

Spark 常见问题维护手册 V1.0

Spark 常见问题维护手册 V1.0

文档版本 01
发布日期 2016-03-31

华为技术有限公司



版权所有 © 华为技术有限公司 2016。 保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： <http://www.huawei.com>

客户服务邮箱： support@huawei.com

客户服务电话： 4008302118

目 录

SPARK	3
1、基本概念.....	3
【概述】	3
【FI 中的 Spark】	5
【日志概述】	5
【任务信息收集】	7
2、常见问题.....	10
【Spark 任务提交失败】	10
[SPARK-10001] Driver 端提示 executor memory 超限	10
[SPARK-10002] Yarn-cluter 模式下， Can't get the Kerberos realm 异常.....	11
[SPARK-10003] 找不到用户组导致提交失败	12
[SPARK-10004] Jdk 版本不匹配启动 spark-sql， spark-shell 失败	13
[SPARK-10005] 提交 spark 任务时，连接 ResourceManager 异常	13
[SPARK-10006]端口绑定不成功导致任务提交失败	15
[SPARK-10007] 集群外安装客户端提交 spark 任务失败	15
【Spark 任务提交后，一直处于 Accept 状态，无法正常运行】	16
[SPARK-20001] Driver 端提示 executor memory 超限	16
【Spark 任务运行出现异常】	17
[SPARK-30001] 使用 Beeline 连接 JDBCServer Load 本地文件执行异常	17
[SPARK-30002] Spark 任务 spark.driver.maxResultSize 不足异常	18
[SPARK-30003] SparkContext 初始化失败	18
[SPARK-30004] 无可用的 Node 导致 Spark 运行失败	19
[SPARK-30005] Spark 任务访问 HBase，报认证异常	20
[SPARK-30006] Driver 内存不足导致任务运行失败	21
[SPARK-30007] Container 运行 memory 超限导致 container 挂掉	21
[SPARK-30008] Container 运行 OOM 异常导致 container 挂掉	22

SPARK

1、基本概念

【概述】

FI 集群中，支持的 Spark 的任务运行模式有 local, yarn-client, yarn-cluster 等，Spark 任务运行是分布式的，由任务的 Driver 驱动任务的运行，由 Executor 来执行具体的任务。大多情况下，Executor 中执行的代码逻辑需要由用户编写。也就是说 Executor 进程运行的命令由用户编写，相当于用户的执行逻辑与 Spark 自身的逻辑是耦合的，出现问题之后难以定位分界，此章主要介绍 Spark 任务架构，FI 集群中 Spark 各角色实例介绍及相关日志说明，Spark 任务出现问题后的定位手段，常见的 spark 任务异常问题及解决办法。

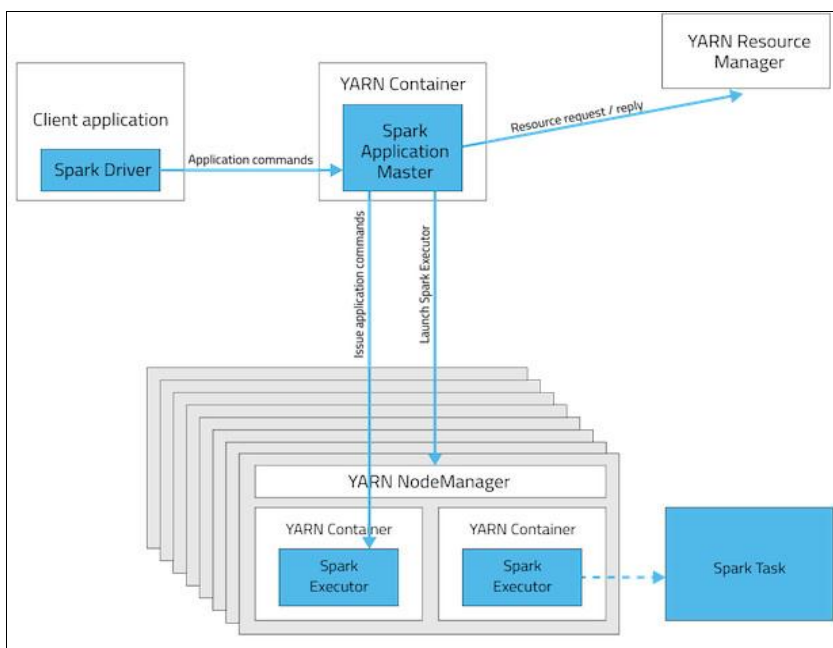
Spark-On-Yarn

一个 Spark 任务主要包含 Driver, ApplicationMaster, Executor 等部分，其中 Driver 负责驱动任务的运行，把相关的函数封装为 Task 并分发到 Executor, ApplicationMaster 主要与 Yarn 交互负责申请资源启动 Executor，汇总 Executor 信息，Executor 主要负责执行具体的 task。

Yarn-client 模式

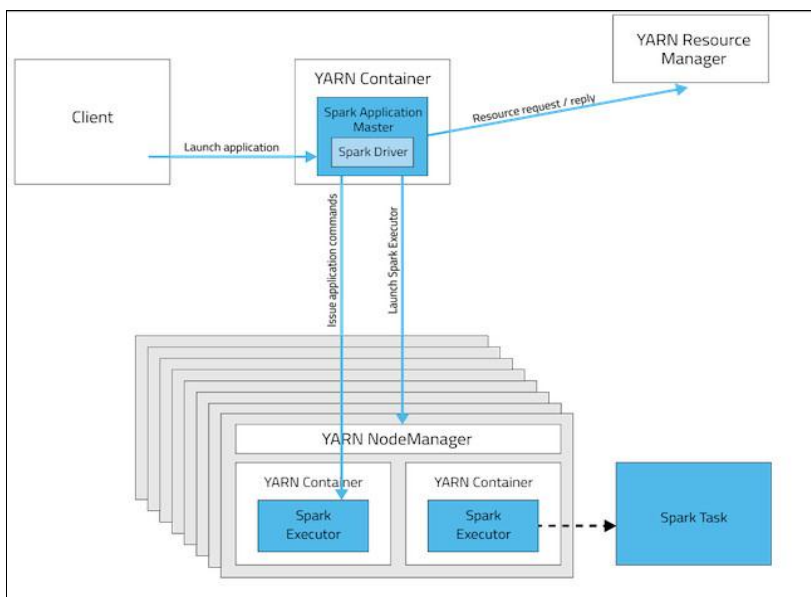
在 yarn-client 模式下，Spark 任务的 driver 作为一个进程运行在提交任务的节点（不是由 yarn 启动，不受 yarn 的管理）。ApplicationMaster 作为一个 container 进程启动在集群中的某个节点上(具体在哪个节点上启动由 Yarn 决定)。ApplicationMaster 作为集群中与 Yarn 交互的核心向 Yarn 申请资源以启动 Container(Executor)，每个 Executor 都作为一个进程，启动后等待 driver 发送 task 并执行 task。

Yarn-client 模式架构图：



Yarn-cluster 模式

在 yarn-cluster 模式下，Spark 任务的 driver 和 ApplicationMaster 运行在同一个 container 进程中，同时受 yarn 的管理。ApplicationMaster 同样作为一个 container 进程运行在集群中的某个节点上(具体在哪个节点上启动由 Yarn 决定)，driver 作为线程运行在 ApplicationMaster 的 container 进程中。ApplicationMaster 作为集群中与 Yarn 交互的核心向 Yarn 申请资源以启动 Container(Executor)，每个 Executor 都作为一个进程，启动后等待 driver 发送 task 并执行 task。



