

Spark线下补充与实战

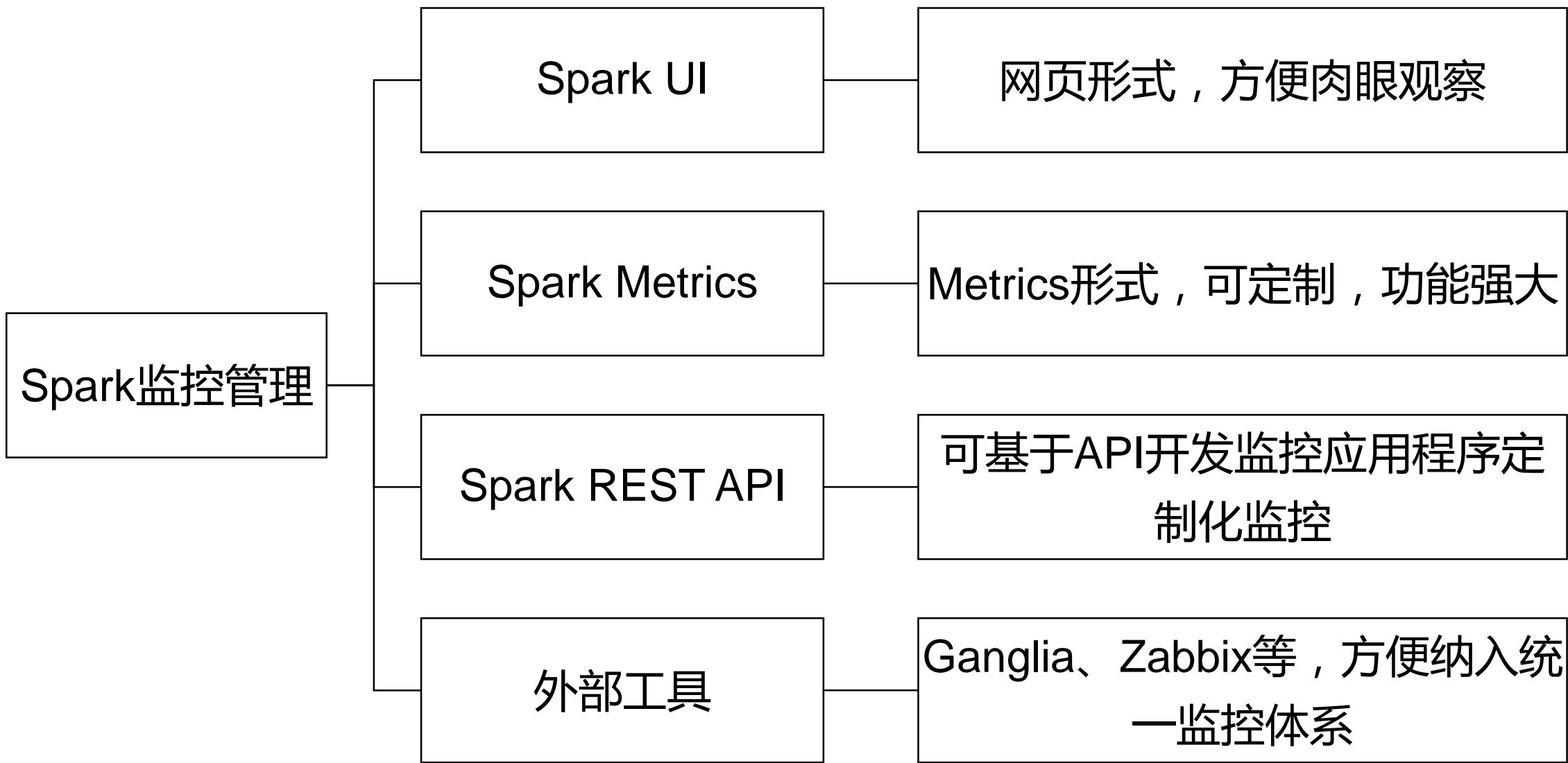
2019.05

相互学习，内部分享，请多多指教

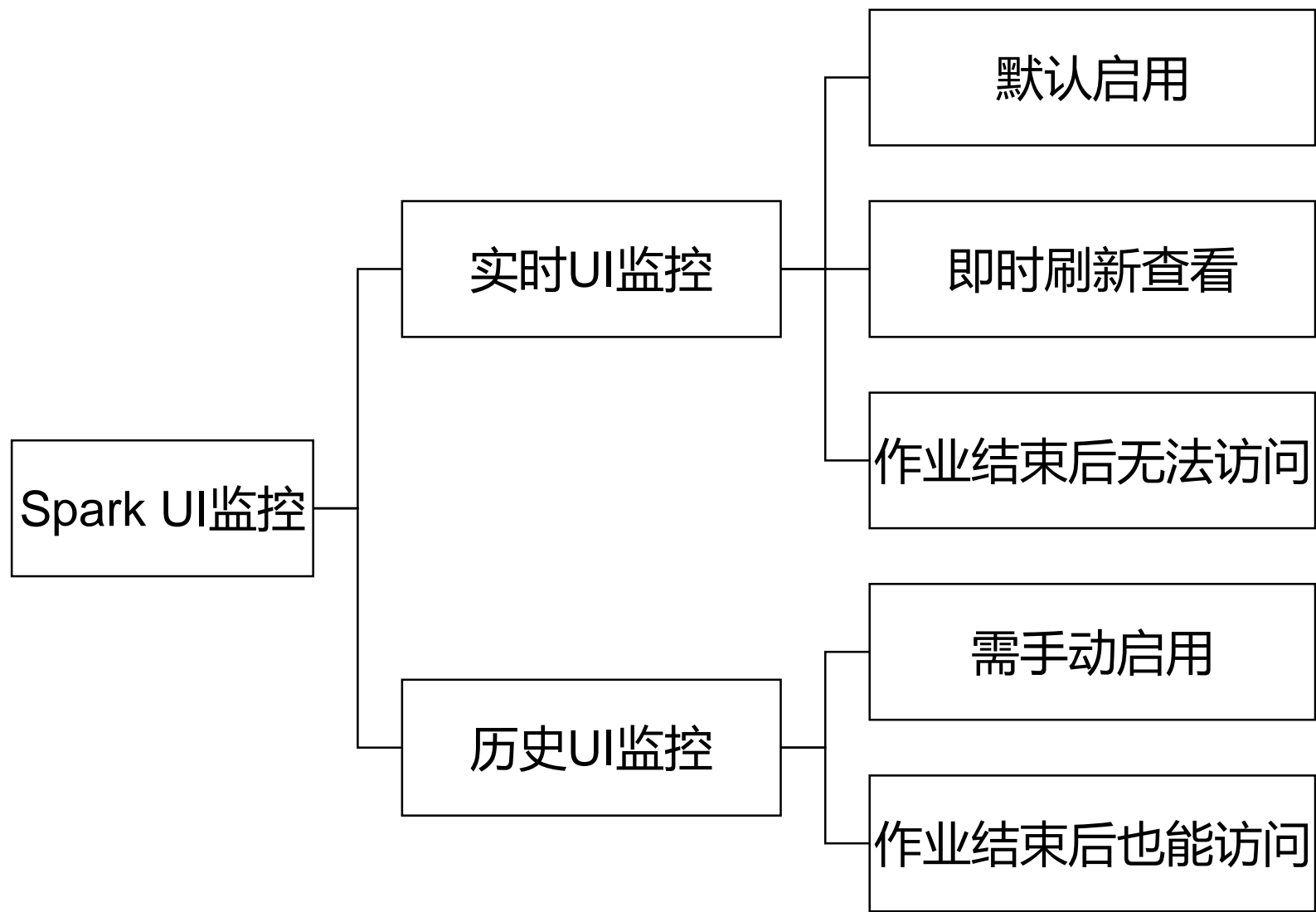
目录

- 知识点1 : Spark监控与UI简介
- 知识点2 : Spark Streaming简介
- 知识点3 : 几个好用的API速查表
- 知识点4 : 集群使用心得与实战

Spark监控管理




Spark UI监控



Spark UI 入口

- 每个 SparkContext 运行时，都会运行一个 Web UI，默认 4040 端口。
- 入口
 - <http://localhost:4040> (如果该端口被占用，则4041/4042依次后推)
 - 从yarn界面进入
- 仅当应用在运行中的时候可以访问，一旦应用执行结束该端口关闭不可访问

 Jobs Stages Storage Environment Executors Spark Pi application UI

Spark Jobs (?)

