

SPARK ON K8S在 阿里云上的实践

范佚伦（子灼），阿里云 开源大数据部
技术专家



目录 CONTENT

01 Spark on K8s介绍

02 Spark on K8s在阿里云 EMR的优化和最佳实践

01

Spark on k8s介绍

- 部署架构
- 社区进展
- 重点特性



Spark的集群部署模式

Spark支持多种类型的Cluster Manager，用于申请和分配程序(driver/executor)的资源

Standalone

- 使用Spark内置调度器，不适用生产环境

Hadoop YARN

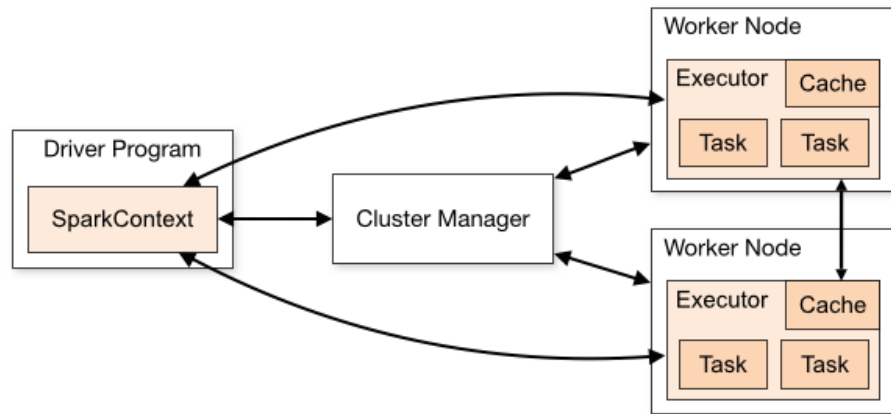
- 生产环境里最常用的部署方式
- 源于Hadoop，具有良好的社区生态

Apache Mesos

- 也是一个分布式资源管理框架，支持容器化
- 随着K8s兴起，使用者越来越少

Kubernetes

- 直接使用K8s调度和申请Spark作业资源，2021年Spark 3.1.1正式GA



Spark部署在K8s的优势

提高资源利用率

- 越来越多的在线应用集群、AI集群都运行在K8s里。Spark作业可以共享这些已有的集群资源，提高利用率。
- 在云上具备更好的弹性（如弹性容器实例），真正做到按量付费



统一运维方式

- 充分利用K8s社区生态，日志监控等工具
- 减少多个集群维护成本

容器化优势

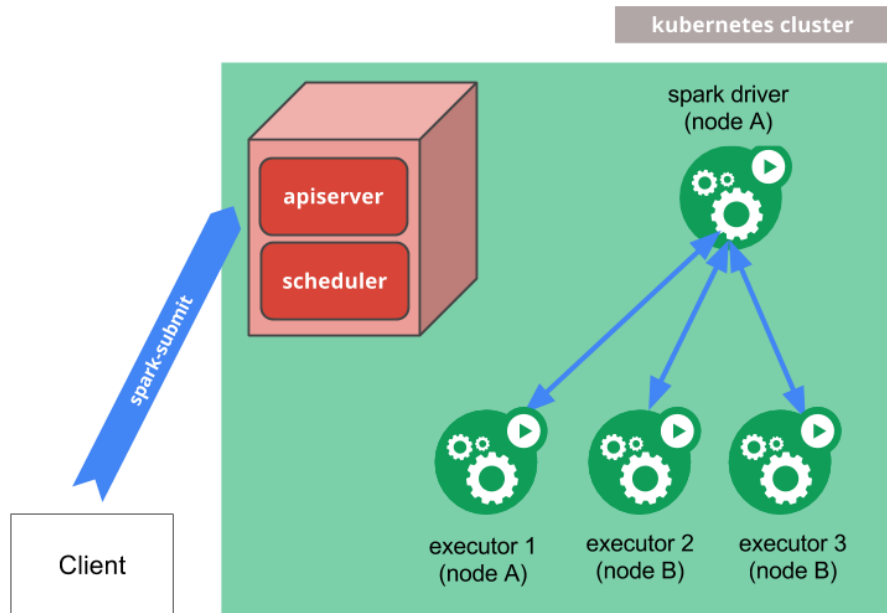
- 通过容器镜像管理依赖，提高Spark任务可移植性
- 多版本支持。不同Spark版本只是不同的镜像，做到了版本只和作业有关，和集群无关
- 对于版本升级、A/B Test更加友好

Spark on K8s部署架构

方式一：使用原生spark-submit

- K8s集群内无需提前配置和安装任何组件
- 提交作业的Client端需要安装Spark环境并配置kubectl
- 通过spark-submit提交作业，需要指定k8s apiserver地址及Spark镜像地址

```
$ ./bin/spark-submit \  
--master k8s://https://<k8s-apiserver-host>:<k8s-apiserver-port> \  
--deploy-mode cluster \  
--name spark-pi \  
--class org.apache.spark.examples.SparkPi \  
--conf spark.executor.instances=5 \  
--conf spark.kubernetes.container.image=<spark-image> \  
local:///path/to/examples.jar
```