总结

spark 任务运行在 yarn 集群中，yarn-client 模式适用于开发测试，driver 直接运行在提交任务的节点，不受 yarn 的控制，便于在控制台观察任务运行状态，在 client 端的进程不能关

掉。Yarn-cluster 模式下，任务的相关进程均运行在集群中，受 yarn 的管理，提交任务后，客户端就可以关闭。

【FI 中的 Spark】

Spark 服务在 FI 中有三个角色，分别为 JDBCServer，JobHistory，SparkResource 三个角色。

名称	说明
JDBCServer	C30 版本中一个集群内可部署一个 JDBCServer 实例，C50 版本中一个集群内可部署两个 JDBCServer 实例，采用主备模式。对外提供 Sparksql 数据库服务，将用户通过 beeline 提交的 SQL 语句进行编译，解析成对应的 job 执行，从而完成数据的提取、转换、分析。
JobHistory	JobHistory 实例进程主要用来记录查看 Spark 正在运行或运行完的任务的历史信息。查看对应任务运行信息包括 DAG 图，每个 task 运行情况，运行环境变量等
SparkResource	SparkResource 的实例并不是一个进程，而是在安装 SparkResource 的节点上会部署 Spark 的 executor 运行需要的配置，认证文件，jar 包等，以便 NodeManager 启动 executor 使用。

【日志概述】

Spark 服务的日志主要存放在 \$BigdataLogHome/Spark 路径下，默认在 /var/log/Bigdata/spark 路径下；

Spark 日志主要包含角色的启动日志，存放在对应角色节点的 \$BigdataLogHome/Spark 路径下的 prestart.log；当组件发生重启时，重启的相关信息记录在 *.out 中，可根据对应时间查看；

JDBCServer 作为服务端的进程，其本身作为一个运行在 Yarn 上的 Spark 任务，为客户端提供 sql 解析和执行服务；如果客户端执行 beeline 提交的 sql 语句出现异常或在界面查看到 JDBCServer 状态异常，可在 JDBCServer 的节点上查看 JDBCServer.log 取相关信息。

在 Spark 安装后启动时会检查 spark 的可用性，当出现安装完 spark 启动失败时，相关信息会打印到 JobHistory 对应节点的 spark-availability-check.log 中；

SparkResource 并非一个实体进程，而仅仅是为了下载客户端分发文件使用的角色，不涉及日志。

日志类型	日志文件名	描述
JDBCServer 日志	JDBCServer.log	JDBCServer 运行日志。JDBCServer 作为服务端的进程，其本身作为一个运行在 Yarn 上的 Spark 任务，为客户端提供 sql 解析和执行服务；如果客户端执行 beeline 提交的 sql 语句出现异常或在界面查看到 JDBCServer 状态异常，可在 JDBCServer 的节点上查看 JDBCServer.log 取相关信息
	JDBCServer*.log.zip	运行日志的历史归档，以归档时的日期作为标识。如，JDBCServer.2016-02-29_15-08-24.[7].log.zip 表示归档时间点为：2016-02-29_15-08-24。
	cleanup.log	安装卸载实例时的清理日志。
	prestart.log	实例启动日志
	jdbc-state-check.log	JDBCServer 状态检查日志(仅 C50)
	spark-omm-org.apache.spark.sql.hive.thriftserver.om.ha.HiveThriftServer2Shell-1-189-39-235-158.out.1	JDBCServer 日志如果发生重启，该日志会记录重启信息。
JobHistory 运行日志	JobHistory.log	JobHistory 运行日志。
	JobHistory***.[1].log.zip	运行日志的历史归档，以归档时的日期作为标识。如，JobHistory.2016-02-29_15-08-24.[7].log.zip 表示归档时间点为：2016-02-29_15-08-24。
	cleanup.log	安装卸载实例时的清理日志。
	spark-omm-org.apache.spark.deploy.history.HistoryServer-1-*.out.1	JobHistory 日志如果发生重启，该日志会记录重启信息。
	prestart.log	实例启动日志
	spark-availability-check.log	Spark 任务启动后会对 spark 服务的可用性进行检查。该日志记录相关信息

【任务信息收集】

通常分析一个 Spark 任务的运行信息，主要需要如下几个资料：

- Driver的日志（主要记录任务运行的执行情况，task的分配调度，结果的打印等）
- Executor的日志（主要记录task的具体执行情况,其中Am(Application Master)也是 Executor，其中记录了与driver，ResourceManager交互，Executor的分配）
- NodeManager（NM）日志（主要记录与Executor交互，监控Executor信息）
- ResourceManager（RM）日志（主要记录与Driver，Am的交互）
- JobHistory Event事件信息（记录任务的dag，task执行等信息）

任务运行过程中

在任务运行过程中，任务的信息查看比较方便：

Driver 日志的查看：在 yarn-client 模式下，driver 直接运行在启动任务的节点上，可以再控制台，直接查看；对于 yarn-cluster 模式，driver 作为线程运行在 AM 进程中，可在 am 日志查看（如图 1 所示）；

Container 日志查看：可以通过 yarn 组件的原生页面跳转到 Spark 的原生页面根据 applicationId 查看任务的相关信息，可跳转到 Spark 原声页面（如图 2 红色矩形框）。也可以从 Spark 组件直接进入原声页面查看相关任务信息。进入原生页面后，找到相关任务 ID（如图 3 所示）。

ResourceManager/NodeManager 日志查看与收集：在集群 web 页面，点击 System→Log Download(Maintenance) 在出现的页面中，选择在 Services 栏 RM，NM 等，选择获取日志的开始和结束时间，点击 Download，即可下载获取相关日志，如图 5 所示。

JobHistoryEvent 事件查看：通过 JobHistory 页面查看可以查看任务运行的 DAG 图，task 等信息。对于 JobHistory 中记录的 Event 事件的文件，可以通过 JobHistory 进程再不同的机器上解析，如需远程查看可从 hdfs 上获取相关资料。Hdfs dfs -get /sparkJobHistory/appld。在任务运行过程中可通过图 3，4 所示方法进入 Spark 原生页面；

Attempt ID	Started	Node	Logs
appattempt_1458007651863_0014_000001	Tue Mar 15 20:25:13 +0800 2016	https://192-168-1-1:26010	Logs

图 1 运行任务过程中 AM 日志查看

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU V-Cores	Allocated Memory MB	Progress	Tracking UI	Blacklist Nodes
application_1458007651863_0014	admin	SparkSQL_192-168-1-1	SPARK	default	Tue Mar 15 20:25:13 +0800	N/A	RUNNING	UNDEFINED	3	3	3584	<div></div>	ApplicationMaster	0

图 2 Yarn 原生页面跳转至 Spark 原声页面

Executor ID	Address	RDD Blocks	Storage Memory	Disk Used	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time	Input	Shuffle Read	Shuffle Write	Logs
2	192.168.1.1:23335	0	0.0 B / 265.1 MB	0.0 B	0	0	0	0	0 ms	0.0 B	0.0 B	0.0 B	stdout stderr
3	192.168.1.2:23494	0	0.0 B / 265.1 MB	0.0 B	0	0	0	0	0 ms	0.0 B	0.0 B	0.0 B	stdout stderr
driver	192.168.1.3:23518	0	0.0 B / 265.1 MB	0.0 B	0	0	0	0	0 ms	0.0 B	0.0 B	0.0 B	

