# mdrill安装指南

目录

[mdrill安装指南 1](#_Toc376437183)

[部署目标 2](#_Toc376437184)

[推荐硬件 2](#_Toc376437185)

[源码编译 2](#_Toc376437186)

[安装依赖环境 2](#_Toc376437187)

[环境变量配置 3](#_Toc376437188)

[mdrill配置 4](#_Toc376437189)

[离线数据的准备（如果要配置实时数据源，可以跳过此步） 4](#_Toc376437190)

[配置数据表 5](#_Toc376437191)

[配置实时数据源（如果使用离线模式，请跳过此步） 6](#_Toc376437192)

[mdrill的启动 7](#_Toc376437193)

[创建离线索引（如果要配置实时数据源，请跳过此步骤） 8](#_Toc376437194)

[启动表 8](#_Toc376437195)

[停止表 9](#_Toc376437196)

[导入实时数据（如果是离线模式，请忽略） 9](#_Toc376437197)

[如何重启服务 9](#_Toc376437198)

修改记录

* 2013年6月6日

子落 创建文档

* 2013年7月3日

子落 配置整理

* 2014年1月2日

子落 添加实时导入的配置

## 部署目标

* 总数据量500亿，100~400个维度.
* 单表最大存储500亿。
* 数据表总数小于10。
* 单表每天数据增量小于1亿。
* 单次扫描最大时间跨度为该时间跨度下数据量小于30亿行。
* 单次扫描的列的总数不超过12列。

## 推荐硬件

* 机器数量：10~12台
* CPU:E5-2630(6核)\*2
* 内存：>=48GB
* 硬盘规格数量：>=480GB\*12 （SSD最佳，普通磁盘也可）。

