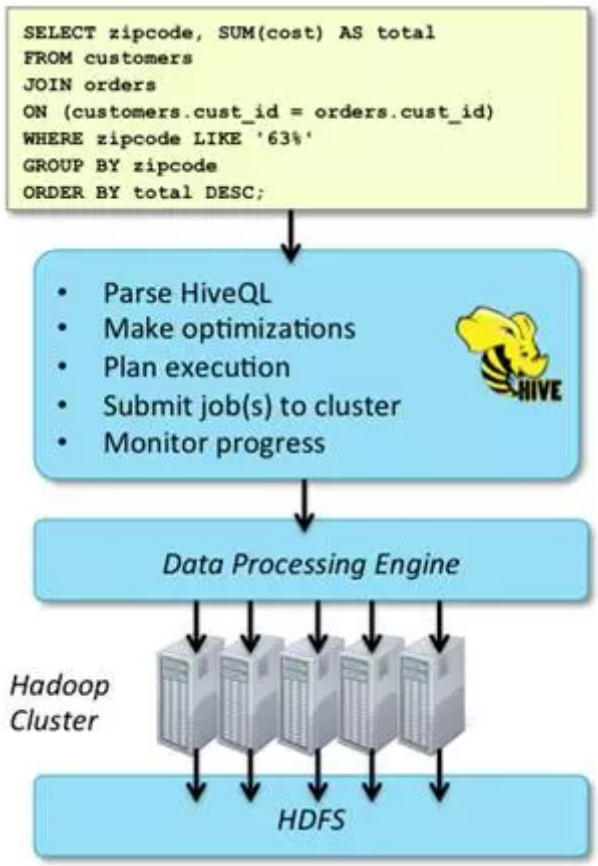
Hive必备手册，这8点你一定用得到！

2016-04-11 丁晔磊 Cloudera中国

Apache Hive是Hadoop的一个数据仓库系统，促进了数据的综述（将结构化的数据文件映射为一张数据库表）、即席查询以及存储在Hadoop兼容系统中的大型数据集分析。本文主要介绍了Hive概念及一些实例。

Hive基本概念

Apache Hive在MapReduce上提供了一个SQL引擎层，是Facebook开发并开源的一个Apache项目。Apache Hive支持HiveQL语言，是SQL-92的一个子集。Hive只是提供了一个SQL的解析，具体的执行依赖于底层的执行引擎，比如MapReduce。Hive将HiveQL查询转换为一系列的MapReduce作业，提交到集群中运行，如下图所示：

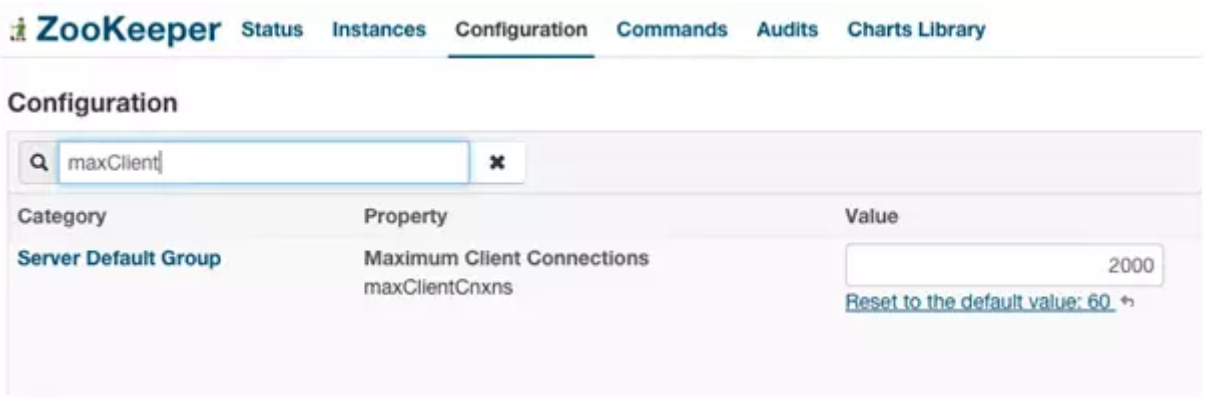


Hadoop基本配置

Hive依赖的组件包括Yarn、HDFS与Zookeeper。为了Hive的正常工作，Cloudera对相关组件有一些推荐配置，主要包括角色并发性、任务资源配置等。

