银河麒麟服务器操作系统 V4

spark 软件适配手册

天津麒麟信息技术有限公司

2019 年 6 月

目 录

1 概述

1.1 系统概述

银河麒麟服务器操作系统主要面向军队综合电子信息系统、金融系统以及电力系统等国家关键行业的服务器应用领域，突出高安全性、高可用性、高效数据处理、虚拟化等关键技术优势，针对关键业务构建的丰富高效、安全可靠的功能特性，兼容适配长城、联想、浪潮、华为、曙光等国内主流厂商的服务器整机产品，以及达梦、金仓、神通、南大通用等主要国产数据库和中创、金蝶、东方通等国产中间件，满足虚拟化、云计算和大数据时代，服务器业务对操作系统在性能、安全性及可扩展性等方面的需求，是一款具有高安全、高可用、高可靠、高性能的自主可控服务器操作系统。

1.2 环境概述

服务器型号	长城信安擎天 DF720 服务器
CPU 类型	飞腾 2000+处理器
操作系统版本	Kylin-4.0.2-server-sp2-2000-19050910.Z1
内核版本	4.4.131
spark 版本	2.4.3

1.3 spark 软件简介

Apache Spark 是专为大规模数据处理而设计的快速通用的计算引擎。Spark 是 UC Berkeley AMP lab (加州大学伯克利分校的 AMP 实验室)所开源的类 Hadoop MapReduce 的通用并行框架，Spark，拥有 Hadoop MapReduce 所具有的优点；但不同于 MapReduce 的是——Job 中间输出结果可以保存在内存中，从而不再需要读写 HDFS，因此 Spark 能更好地适用于数据挖掘与机器学习等需要迭代的 MapReduce 的算法。

Spark 是一种与 Hadoop 相似的开源集群计算环境，但是两者之间还存在一些不同之处，这些有用的不同之处使 Spark 在某些工作负载方面表现得更加优越，换句话说，Spark 启用了内存分布数据集，除了能够提供交互式查询外，它还可以优化迭代工作负载。

Spark 是在 Scala 语言中实现的，它将 Scala 用作其应用程序框架。与 Hadoop 不同，Spark 和 Scala 能够紧密集成，其中的 Scala 可以像操作本地集合对象一样轻松地操作分布式数据集。尽管创建 Spark 是为了支持分布式数据集上的迭代作业，但是实际上它是对 Hadoop 的补充，可以在 Hadoop 文件系统

中并行运行。通过名为 Mesos 的第三方集群框架可以支持此行为。Spark 由加州大学伯克利分校 AMP 实验室 (Algorithms, Machines, and People Lab) 开发，可用来构建大型的、低延迟的数据分析应用程序。

1.4 Spark 的优点

1. 快速

与 Hadoop 的 MapReduce 相比，Spark 基于内存的运算要快 100 倍以上；而基于磁盘的运算也要快 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地处理数据流。

2. 简介易用

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应用。而且 Spark 支持交互式的 Python 和 Scala 的 Shell，这意味着可以非常方便的在这些 Shell 中使用 Spark 集群来验证解决问题的方法，而不是像以前一样，需要打包、上传集群、验证等。这对于原型开发非常重要。

3. 通用性

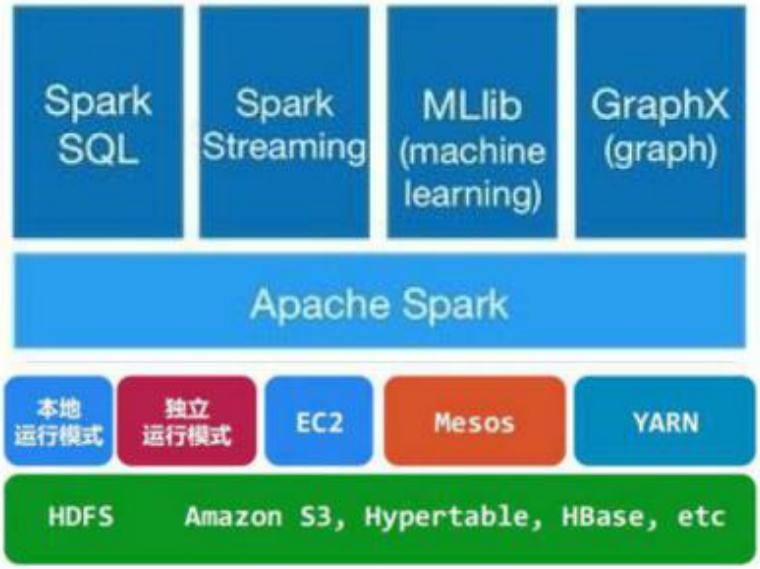
Spark 提供了统一的解决方案。Spark 可以用于批处理、交互式查询（通用 Spark SQL）、实时流处理（通过 Spark Streaming）、机器学习（通过 Spark MLlib）和图计算（通过 Spark GraphX）。这些不同类型的处理都可以在同一应用中无缝使用。Spark 统一的解决方案非常具有吸引力，毕竟任何公司都想用统一的平台处理问题，减少开发和维护的人力成本和部署平台的物理成本。当然还有，作为统一的解决方案，Spark 并没有以牺牲性能为代价。相反，在性能方面 Spark 具有巨大优势。

