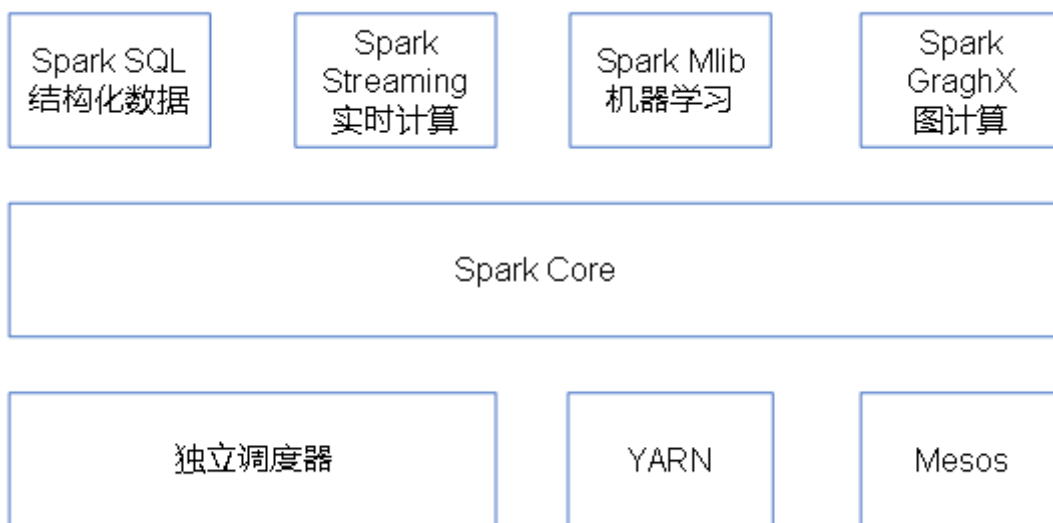


先理解，记忆是为了强化理解，不然它没有意义

# Spark基础入门和环境安装

## Spark概述

Spark是一种**基于内存**的快速、通用、可扩展的大数据分析引擎。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、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。



- 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；

# 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。

## Spark集群角色

### Master和Worker

#### Master

Spark特有资源调度系统的Leader。掌管着整个集群的资源信息，类似于Yarn框架中的ResourceManager  
主要功能：

- （1）监听worker，看worker是否正常工作；
- （2）Master对Worker、Application等的管理（接收Worker的注册并管理所有的worker，接收Client提交的application，调度等待的Application并向Worker提交）。

#### Worker

Spark特有资源调度系统的Slave，有多个。每个Slave掌管着所在节点的资源信息，类似于Yarn框架中NodeManager，主要功能：

- （1）通过RegisterWorker注册到Master；
- （2）定时发送心跳给Master；
- （3）根据Master发送的Application配置进程环境，并启动ExecutorBackend（执行Task所需的临时进程）

### Driver和Executor

### Driver（驱动器）

Spark的驱动器是执行开发程序中的main方法的线程。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）将用户程序转化为作业（Job）；
- （2）在Executor之间调度任务（Task）；
- （3）跟踪Executor的执行情况；
- （4）通过UI展示查询运行情况。

### Executor（执行器）

Spark Executor是一个工作节点，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，并将状态信息返回给驱动器程序；
- （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内的，因此任务可以在运行时充分利用缓存数据加速运算。

总结：Master和Worker是Spark的守护进程，即Spark在特定模式下正常运行所必须的进程。Driver和Executor是临时程序，当有具体任务提交到Spark集群才会开启的程序。

## Local模式

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。它可以通过以下几种方式设置 Master。

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

local[k]：指定使用K个Coe来运行计算，比如local[4]：就是运行4个core来执行；

local[\*]：这种模式直接使用最大Core数。

## Local模式安装

### 1) 上传并解压spark安装包

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

### 2) 官方求PI案例

```
[zhutiansama@hadoop102 spark]$ bin/spark-submit \ 提交任务到集群的命令
--class org.apache.spark.examples.SparkPi \ 主类
--executor-memory 1G \ 运行内存
--total-executor-cores 2 \ cpu核数
./examples/jars/spark-examples_2.11-2.1.1.jar \ jar包路径
50 迭代次数
```

