云数据库 HBase 版

HBase Solr 全文引擎

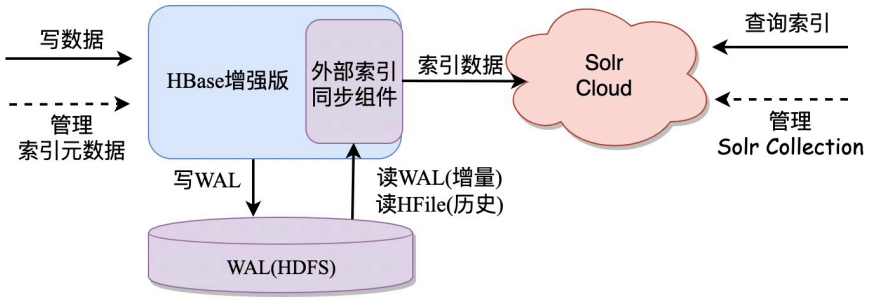
HBase Solr 全文引擎

服务介绍

Solr是分布式全文检索的最佳实践之一。Solr支持各种复杂的条件查询和全文索引。通过结合HBase、Solr，可以最大限度发挥HBase和Solr各自的优点，从而使得我们可以构建复杂的大数据存储和检索服务。
HBase+Solr适用于：**需要保存大数据量数据，而查询条件的字段数据仅占原数据的一小部分，并且需要各种条件组合查询的业务。**例如：

- 常见物流业务场景，需要存储大量轨迹物流信息，并需根据多个字段任意组合查询条件
- 交通监控业务场景，保存大量过车记录，同时会根据车辆信息任意条件组合检索出感兴趣的记录
- 各种网站会员、商品信息检索场景，一般保存大量的商品/会员信息，并需要根据少量条件进行复杂且任意的查询，以满足网站用户任意搜索需求等。

云HBase增强版提供的全文索引服务深度融合了HBase和Solr，**能够自动将用户写入HBase的数据实时数据写入到Solr中，用户无需双写HBase和Solr。**



在使用过程中，用户全程只需要和HBase和Solr交互即可，HBase增强版原生内置了高性能索引同步组件，该组件完全对用户透明，用户只需要通过HBase增强版对外提供的建立、修改外部索引的接口操作，即可完成索引元数据的管理。