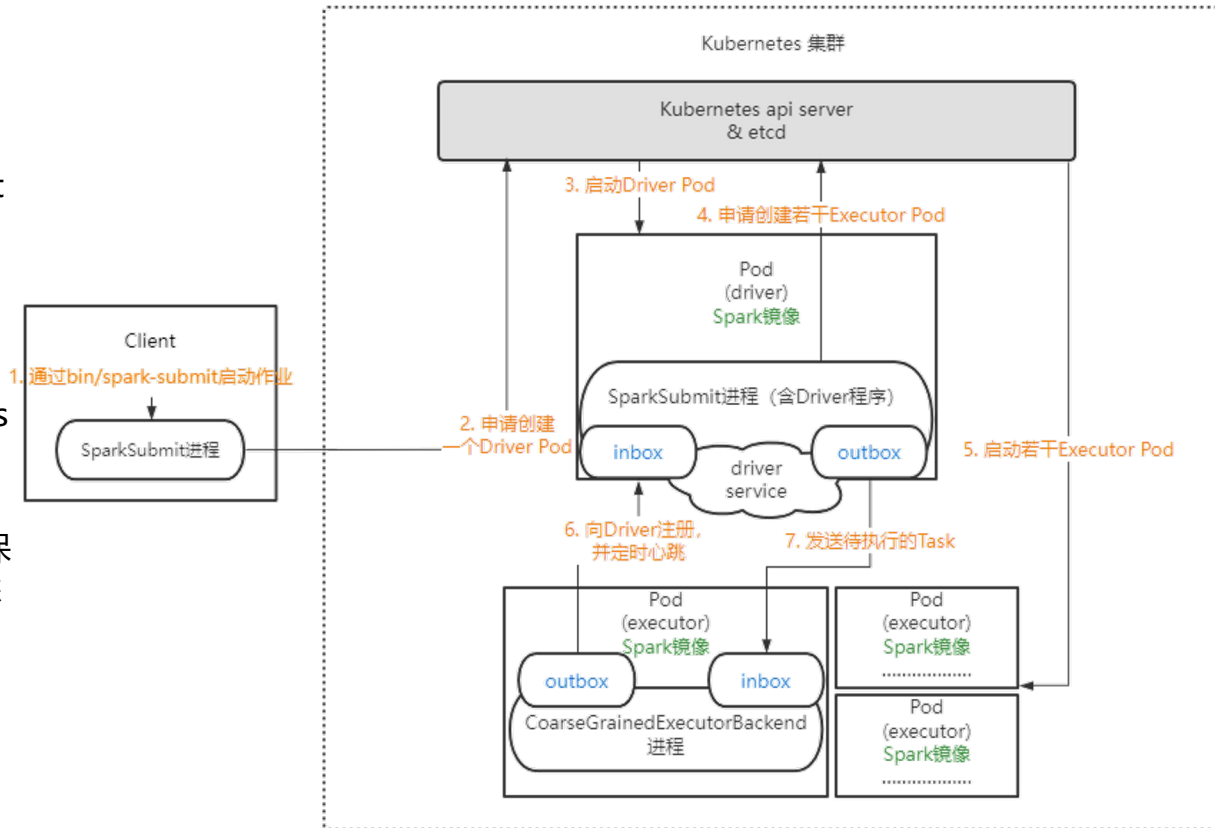


<https://spark.apache.org/docs/latest/running-on-kubernetes.html>

Spark on K8s部署架构

方式一：原生spark-submit运行流程

- 用户在Client端执行/bin/spark-submit命令
- 命令会在Client端启动SparkSubmit Java进程，通过fabric8的kubernetes-client请求K8s集群创建Driver Pod
- Driver Pod启动后，Driver负责连接k8s apiserver按需申请Executor Pod
- 作业运行完成后，Driver负责清理所有Executor Pod。Driver Pod结束后会保留Completed状态，以便于日志等状态查看。



Spark on K8s部署架构

方式二：使用spark-on-k8s-operator

- 由Google开源，是目前最常用的一种提交作业方式
- K8s集群需要事先安装spark-operator
- Client端通过kubectl提交一种yaml来提交作业
- 这种operator+CRD模式也是kubernetes官方推荐的一种部署复杂应用的模式
- **本质上是对原生方式的扩展，最终提交作业依然是使用spark-submit方式**，扩展的功能包括：定时调度，作业管理，监控，Pod增强等

1. 通过kubectl连接Kubernetes集群，详情请参见[通过kubectl工具连接集群](#)。

2. 新建spark-pi.yaml文件，文件内容如下。

```
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: spark-pi-simple
spec:
  type: Scala
  sparkVersion: 2.4.5
  mainClass: org.apache.spark.examples.SparkPi
  mainApplicationFile: "local:///opt/spark/examples/target/scala-2.11/jars/spark-examples_2
  arguments:
    - "1000"
  driver:
    cores: 1
    coreLimit: 1000m
    memory: 4g
  executor:
    cores: 1
    coreLimit: 1000m
    memory: 8g
    memoryOverhead: 1g
    instances: 1
```

3. 执行如下命令，提交作业。

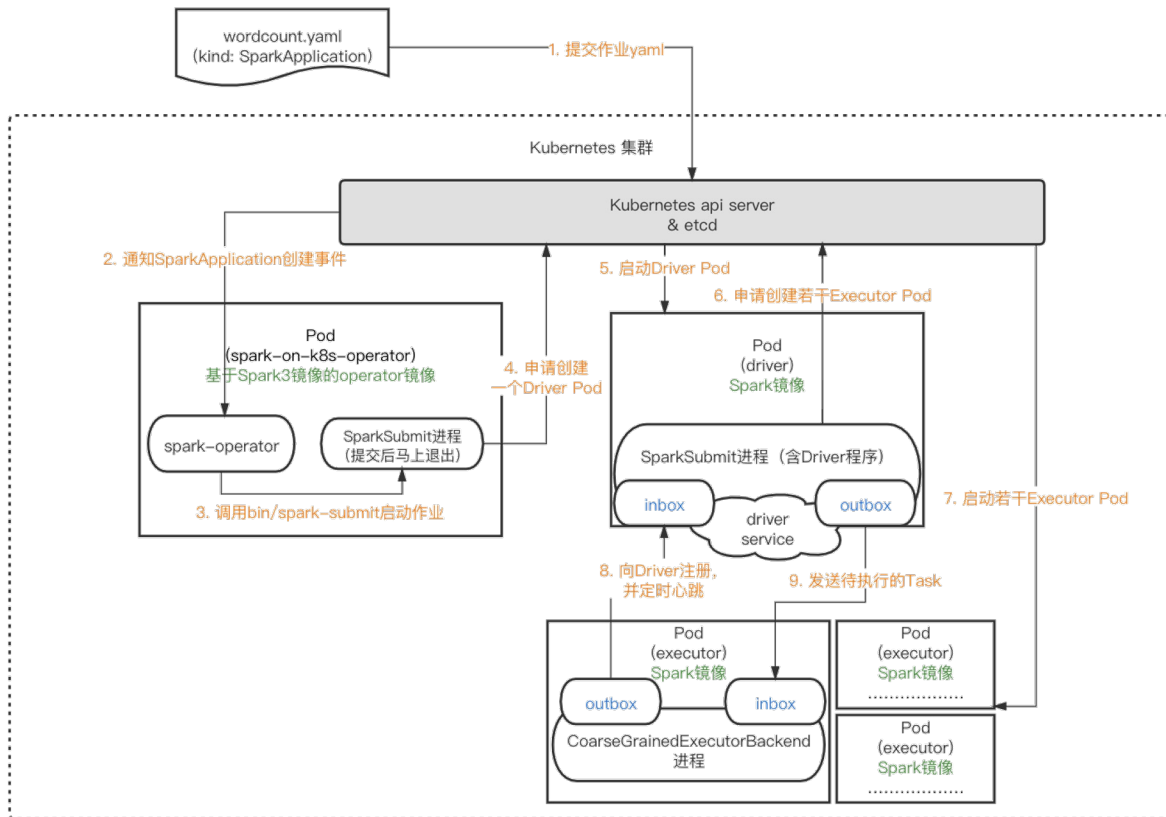
```
kubectl apply -f spark-pi.yaml --namespace <集群对应的namespace>
```

