**UDH Search系统构建及其应用**

作者：费英林

目录

[1. Search概要 3](#_Toc420422689)

[1.1. Search中索引创建的主要流程 3](#_Toc420422690)

[1.2. 主要组件 4](#_Toc420422691)

[2. Search安装与配置 4](#_Toc420422692)

[2.1. 安装节点选择 4](#_Toc420422693)

[2.2. 安装命令 4](#_Toc420422694)

[2.3. 初始化Solr 4](#_Toc420422695)

[2.4. 配置Solr 使用Secure HDFS 5](#_Toc420422696)

[2.5. 在HDFS中创建solr目录 5](#_Toc420422697)

[2.6. 在 ZooKeeper中初始化Solr命名空间 5](#_Toc420422698)

[2.7. 启动Solr 5](#_Toc420422699)

[2.8. 配置 – Hue Search 5](#_Toc420422700)

[3. 批量创建HBase索引 6](#_Toc420422701)

[3.1. 创建HBase表 6](#_Toc420422702)

[3.2. 生成collection 6](#_Toc420422703)

[3.3. 创建Lily HBase Indexer配置文件 6](#_Toc420422704)

[3.4. 创建Morphline配置文件 7](#_Toc420422705)

[3.5. 运行Lily HBase Indexer 7](#_Toc420422706)

[3.6. 验证 7](#_Toc420422707)

[3.7. 应用场景 8](#_Toc420422708)

[4. 实时生成HBase索引 8](#_Toc420422709)

[4.1. 建立HBase主从Replication集群 8](#_Toc420422710)

[4.2. 设置ZK集群 8](#_Toc420422711)

[4.3. 设置HBase表的列族为Replication模式 8](#_Toc420422712)

[4.4. 生成collection和配置文件 9](#_Toc420422713)

[4.5. 注册一个Indexer 9](#_Toc420422714)

[4.6. 在HBase主集群中新增数据，验证结果 9](#_Toc420422715)

[4.7. 应用场景 9](#_Toc420422716)

[5. 安装包 9](#_Toc420422717)

UDH Search是一个索引创建与检索服务，它与UDH高度集成，核心系统是Apache Solr　－　包括Apache Lucene，SolrCloud， Apache Tika和Solr Cell。 Search生成的索引数据存储在HDFS，避免了海量数据的移动问题，缩短了索引的创建周期；同时，HDFS的多数据备份保证了索引数据的安全性。借助于UDH Search，我们实现了对于存储在HDFS和HBase中的数据的近实时访问。在UDH平台，索引的创建有两种方式：批处理方式和近实时方式。

UDH Search提供了标准的Solr API供客户端调用，它还提供了一个简单的UI，可集成在Hue的框架里。 UDH Search的主要特性如下：

1）生成的索引文件存储在HDFS里

2）可以使用MapReduce批量创建索引

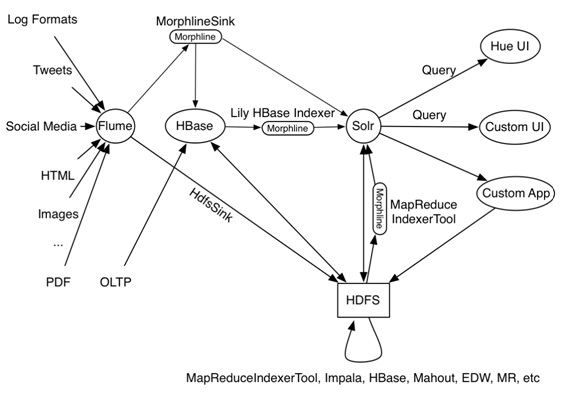
3）可以在数据导入HDFS或HBase时实时创建索引

4）可以与Hue集成，通过Hue实现界面访问

5）可以通过索引数据检索HBase数据

本文以HBase数据的索引创建为例，阐述了利用UDH Search的基本工作原理及配置方法，包括批处理方式和实时处理方式。

# Search概要



## Search中索引创建的主要流程

Search有三种方式创建索引：

1）实时索引创建

这是一种近实时的索引创建方式。通过Flume的配置文件，可以将输入数据中的字段映射到标准的Solr schema， Lucene建立索引并将索引数据写到HDFS，然后加载到Solr搜索引擎。

2）使用MapReduce批量创建索引

对于存储在HDFS中的数据，我们可启动一个MapReduce工作流，在Map阶段执行字段抽取和schema映射，在Reduce阶段生成索引文件或索引分片。

3）HBase数据索引创建

对于存储在HBase中的数据，有两种创建索引的方式：批处理方式和实时方式。

批处理方式 – 通过运行一个MapReduce作业，对已有数据批量创建索引；

实时方式 – 利用HBase Replication机制，实时获取HBase的数据流。

## 主要组件

1）Morphline - 是一个开源框架，用于辅助Search的索引创建工作。通过配置文件来定义ETL流程，可实现数据读取、数据处理及数据加载。

2）Lily HBase NRT Indexer - 在HBase启用了复制模式后，可通过Lily HBase NRT Indexer实时获取HBase的增删改数据，动态创建索引并加载到Solr服务器。