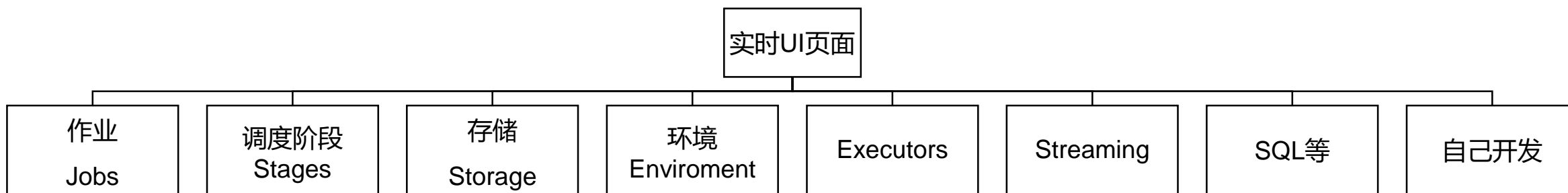
User: Adam
Total Uptime: 52 s
Scheduling Mode: FIFO
Completed Jobs: 1

▶ Event Timeline

▼ Completed Jobs (1)

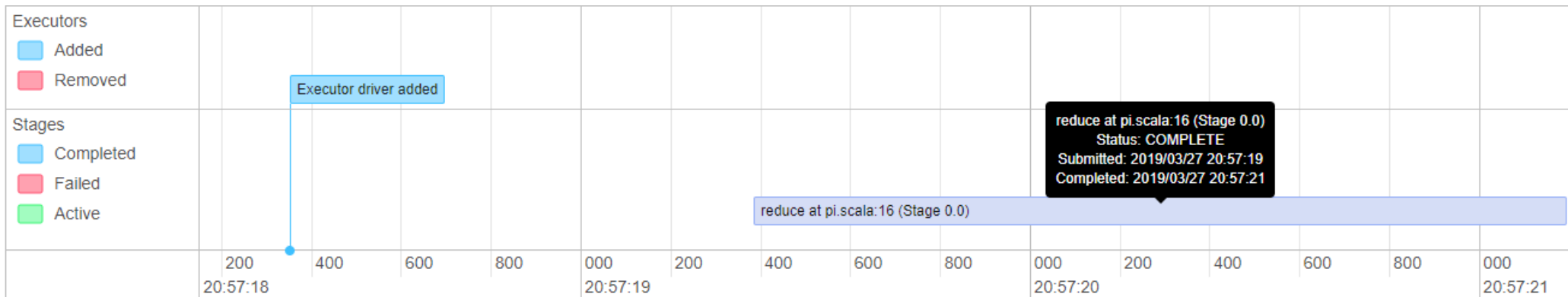
Job Id ▼	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	reduce at SparkPi.scala:38 reduce at SparkPi.scala:38	2019/03/26 20:13:13	1 s	1/1	2/2

实时UI页面




作业Jobs页面

- **Jobs** 展示的是整个spark应用任务的job整体信息
 - 作业概要信息
 - **User**: spark任务提交的用户。
 - **Total Uptime**: 总的运行时间。
 - **Scheduling Mode**: application中task任务的调度策略，由参数spark.scheduler.mode来设置，可选的参数有FAIR和FIFO，默认是FIFO。
 - **Completed Jobs**: 已完成Job的基本信息。
 - **Active Jobs**: 正在运行的Job的基本信息。
 - **Event Timeline**: 在application应用运行期间，图形化展现Job和Executor发生的事件。这个就是用来表示调度job何时启动何时结束，以及Executor何时加入何时移除。我们可以很方便看到哪些job已经运行完成，使用了多少Executor，哪些正在运行。



Job详情页样例

- 点击job列表页上Job的链接即可到达Job详情页

2.4.0

Jobs

Stages

Storage

Environment

Executors

Spark Pi application UI

Details for Job 0

Status: SUCCEEDED

Completed Stages: 1

▶ Event Timeline

▶ DAG Visualization

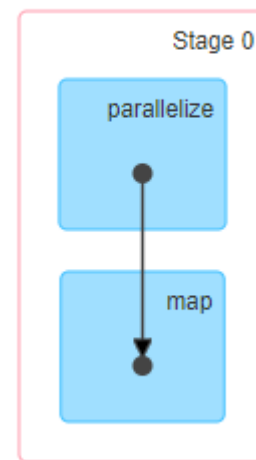
▼ **Completed Stages (1)**

Stage Id ▼	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
0	reduce at pi.scala:16 +details	2019/03/28 10:26:35	0.9 s	24/24				

Job详情页：Jobs Detail


- **Staus:** 展示Job的当前状态信息。
- **Active Stages:** 正在运行的stages信息，点击某个stage可进入查看具体的stage信息。
- **Pending Stages:** 排队的stages信息，根据解析的DAG图stage可并发提交运行，而有依赖的stage未运行完时则处于等待队列中。
- **Completed Stages:** 已经完成的stages信息。
- **Event Timeline:** 展示当前Job运行期间stage的提交与结束、Executor的加入与退出等事件信息。
- **DAG Visualization:** 当前Job所包含的所有stage信息（stage中包含的明细的tranformation操作），以及各stage间的DAG依赖图。DAG也是一种调度模型，在spark的作业调度中，有很多作业存在依赖关系，所以没有依赖关系的作业可以并行执行，有依赖的作业不能并行执行。

