# Spark详解

#### Spark简介

##### 1.1 什么是Spark

要理解大数据，就需要理解大数据相关的查询、处理、机器学习、图形计算和统计分析，Spark是新一代轻量级大数据快速处理平台。

说起大数据，大家最先想到的是开源项目hadoop，hadoop解决了大多数批处理工作负载问题，成为大数据时代企业的首选技术，但是hadoop依然存在他的一些限制，对于一些工作负载不是最优的选择

* 缺少对迭代的支持
* 中间数据需要输出到硬盘存储，产生了较高的延迟

探其究竟，MapReduce设计上的约束比较适合处理离线数据，在实时查询和迭代计算上存在较大的不足。Spark作为Apache的顶级开源项目，是一个快速，通用的大规模数据处理引擎，和Hadoop的MapReduce计算框架类似，但是相对于MapReduce，Spark凭借其可伸缩、基于内存计算等特点，以及可以直接读写Hadoop上任何格式数据的优势，在进行批处理的时候更加高效，延迟更低。Spark已经成为轻量级大数据快速处理的统一平台，各种不同的应用，如实时流处理、机器学习、交互式查询等，都可以通过Spark建立在不同的存储和运行系统上。

###### Spark框架架构

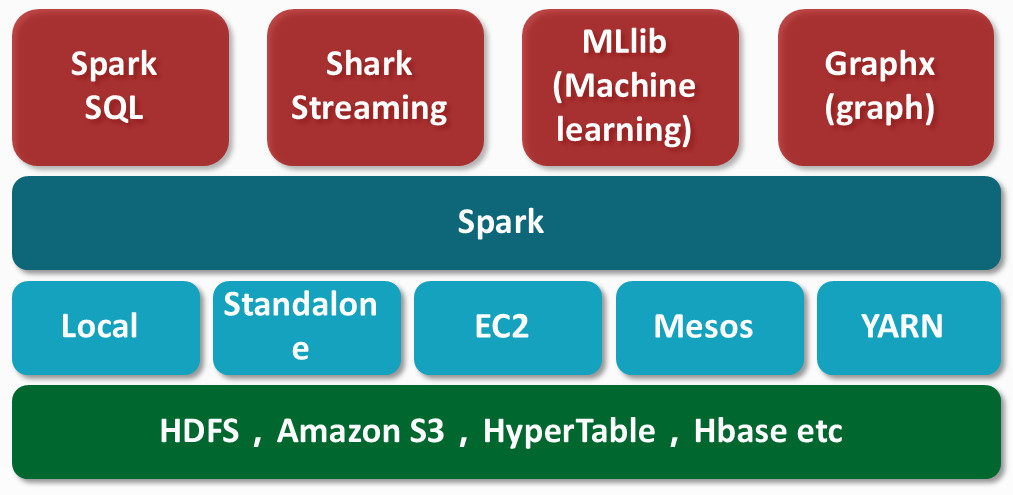


图1.1

图1.1是spark集群的整个生态圈，最底层的是分布式存储系统，或者其他格式的存储系统如HBase；资源管理可以采用那个Mesos、YARN集群资源管理模式或者Spark自带的standalone（独立运行）模式，以及Local（本地）模式。Spark为最上层的多种应用提供服务。Spark SQL提供SQL查询服务，性能比Hive快3~50倍；MLlib提供机器学习服务；GraphX提供图形计算服务；Spark Streaming将流式计算分解成一系列短小的批处理计算，并且提供高可靠和高吞吐量服务。他们都可以使用Spark核心API来处理问题，方法几乎通用，处理的数据也可以共享，不仅减少学习成本，而且使数据无缝集成，大大提高了灵活性。

###### Spark的特点

* **快速**：spark有先进的DAG执行引擎，支持循环数据流和内存计算。
* **易用**：支持Java、Scala、Python语言编写，超过80个高级运算符、编写并行应用程序更容易。
* **通用**：有基于spark的一系列高级工具，支持在一个应用中同时使用这些工具（Spark SQL、Streaming、MLlib、GraphX）。
* **有效集成Hadoop**：Spark可以指定Hadoop，YARN版本来编译出合适的发行版本，有多种运行模式，可以在多种存储系统中读取数据。

###### Spark应用场景

Spark使用内存分布式数据集，除了能提供交互式查询，还优化了迭代工作负载。在互联网领域，Spark在快速查询、实时日志采集处理、业务推荐、定制广告、用户图形计算等方面都有相应的应用。

