

LeDrill 部署文档

Version 1.0

Lenovo Confidential

研究院移动互联网研究室移动互联网服务项目组

2014 年 06 月

修改记录

日期	修改章节	修改类型*	修改描述	修改人	版本
2014-06-01	全部	A	草稿	谷磊、刘宏凯、孙磊	0.1
2012-06-30	全部	M	初稿	谷磊、刘宏凯、孙磊	1.0

*修改类型分为 A - ADDED M - MODIFIED D – DELETED

文件评审记录

时间	评审员	主要评审意见

目录

1	引言	4
1.1	编写目的	4
1.2	背景	4
1.3	定义	4
1.4	参考资料	4
2	集群综述	5
2.1	硬件参数	5
2.2	集群架构	5
2.2.1	Sth	6
3	部署	6
3.1	操作系统的准备和设置	6
3.1.1	操作系统	6
3.1.2	设置	6
3.2	CDH	8
3.2.1	安装 Cloudera Manager	8
3.2.2	下载分发安装 CDH	9
3.2.3	部署服务	9
3.2.4	其他配置	15
3.3	mdrill	16
3.3.1	生成 mdrill 制品包	16
3.3.2	安装依赖库	18
3.3.3	修改 OS 参数	19
3.3.4	安装 mdrill	20
3.3.5	启动/停止 mdrill	20
3.3.6	Sql 查询	23
4	mdrill 相关	24
4.1	storm.yaml 参数解读	24
4.2	实时表的 reader 的 parser	27
4.3	冗余列和多列	28
4.4	添加表或字段	28
4.5	空值 null	28
4.6	数据的清理	28
4.7	failCnt	29
4.8	mdrill jdbc	30
4.8.1	Maven 依赖	30
4.8.2	Spring 注入	30
4.8.3	GetTotal	31
5	Stream	31
6	Kafka	34
6.1	Zookeeper 集群	34
6.2	Kafka 集群	36
6.3	Java API	38
6.4	Kafka 集群监控	38

7	协议说明	39
7.1	PS 与 mdrill	39
8	骚扰度	40
8.1	数据的读取	40
8.1.1	Logstash	40
8.1.2	Java	41
8.2	骚扰度的计算	41
8.2.1	源码位置	41
8.2.2	打包方法	42
8.2.3	部署位置	42
8.2.4	环境准备	43
8.2.5	运行方法	43
9	尚未解决问题	43
10	感谢	44

1 引言

1.1 编写目的

对 Kafka 集群，mdrill 集群的部署进行说明，并对开发过程中积累的经验进行总结。

1.2 背景

对 PushServer 产生的海量数据进行统计，实现系统监控，数据报表。

数据量：15 亿 PV/天；2000 万 UV/天；3000 万反馈/天

1.3 定义

英文缩写	英文全称	说明
PushServer	Push Server	消息推送服务器

1.4 参考资料

mdrill 主页：<https://github.com/alibaba/mdrill>

《INSTALL.docx》： mdrill 的安装文档。

前人的安装经验：<http://my.oschina.net/292672967/blog/209139>

Cloudera 文档：<http://www.cloudera.com/content/support/en/documentation.html>

Kafka 文档: <http://kafka.apache.org/documentation.html>

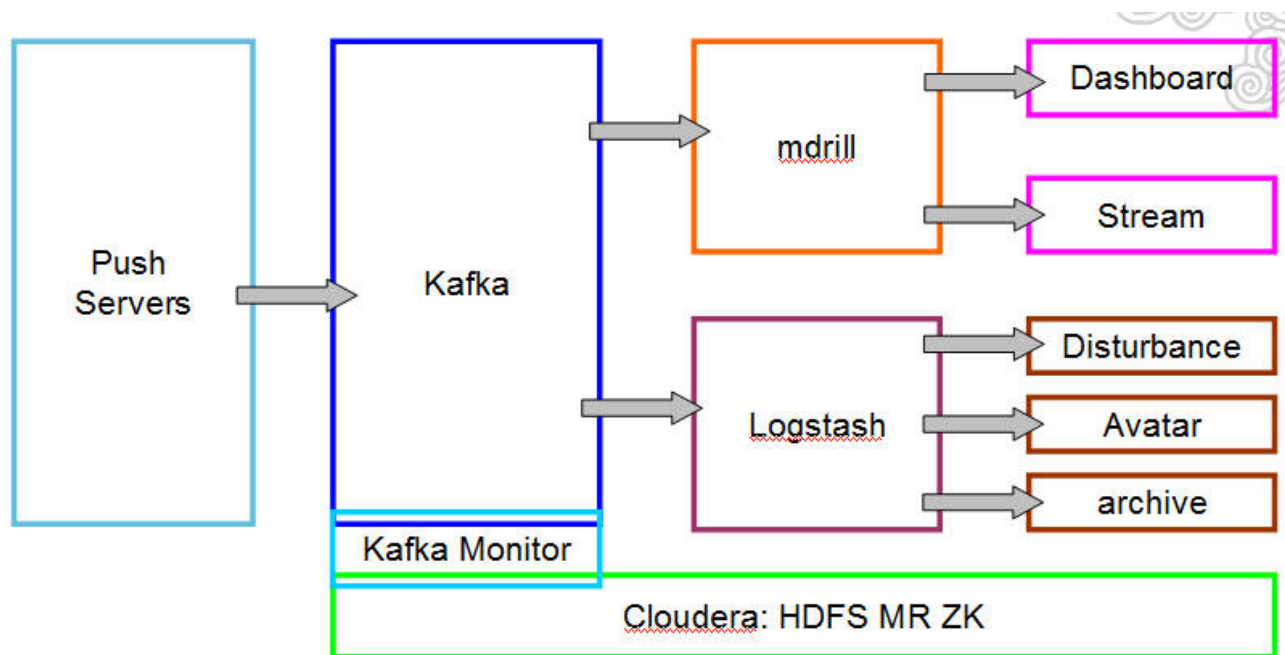
2 集群综述

2.1 硬件参数

Host Name	配置	CPU 核心数	CPU 型号
PUSH-001 ~ 08	40cpu; 256g 内存; 5t 硬盘;	40 = 2 X 10 X 2	Intel(R) Xeon(R) CPU E5-2660 v2 @ 2.20GHz stepping 04
PUSH-009 ~ 13; 18 ~ 20	24cpu; 128g 内存; 5t 硬盘;	24 = 2 X 6 X 2	Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz stepping 04
PUSH-014 ~ 17	32cpu; 256g 内存; 5t 硬盘;	32 = 2 X 8 X 2	Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz stepping 07

Host Name	配置	CPU 核心数	
psb-la-vm02 psb-la-vm04 psb-la-vm05	4cpu; 64g 内存; 2.5t 硬 盘	4 核	虚拟机

2.2 集群架构



2.2.1 Sth

Lenovo Confidential

3 部署

3.1 操作系统的准备和设置

3.1.1 操作系统

CentOS6.4: CDH 的建议版本

http://www.cloudera.com/content/cloudera-content/cloudera-docs/CM4Ent/latest/Cloudera-Manager-Installation-Guide/cmig_cm_requirements.html?scroll=cmig_topic_4_1_unique_1

3.1.2 设置

3.1.2.1 Hosts 文件和无密钥登录

没有 DNS 的情况下需要设置 Hosts 文件:

http://www.cloudera.com/content/cloudera-content/cloudera-docs/CM4Ent/latest/Cloudera-Manager-Installation-Guide/cmig_cm_requirements.html?scroll=cmig_topic_4_1_unique_1

[uide/cmig_cm_requirements.html?scroll=cmig_topic_4_3_3_unique_1](#)

将集群里 hostname 和 IP 追加到每台机器的/etc/hosts 文件中。最后 2 台机器是 Kafka 集群的 IP。

10.0.4.101 PUSH-001

10.0.4.102 PUSH-002

10.0.4.103 PUSH-003

10.0.4.104 PUSH-004

10.0.4.105 PUSH-005

10.0.4.106 PUSH-006

10.0.4.107 PUSH-007

10.0.4.108 PUSH-008

10.0.4.109 PUSH-009

10.0.4.110 PUSH-010

10.0.4.111 PUSH-011

10.0.4.112 PUSH-012

10.0.4.113 PUSH-013

10.0.4.114 PUSH-014

10.0.4.115 PUSH-015

10.0.4.116 PUSH-016

10.0.4.117 PUSH-017

10.0.4.118 PUSH-018

10.0.4.119 PUSH-019

10.0.4.120 PUSH-020

172.17.61.114 psb-la-vm04

172.17.61.115 psb-la-vm05

3.1.2.2 无密钥登录

使用 Cloudera Manager 安装集群，需要满足权限需求。

http://www.cloudera.com/content/cloudera-content/cloudera-docs/CM4Ent/latest/Cloudera-Manager-Installation-Guide/cmig_permissions.html

安装 Cloudera Manager 的服务器：PUSH—009。登录到该服务器，切换到 root 账号。

// 如果没有.ssh 目录的可以执行下 ssh 登录命令生成，例如，ssh -p 22222 root@10.0.4.109

```
# cd /root/.ssh
```

```
# ssh-keygen -q -t rsa -N "" -f /home/hadoop/.ssh/id_rsa
```

```
# cd .ssh/
```

```
# cat id_rsa.pub > authorized_keys
```

```
# chmod go-wx authorized_keys
```

```
[root@PUSH-009 .ssh]# pwd
```

```
/root/.ssh
```

```
[root@PUSH-009 .ssh]# ll
```

```
total 24
```

```
-rw-r--r-- 1 root root 395 May 26 16:11 authorized_keys
```

```
-rw----- 1 root root 1675 May 26 16:10 id_rsa
```

```
-rw-r--r-- 1 root root 395 May 26 16:10 id_rsa.pub
```

```
-rw-r--r-- 1 root root 8340 May 26 17:52 known_hosts
```

将上面的 `authorized_keys` 文件分发到集群所有主机的 `/root/.ssh` 目录下面。如果没有该目录，也可以用上面的方法自动生成。文件 `id_rsa` 为 Cloudera Manager 页面上使用的私钥。

3.1.2.3 yum 库更改

系统默认安装的 yum 库中 `cobbler-config.repo` 提供的 url 无法访问，引起库更新无法完成的错误。为了能够正常使用 yum，使用下面的命令将上述库移除。

```
mv /etc/yum.repos.d/cobbler-config.repo /etc/yum.repos.d/ cobbler-config.repo.bk
```

3.2 CDH

3.2.1 安装 Cloudera Manager

1. 卸载系统中已经安装的 JDK，安装 CDH 推荐的 JDK 版本（注：如果不自己装，Cloudera Manager 安装包会自动为每台机器安装 Oracle JDK 1.6 (1.6.0_31)）。

查看当前系统中安装的 JDK

```
rpm -qa | grep java
```

卸载当前系统中安装的 JDK

```
rpm -e jdk-package-name
```

下载 CDH 推荐的 JDK 版本 Java SE Development Kit 6u31，下载地址为 <http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-javase6-419409.html>

安装 JDK

```
rpm -ivh jdk-package-name
```

系统中 `/etc/profile` 中配置的 `JAVA_HOME` 为 `/usr/java/1.6.0_24`，与安装的 `/usr/java/1.6.0_31` 不一致，使用如下命令创建软链接。

```
ln -s /usr/java/jdk1.6.0_31 /usr/java/jdk1.6.0_24
```