3）MapReduceIndexerTool – 批量索引创建工具，生成的索引文件存储在HDFS。

# Search安装与配置

## 安装节点选择

UDH Search是一个分布式索引创建框架，需要访问HDFS或HBase中的数据。为降低网络IO，提高索引创建效率，可以在每个DataNode或HBase RegionServer所在的主机上部署Search服务。考虑到Search服务可能会影响到整个集群的性能，我们也可以单独搭建一个索引服务器集群。

## 安装命令

安装Solr服务器：sudo yum install solr-server solr-doc

安装Hue Search服务：sudo yum install hue-search

安装Spark Indexer: sudo yum install solr-cruncMapReduceh

安装MapReduce Indexer：sudo yum install solr-mapreduce

安装Lily HBase Indexer：sudo yum install hbase-solr-indexer hbase-solr-doc

## 初始化Solr

编辑/etc/default/solr， 设置如下参数：

1）ZK集群：SOLR\_ZK\_ENSEMBLE=<zkhost1>:2181,<zkhost2>:2181,<zkhost3>:2181/solr

2）HDFS：SOLR\_HDFS\_HOME=hdfs://namenodehost:8020/solr

3）Solr HDFS client：SOLR\_HDFS\_CONFIG=/etc/hadoop/conf (包含core-site.xml/hdfs-site.xml的目录)

## 配置Solr 使用Secure HDFS

1）为每个Solr服务器创建principals/Keytab :

addprinc -randkey solr/fully.qualified.domain.name@YOUR-REALM.COM

xst -norandkey -k solr.keytab solr/fully.qualified.domain.name

2）在每个Solr服务器上配置Keytab :

sudo mv solr.keytab /etc/solr/conf/

sudo chown solr:hadoop /etc/solr/conf/solr.keytab

sudo chmod 400 /etc/solr/conf/solr.keytab

3）编辑/etc/default/solr：

SOLR\_KERBEROS\_ENABLED=true

SOLR\_KERBEROS\_KEYTAB=/etc/solr/conf/solr.keytab

SOLR\_KERBEROS\_PRINCIPAL=solr/fully.qualified.domain.name@YOUR-REALM.COM

## 在HDFS中创建solr目录

sudo -u hdfs hadoop fs -mkdir /solr

sudo -u hdfs hadoop fs -chown solr /solr

## 在 ZooKeeper中初始化Solr命名空间

solrctl init

solrctl init --force # 清除全部Solr data，中断所有运行的Solr服务器

## 启动Solr

sudo service solr-server restart

sudo jps –lm

## 配置 – Hue Search

1）设置solr\_url, 编辑/etc/hue/hue.ini:

# Specify the Solr URL:

[search]

solr\_url=http://SolrHostName:8983/solr/

2）启用安全（在安全集群里）, 编辑/etc/hue/hue.ini:

security\_enabled=true

3）启用代理, 编辑/etc/default/solr:

SOLR\_SECURITY\_ALLOWED\_PROXYUSERS=hue

SOLR\_SECURITY\_PROXYUSER\_hue\_HOSTS=\*

SOLR\_SECURITY\_PROXYUSER\_hue\_GROUPS=\*

4）验证

http://hue-host.com:8888/search/

# 批量创建HBase索引

Search有两种索引创建方式，即批量创建和实时创建。接下的两个章节将以HBase的索引创建为例，阐述相关的配置及运行方式。

## 创建HBase表

在HBase中生成一张测试表，并新增几条数据。例如：

$ hbase shell

hbase(main):002:0> create 'record', {NAME => 'data'}

hbase(main):002:0> put 'record', 'row1', 'data', 'value'

hbase(main):001:0> put 'record', 'row2', 'data', 'value2'

## 生成collection

1）生成一个collection的本地目录：

$ solrctl instancedir --generate $HOME/hbase-collection1

2）编辑这个collection的schema.xml，加入希望生成的字段：

<field name="data" type="string" indexed="true" stored="true" />

3）上传这个collection到ZK集群：

$ solrctl instancedir --create hbase-collection1 $HOME/hbase-collection1

$ solrctl collection --create hbase-collection1

## 创建Lily HBase Indexer配置文件

在collection的本地目录中生成Lily HBase Indexer配置文件，如

如 $HOME/morphline-hbase-mapper.xml

<?xml version="1.0"?>

<indexer table="record" mapper="com.ngdata.hbaseindexer.morphline.MorphlineResultToSolrMapper">