* **快速查询系统：**基于日志数据的快速查询系统，利用其快速查询以及内存表的优势，能够承担大部分日志数据的即时查询工作；性能方面比Hive快2~10倍，如果使用内存表，性能将比Hive快百倍。
* **实时日志采集处理：**通过Spark Streaming 试试进行业务日志采集，快速迭代处理，并进行综合分析，能够满足线上系统的分析要求。
* **业务推荐系统：**能够将业务推荐系统从小时和天级别的模型训练转变为分钟级别的模型训练，有效优化相关排名、个性化推荐以及热点点击分析等。
* **定制广告系统：**实现数据实时采集、算法实时训练、系统实时预测的全流程实时并行高维度算法，支持上亿的请求量处理；模拟广告投放计算效率高、延迟小，同MapReduce相比延迟至少降低一个数量级。
* **用户图计算：**利用GraphX可以解决很多生产问题，包括基于度分布的中枢节点发现、基于最大连通图的社区发现、基于三角形计数的关系衡量、基于随机游走的用户属性传播等。

##### Spark的一些重要扩展

* Spark SQL和DataFrame
* Spark Streaming
* Spark MLlib和ML
* GraphX
* SparkR

#### Spark的部署和运行

##### 2.1 部署准备

###### 2.1.1 工具准备

* **Spark安装包**，最新下载地址：<http://spark.apache.org/downloads.html>。

Spark提供源码压缩包和编译好的二进制文件压缩包，正常开发下载编译好的二进制文件压缩包即可，除非需要改动源码或者编译源码生成相对应Hadoop版本的压缩包。

Maven地址：

groupId: org.apache.spark

artifactId: spark-core\_2.11

version: 2.1.0

* **Intellij IDEA**，下载地址：[http://www.jetbrains.com/idea/download/#section=windows](http://www.jetbrains.com/idea/download/%23section=windows)

IDEA是业界最好的java开发工具之一。

* **JDK**，必须是1.7及以上的版本
* **Scala安装包**，下载地址：<http://www.scala-lang.org/download/>

###### 2.1.2 集群部署概述

目前与Spark相关的集群环境，也就是集群管理器，主要包括：Spark自带的Standalone资源管理器、Mesos集群管理和Hadoop YARN集群管理器。

##### 2.2 Spark部署

###### 2.2.1 Local模式部署

local模式部署比较简单，可用于检测Spark安装包是否安装成功。需要配置Java环境变量和设置主节点。

主节点设置方式：

1. 进入spark主程序conf目录
2. 以spark-env.sh.template文件为模板创建spark-env.sh文件

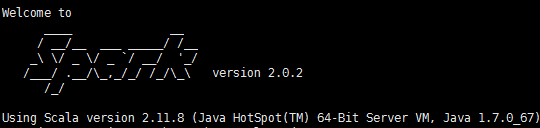
# cp spark-env.sh.template spark-env.sh

1. 修改spark-env.sh配置文件：

export SPARK\_MASTER\_IP=s12183 # 你的master 主机名或者IP

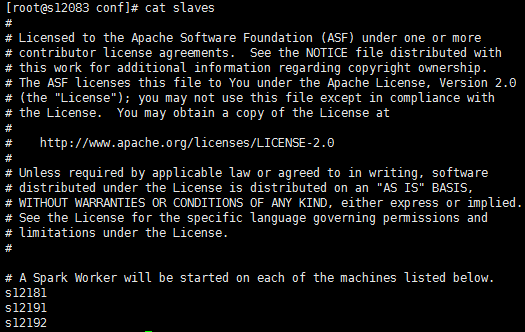
export JAVA\_HOME=$YOUR\_JAVA\_HOME # 你的JDK安装路径

1. 版本验证，安装完Spark并配置环境变量之后，任意目录下运行spark-shell命令即可进入Spark命令行模式，此时可以看到当前的spark版本号



###### 2.2.2 Standalone模式部署

1. 修改spark-env.sh配置文件。参考2.2.1 local模式部署
2. 在spark目录下创建一个名为conf/slaves的文件，可以根据slaves.template文件创建。该文件包含所有将要启动Spark Workers 的机器的hostname（主机名），每行一个。



1. 发送配置文件spark-env.sh和slaves文件到所有worker节点，以s12181为例

scp -r /opt/neu/spark/conf/spark-env.sh neu@s12181:/opt/neu/spark/conf/

scp -r /opt/neu/spark/conf/slaves neu@s12181:/opt/neu/spark/conf/

1. 配置Master无密钥登陆slaves节点
2. 安装openssh-server（Ubuntu12.04为例）

sudo apt-get install openssh-server

1. 建立SSH KEY（每台节点都执行一遍，确保每一台节点有.ssh文件夹）

ssh-keygen –t rsa –P “”

1. 在Master节点上启用SSH KEY（authorized\_keys权限644）

cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys

sudo /e2tc/init.d/ssh reload

1. 验证ssh的配置

ssh localhost

1. 将master节点的authorized\_keys发送到所有slaves节点，并登陆验证。

部署完毕后可以通过手动启动和脚本启动集群的master和worker。

1. 手动启动集群：

./sbin/start-master.sh

启动之后会打印出一个spark://HOST:PORT，可以通过该信息将Worker与Master连接，支持启动一个或更多的worker，通过下面的指令与master连接