## 源码编译

在linux环境下使用maven编译

mvn clean && mvn package assembly:assembly

编译好的程序在/target/alipay-bluewhale.tar.gz 下

## 安装依赖环境

* zookeeper安装

zookeeper:<http://zookeeper.apache.org/>

推荐使用zookeeper-3.4.5版本

zoo.cfg的如下几个参数配置

tickTime=2000

initLimit=10

syncLimit=5

maxClientCnxns=300

maxSessionTimeout=20000

另外zk运行久了，硬盘占用会越来越多，可以使用如下命令清理

cd /xxxx/zookeeper

java -cp

./zookeeper-3.4.5.jar:./lib/log4j-1.2.15.jar:./lib/slf4j-api-1.6.1.jar:./conf

org.apache.zookeeper.server.PurgeTxnLog /xxx/data/zook /xxx/data/zook -n 100

* 部署hadoop环境

hadoop:<http://hadoop.apache.org/>

* ZeroMQ 2.1.7

tar -xzf zeromq-2.1.7.tar.gz

cd zeromq-2.1.7

./configure

make

sudo make install

* JZMQ

./autogen.sh

./configure

make

sudo make install

安装完成后 记得执行sudo ldconfig 刷新动态库

* Java 6
* Python 2.6.6
* unzip

## 环境变量配置

* 每台机器配置好域名hostname
* shell环境变量，编辑~/.bashrc

export HADOOP\_CONF\_DIR=$HOME/config

export HADOOP\_HOME=/home/hadoop/hadoop-current

export HADOOP\_ROOT=/home/hadoop

export HADOOP\_HEAPSIZE=128

export PATH=$PATH:$HADOOP\_HOME/bin

export PATH=$PATH:/usr/java/jdk1.6.0\_13/bin

export JAVA\_HOME=/usr/java/jdk1.6.0\_13

* 使当前环境下的环境变量生效

source ~/.bashrc

* 端口范围

修改你机器的配置

cat /proc/sys/net/ipv4/ip\_local\_port\_range

1024    65535

修改成 10000  65535

如下的修改请参考这个地址

<http://wiki.eclipse.org/Jetty/Howto/High_Load#Jetty_Tuning>

修改机器允许打开的文件数量

修改机器允许listen的个数

sudo sysctl -w net.core.somaxconn=4096

sudo sysctl -w net.core.netdev\_max\_backlog=16384

sudo sysctl -w net.ipv4.tcp\_max\_syn\_backlog=8192

sudo sysctl -w net.ipv4.tcp\_syncookies=1

## mdrill配置

将编译好的/target/alimama-adhoc.tar.gz 解压

编辑/xxxx /alimama/adhoc-core/conf/storm.yaml

一般我们在每台机器上配置6个shard，1个merger server,均分配5GB的内存

也可以根据实际情况进行修改

## 离线数据的准备（如果要配置实时数据源，可以跳过此步）

* mdrill默认处理的是sequencefile格式的数据，对key没有要求，value则为对应数据表中一条记录，记录的列与列之间的分隔符为默认\001

如果使用文本格式的，或者分隔符不是\001，那么后面的创建索引逻辑里要注意传递的参数别写错了

* mdrill要求数据按照天进行分目录

目录命名必须为dt=yyyyMMdd这种格式（你懂的），参考如下



* 数据的列中必须含有一列叫thedate,其值与其所在的目录dt=yyyyMMdd中的yyyyMMdd相等，thedate实际上为mdrill的分区字段，任何的查询都必须指定thedate分区

如果数据列中的thedate与目录中的dt=yyyyMMdd不相等，那么mdrill以目录的时间为准确时间。

* 关于分区的说明

mdrill默认必须按照日期分区，使用分区后总存储的数据量会显著提升，每天创建的索引也是增量的，如果确实因为某种原因无法按照日期分区，可以将分区类型设置为single，但是总的数据存储量会减少很多（10台机器我真实的案例跑过30亿），每天创建的也会是全量索引，每天都会从hadoop集群copy全量的索引。

* 列的数据类型目前只支持4种

string :字符串类型

tlong :long整形

tdouble :浮点型

tdate :日期类型

text:字符串类型（进行了分词，意味着只能进行全文检索，而不能参与统计与分组）

另外如果数据类型用\_mdrill\_进行了分割，后面的值表示是否存储，在全文检索模式中特别有用，全文检索模式要把要进行展示的列配置成存储（速度能提升很多）

举例如下  
下面这几个为 只索引而不存储数据的值

string\_mdrill\_false

tlong\_mdrill\_false

tdouble\_mdrill\_false

tdate\_mdrill\_false

text\_mdrill\_false

下面这几个为 只既索引也存储数据的值

string\_mdrill\_true

tlong\_mdrill\_true

tdouble\_mdrill\_true

tdate\_mdrill\_true

text\_mdrill\_true

另外如果要跳过某些列，可以将数据类型配置成ignored，表示这个列既不存储也不索引。

## 配置数据表

* 首先要确定数据表在hdfs上的根目录，比如说

/group/taobao/external/p4p/p4padhoc/tablelist

这个目录确定好了确保编辑/xxxx/alimama/adhoc-core/conf/storm.yaml的mdrill配置文件里也配置好了

在该目录下可以有一个至多个数据表

参考如下



每个数据表目录下会有三个子目录

solr ：表的配置目录

index：mdrill的索引在hdfs中的存储路径

tmp:临时目录

* 准备建表语句，示例如下

CREATE TABLE fact\_seller\_all\_d (  
thedate string,  
shopid string,  
category\_name string,  
main\_cat\_name1 string,  
is\_p4pvip string,  
p4p\_e\_alipay\_outshop\_cnt tlong,  
p4p\_e\_alipay\_outshop\_amt tdouble,  
p4p\_e\_alipay\_direct\_cnt tlong,  
p4p\_e\_alipay\_direct\_amt tdouble,  
p4p\_e\_alipay\_indirect\_cnt tlong,  
p4p\_e\_alipay\_indirect\_amt tdouble,  
p4p\_e\_gmv\_indirect\_cnt tlong,  
p4p\_e\_gmv\_outshop\_cnt tlong

)

注意

1.建表语句要与数据存储顺序一致

2.thedate:是分区的字段，必须存在，格式为yyyyMMdd格式的字符串,且要与对应的分区目录要对应

* 将建表语句存储成文本文件，执行

cd alimama/adhoc-core/bin/

chmod 777 ./bluewhale

./bluewhale mdrill create ./create.sql

如果需要经常的进行全文检索，那么可以配置成存储明细，使用额外的存储，换取全文检索的时间

./bluewhale mdrill create ./create.sql **true**

## 配置实时数据源（如果使用离线模式，请跳过此步）

编辑storm.yaml配置文件，假设要配置为实时的表的名字为p4p\_v2

1. 先配置editlog记录的位置，可以选择为local与hdfs

hdfs模式没有经过严格的测试，如果你的hadoop集群不是经常的停机维护或者出问题，那么可以考虑使用hdfs模式，这样因为机器宕机或者硬盘损坏导致的数据丢失损失会减少到最小

