

尚硅谷大数据技术之 Spark 基础解析

(作者：尚硅谷大数据研发部)

版本：V1.1

第 1 章 Spark 概述

1.1 什么是 Spark

什么是Spark



1、定义

Spark是一种基于内存的快速、通用、可扩展的大数据分析引擎。

2、历史

2009年诞生于加州大学伯克利分校AMPLab，项目采用Scala编写。

2010年开源；

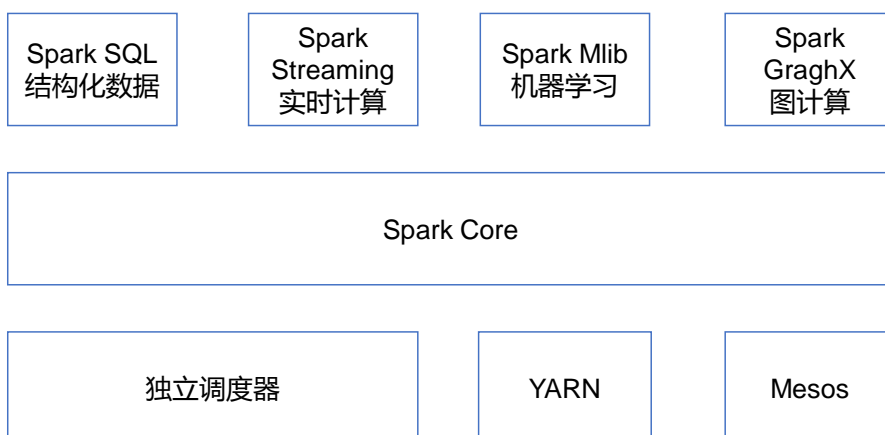
2013年6月成为Apache孵化项目

2014年2月成为Apache顶级项目。

让天下没有难学的技术

1.2 Spark 内置模块

Spark内置模块



让天下没有难学的技术

Spark Core: 实现了 Spark 的基本功能，包含任务调度、内存管理、错误恢复、与存储系统交互等模块。Spark Core 中还包含了对弹性分布式数据集(Resilient Distributed DataSet, 简称 RDD)的 API 定义。

Spark SQL: 是 Spark 用来操作结构化数据的程序包。通过 Spark SQL，我们可以使用 SQL 或者 Apache Hive 版本的 SQL 方言(HQL)来查询数据。Spark SQL 支持多种数据源，比如 Hive 表、Parquet 以及 JSON 等。

Spark Streaming: 是 Spark 提供的对实时数据进行流式计算的组件。提供了用来操作数据流的 API，并且与 Spark Core 中的 RDD API 高度对应。

Spark MLlib: 提供常见的机器学习(ML)功能的程序库。包括分类、回归、聚类、协同过滤等，还提供了模型评估、数据 导入等额外的支持功能。

集群管理器: Spark 设计为可以高效地在一个计算节点到数千个计算节点之间伸缩计算。为了实现这样的要求，同时获得最大灵活性，Spark 支持在各种集群管理器(Cluster Manager)上运行，包括 Hadoop YARN、Apache Mesos，以及 Spark 自带的一个简易调度器，叫作独立调度器。

Spark 得到了众多大数据公司的支持，这些公司包括 Hortonworks、IBM、Intel、Cloudera、MapR、Pivotal、百度、阿里、腾讯、京东、携程、优酷土豆。当前百度的 Spark 已应用于大搜索、直达号、百度大数据等业务；阿里利用 GraphX 构建了大规模的图计算和图挖掘系统，实现了很多生产系统的推荐算法；腾讯 Spark 集群达到 8000 台的规模，是当前已知的世界上最大的 Spark 集群。

1.3 Spark 特点



- 1) **快**: 与Hadoop的MapReduce相比, Spark基于内存的运算要快100倍以上, 基于硬盘的运算也要快10倍以上。Spark实现了高效的DAG执行引擎, 可以通过基于内存来高效处理数据流。计算的中间结果是存在于内存中的。
- 2) **易用**: Spark支持Java、Python和Scala的API, 还支持超过80种高级算法, 使用户可以快速构建不同的应用。而且Spark支持交互式的Python和Scala的Shell, 可以非常方便地在这些Shell中使用Spark集群来验证解决问题的方法。
- 3) **通用**: Spark提供了统一的解决方案。Spark可以用于批处理、交互式查询 (Spark SQL)、实时流处理 (Spark Streaming)、机器学习 (Spark MLlib) 和图计算 (GraphX)。这些不同类型的处理都可以在同一个应用中无缝使用。减少了开发和维护的人力成本和部署平台的物力成本。
- 4) **兼容性**: Spark可以非常方便地与其他开源产品进行融合。比如, Spark可以使用Hadoop的YARN和Apache Mesos作为它的资源管理和调度器, 并且可以处理所有Hadoop支持的数据, 包括HDFS、HBase等。这对于已经部署Hadoop集群的用户特别重要, 因为不需要做任何数据迁移就可以使用Spark的强大处理能力。