详细使用请参考增强版全文索引服务
注：目前只有HBase增强版支持全文索引服务，标准版不支持全文索引服务。

访问Solr WebUI

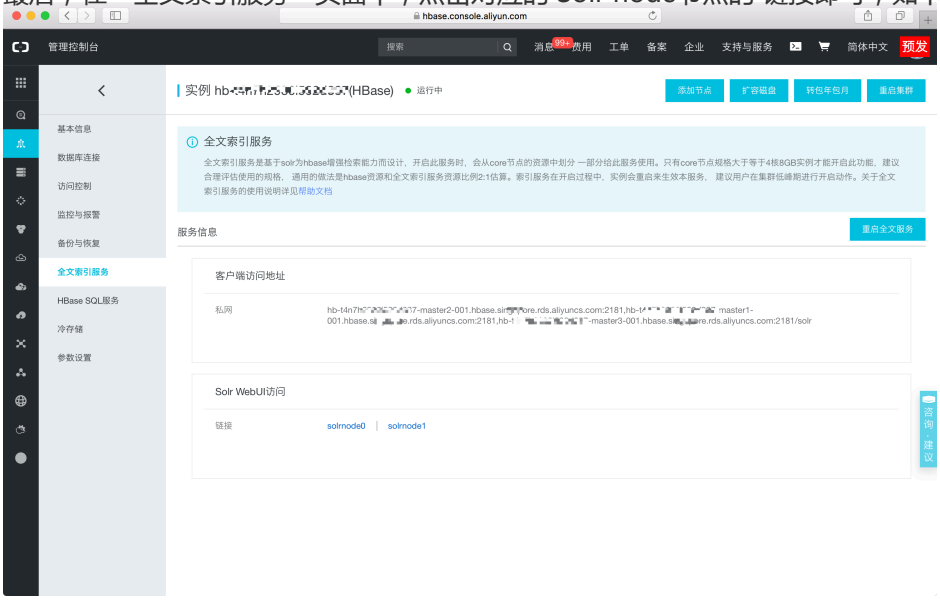
Solr WebUI访问

本文简述Solr WebUI访问方法，其大体与HBase WebUI访问流程一样。下面再进行简单的描述。

访问Solr WebUI前提

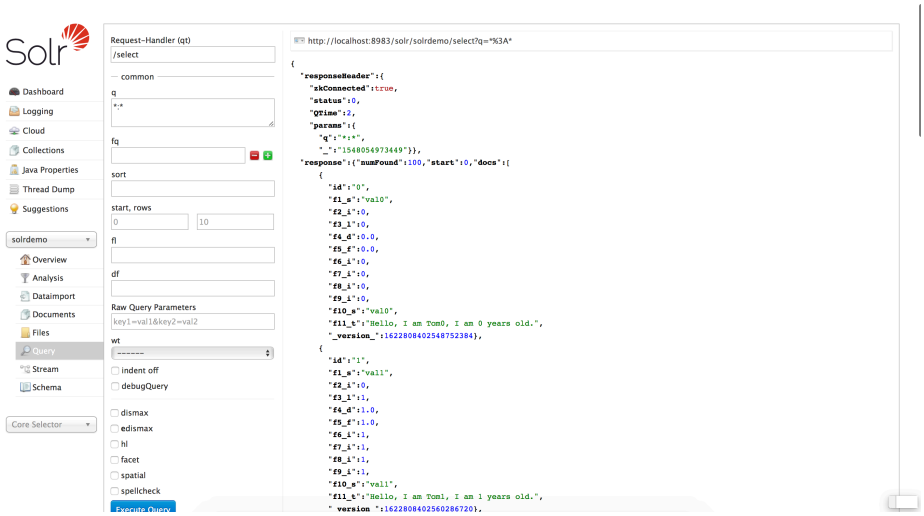
- Solr WebUI访问的用户，和HBase WebUI访问一样，需要设置用户本机的ip到HBase实例白名单中，详见参考
- 第一次访问时，需要设置好WebUI访问账户密码，详见参考

最后，在“全文索引服务”页面中，点击对应的 Solr node节点的 链接即可，如下图：



Solr WebUI界面功能简单介绍

SolrCloud是由多个Solr Server通过 Zookeeper协作起来，每个Solr Server都是一个单独的个体，每个Solr实



例的UI界面如下：
从左边栏可以看出，主要关注的基本功能如下：

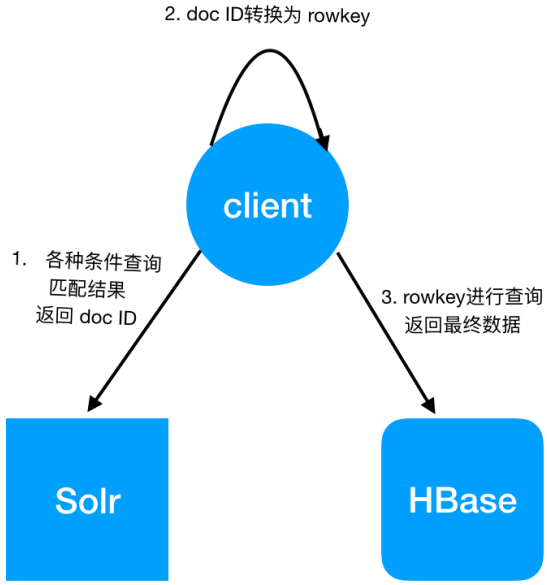
- Dashboard 本Solr node节点概况
- Logging 日志查看及日志级别设置
- Cloud 查看各个 collection的shard/replica 分布与状态概况
- Collections 查看各个collection基本shards属性
- Thread Dump 显示本次访问时，jvm的thread dump快照
- Collection下拉框，支持对collection进行简单的查询和插入操作，以及schema查看
- replica下拉框，查看具体的replica的简单统计情况