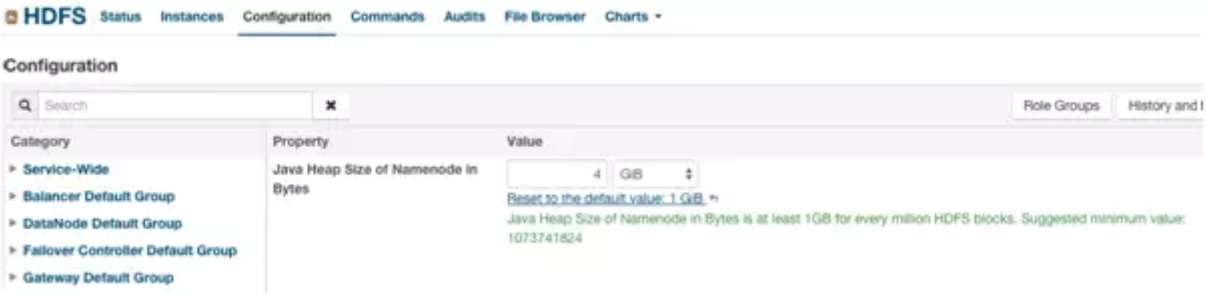
Zookeeper基本配置

配置并发连接数，默认值是60，在高并发的情况下导致访问集群连接失败(connection refused)的错误。建议调整到2000。

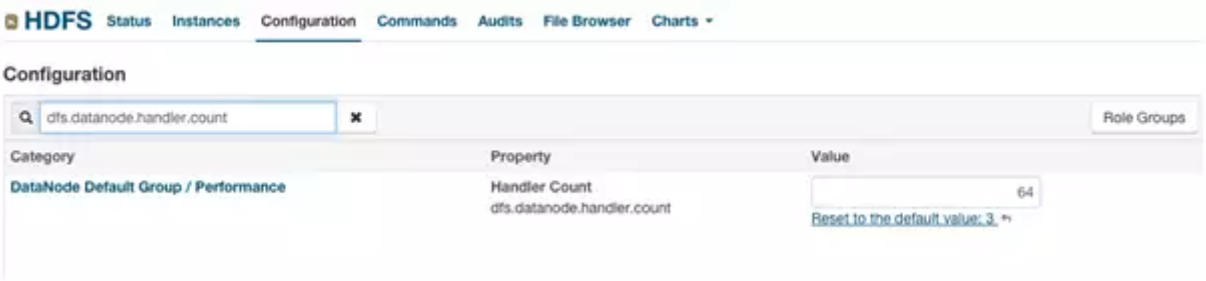


HDFS基本配置

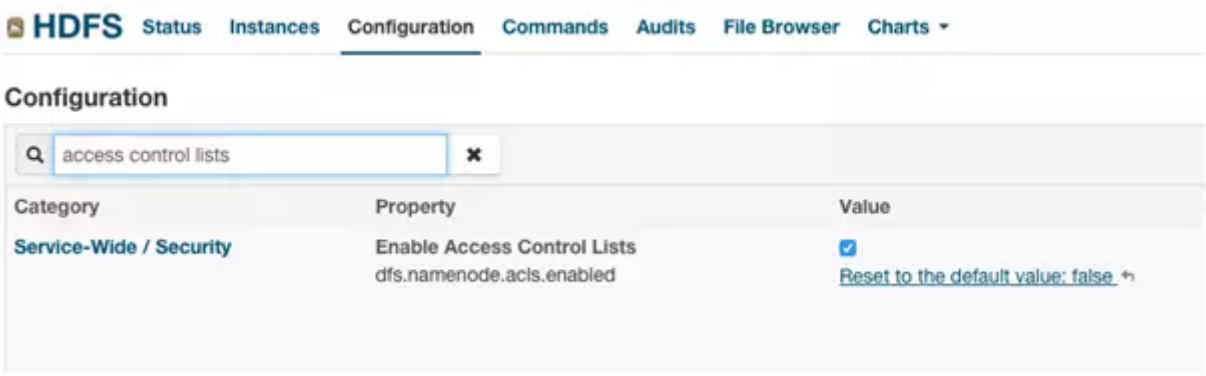
调整NameNode内存的使用量 (同时相应调整Secondary NameNode内存的使用量)，默认值是1GB。建议调整到至少4GB。