2. 关闭 SELinux

修改文件 `/etc/selinux/config`，将其中的 `SELINUX` 选项设置为 `disabled`

```
SELINUX=disabled
```

检查 SELinux 是否已经关闭

```
selinuxenabled && echo enabled || echo disabled
```

3. 关闭 IPv6

向文件 `/etc/sysctl.conf` 末尾追加如下两行

```
net.ipv6.conf.all.disable_ipv6 = 1
```

```
net.ipv6.conf.default.disable_ipv6 = 1
```

检查 IPv6 是否已经关闭

```
cat /proc/sys/net/ipv6/conf/all/disable_ipv6
```

4. 关闭 iptables

```
service iptables stop
```

```
chkconfig iptables off
```

检查 iptables 是否已经关闭


```
service iptables status
```

5. 下载 Cloudera Manager Installer，下载地址为 <http://archive.cloudera.com/cm4/installer/latest/cloudera-manager-installer.bin>，版本为 4.8.3。
6. 在 PUSH-009 上安装 Cloudera Manager Server。
将下载自动安装文件改为可执行并运行自动安装程序安装 Cloudera Manager Server。

```
chmod +x cloudera-manager-installer.bin
```

```
sudo ./cloudera-manager-installer.bin
```
7. 安装好后，Cloudera Manager Server 会在 PUSH-009 上提供 Web 页面的 Admin Console 服务，监听端口 7180。通过浏览器访问 Admin Console: <http://ip-PUSH-009:7180>。首次登录的默认用户名和密码均为 admin。
8. 如果需要打洞，参考: rinetd
<http://www.linuxidc.com/Linux/2011-01/31287.htm>

3.2.2 下载分发安装 CDH

9. 登录 Admin Console 后即可安装 CDH。选择安装 Cloudera Express Standard 免费版本。
10. 输入需要 Cloudera Manager Server 管理的主机列表及 SSH 端口号: PUSH-0[01-20], 22222。Cloudera Manager Server 会自动扫描提供的主机列表并显示各主机的连通性。
11. 向 Cloudera Manager Server 提供 PUSH-009 root 用户 ssh 私钥文件，私钥文件默认位于 /root/.ssh/id_rsa 中，如果在生成私钥时设置了 passphrase，还需要将 passphrase 提供给 Cloudera Manager Server。
12. 使用 Cloudera 推荐的 parcels 包方式安装 CDH。在 Cloudera Manager Server 上下载 CDH 的 parcels 包，并分发到管理的各主机。当前集群中安装的 CDH，IMPALA（可选）和 SOLR（可选）的版本如下：
CDH-4.6.0-1.cdh4.6.0.p0.26-el6.parcel
IMPALA-1.3.1-1.impala1.3.1.p0.1172-el6.parcel
SOLR-1.2.0-1.cdh4.5.0.p0.4-el6.parcel

注意：分发时并发数不宜过大（测试设置为 5 较合适），否则分发过程中会出现进度交替前进回退的问题。为避免该问题，在分发前可以做如下配置。

登入 Cloudera Manager Admin Console 主页面，点击下拉菜单 Administration -> Settings -> Parcels -> 找到配置项 Maximum Parcel Uploads，设置为 5 -> Save Changes，即可生效。

Maximum Parcel Uploads	<input type="text" value="5"/>	Maximum number of concurrent uploads allowed to distribute parcels to individual hosts. The maximum allowed number of concurrent uploads is 50.
	Reset to the default value: 25 ↗	

3.2.3 部署服务

13. 向 Cloudera Manager 管理的主机列表中添加集群，配置服务，分配角色。目前，集群中只安装了 HDFS，MapReduce，ZooKeeper 和管理服务四种服务。
当下载分发并安装 CDH 后，会出现 Version Summary 页面，显示已经安装各个组件的版本汇总信息。确认各组件的版本信息后，点击右下方的 Continue 按钮继续。接下来出现页面提示选择在集群中安装 CDH 服务，如下图所示，其中图下方提示在安装 CDH4 服务的同时也会为集群安装 Cloudera Management Services。

Choose the CDH4 services that you want to install on your cluster.

Choose a combination of services to install.











- ☐ **Core Hadoop**
HDFS, MapReduce, ZooKeeper, Oozie, Hive, and Hue
- ☐ **Core with Real-Time Delivery**
HDFS, MapReduce, ZooKeeper, HBase, Oozie, Hive, and Hue
- ☐ **Core with Real-Time Query**
HDFS, MapReduce, ZooKeeper, Impala, Oozie, Hive, and Hue
- ☐ **All Services**
HDFS, MapReduce, ZooKeeper, HBase, Impala, Oozie, Hive, Hue and Sqoop.
- ☐ **Custom Services**
Choose your own services. Services required by chosen services must also be selected. Note that Flume, Solr and Key-Value Store Indexer services can be added after your initial cluster has been set up.

This wizard will also install the **Cloudera Management Services**. These are a set of components that enable monitoring, reporting, events, and alerts; these components require databases to store information, which will be configured on the next page.

选择 Custom Services 选项后，在下方出现可以选择安装的服务，如下图所示。

☒ Custom Services

Choose your own services. Services required by chosen services must also be selected. Note that Flume, Solr and Key-Value Store Indexer services can be added after your initial cluster has been set up.

Service Type	Description
<input checked="" type="checkbox"/>  HDFS	Apache Hadoop Distributed File System (HDFS) is the primary storage system used by Hadoop applications. HDFS creates multiple replicas of data blocks and distributes them on compute hosts throughout a cluster to enable reliable, extremely rapid computations.
<input checked="" type="checkbox"/>  MapReduce	Apache Hadoop MapReduce supports distributed computing on large data sets across your cluster (requires HDFS).
<input type="checkbox"/>  YARN	Apache Hadoop MapReduce 2.0 (MRv2), or YARN, is a data computation framework that supports MapReduce applications (requires HDFS). The current upstream MRv2 release is not yet considered stable and should not be considered production-ready at this time.
<input checked="" type="checkbox"/>  ZooKeeper	Apache ZooKeeper is a centralized service for maintaining and synchronizing configuration data.
<input type="checkbox"/>  HBase	Apache HBase provides random, real-time, read/write access to large data sets (requires HDFS and ZooKeeper).
<input type="checkbox"/>  Hive	Hive is a data warehouse system that offers a SQL-like language called HiveQL.
<input type="checkbox"/>  Oozie	Oozie is a workflow coordination service to manage data processing jobs on your cluster.
<input type="checkbox"/>  Hue	Hue is a graphical user interface to work with Cloudera's Distribution Including Apache Hadoop (requires HDFS, MapReduce, and Hive).
<input type="checkbox"/>  Impala	Impala provides a real-time SQL query interface for data stored in HDFS and HBase. Impala requires Hive service and shares Hive Metastore with Hue.
<input type="checkbox"/>  Sqoop	Sqoop is a tool designed for efficiently transferring bulk data between Apache Hadoop and structured datastores such as relational databases. The version supported by Cloudera Manager is Sqoop 2 .

This wizard will also install the **Cloudera Management Services**. These are a set of components that enable monitoring, reporting, events, and alerts; these components require databases to store information, which will be configured on the next page.

[Inspect Role Assignments](#)
[Continue](#)

勾选需要安装的服务 HDFS，MapReduce 和 ZooKeeper，并点击页面右下角的 Inspect Role Assignments 按钮，出现 Inspect role assignments 页面为集群中各个主机分配服务角色。我们将 ZooKeeper

Server 分配在 PUSH-010~PUSH-013, PUSH-018 五台主机上; HDFS NameNode 分配在 PUSH-008 上, SecondaryNameNode 分配在 PUSH-007 上, DataNode 分配在 PUSH-001~PUSH-008, PUSH-014~PUSH-017; MapReduce JobTracker 分配在 PUSH-001, TaskTracker 分配在 PUSH-001~PUSH-008, PUSH-014~PUSH-017; 管理服务的 Service Monitor, Activity Monitor, Host Monitor, Event Server 和 Alert Publisher 分配在 PUSH-009 上。

各项服务在每台主机上的角色分配情况如下表所示。

Host Name	Cloudera Manager	HDFS Roles	MRv1 Roles	ZooKeeper Roles	Mdrill Roles
PUSH-001	Agent	DataNode	JobTracker, TaskTracker		Supervisor Map Reduce
PUSH-002	Agent	DataNode	TaskTracker		Supervisor Mdrillui Map Reduce
PUSH-003	Agent	DataNode	TaskTracker		Nimbusserver Supervisor Map Reduce
PUSH-004	Agent	DataNode	TaskTracker		Supervisor Map Reduce
PUSH-005	Agent	DataNode	TaskTracker		Supervisor Map Reduce
PUSH-006	Agent	DataNode	TaskTracker		Supervisor Map Reduce
PUSH-007	Agent	SecondaryNameNode DataNode	TaskTracker		Supervisor Map Reduce
PUSH-008	Agent	NameNode DataNode	TaskTracker		Supervisor Map Reduce
PUSH-009	Server Web UI	Database, Activity Monitor, Host Monitor, Service Monitor, Event Server, Alert Publisher, HDFS Gateway, MapReduce Gateway			
PUSH-010	Agent			ZooKeeper Server	
PUSH-011	Agent			ZooKeeper Server	
PUSH-012	Agent			ZooKeeper Server	
PUSH-013	Agent			ZooKeeper Server	
PUSH-014	Agent	DataNode	TaskTracker		Supervisor Map Reduce
PUSH-015	Agent	DataNode	TaskTracker		Supervisor Map Reduce
PUSH-016	Agent	DataNode	TaskTracker		Supervisor Map Reduce
PUSH-017	Agent	DataNode	TaskTracker		Supervisor Map Reduce

PUSH-018	Agent			ZooKeeper Server	
PUSH-019	Agent				
PUSH-020	Agent				

14. 为管理服务 Service Monitor, Activity Monitor 和 Host Monitor 设置数据库。

Cloudera Manager 支持三种类型的数据库 PostgreSQL, MySQL 和 Oracle。其中 PostgreSQL 提供了嵌入和外置两种安装方式。安装中使用嵌入式 PostgreSQL 数据库, Cloudera Manager 安装向导程序会根据上一步分配管理服务所在主机的配置, 自动为管理服务配置数据库主机名称, 数据库类型, 数据库名称, 用户名和密码, 且配置无法修改。点击页面右下角的 Test Connection 按钮, 可以测试管理服务的数据库连接, 若测试均成功, 点击页面右下角的 Continue 按钮, 继续下一步。安装向导程序为嵌入式 PostgreSQL 数据库配置的默认参数如下图所示。

The screenshot displays three configuration panels for Service Monitor, Activity Monitor, and Host Monitor. Each panel shows the following settings:

- Service Monitor:** Database Host Name: PUSH-009.7432, Database Type: PostgreSQL, Database Name: smon, Username: smon, Password: uUWievclJ0t.
- Activity Monitor:** Database Host Name: PUSH-009.7432, Database Type: PostgreSQL, Database Name: amon, Username: amon, Password: gXMtUgt9y.
- Host Monitor:** Database Host Name: PUSH-009.7432, Database Type: PostgreSQL, Database Name: hmon, Username: hmon, Password: GGV5dQh4ZS.

Each panel also indicates it is 'Currently assigned to run on PUSH-009' and shows a 'Successful' status with a green checkmark.