<https://github.com/GoogleCloudPlatform/spark-on-k8s-operator>

Spark on K8s部署架构

方式二：spark-on-k8s-operator运行流程

- 用户准备好作业的yaml描述文件，通过kubectl提交
- K8s集群内常驻的spark-operator pod会监听SparkApplication类型的事件，收到创建事件后，依然通过原生spark-submit方式提交Spark作业
- spark-operator可以通过k8s的Mutating Admission Webhook机制，拦截Kubernetes API请求，实现对Driver和Executor Pod的一些自定义配置
- spark-operator通过监听Driver和Executor Pod事件，更新SparkApplication的状态



Spark on K8s部署架构 - 对比

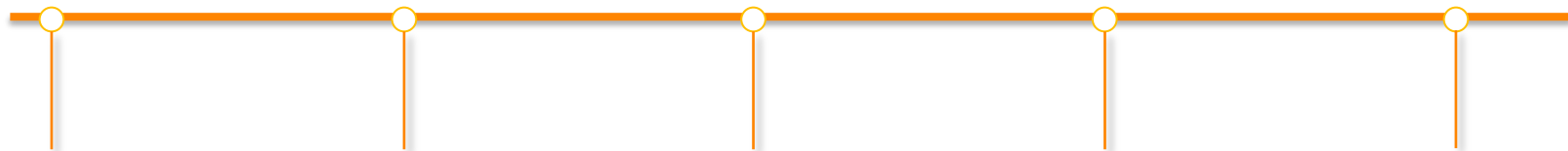
原生spark-submit优点

- 通过spark-submit命令提交，符合**用户习惯**
- 支持Spark Client模式，可以运行**交互式**作业（spark-shell）
- Client端**本地依赖**（jar包等资源）可以直接提交
- 与大数据调度平台**集成性**更好

spark-on-k8s-operator优点

- 所有作业都会记录在etcd中，便于跟踪状态和**管理作业**
- 支持作业**重试和定时**执行
- 提供作业监控指标对接**Prometheus**
- 通过TTL过期时间**自动清理**作业资源
- **自动配置**Spark UI的service/ingress

Spark on K8s社区进展



Spark 2.3 (2018.2)

- 首次支持Spark native on K8s

Spark 2.4 (2018.11)

- 支持运行Client模式
- 支持PySpark & R
- 新增volume挂载等多个k8s配置项

Spark 3.0 (2020.6)

- 支持自定义podTemplate
- 支持DynamicAllocation
- 支持client端本地依赖上传
- 支持Kerberos

Spark 3.1 (2021.3)

- **Spark on K8s 正式GA**
- 支持pvc动态创建
- 优化了executor pod申请的诸多问题
- 支持Node Decommission (节点优雅下线)

Spark 3.2 (2021.10)

- 支持pvc复用
- 支持自定义pod feature step
- 支持driver service自动清理

Spark on K8s重点特性

[SPARK-24434] Support user-specified driver and executor pod templates

- K8s的Pod定义通常采用YAML描述，Spark2对于Driver和Executor的Pod定义只能通过Spark conf逐个配置，如
 - spark.kubernetes.driver.label.[LabelName]
 - spark.kubernetes.driverEnv.[EnvironmentVariableName]
 - spark.kubernetes.driver.volumes.[VolumeType].[VolumeName].mount.path
 - spark.kubernetes.executor.volumes.[VolumeType].[VolumeName].options.[OptionName]
 - 这种方式灵活性很差，很多Pod的配置属性无法通过Spark定义。spark-operator可以通过YAML的方式提交Spark作业并通过Mutating Webhook增强Pod，但是有额外的性能损耗。
- Spark3.0开始，可以自定义podTemplateFile来定义Driver和Executor Pod
 - spark.kubernetes.driver.podTemplateFile=/path/to/driver-pod-template.yaml
 - spark.kubernetes.executor.podTemplateFile=/path/to/executor-pod-template.yaml
- Spark3.2，podTemplateFile支持放在远程S3/OSS存储上

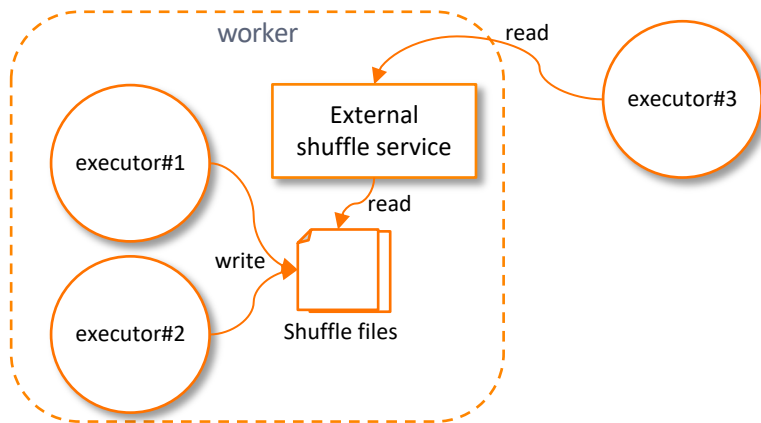
driverPodTemplate.yaml

```
apiVersion: v1
kind: Pod
spec:
  serviceAccountName: spark
  containers:
    - name: spark-kubernetes-driver
      env:
        - name: SPARKLOGENV
          value: spark-driver
        - name: SPARK_CONF_DIR
          value: /etc/spark/conf
      volumeMounts:
        - name: spark-defaults-configmap-volume
          mountPath: /etc/spark/conf
          readOnly: true
  nodeSelector:
    emr-spark: emr-spark
  tolerations:
    - key: emr-node-taint
      effect: NoSchedule
```