4. 可融合性

Spark 非常方便的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 和 Cassanda 等。这对于已部署 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 强大的处理能力。Spark 也可以不依赖第三方的资源管理器和调度器，它实现了 Standalone 作为其内置资源管理器和调度框架，这样进一步降低了 Spark 的使用门槛，使得所有人可以非常容易地部署和使用 Spark。此外 Spark 还提供了在 EC2 上部署 Standalone 的 Spark 集群的工具。

1.5 Spark 架构

spark 架构示意图所示：

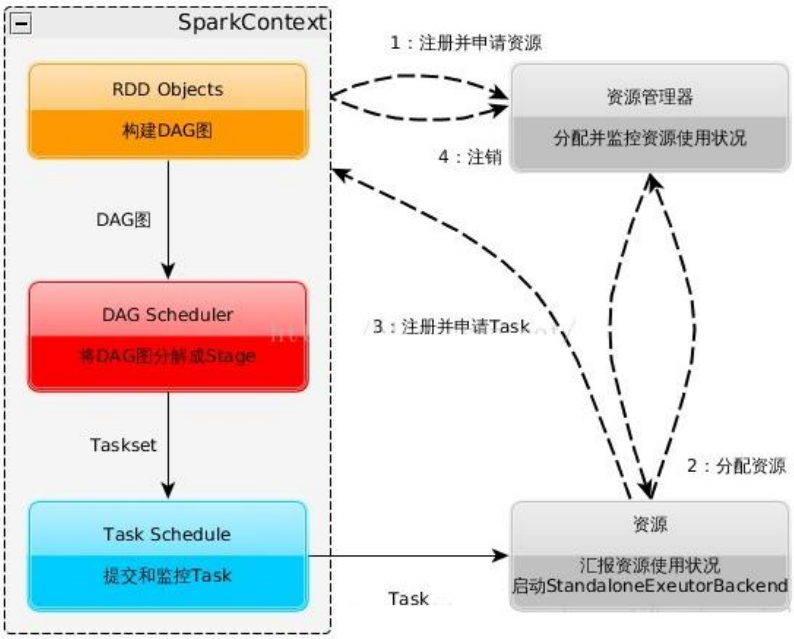


- Spark Core：包含 Spark 的基本功能；尤其是定义 RDD 的 API、操作以及这两者上的动作。其他 Spark 的库都是构建在 RDD 和 Spark Core 之上的。
- Spark SQL：提供通过 Apache Hive 的 SQL 变体 Hive 查询语言（HiveQL）与 Spark 进行交互的 API。每个数据库表被当做一个 RDD，Spark SQL 查询被转换为 Spark 操作。
- Spark Streaming：对实时数据流进行处理和控制。Spark Streaming 允许程序能够像普通 RDD 一样处理实时数据
- MLib：一个常用机器学习的算法库，算法被实现为对 RDD 的 Spark 操作。这个库包含可扩展的学习算法，比如分类、回归等需要对大量数据集进行迭代的操作
- GraphX：控制图、并行图操作和计算的一组算法和工具的集合。GraphX 扩展了 RDD API，包含控制图、创建子图、访问路径上所有顶点的操作
- Spark 架构采用了分布式计算中的 Master-Slave 模型，Master 是对应集群中的含有 Master 进程的节点，Slave 是集群中含有 Worker 进程的节点。Master 作为整个集群的控制器，负责整个集群的正常运行；Worker 相当于是计算节点，接收主节点命令与进行状态汇报；Executor 负责任务的执行；Client 作为用户的客户端负责提交应用，Driver 负责控制一个应用的执行。

1.6 Spark 运行流程及特点

1.6.1 Spark 运行流程

Spark 运行流程图如下：



1. 构建 Spark Application 的运行环境，启动 SparkContext；
2. SparkContext 向资源管理器（可以是 Standalone，Mesos，Yarn）申请运行 Executor 资源，并启动 StandaloneExecutorbackend；
3. Executor 向 SparkContext 申请 Task；
4. SparkContext 将应用程序分发给 Executor；
5. SparkContext 构建成 DAG 图，将 DAG 图分解成 Stage、将 Taskset 发送给 Task Scheduler，最后由 Task Scheduler 将 Task 发送给 Executor 运行；
6. Task 在 Executor 上运行，运行完释放所有资源。