让天下没有难学的技术

第 2 章 Spark 运行模式

2.1 Spark 安装地址

1. 官网地址

<http://spark.apache.org/>

2. 文档查看地址

<https://spark.apache.org/docs/2.1.1/>

3. 下载地址

<https://spark.apache.org/downloads.html>

2.3 Local 模式

2.3.1 概述



Local模式就是运行在一台计算机上的模式，通常就是用于在本机上练手和测试。它可以通过以下集中方式设置Master。

local: 所有计算都运行在一个线程当中，没有任何并行计算，通常我们在本机执行一些测试代码，或者练手，就用这种模式；

local[K]: 指定使用几个线程来运行计算，比如local[4]就是运行4个Worker线程。通常我们的Cpu有几个Core，就指定几个线程，最大化利用Cpu的计算能力；

local[*]: 这种模式直接帮你按照Cpu最多Cores来设置线程数了。

让天下没有难学的技术

2.3.2 安装使用

1) 上传并解压 spark 安装包

```
[atguigu@hadoop102 software]$ tar -zxvf spark-2.1.1-bin-hadoop2.7.tgz -C /opt/module/[atguigu@hadoop102 module]$ mv spark-2.1.1-bin-hadoop2.7 spark
```

2) 官方求 PI 案例

```
[atguigu@hadoop102 spark]$ bin/spark-submit \
--class org.apache.spark.examples.SparkPi \
--executor-memory 1G \
--total-executor-cores 2 \
./examples/jars/spark-examples_2.11-2.1.1.jar \
100
```

(1) 基本语法

```
bin/spark-submit \
--class <main-class> \
--master <master-url> \
--deploy-mode <deploy-mode> \
--conf <key>=<value> \
... # other options
<application-jar> \
[application-arguments]
```

(2) 参数说明:

--master 指定 Master 的地址，默认为 Local

更多 [Java](#) - [大数据](#) - [前端](#) - [python](#) 人工智能资料下载，可百度访问：尚硅谷官网

--class: 你的应用的启动类 (如 org.apache.spark.examples.SparkPi)
--deploy-mode: 是否发布你的驱动到 worker 节点(cluster) 或者作为一个本地客户端 (client) (default: client)*
--conf: 任意的 Spark 配置属性, 格式 key=value. 如果值包含空格, 可以加引号 "key=value"
application-jar: 打包好的应用 jar, 包含依赖. 这个 URL 在集群中全局可见. 比如 hdfs:// 共享存储系统, 如果是 file:// path, 那么所有的节点的 path 都包含同样的 jar
application-arguments: 传给 main()方法的参数
--executor-memory 1G 指定每个 executor 可用内存为 1G
--total-executor-cores 2 指定每个 executor 使用的 cup 核数为 2 个

3) 结果展示

该算法是利用蒙特·卡罗算法求 PI

```
18/09/29 09:04:35 INFO TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
18/09/29 09:04:35 INFO DAGScheduler: Job 0 finished: reduce at SparkPi.scala:38, took 2.810863 s
Pi is roughly 3.1410779141077914
18/09/29 09:04:35 INFO SparkUI: Stopped Spark web UI at http://192.168.9.102:4040
18/09/29 09:04:35 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
18/09/29 09:04:35 INFO MemoryStore: MemoryStore cleared
18/09/29 09:04:35 INFO BlockManager: BlockManager stopped
18/09/29 09:04:35 INFO BlockManagerMaster: BlockManagerMaster stopped
18/09/29 09:04:35 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
18/09/29 09:04:35 INFO SparkContext: Successfully stopped sparkContext
18/09/29 09:04:35 INFO ShutdownHookManager: Shutdown hook called
18/09/29 09:04:35 INFO ShutdownHookManager: Deleting directory /tmp/spark-afa7967e-0946-40d1-b786-b41a992ff624
[atguigu@hadoop102 spark]$
```

4) 准备文件

```
[atguigu@hadoop102 spark]$ mkdir input
```

在 input 下创建 3 个文件 1.txt 和 2.txt, 并输入以下内容

```
hello atguigu
hello spark
```

5) 启动 spark-shell

```
[atguigu@hadoop102 spark]$ bin/spark-shell
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
18/09/29 08:50:52 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
18/09/29 08:50:58 WARN ObjectStore: Failed to get database global_temp, returning NoSuchObjectException
Spark context Web UI available at http://192.168.9.102:4040
Spark context available as 'sc' (master = local[*], app id = local-1538182253312).
Spark session available as 'spark'.
Welcome to
```

```
Using Scala version 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java
1.8.0_144)
Type in expressions to have them evaluated.
Type :help for more information.
```

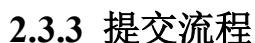
开启另一个 CRD 窗口

可登录 `hadoop102:4040` 查看程序运行

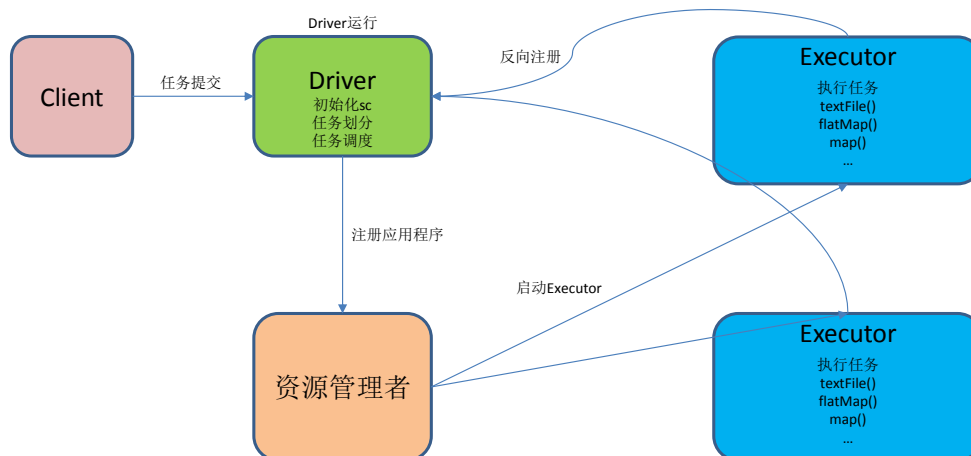


```
scala>
```

可登录 `hadoop102:4040` 查看程序运行



1) 提交任务分析:



让天下没有难学的技术

重要角色：

Driver（驱动器）

Spark 的驱动器是执行开发程序中的 main 方法的进程。它负责开发人员编写的用来创建 SparkContext、创建 RDD，以及进行 RDD 的转化操作和行动操作代码的执行。如果你是用 spark shell，那么当你启动 Spark shell 的时候，系统后台自启了一个 Spark 驱动器程序，就是在 Spark shell 中预加载的一个叫作 sc 的 SparkContext 对象。如果驱动器程序终止，那么 Spark 应用也就结束了。主要负责：

- 1) 把用户程序转为任务
- 2) 跟踪 Executor 的运行状况
- 3) 为执行器节点调度任务
- 4) UI 展示应用运行状况

Executor（执行器）

Spark Executor 是一个工作进程，负责在 Spark 作业中运行任务，任务间相互独立。Spark 应用启动时，Executor 节点被同时启动，并且始终伴随着整个 Spark 应用的生命周期而存在。如果有 Executor 节点发生了故障或崩溃，Spark 应用也可以继续执行，会将出错节点上的任务调度到其他 Executor 节点上继续运行。主要负责：

- 1) 负责运行组成 Spark 应用的任务，并将结果返回给驱动器进程；

2) 通过自身的块管理器 (Block Manager) 为用户程序中要求缓存的 RDD 提供内存式存储。RDD 是直接缓存在 Executor 进程内的, 因此任务可以在运行时充分利用缓存数据加速运算。

2.3.4 数据流程

`textFile("input")`: 读取本地文件 input 文件夹数据;

`flatMap(_.split(" "))`: 压平操作, 按照空格分割符将一行数据映射成一个个单词;

`map((_,1))`: 对每一个元素操作, 将单词映射为元组;

`reduceByKey(_+_)`: 按照 key 将值进行聚合, 相加;