<param name="morphlineFile" value="/etc/hbase-solr/conf/morphlines.conf"/>

<!-- <param name="morphlineId" value="morphline1"/> -->

</indexer>

## 创建Morphline配置文件

生成相应的Morphline配置文件 /etc/hbase-solr/conf/morphlines.conf：

morphlines : [

{

id : morphline1

importCommands : ["org.kitesdk.morphline.\*\*", "com.ngdata.\*\*"]

commands : [

{

extractHBaseCells {

mappings : [

{

inputColumn : "data:\*"

outputField : "data"

type : string

source : value

}

]

}

}

{ logTrace { format : "output record: {}", args : ["@{}"] } }

]

}

]

## 运行Lily HBase Indexer

hadoop --config /etc/hadoop/conf jar \

/usr/lib/hbase-solr/tools/hbase-indexer-mr-\*-job.jar --conf \

/etc/hbase/conf/hbase-site.xml -D 'mapred.child.java.opts=-Xmx500m' \

--hbase-indexer-file $HOME/morphline-hbase-mapper.xml --zk-host \

127.0.0.1/solr --collection hbase-collection1 --go-live --log4j \

src/test/resources/log4j.properties

## 验证

在浏览器中输入地址<http://solrhost:8983/solr>，选择相应的collection，点击查询进行验证。

## 应用场景

批处理模式适用于数据量大，时效性要求不高的情况。

# 实时生成HBase索引

通过HBase的Replication机制，Search实现了HBase数据的（接近）实时索引创建。数据进入主集群，然后Replicate到从集群，从集群的HBase Indexer创建索引到在线Solr服务器。这个过程所用的时间通常是在秒级别。关于HBase备份集群的创建及配置方法请参考‘如何配置UDH HBase Replication集群’。

## 建立HBase主从Replication集群

参见‘如何配置UDH HBase Replication集群’

## 设置ZK集群

在/etc/hbase-solr/conf/hbase-indexer-site.xml中设置HBase Indexer需要访问的ZK集群：

<property>

<name>hbase.zookeeper.quorum</name>

<value>hbase-cluster-zookeeper</value>

</property>

<property>

<name>hbaseindexer.zookeeper.connectstring</name>

<value>hbase-cluster-zookeeper:2181</value>

</property>

## 设置HBase表的列族为Replication模式

对于已经存在的表：

hbase shell> disable 'record'

hbase shell> alter 'record', {NAME => 'data', REPLICATION\_SCOPE => 1}

hbase shell> enable 'record'

对于新表：

hbase shell> create 'record', {NAME => 'data', REPLICATION\_SCOPE => 1}

## 生成collection和配置文件

相关的配置与批量创建索引的配置一致，参考上一个章节。

## 注册一个Indexer

在服务中注册Indexer：

$ hbase-indexer add-indexer \

--name myIndexer \

--indexer-conf $HOME/morphline-hbase-mapper.xml \

--connection-param solr.zk=solr-cloude-zk1,solr-cloude-zk2/solr \

--connection-param solr.collection=hbase-collection1 \

--zookeeper hbase-cluster-zookeeper:2181

检查是否注册成功：

$ hbase-indexer list-indexers

## 在HBase主集群中新增数据，验证结果

1）重启服务：

sudo service hbase-solr-indexer restart

2）添加记录到主集群：

hbase(main):001:0> put 'record', 'row1', 'data', 'value'

hbase(main):002:0> put 'record', 'row2', 'data', 'value2'

3）页面检查索引数据，确认主集群数据已经在从集群中出现并建立了索引。

## 应用场景

实时处理模式适用于有持续数据量，时效性要求很高的情况。不适用于存在删除、修改的数据流。

# 安装包

<http://archive-primary.cloudera.com/cdh5/redhat/6/x86_64/cdh/5.0.4/RPMS/noarch/>

hbase-solr-1.3+cdh5.0.4+49-1.cdh5.0.4.p0.11.el6.noarch.rpm

hbase-solr-doc-1.3+cdh5.0.4+49-1.cdh5.0.4.p0.11.el6.noarch.rpm

hbase-solr-indexer-1.3+cdh5.0.4+49-1.cdh5.0.4.p0.11.el6.noarch.rpm

search-1.0.0+cdh5.0.4+0-1.cdh5.0.4.p0.8.el6.noarch.rpm

solr-4.4.0+cdh5.0.4+198-1.cdh5.0.4.p0.14.el6.noarch.rpm

solr-doc-4.4.0+cdh5.0.4+198-1.cdh5.0.4.p0.14.el6.noarch.rpm

solr-mapreduce-1.0.0+cdh5.0.4+0-1.cdh5.0.4.p0.8.el6.noarch.rpm

solr-server-4.4.0+cdh5.0.4+198-1.cdh5.0.4.p0.14.el6.noarch.rpm

<http://archive-primary.cloudera.com/cdh5/redhat/6/x86_64/cdh/5.0.4/RPMS/x86_64/>

hue-search-3.5.0+cdh5.0.4+387-1.cdh5.0.4.p0.11.el6.x86\_64.rpm