更多关于Solr界面的介绍与基础使用，请参考链接 Overview-Solr-Admin-UI

索引查询示例

索引查询示例

本文介绍如何进行各种条件查询索引，返回匹配结果id后，转换为rowkey查询HBase取出最终完整原数据。流



程如下：

客户端准备

本示例展示使用Java客户端SolrJ来操作solr，并使用Java的HBase API访问HBase。

Java项目工程添加依赖如下：

```
<dependency>
<groupId>org.apache.solr</groupId>
<artifactId>solr-solrj</artifactId>
<version>7.3.1</version>
</dependency>
```

各种条件查询，获取doc ID

```
String zkHost = "zk1:2181,zk2:2181,zk3:2181/solr"
CloudSolrClient cloudSolrClient = new CloudSolrClient.Builder(Collections.singletonList(zkHost),
Optional.empty()).build();
SolrQuery solrQuery = new SolrQuery("f1_s:val99");
QueryResponse response = client.query(collection, solrQuery);
SolrDocumentList documentList = response.getResults();
for(SolrDocument doc : documentList){
String id = (String)doc.getFieldValue("id");
//do something
}
```

更多精确、模糊、范围、facet、stats、and/or组合等查询，见 github demo地址

doc ID转换成rowkey

1. string类型doc id如果默认 index_conf.xml配置中，不指定unique-key-formatter，或者显式指定使用string，如：

```
<indexer table="solrdemo" unique-key-formatter="string">
...
</indexer>
```

那么，doc id转成rowkey过程如下：

```
// String id = "xxxx";
org.apache.hadoop.hbase.util.Bytes.toBytes(docId)
```

2. hex类型doc id如果默认 index_conf.xml配置中，unique-key-formatter指定使用hex，如：

```
<indexer table="solrdemo" unique-key-formatter="hex">
...
</indexer>
```

那么，doc id转成rowkey过程如下：

```
// String id = "xxxx";
org.apache.commons.codec.binary.Hex.decodeHex(id.toCharArray());
```

此过程借助 commons-codec-1.9.jar的方法转换。此jar包依赖如下：

```
<dependency>
<groupId>commons-codec</groupId>
<artifactId>commons-codec</artifactId>
<version>1.9</version>
</dependency>
```

获取最终数据

最终我们拿根据各种条件查询匹配到的rowkey，如需获取这个rowkey的完整数据, 只要进行HBase的 get操作即可。HBase的查询访问支持Java api原生方式 ，详见参考; 也可以通过thrift支持c#、python、go等多语言, 详见参考

分词使用说明

分词使用说明

全文索引服务涵盖丰富的查询功能，facet、排序/分页、任意复杂条件组合查询、function查询、stats统计等，其中还有一项重要的功能，就是分词。如我们常见的视频标题关键字搜索、商品标题关键字搜索等，都是利用了全文引擎的分词功能。
本文重点介绍两个类型分词器，分别是默认的英文分词、ik中文分词。

英文分词器

默认schema中定义了 text_general字段类型，此字段类型的分词器配置如下：

```
<!-- A general text field that has reasonable, generic
cross-language defaults: it tokenizes with StandardTokenizer,
removes stop words from case-insensitive "stopwords.txt"
(empty by default), and down cases. At query time only, it
also applies synonyms.
-->
<fieldType name="text_general" class="solr.TextField" positionIncrementGap="100" multiValued="true">
<analyzer type="index">
<tokenizer class="solr.StandardTokenizerFactory"/>
<filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt" />
<!-- in this example, we will only use synonyms at query time
<filter class="solr.SynonymGraphFilterFactory" synonyms="index_synonyms.txt" ignoreCase="true"
expand="false"/>
<filter class="solr.FlattenGraphFilterFactory"/>
-->
<filter class="solr.LowerCaseFilterFactory"/>
</analyzer>
<analyzer type="query">
<tokenizer class="solr.StandardTokenizerFactory"/>
<filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt" />
<filter class="solr.SynonymGraphFilterFactory" synonyms="synonyms.txt" ignoreCase="true" expand="true"/>
<filter class="solr.LowerCaseFilterFactory"/>
</analyzer>
</fieldType>
```