./bin/spark-class org.apache.spark.deploy.worker.Worker park://IP:PORT

一旦启动一个worker节点，在master的webui中（默认http://localhost:8080），可以看到新增的worker节点，以及cpu数目、内存大小（减去1GB留给系统）在列表中呈现

1. 脚本启动集群

* sbin/start-master.sh - Starts a master instance on the machine the script is executed on.
* sbin/start-slaves.sh - Starts a slave instance on each machine specified in the conf/slaves file.
* sbin/start-slave.sh - Starts a slave instance on the machine the script is executed on.
* sbin/start-all.sh - Starts both a master and a number of slaves as described above.
* sbin/stop-master.sh - Stops the master that was started via the bin/start-master.sh script.
* sbin/stop-slaves.sh - Stops all slave instances on the machines specified in the conf/slaves file.
* sbin/stop-all.sh - Stops both the master and the slaves as described above.

执行启动脚本之后，执行jps命令可以查看当前的进程，如果是主节点，可以看到Matser进程，如果是子节点可以看到worker进程，这样就表示spark在服务器上配置完毕。

还可以通过修改sparkl-env.sh文件中的环境变量来进一步配置集群

|  |  |
| --- | --- |
| **Environment Variable** | **Meaning** |
| SPARK\_MASTER\_HOST | Bind the master to a specific hostname or IP address, for example a public one. |
| SPARK\_MASTER\_PORT | Start the master on a different port (default: 7077). |
| SPARK\_MASTER\_WEBUI\_PORT | Port for the master web UI (default: 8080). |
| SPARK\_MASTER\_OPTS | Configuration properties that apply only to the master in the form "-Dx=y" (default: none). See below for a list of possible options. |
| SPARK\_LOCAL\_DIRS | Directory to use for "scratch" space in Spark, including map output files and RDDs that get stored on disk. This should be on a fast, local disk in your system. It can also be a comma-separated list of multiple directories on different disks. |
| SPARK\_WORKER\_CORES | Total number of cores to allow Spark applications to use on the machine (default: all available cores). |
| SPARK\_WORKER\_MEMORY | Total amount of memory to allow Spark applications to use on the machine, e.g. 1000m, 2g (default: total memory minus 1 GB); note that each application's *individual* memory is configured using its spark.executor.memory property. |
| SPARK\_WORKER\_PORT | Start the Spark worker on a specific port (default: random). |
| SPARK\_WORKER\_WEBUI\_PORT | Port for the worker web UI (default: 8081). |
| SPARK\_WORKER\_DIR | Directory to run applications in, which will include both logs and scratch space (default: SPARK\_HOME/work). |
| SPARK\_WORKER\_OPTS | Configuration properties that apply only to the worker in the form "-Dx=y" (default: none). See below for a list of possible options. |
| SPARK\_DAEMON\_MEMORY | Memory to allocate to the Spark master and worker daemons themselves (default: 1g). |
| SPARK\_DAEMON\_JAVA\_OPTS | JVM options for the Spark master and worker daemons themselves in the form "-Dx=y" (default: none). |
| SPARK\_PUBLIC\_DNS | The public DNS name of the Spark master and workers (default: none). |

###### 2.2.3 YARN模式启动

1. 准备工作：

在配置spark之前，使用root账户登陆所有节点，cat /etc/hosts文件查看是否已经将所有节点已经配置完毕，即是否所有的”服务器IP 服务器名”都存在，如果没有配置完毕需要scp 将主节点的hosts文件发到子节点上。

然后配置环境变量 vim /etc/profile 或者 vim ~/.bash\_profile

export SPARK\_HOME=/opt/neu/spark

export PATH=$PATH:$SPARK\_HOME/bin: $SPARK\_HOME/sbin

export HADOOP\_CONF\_DIR=$HADOOP\_HOME/etc/hadoop

export YARN\_CONF\_DIR=$HADOOP\_HOME/etc/hadoop

最后在任意目录下执行 source /etc/profile命令让环境变量生效

1. Spark本身配置：

同standalone模式的a)、b)两个步骤，其余的步骤不需要。

最后将spark安装包、环境变量配置、hosts文件批量发送到各个节点所在的服务器，再批量执行source /etc/profile即可，按照先启动hadoop再启动spark的顺序启动。

##### 提交spark程序运行

spark-shell交互式命令，在spark上运行程序需要将Spark://IP:PROT传递给SparkContext，使用./bin/spark-shell命令会默认调用spark-env.sh的SPARK\_MASTER\_IP和SPARK\_MASTER\_PORT来给SparkContext赋默认值

也可以在命令行手动给SparkContext赋值，方式如下：

./bin/spark-shell –master spark://IP:PORT