▼ DAG Visualization



Stages列表页

- 直接点击Stages就可到达Stages页面，展示所有Job的所有stage

2.4.0

Jobs

Stages

Storage

Environment

Executors

Spark Pi application UI

Stages for All Jobs

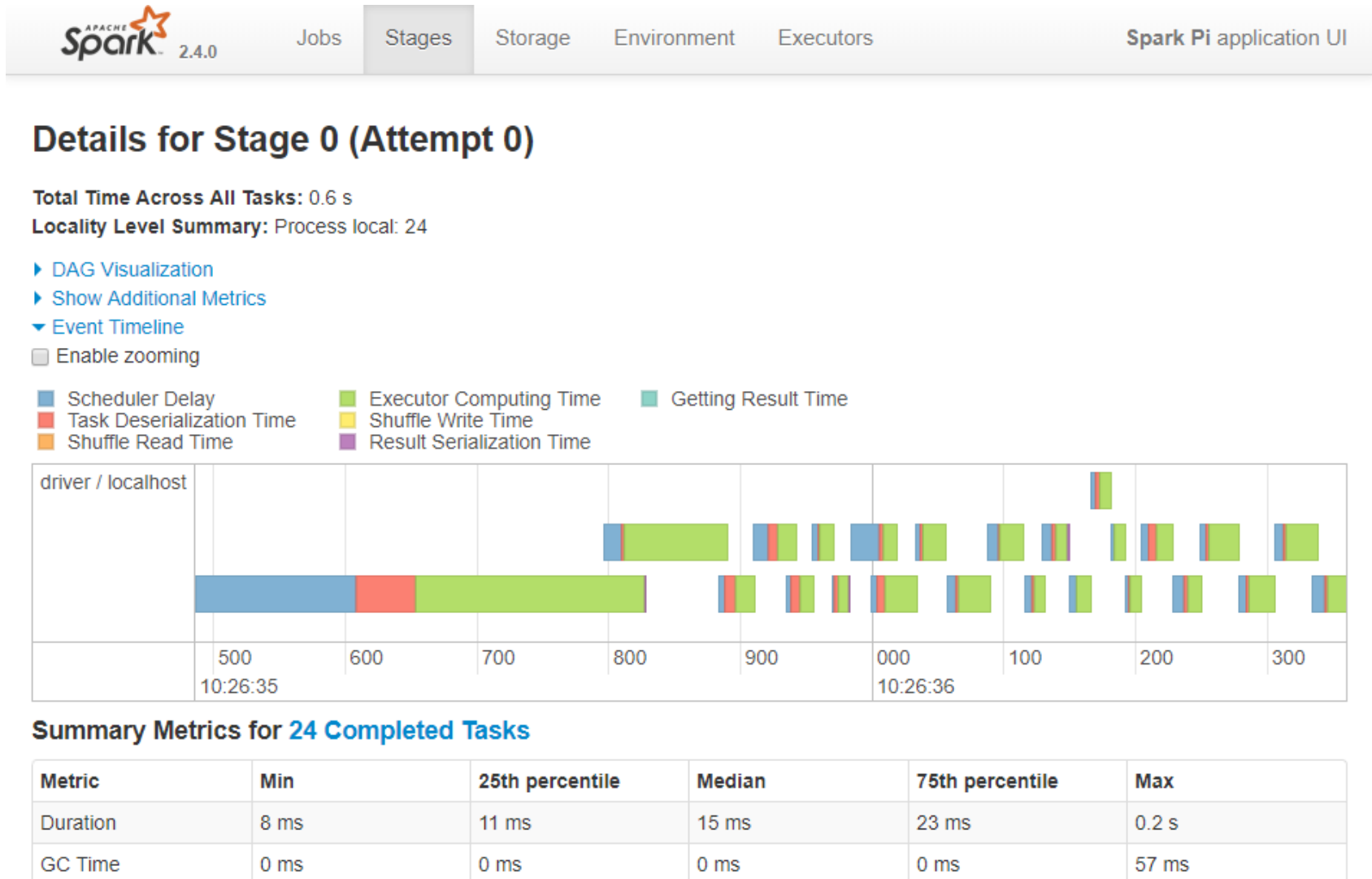
Completed Stages: 1

▼ Completed Stages (1)

Stage Id ▼	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
0	reduce at pi.scala:16 +details	2019/03/28 10:26:35	0.9 s	24/24				

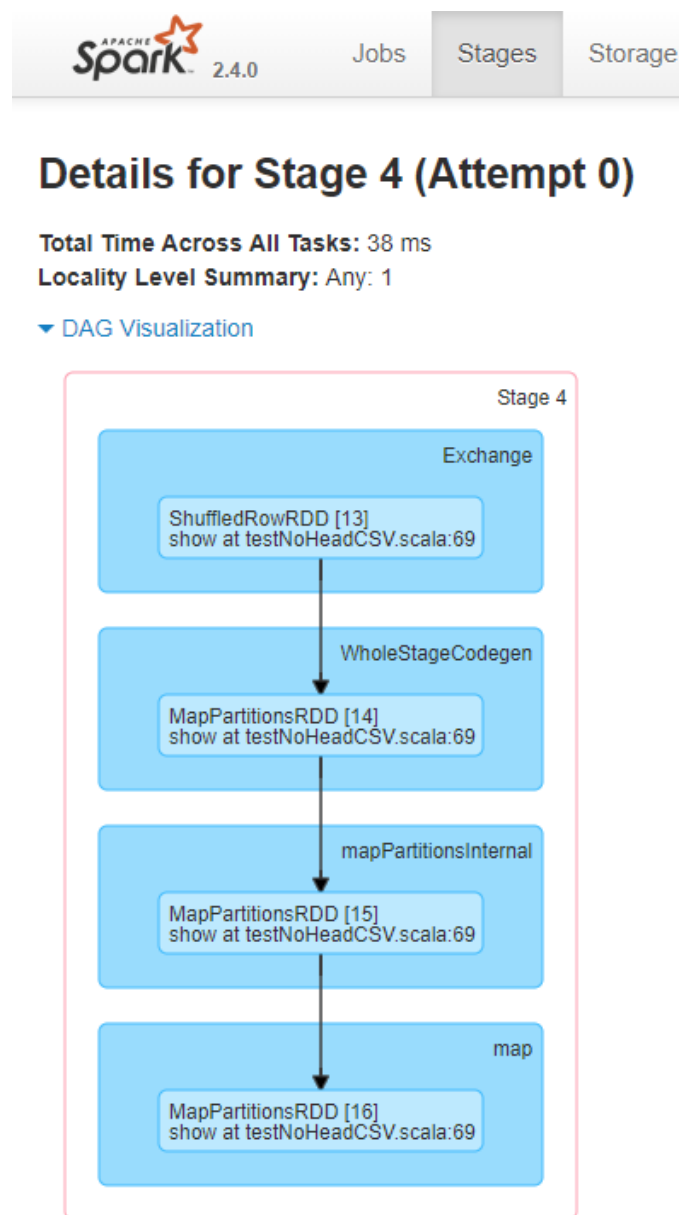
Stages详情页：样例

- 入口：
 - Stages列表页点击具体的Stage进入
 - 从Job详情页点击具体Stage进入
 - 从Job详情页点击DAG图进入
- 用于查看该调度阶段的运行细节



Stages详情页1:调度阶段概要信息

- **Total time across all tasks:** 当前stage中所有task花费的时间和。
- **Locality Level Summary:** 不同本地化级别下的任务数，本地化级别是指数据与计算间的关系
 - **PROCESS_LOCAL**进程本地化：task与计算的数据在同一个Executor中。
 - **NODE_LOCAL**节点本地化：
 - 情况一：task要计算的数据是在同一个Worker的不同Executor进程中；
 - 情况二：task要计算的数据是在同一个Worker的磁盘上，或在HDFS上，恰好有block在同一个节点上。
 - **RACK_LOCAL**机架本地化，数据在同一机架的不同节点上：
 - 情况一：task计算的数据在Worker2的Executor中；
 - 情况二：task计算的数据在Worker2的磁盘上。
 - **ANY** 跨机架，数据在非同一机架的网络上，速度最慢。
- **DAG Visualization:** 当前Stage所包含的所有算子的DAG信息。显示具体的transform算子，精确到代码行；



Stages详情页2

- **Input Size/Records:** 输入的数据字节数大小/记录条数。
- **Shuffle Read/Write:** 为下一个依赖的stage提供输入数据，shuffle过程中通过网络传输的数据字节数/记录条数。
- **Metrics:** 当前stage中所有task的一些指标统计信息。
- **Event Timeline:** 清楚地展示在每个Executor上各个task的各个阶段的时间统计信息，可以清楚地看到task任务时间是否有明显倾斜，以及倾斜的时间主要是属于哪个阶段，从而有针对性的进行优化。
- **Aggregated Metrics by Executor:** 将task运行的指标信息按excutor做聚合后的统计信息，并可查看某个Excutor上任务运行的日志信息。可以发现由executor(机器)原因导致的错误；
- **Tasks:** 当前stage中所有任务运行的明细信息，是与Event Timeline中的信息对应的文字展示（可以点击某个task查看具体的任务日志）。

存储监控页面 Storage

- storage页面能看出application当前使用的缓存情况，可以看到有哪些RDD被缓存了，以及占用的内存资源。
- 如果job在执行时持久化（persist）/缓存（cache）了一个RDD，那么RDD的信息可以在这个选项卡中查看。
- 包括缓存级别、partition数、缓存百分比、磁盘/内存占用情况等

<div><div>2.4.0</div><div>Jobs</div><div>Stages</div><div>Storage</div><div>Environment</div><div>Executors</div><div>952d8fc0-ac37-40d0-8c86-3b118b85... application UI</div></div>						
Storage						
▼ RDDs						
ID	RDD Name	Storage Level	Cached Partitions	Fraction Cached	Size in Memory	Size on Disk
4	ShuffledRDD	Memory Deserialized 1x Replicated	1	100%	384.0 B	0.0 B

