data Sources
HDFS, S3, Cassandra, Hana

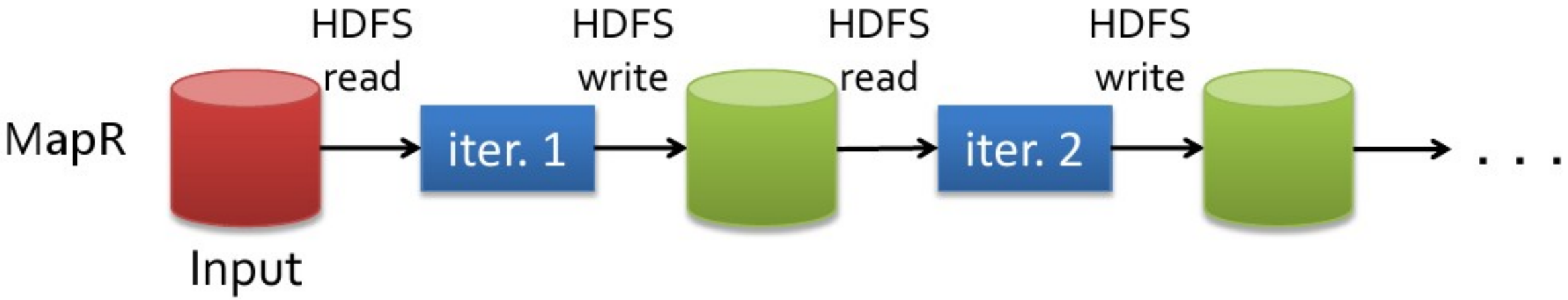
- 基于Hadoop的HDFS，Spark采用Driver、Worker的主从结构，由Driver节点调度，负责任务分配、资源安排、结果汇总、容错等处理。Worker节点主要是存放数据和进行计算。
- 第一次从外设读取数据，之后主要在内存计算。
- 案例中涉及到RDD、Tranformation、Action等操作，从中可以发现，处理方式是MR+SQL语法，包括map、reduce、count、groupby、join、union等

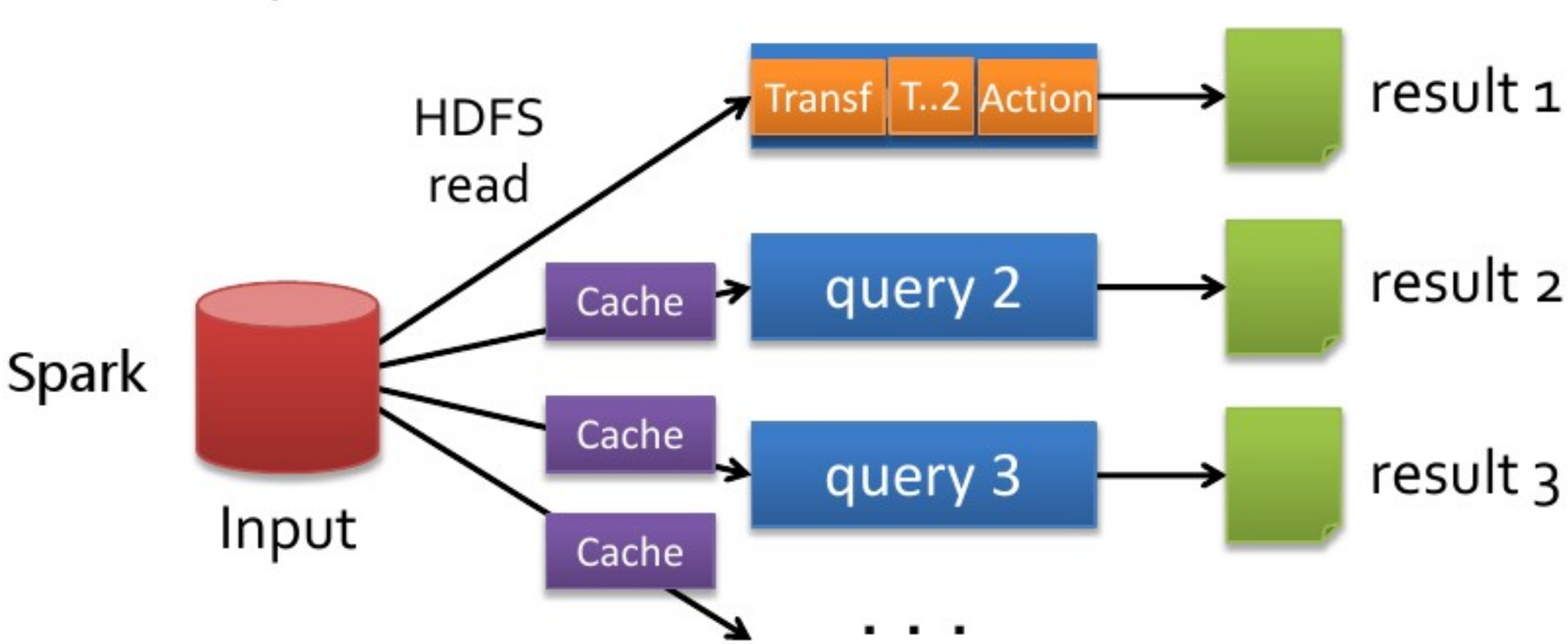
```
lines = spark.textFile("hdfs://...")
errors = lines.filter(_.startsWith("ERROR"))
messages = errors.map(_.split('\t')(2))
cachedMsgs = messages.cache()

cachedMsgs.filter(_.contains("foo")).count
cachedMsgs.filter(_.contains("bar")).count
. . .
```



Result: scaled to 1 TB data in 5-7 sec
(vs 170 sec for on-disk data)

