一种基于UDH Search的HBase二级索引构建方案

作者：费英林

目录

[1. 系统概要 2](#_Toc423333340)

[1.1. 背景介绍 2](#_Toc423333341)

[1.2. 系统架构 3](#_Toc423333342)

[1.3. 系统构成 3](#_Toc423333343)

[2. 系统安装与配置 4](#_Toc423333344)

[2.1. HBase Replication集群配置 4](#_Toc423333345)

[2.2. UDH Search服务安装 4](#_Toc423333346)

[3. 索引创建 – 写路径 4](#_Toc423333347)

[3.1. 配置文件 4](#_Toc423333348)

[3.2. 数据流 5](#_Toc423333349)

[4. 索引读取 – 读路径 5](#_Toc423333350)

[4.1. Coprocessor实现 6](#_Toc423333351)

[4.2. 部署 6](#_Toc423333352)

[4.3. 数据流 6](#_Toc423333353)

本文实现了一种HBase二级索引的构建方案，通过该方案，用户可以方便的实现HBase中单列或多列数据的索引创建及查询。为了实现分布式索引的创建和查询，该方案利用CDH Search、HBase主从集群以及索引读取Coprocessor来实现索引的生成与读取。该系统有效的结合了批处理和实时处理两种方式，充分利用了HBase Coprocessor的工作机制，系统本身具有很好的扩展性，可以快速方便的扩展计算能力和存储容量。

# 系统概要

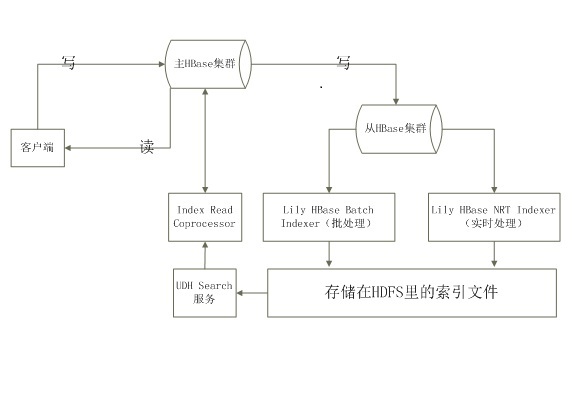
## 背景介绍

HBase是一个列存数据库，每行数据只有一个主键-RowKey，无法依据指定列的数据进行检索。查询时需要通过RowKey进行检索，然后查看指定列的数据是什么，效率低下。在实际应用中，我们经常需要根据指定列进行检索，或者几个列进行组合检索，这就提出了建立HBase二级索引的需求。

目前的HBase二级索引构建方案主要有两种。第一种方案是将索引数据单独存储为一张表，通过HBase Coprocessor生成并访问索引数据。第二种方案是将索引数据与源数据存储在相同的Region里，索引数据定义为一个单独的列族，也是利用Coprocessor来生成并访问索引数据。对于第一种方案，源数据表与索引表的数据一致性很难保证，访问两张不同的表也会增加IO开销和远程调用的次数。对于第二种方案，单表的数据容量会急剧增加，对同一Region里的多个列族进行Split或Merge等操作时可能会造成数据丢失或不一致。

## 系统架构

本方案针对已有HBase二级索引构建方案中的不足，提出了基于UDH Search、HBase主从集群以及HBase Coprocessor的分布式索引构建方案。该方案分离了索引的创建和读取过程，将索引文件与HBase数据文件分开存储，提高了系统的IO性能，充分结合了批处理和实时处理两种流程，解决了数据一致性问题，减少了索引中的冗余数据。基于Hadoop的整体设计也保证了系统的高可用性和高可扩展性。



## 系统构成

1） 主HBase集群：使用UDH部署的Hadoop集群。主集群用于满足高速读写的需求，同时以异步的方式将接收到的数据复制到从集群。

2）从HBase集群：使用UDH部署的Hadoop集群。用于数据备份及索引创建，也可以作为分布式索引服务器。

3）Lily HBase NRT Indexer：是UDH的一个实时索引创建服务，部署在从集群。

4）Lily HBase Batch Indexer：是UDH的一个基于MapReduce计算模型的批量索引创建服务，部署在从集群。

5）HDFS：Indexer生成的索引文件存储在HDFS里，Search服务器读取并缓存这些索引数据。

6）UDH Search服务：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。

7）Index Read Coprocessor：是HBase的插件。

# 系统安装与配置

## HBase Replication集群配置

参见“技术分享-如何配置UDH HBase Replication集群”一文，用UDH分别部署主集群和从集群。在主集群上对数据进行增删改操作，检查从集群上的数据情况，应与主集群数据一致。

## UDH Search服务安装

在从集群上部署UDH Search服务，参见“专题-UDH Search系统构建及其应用”一文。需要配置Lily HBase Batch Indexer和Lily HBase NRT Indexer，分别用于索引的批量创建和实时创建，其中索引的实时创建是利用HBase的Replication机制实现的。

# 索引创建 – 写路径

Search有两种索引创建方式，即批量创建和实时创建。

## 配置文件

配置Solr Collection的配置文件schema.xml和Morphline的相关文件，对RowKey和列族中相关列建立索引。例如，我们希望对列族colfam1的列col1和col2建立索引，相应的配置项应为：

1）Morphline配置文件：

morphlines : [

{

id : morphline1

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

commands : [

{

extractHBaseCells {

mappings : [

{

inputColumn : "colfam1:col1"

outputField : " colfam1\_col1"

type : string

source : value

}

{

inputColumn : "colfam1:col2"

outputField : " colfam1\_col2"

type : string

source : value

}

]

}

}

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

]

}

]

注：RowKey字段默认对应Solr Schema中的id字段，不需要显式配置。

2）Solr配置文件片段：

<field name="colfam1\_col1 " type="text\_general" indexed="true" stored="true" />

<field name="colfam1\_col2 " type="text\_general" indexed="true" stored="true" />

## 数据流

索引生成有两种方式。

1）批处理模式

步骤1：客户端输入数据到主集群

步骤2：主集群复制数据到从集群

步骤3：在某时间点启动Lily HBase Batch Indexer作业，生成索引数据

步骤4：索引数据加载到Search服务器

2）实时模式

步骤1：客户端输入数据到主集群

步骤2：主集群复制数据到从集群

步骤3：从集群向Lily HBase NRT Indexer服务中写入数据，生成索引数据

步骤4：索引数据加载到Search服务器

# 索引读取 – 读路径

读路径只有一种，即通过Coprocessor插件实现索引数据的查询。

## Coprocessor实现

我们需要实现一个Solr索引读取的Coprocessor，它要继承RegionObserver接口。在实现类里加入RowKey解析及Solr索引数据读取的逻辑。

## 部署

通过HBase Shell客户端将我们的Coprocessor部署到对应的HBase表里，重启表。

## 数据流

索引读取流程：

步骤1：客户端构造新的RowKey，将索引列作为RowKey的一部分，比如将colfam1\_col1的值加入新的RowKey

步骤2：客户端以新的RowKey向主集群发起查询

步骤3：主集群调用Coprocessor，根据步骤1的构造规则检查RowKey中是否存在索引列，即检查是否存在colfam1\_col1的数据

步骤4：如果存在索引列colfam1\_col1，调用Solr API，根据colfam1\_col1的值读取相关的原始RowKey，即索引数据中的id字段

步骤5：如果不存在索引列，则认为传入的数据就是原始的RowKey，不需要检索Solr数据

步骤6：依据4或5的处理结果执行查询