15. 配置各项服务数据存放路径

在页面 Review configuration changes 中配置 HDFS, MapReduce 和 ZooKeeper 服务各角色数据存放路径配置。在页面中在不同服务标签下找到某一角色需要配置的 Parameter 列, 修改右侧对应的 Recommended Value 列中对应的值, 所有配置都修改后, 点击页面右下角的 Continue 按钮继续。下图展示了 HDFS 服务中 DataNode 角色数据存放路径。

Review configuration changes

The screenshot shows the 'Review configuration changes' page for the 'Service hdfs1' group. The table lists the following configuration:

Group	Parameter	Recommended Value	Description
Service hdfs1	DataNode (Default)	/data/hadoop/hdfs/dn	Comma-delimited list of directories on the local file system where the DataNode stores HDFS block data. Typical values are /data/N/dfs/dn for N = 1, 2, 3... These directories should be mounted using the noatime option and the disks should be configured using JBOD. RAID is not recommended.
	dfs.datanode.data.dir		

The 'Recommended Value' for the DataNode (Default) is shown as a text input field containing '/data/hadoop/hdfs/dn'. A '+' icon is next to the field, and a link 'Reset to empty default value' is below it.

需要修改的具体配置如下:

HDFS 各角色数据存放路径:

NameNode Data Directories: /data/hadoop/dfs/nn
HDFS Checkpoint Directory : /data/hadoop/dfs/snn
DataNode Data Directory: /data/hadoop/dfs/dn
MapReduce 各角色数据存放路径:
JobTracker Local Data Directory: /data/hadoop/mapred/jt
TaskTracker Local Data Directory: /data/hadoop/mapred/local
ZooKeeper 数据存放路径:
Data Directory: /data/zookeeper/data
Transaction Log Directory: /data/zookeeper/dataLog

如果安装后需要修改, 可以按如下流程进行操作, 配置完成需要重启服务。

Admin Console 主页面 -> Services -> *Service-Name* -> Configuration -> View and Edit -> 搜索框中搜索属性名称 -> 设置值 -> Save Changes -> Actions -> Restart。

其中 *Service-Name* 表示 hdfs1, mapreduce1 或者 zookeeper1。

16. 启动各项服务, 安装完毕。

最后安装向导程序会分 8 个步骤按顺序启动集群中的各项服务, 如下图所示。

Starting your cluster services.

Completed 8 of 8 steps.	
✓	Waiting for ZooKeeper Service to initialize Finished waiting
✓	Starting ZooKeeper Service Service started successfully.
✓	Checking if the name directories of the NameNode are empty. Formatting HDFS only if empty. Failed to format NameNode.
✓	Starting HDFS Service Service started successfully.
✓	Creating HDFS /tmp directory HDFS directory /tmp already exists.
✓	Starting MapReduce Service Service started successfully.
✓	Starting Cloudera Management Services Service started successfully.
✓	Deploying Client Configuration Successfully deployed all client configurations

如果每个步骤都成功执行, 可以点击页面右下角的 Continue 按钮继续下一步。

当页面出现下图所示信息, 说明 HDFS, MapReduce 和 ZooKeeper 各项服务部署成功。

Congratulations!

The Hadoop services are installed, configured, and running on your cluster.

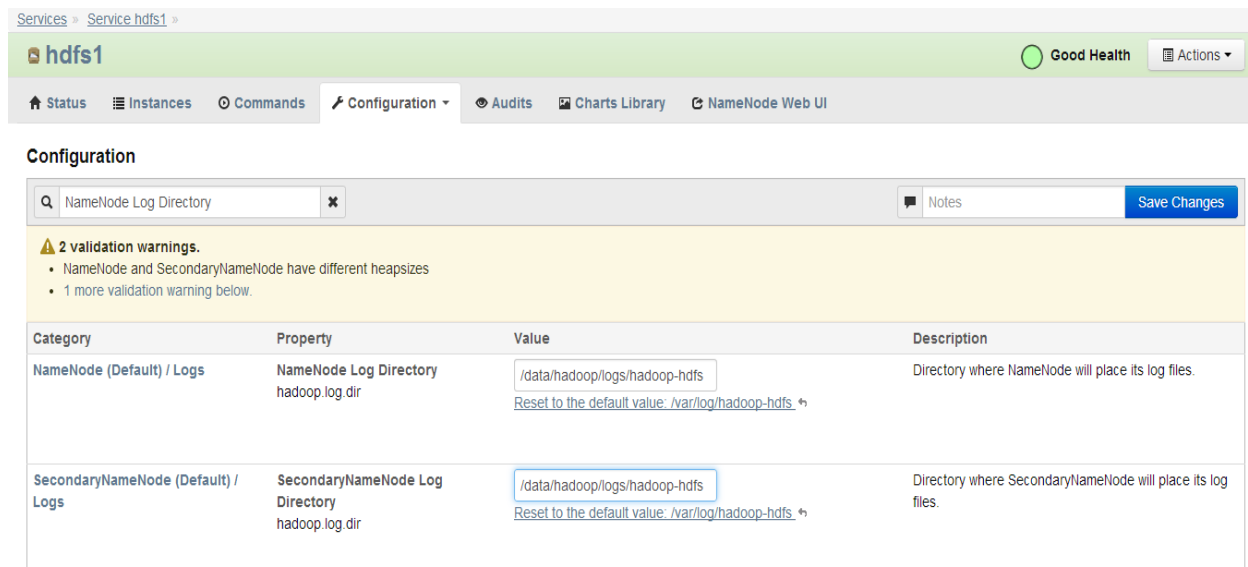
17. 修改各项服务角色日志输出路径。

可以按如下流程进行操作, 配置完成需要重启服务。

Admin Console 主页面 -> Services -> Service-Name -> Configuration -> View and Edit -> 搜索框中搜索属性名称 -> 设置值 -> Save Changes -> Actions -> Restart。

其中 Service-Name 表示 hdfs1, mapreduce1 或者 zookeeper1。

下图是修改 NameNode Log Directory 和 SecondaryNameNode Directory 的日志输出路径的示例。



需要修改的所有配置如下：

HDFS 各角色日志输出路径：

NameNode Log Directory: /data/hadoop/logs/hadoop-hdfs

SecondaryNameNode Log Directory: /data/hadoop/logs/hadoop-hdfs

DataNode Log Directory: /data/hadoop/logs/hadoop-hdfs

HttpFS Log Directory: /data/hadoop/logs/hadoop-httpfs

MapReduce 各角色日志输出路径：

JobTracker Log Directory: /data/hadoop/logs/hadoop-0.20-mapreduce

TaskTracker Log Directory: /data/hadoop/logs/hadoop-0.20-mapreduce

ZooKeeper 各角色日志输出路径：

ZooKeeper Log Directory: /data/zookeeper/logs

18. 配置 mdrrill 所需的 ZooKeeper 各项参数。

可以按如下流程进行操作，配置完成需要重启 ZooKeeper 服务。

Admin Console 主页面 -> Services -> zookeeper1 -> Configuration -> View and Edit -> 搜索框中搜索属性名称 -> 设置值 -> Save Changes -> Actions -> Restart。

需要修改的所有配置如下：

ZooKeeper 配置项：黄色的是添加的配置；没有黄色的是默认值，检查是否与 cdh 中的配置相符。

The number of milliseconds of each tick

tickTime=2000

The number of ticks that the initial synchronization phase can take

initLimit=10

The number of ticks that can pass between sending a request and getting an acknowledgement

syncLimit=5

The directory where the snapshot is stored.

dataDir=/data/tmp/zookeeper/

The port at which the clients will connect


```
clientPort=2181
# The number of snapshots to retain in dataDir
autopurge.snapRetainCount=100
# Purge task interval in hours. Set to "0" to disable auto purge feature
autopurge.purgeInterval=12
#for mdri11
maxClientCnxns=300
maxSessionTimeout=20000
```

3.2.4 其他配置

1. 网卡监控

安装部署 CDH 后，由 Cloudera Manager 监控的集群的每台主机都有一个相同的警告：The host has 1 network interface(s) that appear to be operating at less than full speed. Warning threshold: any.

Cloudera Manager 对集群中主机网卡速度监控的属性主要有三个：Network Interface Expected Link Speed，Host's Network Interfaces Slow Link Modes Thresholds 和 Network Interface Collection Exclusion Regex。

Network Interface Expected Link Speed 指的是每台主机期望的链接速度，默认配置为 1000Mbps

Host's Network Interfaces Slow Link Modes Thresholds 指的是当集群中的每台主机上有几个网卡处于低速状态运行时报警，默认配置 any，含义是只要有一个网卡处于低速状态，就报警。

Network Interface Collection Exclusion Regex 指的是不需要监控的网卡集合，默认配置是 ^lo\$，含义是不监控 loop 端口的网速状态。

通过在每台主机上运行命令 ifconfig 可以看到每台主机上共有六个网卡：eth0, eth1, eth2, lo, virbr0, virbr0-nic，其中处于启动状态的网卡有三个：eth2, lo 和 virbr0-nic。其中 lo 网卡已经排除没有监控，virbr0-nic 处于监控状态，但是其速度只有 10Mbps，是引起警告的原因。由于在实际生产环境下并不需要使用网卡 virbr0-nic，所以也将该网卡排除在监控范围之外。具体配置方法如下：

在 Cloudera Manager 主页面 Hosts -> Configuration -> View and Edit -> Monitoring -> 搜索框中搜索属性 Network Interface Collection Exclusion Regex -> 修改值为 lo|virbr0-nic -> 右上角点击 Save Changes。

2. MapReduce 的资源

限制每个结点上 Map 任务最大并发量

集群部署了 TaskTracker 角色的结点上都配置了一个参数用于限定每个结点同时执行 Map 任务的最大个数 (Maximum Number of Simultaneous Map Tasks)，具体配置项为 mapred.tasktracker.map.tasks.maximum，该配置项默认值设置为机器中 CPU 的核数，即 PUSH-001~008 设置为 40，PUSH-014~017 设置为 32，按当前的默认设置，整个 Hadoop 集群（12 个结点）同时执行的最大 Map 任务数量是 448 (40 x 8 + 32 x 4) 个。Hadoop 官方 [Wiki](#) 推荐的同时运行的 Map 任务级别为每个结点 10~100 个 Map 任务，虽然当前配置符合推荐，但为了缓解每个结点的压力，将该配置项设置为 3，以限制每个结点同一时间 Map 任务个数，从而降低对系统资源的占用。具体配置方法如下：

Cloudera Manager 主页面 -> Services -> mapreduce1 -> Configuration -> View and Edit -> TaskTracker/TaskTracker(1) -> Performance 中设置 Maximum Number of Simultaneous Map Tasks 值为 20/16 -> Save Changes -> Actions -> Restart 重启 MapReduce 服务。

调低该配置项后，并发性降低，整个作业执行时间会增加。

控制每个 Map 任务和 Reduce 任务最大内存占用量

默认情况下，每个 Map 任务占用内存并无限制，这样在运行 MapReduce 作业时，每个机器上并发运行 20(或者 12)个 Map 任务，会消耗掉主机上的几乎所有内存。为了不对其他进程造成影响，限制了每个 Map 任务的内存占用最大为 512MB，每个 Reduce 任务的内存占用最大为 512MB。

具体设置方法如下。

Cloudera Manager 主页面 -> Services -> mapreduce1 -> Configuration -> View and Edit -> TaskTracker/TaskTracker(1) -> Resource -> Resource Management 中设置如下两项