- MapReduce每次读写，都需要序列化到磁盘。一个复杂任务，需要多次处理，几十次磁盘读写。
- Spark只需要一次磁盘读写，大部分处理在内存中进行。



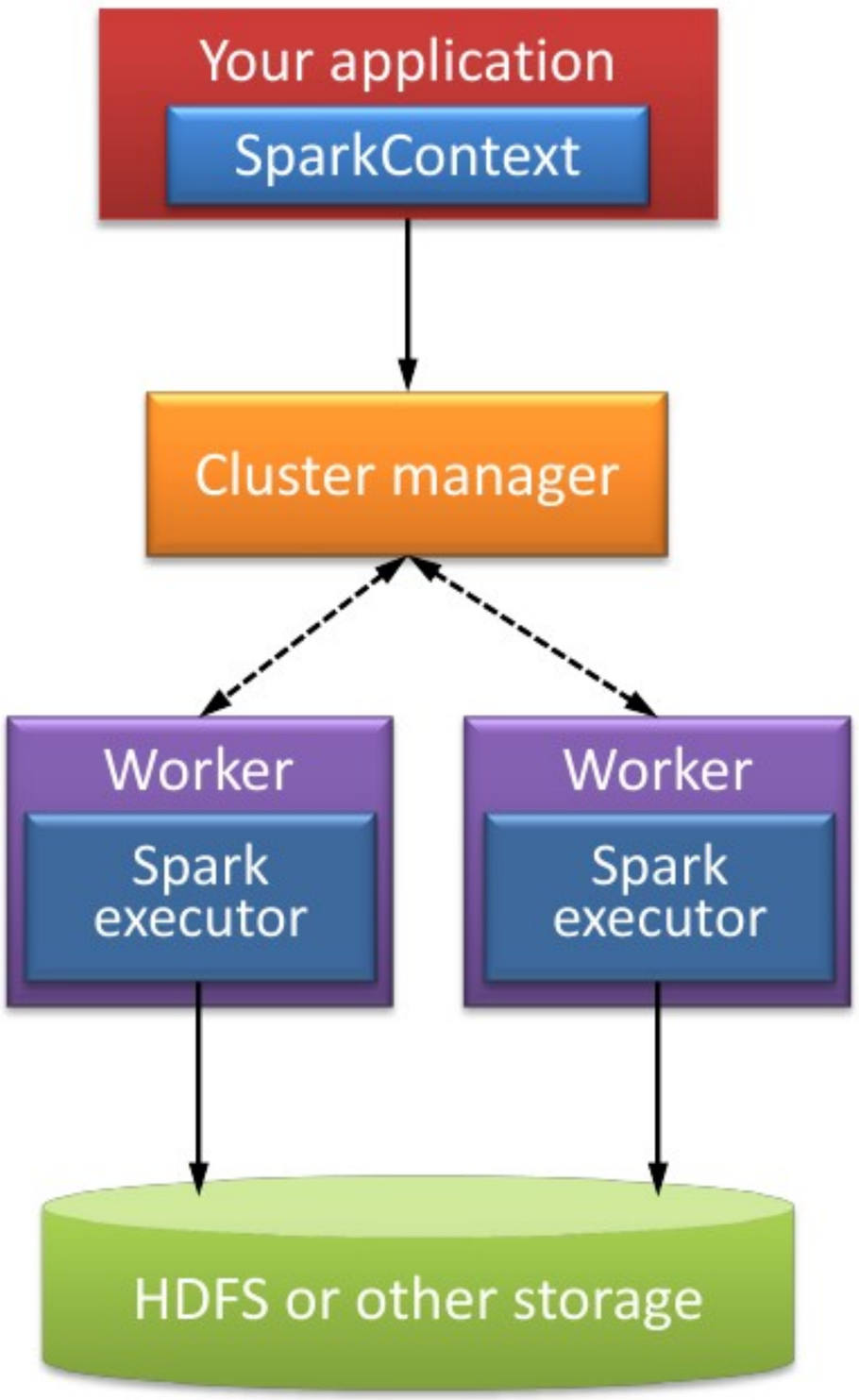


I/O and serialization can take **90%** of the time

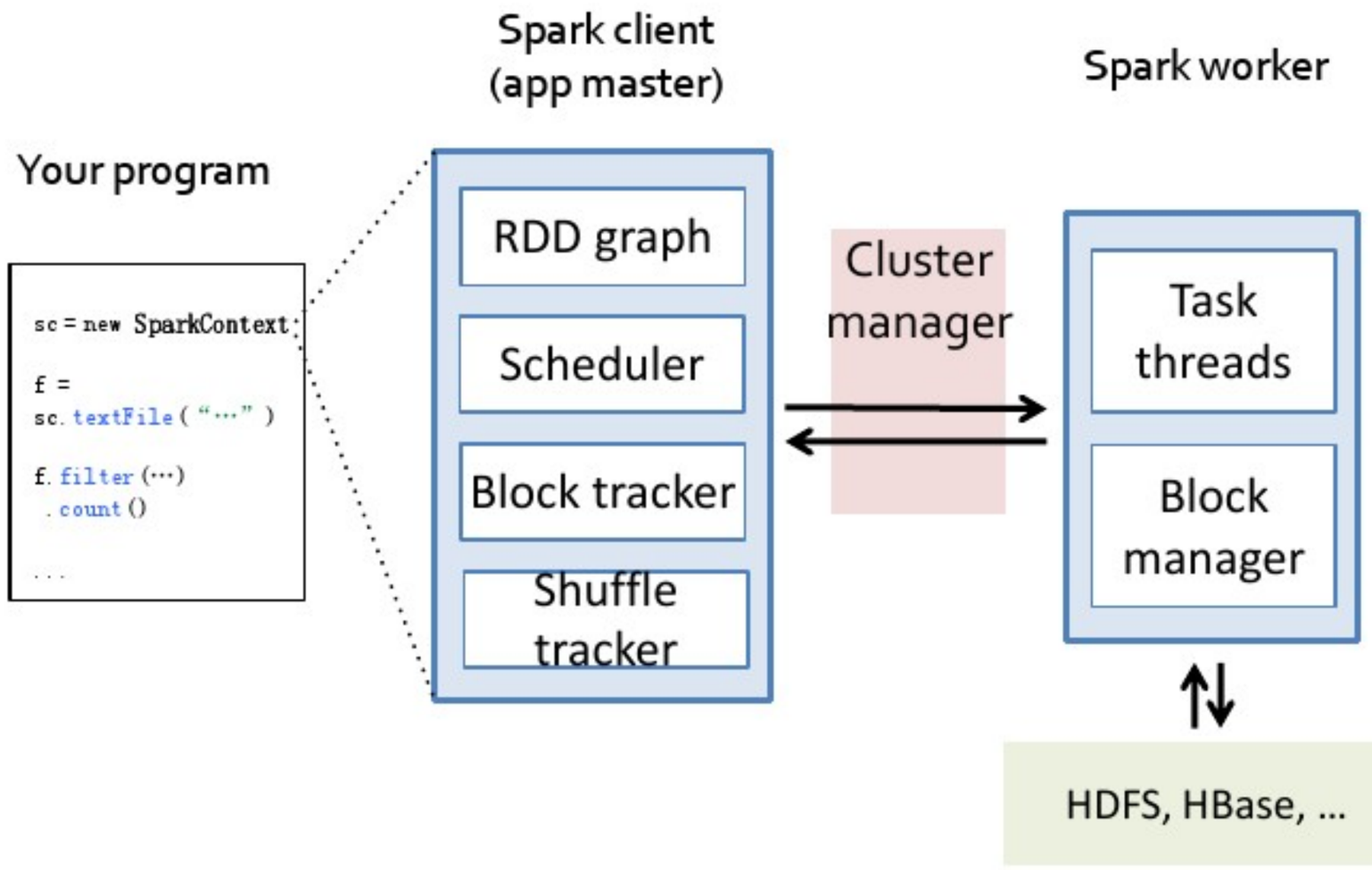
- RDD是一个数据集，不可改变，分布在集群上；通过DAG来实现自动数据恢复；支持内存物化(Cache)和硬盘物化(Checkpoint)，来保存中间结果；

Transformations	*map*(*f* : T ⇒ U)	:	RDD[T] ⇒ RDD[U]
	filter(*f* : T ⇒ Bool)	:	RDD[T] ⇒ RDD[T]
	flatMap(*f* : T ⇒ Seq[U])	:	RDD[T] ⇒ RDD[U]
	sample(*fraction* : Float)	:	RDD[T] ⇒ RDD[T] (Deterministic sampling)
	groupByKey()	:	RDD[(K, V)] ⇒ RDD[(K, Seq[V])]
	reduceByKey(*f* : (V, V) ⇒ V)	:	RDD[(K, V)] ⇒ RDD[(K, V)]
	union()	:	(RDD[T], RDD[T]) ⇒ RDD[T]
	join()	:	(RDD[(K, V)], RDD[(K, W)]) ⇒ RDD[(K, (V, W))]
	cogroup()	:	(RDD[(K, V)], RDD[(K, W)]) ⇒ RDD[(K, (Seq[V], Seq[W]))]
	crossProduct()	:	(RDD[T], RDD[U]) ⇒ RDD[(T, U)]
	mapValues(*f* : V ⇒ W)	:	RDD[(K, V)] ⇒ RDD[(K, W)] (Preserves partitioning)
	sort(*c* : Comparator[K])	:	RDD[(K, V)] ⇒ RDD[(K, V)]
	partitionBy(*p* : Partitioner[K])	:	RDD[(K, V)] ⇒ RDD[(K, V)]
Actions	*count*()	:	RDD[T] ⇒ Long
	collect()	:	RDD[T] ⇒ Seq[T]
	reduce(*f* : (T, T) ⇒ T)	:	RDD[T] ⇒ T
	lookup(*k* : K)	:	RDD[(K, V)] ⇒ Seq[V] (On hash/range partitioned RDDs)
	save(*path* : String)	:	Outputs RDD to a storage system, *e.g.*, HDFS

所有的操作都是针对RDD，类似于MPPDB的技术实现：分布、并行、内存计算和压缩。优于MPP点在于毫秒级的调度，适用于复杂计算；逊于MPP点在于数据处理没有SQL方便和强大。

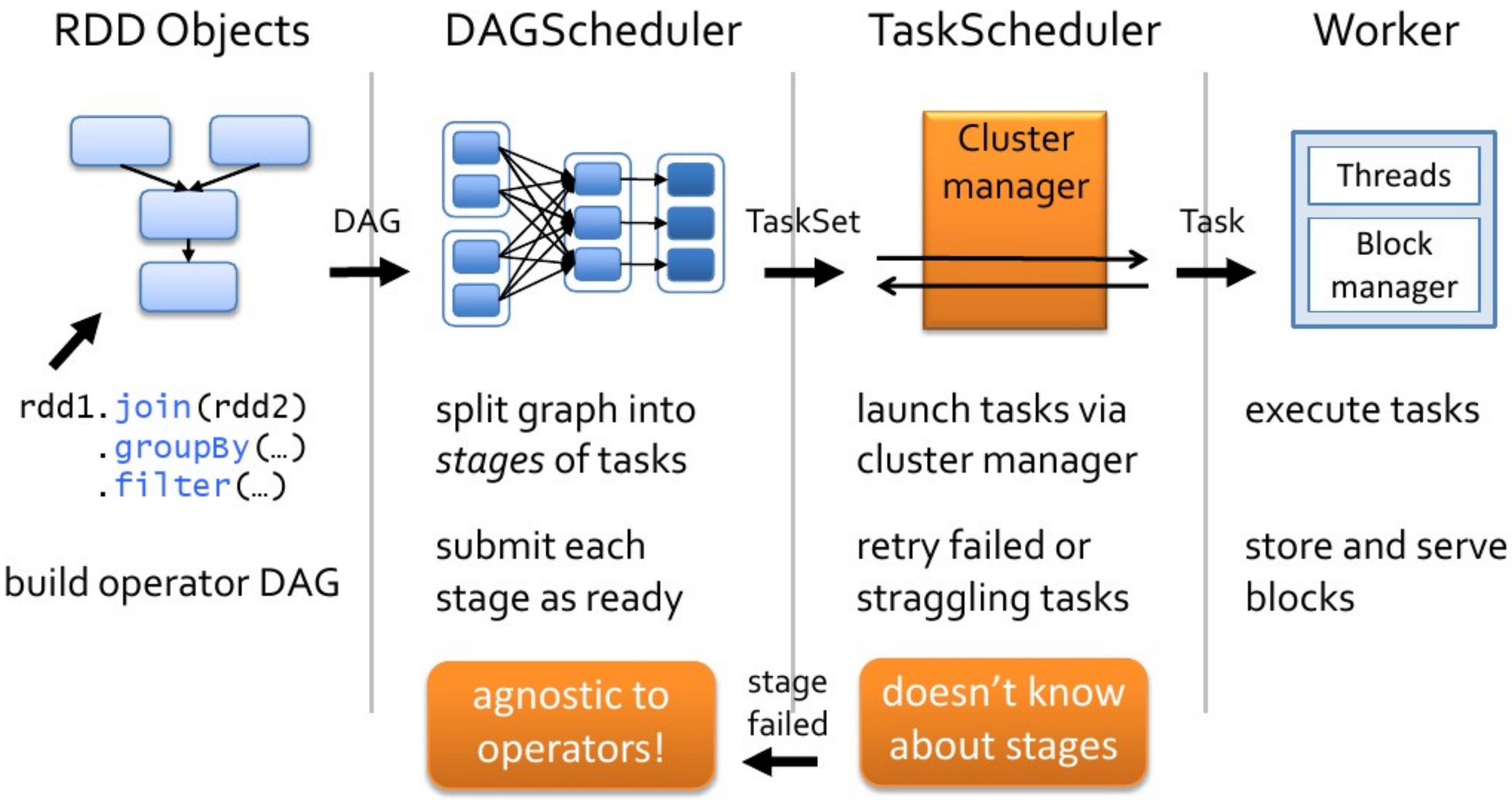


- RDD graph记录各个RDD的来源；Scheduler进行快速调度；Block tracker跟踪HDFS块位置；Shuffle记录RDD之间的数据分发。Cluster采用Yarn等产品。Task在线程上执行。