图 3 任务运行过程中，executor(container)日志查看



图 4 登录 JobHistory

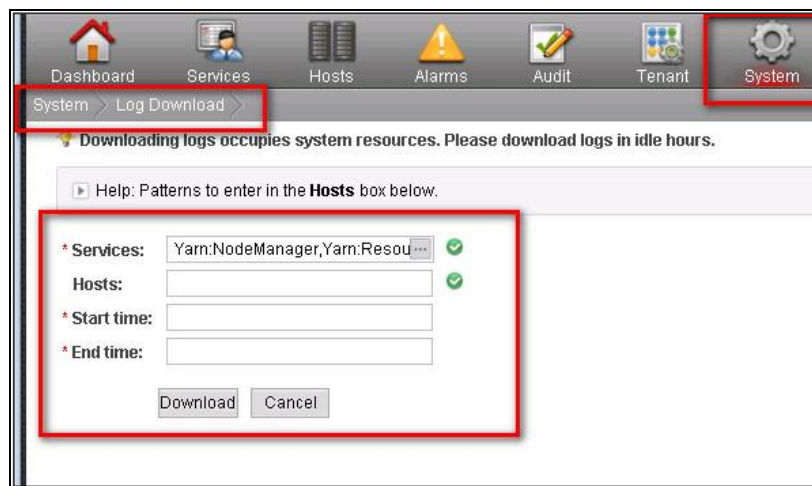


图 5 RM/NM 等日志获取方式

任务运行结束后

Driver 日志：对于 yarn-client 模式，如果没有保存，就无法再查看了。对于 yarn-cluster 模式，driver 日志可以再 am 日志中查看；

Contaienr 日志查看：默认情况下，container 日志在任务运行完后，会自动归集到 hdfs

上。可从 hdfs 上获取 container 日志。命令：`hdfs dfs -get /tmp/logs/username/logs/appId /path` 其中 username 为提交任务的用户名。如 admin 或 spark; 或者通过 `yarn logs -applicationId appId >> yourlogfile`。如果日志归集功能没有打开，则日志会存放在本地。可在 JobHistory 页面查看到相关 executor(container) 在哪个节点上，然后可到相应节点的 `${yarn.nodemanager.log-dirs}` 路径下查看相应日志。默认为 `/srv/BigData/hadoop/data1/nm/containerlogs`。

RM/NM 日志的获取同运行过程中获取方法一致。

JobHistoryEvent 事件信息同运行过程中获取方法一致。

Note:

Spark 任务与其他组件也有较多交互，如 Kafka, Hbase, Zookeeper 等组件，在设计到这些组件时，也需要查看相关组件日志协助分析相关任务。通常组件的日志信息获取，可通过与去 RM/NM 日志同样的方式。

在分析 Spark 任务的过程中，由于业务与 Spark 框架是耦合的，在任务运行出现问题时，通常也需要根据用户的业务逻辑，执行流程，数据情况等来具体分析。

在一些任务运行如出现某个 container 运行较慢，其他 container 运行正常情况下，可以到相应节点上分析该 container 的进程状态，可通过如图 2 中的信息找到 container 对应的 host，然后在该节点使用 `ps -ef | grep appId | grep -v bash | grep java` 等命令查找出对应 containerid。使用如下命令如 jmap 查看进程中每个对象占用的内存空间，jstack 查看进程中某个现场的运行状态。

Spark 任务运行的 container 日志由 yarn 管理，而任务运行中的 event 时间记录则有 spark 的 jobhistory 来负责管理。JobHistory 会定期清理相关时间记录。

- Spark 的 JobHistory 记录 spark 任务运行情况，由参数 `spark.eventLog.enabled` 控制是否记录，默认开启，由 `spark.eventLog.dir` 来控制记录的路径，默认配置在 hdfs 上的 `/sparkJobHistory` 路径。

- EventLog 日志清理逻辑：

1. `spark.history.fs.cleaner.enable`：是否周期删除 event log 日志；
2. `spark.history.fs.cleaner.interval.seconds`：定期扫描日志的时间间隔，默认一天；
3. `spark.history.fs.cleaner.maxAge.seconds`：EventLog 在 HDFS 上存放的最大生命周期，单位 s，默认为 15 天；

通过配置项 1) 开启 EventLog 定期删除日志的功能，每隔配置项 2) 的时间间隔去扫描 HDFS 上的 JobHistory 目录下的日志文件，如果当前时间减去该日志文件最后被修改的时间之差大于配置项 3) 的时间，则表示该日志文件已经超过配置项 3) 的时间未被使用，则删除该日志文件。

- Container 日志清理：

同 `spark.eventlog` 记录一样，container 日志也会定期清理。分别由如下参数控制：
`yarn.log-aggregation.retain-check-interval-seconds`, `yarn.log-aggregation.retain-seconds`

2、常见问题

【Spark 任务提交失败】

[SPARK-10001] Driver 端提示 executor memory 超限

【问题背景与现象】

内存超限导致提交 Spark 任务失败

【原因分析】

1. 在 driver 日志中直接打印申请的 executor memory 超过集群限制

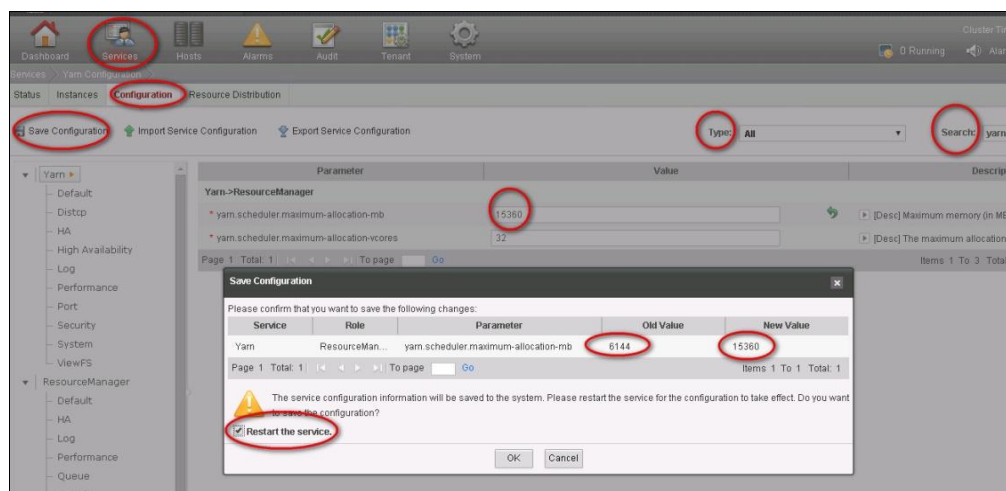
```
16/02/06 14:11:25 INFO Client: Verifying our application has not requested more than
the maximum memory capability of the cluster (6144 MB per container)
16/02/06 14:11:29 ERROR SparkContext: Error initializing SparkContext.
java.lang.IllegalArgumentException: Required executor memory (10240+1024 MB) is
above the max threshold (6144 MB) of this cluster!
```

2. Spark 任务提交至 yarn 上面，运行 task 的 executor 使用的资源受 yarn 的管理。从报错信息可看出，用户申请启动 executor 时，指定 10g 的内存，超出了 yarn 设置的每个 container 的最大内存的限制，导致任务无法启动。

【解决办法】

此时可通过修改 yarn 的配置，提高对 container 的限制。如可通过调整 yarn.scheduler.maximum-allocation-mb 参数的大小，可控制启动的 executor 的资源，修改之后要重启下 yarn 服务。