调整DataNode的Handler的数量，默认值是3。建议调整到32或者64。

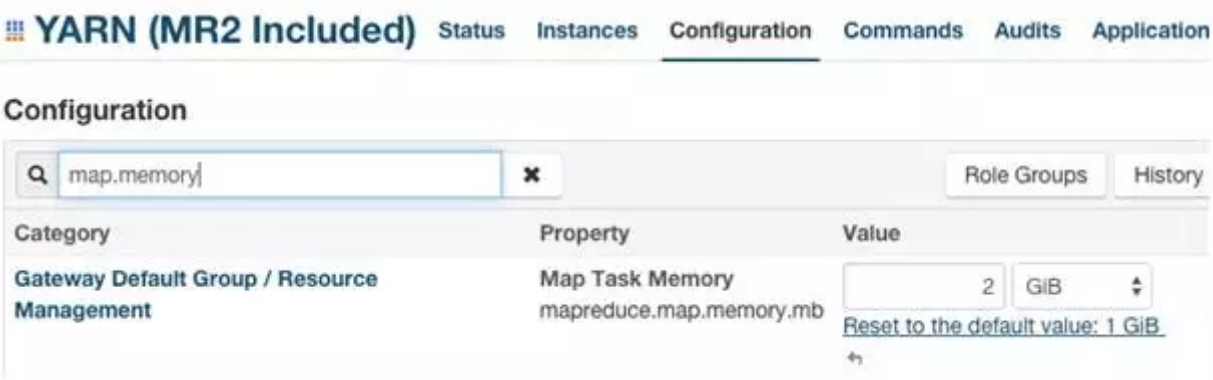


开启HDFS ACLs (Access Control Lists)，默认是关闭的。建议开启，可以更加灵活地管理HDFS文件系统的访问权限。

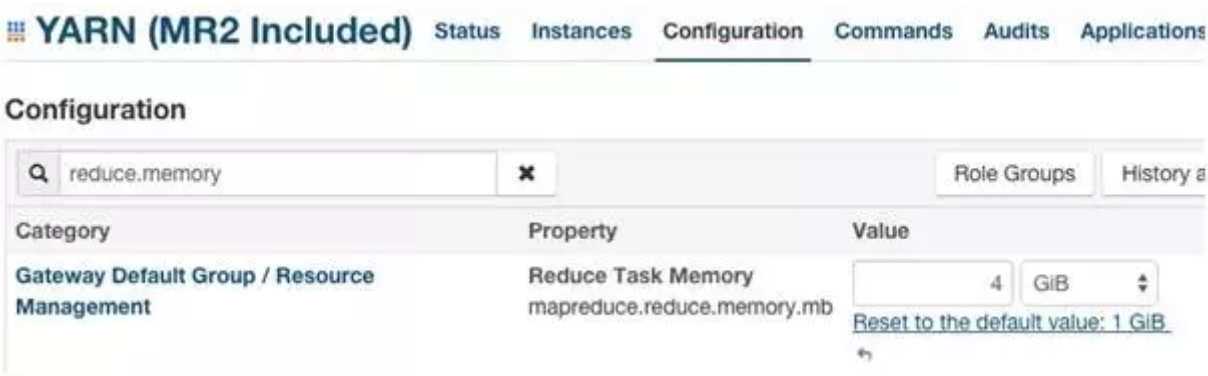


Yarn基本配置

调整Map Container的内存使用量，默认值是1GB。推荐使用2GB作为起始点，根据应用程序的运行状况进行调整。

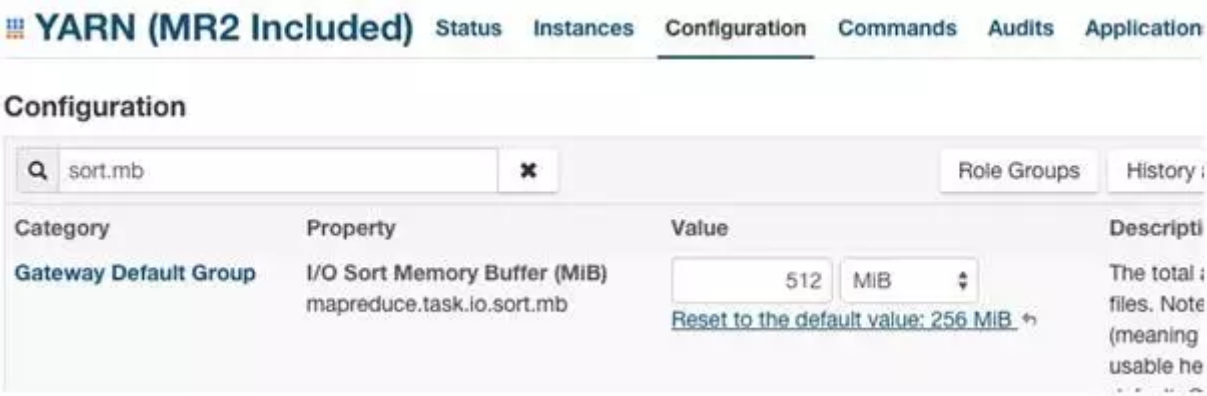


调整Reduce Container的内存使用量，默认值是1GB。推荐使用4GB作为起始点，根据应用程序的运行状况进行调整。



注意对于以上两个参数的调整，需要调整相应的java.opts参数以设置JVM的内存堆栈大小。

调整Map/Reduce任务(排序用)内存缓冲区大小，默认值是256MB。推荐使用512MB作为起始点，根据应用程序的运行状况进行调整。

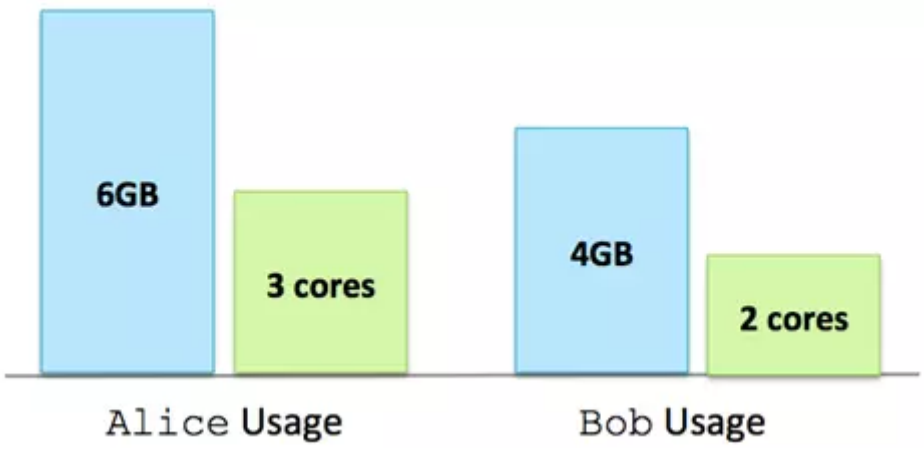


Yarn调度算法

Yarn的(Fair Scheduler)调度算法主要有3类：DRF (Dominant Resource Allocation)、FAIR、FIFO (First In First Out)。默认使用DRF，利用CPU以及Memory实现资源的公平调度。FAIR算法与DRF相似，资源是在多任务之间共享的，区别在于FAIR算法仅仅根据Memory实现资源的调度。FIFO算法，顾名思义，提交到某一个资源池的多个任务根据提交的时间顺序依次运行，后提交的任务在先提交的任务执行完毕前不会被调度(即Resource Manager不会为后提交的任务分配运行Application Master的Container)。

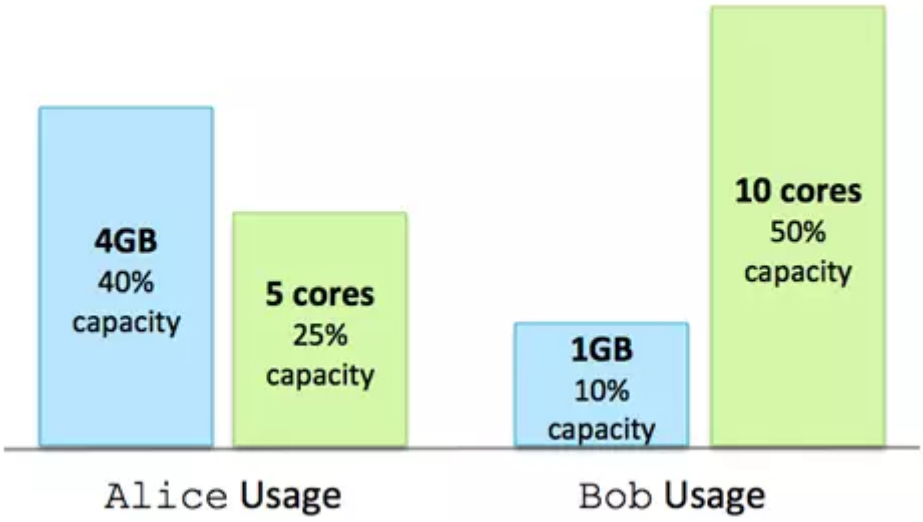
DRF调度算法举例，假设两个用户Alice与Bob分别提交任务：

情况一：Alice获得6GB内存、3个CPU核；Bob获得4GB内存、2个CPU核



该情况下，无论是CPU 还是Memory，Alice都具有绝对优势，因此Resource Manager在分配下一个Container时会先满足Bob的资源请求。

情况二：集群总共有10GB内存、20个CPU核。Alice获得4GB内存、5个CPU核；Bob获得1GB内存、10个CPU核



该情况下，Alice拥有更多的内存(40% > 10%) 而Bob拥有更多的CPU核 (50% > 25%)。但是相比而言 50% > 40%，因此Resource Manager在分配下一个Container时会先满足Alice的资源请求。