磁盘数据本地化，内存数据本地化，计算本地化。

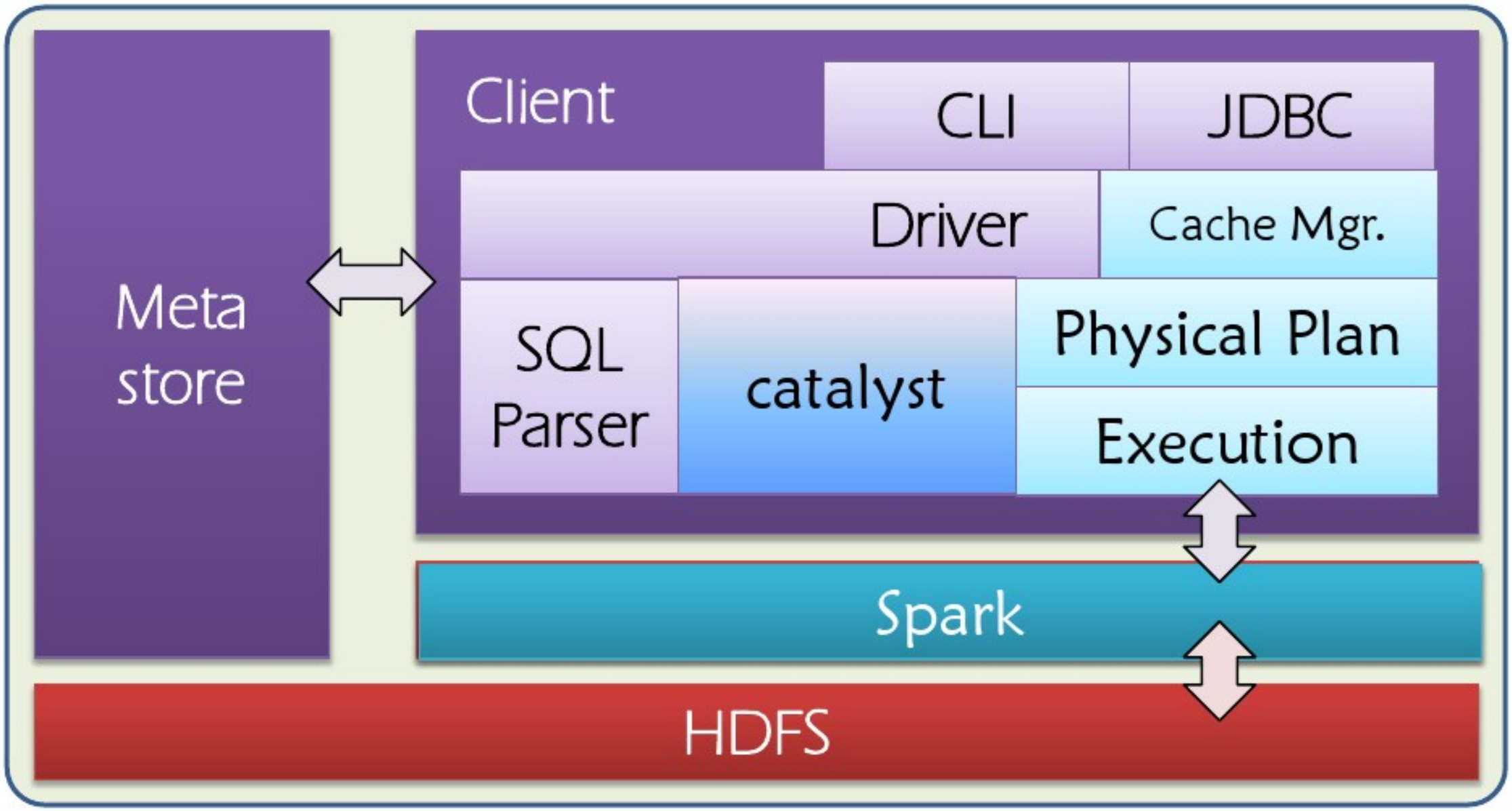
- 容错：Driver中记录每个RDD生成的图，在RDD失效的时候，能够根据这个链条，重新生成出RDD，确保所有的RDD都是可以再生。RDD在内存中做Partition，作为备份。
- 延迟调度：RDD包括Tranform、Action两类操作，只有在Action的时候才处理数据，成为延迟调度，是一种聪敏的方法。
- 包含FIFO和Fair Schedule两种调度算法。



Spark知识分享

- Spark 简介
- Spark SQL 简介
- Spark Streaming 简介
- Spark应用场景

- Shark在Hive的架构基础上，改写了“内存管理”、“执行计划”和“执行模块”三个模块，使HQL从MapReduce转到Spark上。



Spark SQL沿袭了Shark的架构，在原有架构上，重写了优化部分，并增加了RDD-Aware optimizer和多语言接口。

- 支持Scala、Java和Python三种语言。
- 支持SQL-92规范和HQL

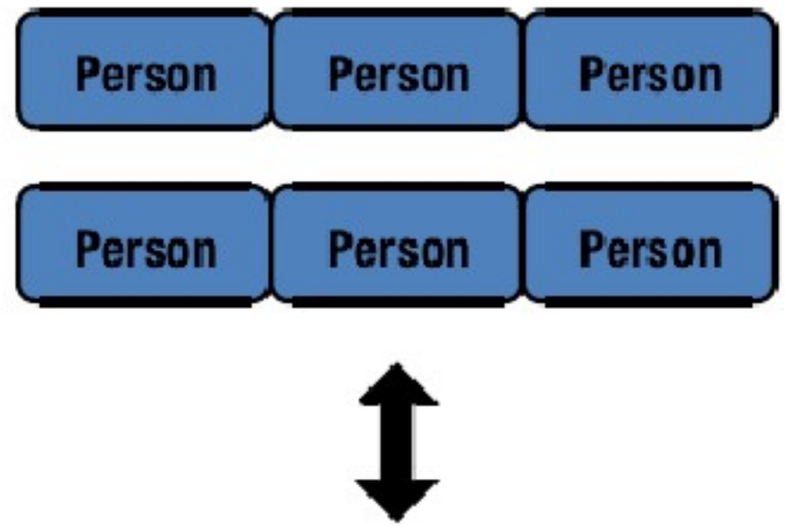
增加了SchemaRDD，读取JSON、Nosql、RDBMS和HDFS数据。

继续兼容Hive和Shark。

Adding Schema to RDDs

Spark + RDDs

Functional transformations on partitioned collections of **opaque** objects.



SQL + SchemaRDDs

Declarative transformations on partitioned collections of **tuples.**

- Scala语言的案例，定义一个Person类型的schema，读取一个文本文件，将读取的值赋值给Person最终赋给people对象，通过resisterAsTable转换成表，供SQL使用
- 后台是将文本文件变成一个RDD，通过Spark来计算得出SQL的查询结果

RDDs into Relations (Scala)

```scala
val sqlContext = new org.apache.spark.sql.SQLContext(sc)
import sqlContext._

// Define the schema using a case class.
case class Person(name: String, age: Int)

// Create an RDD of Person objects and register it as a table.
val people =
  sc.textFile("examples/src/main/resources/people.txt")
    .map(_.split(","))
    .map(p => Person(p(0), p(1).trim.toInt))

people.registerAsTable("people")
```

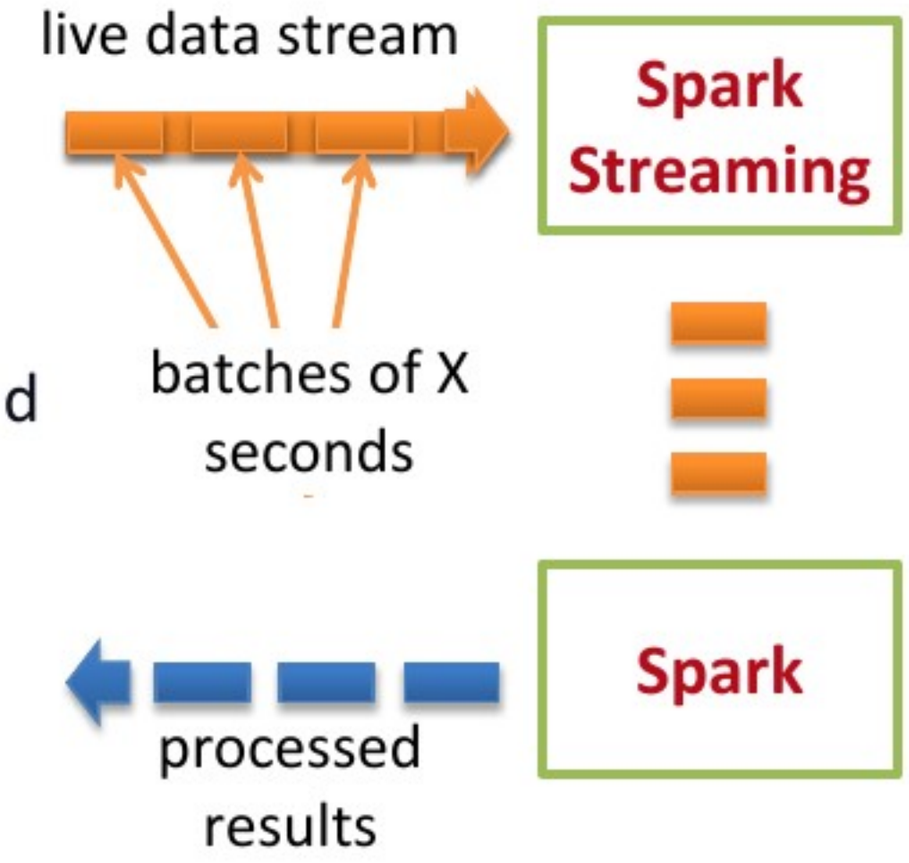
Shark只能通过HQL来操作数据，而spark sql即可以使用RDD，并提供了 schema功能 。