存储监控页面 Storage Detail

- 点击某个RDD即可查看该RDD缓存的详细信息：
 - 包括存储级别、缓存partition数，内存占用情况、磁盘占用情况、在哪个Executor中，使用的block情况，RDDpartitions的信息以及存储RDD的主机的地址等。



RDD Storage Info for ShuffledRDD

Storage Level: Memory Deserialized 1x Replicated
Cached Partitions: 1
Total Partitions: 1
Memory Size: 384.0 B
Disk Size: 0.0 B

Data Distribution on 1 Executors

Host	On Heap Memory Usage	Off Heap Memory Usage	Disk Usage
windows10.microdone.cn:61431	384.0 B (897.6 MB Remaining)	0.0 B (0.0 B Remaining)	0.0 B


1 Partitions

Block Name ^	Storage Level	Size in Memory	Size on Disk	Executors
rdd_4_0	Memory Deserialized 1x Replicated	384.0 B	0.0 B	windows10.microdone.cn:61431

环境信息页面 Enviroment

- Environment选项卡提供有关Spark应用程序（或SparkContext）中使用的各种属性和环境变量的信息。
- 用户可以通过这个选项卡得到非常有用的各种Spark属性信息，而不用去翻找属性配置文件。也可以确认Spark设置的参数是否正确。

- 包括
 - 运行时信息
 - Spark参数
 - 系统参数
 - Classpath信息

 2.4.0

[Jobs](#) [Stages](#) [Storage](#) **Environment** [Executors](#)

Environment

Runtime Information

Name	Value
Java Version	1.8.0_91 (Oracle Corporation)
Java Home	C:\Java\jdk1.8.0_91\jre
Scala Version	version 2.11.11


▶ Spark Properties

▶ System Properties

▶ Classpath Entries







Executors页面

- Summary: 该application运行过程中使用Executor的统计信息。
- Executors: 每个Excutor的详细信息（包含driver），
 - 一些基本信息，如：编号、地址、状态等；
 - 运行期间的状态信息，关于内存、CPU、磁盘、Task，Shuffle等等；
 - 可以点击查看某个Executor中任务运行的详细日志。

2.4.0

[Jobs](#)[Stages](#)[Storage](#)[Environment](#)[Executors](#)

Spark Pi applicati

中

Executors

Summary

	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Blacklisted
Active(1)	0	0.0 B / 941.2 MB	0.0 B	1	0	0	24	24	1.0 s (61 ms)	0.0 B	0.0 B	0.0 B	0
Dead(0)	0	0.0 B / 0.0 B	0.0 B	0	0	0	0	0	0 ms (0 ms)	0.0 B	0.0 B	0.0 B	0
Total(1)	0	0.0 B / 941.2 MB	0.0 B	1	0	0	24	24	1.0 s (61 ms)	0.0 B	0.0 B	0.0 B	0

Executors

Show entries

Search:

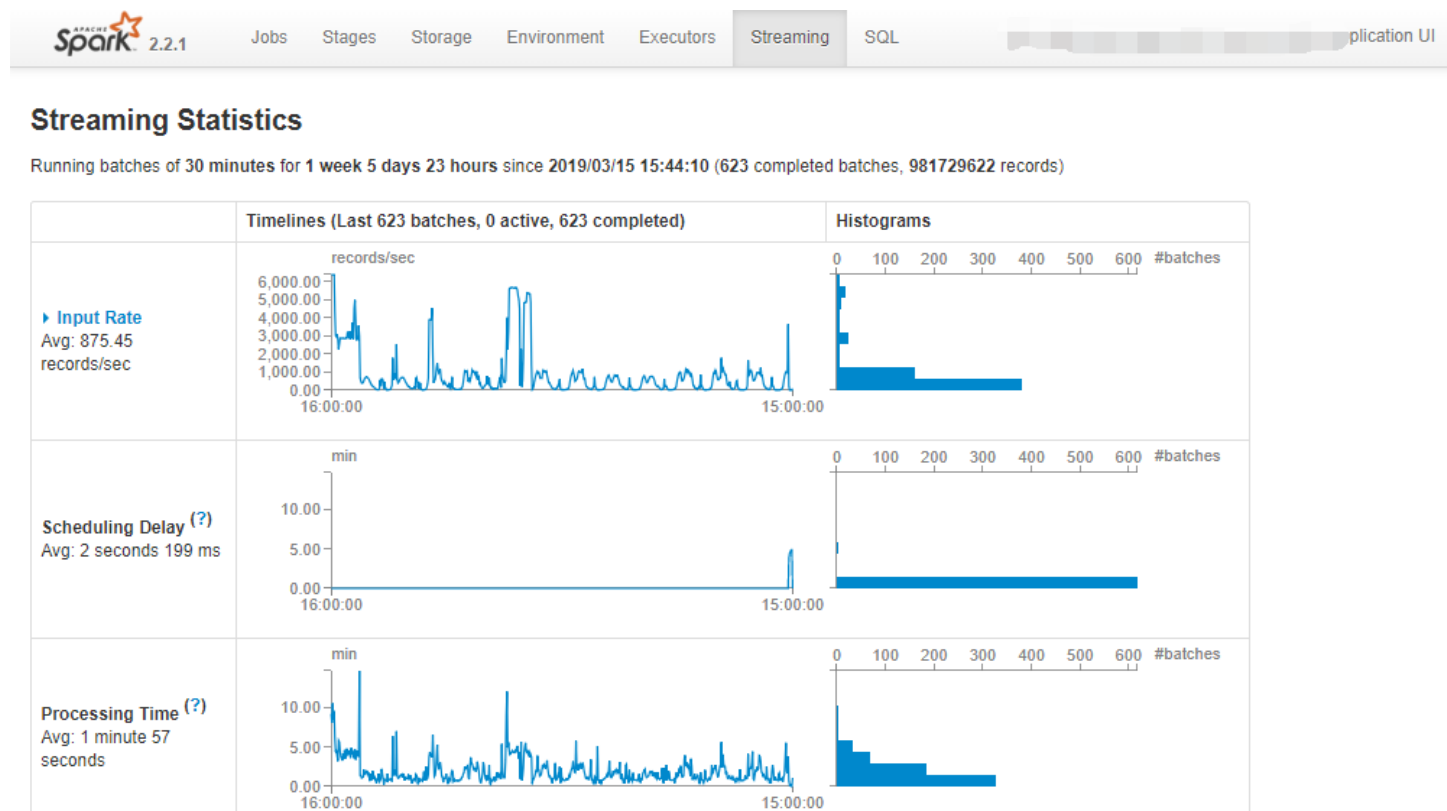
Executor ID	Address	Status	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Thread Dump
driver	windows10.microdone.cn:61650	Active	0	0.0 B / 941.2 MB	0.0 B	1	0	0	24	24	1.0 s (61 ms)	0.0 B	0.0 B	0.0 B	Thread Dump

Showing 1 to 1 of 1 entries

[Previous](#) [1](#) [Next](#)

SQL、Streaming等

- SQL页面（只有执行了spark SQL查询才会有SQL选项卡）
 - 可以查看SQL执行计划的细节。
- Streaming监控页面（只有执行了流处理程序才会有Streaming选项卡）
 - 显示了Spark Streaming处理情况



历史UI监控

- 需配置Spark history server服务

- 配置 spark-env.sh文件

```
SPARK_HISTORY_OPTS="-Dspark.history.ui.port=8778 -Dspark.history.retainedApplications=10  
-Dspark.history.fs.logDirectory=hdfs:///user/root/op/history2"
```