1.6.2 Spark 运行特点

1. 每个 Application 获取专属的 executor 进程，该进程在 Application 期间一直驻留，并以多线程方式运行 Task。这种 Application 隔离机制是有优势的，无论是从调度角度看（每个 Driver 调度它自己的任务），还是从运行角度看（来自不同 Application 的 Task 运行在不同 JVM 中），当然这样意味着 Spark Application 不能跨应用程序共享数据，除非将数据写入外部存储系统。
2. Spark 与资源管理器无关，只要能够获取 Executor 进程，并能保持互相通信就可以了。

3. 提交 SparkContext 的 Client 应该靠近 Worker 节点（运行 Executor 的节点），最好是在同一个 Rack 里，因为 Spark Application 运行过程中 SparkContext 和 Executor 之间有大量的信息互换。
4. Task 采用了数据本地性和推测执行的优化机制。

2 spark 软件适配

1) 下载并解压 spark

```
$ wget http://mirror.bit.edu.cn/apache/spark/spark-2.4.3/spark-2.4.3-bin-hadoop2.7.tgz/
$ tar xvf spark-2.4.3-bin-hadoop2.7.tgz -C /usr/local/
```

2) 安装 scala

```
$ apt install scala
```

修改环境变量：

```
$ vim ~/.bashrc
```

添加如下内容：

```
export SCALA_HOME=/usr/share/scala-2.11
export PATH=$PATH:${SCALA_HOME}/bin
```

使环境变量生效：

```
$ source ~/.bashrc
```

3 运行测试

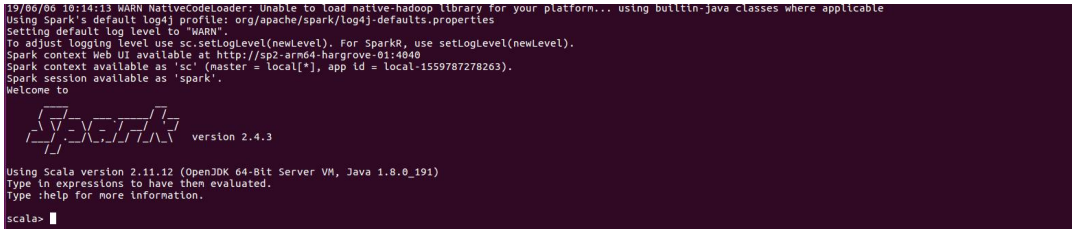
3.1 使用自带的 python shell 进行测试

```
$ cd /usr/local/spark-2.4.3-bin-hadoop2.7/bin/
$ ./pyspark
>>> lines = sc.textFile("/usr/local/spark-2.4.3-bin-hadoop2.7/README.md")
>>> lines.count()
>>> lines.first()
```

3.2 使用自带的 spark shell 进行测试

```
$ ./spark-shell
```

运行结果如下图：



查看当前节点运行情况：

```
$ cd /usr/local/spark-2.4.3-bin-hadoop2.7/sbin/
$ ./start-all.sh
```

访问:http://YOUR_IP:8082

3.3 执行 PageView 测试

在终端执行：

```
$ ./bin/run-example org.apache.spark.examples.streaming.clickstream.PageViewGenerator 44444 10
```

在另一个终端执行：

```
$ bin/run-example org.apache.spark.examples.streaming.clickstream.PageViewStream errorRatePerZipCode localhost 44444
```

执行过程如下：

```
19/06/06 10:31:36 WARN BlockManager: Block input-0-1559788296000 replicated to only 0 peer(s) instead of 1 peers
19/06/06 10:31:36 WARN RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
19/06/06 10:31:36 WARN BlockManager: Block input-0-1559788296200 replicated to only 0 peer(s) instead of 1 peers
19/06/06 10:31:36 WARN RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
19/06/06 10:31:36 WARN BlockManager: Block input-0-1559788296400 replicated to only 0 peer(s) instead of 1 peers
-------------------------------------------
Time: 1559788296000 ms
-------------------------------------------
94117: 0.026143791
94709: **0.062068965**

19/06/06 10:31:36 WARN RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
19/06/06 10:31:36 WARN BlockManager: Block input-0-1559788296600 replicated to only 0 peer(s) instead of 1 peers
19/06/06 10:31:37 WARN RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
19/06/06 10:31:37 WARN BlockManager: Block input-0-1559788296800 replicated to only 0 peer(s) instead of 1 peers
19/06/06 10:31:37 WARN RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
19/06/06 10:31:37 WARN BlockManager: Block input-0-1559788297000 replicated to only 0 peer(s) instead of 1 peers
19/06/06 10:31:37 WARN RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
19/06/06 10:31:37 WARN BlockManager: Block input-0-1559788297200 replicated to only 0 peer(s) instead of 1 peers
```