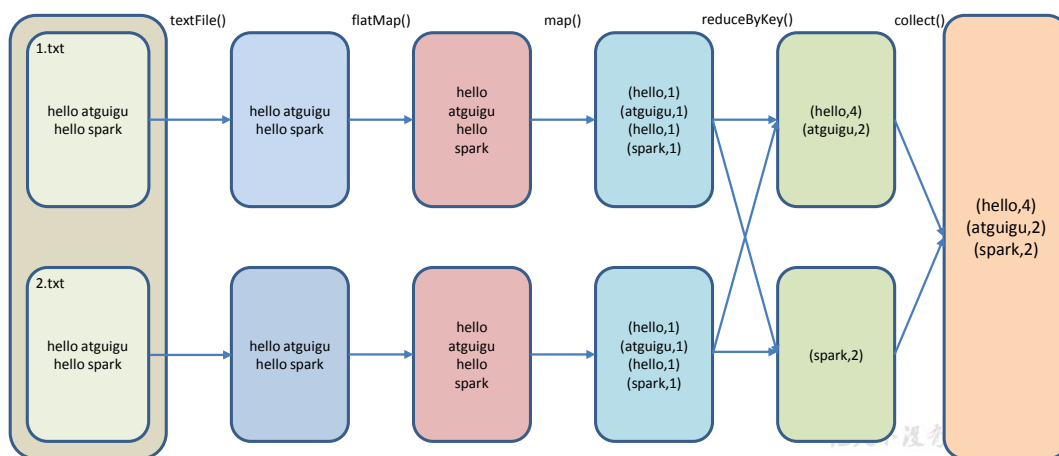
`collect`: 将数据收集到 Driver 端展示。



WordCount案例分析



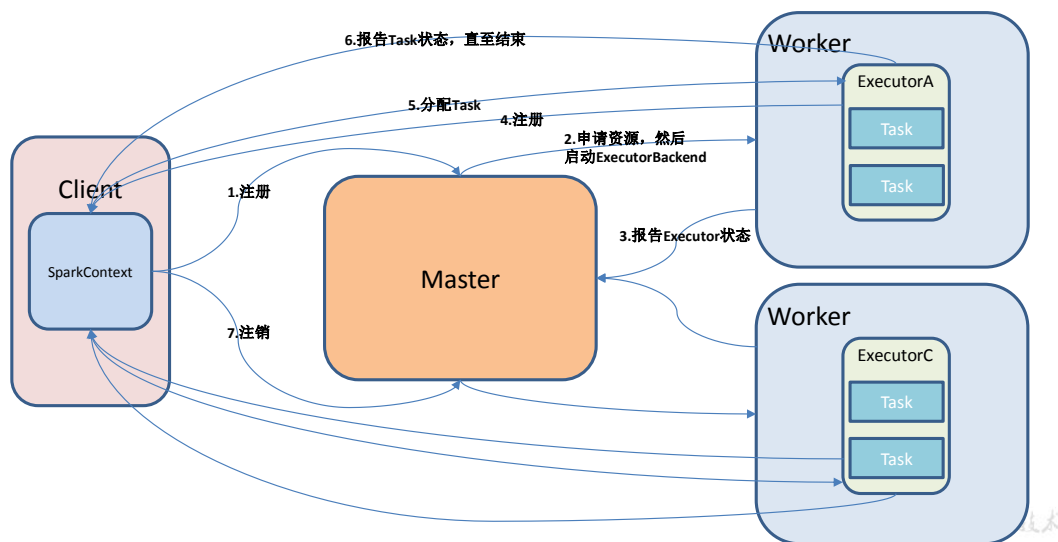
```
sc.textFile("input").flatMap(_.split(" ")).map((_,1)).reduceByKey(_+_).collect
```



2.4 Standalone 模式

2.4.1 概述

构建一个由 Master+Slave 构成的 Spark 集群, Spark 运行在集群中。



2.4.2 安装使用

1) 进入 spark 安装目录下的 conf 文件夹

```
[atguigu@hadoop102 module]$ cd spark/conf/
```

2) 修改配置文件名称

```
[atguigu@hadoop102 conf]$ mv slaves.template slaves
[atguigu@hadoop102 conf]$ mv spark-env.sh.template spark-env.sh
```

3) 修改 slave 文件，添加 work 节点：

```
[atguigu@hadoop102 conf]$ vim slaves
```

```
hadoop102
hadoop103
hadoop104
```

4) 修改 spark-env.sh 文件，添加如下配置：

```
[atguigu@hadoop102 conf]$ vim spark-env.sh
```

```
SPARK_MASTER_HOST=hadoop101
SPARK_MASTER_PORT=7077
```

5) 分发 spark 包

```
[atguigu@hadoop102 module]$ xsync spark/
```

6) 启动

```
[atguigu@hadoop102 spark]$ sbin/start-all.sh
[atguigu@hadoop102 spark]$ util.sh
=====atguigu@hadoop102=====
3330 Jps
3238 Worker
3163 Master
=====atguigu@hadoop103=====
2966 Jps
2908 Worker
```

更多 Java -大数据 -前端 -python 人工智能资料下载，可百度访问：尚硅谷官网

```
=====atguigu@hadoop104=====
2978 Worker
3036 Jps
```

网页查看: hadoop102:8080

注意: 如果遇到“JAVA_HOME not set”异常, 可以在 sbin 目录下的 spark-config.sh 文件中加入如下配置:

```
export JAVA_HOME=XXXX
```

7) 官方求 PI 案例

```
[atguigu@hadoop102 spark]$ bin/spark-submit \
--class org.apache.spark.examples.SparkPi \
--master spark://hadoop102:7077 \
--executor-memory 1G \
--total-executor-cores 2 \
./examples/jars/spark-examples_2.11-2.1.1.jar \
100
```

```
99, PROCESS_LOCAL, 6030 bytes)
17/07/26 00:21:15 INFO TaskSetManager: Finished task 97.0 in stage 0.0 (TID 97) in 26 ms on 172.16.148.152 (executor 0) (98/100)
17/07/26 00:21:15 INFO TaskSetManager: Finished task 99.0 in stage 0.0 (TID 99) in 90 ms on 172.16.148.152 (executor 0) (99/100)
17/07/26 00:21:15 INFO TaskSetManager: Finished task 98.0 in stage 0.0 (TID 98) in 100 ms on 172.16.148.151 (executor 1) (100/100)
17/07/26 00:21:15 INFO DAGScheduler: ResultStage 0 (reduce at SparkPi.scala:38) finished in 8.648 s
17/07/26 00:21:15 INFO TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
17/07/26 00:21:15 INFO DAGScheduler: Job 0 finished: reduce at SparkPi.scala:38, took 9.823011 s
Pi is roughly 3.1421011142101114
17/07/26 00:21:15 INFO SparkUI: Stopped Spark web UI at http://172.16.148.150:4040
17/07/26 00:21:15 INFO StandaloneSchedulerBackend: Shutting down all executors
17/07/26 00:21:15 INFO CoarseGrainedSchedulerBackend$DriverEndpoint: Asking each executor to shut down
17/07/26 00:21:15 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
17/07/26 00:21:15 INFO MemoryStore: MemoryStore cleared
17/07/26 00:21:15 INFO BlockManager: BlockManager stopped
17/07/26 00:21:15 INFO BlockManagerMaster: BlockManagerMaster stopped
17/07/26 00:21:15 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
17/07/26 00:21:15 INFO SparkContext: Successfully stopped SparkContext
17/07/26 00:21:15 INFO ShutdownHookManager: Shutdown hook called
17/07/26 00:21:15 INFO ShutdownHookManager: Deleting directory /tmp/spark-8d310137-d847-4480-bb69-3b80b4c52585
```

8) 启动 spark shell

```
/opt/module/spark/bin/spark-shell \
--master spark://hadoop101:7077 \
--executor-memory 1g \
--total-executor-cores 2
```