- 到spark跟目录下，执行 ./sbin/start-history-server.sh

- 作业提交需配置以下选项

- spark.eventLog.enabled true 是否记录Spark事件
 - spark.eventLog.dir hdfs:///user/root/op/history 记录spark事件的根目录
 - spark.eventLog.compress false 是否压缩记录Spark事件，默认是snappy
 - spark.yarn.historyServer.address master:18080

Spark History Server

- 实战二：集群中部署Spark History Server
 - Spark客户端的下载与配置
 - 部署Spark History Server

←

→

↺

Not secure | namenode:18080

APACHE

Spark

2.4.0

History Server

Event log directory: hdfs://namenode:9000/spark-logs

Last updated: 2019-01-10 19:14:42

Client local time zone: America/Los_Angeles

App ID	App Name	Started	Completed
application_1547102810368_0003	Spark Pi	2019-01-10 19:00:41	2019-01-10 19:01:18

Spark REST API

- 方便用户对任务做监控
- Spark的REST API返回的信息是JSON格式，可通过这个API来创建可视化的Spark监控工具
- 支持正在运行的应用程序，也支持历史服务器。
在请求URL都有/api/v1
 - 正在运行的程序所在master地址：
 - <http://localhost:4040/api/v1/applications>
 - <http://localhost:4040/api/v1/applications/local-1553770789647/jobs>
 - 正在yarn上运行的作业：
 - http://yarnaddress/proxy/application_1546863598863_9813812/api/v1/applications
 - 历史作业可以在history Server获取：
 - <http://:18080/api/v1/applications>



The screenshot shows a web browser window with the address bar displaying `localhost:4040/api/v1/applications/local-1553770789647/jobs`. The browser's toolbar includes navigation buttons and search engines like Bing, Yandex, and Baidu. The main content area displays a JSON response for a specific job, enclosed in a dashed box. The JSON object contains the following fields: `jobId` (0), `name` ("reduce at pi.scala:16"), `submissionTime` ("2019-03-28T10:59:51.319GMT"), `completionTime` ("2019-03-28T10:59:52.904GMT"), `stageIds` (an array containing 0), `status` ("SUCCEEDED"), `numTasks` (24), `numActiveTasks` (0), `numCompletedTasks` (24), `numSkippedTasks` (0), `numFailedTasks` (0), `numKilledTasks` (0), `numCompletedIndices` (24), `numActiveStages` (0), `numCompletedStages` (1), `numSkippedStages` (0), `numFailedStages` (0), and `killedTasksSummary` (an empty object).

```
[
  {
    jobId: 0,
    name: "reduce at pi.scala:16",
    submissionTime: "2019-03-28T10:59:51.319GMT",
    completionTime: "2019-03-28T10:59:52.904GMT",
    stageIds: [
      0
    ],
    status: "SUCCEEDED",
    numTasks: 24,
    numActiveTasks: 0,
    numCompletedTasks: 24,
    numSkippedTasks: 0,
    numFailedTasks: 0,
    numKilledTasks: 0,
    numCompletedIndices: 24,
    numActiveStages: 0,
    numCompletedStages: 1,
    numSkippedStages: 0,
    numFailedStages: 0,
    killedTasksSummary: {}
  }
]
```

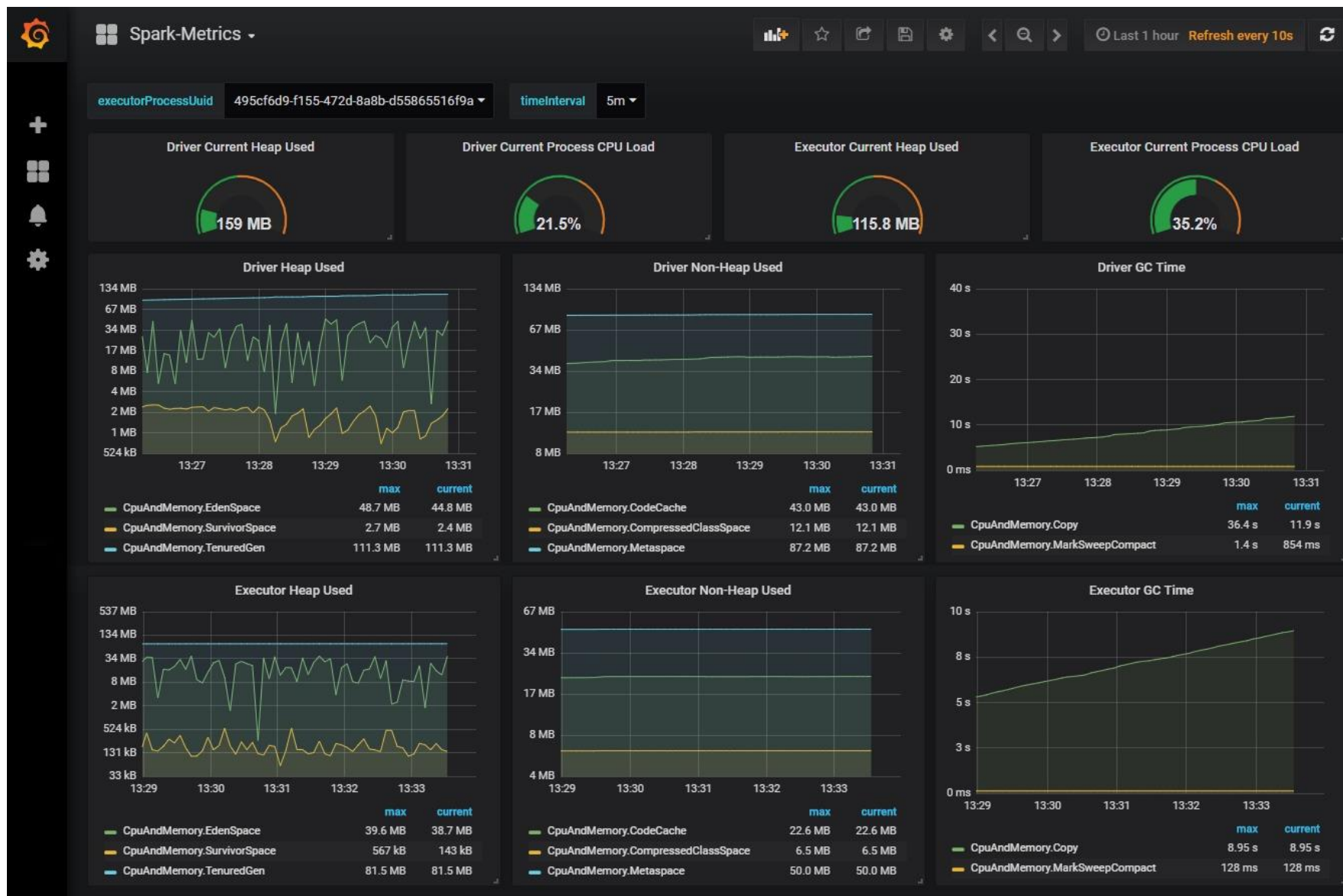
Spark Metrics系统

- Spark内置了可配置的度量系统Metrics，基于Coda Hale Metrics Library库构建。
 - 将Spark内部状态通过 Http、JMX、CSV等形式呈现给用户；
 - 用户可以以插件的方式将自己实现的Source 和Sink添加到Metrics系统中；
 - 相关配置文件：“\$SPARK_HOME/conf/metrics.properties”
- **Metrics Source** 定义了所采集的Metrics以及如何采集
 - Spark组件的内部状态，eg：MasterSource、WorkerSource、ApplicationSource...等
 - 自定义Source，eg：JvmSource
- **Metrics Sink** metrics信息发送到哪，可以设置一个或多个Sink
 - **ConsoleSink** 记录Metrics信息到Console中。
 - **CSVSink** 定期的把Metrics信息导出到CSV文件中。
 - **JmxSink** 可以通过JMX方式访问Metrics信息
 - **GraphiteSink**
 - **GangliaSink** 由于Licene限制，默认没有放到默认的build里面。