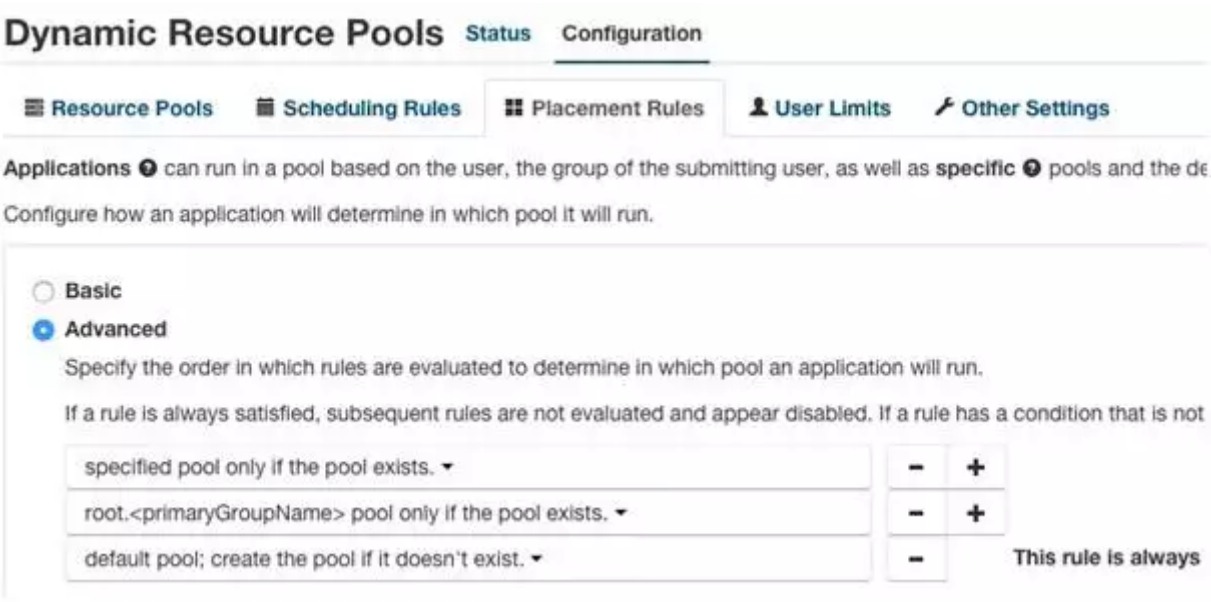
Yarn动态资源池

任何Yarn应用程序的运行都是在动态资源池中进行的，至于具体使用哪个资源池，用户可以手工指定(比如MapReduce应用程序，通过参数mapreduce.job.queuename确定)，也可以通过Resource Manager的“Placement Rules”自动分配。

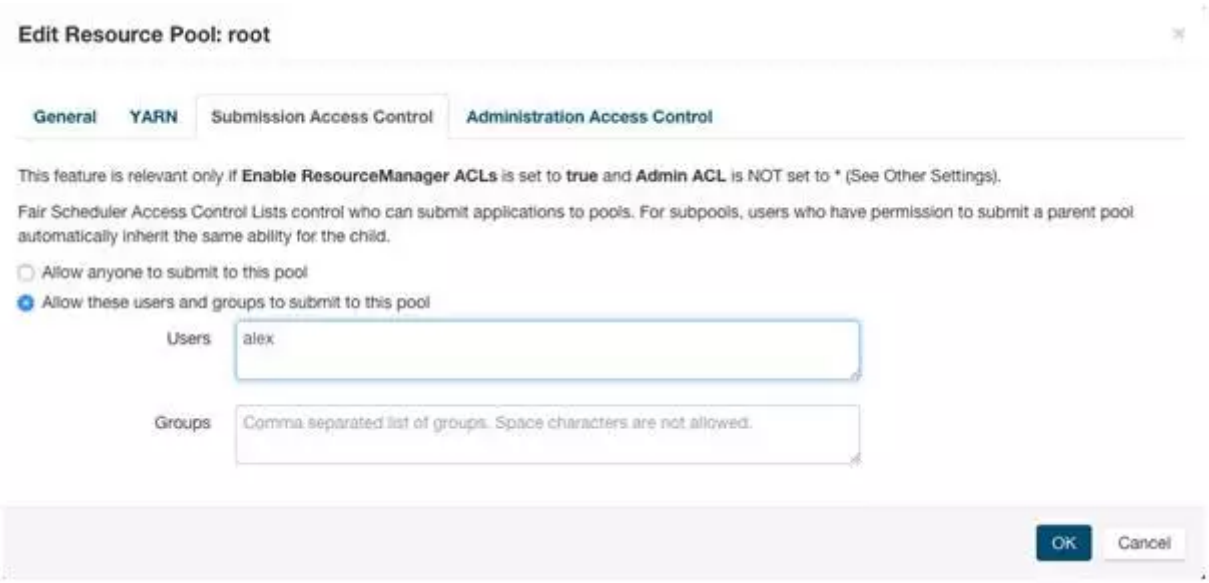
默认情况下，“Placement Rules”使用root.USER_NAME。例如用户alex提交一个MapReduce作业，在没有显示使用mapreduce.job.queuename的条件下，Resource Manager会自动将该作业放置到资源池root.alex中。若root.alex资源池尚未创建，Resource Manager会自动(动态)创建该资源池。重启Yarn服务后，所有动态创建的资源池会被自动删除。

使用动态创建的资源池并不能非常严格地控制资源的使用。推荐使用“Placement Rules”的高级配置，预分配后续需要的资源池。如下图所示，Resource Manager资源池选择过程为：

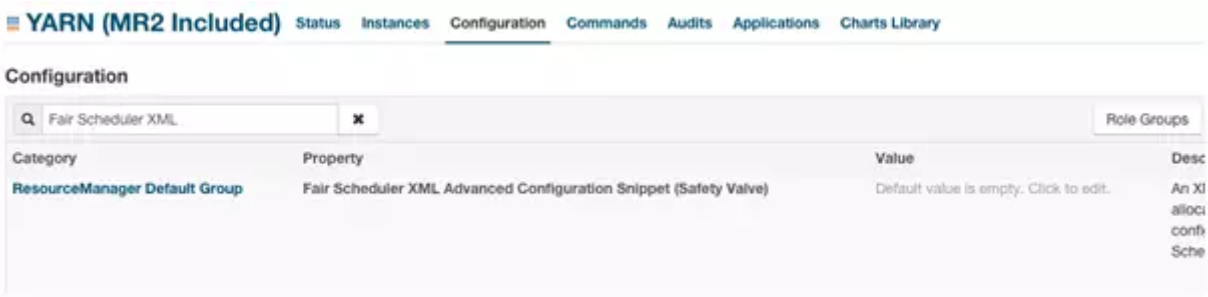
判断用户alex:prod是否显示设置参数mapreduce.job.queuename
➞ 是并且该资源池预定义，在指定资源池中运行
➞ 否，判断资源池root.prod是否预定义
➞ 是，在root.prod资源池中运行
➞ 否，在default资源池中运行