配置的分词组件功能如：

- StandardTokenizerFactory 标准切词器，负责切分分词汇
- StopFilterFactory 停用词过滤器，过滤类似 “the”、 “are”这种停用词
- SynonymGraphFilterFactory 近义词过滤器
- FlattenGraphFilterFactory 配合上述近义词过滤器完成近义词的过滤替换处理
- LowerCaseFilterFactory 小写过滤器

例如有句子description为： “A contented mind is the greatest blessing a man can enjoy in this world”
按照分词的短语查询如下：

```
SolrQuery solrQuery = new SolrQuery("description:\"greatest blessing\"");
QueryResponse response = client.query(collection, solrQuery);
```

就可以匹配到这个句子。另外属于分词的其他查询方式如：按某词汇term查询、短语查询、近似查询，详见github demo例子
要使用这个英文的分词，只要在schema中设置字段类型为“text_general”即可。如需定制分词，可以进一步了解 Solr的analyzers章节

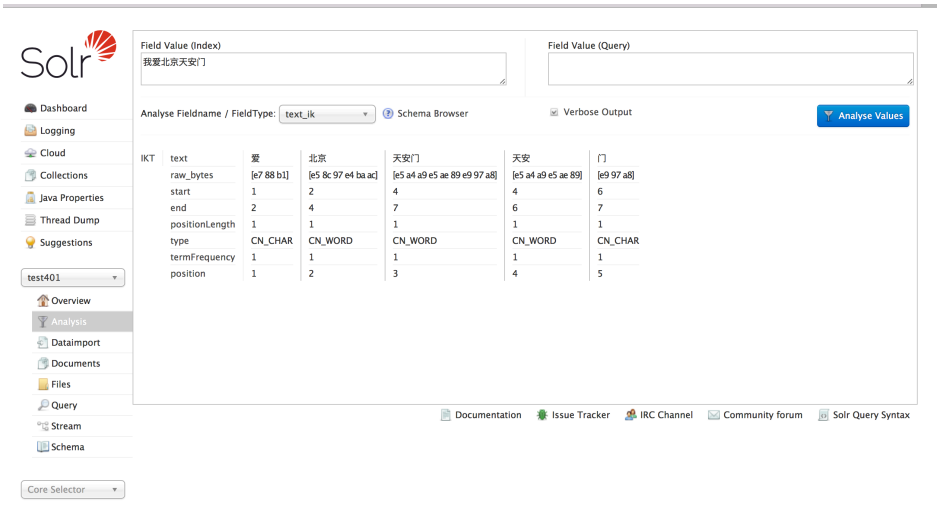
中文分词器

中文分词器有很多，这里介绍一款开源的ik分词器，官方介绍地址
使用配置参考 solr-7.3.1-ali-1.0/server/solr/configsets/_democonfig/conf/managed-schema 配置文件。主要配置如下：

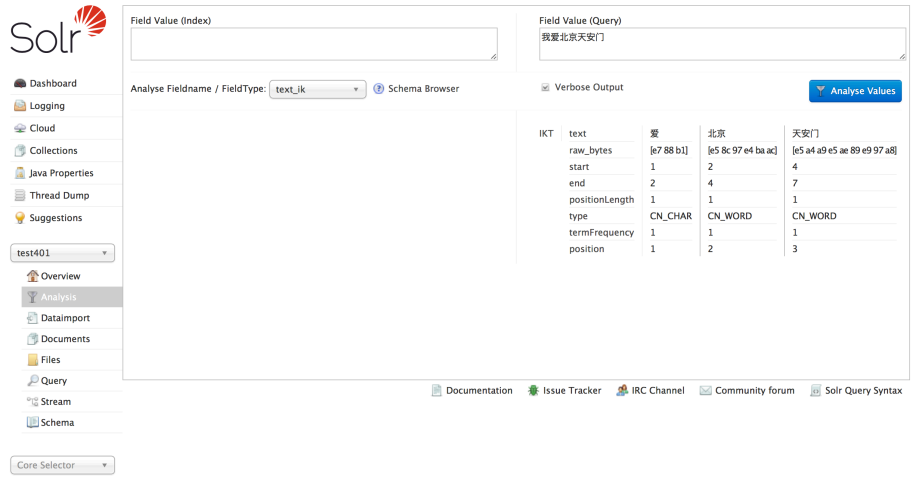
```
<fieldType name="text_ik" class="solr.TextField">
<analyzer type="index">
<tokenizer class="org.wltea.analyzer.lucene.IKTokenizerFactory" useSmart="false" />
</analyzer>
<analyzer type="query">
<tokenizer class="org.wltea.analyzer.lucene.IKTokenizerFactory" useSmart="true" />
</analyzer>
</fieldType>
```