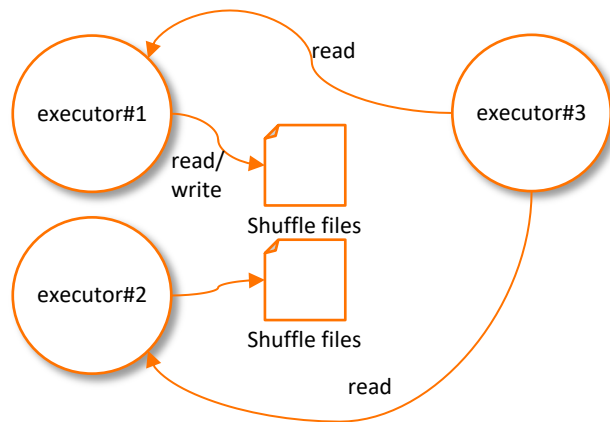
Spark on K8s重点特性

[SPARK-27963] Allow dynamic allocation without an external shuffle service

- Spark2的Dynamic Allocation强依赖External shuffle service (ESS), ESS负责维护节点上的Shuffle数据
- ESS通常由YARN的NodeManager启动, 难以在K8s等环境部署
- Spark3.0提供了ShuffleTracking特性, 在Spark内部跟踪每个executor的shuffle文件的生命周期, 允许在没有ESS的环境下开启Dynamic Allocation
- 但由于没有shuffle service, executor还需要负责给其他executor提供shuffle数据, 资源释放效率低



使用ESS: executor完成shuffle write就可以回收

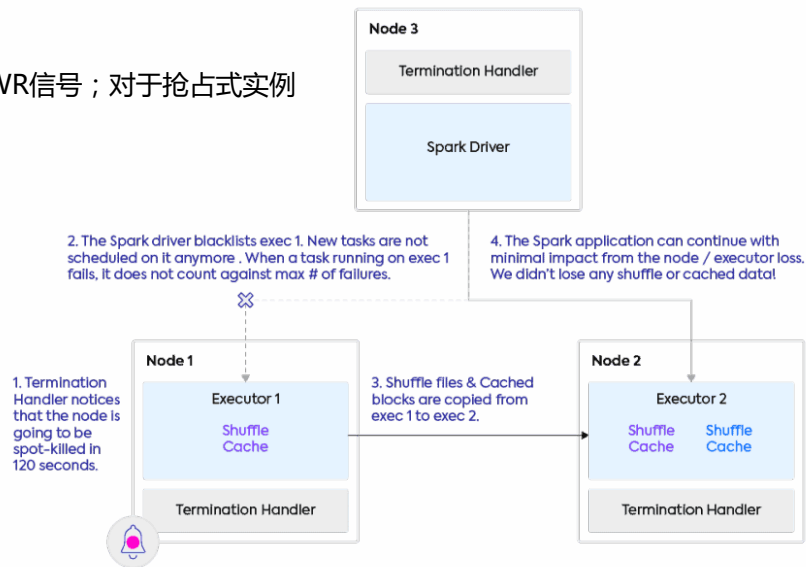


使用ShuffleTracking: 持有的shuffle数据失效的executor才会回收

Spark on K8s重点特性

[SPARK-20624] Add better handling for node shutdown

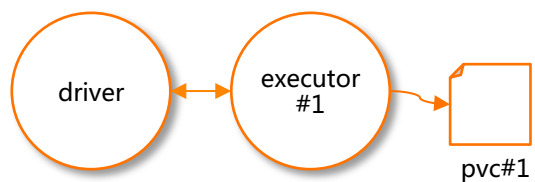
- Spark3.1支持K8s环境下node decommissioning (优雅下线) 功能, 目前属于 experimental
- 适用于节点下线、抢占式实例回收、pod驱逐等场景
- 运行流程：
 - Spark会在Executor Pod设置preStop hook脚本, 触发时发出SIGPWR信号; 对于抢占式实例可以自行在云服务器部署Termination Handler来触发。
 - Executor接收SIGPWR信号, 通知Driver不再调度新的Task
 - 当前Executor的Cache data和Shuffle files将会迁移到其他Executor
 - 如果迁移失败, 也可以fallback迁移到S3/OSS远程存储
 - 后续的Spark Task自动衔接到其他Executor运行, 避免重算



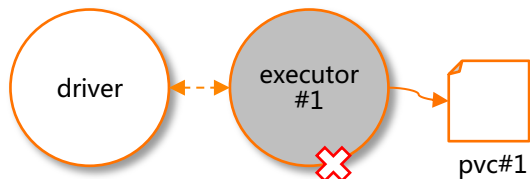
Spark on K8s重点特性

[SPARK-35416] Support PersistentVolumeClaim Reuse

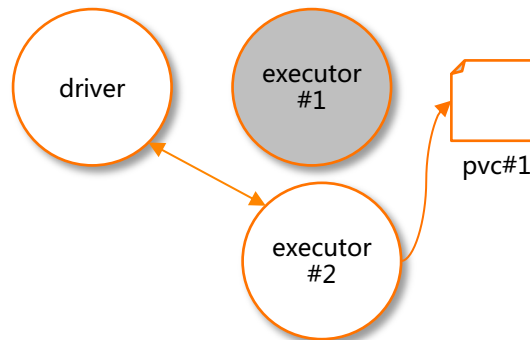
- Spark3.1支持Executor动态创建pvc，pvc生命周期随Executor释放
- Spark3.2支持pvc重用：pvc生命周期随Driver释放，这样即使Executor挂掉，新拉起的Executor会复用之前的pvc
 - 避免了pvc申请的消耗，提高性能
 - 丢失的Shuffle数据会自动恢复