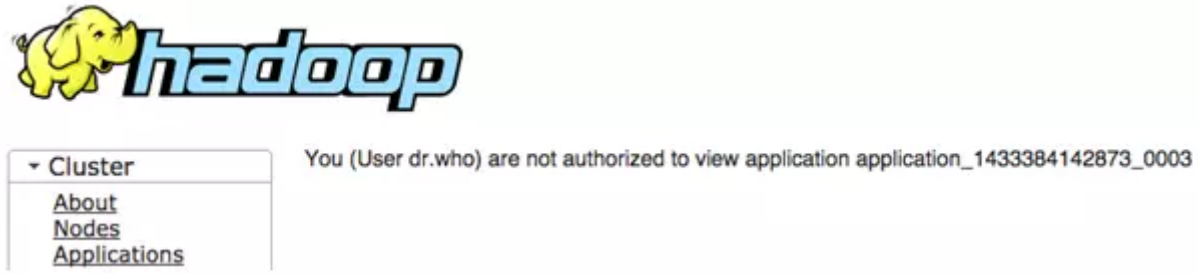
用户允许为每个资源池配置不同的调度算法、资源的约束(资源池权重、CPU核数、内存数量、最大运行应用程序个数)、提交访问控制(Submission Access Control)、管理访问控制(Administration Access Control)。子资源池会自动继承父资源池的提交访问控制以及管理访问控制。例如，如果允许用户alex向root资源池提交应用程序，那么该用户可以向任意root的子资源池，比如root.dev、root.prod，提交应用程序。



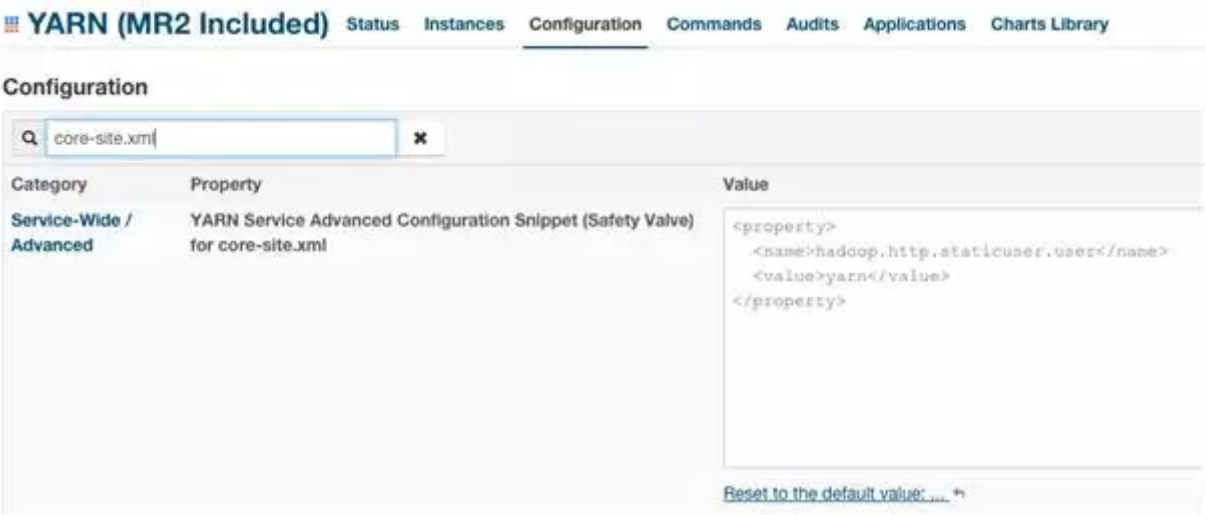
资源池的属性是允许动态修改的，保存在fair-scheduler.xml文件中。Resource Manager每10秒会自动读取该文件，刷新资源池的属性。如果需要人为编写fair-scheduler.xml，使用Yarn配置参数“Fair Scheduler XML Advanced Configuration Snippet (Safety Valve)”：