Map Task Maximum Heap Size (Client Override) 512MB

Reduce Task Maximum Heap Size (Client Override) 512MB

-> Save Changes -> Actions -> Restart 重启 MapReduce 服务。

3. 清空 Cache

Cloudera 监控的天津机房主机 PUSH-016 报警，警告有大量页面被交换到磁盘上。通过监控平台观察 PUSH-016 上真实物理内存占用并不多，只用了 18G，大量内存（大约 230G）被用于 Cache。Cache 是 Linux 系统特性，当程序读写文件的时候，Linux 内核为了提高读写效率与速度，会将文件在内存中进行缓存，这部分内存就是 Cache Memory（缓存内存）。即使程序运行结束后，Cache Memory 也不会自动释放。这就会导致在 Linux 系统中程序频繁读写文件后，发现可用物理内存会很少。若内存被 Cache 占满，系统会启动置换策略，将内存中不常使用的数据置换到 Swap 分区，导致警告出现。

在 PUSH-016 上运行如下命令，将当前的 Cache 清空后，警告消失。

```
sudo sync && sudo echo 3 | sudo tee /proc/sys/vm/drop_caches
```

3.3 mdrill

部署的机器和角色分布见前面“部署服务”小结中的表格。

3.3.1 生成 mdrill 制品包

3.3.1.1 源码和依赖库下载

源代码下载地址

<https://github.com/alibaba/mdrill>

2014.03.05 version 0.20.9-beta bugfix

注意，依赖的 zeromq 和 jzmq 也都在页面上的连接下载。

3.3.1.2 实时数据的读取

见 svn 地址

<http://rta.lenovo.com/svn/beacon09/06 Service/Project/06 Code/LpsPushService2.0/LeDrill>

在 mdrill 源代码的基础上添加了 ledrill-common 和 ledrill-internal 工程

3.3.1.3 修改 storm.yaml



storm.yaml

3.3.1.4 修改 pom.xml

修改父工程的 pom.xml 文件

1. 使用 CDH 的 hadoop 库

http://www.cloudera.com/content/cloudera-content/cloudera-docs/CDH4/latest/CDH-Version-and-Packaging-Information/cdhvd_topic_8.html

加入 CDH repo:

```
<repository>
  <id>cloudera</id>
  <url>https://repository.cloudera.com/artifactory/cloudera-repos</url>
</repository>
```

原来的 hadoop 依赖

```
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-core</artifactId>
<version>0.20.2</version>
<exclusions>
```

...

```
</exclusions>
```

```
</dependency>
```

修改为(其中, exclusions 部分不变)

```
<dependency>
  <groupId>org.apache.hadoop</groupId>
  <artifactId>hadoop-common</artifactId>
  <version>2.0.0-cdh4.6.0</version>
```

```
<exclusions>
```

...

```
</exclusions>
```

```
</dependency>
```

```
<dependency>
```

```
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-hdfs</artifactId>
<version>2.0.0-cdh4.6.0</version>
<exclusions>
```

...

```
</exclusions>
```

```
</dependency>
```

```
<dependency>
```

```
  <groupId>org.apache.hadoop</groupId>
  <artifactId>hadoop-core</artifactId>
  <version>2.0.0-mr1-cdh4.6.0</version>
```

```
<exclusions>
```

...

```
</exclusions>
```

```
</dependency>
```

并且修改 commons-io 的版本为 1.4:

```
<dependency>
  <groupId>commons-io</groupId>
  <artifactId>commons-io</artifactId>
  <!-- <version>1.4</version> -->
  <version>2.1</version>
</dependency>
```

3.3.1.5 打包

Linux 下使用 mvn 打包

```
$ mvn clean && mvn package assembly:assembly
```

会生成 alimama-adhoc.tar.gz

```
[root@htest01 target]# pwd
```

```
/root/mysvn/LeDrill/mdrill/0.20.9/target
```

```
[root@htest01 target]# ll
```

```
total 65808
```

```
-rw-r--r-- 1 root root 67308075 May 26 18:22 alimama-adhoc.tar.gz
```

```
drwxr-xr-x 2 root root      4096 May 26 18:19 archive-tmp
```

3.3.2 安装依赖库

参考

<http://my.oschina.net/292672967/blog?catalog=450148>

将下面两个文件上传到 mdrill 机器的所有服务器（zeromq-tested.zip 中包含 zeromq-2.1.7.tar.gz 和 jzmq-master.zip）

```
[root@PUSH-009 .script]# ll ../mydownload/ledrill/
```

```
total 67632
```

```
-rw-r--r-- 1 root root 67308075 May 27 13:59 alimama-adhoc.tar.gz
```

```
-rw-r--r-- 1 root root  1944192 May 27 13:59 zeromq-tested.zip
```

制作安装脚本

```
# vi install-libs.sh
```

添加如下内容

```
#!/bin/bash
```

```
# yum required libs
```

```
yum -y install libtool
```

```
yum -y install gcc-c++
```

```
yum -y install uuid-devel
```

```
yum -y install libuuid-devel
```

```
cd /root/mydownload/ledrill
```

```
unzip zeromq-tested.zip
```

```
# install zeromq
cd /root/mydownload/ledrill
tar -xvf zeromq-2.1.7.tar.gz
cd zeromq-2.1.7
./autogen.sh
./configure
make
make install
```

```
# install jzmq
cd /root/mydownload/ledrill
unzip jzmq-master.zip
cd jzmq-master
./autogen.sh
./configure
make
make install
```

```
echo "export LD_LIBRARY_PATH=/usr/local/lib" > /etc/profile.d/ldrill.sh
source /etc/profile.d/ldrill.sh
```

```
# chmod +x install-libs.sh
在 mdrill 的所有机器上执行该脚本
```

```
测试 jzmq 时候装好
cd /root/mydownload/ledrill/jzmq-master/perf
sh local_lat.sh    tcp://127.0.0.1:5000 1 100
```

```
另外启动控制台：
cd /root/mydownload/ledrill/jzmq-master/perf
sh remote_lat.sh    tcp://127.0.0.1:5000 1 100
message size: 1 [B]
roundtrip count: 100
mean latency: 275.0 [us]
看到 message size: 1 [B]...则说明配置成功。
```

3.3.3 修改 OS 参数

```
所有 mdrill 集群中的服务器
// 在下面的文件里追加
# vi /etc/sysctl.conf
# mdrill
net.ipv4.ip_local_port_range = 10000 65535
```

```
net.core.somaxconn = 4096
net.core.netdev_max_backlog=16384
net.ipv4.tcp_max_syn_backlog=8192
net.ipv4.tcp_syncookies=1
fs.file-max = 100000
```

```
// 使配置生效
# sysctl -p /etc/sysctl.conf
```

如果 sysctl.conf 中已经有了其中的某些项配置，追加也没有关系，后面的配置起作用。

3.3.4 安装 mdriII

所有 mdriII 集群中的服务器
编辑 prepare-alimama.sh 脚本
vi prepare-alimama.sh

```
#!/bin/bash
cd /root/mydownload/ledriII/
mv alimama-adhoc.tar.gz.0526.cdh alimama-adhoc.tar.gz
tar -xzf alimama-adhoc.tar.gz
mv alimama /data/
cd /data/alimama/adhoc-core/bin
chmod 777 ./bluewhale
dos2unix ./bluewhale
# 创建 mdriII 使用的本地目录（与 storm.yamlI 中的配置相符）
mkdir /data/storm
mkdir /data/mdriII
```

在 mdriII 集群其中一台机器上执行

```
# HDFS 中创建 mdriII 所需的目录结构，并修改所有者权限（与 storm.yamlI 中的配置相符）
su hdfs
hadoop fs -mkdir -p /mdriII/tablelist
hadoop fs -chown -R root:root /mdriII
```

3.3.5 启动/停止 mdriII

检查 python 版本：centos6.4 默认的版本是 2.6.6，满足 mdriII 要求。

```
[root@PUSH-001 ~]# python -V
Python 2.6.6
```

进入

```
[root@PUSH-002 bin]# pwd
/data/alimama/adhoc-core/bin
[root@PUSH-002 bin]# ls
bluewhale  create_feedbackred.sql  create_feedback.sql  create_hitred.sql  create_hit.sql  init-bluewhale.sh
mdrill.jar  start-bluewhale.sh  stop-bluewhale.sh  supervisor.log  ui  ui.log
[root@PUSH-002 bin]#
```

创建表 (*.sql 的文件根据表的需求编写，详见《INSTALL.docx》)

```
./bluewhale mdrill create ./create_feedback.sql
./bluewhale mdrill create ./create_feedbackred.sql
./bluewhale mdrill create ./create_hit.sql
./bluewhale mdrill create ./create_hitred.sql
例子
```



create_hit.sql

建好了表，在 hdfs 上回看到相应的文件夹。

```
[root@PUSH-016 ~]# hadoop fs -ls /mdrill/tablelist
```

Found 5 items

```
drwxr-xr-x  - root root          0 2014-06-06 14:53 /mdrill/tablelist/error
drwxr-xr-x  - root root          0 2014-05-27 18:11 /mdrill/tablelist/feedback
drwxr-xr-x  - root root          0 2014-06-05 17:46 /mdrill/tablelist/feedbackred
drwxr-xr-x  - root root          0 2014-05-27 18:10 /mdrill/tablelist/hit
drwxr-xr-x  - root root          0 2014-06-05 17:47 /mdrill/tablelist/hitred
```

在启动 UI 后也可以看到

海狗数据表列表

数据表名	监控	表schema
hitred	查看	查看
feedback	查看	查看
error	查看	查看
hit	查看	查看
feedbackred	查看	查看
adhoc	查看	查看

PUSH-003 上执行

```
nohup ./bluewhale nimbus >nimbus.log &
```

mdrill 集群中每台机器执行

```
nohup ./bluewhale supervisor >supervisor.log &
```

PUSH-002 上执行

```
mkdir ./ui
```

```
nohup ./bluewhale mdrillui 1107 ./lib/adhoc-web-0.18-beta.jar ./ui >ui.log &
```

登录 PUSH-002 的 ip 是 10.0.4.102

<http://10.0.4.102:1107/mdrill.jsp>

adhoc-mdrill后台监控

[机器列表](#)
[海狗数据表](#)
[蓝鲸任务](#)

可以检查机器列表。

启动表

mdrill 集群中任意一台机器上执行

```
./bluewhale mdrill table feedback,feedbackred,hit,hitred
```

查看 logs 文件夹：

“如果有*6901.log，一定是任务调度配置错了。”

--- 母延年（mdrill 作者）

```
[root@PUSH-001 ~]# ll /data/alimama/adhoc-core/logs/
```

```
total 224652
```

```
-rw-r--r-- 1 root root      663 May 28 16:29 gc-6601.log
-rw-r--r-- 1 root root      377 May 28 15:38 gc-6701.log
-rw-r--r-- 1 root root      377 May 28 15:47 gc-6702.log
-rw-r--r-- 1 root root      377 May 28 15:28 gc-6703.log
-rw-r--r-- 1 root root      377 May 28 15:16 gc-6704.log
-rw-r--r-- 1 root root      377 May 28 15:29 gc-6705.log
-rw-r--r-- 1 root root    90612 May 28 13:58 supervisor.log
-rw-r--r-- 1 root root 21050272 May 28 17:22 worker-6601.log
-rw-r--r-- 1 root root 15480327 May 28 17:22 worker-6701.log
-rw-r--r-- 1 root root 15496944 May 28 17:22 worker-6702.log
-rw-r--r-- 1 root root 15539973 May 28 17:22 worker-6703.log
-rw-r--r-- 1 root root 15509500 May 28 17:22 worker-6704.log
-rw-r--r-- 1 root root 15568886 May 28 17:22 worker-6705.log
```

