**Spark history-server详解**

2018-05-21 18:43:05 [爆发的~小宇宙](https://me.csdn.net/yu0_zhang0) 阅读数 5418  收藏 更多

分类专栏： [Spark](https://blog.csdn.net/yu0_zhang0/category_7442602.html) [spark学习专栏](https://blog.csdn.net/yu0_zhang0/category_9274553.html" \t "_blank)

版权声明：本文为博主原创文章，遵循[CC 4.0 BY-SA](http://creativecommons.org/licenses/by-sa/4.0/)版权协议，转载请附上原文出处链接和本声明。

本文链接：<https://blog.csdn.net/yu0_zhang0/article/details/80396080>

**概述**

这里作者和大家一起学习Spark 中的history-server，那他到底是什么呢？该如何去学习呢？  
我们可以带着下面几个问题进行详细的学习于思考：  
1. history-server产生背景  
2. history-server的作用  
3. 如何配置和使用  
4. history-server重要参数  
5. sc.stop的重要性  
6. REST API

**1 history-server产生背景**

[官方网站](http://spark.apache.org/docs/latest/monitoring.html)**大家可以看完这篇博客后再去官网看一看相信大家会有更多的收获。**这里我并不建议大家一上来就看官网，可以有了一定的学习后再对官网进行进一步的学习更深入的了解，这里仅仅是我的个人观点哈，大家找到符合自己的学习方式。

学习之前大家猜一猜这个history-server应该是在bin目录下还是sbin目录下呢？嘿嘿。不知道的同学去百度下这个的区别吧。。  
- 首先我们先启动一个spark-sell  
spark-shell --master local[2]

Spark context Web UI available at http://192.168.137.130:4040

Spark context available as 'sc' (master = local[2], app id = local-1524419194880).

Spark session available as 'spark'.

Welcome to

\_\_\_\_ \_\_

/ \_\_/\_\_ \_\_\_ \_\_\_\_\_/ /\_\_

\_\ \/ \_ \/ \_ `/ \_\_/ '\_/

/\_\_\_/ .\_\_/\\_,\_/\_/ /\_/\\_\ version 2.2.0

/\_/

Using Scala version 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0\_45)

Type in expressions to have them evaluated.

Type :help for more information.

日志中我们可以看到，可以在http://192.168.137.130:4040这个网址中查看我们的任务信息，每提交一个任务就可以通过上面的网址查看信息。那是不是就意味着我们只要通过该UI就可以了呢？  
答案肯定是不行的，同学们试试使用sc.stop看还能打开吗？这时候上面的网址就打不开了。同样的我们通过spark-submit提交应用程序，也是需要sc.stop的，总的来说，一个任务完成或者出现异常停止，我们应该怎么样去监控一个任务，因为监控对于我们来说是相当重要的，可以查看任务的执行状况，调优等等。这里就出现在一个问题，如何在任务结束后仍然能看到我们的任务等相关信息呢？该是我们的history-server登场的时候了，大家掌声欢迎！！！

**2 history-server的作用**

通过上面的介绍我相信大家应该知道了history-server的作用了把。我来给大家总结一下：通过相应的配置（**打开开关**），Spark应用程序在运行完应用程序之后，将应用程序的运行信息写入指定目录（**指定日志记录的目录**），而Spark history server可以将这些运行信息装载并以web的方式供用户浏览（**展示**）。  
**三部曲**：  
1. 打开开关  
2. 指定日志记录的目录  
3. 展示

**3 如何配置和使用**

* vi spark-defaults.conf

#开关

spark.eventLog.enabled true

#指定日志记录的目录

spark.eventLog.dir hdfs:*//192.138.137.130:9000/spark-logs*

hdfs dfs -mkdir /spark-logs

* vi spark-env.sh

#展示

SPARK\_HISTORY\_OPTS="-Dspark.history.fs.logDirectory=hdfs://192.138.137.130:9000/spark-logs"

* 启动

./start-history-server.sh

启动后出现下面的日志

starting org.apache.spark.deploy.history.HistoryServer, logging to /opt/software/spark-2.2.0-bin-2.6.0-cdh5.7.0/logs/spark-hadoop-org.apache.spark.deploy.history.HistoryServer-1-hadoop.out

查看日志是否启动成功

tail /opt/software/spark-2.2.0-bin-2.6.0-cdh5.7.0/logs/spark-hadoop-org.apache.spark.deploy.history.HistoryServer-1-hadoop.out

18/04/22 10:50:45 INFO SignalUtils: Registered signal handler for INT

* 查看web UI  
  192.168.137.130:18080  
  这时候你就可以通过该页面查看执行的任务，小伙伴们可以启动一个shark-shell执行一个任务，并执行sc.stop 查看两个页面（4040，18080）