Spark知识分享

- Spark 简介
- Spark SQL 简介
- Spark Streaming 简介
- Spark应用场景

- 1、不同于一般流处理软件，Spark Streaming采用一系列毫秒级的批量处理，实现快速计算。
- 2、将一个需要处理的任务，转化为多个RDD计算，运行在Spark上。

- Chop up the live stream into batches of X seconds
- Spark treats each batch of data as RDDs and processes them using RDD operations
- Finally, the processed results of the RDD operations are returned in batches

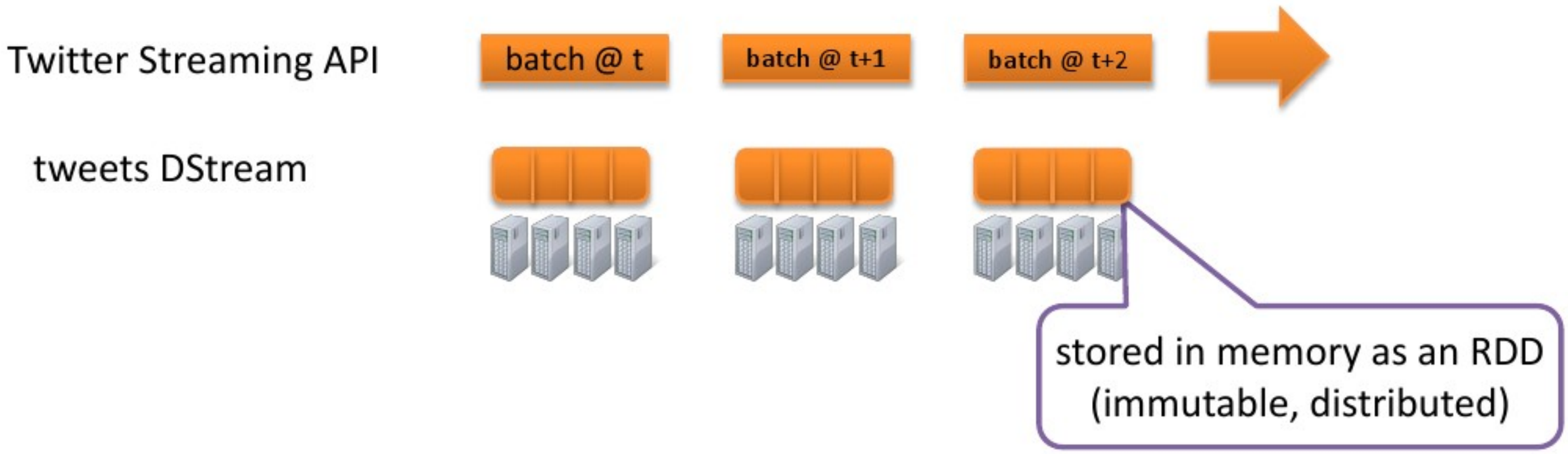


- Batch sizes as low as ½ second, latency of about 1 second

- 1、不同于一般流处理软件，Spark Streaming采用一系列毫秒级的批量处理，实现快速计算。
- 2、将一个需要处理的任务，转化为多个RDD计算，运行在Spark上。

```
val tweets = ssc.twitterStream()
```

DStream: a sequence of RDDs representing a stream of data

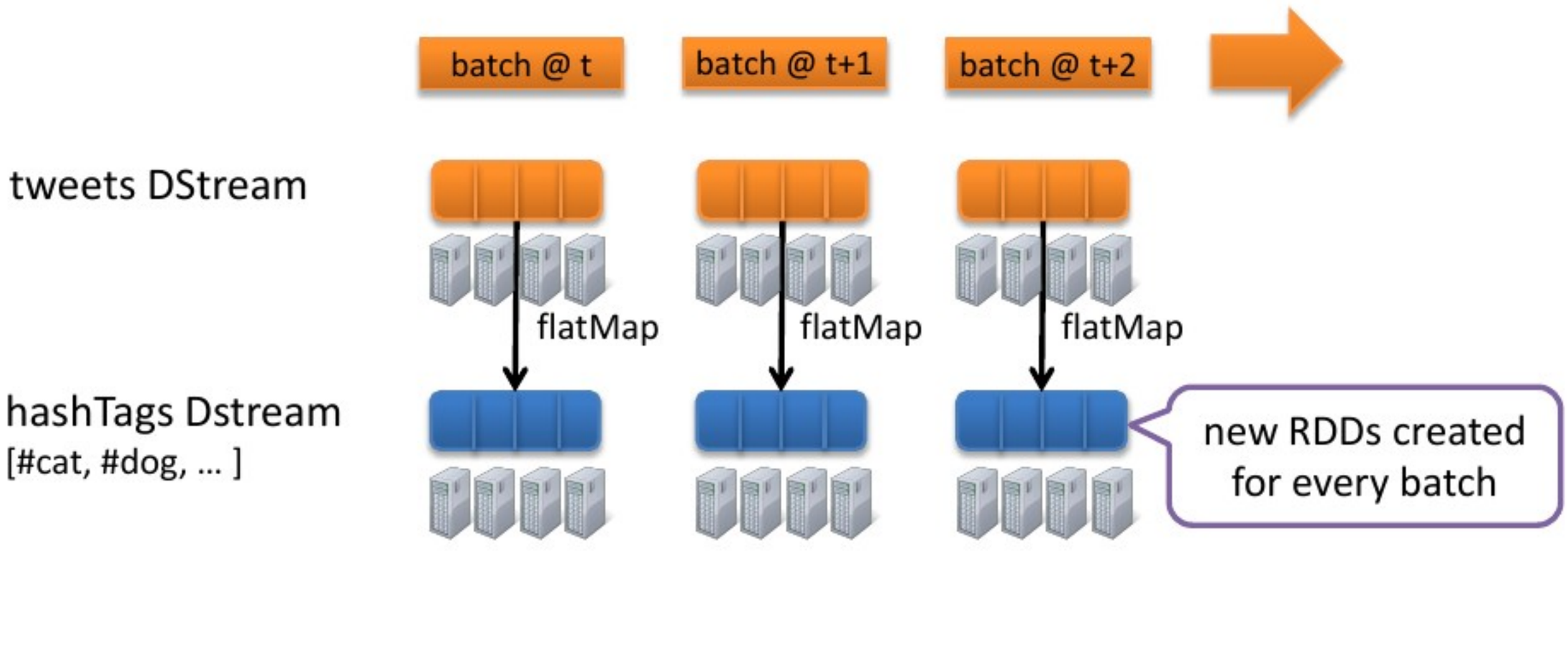


- 1、不同于一般流处理软件，Spark Streaming采用一系列毫秒级的批量处理，实现快速计算。
- 2、将一个需要处理的任务，转化为多个RDD计算，运行在Spark上。

```
val tweets = ssc.twitterStream()
val hashTags = tweets.flatMap (status => getTags(status))
```