启动读任务

mdrill 集群中任意一台机器上执行

```
./bluewhale jar ./mdrill.jar com.alimama.mdrillImport.Topology feedbacktopology  
feedback,feedbackred,hit,hitred 2 512 1000 2014030520050
```

在 UI 中可以查看状态，点击“蓝鲸任务”。

```
active_storms  
feedbacktopology-6-1401256695  
adhoc-5-1401256687
```

adhoc-*: 对应的是启动表的任务

feedbacktopology: 对应的是读实时表的任务

如果需要:

停止表

mdrill 集群中任意一台机器上执行

```
./bluewhale mdrill drop feedback,feedbackred,hit,hitred
```

停止读任务

mdrill 集群中任意一台机器上执行

```
./bluewhale kill feedbacktopology
```

3.3.6 Sql 查询

浏览器访问: <http://10.0.4.102:1107/sql.jsp>

jdbc:mdrill://10.0.4.102:1107

select thedate,sum(records),count(records),dist(pid) from hit group by thedate

提交

totalRecords:2

thedata	sum(records)	count(records)	dist(pid)
20140528	1.15391944E8	7145059.0	5242752
20140527	5.070723E7	5630367.0	4652400

times taken 3.613 seconds

4 mdrill 相关

4.1 storm.yaml 参数解读

多么痛的领悟

有些参数仍然在领悟中，如果下面哪个参数的感觉错误，欢迎指正 ^_^

Yaml 格式的配置文件，检查的挺严格

冒号 “:” 后面要有空格，不然解析出错。

1. storm.zookeeper.root: "/mdrill"

mdrill 在 zk 中的目录名字

可以使用下面的方法查看

```
[root@PUSH-011 ~]# zookeeper-client
```

```
[zk: localhost:2181(CONNECTED) 0] ls /mdrill
```

```
[higo, supervisors, taskbeats, taskerrors, tasks, storms, assignments]
```

2. shard; parallel; merge;

#---创建索引生成的每个 shard 的并行----

higo.index.parallel: 5

#---启动的 shard 的数，每个 shard 为一个 solr 实例，结合 cpu 个数和内存进行配置，10 台 48G 内存配置 60----

higo.shards.count: 60

#---启动的 merger server 的 worker 数量，建议根据机器数量设定----

higo.mergeServer.count: 12

基本的算法是 $5 * 12 = 60$

如果是一台机器，而且配置并不高，这些参数（还有启动任务的参数等）都需要下降。可以参考这个文件（感谢延年的指导）



storm.yarm.baobei

这个 60 很关键，和下面的配置对应

#---任务分配，注意，原先分布在同一个位置的任务，迁移后还必须分布在一起，比如说 acker:0 与 merger:10 分布在一起了，那么迁移后他们还要在一起，为了保险起见，建议同一个端口跑多个任务的，可以像下面 acker, heartbeat 那样----

mdrill.task.ports.adhoc: "6901~6902"

mdrill.task.assignment.adhoc:

- "####看到这里估计大家都会晕，但是这个任务是任务调度很重要的地方，出去透透气，回来搞定这里吧####"

- "####注解: merge 表示是 merger server; shard@0 表示是 shard 的第 0 个冗余, __ack 与 heartbeat 是内部线程, 占用资源很小####"

- "####下面为初始分布, 用于同一台机器之间没有宕机的调度####"

- "####切记, 一个端口只能分配一个 shard, 千万不要将 merge 与 shard 或 shard 与 shard 分配到同一个端口里, 这会引起计算数据的不准确的####"

```
- "merge:1&shard@0:0~4;PUSH-001:6601,6701~6705;6602,6711~6715;6603,6721~6725"
- "merge:2&shard@0:5~9;PUSH-002:6601,6701~6705;6602,6711~6715;6603,6721~6725"
- "merge:3&shard@0:10~14;PUSH-003:6601,6701~6705;6602,6711~6715;6603,6721~6725"
- "merge:4&shard@0:15~19;PUSH-004:6601,6701~6705;6602,6711~6715;6603,6721~6725"
- "merge:5&shard@0:20~24;PUSH-005:6601,6701~6705;6602,6711~6715;6603,6721~6725"
- "merge:6&shard@0:25~29;PUSH-006:6601,6701~6705;6602,6711~6715;6603,6721~6725"
- "merge:7&shard@0:30~34;PUSH-007:6601,6701~6705;6602,6711~6715;6603,6721~6725"
- "merge:8&shard@0:35~39;PUSH-008:6601,6701~6705;6602,6711~6715;6603,6721~6725"
- "merge:9&shard@0:40~44;PUSH-014:6601,6701~6705;6602,6711~6715;6603,6721~6725"
- "merge:10&shard@0:45~49;PUSH-015:6601,6701~6705;6602,6711~6715;6603,6721~6725"
- "merge:11&shard@0:50~54;PUSH-016:6601,6701~6705;6602,6711~6715;6603,6721~6725"
- "shard@0:55~59;PUSH-017:6701~6705;6711~6715;6721~6725"
- "__ack:0;PUSH-017:6601;6602;6603"
- "heartbeat:0;PUSH-017:6601;6602;6603"
- "merge:0;PUSH-017:6601;6602;6603"
```

merge 的算法: 绿色, 共 12 个 (序号 0 到 11), 注意最后一行的 "merge:0"。

shard 的算法: 蓝色, 共 60 个 (序号 0~59)。

其中, "merge:0" 和 "shard@0" 是分开写的, 原因 延年 已经在注释里解释过了。

"##__ack,heartbeat,merge:0 他们共用同一个端口, 记住要时时刻刻, 他们共享的都是同一个端口, 别拆开了##"

延年用了 11 台机器, 也配了 60 个 shard。注意, 其中有一台机器上 6 个端口的, 比如,

- "merge:6&shard@0:25~30;adhoc5.kgb.cm6:6601,6701~6706;6602,6711~6716;6603,6721~6726"

这个和前面的 5 个并行, 是否冲突呢? 既然 延年 的配置是这样, 应该不冲突, **继续领悟...**!

启动后, 会看见 mdrill 的日志文件夹 (logs) 中, 日志的文件命名是包含端口号的

```
[root@psb-la-vm06 logs]# ll *.log
```

```
Apr 30 11:16 mdrillui.log
```

```
May 27 18:41 nimbus.log
```

```
May 27 18:37 supervisor.log
```

```
May 27 18:37 worker-6601.log
```

```
May 27 18:37 worker-6701.log
```

```
Apr 18 21:57 worker-6702.log
```

```
Apr 18 21:57 worker-6703.log
```

```
May 27 18:37 worker-6704.log
```

```
Apr 18 21:57 worker-6705.log
```

启动表以后, 会看见上面的这些 worker。这些就是这台机器上的 merge(6601)和 shard(6701~6705)啦。配置

里的其他端口，应该是备用的。

实时表的配置参数：见前面的配置文件

higo.mode.feedbackred: "@realtime@cleanPartionDays:90@"

实时数据，保存 90 天。

“本地硬盘上的数据会自动清除，hdfs 上的还是得手动清除。”

feedback-threads: 12

feedback-threads_reduce: 12

这个不确定：感觉是集群中启动的读数据进程数。延年说是线程数，进程数的控制在启动时候的参数里体现。

设置成 12, 4 台机器(根据数据量, 不需要把 mdrill 集群的所有机器资源都放在读数据上), 每台 3 个 worker, 似乎也对得上。

mdrill.task.ports.feedbacktopology: "6902"

mdrill.task.assignment.feedbacktopology:

- "map_feedback:0~2;PUSH-004:6801~6803;6804~6806"
- "map_feedback:3~5;PUSH-005:6801~6803;6804~6806"
- "map_feedback:6~8;PUSH-006:6801~6803;6804~6806"
- "map_feedback:9~11;PUSH-007:6801~6803;6804~6806"
- "reduce_feedback:0~2;PUSH-004:6801~6803;6804~6806"
- "reduce_feedback:3~5;PUSH-005:6801~6803;6804~6806"
- "reduce_feedback:6~8;PUSH-006:6801~6803;6804~6806"
- "reduce_feedback:9~11;PUSH-007:6801~6803;6804~6806"

同样地，启动读表任务后，会看见下面的日志文件。数字也和上面的第一组端口号相符。后面的也应该是配用。

读表任务启动后，会看见相应的下面的日志

May 27 18:08 worker-6801.log

May 27 18:08 worker-6802.log

May 27 18:08 worker-6803.log

读任务个数和表数量的困扰

可以从源码的配置文件看出，一个 topology 任务可以为多个表读数据。启动中的参数包括表名列表名列表页可以看出这一点。

```
./bluewhale      jar      ./mdrill.jar      com.alimama.mdrillImport.Topology      feedbacktopology  
feedback,feedbackred,hit,hitred 2 512 1000 2014030520050
```

如果需要，也可以为某些表，单独配置 topology。

2014/6/23 更新：延年说，4 个机器读，数据就会写在那 4 台机器的本地，负载也就在那 4 台机器上。12 台机器其实也就用了 4 台。果断改成 12 台配置。

feedback-mode: "merger"

在 reader 和 parser 小结会详细说。

4.2 实时表的 reader 的 parser

在源码的 adhoc-internal module 里面有例子（比如 P4PPVLogParser）可以参考。

Parser 中的 `getGroupName()`和 `getSumName()`方法返回的是列名的数组，里面 `DataIter` 实现的 `getGroup()`和 `getSum()`返回的是对应的列的值（注意顺序一定要一致）。列名被这两个函数分成两部分，在 `group` 里面的是不能合并的字段，在 `sum` 里面的是可以合并的字段。

合并的意义：节省记录数，提高查询速度。

比如，字段为 `pid`, `operator`, `time`, `records`。这里的 `records` 字段是统计原始的记录数的，配合 `mdrill` 的 `group` 和 `sum` 使用结果极佳。按照 `mdrill` 的例子，可以这样设置：

```
getGroupName() = {pid, operator, time}; getSumName() = {records};
```

比如有 2 行数据：

```
(a, cmcc, 2014/6/23 12:12:12, 1)
```

```
(a, cmcc, 2014/6/23 12:12:12, 1)
```

也就是说，用户 `a` 在同一时刻有 2 条访问记录，通过上面的 `group` 和 `name` 的设置，`mdrill` 会存储为

```
(a, cmcc, 2014/6/23 12:12:12, 2)
```

一条记录。

但是，`time` 字段有可能使得，`group` 字段的相同内容很少，很难合并。这个时候，例子 `parser` 的另外一列 (`miniute_5`)就发挥作用了。它可以用来设置数据的时间粒度。如此设置列名（分钟拼写有错误^_^）：

```
getGroupName() = {pid, operator, miniute_5}; getSumName() = {records};
```

假设我们有这样两列数据（时间不同）：

```
(a, cmcc, 2014/6/23 12:12:12, 1)
```

```
(a, cmcc, 2014/6/23 12:13:13, 1)
```

`mdrill` 会将数据存为：

```
(a, cmcc, 12:10, 2)
```

注意，具体的 `time` 不在存储，5 分钟以内的数据被认为是同样的时间。不用担心 `time` 里的日期信息，因为 `mdrill` 的 `thedata` 字段是必须的，最终的字段定义为：

```
getGroupName() = {thedata, pid, operator, miniute_5}; getSumName() = {records};
```