参数: --master spark://hadoop102:7077 指定要连接的集群的 master

执行 WordCount 程序

```
scala>sc.textFile("input").flatMap(_.split("
")).map((_,1)).reduceByKey(_+_).collect
res0: Array[(String, Int)] = Array((hadoop,6), (oozie,3), (spark,3), (hive,3), (atguigu,3), (hbase,6))

scala>
```

2.4.3 JobHistoryServer 配置

1) 修改 spark-default.conf.template 名称

```
[atguigu@hadoop102 conf]$ mv spark-defaults.conf.template spark-defaults.conf
```

2) 修改 spark-default.conf 文件, 开启 Log:

```
[atguigu@hadoop102 conf]$ vi spark-defaults.conf
spark.eventLog.enabled      true
```

更多 Java -大数据 -前端 -python 人工智能资料下载, 可百度访问: 尚硅谷官网

```
spark.eventLog.dir          hdfs://hadoop102:9000/directory
```

注意：HDFS 上的目录需要提前存在。

3) 修改 spark-env.sh 文件，添加如下配置：

```
[atguigu@hadoop102 conf]$ vi spark-env.sh

export SPARK_HISTORY_OPTS="-Dspark.history.ui.port=18080
-Dspark.history.retainedApplications=30
-Dspark.history.fs.logDirectory=hdfs://hadoop101:9000/directory"
```

参数描述：

spark.eventLog.dir: Application 在运行过程中所有的信息均记录在该属性指定的路径下

spark.history.ui.port=18080 WEBUI 访问的端口号为 18080

spark.history.fs.logDirectory=hdfs://hadoop102:9000/directory 配置了该属性后，在 start-history-server.sh 时就无需再显式的指定路径，Spark History Server 页面只展示该指定路径下的信息

spark.history.retainedApplications=30 指定保存 Application 历史记录个数，如果超过这个值，旧的应用程序信息将被删除，这个是内存中的应用数，而不是页面上显示的应用数。

4) 分发配置文件

```
[atguigu@hadoop102 conf]$ xsync spark-defaults.conf
[atguigu@hadoop102 conf]$ xsync spark-env.sh
```

5) 启动历史服务

```
[atguigu@hadoop102 spark]$ sbin/start-history-server.sh
```

6) 再次执行任务

```
[atguigu@hadoop102 spark]$ bin/spark-submit \
--class org.apache.spark.examples.SparkPi \
--master spark://hadoop101:7077 \
--executor-memory 1G \
--total-executor-cores 2 \
./examples/jars/spark-examples_2.11-2.1.1.jar \
100
```

7) 查看历史服务

hadoop102:18080

History Server

Event log directory: hdfs://hadoop102:9000/directory

Last updated: 2018-10-16, 上午10:50:40

Search:

App ID	App Name	Started	Completed	Duration	Spark User	Last Updated	Event Log
application_1539651441404_0011	Spark Pi	2018-10-16 02:41:13	2018-10-16 02:41:37	24 s	atguigu	2018-10-16 02:41:37	Download
application_1539651441404_0010	WordCount	2018-10-16 02:35:17	2018-10-16 02:35:39	22 s	atguigu	2018-10-16 02:35:39	Download
local-1539656878416	WordCount	2018-10-16 02:27:56	2018-10-16 02:28:01	5 s	atguigu	2018-10-16 02:28:01	Download
local-1539656869974	Spark shell	2018-10-16 02:27:35	2018-10-16 02:27:48	12 s	atguigu	2018-10-16 02:27:48	Download
application_1539651441404_0009	WordCount	2018-10-16 02:15:56	2018-10-16 02:16:20	24 s	atguigu	2018-10-16 02:16:20	Download
application_1539651441404_0008	WordCount	2018-10-16 02:11:59	2018-10-16 02:12:23	24 s	atguigu	2018-10-16 02:12:23	Download
application_1539651441404_0007	WordCount	2018-10-16 02:05:24	2018-10-16 02:05:50	26 s	atguigu	2018-10-16 02:05:50	Download
local-1539654871705	WordCount	2018-10-16 01:54:29	2018-10-16 01:54:34	5 s	atguigu	2018-10-16 01:54:34	Download
application_1539651441404_0006	WordCount	2018-10-16 01:45:03	2018-10-16 01:45:30	27 s	atguigu	2018-10-16 01:45:30	Download
application_1539651441404_0005	WordCount	2018-10-16 01:43:11	2018-10-16 01:43:35	25 s	atguigu	2018-10-16 01:43:35	Download
application_1539654893667_0003	Spark shell	2018-10-15 06:45:53	2018-10-15 07:09:15	22 min	atguigu	2018-10-15 07:09:15	Download

Showing 1 to 11 of 11 entries

[Show incomplete applications](#)