提交任务语法如下：

```
bin/spark-submit \
--class <main-class>
```

```
--master <master-url> \  
--deploy-mode <deploy-mode> \  
--conf <key>=<value> \  
... # other options  
<application-jar> \  
[application-arguments]
```

参数说明

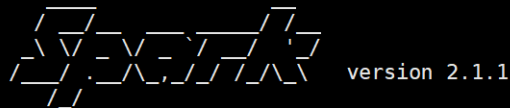
--master 指定Master的地址;  
--class: 你的应用的启动类 (如 `org.apache.spark.examples.SparkPi`);  
--deploy-mode: 是否发布你的驱动到worker节点(cluster) 或者作为一个本地客户端 (client) (default: client);  
--conf: 任意的Spark配置属性, 格式key=value. 如果值包含空格, 可以加引号“key=value”;  
application-jar: 打包好的应用jar, 包含依赖. 这个URL在集群中全局可见. 比如hdfs:// 共享存储系统, 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cpu核数为2个。

## 进入spark-shell命令

```
bin/spark-shell
```

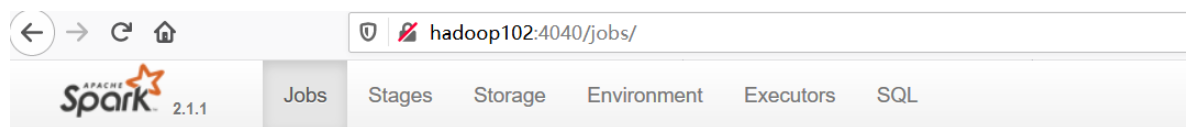
```
scala> 这里控制台已经帮你初始化了一些环境变量比如SparkContext和SparkSession
```

```
Spark context Web UI available at http://192.168.159.102:4040  
Spark context available as 'sc' (master = local[*], app id = local-1592547404943).  
Spark session available as 'spark'.  
Welcome to
```



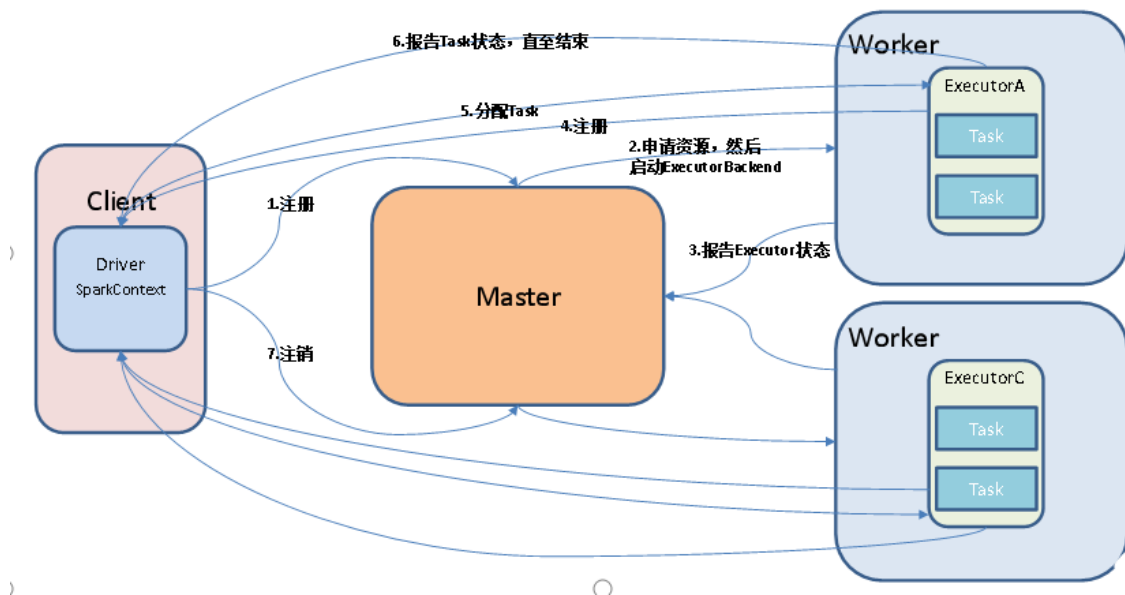
## 打开Spark监控页面

默认端口4040, hadoop102是我在hosts文件中设置的域名映射



## Standalone模式

构建一个由Master+Slave构成的Spark集群, 使Spark程序运行在集群中, 且有Cluster与Client模式两种。主要区别在于: Driver程序的运行节点。