基于外部工具的监控

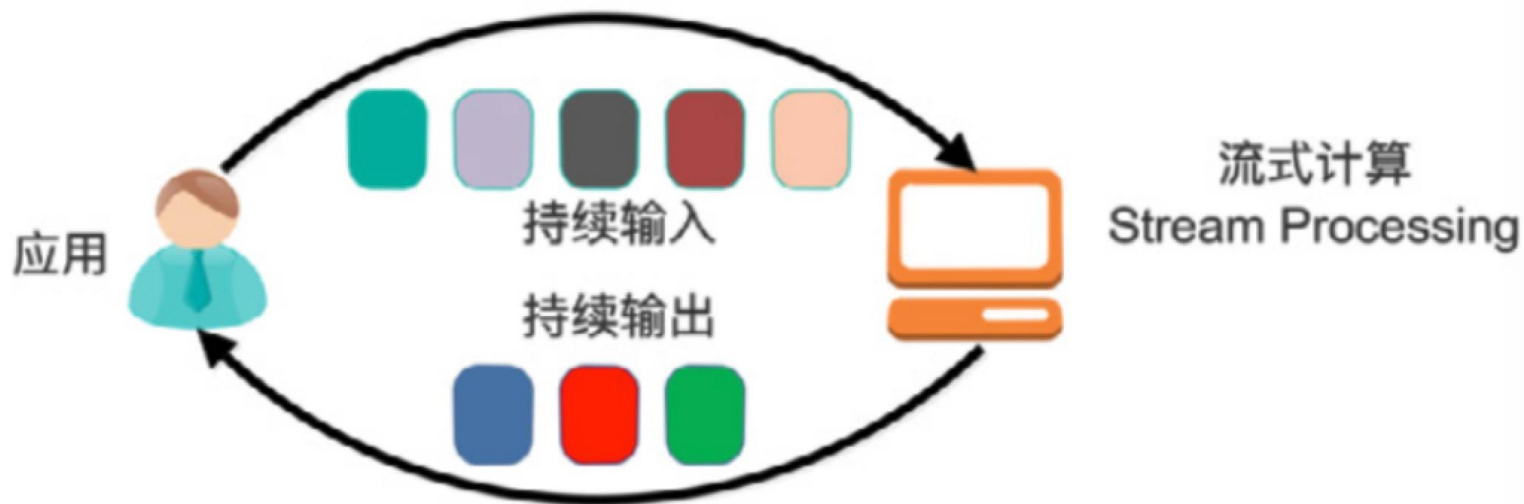
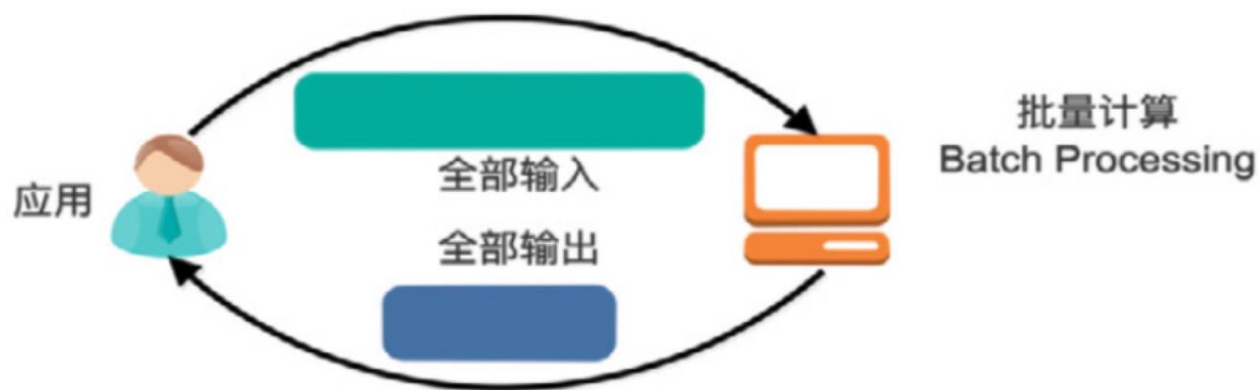
- eg Grafana

- Metrics收集
- 时序数据库存储
- 页面展示
- 告警功能



Spark Streaming 简介

- 为什么需要流计算



原生处理流与微批处理流

● 原生处理流

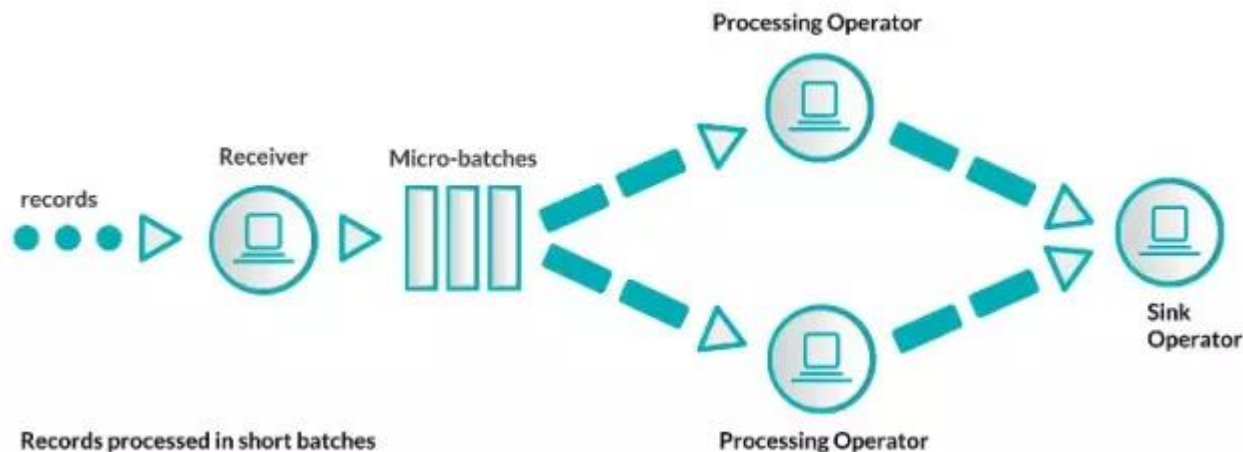
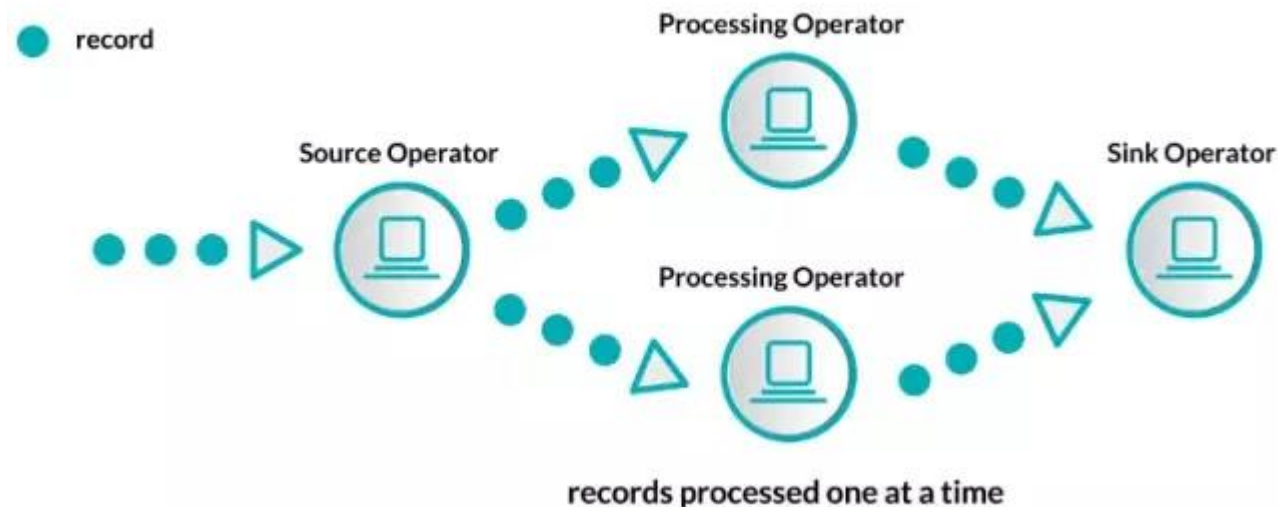
- 低延迟
- 状态管理简单
- 容错性成本较高