设置资源池提交访问控制、管理访问控制后，用户通过Yarn 内嵌管理页面查看应用程序时可能会遇到如下问题：

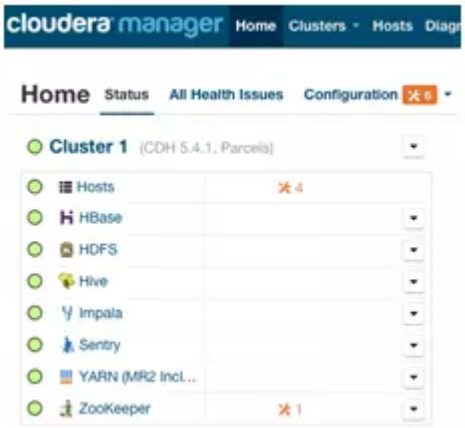


加入参数hadoop.http.staticuser.user：

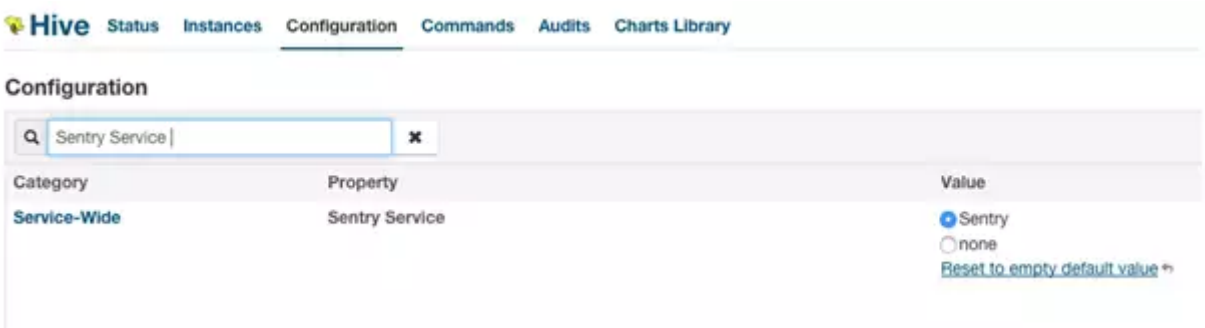


Sentry授权

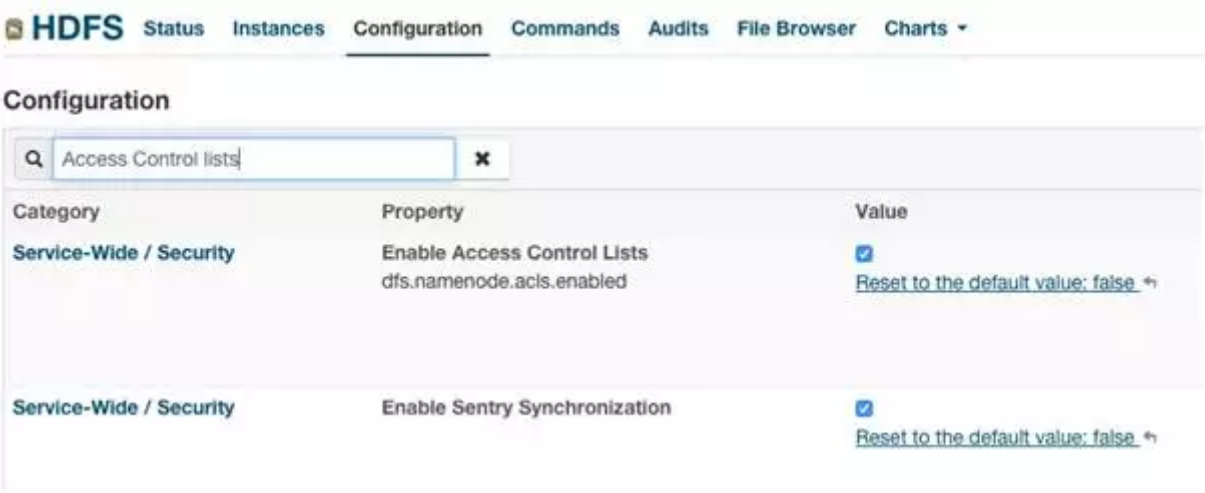
虽然推荐在Kerberos模式下开启Sentry授权，但是用户也允许在普通模式下使用Sentry。安装Sentry可以采用常规“Add a Service”的流程。安装完毕后可以在Cloudera Manager主页看到新添加的Sentry服务。



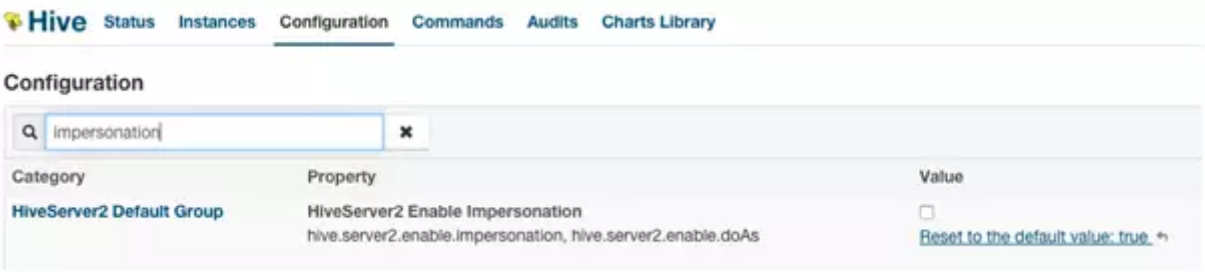
配置Hive采用Sentry授权：



配置HDFS开启ACLs (Access Control Lists) 与SentryHDFS权限同步：



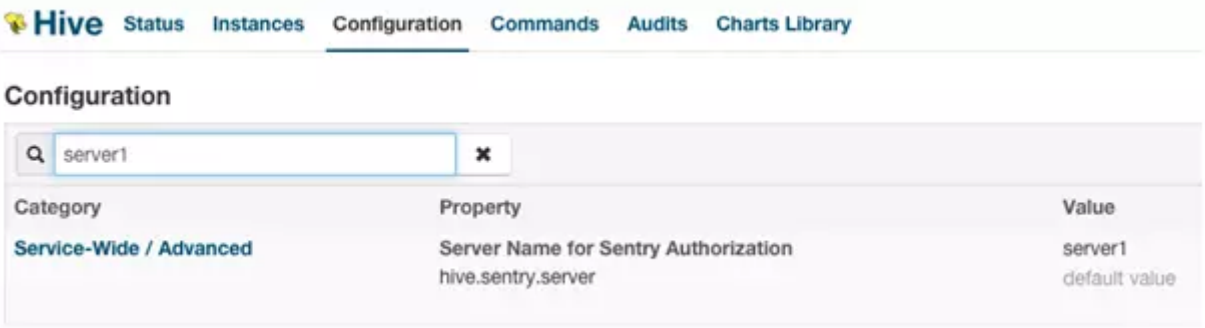
关闭Hive Impersonation功能：



为了在非Kerberos环境下使用Sentry授权，需要设置参数sentry.hive.testing.mode为true (加入Hive Service Advanced Configuration Snippet for sentry-site.xml中)。

Sentry授权模型

Sentry的授权是基于对象进行的，最常用的包括服务器(Server)、数据库(Database)、表(Table)、视图(View)。后面三者比较好理解，第一个Server指代整个Hive Service，默认值是server1：



目前最新版本CDH所包含的Hive并不支持基于列的授权。因此如果需要实现列级别的授权，可以使用视图(view)。Sentry的权限有三种：INSERT、SELECT、ALL：

权限	对象
INSERT	DB、TABLE
SELECT	DB、TABLE
ALL	SERVER、TABLE、DB、URI

一般Sentry授权流程：超级用户将“权限”授权给“角色”à超级用户将“角色”授权给“用户组”。用户组中的所有用户享用赋予的权限，这里提到的用户组可以是用户的Primary Group，也可以是Secondary Groups。另外，还需要注意的是，Sentry中“角色”只能授权给“用户组”，并不能直接授权给特定“用户”。