配置修改方法：登录 FIweb 页面，点击 Service->Yarn->Configuration->“将 Type 修改由 Basic 为 ALL”->“在 Search 栏输入 yarn.scheduler.maximum-allocation-mb”修改参数并保存重启服务。如下图所示：



[SPARK-10002] Yarn-clsuter 模式下，Can't get the Kerberos realm 异常

【问题背景与现象】

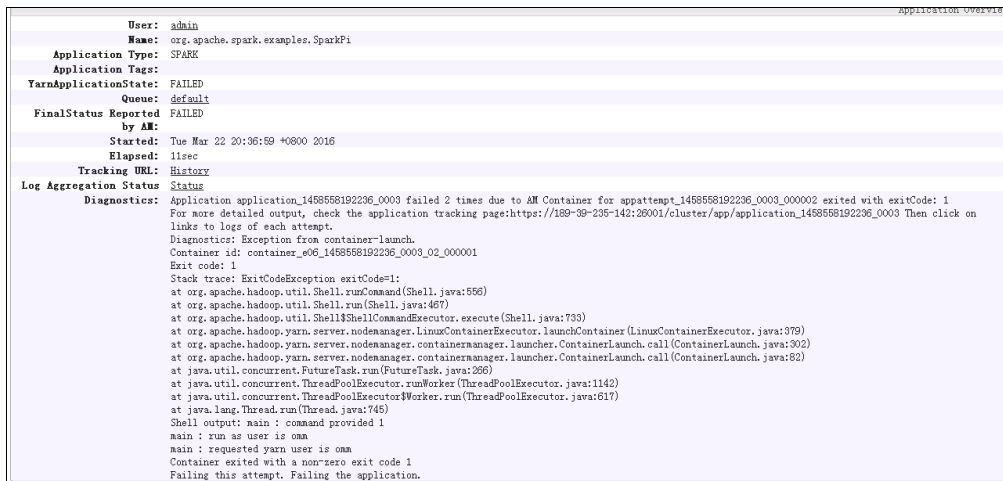
认证异常导致提交 Spark 任务失败

【原因分析】

1. 在 driver 端打印异常找不到连接 hdfs 的 token，如下：

```
16/03/22 20:37:10 WARN Client: Exception encountered while connecting to the server :  
org.apache.hadoop.ipc.RemoteException(org.apache.hadoop.security.token.SecretMa  
nager$InvalidToken): token (HDFS_DELEGATION_TOKEN token 192 for admin) can't be  
found in cache  
16/03/22 20:37:10 WARN Client: Failed to cleanup staging  
dir .sparkStaging/application_1458558192236_0003  
org.apache.hadoop.ipc.RemoteException(org.apache.hadoop.security.token.SecretMa  
nager$InvalidToken): token (HDFS_DELEGATION_TOKEN token 192 for admin) can't be  
found in cache
```

2. 在 Yarn 原声页面显示 am 启动两次均失败，任务退出，如下图信息：



Application Overview

User: admin
Name: org.apache.spark.examples.SparkPi
Application Type: SPARK
Application Tags:
YarnApplicationState: FAILED
Queue: default
FinalStatus Reported by AM: FAILED
Started: Tue Mar 22 20:36:59 +0800 2016
Elapsed: 11sec
Tracking URL: History
Log Aggregation Status: Status
Diagnostics: Application application_1458558192236_0003 failed 2 times due to AM Container for appattempt1458558192236_0003_000002 exited with exitCode: 1
For more detailed output, check the application tracking page: https://189-39-235-142:26001/cluster/app/application_1458558192236_0003 Then click on links to logs of each attempt.
Diagnostics: Exception from container-launch.
Container id: container_e06_1458558192236_0003_02_000001
Exit code: 1
Stack trace: ExitCodeException exitCode=1:
at org.apache.hadoop.util.Shell.runCommand(Shell.java:556)
at org.apache.hadoop.util.Shell.run(Shell.java:467)
at org.apache.hadoop.util.Shell\$ShellCommandExecutor.execute(Shell.java:733)
at org.apache.hadoop.yarn.server.nodemanager.LinuxContainerExecutor.launchContainer(LinuxContainerExecutor.java:379)
at org.apache.hadoop.yarn.server.nodemanager.containermanager.launcher.ContainerLaunch.call(ContainerLaunch.java:302)
at org.apache.hadoop.yarn.server.nodemanager.containermanager.launcher.ContainerLaunch.call(ContainerLaunch.java:82)
at java.util.concurrent.FutureTask.run(FutureTask.java:266)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
at java.util.concurrent.ThreadPoolExecutor\$Worker.run(ThreadPoolExecutor.java:617)
at java.lang.Thread.run(Thread.java:745)
Shell output: main : command provided 1
main : run as user is oom
main : requested yarn user is oom
Container exited with a non-zero exit code 1
Failing this attempt. Failing the application.

3. 查看 ApplicationMaster 日志看到如下异常信息：

```
Exception in thread "main" java.lang.ExceptionInInitializerError  
Caused by: org.apache.spark.SparkException: Unable to load YARN support  
Caused by: java.lang.IllegalArgumentException: Can't get Kerberos realm  
Caused by: java.lang.reflect.InvocationTargetException  
Caused by: KrbException: Cannot locate default realm  
Caused by: KrbException: Generic error (description in e-text) (60) - Unable to locate  
Kerberos realm  
org.apache.hadoop.hive.metastore.MetaStoreUtils.newInstance(MetaStoreUtils.java:1  
410)  
... 86 more  
Caused by: javax.jdo.JDOFatalInternalException: Unexpected exception caught.  
NestedThrowables:java.lang.reflect.InvocationTargetException  
... 110 more
```

4. `./spark-submit --class yourclassname --master yarn-cluster /yourdependencyjars` 任务以 yarn-cluster 模式提交任务，driver 端会在集群中启用，由于加载的是客户端的

spark.driver.extraJavaOptions,在集群节点上对应路径下找不到对应的 kdc.conf 文件,无法获取 kerberos 认证所需信息,导致 am 启动失败。

【解决办法】

在客户端提交任务时,在命令行中配置自定义的 spark.driver.extraJavaOptions 参数 这样任务运行时就不会自动加载客户端路径下的 spark-defaults.conf 中的 spark.driver.extraJavaOptions; 或者在启动 spark 任务时,通--conf 来指定 driver 的配置,如下(此处=号后面的红色引号部分不能缺少):

```
./spark-submit -class yourclassname --master yarn-cluster --conf
spark.driver.extraJavaOptions="
-Dlog4j.configuration=/opt/huawei/Bigdata/FusionInsight-Spark-
1.3.0/spark/conf/log4j.properties.template
-Djetty.version=x.y.z
-Dzookeeper.server.principal=zookeeper/hadoop.hadoop.com
-Djava.security.krb5.conf=/opt/huawei/Bigdata/FusionInsight-Spark-1.3.0/kdc.conf
-Djava.security.auth.login.config=/opt/huawei/Bigdata/FusionInsight-Spark-
1.3.0/jaas.conf" ./yourdependencyjars
```

[SPARK-10003] 找不到用户组导致提交失败

【问题背景与现象】

找不到用户组信息导致 Spark 任务提交失败

【原因分析】

1. 在 driver 端打印找不到用户组异常, No groups found

```
ERROR      SparkContext:      Error      initializing      SparkContext.
org.apache.hadoop.yarn.exceptions.YarnException:      Failed      to      submit
application_1457059698448_0024      to      YARN      :      Failed to submit application
application_1457059698448_0024 submitted by user sparktest reason: No groups
found      for      user      sparktest      at
org.apache.hadoop.yarn.client.api.impl.YarnClientImpl.submitApplication
```