使用local模式表示editlog会记录到本地的硬盘上，如果这台机器硬件损坏或者宕机，那么那些还没有同步到hdfs中的索引数据会丢失（默认同步间隔为4小时，配置目前为硬编码，请参考com.alimama.mdrill.utils.UniqConfig里面的设置）

mdrill的editlog的代码并非本人开发，而是直接将hadoop的namenode的editlog的实现扣出来，放到了mdrill里，大神们误喷，有稳定的实现是在没必要自己单独写一套了。

higo.realtime.binlog: "local"

1. 需要先将表配置为实时模式，配置方法如下

higo.mode.p4p\_pv2: "@realtime@"

1. 配置表的分区：可以为day,month,default

higo.partion.type.p4p\_pv2: "day"

1. 配置日志解析类

解析类需要实现com.alimama.mdrillImport.DataParser接口，示例程序如下

p4p\_pv2-parse: "com.alimama.quanjingmonitor.mdrillImport.parse.p4p\_pv2"

1. 配置reader

reader用于实时的读取数据，然后交给解析类解析后导入到mdrill里

reader在阿里可以使用tt4(默认已经实现)，阿里外部可以考虑使用metaq或者kafka

需要实现的接口为

com.alimama.mdrillImport.ImportReader.RawDataReader

TT4实现的示例为

com.alimama.quanjingmonitor.mdrillImport.reader.TT4Reader

1. 配置导入模式为local与merger

数据导入到mdrill前，如果数据可以进行简单的合并，那么会大量的减少导入到mdrill里的数据量，合并的方式很简单就是按照解析类的getGroup方法进行分组，getSum里的值进行累加而已，如果我们的数据很难进行合并，则建议使用local模式以减少没必要的传输，如果合并比率很高，那么建议使用merger

p4p\_pv2-mode: "local"

1. 配置各种缓存参数

p4p\_pv2-commitbatch: 5000 //单次向mdrill递交的记录数

p4p\_pv2-commitbuffer: 10000//commit阶段的buffer长度

p4p\_pv2-spoutbuffer: 10000//spout阶段的buffer长度

p4p\_pv2-boltbuffer: 10000 //bolt阶段的buffer长度

1. 配置缓存的刷新时间：与解析类的getTs()方法组合使用

p4p\_pv2-timeoutSpout: 240 //配置在spout的最大缓存时间

p4p\_pv2-timeoutBolt: 60 //配置在bolt里的最大缓存时间

p4p\_pv2-timeoutCommit: 30 //配置在commit阶段的最大缓存时间

完整的配置示例

#----从tt实时导入到mdrill----

higo.mode.p4p\_pv2: "@realtime@"

higo.partion.type.p4p\_pv2: "day"

p4p\_pv2-log: "p4p\_pv2"

p4p\_pv2-subid: "1226132513ZWOWEH3B"

p4p\_pv2-accesskey: "2ac04eb8-b3a4-4fb3-ab89-e8a689501b37"

p4p\_pv2-parse: "com.alimama.quanjingmonitor.mdrillImport.parse.p4p\_pv2"

p4p\_pv2-reader: "com.alimama.quanjingmonitor.mdrillImport.reader.TT4Reader"

p4p\_pv2-commitbatch: 5000

p4p\_pv2-commitbuffer: 10000

p4p\_pv2-spoutbuffer: 10000

p4p\_pv2-boltbuffer: 10000

p4p\_pv2-mode: "local"