启动shark-shell

spark-shell *--master local[2]*

执行一个任务

sc.textFile("file:///home/hadoop/data/input.txt").count

关闭

sc.stop

查看UI

192.168.137.130:18080 可以打开（任性的‘我’啥时候都可以查看）

http://192.168.137.130:4040 已经打不开了哦

**注意：**在18080页面显示的任务local模式是以local开头，yarn模式是以application开头

**4 history-server重要参数**

**注意：**以spark.history开头的需要配置在spark-env.sh中的SPARK\_HISTORY\_OPTS，以spark.eventLog开头的配置在spark-defaults.conf

spark.history.fs.update.interval 默认值10秒  
这个参数指定刷新日志的时间，更短的时间可以更快检测到新的任务以及任务执行情况，但过快会加重服务器负载

spark.history.ui.maxApplication 默认值intMaxValue  
这个参数指定UI上最多显示的作业的数目

spark.history.ui.port 默认值18080  
这个参数指定history-server的网页UI端口号

spark.history.fs.cleaner.enabled 默认为false  
这个参数指定history-server的日志是否定时清除，true为定时清除，false为不清除。这个值一定设置成true啊，不然日志文件会越来越大。

spark.history.fs.cleaner.interval 默认值为1d  
这个参数指定history-server的日志检查间隔，默认每一天会检查一下日志文件

spark.history.fs.cleaner.maxAge 默认值为7d  
这个参数指定history-server日志生命周期，当检查到某个日志文件的生命周期为7d时，则会删除该日志文件

spark.eventLog.compress 默认值为false  
这个参数设置history-server产生的日志文件是否使用压缩，true为使用，false为不使用。这个参数务可以成压缩哦，不然日志文件岁时间积累会过大

spark.history.retainedApplications 　默认值：50  
在内存中保存Application历史记录的个数，如果超过这个值，旧的应用程序信息将被删除，当再次访问已被删除的应用信息时需要重新构建页面。

**5 sc.stop的重要性**

这里为什么单独把sc.stop单独拿出来说一说？ 害怕有些小伙伴对于页面中的Show incomplete applications和Back to completed applications概念不清楚。  
上面我们已经启动了一个spark-shell ，我们可以在UI中查看小伙伴们会发现有incomplete和completed两个页面，我们看看官网给我们的解释：  
Note  
1. The history server displays both completed and incomplete Spark jobs. If an application makes multiple attempts after failures, the failed attempts will be displayed, as well as any ongoing incomplete attempt or the final successful attempt.  
2. Incomplete applications are only updated intermittently. The time between updates is defined by the interval between checks for changed files (spark.history.fs.update.interval). On larger clusters the update interval may be set to large values. The way to view a running application is actually to view its own web UI.  
3. Applications which exited without registering themselves as completed will be listed as incomplete —even though they are no longer running. This can happen if an application crashes.  
4. One way to signal the completion of a Spark job is to stop the Spark Context explicitly (sc.stop()), or in Python using the with SparkContext() as sc: construct to handle the Spark Context setup and tear down.

同学们自己进行翻译试试看哈。  
**我这里给出一个总结：**  
启动一个spark-shell或者通过spark-submit可提交一个应用程序，执行sc.stop就会在completed页面中出现。如果没有执行sc.stop或者直接退出的都会显示在incomplete页面中，同时在hdfs目录中对于完成和未完成的页面文件格式也不同，如下是我的测试：

hdfs dfs -ls /spark-logs

Found 3 items

-rwxrwx--- 3 hadoop supergroup 213 2018-04-22 10:55 /spark-logs/local-1524423324577.inprogress

-rwxrwx--- 3 hadoop supergroup 24533 2018-04-22 11:08 /spark-logs/local-1524424039150

-rwxrwx--- 3 hadoop supergroup 213 2018-04-22 11:09 /spark-logs/local-1524424187239.inprogress

.inprogress为后缀的文件，这个文件是未完成的作业或者正在运行的作业的日志文件。

**6 REST API**

* Rest服务  
  　　Spark源为了方便用户对任务做监控，从1.4版本启用Rest服务，用户可以通过访问地址，得到application的运行状态。  
  　　Spark的REST API返回的信息是JSON格式的，开发者们可以很方便地通过这个API来创建可视化的Spark监控工具。目前  
  　　这个API支持正在运行的应用程序，也支持历史服务器。在请求URL都有/api/v1。比如，对于历史服务器来说，我们可以通过  
  　　http://***:18080/api/v1 来获取一些信息，端口可以改；对于正在运行的Spark应用程序，我们可以通过 https://***/api/v1  
  　　来获取一些信息。

　　主要用途： 通过rest服务，可以轻松对任务时长、stage等做监控，同时可以配合时间序列数据库，对集群各个任务做监控。  
官方也给了相关的[接口](http://spark.apache.org/docs/latest/monitoring.html#rest-api)有兴趣的同学可以去看看。