2. FI 集群中的用户管理由 LDAP 服务管理提供,又依赖于操作系统的 sssd(redhat),nscd(suse)服务,用户的建立到同步到 sssd 服务需要一定时间,如果此时用户没有生效,或者 sssd 版本存在 bug 的情况下,某些情况下在主 RM 节点会出现用户无效的情况,导致任务提交失败。可到 ResourceManager 主节点通过 id username 命令查看是否存在此用户。

【解决办法】

1. 可通过重启 sssd 服务使用户生效,

```
service sssd restart(redhat)
```

```
service nscd restart(suse)
```

如下打印说明存在重启成功:

```
linux-33:/opt/huawei/Bigdata/FusionInsight-Spark-1.3.0 # service nscd restart
Shutting down Name Service Cache Daemon
Starting Name Service Cache Daemon
linux-33:/opt/huawei/Bigdata/FusionInsight-Spark-1.3.0 #
```

2. 重启相关服务后，可 ResourceManager 主节点通过 `id username` 命令查看相应用户信息是否已有效。如下打印说明存在相关用户：

```
linux-33:/opt/huawei/Bigdata/FusionInsight-Spark-1.3.0 # id spark
uid=20009(spark) gid=10000(supergroup) groups=10000(supergroup),10001(hadoop)
linux-33:/opt/huawei/Bigdata/FusionInsight-Spark-1.3.0 #
```

3. 之后可再次提交任务查看是否可以成功提交。

[SPARK-10004] Jdk 版本不匹配启动 spark-sql，spark-shell 失败

【问题背景与现象】

Jdk 版本不匹配导致客户端启动 spark-sql，spark-shell 失败

【原因分析】

1. 在 driver 端打印异常如下

```
Line188@PasswordUtil.java: Exception Occurs: BadPadding 16/02/22 14:25:38 ERROR
Schema: Failed initialising database. Unable to open a test connection to the given
database. JDBC url = jdbc:postgresql://ip:port/sparkhivemeta, username = spark.
Terminating connection pool (set lazyInit to true if you expect to start your database after
your app).
```

2. Sparksql 任务使用时，需要访问 DBService 以获取元数据信息，在客户端需要解密密文来访问，在使用过程中，用户使用时，没有按照流程操作，没有执行 `source` 操作，且在其客户端环境变量中存在默认的 jdk 版本，导致在执行解密过程中调用的解密程序执行解密异常，会引起用户被锁。

【解决办法】

1. 使用 `which java` 命令查看默认的 java 命令是否是客户端的 java。
2. 如果不是，请按正常的客户端执行流程，

```
source $client_path/bigdata_env
```

```
kinit *****, 然后输入用户名对应的密码，启动任务即可。
```

[SPARK-10005] 提交 spark 任务时，连接 ResourceManager 异常

【问题背景与现象】

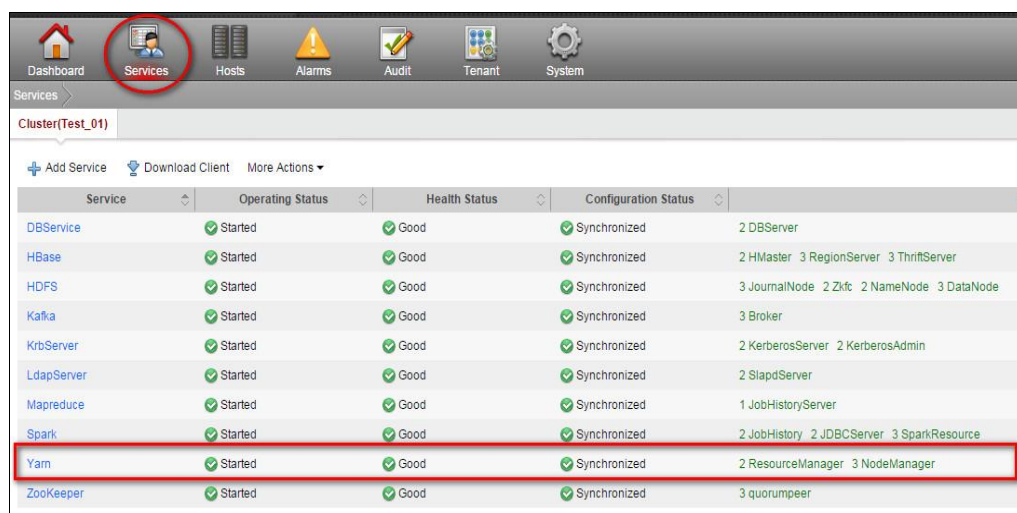
连接 ResourceManager 异常，导致 Spark 任务提交失败

【原因分析】

1. 在 driver 端打印异常如下，打印连接两个 ResourceManager 主备节点的 26004 端口均被拒绝：

```
15/08/19 18:36:16 INFO RetryInvocationHandler: Exception while invoking
getClusterMetrics of class ApplicationClientProtocolPBClientImpl over 33 after 1 fail over
attempts. Trying to fail over after sleeping for 17448ms.
java.net.ConnectException: Call From ip0 to ip1:26004 failed on connection exception:
java.net.ConnectException: Connection refused.
INFO RetryInvocationHandler: Exception while invoking getClusterMetrics of class
ApplicationClientProtocolPBClientImpl over 32 after 2 fail over attempts. Trying to fail
over after sleeping for 16233ms.
java.net.ConnectException: Call From ip0 to ip2:26004 failed on connection exception:
java.net.ConnectException: Connection refused;
```

2. 在 FI 页面查看 ResourceManager 此时是否正常功能；如下图所示，如果 Yarn 状态出现 bad 或某个 yarn 服务的实例出现 unknown 之类的异常说明此时集群的 RM 可能异常



Service	Operating Status	Health Status	Configuration Status	Details
DBService	Started	Good	Synchronized	2 DBServer
HBase	Started	Good	Synchronized	2 HMaster 3 RegionServer 3 ThriftServer
HDFS	Started	Good	Synchronized	3 JournalNode 2 Zkfc 2 NameNode 3 DataNode
Kafka	Started	Good	Synchronized	3 Broker
KrbServer	Started	Good	Synchronized	2 KerberosServer 2 KerberosAdmin
LdapServer	Started	Good	Synchronized	2 SlapdServer
Mapreduce	Started	Good	Synchronized	1 JobHistoryServer
Spark	Started	Good	Synchronized	2 JobHistory 2 JDBCServer 3 SparkResource
Yarn	Started	Good	Synchronized	2 ResourceManager 3 NodeManager
ZooKeeper	Started	Good	Synchronized	3 quorumpeer

3. 排查使用的客户端是否是集群最新的客户端。
排查集群是否做过实例 RM 迁移相关操作（先卸载某个 RM 实例，然后在其他节点添加回来）
在 FI 页面点击 Audit 模块，查看 audit 审计日志，是否有相关操作的记录。
4. 使用 ping 命令，查看 ip 是否可联通；

【解决办法】

1. 如果 RM 出现异常，可参考 Yarn 相关章节查看解决方法；
2. 重新下载客户端
3. 不通，需要协调网络管理相关人员协助排查网络。

[SPARK-10006]端口绑定不成功导致任务提交失败

【问题背景与现象】

在提交任务控制台打印如下异常：

```
Line 981: 15/12/16 14:38:38 WARN AbstractLifeCycle: FAILED SelectChannelConnector:  
java.net.BindException: Address already in use
```