new DStream

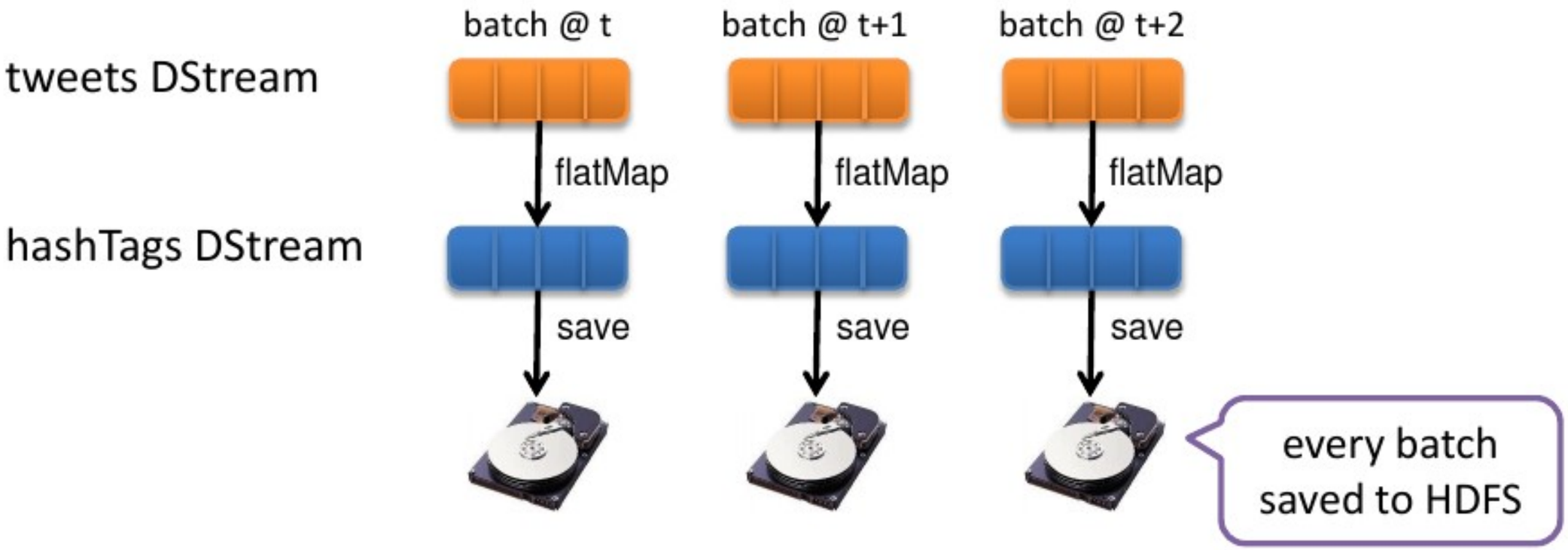
transformation: modify data in one DStream to create another DStream



- 1、不同于一般流处理软件，Spark Streaming采用一系列毫秒级的批量处理，实现快速计算。
- 2、将一个需要处理的任务，转化为多个RDD计算，运行在Spark上。

```
val tweets = ssc.twitterStream()
val hashTags = tweets.flatMap (status => getTags(status))
hashTags.saveAsHadoopFiles("hdfs://...")
```

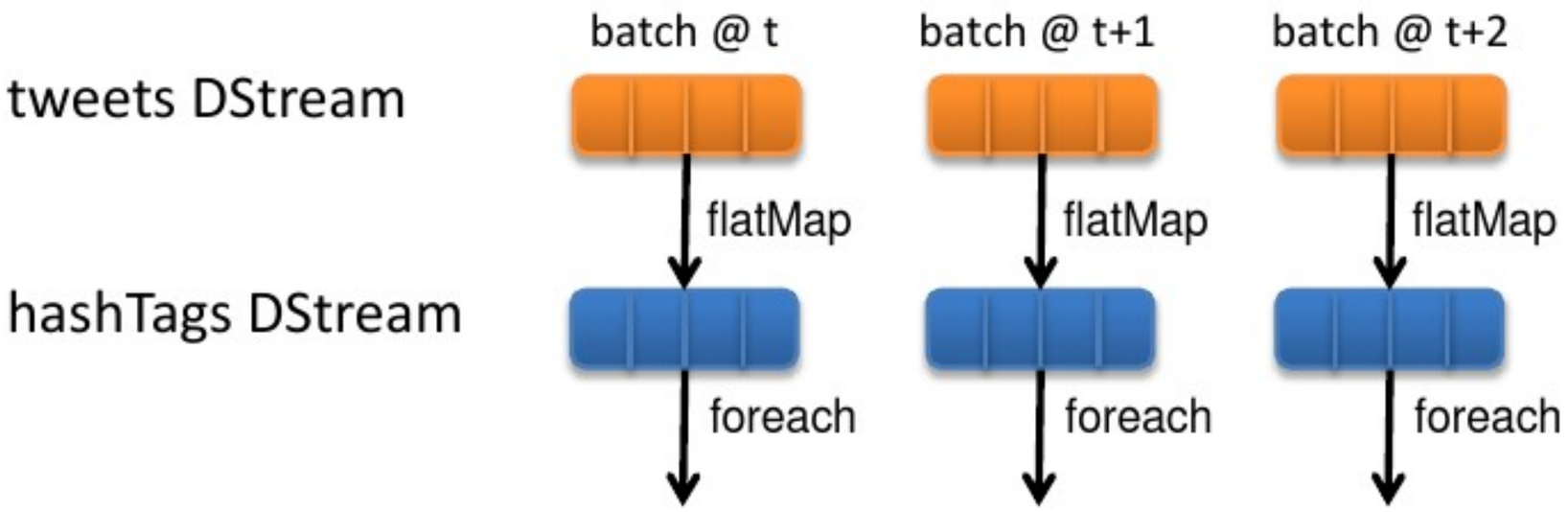
output operation: to push data to external storage



- 1、不同于一般流处理软件，Spark Streaming采用一系列毫秒级的批量处理，实现快速计算。
- 2、将一个需要处理的任务，转化为多个RDD计算，运行在Spark上。

```
val tweets = ssc.twitterStream()
val hashTags = tweets.flatMap (status => getTags(status))
hashTags.foreach(hashTagRDD => { ... })
```