配置这个类型后，只要定义字段的类型为 “text_ik”,那么它就可以默认按照中文分词了。useSmart表示是否开启智能分词，智能分词会根据分词语法分词后，根据一些规则进一步挑选更合理的词汇，例如最长的完整词汇等规则。我们可以在Solr WebUI里尝试分词效果，如下：

index阶段 useSmart=false，效果如下：



query阶段 useSmart=true，效果如下：



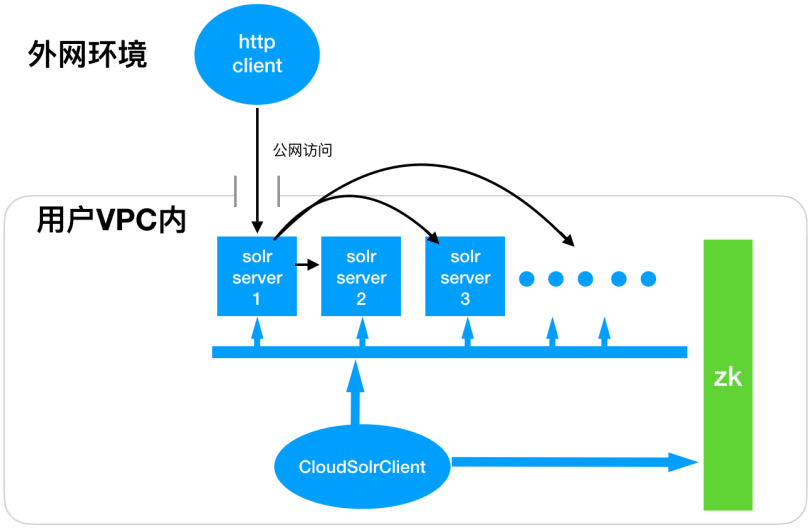
中文分词的词库扩展

如需扩展ik的中午字典库，请联系“云HBase答疑”客服。
如果还有自定义的分词器，也可以联系“云HBase答疑”客服 咨询如何集成。

Solr公网访问

Solr公网访问

SolrCloud模式访问如同HBase一样，使用zookeeer进行进行获取可用的Solr Server进行访问。不同的是，Solr的访问不是在客户端路由，CloudSolrClient也只是封装load balance的http client循环访问不通的Solr Server进行负载均衡。真正发生路由是在一个Solr Server上，针对这点，就表明我们所有的访问都可以发送到一个Solr Server上。
另外，Solr使用http的方式访问，有大量的开源工具、代码库可以使用，当然也可以使用官网的各种语言的客户端。为了保持所有客户端都是和开源同步，为了公网访问Solr集群，我们只需要访问固定某个开好公网的Server进行访问即可，这样既可以满足开发测试阶段需求，也可以使用各种开源的http访问工具进行开发测试。



大致流程如下：

公网开放

Solr的公网开放和HBase的公网开放一样，当用户申请公网访问时，在“数据库链接”中显示如下：

申请完成后，可以看到，公网zk地址有3个，我们获取最后一个master3中缀的链接节点，即为我们可以进行公网访问的链接。如：

```
hb-proxy-pub-xxxxxxxxxxx-master3-001.hbase.singapore.rds.aliyuncs.com
```