【原因分析】

Spark 任务的 driver 启动时，会绑定 ui 端口，绑定时会进行 16 次尝试，日志显示端口均被占用，导致 driver 启动不成功。

【解决办法】

修改 spark.ui.port 为 0，这样 Spark 客户端会依次随机选择 16 个端口来绑定，可以解决端口绑定冲突导致任务提交失败的问题。

[SPARK-10007] 集群外安装客户端提交 spark 任务失败

【问题背景与现象】

FusionInsight HD C30 版本

集群外安装客户端提交 spark 任务处于 accept 状态一段时间后失败

【原因分析】

1. drive 日志中打印如下异常：

```
diagnostics: Application application_1457084678413_0373 failed 2 times due to AM  
Container for appattempt_1457084678413_0373_000002 exited with exitCode: 10  
For more detailed output, check application tracking page:https://linux-  
32:26001/proxy/application_1457084678413_0373/Then, click on links to logs of each  
attempt.  
Diagnostics: Exception from container-launch.  
Container id: container_1457084678413_0373_02_12582913  
Exit code: 10  
Stack trace: ExitCodeException exitCode=10:
```

2. 在 ApplicationMaster 日志中异常如下：

```
Failed to connect to driver at linux-34:23006, retrying ... |  
org.apache.spark.Logging$class.logError(Logging.scala:75)  
2016-03-15 19:45:21,462 | ERROR | [main] | Failed to connect to driver at linux-  
34:23006, retrying ... | org.apache.spark.Logging$class.logError(Logging.scala:75)  
2016-03-15 19:45:21,563 | ERROR | [main] | Failed to connect to driver at linux-  
34:23006, retrying ... | org.apache.spark.Logging$class.logError(Logging.scala:75)
```


3. Spark 任务在客户端启动 driver，applicationmaster 运行在集群中，启动时会向 driver 注册自己的信息，C30 版本中 applicationmaster 启动后默认通过 hostname 来查找 driver，如果无法解析此 hostname，则 applicationmaster 无法启动。在 am 启动的节点上查看相关使用 `cat /etc/hosts` 命令查看是否有配置 driver 所在节点的 ip 和 hostname。

【解决办法】

在客户端 `spark-default.conf` 配置文件中设置 `spark.driver.host` 参数，指定 Driver 也就是本机 ip，后续 AppMaster 会直接通过此 ip 向 Driver 注册

【Spark 任务提交后，一直处于 Accept 状态，无法正常运行】

[SPARK-20001] Driver 端提示 executor memory 超限

【问题背景与现象】

Spark 任务提交后，任务一直处于 Accept 状态，无法正常运行

【原因分析】

FI 集群中，Spark 任务运行在 Yarn 的资源管理框架下，由 Yarn 来负责资源的调度管理，Yarn 在接收 Spark 任务后，任务会先处于 Accept 状态，待为该任务分配到资源之后，任务才会进入 Running 状态执行。如果 Yarn 上没有足够资源，则任务将一直处于 Accept 状态。可通过如下方式查看 Yarn 上资源

1. 查看当前 yarn 是否有资源可供调度，如果资源不足，则无法正常运行任务。
需要等其他任务运行完才能运行该任务或者考虑扩容，如果资源充足，参考步骤 2；

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
12	0	1	11	3	4.50 GB	24 GB	0 B	3	24	0	3	0	0	0

2. 确认任务提交的队列是否有足够资源，Spark 任务默认提交到 default 队列，以下以 default 队列为例。如果队列资源不足，请考虑提交任务到其他队列。如果队列资源充足，参考步骤 3；



3. 确认当前队列的 appmaster 资源是否耗尽；
Yarn 的资源有参数专门控制有多少比例是给 appmaster 使用，如果该值过小，则会导致，队列有资源的情况下，appmaster 正常启动，任务依然无法正常运行。如下图所示，appmaster 一共有 21 个 vcore 已经用尽，此时任务会处于 Accept 状态。

```

Queue State: RUNNING
Used Capacity: 39.4%
Absolute Used Capacity: 31.5%
Absolute Capacity: 80.0%
Absolute Max Capacity: 100.0%
Used Resources: <memory:76288, vCores:63>
Num Schedulable Applications: 21
Num Non-Schedulable Applications: 3
Num Containers: 63
Max Applications: 10000
Max Applications Per User: 100000
Max Application Master Resources: <memory:50176, vCores:21>
Used Application Master Resources: <memory:32256, vCores:21>
Max Application Master Resources Per User: <memory:400384, vCores:161>
Configured Capacity: 80.0%
Configured Max Capacity: 100.0%
Configured Minimum User Limit Percent: 100%
Configured User Limit Factor: 10.0
Accessible Node Labels: *
Ordering Policy: FIFOOrderingPolicy
Preemption: disabled

```

【解决办法】

1. 对于此种情况属于资源分配不合理，可通过调整参数来增加 appmaster 的可用资源。
2. 在 C30 版本中，可通过调整 `yarn.scheduler.capacity.maximum-am-resource-percent` 参数来提高 am 可用资源比例，修改之后要重启 yarn 服务。
3. 在 C50 及以后的版本中，该参数废弃掉，需要修改多租户相关参数，在页面点击 Tenant → Dynamic Resource Plan → 修改相应队列的 Master Am Resource Percent，修改后，无需重启 Yarn。

【Spark 任务运行出现异常】

[SPARK-30001] 使用 Beeline 连接 JDBCServer Load 本地文件执行异常

【问题背景与现象】

使用 Beeline 连接 JDBCServer，在执行数据加载时使用 `load data local inpath '/opt/sparktest.txt' overwrite into table test;`

Beeline 中提示找不到本地文件。

【原因分析】

JDBCServer 在进行本地加载文件时，需要 driver 所在的节点放置相应文件，如果仅仅在客户端放置 '/opt/sparktest.txt'，且客户端和 JDBCServer 的 driver 不在同一个节点上，会找不到文件。如果起始时，JDBCServer 所在节点就是客户端的节点，则执行可以通过，如果中间 JDBCServer 发生主备倒换，driver 节点和客户端节点不是同一个节点则会执行异常。

【解决办法】

本地文件需要放在 JDBCServer 的 driver 所在节点（也就是主 JDBCServer 节点）。

[SPARK-30002] Spark 任务 spark.driver.maxResultSize 不足异常

【问题背景与现象】

Spark 任务运行失败，driver 端出现 spark.driver.maxResultSize 不足的异常。

【原因分析】

1. 在 driver 端直接打印如下异常：

```
ERROR      Jetty_2-72][ROOT][com.huawei.seq.common.Job.InjectedJobBase      130]
sdr_voice_vap_day get Exception : org.apache.spark.SparkException: Job aborted due to
stage failure: Total size of serialized results of 295 tasks (1025.2 MB) is bigger than
spark.driver.maxResultSize (1024.0 MB)
```

2. Spark 任务的计算由 executor 负责，多个 executor 运行得到结果后会返回给 driver，如果 driver 设置的接收数据的参数过小，则会报出如上错误。

【解决办法】

1. 根据场景需求，最好不要返回 driver 过多数据，同时可通过增大 spark.driver.maxResultSize 配置以接收更多结果。在 \$client_home/spark./conf/spark-defaults.conf 中添加一行 `spark.driver.maxResultSize = 2048m`