2.4.4 HA 配置

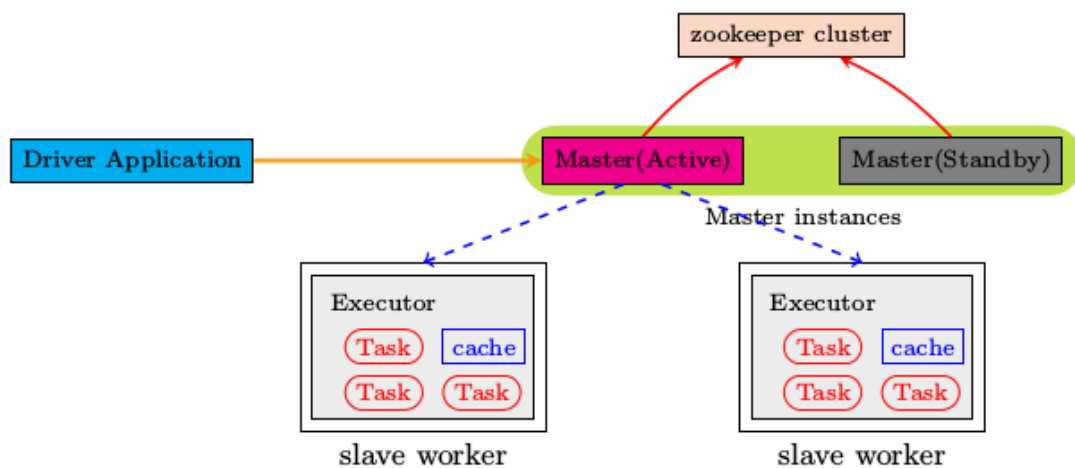


图 1 HA 架构图

- 1) zookeeper 正常安装并启动
- 2) 修改 spark-env.sh 文件添加如下配置：

```
[atguigu@hadoop102 conf]$ vi spark-env.sh
```

注释掉如下内容：

```
#SPARK_MASTER_HOST=hadoop102
#SPARK_MASTER_PORT=7077
```

添加上如下内容：

```
export SPARK_DAEMON_JAVA_OPTS="
-Dspark.deploy.recoveryMode=ZOOKEEPER
-Dspark.deploy.zookeeper.url=hadoop101,hadoop102,hadoop103
-Dspark.deploy.zookeeper.dir=/spark"
```

- 3) 分发配置文件

```
[atguigu@hadoop102 conf]$ xsync spark-env.sh
```

- 4) 在 hadoop102 上启动全部节点

```
[atguigu@hadoop102 spark]$ sbin/start-all.sh
```

- 5) 在 hadoop103 上单独启动 master 节点

```
[atguigu@hadoop103 spark]$ sbin/start-master.sh
```

更多 Java - 大数据 - 前端 - python 人工智能资料下载，可百度访问：尚硅谷官网

6) spark HA 集群访问

```
/opt/module/spark/bin/spark-shell \  
--master spark://hadoop101:7077,hadoop102:7077 \  
--executor-memory 2g \  
--total-executor-cores 2
```

2.5 Yarn 模式

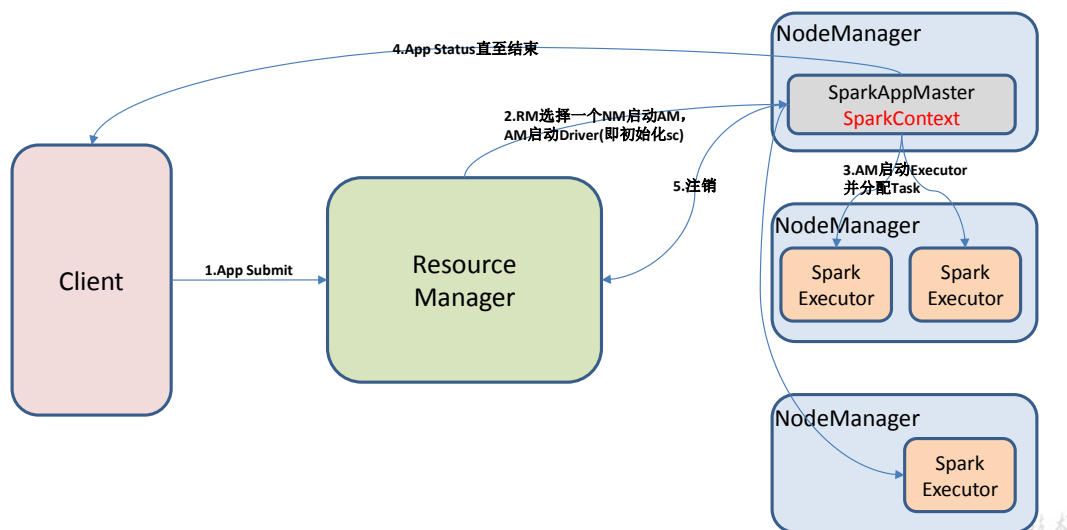
2.5.1 概述 Spark 客户端直接连接 Yarn，不需要额外构建 Spark 集群。有 **yarn-client** 和 **yarn-cluster** 两种模式，**主要区别在于：Driver 程序的运行节点。**

yarn-client: Driver 程序运行在客户端，适用于交互、调试，希望立即看到 app 的输出

yarn-cluster: Driver 程序运行在由 RM (ResourceManager) 启动的 AP (AppMaster)
适用于生产环境。



Yarn运行模式介绍



2.5.2 安装使用

1) 修改 **hadoop** 配置文件 **yarn-site.xml**, 添加如下内容:

```
[atguigu@hadoop102 hadoop]$ vi yarn-site.xml  
<!--是否启动一个线程检查每个任务正使用的物理内存量, 如果任务超出分配值,  
则直接将其杀掉, 默认是 true -->  
<property>  
  <name>yarn.nodemanager.pmem-check-enabled</name>  
  <value>>false</value>  
</property>  
<!--是否启动一个线程检查每个任务正使用的虚拟内存量, 如果任务超出分配值,  
则直接将其杀掉, 默认是 true -->  
<property>  
  <name>yarn.nodemanager.vmem-check-enabled</name>
```