foreach: do whatever you want with the processed data

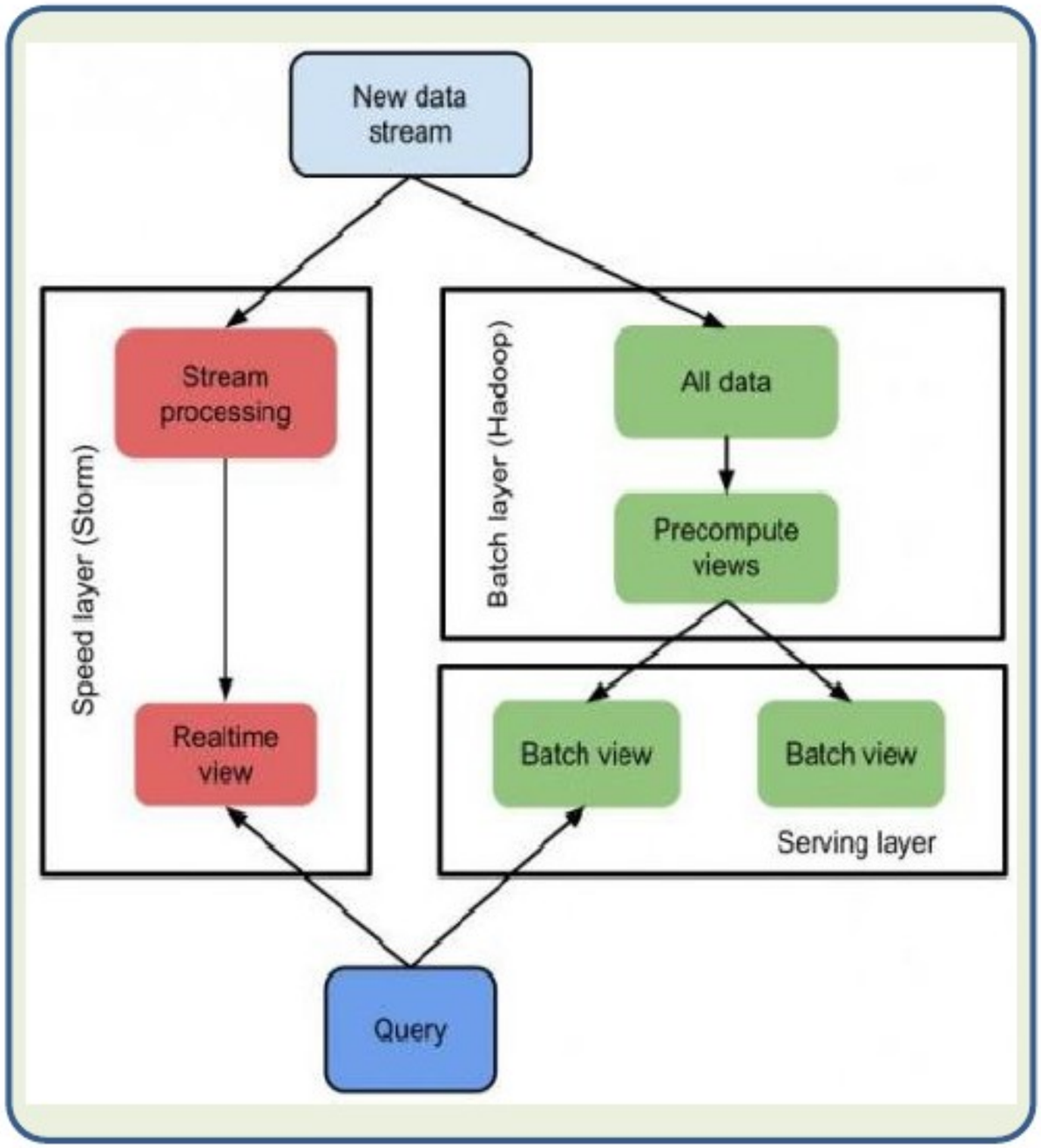


Write to database, update analytics UI, do whatever you want

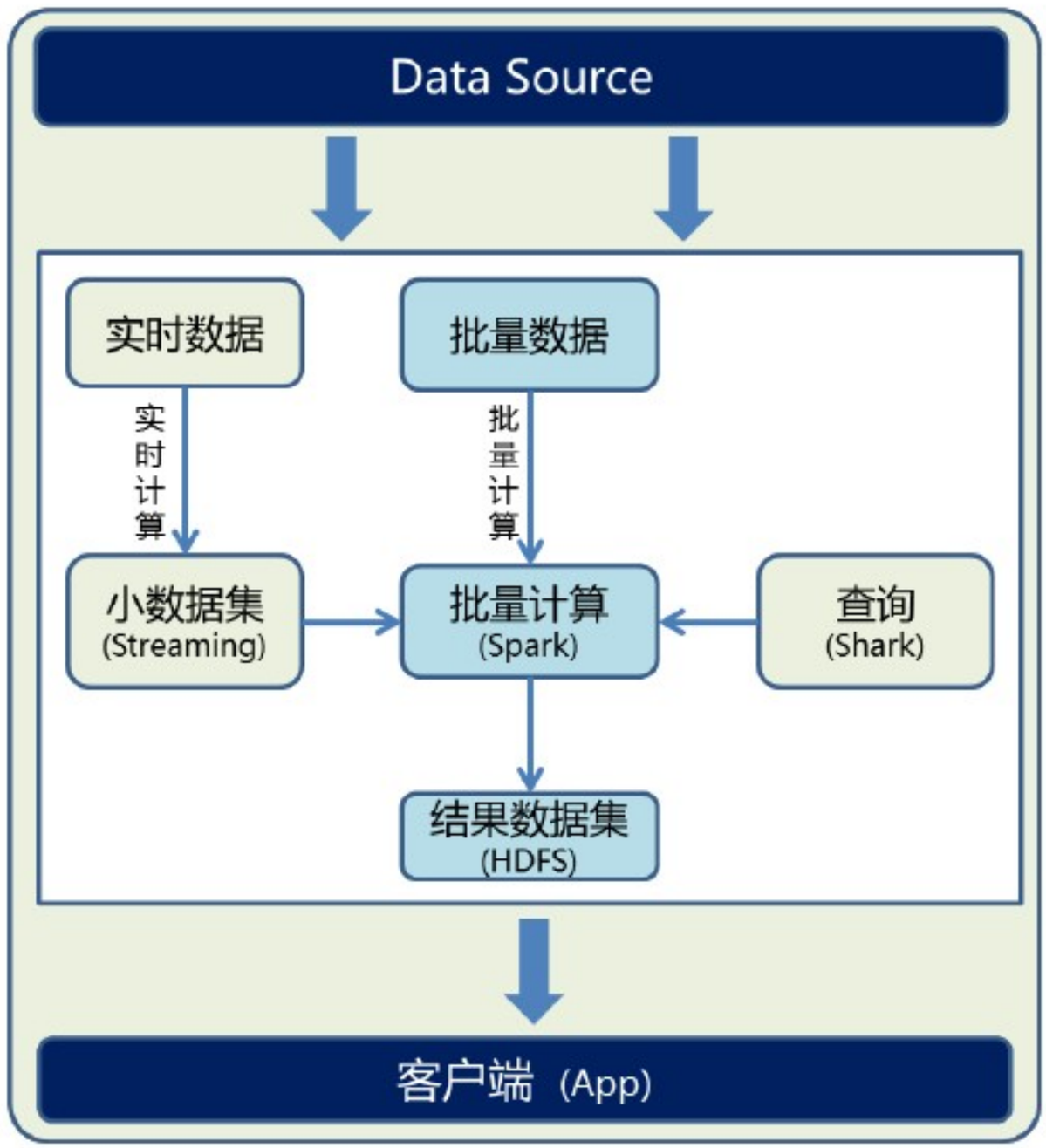
Spark知识分享

- Spark 简介
- Spark SQL 简介
- Spark Streaming 简介
- Spark应用场景

将Hadoop+Storm的架构，简化为Spark架构。实现一键式安装和配置，线程级别的任务监控和告警，降低硬件集群、软件维护、任务监控和应用开发的难度。后续要做成统一的硬件、计算平台资源池，发展到云计算。