1



2



3

02

Spark on K8s在阿里云EMR的优化和最佳实践

- 充分利用云上弹性优势
- 使用RSS优化shuffle和动态资源
- 增强K8s作业级别调度
- 云上数据湖存储加速
- 易用性提升



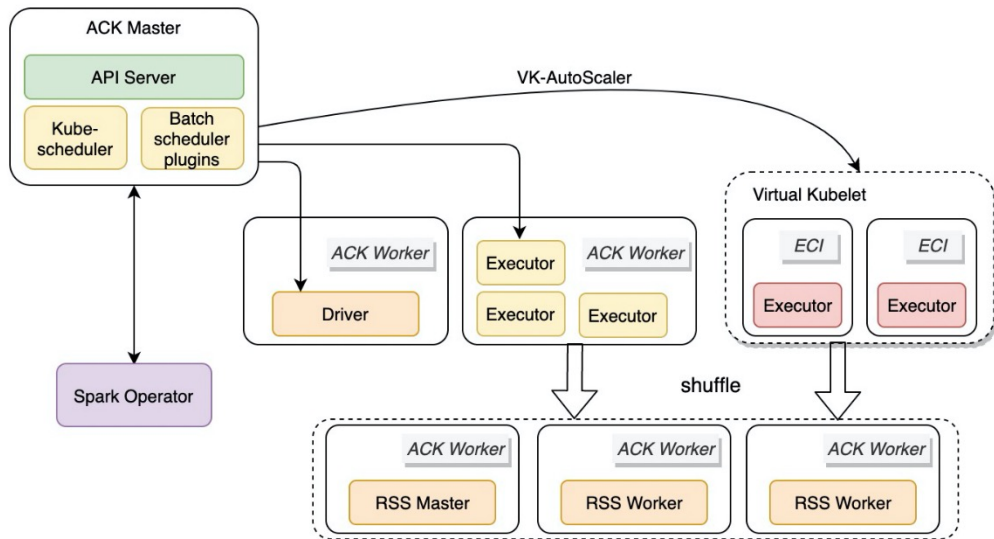
EMR Spark on ACK介绍

Spark on ACK介绍

- 阿里云EMR提供了EMR on ACK的产品，其中包含了Spark类型的集群（简称**Spark on ACK**），用于快速构建Spark on K8s的大数据平台
- *阿里云容器服务Kubernetes版，简称ACK
- *阿里云开源大数据平台E-MapReduce，简称EMR

Spark on ACK架构

- 采用虚拟集群的方式，直接在用户已有的ACK集群安装Spark相关组件
- 提供独立的RSS集群类型，优化容器环境下的Shuffle
- 支持常驻节点池+弹性ECI实例部署方式，节省成本



Spark on ACK架构

充分利用云上弹性优势

使用弹性实例调度Spark作业

阿里云提供了弹性容器实例ECI，无需提前申请底层服务器，按需申请，秒级启动。**非常适合Spark**类负载峰谷明显的计算场景。Spark on ACK可以便捷地接入ECI实现弹性调度。

- 优点1：节省成本
 - 无需购买大量常驻资源，完全按量申请
 - 按秒计费，启动和销毁的损耗低
- 优点2：高性能
 - 秒级启动，镜像缓存，对Spark作业速度几乎无延迟
- 优点3：提升Spark作业速度
 - 通过申请更多executor，加快作业速度，总CU时成本几乎不变
- 优点4：资源充足，避免排队
 - 传统固定集群中，高峰期作业大量排队，低优任务面临饥饿等问题。使用弹性实例随时可以拉起，大幅降低集群整体运行时间

Spark集群当前选定一个100个节点的集群进行固定ACK+弹性ECI的模式进行稳定性效果验证，考虑到按量库存，比较保守



包月固定集群规模：
100台Spark
64c 256g



包月固定集群规模：
50台Spark
64c 256g

弹性ECI，使用ECI时间段：高峰期凌晨2点（ETL任务抽取结束）到早上6点（预留两个小时补救时间）

按照高峰期64c 256g*50成本核算，平均成本降低约35%~40%（原先节点包年包月，小规格实例按量更便宜），之前ACK采用的是64c 256g节点，在新的弹性ECI模式下，部分任务用户采用多executor单task代替单个executor多个task的方式，整体性能平均提升10%~20%。

充分利用云上弹性优势

使用抢占式 (Spot) ECI实例进一步节省成本

- 抢占式实例是一种低成本竞价型实例，可以获得大幅度**价格优惠**
- 使用**配置简单**，只需添加配置特定的Spark配置项
- 默认有**1小时**的保护期，适用于大部分Spark批处理场景
- 超出保护期后，抢占式实例可能被强制回收
 - 强制回收之前10分钟左右，实例会发出中断事件提示
 - 利用Spark3的**Decommission**特性，可以提前移动缓存和shuffle数据块，提高运行效率。

使用RSS优化shuffle和动态资源

Spark Shuffle在K8s环境下的挑战

• Spark Shuffle对本地存储的依赖

- 云上有许多计算存储分离的机型没有自带本地盘，Spark作业难以直接利用到这些节点的计算资源
- 如果对Pod挂载云盘或者使用弹性实例，挂载云盘的大小难以确定，考虑到数据倾斜等因素，云盘的使用率也会比较低，且性价比不如本地存储

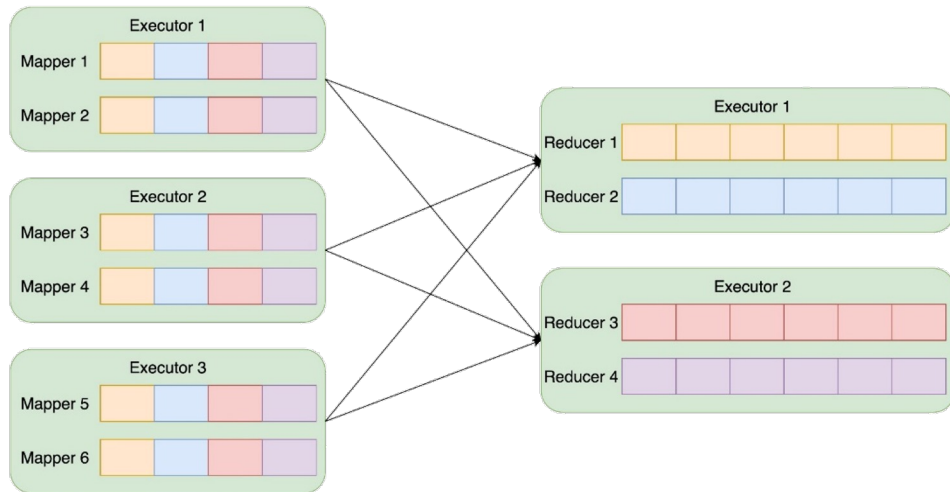
• 不完美的Dynamic Allocation

- 受益于K8s集群混部和弹性容器的支持，Dynamic Allocation在K8s环境尤其重要。
- Dynamic Allocation依赖ESS，而ESS不支持在K8s环境部署（因为社区倾向remote shuffle，ESS相关支持并未合并）
- Spark2不支持无ESS的Dynamic Allocation
- Spark3支持的无ESS的Dynamic Allocation[SPARK-27963]，但Executor无法及时回收，造成资源浪费
 - 整个shuffle阶段结束后，Driver端GC触发shuffle file清理才会回收Executor
 - 尤其是长尾任务会拖住所有Executor资源不能释放