更多 Java - 大数据 - 前端 - python 人工智能资料下载, 可百度访问: 尚硅谷官网

```
<value>false</value>
</property>
```

2) 修改 spark-env.sh, 添加如下配置:

```
[atguigu@hadoop102 conf]$ vi spark-env.sh

YARN_CONF_DIR=/opt/module/hadoop-2.7.2/etc/hadoop
```

3) 分发配置文件

```
[atguigu@hadoop102 conf]$ xsync /opt/module/hadoop-2.7.2/etc/hadoop/yarn-site.xml
[atguigu@hadoop102 conf]$ xsync spark-env.sh
```

4) 执行一个程序

```
[atguigu@hadoop102 spark]$ bin/spark-submit \
--class org.apache.spark.examples.SparkPi \
--master yarn \
--deploy-mode client \
./examples/jars/spark-examples_2.11-2.1.1.jar \
100
```

注意: 在提交任务之前需启动 HDFS 以及 YARN 集群。

2.5.3 日志查看

1) 修改配置文件 spark-defaults.conf

添加如下内容:

```
spark.yarn.historyServer.address=hadoop101:18080
spark.history.ui.port=4000
```

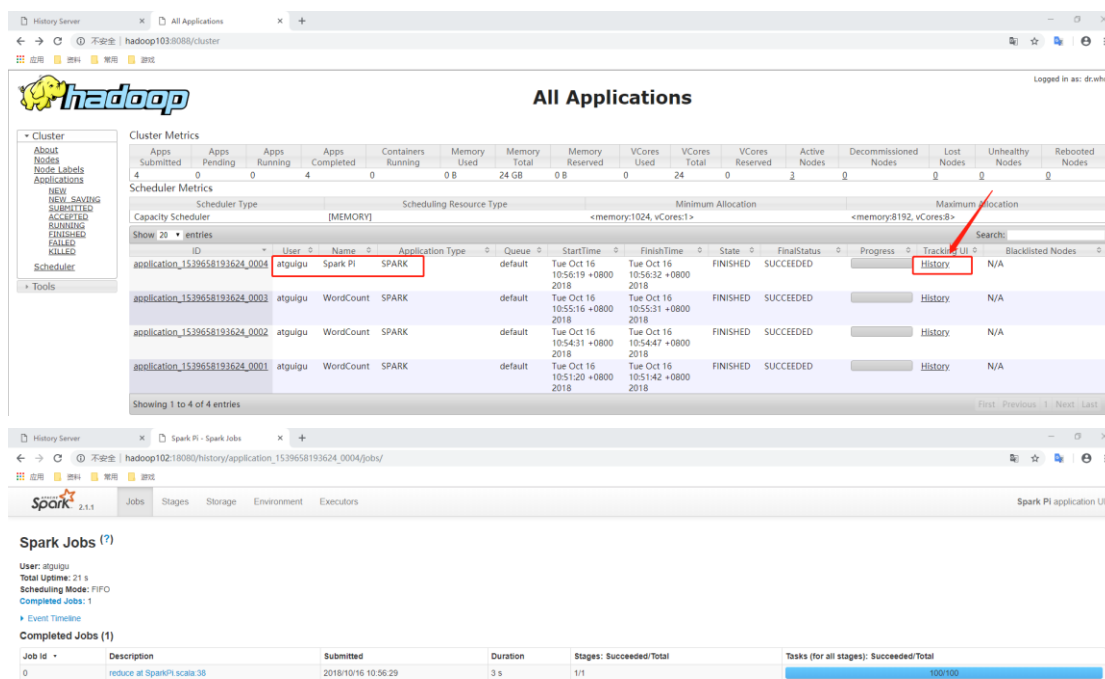
2) 重启 spark 历史服务

```
[atguigu@hadoop102 spark]$ sbin/stop-history-server.sh
stopping org.apache.spark.deploy.history.HistoryServer
[atguigu@hadoop102 spark]$ sbin/start-history-server.sh
starting org.apache.spark.deploy.history.HistoryServer, logging to
/opt/module/spark/logs/spark-atguigu-
org.apache.spark.deploy.history.HistoryServer-1-hadoop102.out
```

3) 提交任务到 Yarn 执行

```
[atguigu@hadoop102 spark]$ bin/spark-submit \
--class org.apache.spark.examples.SparkPi \
--master yarn \
--deploy-mode client \
./examples/jars/spark-examples_2.11-2.1.1.jar \
100
```

4) Web 页面查看日志



The image shows two screenshots from a Hadoop cluster management interface. The top screenshot, titled 'All Applications', displays a table of running applications. A red box highlights the 'History' link for a specific application. The bottom screenshot, titled 'Spark Jobs', shows the details of a Spark job, including its status and progress.