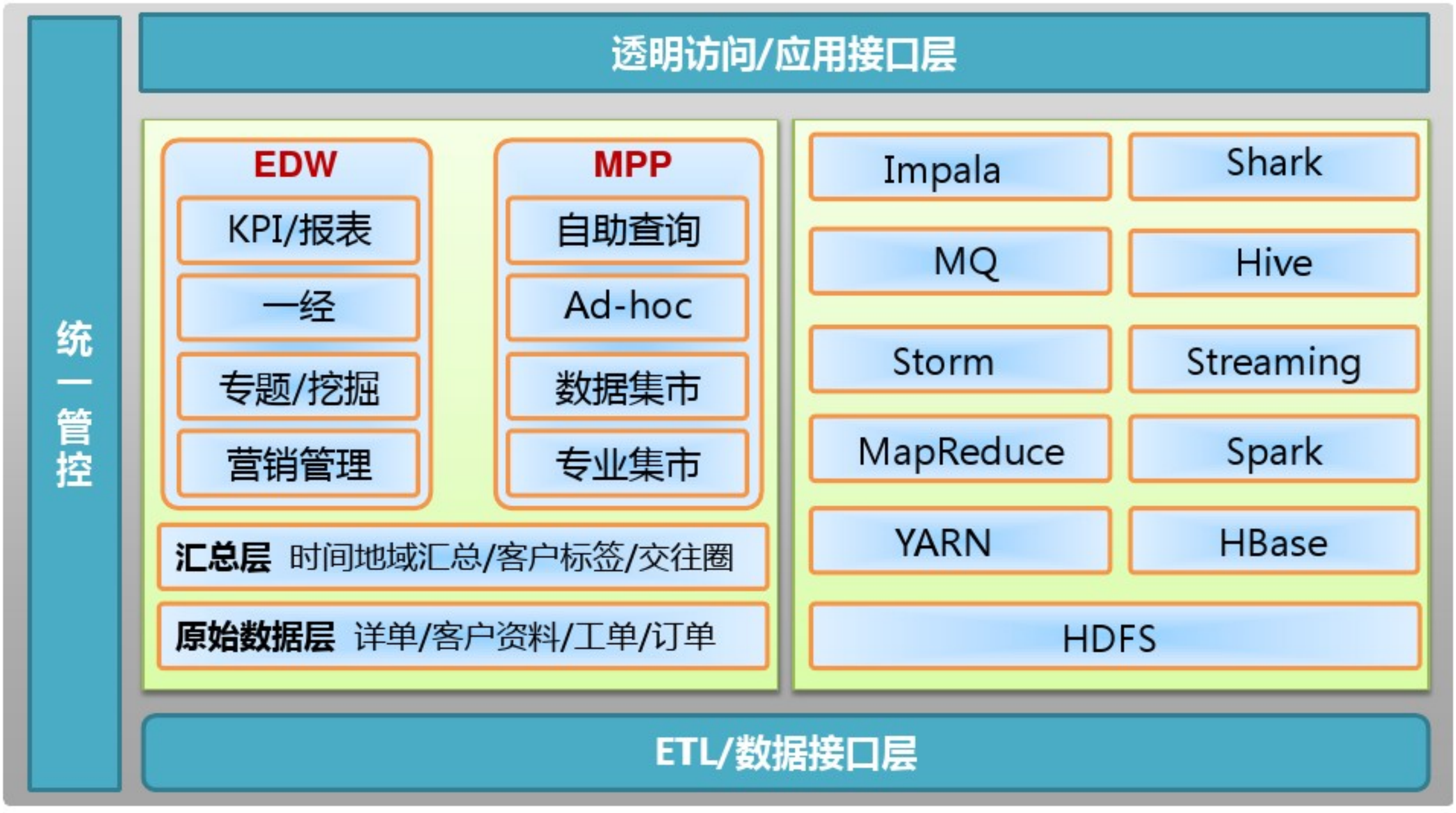


Lambda架构



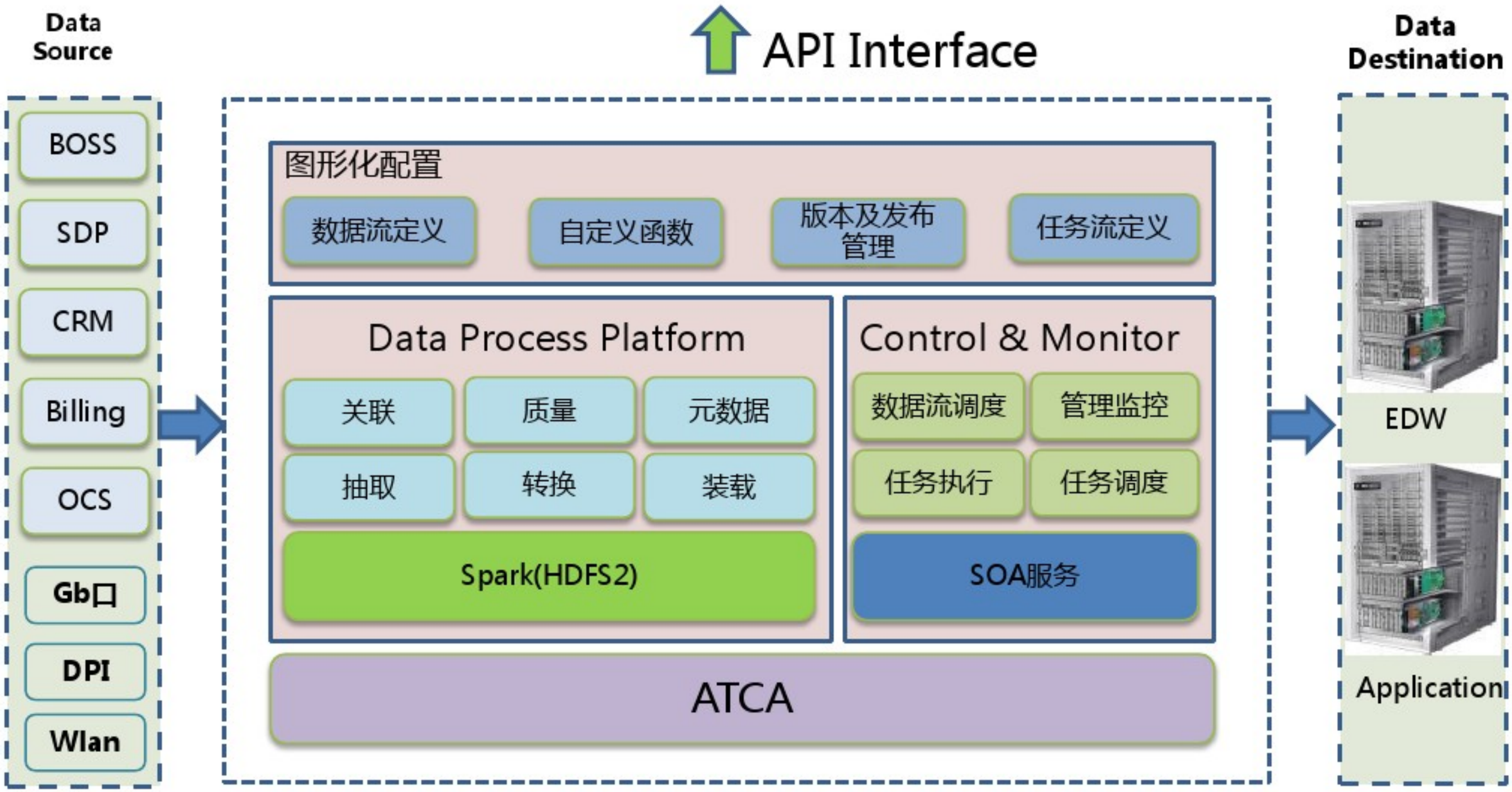
Spark架构

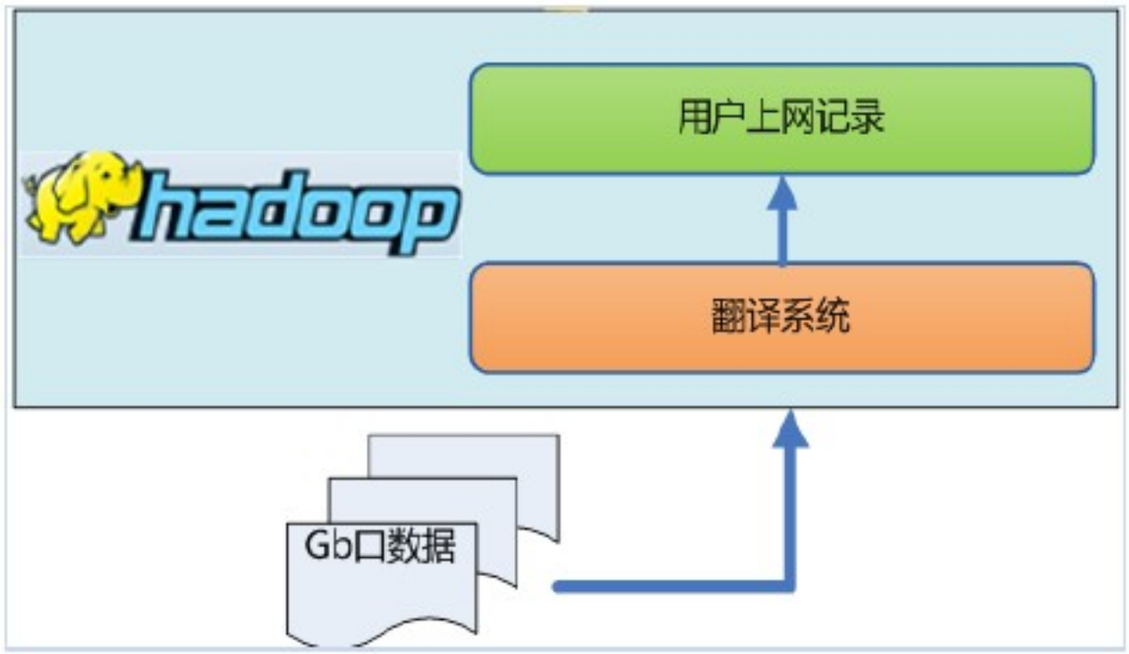
混搭架构是当前运营商的自然选择，根据数据的热度和存储成本来分布。通过三者的有效**融合**，以提供最大的计算能力。



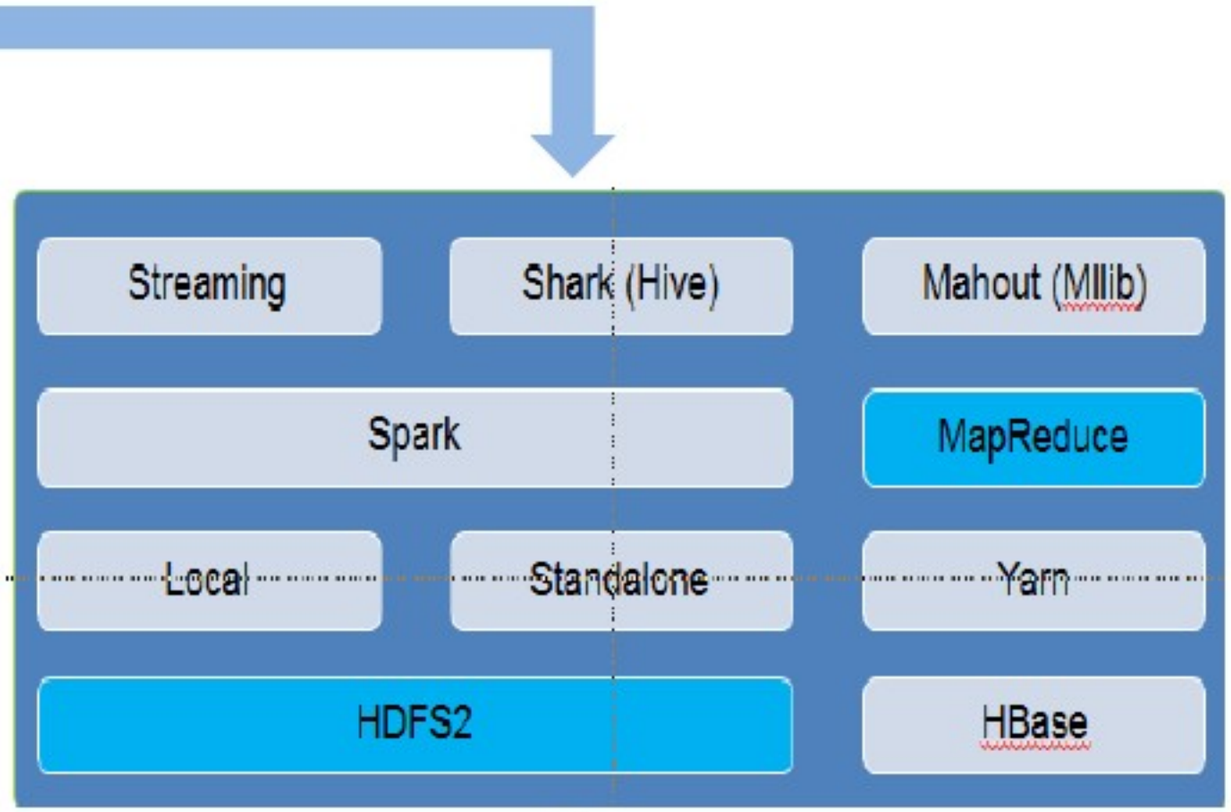
1、采用Tableau作为图形化配置和管理工具，将ETL过程、原子处理等转化为Spark的Task