容错性是影响吞吐量的一个重要原因

● 微批处理流

- 容错性和负载均衡实现简单
- 状态管理实现困难

Native stream processing systems
continuous operator model



-
- DStream是sparkstreaming提供的基本抽象，它代表一个连续的数据流。
 - 它要么是从源获取的输入流，要么是输入流通过转换算子生成的处理后的数据流
 - 在内部，Dstreams由一系列连续的RDD组成。
 - Dstream中的每个RDD都包含确定时间间隔内的数据；
 - 每一个输入流Dstream和一个Reciver对象相关联，这个Reciver从源中获取数据，并将数据存入内存中用于处理；
 - Reciver作为一个长期运行的任务，需要占用一个核，所以在任务提交时需要分配足够的核；

DStream数据集相关操作

- **Input——从外部源源不断产生新数据的操作**

- socketTextStream fileInputStream kafkaInputStream flumeInputStream等

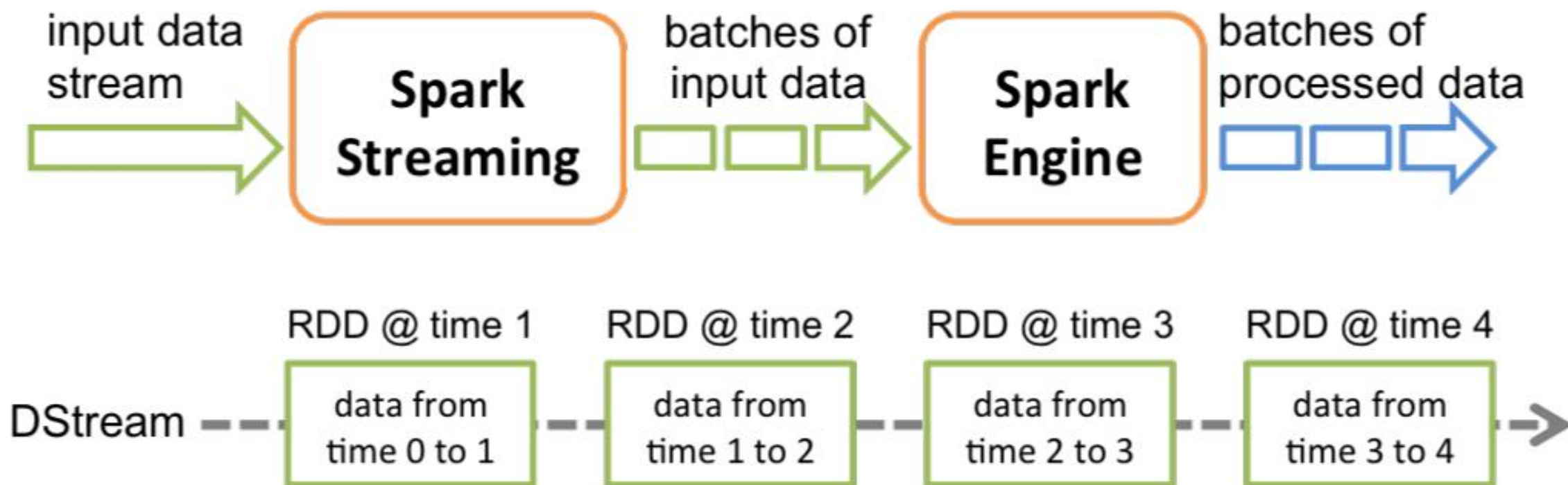
- **transformation——由已有的 DStream 产生新 DStream 的操作**

- map filter reduce join 等

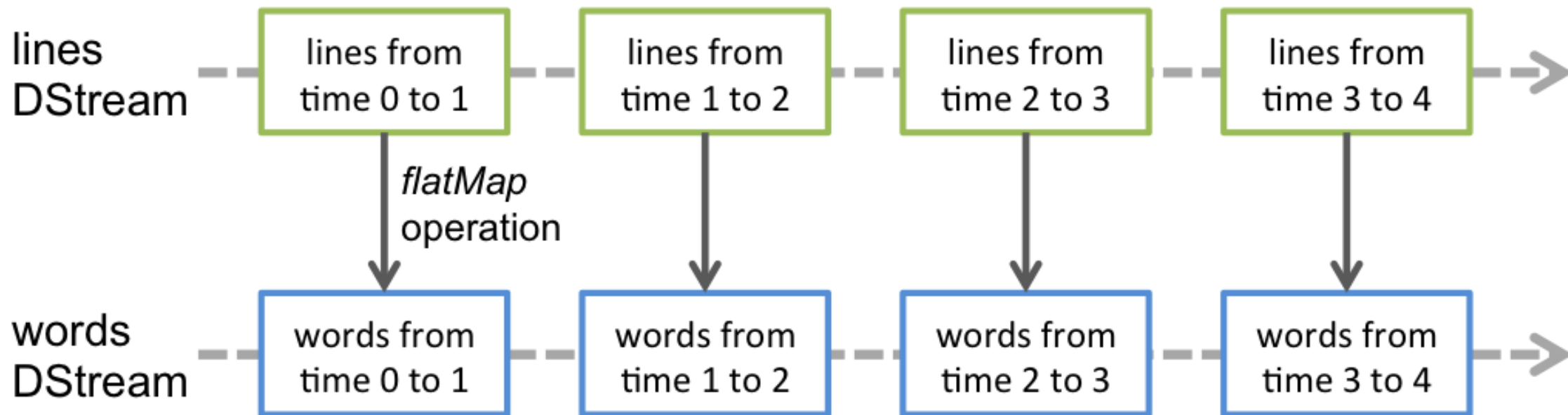
- **output ——在已有 DStream 数据集上进行的操作和输出**

- print saveAsTextFiles saveAsHadoopFiles foreachRDD 等

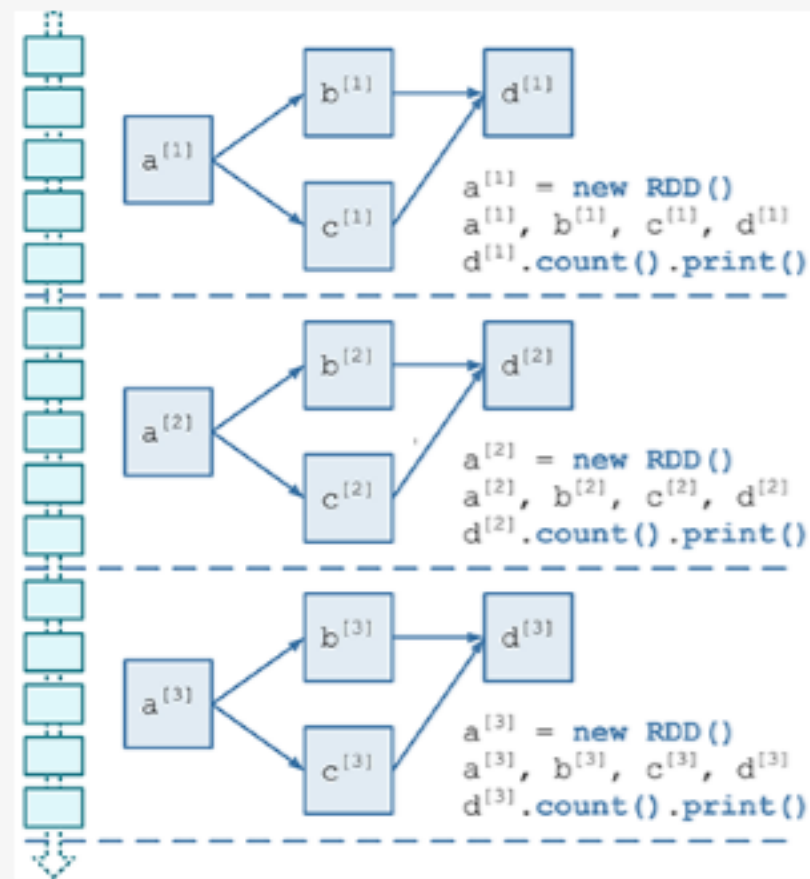
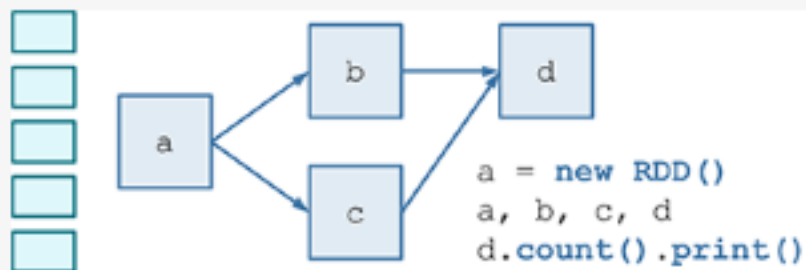
Spark Streaming运行原理