[SPARK-30003] SparkContext 初始化失败

【问题背景与现象】

Spark 任务运行失败，driver 端出现 Error initializing sparkcontext 异常。

【原因分析】

1. driver 端初始化异常 Error initializing sparkcontext，日志如下：

```
5/12/26 11:23:28 INFO Client: Application report for application_1451100085876_0004 (state: ACCEPTED)
5/12/26 11:23:29 INFO Client: Application report for application_1451100085876_0004 (state: ACCEPTED)
5/12/26 11:23:30 INFO Client: Application report for application_1451100085876_0004 (state: ACCEPTED)
5/12/26 11:23:31 INFO Client: Application report for application_1451100085876_0004 (state: KILLED)
5/12/26 11:23:31 INFO Client:
client token: N/A
diagnostics: Application killed by user.
ApplicationMaster host: N/A
ApplicationMaster RPC port: -1
queue: QueueB
start time: 1451100189320
final status: KILLED
tracking URL: https://hadoopc01h2:26001/cluster/app/application_1451100085876_0004
user: ocsuser
5/12/26 11:23:31 INFO Client: Deleting staging directory .sparkStaging/application_1451100085876_0004
5/12/26 11:23:31 ERROR SparkContext: Error initializing SparkContext.
org.apache.spark.SparkException: Yarn application has already ended! It might have been killed or unable to launch application master.
    at org.apache.spark.scheduler.cluster.YarnClientSchedulerBackend.waitForApplication(YarnClientSchedulerBackend.scala:120)
    at org.apache.spark.scheduler.cluster.YarnClientSchedulerBackend.start(YarnClientSchedulerBackend.scala:65)
    at org.apache.spark.scheduler.TaskSchedulerImpl.start(TaskSchedulerImpl.scala:141)
```

2. RM 中异常如下：

```

36, vCores:14> exceeds amlimit: <memory:-4608, vCores:5> | LeafQueue.java:626
015-12-26 11:23:18,901 | INFO | ResourceManager Event Processor | Not activating application application_1450500781428_19756 as amlfStarted: <memory:14
36, vCores:14> exceeds amlimit: <memory:-4608, vCores:5> | LeafQueue.java:626
015-12-26 11:23:18,902 | INFO | ResourceManager Event Processor | Not activating application application_1450500781428_19757 as amlfStarted: <memory:14
36, vCores:14> exceeds amlimit: <memory:-4608, vCores:5> | LeafQueue.java:626
015-12-26 11:23:18,902 | INFO | ResourceManager Event Processor | Not activating application application_1450500781428_19758 as amlfStarted: <memory:14
36, vCores:14> exceeds amlimit: <memory:-4608, vCores:5> | LeafQueue.java:626
015-12-26 11:23:18,902 | INFO | ResourceManager Event Processor | Not activating application application_1450500781428_19759 as amlfStarted: <memory:14
36, vCores:14> exceeds amlimit: <memory:-4608, vCores:5> | LeafQueue.java:626
015-12-26 11:23:18,902 | INFO | ResourceManager Event Processor | Not activating application application_1451100085876_0004 as amlfStarted: <memory:14
36, vCores:14> exceeds amlimit: <memory:-4608, vCores:5> | LeafQueue.java:626
015-12-26 11:23:18,902 | INFO | ResourceManager Event Processor | Not activating application application_1450500781428_19466 as amlfStarted: <memory:13
24, vCores:9> exceeds amlimit: <memory:5632, vCores:7> | LeafQueue.java:626
015-12-26 11:23:18,902 | INFO | ResourceManager Event Processor | Not activating application application_1450500781428_19471 as amlfStarted: <memory:13

```

3. NM 日志中显示/srv/Bigdata/...目录使用率超过 90%，打印如下：

```

ContainerExecutor.java:111
015-12-26 12:07:17,951 | INFO | Task killer for 37553 | should not do kill, other process uses this pid | ContainerExecutor.java:495
015-12-26 12:07:18,019 | WARN | DiskHealthMonitor-Timer | Directory /srv/BigData/hadoop/data3/nm/localdir error, used space above threshold of 90.0%, re
moving from list of valid directories | DirectoryCollection.java:250
015-12-26 12:07:18,019 | WARN | DiskHealthMonitor-Timer | Directory /srv/BigData/hadoop/data1/nm/localdir error, used space above threshold of 90.0%, re
moving from list of valid directories | DirectoryCollection.java:250
015-12-26 12:07:18,021 | WARN | DiskHealthMonitor-Timer | Directory /srv/BigData/hadoop/data1/containerlogs error, used space above threshold of 90.0%, re
moving from list of valid directories | DirectoryCollection.java:250
015-12-26 12:07:18,021 | WARN | DiskHealthMonitor-Timer | Directory /srv/BigData/hadoop/data3/containerlogs error, used space above threshold of 90.0%, re
moving from list of valid directories | DirectoryCollection.java:250
015-12-26 12:07:18,021 | INFO | DiskHealthMonitor-Timer | Disk(s) failed: 4/4 local-dirs are bad: /srv/BigData/hadoop/data4/nm/localdir,/srv/BigData/had
oop/data3/nm/localdir,/srv/BigData/hadoop/data2/nm/localdir,/srv/BigData/hadoop/data1/nm/localdir: 4/4 log-dirs are bad: /srv/BigData/hadoop/data1/contain
erlogs,/srv/BigData/hadoop/data4/containerlogs,/srv/BigData/hadoop/data3/containerlogs,/srv/BigData/hadoop/data2/containerlogs | LocalDirsHandlerService.j

```

4. 在 Yarn 原生 WebUi 上可看到 nm 处于非正常状态，如下：

Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rel N
0	0B	0B	0B	0	0	0	0	0	0	3	0

5. Spark 任务在启动过程中，会将一些文件 jar 包通过 yarn 上传至 hdfs，在 yarn.nodemanager.local-dirs 或 yarn.nodemanager.log-dirs，指定的目录使用率过高时，Nm 会出现异常，导致基于 Yarn 的任务出现异常。

【解决办法】

1. 清理磁盘，腾出更多空间或添加磁盘，扩容集群。

[SPARK-30004] 无可用 Node 导致 Spark 运行失败

【问题背景与现象】

Spark 在运行任务时出错，Executor 端打印没有可用 node

【原因分析】

1. spark 任务失败后，查看 executor 日志发现如下打印：

```

| ERROR | [Executor task launch worker-2] | Exception in task 5.0 in stage 1.0 (TID
85) | org.apache.spark.Logging$class.logError(Logging.scala:96)
org.apache.hadoop.ipc.RemoteException(java.io.IOException): File
/srv/smartcare/calc_input/appsrv/analyze_chain_resource/na/20160121/1400_1405_n
a_20160121000000/20160121000000_20160122000000_na_na_0.dat.tmp/_temporar
y/0/_temporary/attempt_201603081504_0003_r_000005_0/part-r-00005 could only
be replicated to 0 nodes instead of minReplication (=1). There are 3 datanode(s)
running and no node(s) are excluded in this operation.