使用RSS优化shuffle和动态资源

除此之外，Spark Shuffle本身的不足

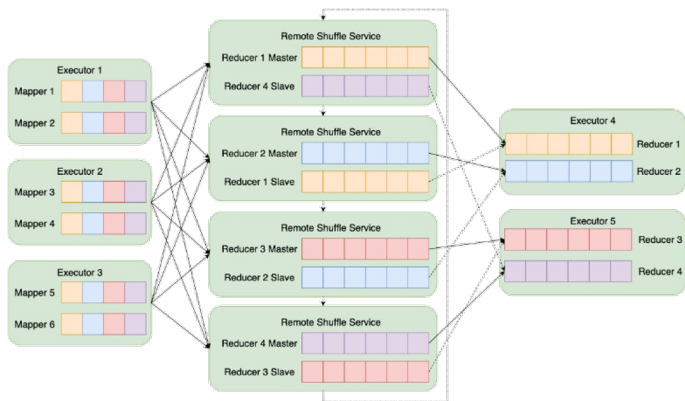
- **写放大**。当Mapper输出数据量超过内存时触发外排，Spill到本地磁盘，从而引入额外磁盘IO。
- **大量随机读**。Reducer并发拉取Mapper端的数据，导致大量小粒度随机读，影响性能。
- **高网络连接数**。产生 $\text{numMapper} * \text{numReducer}$ 个网络连接，会导致线程池消耗过多CPU，带来性能和稳定性问题。
- **Shuffle数据单副本**。Shuffle数据丢失会引发的Stage重算。在K8s环境里更为普遍：遭遇Pod驱逐、抢占式实例回收等情况。



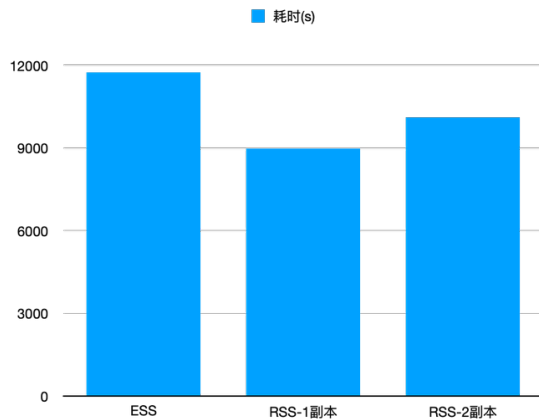
使用RSS优化shuffle和动态资源

阿里云Remote Shuffle Service (RSS)

- 作为**独立服务**保存Shuffle数据，更适合存算分离架构。让Executor不再依赖本地盘。
- 采用Push Shuffle模式， Shuffle过程优化为**追加写、顺序读**。
- Master-Worker架构，Master节点管理RSS服务状态，并支持HA部署、多副本。
- 完美支持**动态资源**，Executor及时释放。
- Spark on ACK可以一键创建并关联RSS集群，充分利用RSS的优势。



RSS架构图

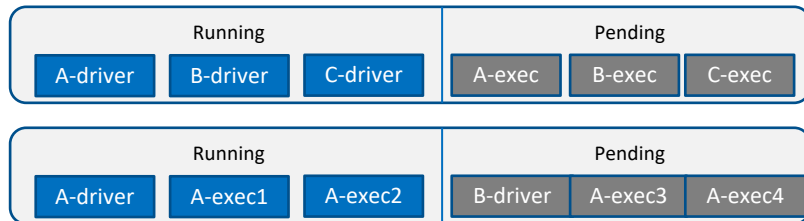


TPC-DS 10T测试 RSS单副本/两副本分别比ESS快23.5%/13.8%

增强K8s作业级别调度

K8s默认调度器调度大数据作业的挑战

- **K8s调度的粒度是pod，而大数据需要调度application**
 - 没有针对Driver做资源限制。例如，一瞬间提交大量作业，然后资源全部被Driver占满，形成死锁
 - 按Pod提交顺序排队，不是按App提交顺序排队。排在前面的App中途资源请求不能优先满足
 - 缺乏统一的作业视图（如YARN UI）
- **多租户场景支持不佳**（基于namespace的resource quota）
 - namespace超过配额的请求会被拒绝，无法形成队列
 - K8s的user仅用于authentication，没有真实的**user**概念
 - 不支持**动态配额**，无法适应扩缩容
 - 不支持**多层级**的树状结构配额设置
 - 不支持租户之间**弹性调度**和资源抢占
- **调度策略单一**（默认优先级+FIFO）
 - 同一个队列里，需要支持**公平**调度
 - 不同队列之间，需要支持**公平**调度



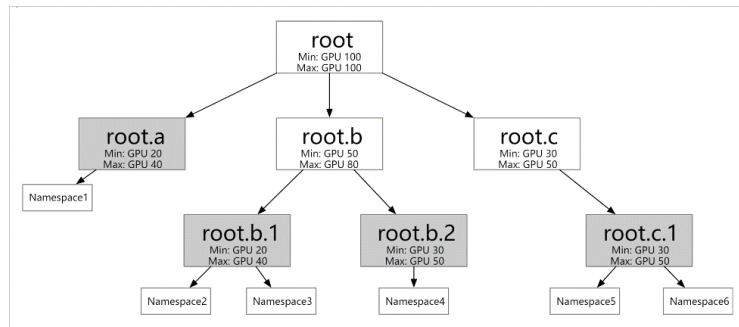
增强K8s作业级别调度

阿里云ACK对大数据作业调度的增强

- ACK基于Kubernetes Scheduling Framework扩展机制，增强了批计算的调度能力
- Spark on ACK作业可以基于如下方案，优化大数据作业调度
- **Gang scheduling**
 - All-or-Nothing调度
 - 当集群资源满足该pod-group最少运行个数时，才会统一调度
 - 适用于executor批量调度
- **Capacity Scheduling**
 - 基于namespace实现多租户树状队列
 - 每个队列可以设置资源上限、下限，支持队列间抢占
- **Kube-queue**
 - 实现了按App粒度调度Spark作业的优先级队列
 - 当作业所有申请资源满足集群剩余空间时，才会调度当前作业
 - 支持队列间公平调度
 - 基于spark-operator扩展，提交作业自动接入队列