p4p\_pv2-timeoutSpout: 240

p4p\_pv2-timeoutBolt: 60

p4p\_pv2-timeoutCommit: 30

## mdrill的启动

1. 将程序和配置分别copy到每台机器
2. 启动nimbus

nimbus是整个mdrill集群任务的总调度，有点类似hadoop的jobtracker

只需要在其中一台机器启动即可，启动命令如下

cd alimama/adhoc-core/bin/

chmod 777 ./bluewhale

nohup ./bluewhale nimbus >nimbus.log &

启动完毕后可以看logs目录查看nimbus的运行日志，一般排错都去看nimbus.log来定位原因

1. 启动supervisor

supervisor用来管理其所在机器的work进程，其角色有点类似hadoop的tasktracker

需要在每台机器上启动，启动命令如下

cd alimama/adhoc-core/bin/

chmod 777 ./bluewhale

nohup ./bluewhale supervisor >supervisor.log &

1. 启动ui和jdbc接口

ui里面暂时了mdrill执行状态，内存使用，任务的分配等

jdbc接口是mdrill对外的统一接口，应用层通过jdbc的方式连接mdrill来进行查询

启动命令如下

cd alimama/adhoc-core/bin/

chmod 777 ./bluewhale

mkdir ./ui

nohup ./bluewhale mdrillui 1107 ../lib/ adhoc-web-{xxx}.war ./ui >ui.log &

启动后，可以通过浏览器打开对应ip的1107端口，看是否能正常打开即可。

## 创建离线索引（如果要配置实时数据源，请跳过此步骤）

使用命令如下  
./bluewhale mdrill index {表名}{hdfs源数据地址} 3650 {起始日期} {数据格式seq|txt，不写默认seq} {分隔符,不写默认\001} {分区下的文件匹配规则默认匹配\*0\* }

给几个使用的示例  
./bluewhale mdrill index rpt\_p4padhoc\_product /group/tbads/p4pdata/hive\_data/rpt/rpt\_p4padhoc\_product 3650 20121001 txt ,

./bluewhale mdrill index fact\_seller\_all\_d /group/tbads/p4pdata/hive\_data/dw/fact\_seller\_all\_d 3650 20010101 seq "\\\"" \*/\*0\*

./bluewhale mdrill index rpt\_seller\_all\_m /group/tbads/p4pdata/hive\_data/rpt/rpt\_seller\_all\_m 3650 20010101 seq default \*0\*  
首次构建由于需要构建全部的索引，故会慢些，默认会采用按照每个月的上中下旬进行增量构建

## 启动表

1. ./bluewhale mdrill table {表1,表2,表3}

示例如下：启动7个表

./bluewhale mdrill table rpt\_hitfake\_auctionall\_d,rpt\_p4padhoc\_product,rpt\_p4padhoc\_cust,rpt\_b2bad\_hoc\_memb\_sum\_d,r\_rpt\_cps\_luna\_item,r\_rpt\_cps\_adhoc\_pid,r\_rpt\_cps\_adhoc\_seller

启动后大家可以在每台机器的logs目录去看系统日志，以及通过启动的监控ui查看表的一些监控信息

注意，启动表只能执行一次，如果想添加新的表请先停止表后在启动表。

## 停止表

./bluewhale mdrill drop {表1,表2,表3}

./bluewhale mdrill drop rpt\_hitfake\_auctionall\_d,rpt\_p4padhoc\_product,rpt\_p4padhoc\_cust adhoc

## 导入实时数据（如果是离线模式，请忽略）

./bluewhale jar ./mdrill.jar com.alimama.mdrillImport.Topology {自定义任务名称} {导入的表列表多个表之间用逗号分隔} {使用的进程总数} {每个进程使用多少mb的内存} {从什么时间开始导入}

说明：

{从什么时间开始导入}，这个参数为可选参数在默认的TT4Reader实现中有用，可以指定tt4只读取某一时间的数据，这个作用是将来某一天如果服务停了，在启动的时候可以指定某一时间点，从某一时间开始补数据

如果其他数据源想读取该方法，可以通过confPrefix + "-start-time"，这个配置项将传递的值读入即可。

示例如下

./bluewhale jar ./mdrill.jar com.alimama.quanjingmonitor.mdrillImport.Topology p4p\_pv2topology p4p\_pv2 66 512 20131231074710

如果想停止导入数据的程序

那么使用

./bluewhale kill {自定义的任务名称}

示例如下

./bluewhale kill p4p\_pv2topology

## 如何重启服务

* 简单的通过停止表->启动表 重启服务
* 如果是因为mdrill本身的bug，需要重启mdrill，则可以这样做

1. 先停止表
2. 在每台机器上杀死所有的mdrill任务

ps -x|grep bluewhale|grep server|awk '{printf("%s\n",$1)}'|xargs kill

1. 清理mdrill的临时目录

rm xxx/bluewhale/stormwork/\* -rf

1. 重新按照原先的步骤 启动mdrill
2. 启动表