上面的数据被存储为：

```
(20140623, a, cmcc, 12:10, 2)
```

这样的合并数据的方法，非常适合我们对于 `pv` 的统计需求。时间精确到 5 分钟已经足够了，查询快速的需求来的更猛烈。

进一步合并：如果 `pv` 量巨大，`pid` 也为了合并的瓶颈。这个时候就需要其他的辅助手段让 `mdrill` 进一步合并。比如，一天内第二次访问的 `pid`，则 `pid` 设置为常量。而 `pid` 的上一次访问时间，可以用缓存来记录。这样，`mdrill` 可以进一步合并数据，可以统计 `pv`，也可以统计天粒度的 `uv`。

如果需要在实时表的时候，需要做数据合并，需要在设置中：

feedback-mode: "merger"

4.3 冗余列和多列

参考，mdrill 的例子，我们可以看到 hour 字段，这个字段是数据的小时字段，消失的信息可以从 minute_5 里获得，但是有了这个字段可以通过 group by hour 使小时粒度的统计变得更方便。

查询的速度依赖于扫描纪录数，比如说我们需要统计字段，比如说，result 的值的分布。而 result 有 0,1,2 这样的值。

```
getGroupName() = {thedata, pid, operator, result, minute_5}; getSumName() = {records};
```

如果数据为：

```
(20140623, a, cmcc, 1,12:10, 1)
```

```
(20140623, a, cmcc, 1,12:10, 1)
```

```
(20140623, a, cmcc, 2,12:10, 1)
```

我们可以把 result 的值分成多列：

```
getGroupName() = {thedata, pid, operator, minute_5}; getSumName() = {val_0, val_1, val_2, records};
```

被 mdrill 存储的数据为：

```
(20140623, a, cmcc, 12:10, 0, 2, 0, 2)
```

```
(20140623, a, cmcc, 12:10, 0, 0, 1, 1)
```

这样，在统计 result=2 的记录时候，mdrill 扫描的记录数会减少。

4.4 添加表或字段

添加表步骤：

建表，停表，启动表（启动表的时候，加上新建的表），之后再 ui 上可以看到。

加字段的步骤：

停表，重新运行建表命令（加上新字段），启动表。这时，ui 的 schema 中也许没有显示出新的字段，但是已经可以支持对新字段的查询了。对于以前的数据，新字段为 null。

4.5 空值 null

对于空值 null 如果有统计需求，可以在入库前，用特殊值处理。特殊值可以选“nil”，“unknown”等。注意，延年说，空字符串和 null 都建索引，也就是不支持查询。

4.6 数据的清理

数据被 mdrill 存放在这么几个地方：

本地：

```
higo.workdir.list: "/data/mdrill"
```

hdfs：

```
higo.table.path: "/mdrill/tablelist"
```

zk 里面也有，可以量很大，但是可以通过 zk 设置的快照个数设置来控制。
storm 文件夹也有数据，但是量很少，可以不用管。

实时表的数据清理可以通过下面设置来做：

higo.mode.feedbackred: "@realtime@cleanPartionDays:90@"

90 天后，服务器本地的数据会自动被 mdrill 清除。hdfs 里的数据需要手动删除。

如果在配置实时表的时候，没有设置 cleanPartionDays，就手动清理。

清理步骤：

停表，删除本地的表名下面的日期文件夹，删除 hdfs 中表名对应的日期文件夹。

```
[root@PUSH-016 feedback]# pwd
```

```
/data/mdrill/higo/feedback
```

```
[root@PUSH-016 feedback]# ll
```

```
total 8
```

```
drwxr-xr-x 7 root root 4096 Jun 23 15:55 20140623
```

```
drwxr-xr-x 7 root root 4096 Jun 24 00:23 20140624
```

```
[root@PUSH-016 feedback]# hadoop fs -ls /mdrill/tablelist/feedback/index
```

```
Found 9 items
```

```
drwxr-xr-x - root root 0 2014-06-16 04:04 /mdrill/tablelist/feedback/index/20140616
```

```
drwxr-xr-x - root root 0 2014-06-17 04:02 /mdrill/tablelist/feedback/index/20140617
```

```
drwxr-xr-x - root root 0 2014-06-18 04:02 /mdrill/tablelist/feedback/index/20140618
```

```
drwxr-xr-x - root root 0 2014-06-19 04:01 /mdrill/tablelist/feedback/index/20140619
```

```
drwxr-xr-x - root root 0 2014-06-21 00:02 /mdrill/tablelist/feedback/index/20140620
```

```
drwxr-xr-x - root root 0 2014-06-21 04:03 /mdrill/tablelist/feedback/index/20140621
```

```
drwxr-xr-x - root root 0 2014-06-22 04:02 /mdrill/tablelist/feedback/index/20140622
```

```
drwxr-xr-x - root root 0 2014-06-23 20:52 /mdrill/tablelist/feedback/index/20140623
```

```
drwxr-xr-x - root root 0 2014-06-24 04:10 /mdrill/tablelist/feedback/index/20140624
```

延年 说不需要停表。可以试验的时候如果不停，删除的文件夹有可能会自动回复。

4.7 failCnt

实时表的日志中，注意 failCnt 的值，数量应该很小才说明数据入库正常

```
# tail -f /data/alimama/adhoc-core/logs/worker-6801.log
```

```
2014-06-24 17:22:48 ImportSpout [INFO] feedback####total:group=584,ts:17:22:40,status:SpoutStatus  
[ackCnt=301130, failCnt=0, ttInput=754450, groupInput=377225, groupCreate=301714]
```

4.8 mdrill jdbc

代码查询很简单，可以参考 mdrill 文档的例子。

4.8.1 Maven 依赖

如果是 mvn 工程，可以依赖

```
<dependency>
  <groupId>com.alimama.dw</groupId>
  <artifactId>adhoc-jdbc</artifactId>
  <version>0.18-beta</version>
</dependency>
```

4.8.2 Spring 注入

如果工程中使用了 Spring 或者 mybatis 框架，可以这样配置：

```
<bean id="dataSourceMdrill"
      class="org.springframework.jdbc.datasource.DriverManagerDataSource">
  <property name="driverClassName">
    <value>com.alimama.mdrill.jdbc.MdrillDriver</value>
  </property>
  <property name="url">
    <value>jdbc:mdrill://10.0.4.102:1107</value>
  </property>
</bean>

<bean id="sqlSessionFactoryMdrill" class="org.mybatis.spring.SqlSessionFactoryBean"
      name="sqlSessionFactoryMdrill">
  <property name="dataSource" ref="dataSourceMdrill" />
  <property name="configLocation" value="classpath:mybatis1.xml" />
  <property name="mapperLocations">
    <list>
      <value>classpath:config/mybatis/sql1/*-mapper.xml</value>
      <value>classpath:config/mybatis/resultMap1/*-resultMap.xml</value>
    </list>
  </property>
</bean>

<bean id="sqlSessionTemplateMdrill"
      class="org.mybatis.spring.SqlSessionTemplate">
  <constructor-arg index="0" ref="sqlSessionFactoryMdrill" />
</bean>

<!-- 自动扫描cn.info.platform.mapper包下的Mapper接口,并实现其功能 -->
<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
  <property name="sqlSessionTemplateBeanName" value="sqlSessionTemplateMdrill"/>
</bean>
```

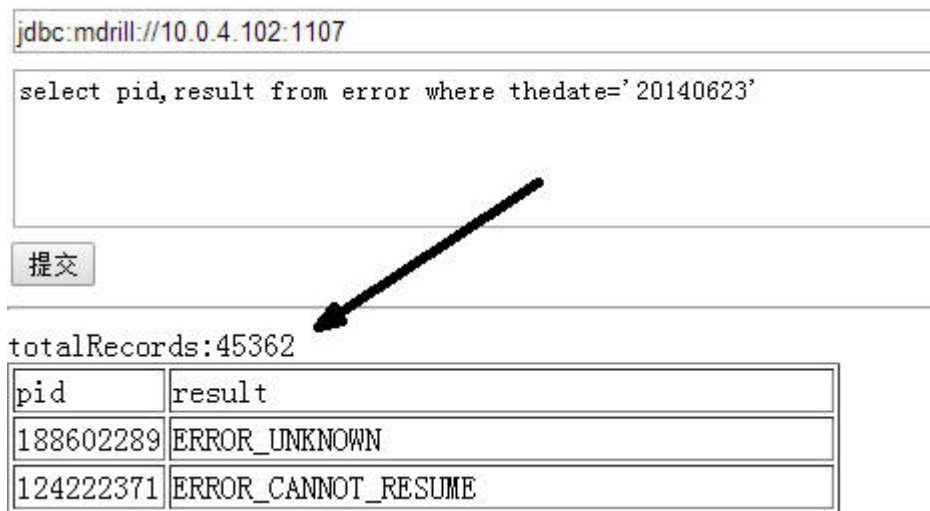
```
<property name="basePackage" value="com.Lenovo.lps.push.marketing.monitor.mapper" />
<property name="markerInterface"
value="com.Lenovo.lps.push.marketing.monitor.mapper.BaseMapper" />
</bean>
```

修改 MdrillResultSetMetaData 中的

```
public String getColumnClassName(int column) throws SQLException {
    //throw new SQLException("Method not supported");
    return "java.lang.String";
}
```

4.8.3 GetTotal

对于 mdrill 的查询，limit 的默认值是 0,20。从 mdrill 的 ui 上可以看到，查询可以返回总的记录数，这个值对于分页的处理非常有用。



jdbc:mdrill://10.0.4.102:1107

select pid,result from error where thedate='20140623'

提交

totalRecords:45362

pid	result
188602289	ERROR_UNKNOWN
124222371	ERROR_CANNOT_RESUME

这个值可以通过 MdrillQueryResultSet 的 getTotal()方法获得。如果用 mybatis，可以通过拦截器来实现。

<http://haohaoxuexi.iteye.com/blog/1851081>

<http://wyuxiao729.iteye.com/blog/1871707>

<http://www.cnblogs.com/samkin/articles/2864689.html>

源码参考 Stream 工程。

5 Stream

Stream 为系统监控的 UI 展示，数据来自对 mdrill 的查询。

源代码 svn:

<http://rta.lenovo.com/svn/beacon09/06 Service/Project/06 Code/LpsPushService2.0/PushMarketingMonitor>

打包方法: eclipse 导出 war 包。

部署位置:

psb-la-vm04

环境准备: mysql, Tomcat6

安装 tomcat6

```
# yum install tomcat6 tomcat6-webapps tomcat6-admin-webapps tomcat6-docs-webapp tomcat6-javadoc
```

```
# service tomcat6 status
```

```
tomcat6 is stopped [ OK ]
```

```
[root@master mydownload]# service tomcat6 start
```

```
Starting tomcat6: [ OK ]
```

```
[root@master mydownload]# chkconfig tomcat6 on
```

停止 tomcat

```
[root@master mydownload]# service tomcat6 stop
```

修改配置文件

```
[root@psb-la-vm04 conf]# pwd
```

```
/usr/share/tomcat6/conf
```

```
[root@psb-la-vm04 conf]# vi tomcat6.conf
```

```
JAVA_OPTS="-server -Xms1024m -Xmx1024m -XX:PermSize=128M -XX:MaxNewSize=400m  
-XX:MaxPermSize=256m -Djava.awt.headless=true "
```