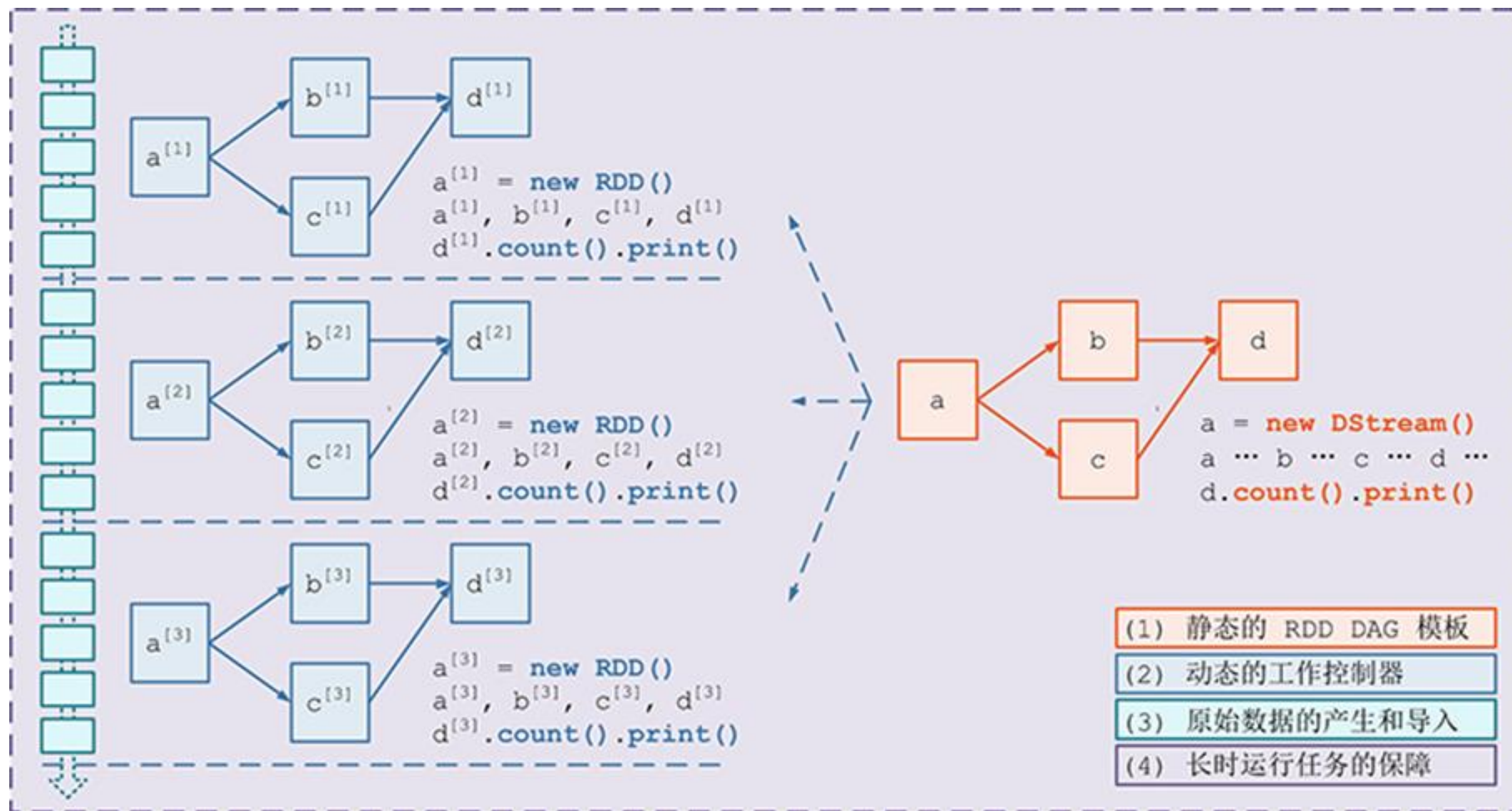
Spark Streaming运行原理



Spark Streaming运行原理



Spark Streaming运行原理



- <http://spark.apache.org/docs/latest/streaming-programming-guide.html#transformations-on-dstreams>

map (<i>func</i>)	Return a new DStream by passing each element of the source DStream through a function <i>func</i> .
flatMap (<i>func</i>)	Similar to map, but each input item can be mapped to 0 or more output items.
filter (<i>func</i>)	Return a new DStream by selecting only the records of the source DStream on which <i>func</i> returns true.
repartition (<i>numPartitions</i>)	Changes the level of parallelism in this DStream by creating more or fewer partitions.
union (<i>otherStream</i>)	Return a new DStream that contains the union of the elements in the source DStream and <i>otherDStream</i> .
count ()	Return a new DStream of single-element RDDs by counting the number of elements in each RDD of the source DStream.
reduce (<i>func</i>)	Return a new DStream of single-element RDDs by aggregating the elements in each RDD of the source DStream using a function <i>func</i> (which takes two arguments and returns one). The function should be associative and commutative so that it can be computed in parallel.

reduceByKey (<i>func</i> , [<i>numTasks</i>])	When called on a DStream of (K, V) pairs, return a new DStream of (K, V) pairs where the values for each key are aggregated using the given reduce function. Note: By default, this uses Spark's default number of parallel tasks (2 for local mode, and in cluster mode the number is determined by the config property <code>spark.default.parallelism</code>) to do the grouping. You can pass an optional <code>numTasks</code> argument to set a different number of tasks.
join (<i>otherStream</i> , [<i>numTasks</i>])	When called on two DStreams of (K, V) and (K, W) pairs, return a new DStream of (K, (V, W)) pairs with all pairs of elements for each key.
cogroup (<i>otherStream</i> , [<i>numTasks</i>])	When called on a DStream of (K, V) and (K, W) pairs, return a new DStream of (K, Seq[V], Seq[W]) tuples.
transform (<i>func</i>)	Return a new DStream by applying a RDD-to-RDD function to every RDD of the source DStream. This can be used to do arbitrary RDD operations on the DStream.
updateStateByKey (<i>func</i>)	Return a new "state" DStream where the state for each key is updated by applying the given function on the previous state of the key and the new values for the key. This can be used to maintain arbitrary state data for each key.

课后作业

- 1. 观察Spark UI中的各项指标，深入理解；
- 2. 将Spark作业提交到hadoop集群中执行；
- 3. 尝试获取到spark作业的RestFul API；
- 4. spark提交到yarn上后的日志总共有哪些种类？分别表示什么日志？
- 5. Spark UI 与 Spark History Server的UI大体类似，但他们有什么区别吗？原因是什么？

敬请指正！