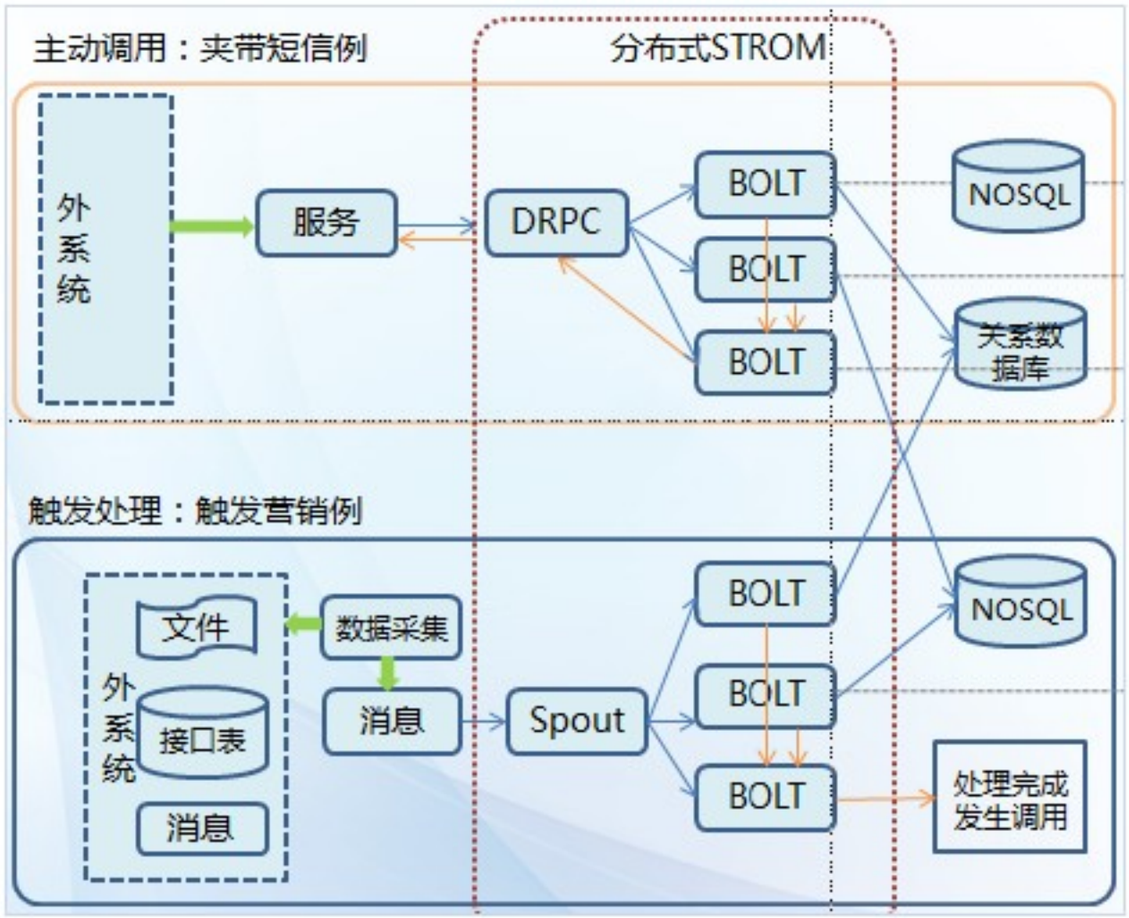
2、离线批量接口和实时接口采用同样的配置，只有处理的时间间隔属性不同





改造流量经营系统：用Spark替换MapReduce，迁移“翻译系统”的Java代码。目标：提升性能3-5倍。

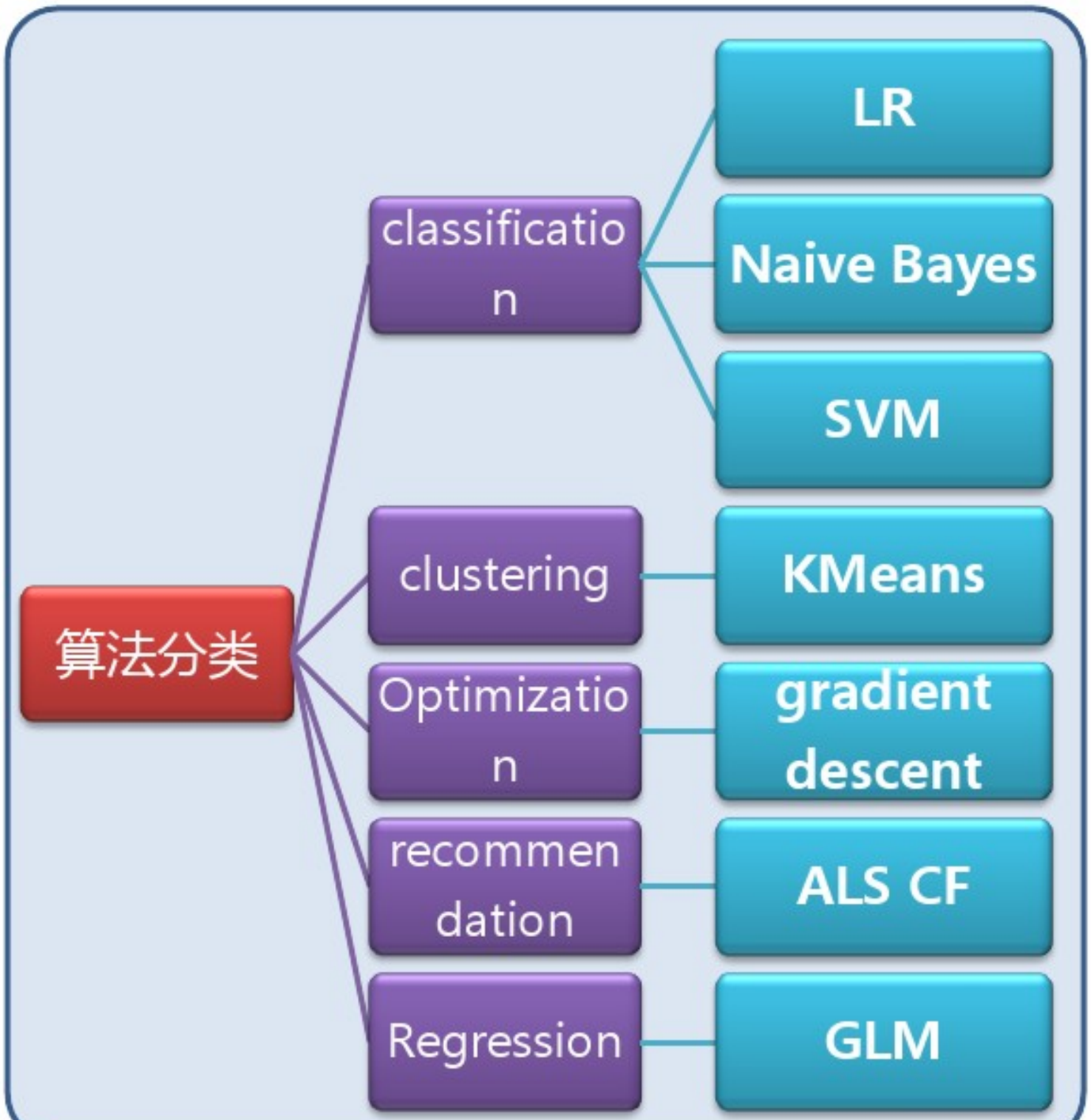




对比实时营销系统：用Streaming替换storm，采用Scala重新开发。目标：原系统对比业务满足能力和性能。

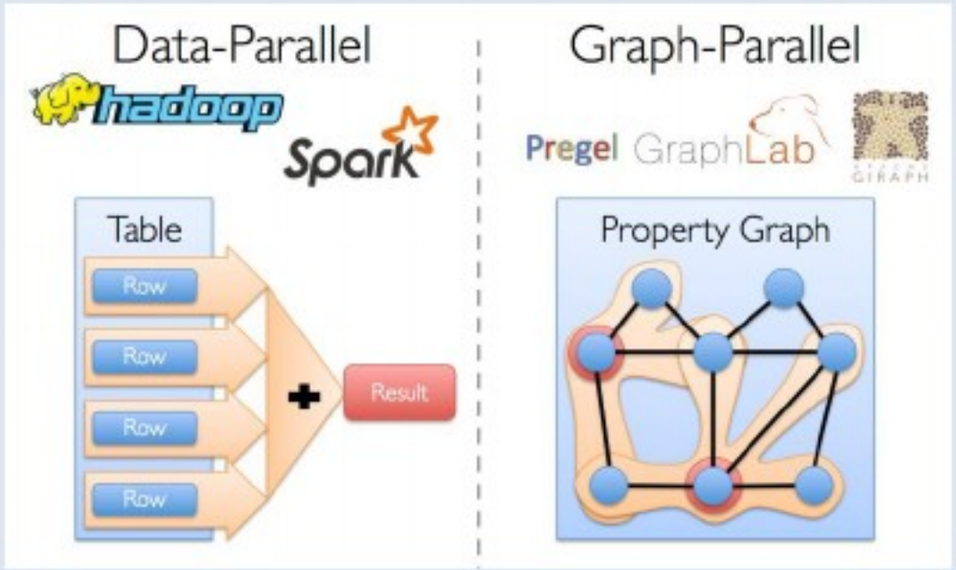
Spark MLlib

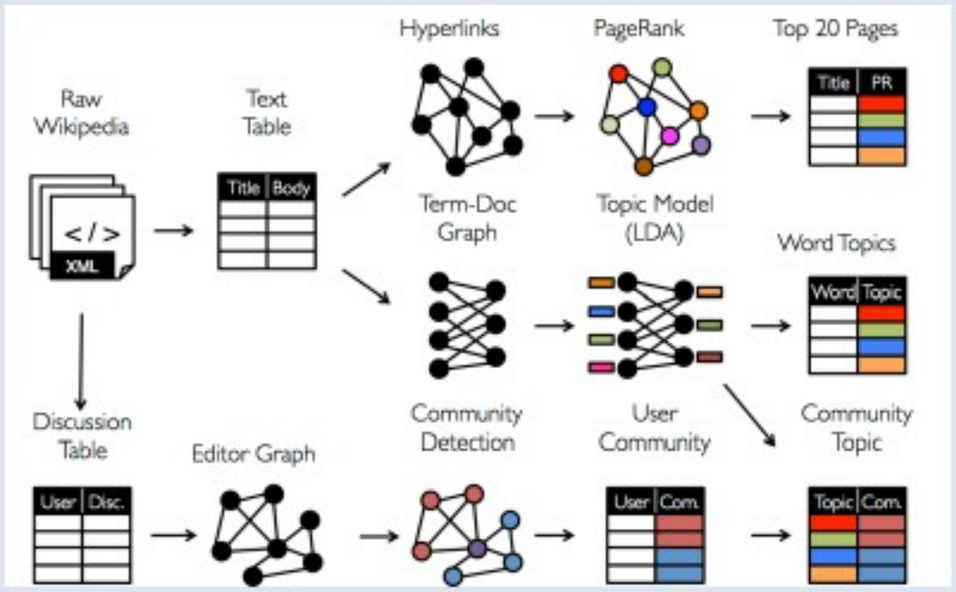
传统数据挖掘、机器学习应用



Spark GraphX

复杂社会、图分析算法





谢谢！