Sentry授权实例

案例：某公司数据中心部门在Hive中部署数据库sjzx，现在希望对两类用户——管理员与开发员——实现权限的控制。管理员对数据库sjzx享有所有权限；开发员对数据库sjzx仅享有只读权限。

步骤1：在集群中新建用户/用户组

步骤2：HDFS超级用户hdfs创建用户工作目录

```
hdfsdfs -mkdir /user/sjzx
hdfsdfs -chownsjzx:sjzx /user/sjzx
hdfsdfs -mkdir /user/sjzx_dev
hdfsdfs -chownsjzx_dev:sjzx_dev /user/sjzx_dev
```

步骤3：HDFS超级用户hdfs对“Hive内部表数据存储目录”设置正确的权限

```
hdfsdfs -chmod -R 771 /user/hive/warehouse
hdfsdfs -chown -R hive:hive /user/hive/warehouse
```

步骤4：Hive超级用户hive创建数据库并授权

```
beeline>create role admin;
beeline>grant all on server to role admin;
beeline> grant role admin to group hive;
beeline> create database sjzx;
beeline> create role sjzx_dev;
beeline> grant role sjzx_dev to group sjzx_dev;
beeline> grant select on database sjzx to role sjzx_dev;
beeline> create role sjzx;
beeline> grant role sjzx to group sjzx;
beeline> grant all on database sjzx to role sjzx;
```

步骤5：数据中心管理员sjzx验证

```
#权限验证
beeline>use sjzx;

beeline>create table t1 (name string, company string) row format delimited fields terminated by ',';
beeline>show tables;
Query: show tables
+-----------+--+
| tab_name  |
+-----------+--+
| t1        |
+-----------+--+

# 编写测试数据加入内部表 sjzx.t1，例如“alex,cloudera”
#数据查询验证
beeline>select * from t1;
+----------+------------+--+
| t1.name  | t1.company |
+----------+------------+--+
| alex     | cloudera   |
+----------+------------+--+
```

步骤6：数据中心开发员sjzx_dev验证

```
beeline>use sjzx;

beeline>show tables;
+-----------+--+
| tab_name  |
+-----------+--+
| t1        |
+-----------+--+

beeline>select * from t1;
+----------+------------+--+
| t1.name  | t1.company |
+----------+------------+--+
| alex     | cloudera   |
+----------+------------+--+

beeline>create table t2 (name string, company string) row format delimited fields terminated by ',';
Error: Error while compiling statement: FAILED: SemanticException No valid privileges
 Required privileges for this query: Server=server1->Db=sjzx->action=*; (state=42000,code=40000)
```

Sentry授权是通过HiverServer2实现的，因此为了使用授权功能就必须使用beeline客户端 (Hive CLI并不经过HiveServer2，因此会绕过授权管理部分，通过步骤3在HDFS文件系统级别屏蔽Hive CLI)。

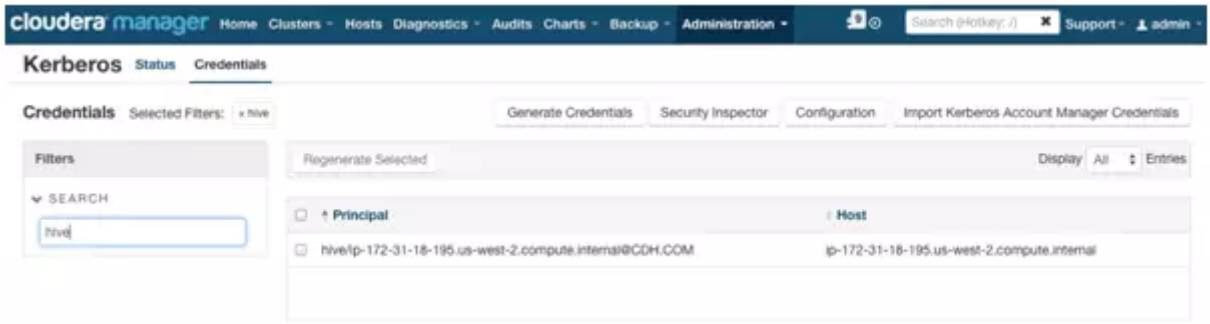
Kerberos认证

Hadoop安全包括认证、授权和审计。其中认证是安全的第一步，是授权和审计的基础。Hadoop默认情况下采用Linux的用户、用户组验证HDFS文件权限，Yarn动态资源池的权限等。如果要实现强认证，推荐使用Kerberos。当前常用的两种实现是MIT KDC以及Active Directory。前者是Kerberos开源实现版本；后者不仅实现了Kerberos协议，同时也实现了LDAP协议。

Hive的认证其实分为两个阶段：第一阶段，客户端到HiveServer2的认证，允许使用Kerberos、LDAP实现认证；第二阶段，HiveServer2到HDFS、Yarn等Hadoop后台服务的认证，使用Kerberos实现认证。当集群开启

Kerberos认证后，两个阶段都默认使用Kerberos进行认证，其中第一阶段使用用户自己的Kerberos账户，第二阶段使用运行HiverServer2的Kerberos账户。

在Cloudera Manager页面，管理员可以看到HiveServer2的Kerberos账户：



运行beeline前，普通用户需要使用其Kerberos账户进行登录，例如：

```
kinitalex
Password for alex@CDH.COM:

klist
Ticket cache: FILE:/tmp/krb5cc_500
Default principal: alex@CDH.COM

Valid starting     Expires            Service principal
06/04/15 13:20:34  06/04/15 23:20:40  krbtgt/CDH.COM@CDH.COM
        renew until 06/11/15 13:20:34
```

否则在使用beeline连接HiveServer2的过程中，会出现错误：

```
15/06/04 13:32:11 [main]: ERROR transport.TSaslTransport: SASL negotiation failure
javax.security.sasl.SaslException: GSS initiate failed [Caused by GSSException: No valid credentials provided (Mechanism level: Failed to find any Kerberos tgt)]
```