使用Gang scheduling时，在创建的Pod处通过设置labels的形式配置min-available和name。

```
labels:  
  pod-group.scheduling.sigs.k8s.io/name: tf-smoke-gpu  
  pod-group.scheduling.sigs.k8s.io/min-available: "3"
```



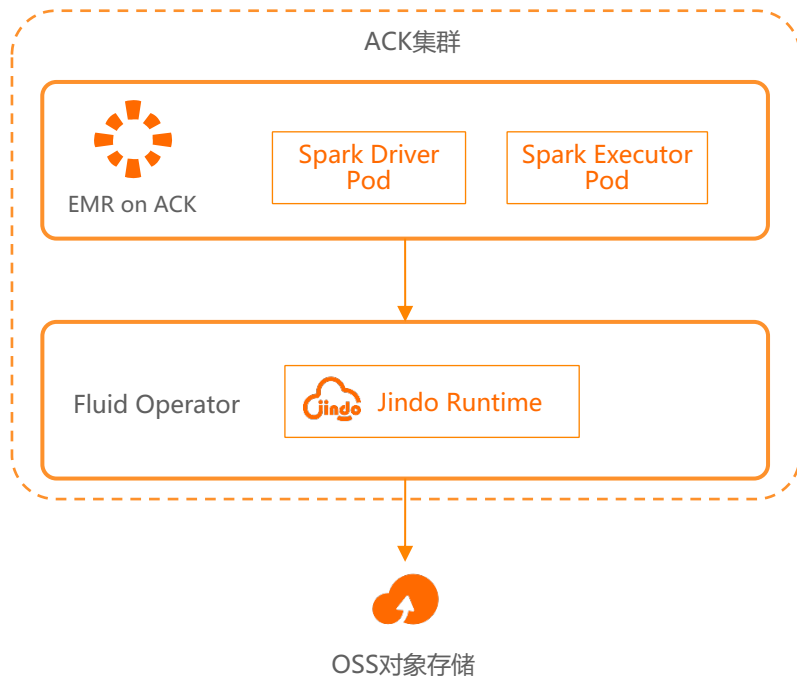
云上数据湖存储与加速

Spark on K8s更适合存储计算分离架构

- K8s环境下，**不再依赖**传统的**Hadoop集群**（YARN+HDFS），云上**数据湖存储OSS**是HDFS更好的替代
- OSS具备高可靠，免运维，高安全，低成本等特点
- Spark on ACK内置**Jindo SDK**，引擎无缝对接OSS读写

使用Fluid+JindoFS加速OSS文件访问

- **Fluid**是一个K8s原生的分布式数据集编排和加速引擎
- 可以同时给大数据和AI应用提供缓存加速
- 在TPC-DS场景下，使用Fluid加速可以**提升运行速度30%左右**



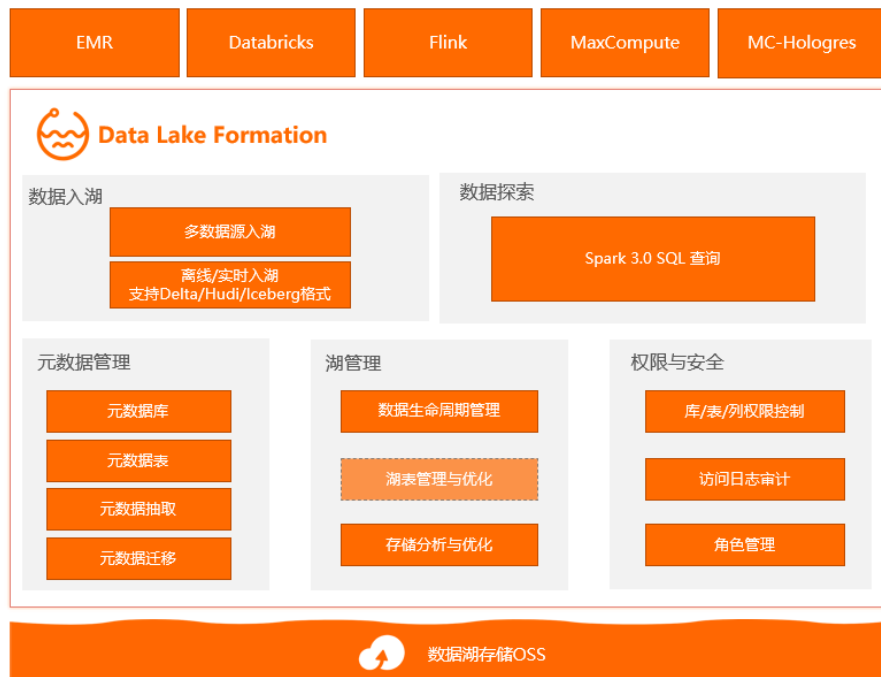
使用DLF构建云上数据湖

K8s上没有整套Hadoop集群，Hadoop生态圈组件如何使用？

- 要实现SparkSQL元数据管理，需要部署Hive Metastore
- 要实现MySQL数据入湖，需要部署Sqoop
- 要实现权限管理与审计，需要部署Ranger
- 要实现交互式查询，需要部署Hue
-

Spark on ACK无缝对接阿里云DLF (Data Lake Formation)