```

2. 查看 FI 页面，发现有如下告警：

14002	DataNode Disk Usage Exceeds the Threshold	Major	2016-03-08 08:05:03
12030	No Valid License	Major	2016-03-08 00:00:00
14002	DataNode Disk Usage Exceeds the Threshold	Major	2016-03-07 23:23:11

- 查看 HDFS 原生页面看到如下异常：

Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Fail
2	In Service	72.68 GB	15.07 GB	26.52 GB	31.09 GB	3365	15.07 GB (20.73%)	0
2	In Service	72.68 GB	15.05 GB	57.63 GB	0 B	3208	15.05 GB (20.71%)	0
0	In Service	72.68 GB	12.46 GB	60.22 GB	0 B	3173	12.46 GB (17.14%)	0

- 磁盘空间不足

【解决办法】

- 清理磁盘，腾出更多空间，如果磁盘相关内容不能清除则可考虑添加磁盘或扩容集群。

[SPARK-30005] Spark 任务访问 HBase，报认证异常

【问题背景与现象】

Spark 在运行任务时出错，且 Spark 任务中访问 HBase 报认证异常。

【原因分析】

- driver 端可能有如下打印：

```
Caused by: org.apache.hadoop.hbase.client.RetriesExhaustedException: Failed after
attempts=35, exceptions: Thu Mar 10 17:28:43 CST 2016,
RpcRetryingCaller{globalStartTime=1457602122550, pause=100, retries=35},
java.io.IOException: Could not set up IO Stream
```

- 在 executor 端日志可能有打印找不到 tgt 的异常：

```
ERROR | [Executor task launch worker-1] | SASL authentication failed. The most likely
cause is missing or invalid credentials. Consider 'kinit'.
javax.security.sasl.SaslException: GSS initiate failed [Caused by GSSException: No
valid credentials provided (Mechanism level: Failed to find any Kerberos tgt)]
```

- 在 C50SPC200 版本以后，spark 解除对 hbase 的依赖，在程序需要访问 hbase 时，需要获取访问 hbase 的 token，如果用户需要访问 hbase，则需要打开获取访问 hbase 需要的 token 配置项，

【解决办法】

- 需要在客户端的 \$client_path/Spark/spark/conf/spark-defaults.conf 中修改配置
spark.hbase.obtainToken.enabled 为 true，
spark.inputFormat.cache.enabled 为 false。

[SPARK-30006] Driver 内存不足导致任务运行失败

【问题背景与现象】

Driver 内存不足导致任务运行失败。

console 端报出 **java.lang.OutOfMemoryError: java heap space** 直接报异常如下：

```
scala> documents_ans.count
res5: Long = 10780

scala> documents_ans.collect
Exception in thread "task-result-getter-3" java.lang.OutOfMemoryError: Java heap space
    at java.io.ObjectInputStream$HandleTable.grow(ObjectInputStream.java:3465)
    at java.io.ObjectInputStream$HandleTable.assign(ObjectInputStream.java:3271)
    at java.io.ObjectInputStream.readString(ObjectInputStream.java:1649)
    at java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1341)
    at java.io.ObjectInputStream.readArray(ObjectInputStream.java:1706)
    at java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1344)
    at java.io.ObjectInputStream.defaultReadFields(ObjectInputStream.java:1990)
    at java.io.ObjectInputStream.readSerialData(ObjectInputStream.java:1915)
    at java.io.ObjectInputStream.readOrdinaryObject(ObjectInputStream.java:1798)
    at java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1350)
    at java.io.ObjectInputStream.readArray(ObjectInputStream.java:1706)
    at java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1344)
    at java.io.ObjectInputStream.readObject(ObjectInputStream.java:370)
    at org.apache.spark.serializer.JavaDeserializationStream.readObject(JavaSerializer.scala:62)
    at org.apache.spark.serializer.JavaSerializerInstance.deserialize(JavaSerializer.scala:81)
    at org.apache.spark.scheduler.DirectTaskResult.value(TaskResult.scala:79)
    at org.apache.spark.scheduler.TaskSetManager.handleSuccessfulTask(TaskSetManager.scala:533)
    at org.apache.spark.scheduler.TaskSchedulerImpl.handleSuccessfulTask(TaskSchedulerImpl.scala:353)
    at org.apache.spark.scheduler.TaskResultGetter$$anon$2$$anonfun$run$1.apply$Volatile(TaskResultGetter.scala:70)
    at org.apache.spark.scheduler.TaskResultGetter$$anon$2$$anonfun$run$1.apply(TaskResultGetter.scala:49)
    at org.apache.spark.scheduler.TaskResultGetter$$anon$2$$anonfun$run$1.apply(TaskResultGetter.scala:49)
    at org.apache.spark.scheduler.TaskResultGetter$$anon$2$$anonfun$run$1.apply(TaskResultGetter.scala:49)
    at org.apache.spark.scheduler.TaskResultGetter$$anon$2.run(TaskResultGetter.scala:48)
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145)
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615)
    at java.lang.Thread.run(Thread.java:745)

Exception in thread "qtp807507959-198" java.lang.OutOfMemoryError: GC overhead limit exceeded
    at java.util.concurrent.ConcurrentHashMap$KeySet.iterator(ConcurrentHashMap.java:1428)
    at org.eclipse.jetty.io.nio.SelectorManager$SelectSet$1.run(SelectorManager.java:712)
    at org.eclipse.jetty.util.thread.QueuedThreadPool.runJob(QueuedThreadPool.java:608)
    at org.eclipse.jetty.util.thread.QueuedThreadPool$3.run(QueuedThreadPool.java:543)
    at java.lang.Thread.run(Thread.java:745)
```

【原因分析】

1. 通常是在 Driver 中进行了不合适的数据处理（例如把一个巨大的数据集想在 Driver 中输出等）或 Driver 占用内存指定过小。

【解决办法】

1. 对于前者我们需要避免这种处理方式，对于后者可以通过在提交任务时设置 driver-memory 参数为更大内存来解决

[SPARK-30007] Container 运行 memory 超限导致 container 挂掉

【问题背景与现象】

任务运行失败，其中部分 Container 由于 memory 超限导致 container 挂掉。

【原因分析】

1. Executor 日志中有如下打印：

```
ERROR | [SIGTERM handler] | RECEIVED SIGNAL 15: SIGTERM |
org.apache.spark.util.SignalLoggerHandler.handle(SignalLogger.scala:57)
```

2. 在 NM 日志中有类似如下打印：


```
WARN | Container Monitor | Container
[pid=97806,containerID=container_1444732058587_5116_01_000003] is running
beyond virtual memory limits. Current usage: 1GB of 1 GB physical memory used; 2.5
GB of 2.5 GB virtual memory used. Killing container.
```

3. 在 Container 运行过程中，yarn 默认会监控 container 进程的 physical mem 和 virtual mem，任何一个超限，都会导致 Container 被 yarn 杀掉。

【解决办法】

对于 Virtual 内存超限，通常以下两种方式可解决：

1. 增加 executor 内存，在提交任务时，使用 `--executor-memory 2g` 来指定 executor 大小；
2. 提升虚拟内存和物理内存的比例 `yarn.nodemanager.vmem-pmem-ratio`，此配置在 FI 的 Yarn 服务中可查到相关配置进行修改，然后重启 yarn 服务使之生效。

对于 Physical 内存超限，通常以下两种方式可解决：

1. 增加 executor 内存，在提交任务时，使用 `--executor-memory 2g` 来指定 executor 大小；
2. 增大配置参数 `spark.yarn.executor.memoryOverhead` 来提升 memoryOverhead 内存的大小，默认为执行器内存的 10%。

[SPARK-30008] Container 运行 OOM 异常导致 container 挂掉

【问题背景与现象】

任务运行失败，其中部分 Container 由于 OutOfMemory 异常导致 container 挂掉。

【原因分析】

1. 在 Spark 任务运行失败，Executor 进程日志中可能打印如下异常：

```
java.lang.OutOfMemoryError: Java heap space
-XX:OnOutOfMemoryError="kill %p"
Executing /bin/sh -c "kill 200016"...
```

2. Executor 内存超限，直接被 JVM 杀掉

【解决办法】

1. 提高 executor 的启动内存