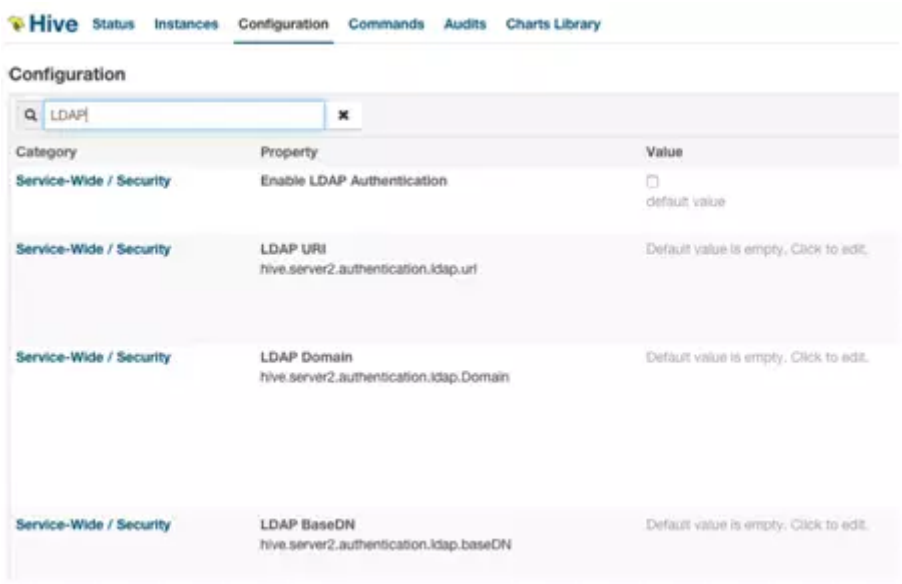
在启动Kerberos后，beeline连接HiveServer2时，在指定的JDBCURL路径末尾添加HiveServer2的principal名称：

```
beeline
Beeline version 1.1.0-cdh5.4.1 by Apache Hive
beeline> !connect jdbc:hive2://ip-172-31-18-195.us-west-2.compute.internal:10000/default;principal=hive/ip-172-31-18-195.us-west-2.compute.internal@CDH.COM
scan complete in 4ms
Connecting to jdbc:hive2://ip-172-31-18-195.us-west-2.compute.internal:10000/default;principal=hive/ip-172-31-18-195.us-west-2.compute.internal@CDH.COM
Enter username for jdbc:hive2://ip-172-31-18-195.us-west-2.compute.internal:10000/default;principal=hive/ip-172-31-18-195.us-west-2.compute.internal@CDH.COM:
Enter password for jdbc:hive2://ip-172-31-18-195.us-west-2.compute.internal:10000/default;principal=hive/ip-172-31-18-195.us-west-2.compute.internal@CDH.COM:
Connected to: Apache Hive (version 1.1.0-cdh5.4.1)
Driver: Hive JDBC (version 1.1.0-cdh5.4.1)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://ip-172-31-18-195.us-west-2.co>
```

beeline连接过程中会提示用户输入用户名(username)、密码(password)，直接回车即可 (实际HiveServer2认证时只会判断启动beeline用户的Kerberos token)。

LDAP认证

在Kerberos环境下，客户端到HiveServer2的认证还可以通过LDAP的方式进行。默认情况下，Hive中LDAP的配置都是“Service-Wide”的。



从参数的命名可以发现，LDAP相关的设置都是基于HiveServer2的，即允许用户启动多个HiveServer2实例(instance)，不同实例采用不同的认证方式，提高Hive的使用人群范围。比如，在Kerberos环境中，用户必须拥有Kerberos凭证才可以使用Hive服务，但是如果用户拥有了Kerberos凭证，那么该用户就被允许使用Hadoop中所有的服务了，如果只希望将Hive服务暴露给外部用户使用，可以采用LDAP认证的方式。

LDAP协议的两种常用实现是OpenLDAP和Active Directory。因此针对不同的实现，需要配置不同的参数：

以Windows Server 2008 R2 Active Directory为例，需要配置上述参数列表中的前三项。修改HiveServer2配置，覆盖Kerberos认证方式：

重启HiveServer2实例后，就可以使用beeline进行访问了(无需Kerberos Token)，由于采用LDAP认证，连接HiveServer2时直接使用普通的JDBC URL即可。根据LDAP中配置的用户名/密码登录。

Hive/Impala互操作

Hive与Impala本身是共享MetaStore的，即Hive(或者Impala)对表的操作都会反映到Impala(或者Hive)中。Hive与Impala的授权都是通过Sentry实现的，进而提供了一个统一的SQL授权访问层 (推荐Hive/Impala设置统一的认证方式)。

从性能角度出发，推荐使用Parquet文件存储格式，这种列式存储方式可以有效降低table scan过程中读取数据的数量。实际操作中，使用Impala生成Parquet数据时，每个输出文件的大小基本一致 (~256MB)。但是如果利用Hive生成Parquet数据，例如insert overwrite table t1_parquet select * from t1_text，每个Parquet文件会出现过小、大小不均匀等问题。需要对一些参数进行调整以解决这一系列问题：

参数	描述
mapreduce.input.fileinputformat.split.maxsize	Mapper 任务处理的 InputSplit 最大值
mapreduce.input.fileinputformat.split.minsize	Mapper 任务处理的 InputSplit 最小值
dfs.blocksize	底层 HDFS 存储时 Block 的大小
parquet.block.size	生成 Parquet 时使用内存缓冲区大小
parquet.compression	压缩算法

运行hive生成Parquet示例：

```
setdfs.blocksize=268435456;
setparquet.compression=snappy;

#不同文件的压缩率不同，需要调整以下3个参数以保证最终生成的文件小于一个HDFS block
setmapred.min.split.size=805306368;
setmapred.max.split.size=805306368;
setparquet.block.size=536870912;

drop table if exists hive_demo;
create table hive_demo
stored as parquet
as
select * from t1;
```

如果生成的Impala小文件超过一个HDFS Block (执行时需要通过网络读取超出的部分)，那么在运行Impala查询时会出现警告：

```
WARNINGS: Parquet files should not be split into multiple hdfs-blocks.
file=hdfs://../warehouse/hive_demo/000000_0 (1 of 5 similar)
```

如需源文件，请留下您的邮箱，我们会尽快发给您！