## Standalone模式安装

- 1) 进入spark安装目录下的conf文件夹

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

- 2) 修改配置文件名称

```
[zhutiansama@hadoop102 conf]$ mv slaves.template slaves
```

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

- 3) 修改slave文件, 添加work节点:

```
[zhutiansama@hadoop102 conf]$ vim slaves
hadoop102
hadoop103
hadoop104
```

- 4) 修改spark-env.sh文件, 添加如下配置:

```
[zhutiansama@hadoop102 conf]$ vim spark-env.sh
SPARK_MASTER_HOST=hadoop102
SPARK_MASTER_PORT=7077
```

- 5) 分发spark包

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

- 6) 启动

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

```
[zhutian@hadoop102 spark-2.1.1]$ j
----- hadoop102 -----
3042 Worker
2936 Master
3310 Jps
----- hadoop103 -----
2915 Worker
3020 Jps
----- hadoop104 -----
3028 Jps
2923 Worker
```

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=/opt/module/jdk1.8.0_144
```

## 7) 官方求PI案例

```
[zhutiansama@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
```

# JobHistoryServer配置

## 1) 修改spark-default.conf.template名称

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

## 2) 修改spark-default.conf文件，开启Log：

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

```
[zhutiansama@hadoop102 conf]$ vi spark-env.sh
export SPARK_HISTORY_OPTS="-Dspark.history.ui.port=18080
-Dspark.history.retainedApplications=30
-Dspark.history.fs.logDirectory=hdfs://hadoop102:9000/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。注意：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

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

#### 5) 启动历史服务

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

```
[zhutian@hadoop102 spark-2.1.1]$ jps -l
3601 org.apache.hadoop.hdfs.server.datanode.DataNode
4290 org.apache.spark.deploy.history.HistoryServer
3458 org.apache.hadoop.hdfs.server.namenode.NameNode
3042 org.apache.spark.deploy.worker.Worker
2936 org.apache.spark.deploy.master.Master
4344 sun.tools.jps.Jps
3868 org.apache.hadoop.yarn.server.nodemanager.NodeManager
3934 org.apache.hadoop.mapreduce.v2.hs.JobHistoryServer
[zhutian@hadoop102 spark-2.1.1]$
```