2.6 Mesos 模式

Spark 客户端直接连接 Mesos; 不需要额外构建 Spark 集群。国内应用比较少, 更多的是运用 yarn 调度。

2.7 几种模式对比

模式	Spark 安装机器数	需启动的进程	所有者
Local	1	无	Spark
Standalone	3	Master 及 Worker	Spark
Yarn	1	Yarn 及 HDFS	Hadoop

第 3 章 案例实操

Spark Shell 仅在测试和验证我们的程序时使用的较多, 在生产环境中, 通常会在 IDE 中编制程序, 然后打成 jar 包, 然后提交到集群, 最常用的是创建一个 Maven 项目, 利用 Maven 来管理 jar 包的依赖。

3.1 编写 WordCount 程序

1) 创建一个 Maven 项目 WordCount 并导入依赖

```
<dependencies>
  <dependency>
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-core_2.11</artifactId>
```

更多 Java - 大数据 - 前端 - python 人工智能资料下载, 可百度访问: 尚硅谷官网

```
<version>2.1.1</version>
</dependency>
</dependencies>
<build>
  <finalName>WordCount</finalName>
  <plugins>
    <plugin>
      <groupId>net.alchim31.maven</groupId>
      <artifactId>scala-maven-plugin</artifactId>
      <version>3.2.2</version>
      <executions>
        <execution>
          <goals>
            <goal>compile</goal>
            <goal>testCompile</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
```

2) 编写代码

```
package com.atguigu

import org.apache.spark.{SparkConf, SparkContext}

object WordCount{

  def main(args: Array[String]): Unit = {

    //1.创建 SparkConf 并设置 App 名称
    val conf = new SparkConf().setAppName("WC")

    //2.创建 SparkContext, 该对象是提交 Spark App 的入口
    val sc = new SparkContext(conf)

    //3.使用 sc 创建 RDD 并执行相应的 transformation 和 action
    sc.textFile(args(0)).flatMap(_.split(" ")).map((_, 1)).reduceByKey(_+_).sortBy(_._2, false).saveAsTextFile(args(1))

    //4.关闭连接
    sc.stop()
  }
}
```

3) 打包插件

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-assembly-plugin</artifactId>
  <version>3.0.0</version>
  <configuration>
    <archive>
      <manifest>
        <mainClass>WordCount</mainClass>
      </manifest>
    </archive>
```

更多 Java - 大数据 - 前端 - python 人工智能资料下载，可百度访问：尚硅谷官网


```

        <descriptorRefs>
            <descriptorRef>jar-with-
dependencies</descriptorRef>
        </descriptorRefs>
    </configuration>
    <executions>
        <execution>
            <id>make-assembly</id>
            <phase>package</phase>
            <goals>
                <goal>single</goal>
            </goals>
        </execution>
    </executions>
</plugin>

```

4) 打包到集群测试

```

bin/spark-submit \
--class WordCount \
--master spark://hadoop102:7077 \
WordCount.jar \
/word.txt \
/out

```

3.2 本地调试

本地 Spark 程序调试需要使用 local 提交模式，即将本机当做运行环境，Master 和 Worker 都为本地。运行时直接加断点调试即可。如下：

创建 SparkConf 的时候设置额外属性，表明本地执行：

```
val conf = new SparkConf().setAppName("WC").setMaster("local[*]")
```

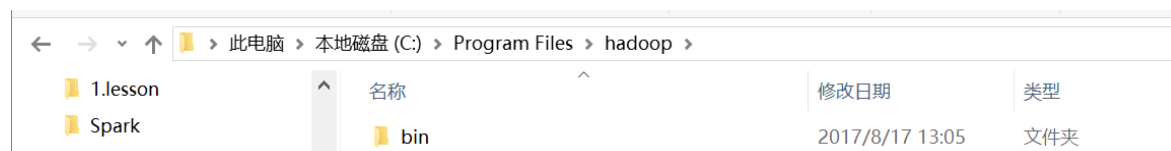
如果本机操作系统是 windows，如果在程序中使用了 hadoop 相关的东西，比如写入文件到 HDFS，则会遇到如下异常：

```

2017-09-14 16:08:34,907 ERROR --- [main] org.apache.hadoop.util.Shell(line:303) : Failed to locate the winutils binary in the hadoop binary path
java.io.IOException: Could not locate executable null\bin\winutils.exe in the Hadoop binaries.
    at org.apache.hadoop.util.Shell.getQualifiedBinPath(Shell.java:278)
    at org.apache.hadoop.util.Shell.getWinUtilsPath(Shell.java:300)
    at org.apache.hadoop.util.Shell.<clinit>(Shell.java:293)
    at org.apache.hadoop.util.StringUtils.<clinit>(StringUtils.java:76)
    at org.apache.hadoop.conf.Configuration.getTrimmedStrings(Configuration.java:1546)
    at org.apache.hadoop.hdfs.DFSClient.<init>(DFSClient.java:519)
    at org.apache.hadoop.hdfs.DFSClient.<init>(DFSClient.java:453)
    at org.apache.hadoop.hdfs.DistributedFileSystem.initialize(DistributedFileSystem.java:136)
    at org.apache.hadoop.fs.FileSystem.createFileSystem(FileSystem.java:2433)

```

出现这个问题的原因，并不是程序的错误，而是用到了 hadoop 相关的服务，解决办法是将附加载里面的 hadoop-common-bin-2.7.3-x64.zip 解压到任意目录。



更多 Java - 大数据 - 前端 - python 人工智能资料下载，可百度访问：尚硅谷官网

在 IDEA 中配置 Run Configuration，添加 HADOOP_HOME 变量