进行公网访问

拿到了上面的master3的solr server公网地址后，可以使用这个链接进行访问solr。下面介绍两种访问方式时，设置的链接地址如何设置。

curl方式

node的hostname为上面拿到的公网master3地址，端口使用solr专用的8983

```
curl "http://hb-proxy-pub-xxxxxxxxxxx-master3-
001.hbase.singapore.rds.aliyuncs.com:8983/solr/solrdemo/query?q=*:*"
```

注：如果上述curl无法访问，请确认是否白名单设置完成，详见hbase公网访问中的白名单设置
如果还不能访问，确认是否在开solr之前已经开过公网，如果是，可以尝试关闭公网再开启公网后验证一下。

SolrJ代码方式

代码构造SolrClient的时候，使用HttpSolrClient即可，如下：

```
HttpSolrClient solrClient = new HttpSolrClient.Builder("http://hb-proxy-pub-xxxxxxxxxxx-master3-
001.hbase.singapore.rds.aliyuncs.com:8983/solr").build();
SolrQuery solrQuery = new SolrQuery("*:*");
QueryResponse response = solrClient.query("solrdemo", solrQuery);
//do something
```

注：开发测试使用如上方式进行公网访问，当上生产环境时，请使用CloudSolrClient 客户端进行构建应用。

常见FAQ

常见FAQ

创建solr collection的shard个数、replica个数设置多少合适？

我们先列举一下几条规则，尽量满足如下规则即可：
1）单个shard的最大document条数不能超过 int的最大值，大概21亿。否则就会因lucene底层循环复用int值而导致覆盖。
2）对于replicationFactor副本数，我们推荐默认设置为1，并且把autoAddReplicas属性设置为false。对于有特殊要求的，比如写入非常少，读非常多，且读可能会有大量热点，那可以考虑使用replicationFactor=2、3这种，可以缓解读负载均衡。
3) 我们假设实例有n个solr server节点，每个节点放m个shard。我们得遵循如下公式：

```
这个collection未来的总数据量 > n X m X 21亿
```

对于一个collection而言，每个solr server节点放一个shard开始，即m=1，如果不满足，我们再m = 2、3...直到能满足“未来collection的总数据量 > n X m X 21亿” 这个公式为止即可。
如果你发现一个server要放的m的个数太大了，比如超过物理linux机器cpu的个数了，那就可能要考虑扩容节点了。推荐一个solr server节点尽量不要太多相同collection的shard
4）对于单个shard在条数不能超过 int的最大值，大概21亿的情况下，它的存储也尽量不能太大，如果比如一个shard保存了20亿，按照1k一个doc，总数据量达到2T左右, 这对一个server来说可能会有点大了，对应如果大量扫描估计扛不住，推荐扩容节点，分担大量存储扫描取数据的压力。控制在单个shard的总量数据也不要太大。当然这个要结合查询特点和数据特点，比如单个document就10k和200字节碰到这种情况都是不一样的。
注：对于solr的shard、replica分配，是可以后期再调整和split的，按照上述设置之后，后续有变化再调整也可

以。另外，我们推荐用户后续如果需要调整shard、replica个数的，请联系“云hbase答疑”进行沟通，在solr在solr 7.3.1.4版本之前，请勿自行split shard。

hbase同步数据到solr索引的延时是多少？多少秒可见？

```
索引同步的延时时间 = 数据同步延迟 + solr commitWithin时间
```

没有堆积情况下, 同步延时主要为框架开销，毫秒级别(如果有积压情况下，延时会变长，需要增加节点来增加同步能力)
对于solr 的commitWithin,默认是15000ms。再写入压力不大，没有积压的情况下，几乎主要决定索引的可见性即为 commitWithin时间，对于写很少的客户可以设置为1秒、3秒、5秒，对于写入量大的用户，不推荐设置过小，不然小文件会比较多，进而系统会频繁merge，也会侧面影响整体性能。

Solr如何使用预排序功能？

我们都知道排序是非常消耗资源的，在数据量特别大的时候，不仅查的慢，还特别占用系统资源，如果本身存储的数据已经按照某个字段预先排好序，那么solr的检索会有明显的提升，特别是在大数据量上对比的时候，此特点效果更明显。那么在solr层面是怎么支持的呢？下面我们简单描述下步骤：