#### 6) 再次执行任务

```
[zhutiansama@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
```

#### 7) 查看历史服务

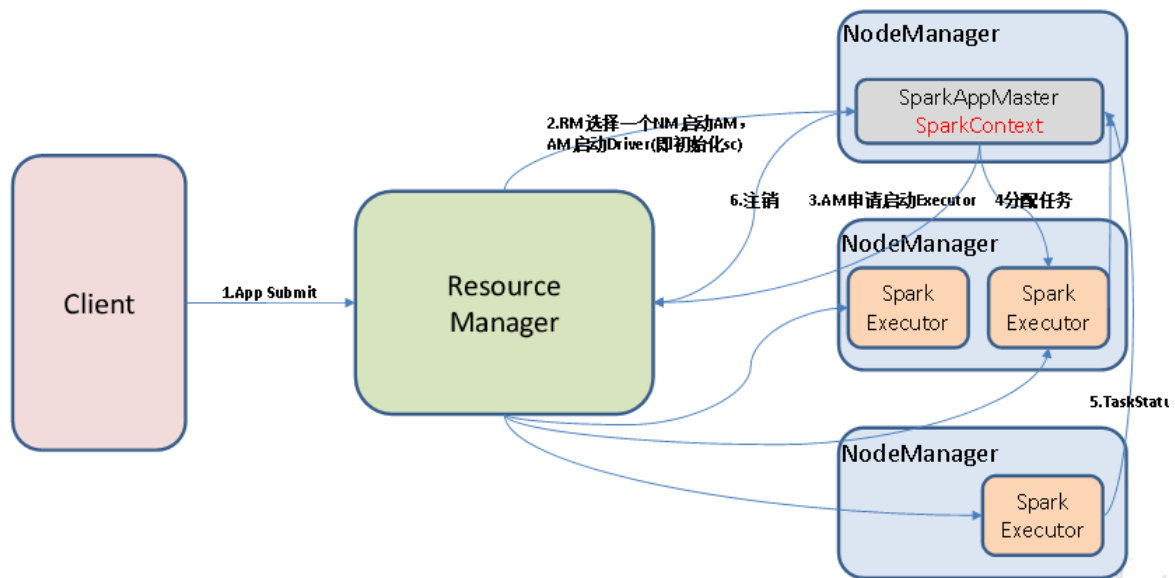
```
hadoop102:18080
```

## Yarn模式【生产环境】

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（Resource Manager）启动的AM（AppMaster）适用于生产环境。



## Yarn模式安装

1) 修改hadoop配置文件yarn-site.xml,添加如下内容

```
<!--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>
        <value>>false</value>
    </property>
```

2) 修改spark-env.sh, 添加如下配置

```
[zhutiansama@hadoop102 conf]$ vi spark-env.sh
YARN_CONF_DIR=/opt/module/hadoop-2.7.2/etc/hadoop
```

3) 分发配置文件

```
[zhutiansama@hadoop102 conf]$ xsync /opt/module/hadoop-2.7.2/etc/hadoop/yarn-site.xml
```

4) 执行一个程序

```
[zhutiansama@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。



# 重新配置日志

## 1) 修改配置文件spark-defaults.conf

```
spark.yarn.historyServer.address=hadoop102:18080
spark.history.ui.port=18080
```

## 2) 重启Spark历史服务

```
[zhutiansama@hadoop102 spark]$ sbin/stop-history-server.sh
stopping org.apache.spark.deploy.history.HistoryServer

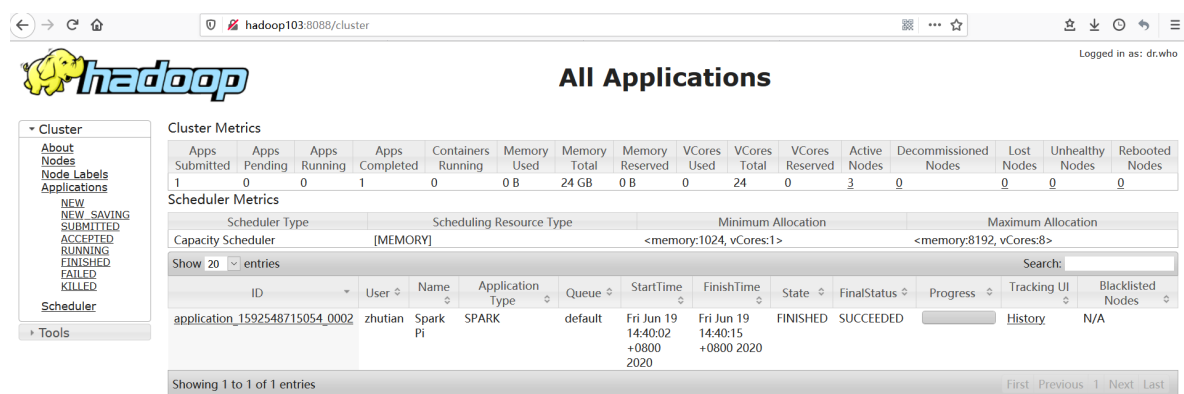
[zhutiansama@hadoop102 spark]$ sbin/start-history-server.sh
starting org.apache.spark.deploy.history.HistoryServer, logging to
/opt/module/spark/logs/spark-zhutiansama-
org.apache.spark.deploy.history.HistoryServer-1-hadoop102.out
```

## 3) 提交任务到Yarn执行

```
[zhutiansama@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

直接到yarn上看  
hadoop103:8088



Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
1	0	0	1	0	0 B	24 GB	0 B	0	24	0	3	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>	<memory:8192, vCores:8>

Showing 20 entries

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI	Blacklisted Nodes
application_1592548715054_0002	zhutian	Spark Pi	SPARK	default	Fri Jun 19 14:40:02 +0800 2020	Fri Jun 19 14:40:15 +0800 2020	FINISHED	SUCCEEDED		History	N/A

Showing 1 to 1 of 1 entries

# 模式对比

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