修改 logs 文件夹的软连接

```
[root@psb-la-vm04 tomcat6]# pwd
```

```
/usr/share/tomcat6
```

```
[root@psb-la-vm04 tomcat6]# ll logs
```

```
lrwxrwxrwx 1 root root 17 May 23 09:41 logs -> /data/tomcatlogs/
```

安装 mysql

```
# yum install mysql mysql-server mysql-devel mysql-connector-odbc libdbi-dbd-mysql
```

修改配置

```
[root@psb-la-vm04 ~]# vi /etc/my.cnf
```

```
[mysqld]
```

```
datadir=/var/lib/mysql
```

```
socket=/var/lib/mysql/mysql.sock
```

```
user=mysql
```

```
# Disabling symbolic-links is recommended to prevent assorted security risks
```

```
symbolic-links=0
```

```
max_allowed_packet=128M
```

```
default-character-set=utf8
```


[client]

default-character-set=utf8

[mysqld_safe]

log-error=/var/log/mysql.log

pid-file=/var/run/mysql/mysql.pid

启动/停止

service mysqld status

service mysqld start

service mysqld stop

chkconfig mysqld on

导入数据库

db_avatarui.sql 在源码的 sql 目录下

[root@psb-la-vm04 ~]# mysql -uroot

source /home/bastion/mydownload/db_avatarui.sql

部署

将 eclipse 导出的 war 包放在/usr/share/tomcat6/webapps 目录下面，启动 tomcat6

service tomcat6 start



6 Kafka

Kafka 和其依赖的 Zookeeper 目前部署在北京机房。

psb-la-vm02: zk

psb-la-vm04: zk, kafka

psb-la-vm05: zk, kafka

Kafka 集群 2 台，Zookeeper 集群 3 台（HA 集群的最小机器数）。

6.1 Zookeeper 集群

Zookeeper 的部署（可以和前面 CDH 使用 Manager 安装的部署过程进行比较）

参考文档

<http://blog.csdn.net/jmy99527/article/details/17582349>

没有使用 cdh 集群的 zk(在天津机房，而 Kafka 目前部署在北京)；也没有使用 Kafka 自带的 zk（配置项不熟悉）。

安装和使用 zk 使用了 hadoop 组的 hadoop 用户，可以下面的命令添加

```
# /usr/sbin/groupadd hadoop
```

```
# /usr/sbin/useradd hadoop -g hadoop
```

下载 Zookeeper

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

```
zookeeper-3.4.5.tar.gz
```

```
# wget http://mirrors.cnnic.cn/apache/zookeeper/zookeeper-3.4.5/zookeeper-3.4.5.tar.gz
```

```
# tar -zxvf zookeeper-3.4.5.tar.gz
```

```
# mkdir /opt/modules/zookeeper
```

```
# mv zookeeper-3.4.5 /opt/modules/zookeeper/
```

```
# chown -R hadoop:hadoop /opt/modules/zookeeper/
```

```
# su hadoop
```

```
$ /opt/modules/zookeeper/zookeeper-3.4.5/conf
```

```
$ cp zoo_sample.cfg zoo.cfg
```

```
# 修改配置
```

```
$ vi zoo.cfg
```

```
# The number of milliseconds of each tick
```

```
tickTime=2000
```

```
# The number of ticks that the initial
```

```
# synchronization phase can take
initLimit=10
# The number of ticks that can pass between
# sending a request and getting an acknowledgement
syncLimit=5
# the directory where the snapshot is stored.
# do not use /tmp for storage, /tmp here is just
# example sakes.
#dataDir=/tmp/zookeeper
dataDir=/data/tmp/zookeeper/
# the port at which the clients will connect
clientPort=2181
#
# Be sure to read the maintenance section of the
# administrator guide before turning on autopurge.
#
# http://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc_maintenance
#
# The number of snapshots to retain in dataDir
autopurge.snapRetainCount=100
# Purge task interval in hours
# Set to "0" to disable auto purge feature
autopurge.purgeInterval=12
server.2=psb-la-vm02:2888:3888
server.4=psb-la-vm04:2888:3888
server.5=psb-la-vm05:2888:3888
```

准备文件夹: dataDir

```
mkdir /data/tmp/zookeeper
```

```
chown -R hadoop:hadoop /data/tmp/zookeeper
```

设置 zk 服务器在集群中的 id(每台机器使用不同的值, 例子中用的是“2”)

```
echo "2" > /data/tmp/zookeeper/myid
```

默认的 logs 卸载 zk 的安装目录下面, 如果需要可以按照下面的步骤更改

<http://yangyoupeng-cn-fujitsu-com.iteye.com/blog/1922459>

```
mkdir /data/tmp/zklog
```

```
chown -R hadoop:hadoop /data/tmp/zklog
```

cd 到 zk 的安装目录下面

```
[root@psb-la-vm02 zookeeper-3.4.5]# cd /opt/modules/zookeeper/zookeeper-3.4.5
```

```
[root@psb-la-vm02 zookeeper-3.4.5]# vi conf/log4j.properties
```

```
#zookeeper.root.logger=INFO, CONSOLE
```

```
zookeeper.root.logger=INFO, ROLLINGFILE
```

```
...
#zookeeper.log.dir=.
zookeeper.log.dir=/data/tmp/zklog
zookeeper.log.file=zookeeper.log
zookeeper.log.threshold=DEBUG
#zookeeper.tracelog.dir=.
zookeeper.tracelog.dir=/data/tmp/zklog
```

```
[root@psb-la-vm02 zookeeper-3.4.5]# vi bin/zkEnv.sh
```

```
...
if [ "x${ZOO_LOG_DIR}" = "x" ]
then
    #ZOO_LOG_DIR="."
    ZOO_LOG_DIR="/data/tmp/zklog/"
fi

if [ "x${ZOO_LOG4J_PROP}" = "x" ]
then
#    ZOO_LOG4J_PROP="INFO,CONSOLE"
    ZOO_LOG4J_PROP="INFO,ROLLINGFILE"
fi
```

zk 的启动和停止（需要在每台服务器上执行）

进入安装目录下的 bin（使用 hadoop 用户）

su hadoop

启动

zkServer.sh start

停止

zkServer.sh stop

检查状态

```
[hadoop@psb-la-vm04 root]$ zkServer.sh status
```

```
JMX enabled by default
```

```
Using config: /opt/modules/zookeeper/zookeeper-3.4.5/bin/./conf/zoo.cfg
```

```
Mode: follower
```

zkCli.sh 进入之后可以查看。

6.2 Kafka 集群

Kafka 主页

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

下载 kafka_2.8.0-0.8.1.tgz
解压部署步骤见官方文档。

<http://kafka.apache.org/documentation.html>

Kafka 的安装和启动都使用的是 root 用户

集群所在服务器

psb-la-vm05, psb-la-vm04

服务器上的部署目录位置

/data/kafka/kafka_2.8.0-0.8.1

之后在/opt/modules/kafka 下面建立连接（可选）

```
# ln -s /data/kafka/kafka_2.8.0-0.8.1 ./kafka_2.8.0-0.8.1
```

修改配置（已有的项目修改，否则就增加）

```
# vi config/server.properties
```

```
# The id of the broker. This must be set to a unique integer for each broker.
```

```
broker.id=4
```

```
#log.dirs=/tmp/kafka-logs
```

```
log.dirs=/data/tmp/kafkalog
```

```
auto.create.topics.enable=false
```

```
# The minimum age of a log file to be eligible for deletion
```

```
# log.retention.hours=168
```

```
log.retention.hours=72
```

```
#zookeeper.connect=localhost:2181
```

```
zookeeper.connect=psb-la-vm04:2181,psb-la-vm05:2181,psb-la-vm02:2181
```

启动 Kafka（需要在每台服务器上执行）

```
nohup bin/kafka-server-start.sh config/server.properties > sv.out &
```

停止

```
bin/kafka-server-stop.sh config/server.properties
```

建立 topic（只需在任何一台 Kafka 机器上执行）

```
bin/kafka-topics.sh --create --zookeeper psb-la-vm05:2181,psb-la-vm04:2181 --replication-factor 2 --partitions 24  
--topic feedback-topic
```

```
bin/kafka-topics.sh --create --zookeeper psb-la-vm05:2181,psb-la-vm04:2181 --replication-factor 2 --partitions 24  
--topic hit-topic
```

```
bin/kafka-topics.sh --create --zookeeper psb-la-vm05:2181,psb-la-vm04:2181 --replication-factor 2 --partitions 24  
--topic error-topic
```

测试生成和消费

生产

bin/kafka-console-producer.sh --broker-list psb-la-vm05:9092,psb-la-vm04:9092 --topic feedback-topic
之后在输入消息，回车。

bin/kafka-console-consumer.sh --zookeeper psb-la-vm05:2181,psb-la-vm04:2181 --topic feedback-topic
可以看到生产者发送的消息。

6.3 Java API

生产者

<http://rta.lenovo.com/svn/beacon09/06 Service/Project/06 Code/LpsPushService2.0/PushMarketing2>

消费者

<http://rta.lenovo.com/svn/beacon09/06 Service/Project/06 Code/LpsPushService2.0/LeDrill>

6.4 Kafka 集群监控

列出所有 topic

bin/kafka-topics.sh --list --zookeeper psb-la-vm05:2181,psb-la-vm04:2181

查看 topic 的情况

bin/kafka-topics.sh --describe --zookeeper psb-la-vm05:2181,psb-la-vm04:2181 --topic feedback-topic

监控消费情况

bin/kafka-run-class.sh kafka.tools.ConsumerOffsetChecker --group logstash --zkconnect localhost:2181 --topic feedback-topic

WebUI

<https://github.com/quantifind/KafkaOffsetMonitor/>

<http://blog.csdn.net/lizhitao/article/details/27199863>

下载

KafkaOffsetMonitor-assembly-0.2.0.jar

部署位置

[root@PUSH-016 feedback]# cd /data/kmonitor/

[root@PUSH-016 kmonitor]# ll

-rw-r--r-- 1 root root 57786330 Jun 18 17:59 KafkaOffsetMonitor-assembly-0.2.0.jar

-rw-r--r-- 1 root root 1075 Jun 20 10:40 km.out

-rw-r--r-- 1 root root 46487552 Jun 24 12:15 offsetapp.db

编写 start-up.sh

vi start-up.sh

```
#!/bin/bash
```

```
nohup java -Xms512M -Xmx512M -Xss1024K -XX:PermSize=256m -XX:MaxPermSize=512m -cp  
KafkaOffsetMonitor-assembly-0.2.0.jar com.quantifind.kafka.offsetapp.OffsetGetterWeb --zk  
psb-la-vm04:2181,psb-la-vm05:2181 --port 8092 --refresh 300.seconds --retain 3.days > km.out &
```