- 修改solrconfig.xml中的MergePolicy, 详见链接
- 查询时，指定参数segmentTerminateEarly=true即可
  下面简单给个demo演示：

```
<mergePolicyFactory class="org.apache.solr.index.SortingMergePolicyFactory">
<str name="sort">timestamp desc</str>
<str name="wrapped.prefix">inner</str>
<str name="inner.class">org.apache.solr.index.TieredMergePolicyFactory</str>
<int name="inner.maxMergeAtOnce">10</int>
<int name="inner.segmentsPerTier">10</int>
</mergePolicyFactory>
```

此时，我们主要关心“< str name=” sort” >timestamp desc< /str>”配置，此时插入数据将会按照timestamp字段进行预先倒序排序，执行查询如下：

```
curl
"http://localhost:8983/solr/testcollection/query?q=*:*&sort=timestamp+desc&rows=10&segmentTerminateEarly=true"
```

参数上加上“segmentTerminateEarly=true”后，显示效果会比没有设置预排序的快很多，尤其是排序数据量T级别之后，效果更明显。

需要注意的是:

- 查询时，指定的sort必须与配置的MergePolicy中指定的一致，否则不起效果
- 查询时需要指定segmentTerminateEarly参数，否则也会进行全排
- 使用了这个预排序返回的结果中，“numFound”是不准确的

关于solr的各种客户端连接使用说明

在云hbase 全文服务solr的使用中，我们一共提供了几个地址：zk内网地址、和zk公网地址、solr的webui公网地址、solr的CloudSolrClient连接地址。下面分别描述这几种地址的使用：

- solr CloudSolrClient内网链接地址

见上图，作为java api 中 CloudSolrClient的内网访问地址，此时CloudSolrClient的构造方法如下：

```
List<String> zkHostList = new ArrayList<>();
zkHostList.add("
hb-m5eXXXX-master1-001.hbase.rds.aliyuncs.com:2181");
zkHostList.add("
hb-m5eXXXX-master2-001.hbase.rds.aliyuncs.com:2181");
zkHostList.add("
hb-m5eXXXX-master3-001.hbase.rds.aliyuncs.com:2181");
CloudSolrClient cloudSolrClient = new Builder(zkHostList, Optional.of("/solr");
```

注意，替换“hb-m5eXXXX.....” 为你的对应zk 单个host地址。

- solr webui公网地址，见如下图

如上图，直接点击即可打开solr WebUI，其访问控制和hbase WebUI一样，详见文档链接
注意这个仅为公网浏览器打开的solr admin WebUI访问地址，不能用其浏览器显示的url进行solr数据访问。

- solr HttpSolrClient、curl 公网单点开发测试访问地址

如上图，当用户在“数据库链接”中，也打开 zk公网时，同样solr也具备了公网开发测试的单点访问功能，其开通的节点为 关键字带有“master3-1”的 host名字。如这里为的公网单点访问地址为：

```
http://hb-proxy-pub-m5eXXXX-master3-001.hbase.rds.aliyuncs.com:8983
```

此时，如果想公网访问solr进行开发测试，命令行运维是可以直接类似如下即可：

```
curl "http://hb-proxy-pub-m5eXXXX-master3-
001.hbase.rds.aliyuncs.com:8983/solr/admin/collections?action=list"
```

如果是想公网通过java api的HttpSolrClient单点公网开发测试访问solr，可以初始化实例如下：

```
HttpSolrClient solrClient = new HttpSolrClient.Builder("http://hb-proxy-pub-m5eXXXX-master3-
001.hbase.rds.aliyuncs.com:8983/solr").build();
```

注意，solr的路由核心是在server端的，所以客户端访问任何一个节点都可以访问整个集群，当然如果您是内网访问solr集群，推荐使用CloudSolrClient api，它会有一些负责均衡相关的功能，这里的HttpSolrClient仅推荐用来做开发测试使用。