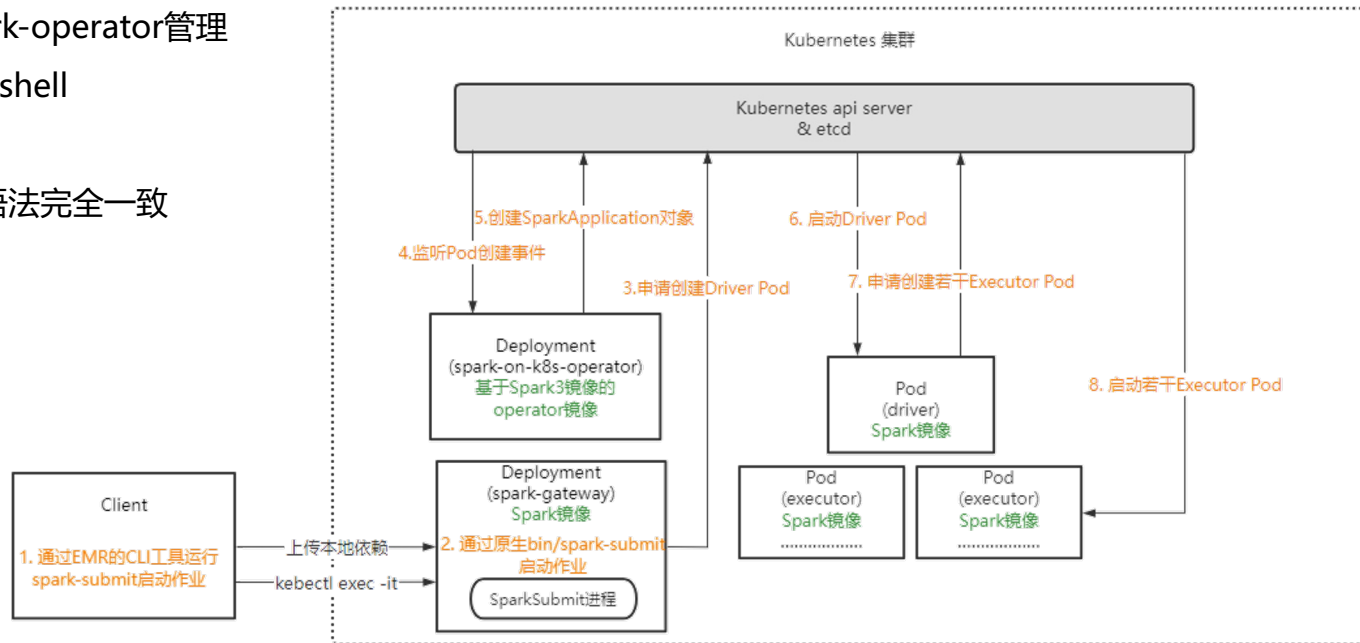
- 统一**元数据服务**，兼容开源HMS协议，支持多引擎
 - 相比Hive Metastore，高可用免运维，支持亿级Partition
- 支持库/表/列级别**权限控制**和审计
- 提供多种数据源**数据入湖**功能
- 提供数据探索功能，便捷的Spark SQL**交互式分析**
- 提供湖管理功能，可以进行**存储分析**与成本优化



易用性提升

让spark-operator作业也能通过spark-submit提交

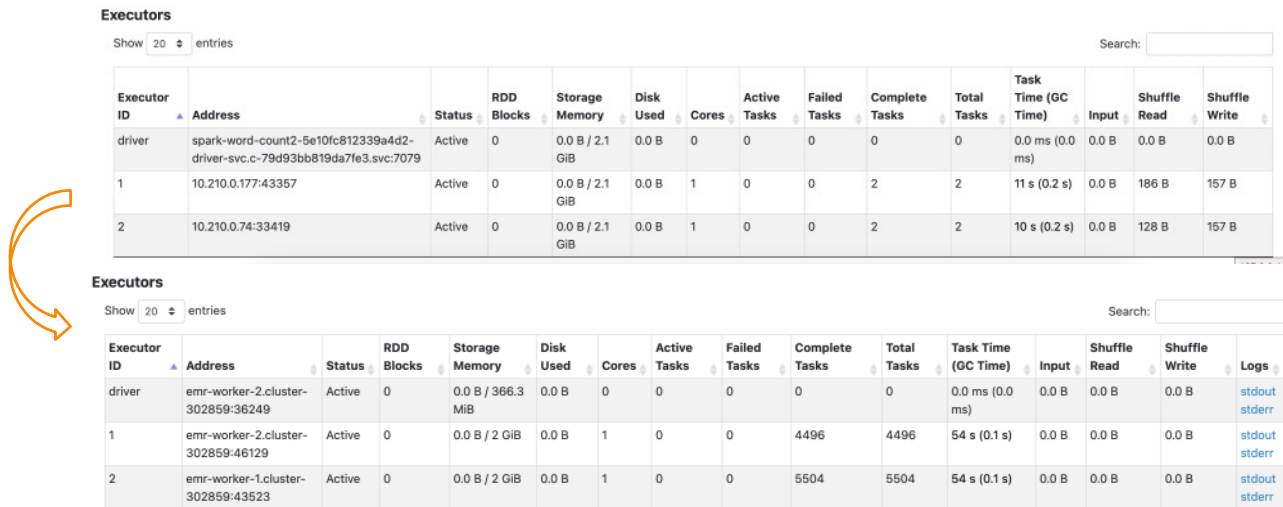
- Spark on ACK提供了CLI工具，在**无需安装**Spark的客户端通过spark-submit提交作业
- 融合**spark-submit和spark-operator两种作业提交方式的**优点**
 - 所有作业都能通过spark-operator管理
 - 支持运行交互式spark-shell
 - 支持本地依赖提交
 - 与原生spark-submit语法完全一致



易用性提升

HistoryServer透出Spark日志

- YARN具备Log Aggregation功能，容器日志统一收集在HDFS上，可以在HistoryServer透出
- K8s可以通过云上日志服务收集日志，但无法在HistoryServer等UI页面透出
- (进行中) Spark on ACK提供日志收集方案，并直接通过HistoryServer透出，延续了Spark的用户体验



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
driver	spark-word-count2-5e10fc812339a4d2-driver-svc.c-79d93bb819da7fe3.svc:7079	Active	0	0.0 B / 2.1 GiB	0.0 B	0	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B
1	10.210.0.177:43357	Active	0	0.0 B / 2.1 GiB	0.0 B	1	0	0	2	2	11 s (0.2 s)	0.0 B	186 B	157 B
2	10.210.0.74:33419	Active	0	0.0 B / 2.1 GiB	0.0 B	1	0	0	2	2	10 s (0.2 s)	0.0 B	128 B	157 B

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	Logs
driver	emr-worker-2.cluster-302859:36249	Active	0	0.0 B / 366.3 MiB	0.0 B	0	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B	stdout stderr
1	emr-worker-2.cluster-302859:46129	Active	0	0.0 B / 2 GiB	0.0 B	1	0	0	4496	4496	54 s (0.1 s)	0.0 B	0.0 B	0.0 B	stdout stderr
2	emr-worker-1.cluster-302859:43523	Active	0	0.0 B / 2 GiB	0.0 B	1	0	0	5504	5504	54 s (0.1 s)	0.0 B	0.0 B	0.0 B	stdout stderr

非常感谢您的观看

阿里云 | DataFun.