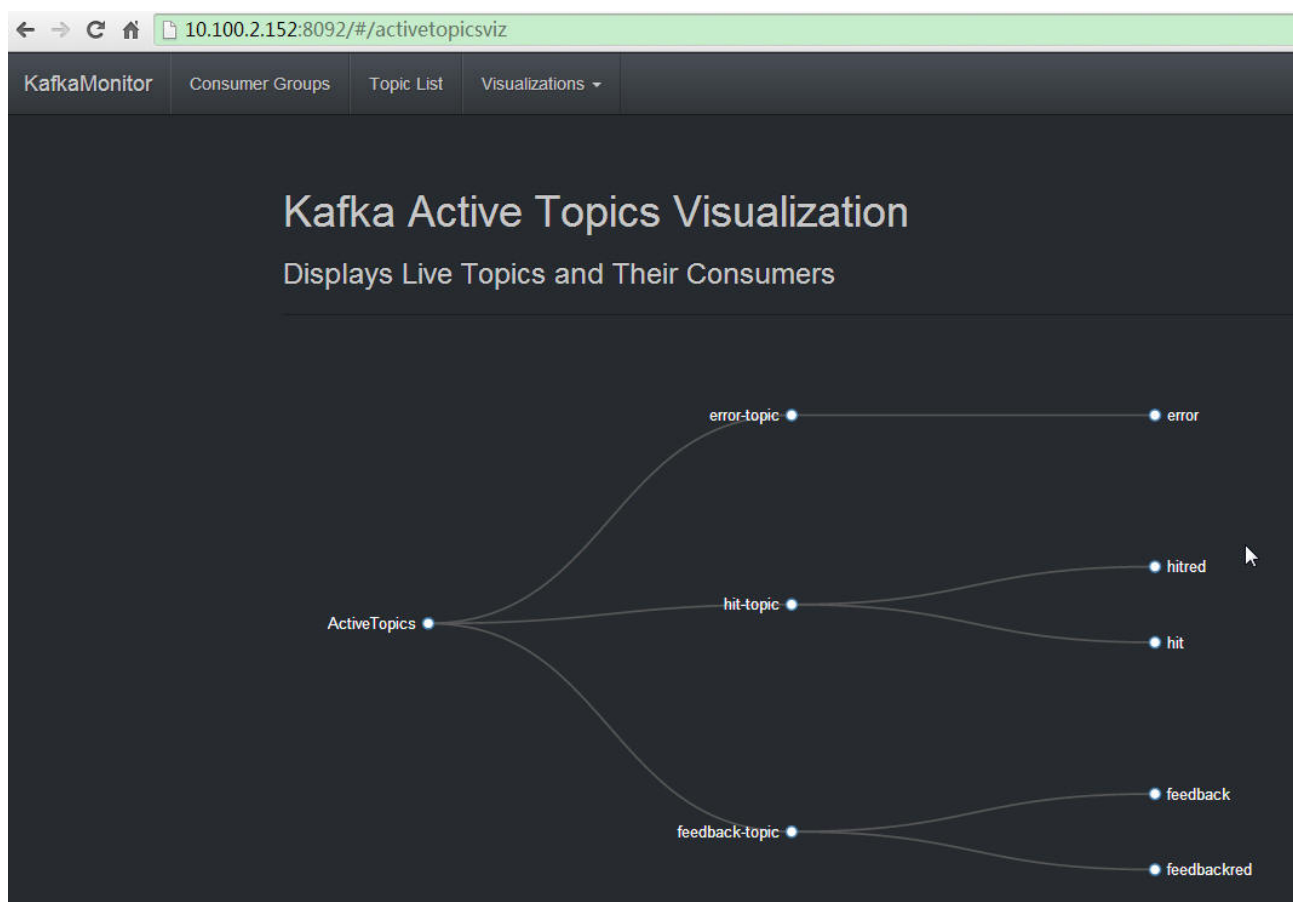
```
# chmod +x ./start-up.sh
```

运行

```
# ./start-up.sh
```

```
[root@PUSH-016 kmonitor]# jps
```

```
23772 OffsetGetterWeb
```



7 协议说明

7.1 PS 与 mdrill

Json 格式

使用 Java API: Gson


```
<dependency>
  <groupId>com.google.code.gson</groupId>
  <artifactId>gson</artifactId>
  <version>2.2.4</version>
</dependency>
```

8 骚扰度

8.1 数据的读取

8.1.1 Logstash

从 kafka 中通过 logstash 的 kafka 插件 (<https://github.com/joekiller/logstash-kafka>) 将数据保存到本地。
配置文件:

 logstash_file.conf.624

部署位置:

4 台服务器: PUSH-016, 17, 19, 20

解压 logstash.zip.613 文件到/data/logstash/

将 logstash_file.conf 复制到/data/logstash/logstash/logstash-1.4.1

启动 logstash

```
[root@PUSH-016 logstash-1.4.1]# pwd
```

```
/data/logstash/logstash/logstash-1.4.1
```

```
[root@PUSH-016 logstash-1.4.1]# nohup bin/logstash agent -f logstash_file.conf > ls.out &
```

数据文件

```
[root@PUSH-019 data]# pwd
```

```
/data/logstash/logstash/data
```

```
[root@PUSH-019 data]# ll
```

```
total 19884676
```

```
-rw-r--r-- 1 root root    3271496 Jun 24 16:00 afile-PUSH-019-2014-06-24-15-11-13672
```

```
-rw-r--r-- 1 root root    665636 Jun 24 15:25 afile-PUSH-019-2014-06-24-15-4-13672
```

```
-rw-r--r-- 1 root root   537059529 Jun 24 16:06 tfile-PUSH-019-2014-06-24-16-0-13672
```

```
-rw-r--r-- 1 root root 1066947942 Jun 24 16:55 tfile-PUSH-019-2014-06-24-16-10-13672
```


说明:

tfile: 归档数据

afile: 给 Avatar 的数据

命名规则:

afile-PUSH-019-2014-06-24-15-11-13672

afile-: 文件名前缀

PUSH-019: 主机名

-2014-06-24-: 日志日期

15-: 日志小时

11-: 日志第 n 个 5 分钟

13672: 进程号

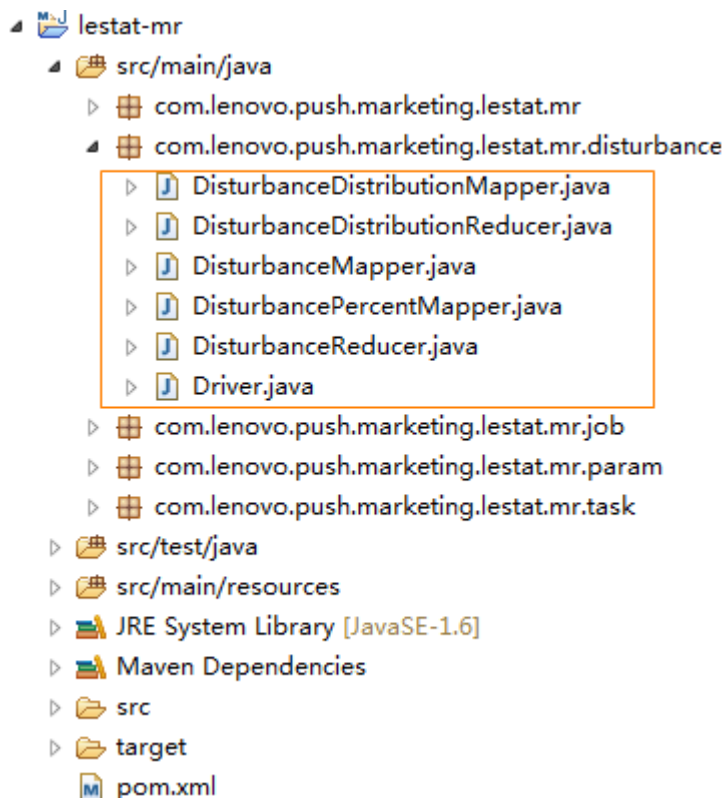
8.1.2 Java

TODO

8.2 骚扰度的计算

8.2.1 源码位置

骚扰度计算程序作为 LeStat 项目的一个独立模块存在，若 LeStat 项目的主目录为\$LeStat，骚扰度计算模块的路径为：\$LeStat/trunk/lestat-all/lestat-mr，核心 MapReduce 代码位于 src/main/java 路径下的 com.lenovo.push.marketing.lestat.mr.disturbance 包中，详见下图。



8.2.2 打包方法

安装 ledrill-common 模块

LeStat 项目依赖 LeDrill 项目中的 ledrill-common 模块，要打包 LeStat 项目，首先需要在本地安装 ledrill-common 模块，设 LeDrill 项目的主目录为\$LeDrill，使用如下命令将 ledrill-common 模块安装在本地 maven 库中。

```
cd $LeDrill\mdrill\0.20.9\ledrill-common
```

```
mvn install
```

打包 LeStat 项目

使用如下命令打包 LeStat 项目

```
cd $LeStat\trunk\lestat-all
```

```
mvn clean package
```

命令成功执行后，在路径\$LeStat\ trunk\lestat-all\lestat-core\target 下会生成 LeStat 项目的压缩包 lestat-0.0.1-SNAPSHOT.tar.gz。如下图所示。



8.2.3 部署位置

天津机房 PUSH-017 结点上，以 root 用户身份，使用如下命令创建 lestat 用户和组。

```
groupadd lestat
```

```
useradd lestat -g lestat
```

为 LeStat 项目创建文件夹，并修改所有者为 lestat

```
mkdir /data/lestat/
```

```
chown -R lestat:lestat /data/lestat
```

将上一步打好的 LeStat 项目包上传到 PUSH-017 结点的/data/lestat 中，并修改所有者权限。

```
chown lestat:lestat /data/lestat/lestat-0.0.1-SNAPSHOT.tar.gz。
```

切换为 lestat 用户

```
su lestat
```

解压 LeStat 项目包

```
cd /data/lestat
```

```
tar xzf lestat-0.0.1-SNAPSHOT.tar.gz。
```

解压后会在文件夹/data/lestat 中生成 LeStat 项目的文件夹 lestat-0.0.1-SNAPSHOT。

将启动脚本从 Windows 格式转换为 Unix 格式。

```
cd /data/lestat/lestat-0.0.1-SNAPSHOT/bin
```

```
dos2unix *
```

至此，LeStat 项目部署完毕。

8.2.4 环境准备

骚扰度计算结果需要存入 MySQL 数据库中，为此北京机房 PSB-LA-VM04 结点上创建存放骚扰度结果的数据表，并为天津 PUSH-017 结点配置访问权限，用户名及密码。

首先在结点 PSB-LA-VM04 上使用下面的命令以 root 用户身份登录 MySQL。

```
mysql -uroot -p
```

在数据库 db_stream 中创建 disturbance 表。

```
CREATE TABLE `db_stream`.`disturbance` (  
  `id` INT NOT NULL AUTO_INCREMENT,  
  `thedata` VARCHAR(45) NOT NULL,  
  `hit` INT NOT NULL,  
  `users` INT NOT NULL,  
  `percent` DOUBLE NOT NULL,  
  `inserttime` TIMESTAMP NOT NULL,  
  PRIMARY KEY (`id`));
```

为 PUSH-017 结点上的 lestat 用户配置访问权限，用户名和密码。

```
grant select, insert, update, delete on db_stream.disturbance to lestat@'10.0.4.117' IDENTIFIED BY 'lestat';
```

8.2.5 运行方法

在天津机房 PUSH-017 结点上以 lestat 用户使用如下命令启动骚扰度计算程序。

```
cd /data/lestat/lestat-0.0.1-SNAPSHOT/bin
```

```
nohup ./mapred-startup.sh logstash> mapred.out &
```

若启动无误，使用 jps 命令会看到一个 APP 进程在后台运行。

9 尚未解决问题

1. mdrill 的宕机处理
2. 用户骚扰度查询
3. 数据的恢复

10 感谢

延年 * 无穷大，感谢 Apache，感谢 Kafka，kmonitor，感谢 Google 大神

Lenovo Confidential