# Solr分享

1、            Solr 简介

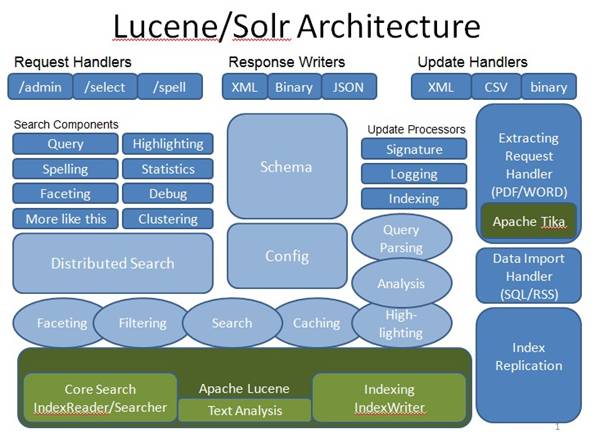
1.1  solr与lucene

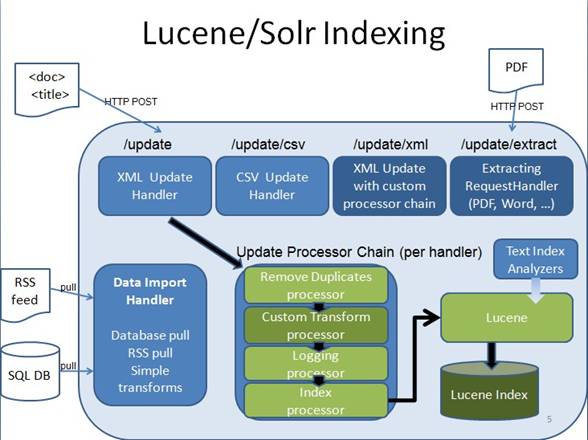
²  Solr是一个基于Lucene java库的企业级搜索服务器，包含XML/HTTP，JSON API, 高亮查询结果，faceted search，缓存，复制还有一个WEB管理界面。Solr运行在Servlet容器中。

²Solr和Lucene的本质区别有以下三点：搜索服务器，企业级和管理。Lucene本质上是搜索库，不是独立的应用程序，而Solr是。Lucene专注于搜索底层的建设，而Solr专注于企业应用。Lucene不负责支撑搜索服务所必须的管理，而Solr负责。

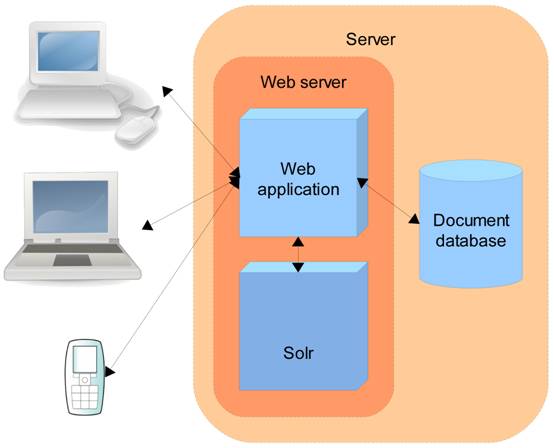
² 一句话概括Solr: Solr是Lucene面向企业搜索应用的扩展。

1.2  架构





1.3 最简版用法



展现在我们眼前的solr是： 一个war包 + 各种jar包 + 配置

1、Define a  the schema tells Solr about the contents of documents it will be indexing.定义schema.xml。

2、Deploy Solr to your application server.布署solr服务

3、Feed Solr the document for which your users will search. 建索引

4、Expose search functionality in your application. 查询

注：服务的形式有三种Embedded(直接调用，无传输问题，不能对外服务)、standalone、cluster。

以上指的是standalone的形式。

2、            Solr 配置

2.1 schema.xml

一、field

*<!-- id-->*

   <field name="id"type="string"indexed="true"stored="true"required="true"multiValued="false" />

*<!--搜索字段由title ,tags组成-->*

   <field name ="queryText"type="text\_ik"indexed="true"stored="false"multiValued="true"/>

*<!--标题-->*

   <field name="title"type="text\_ik"indexed="true"stored="true" />

*<!--商品标签-->*

   <field name="tags"type="text\_ik"indexed="true"stored="true" />

二、CopyFiled

   <copyField source="title"dest="queryText"/>

   <copyField source="tags"dest="queryText"/>

三、FieldType 部份常用数据类型

   <fieldType name="tint"class="solr.TrieIntField"precisionStep="8"positionIncrementGap="0"/>

    <fieldType name="tfloat"class="solr.TrieFloatField"precisionStep="8"positionIncrementGap="0"/>

    <fieldType name="tlong"class="solr.TrieLongField"precisionStep="8"positionIncrementGap="0"/>

    <fieldType name="tdouble"class="solr.TrieDoubleField"precisionStep="8"positionIncrementGap="0"/>

    <fieldType name="date"class="solr.TrieDateField"precisionStep="0"positionIncrementGap="0"/>

    <fieldType name="pint"class="solr.IntField"/>

    <fieldType name="plong"class="solr.LongField"/>

    <fieldType name="pfloat"class="solr.FloatField"/>

    <fieldType name="pdouble"class="solr.DoubleField"/>

<fieldType name="pdate"class="solr.DateField"sortMissingLast="true"/>

<!--兼容用-->

   <fieldType name="pint"class="solr.IntField"/>

    <fieldType name="plong"class="solr.LongField"/>

    <fieldType name="pfloat"class="solr.FloatField"/>

    <fieldType name="pdouble"class="solr.DoubleField"/>

 <fieldType name="pdate"class="solr.DateField"sortMissingLast="true"/>

TrieField介绍：[http://www.kuqin.com/appserver/20120505/320366.html#source](http://www.kuqin.com/appserver/20120505/320366.html" \l "source)

其中用了做加法的方式，归并出结果。个人观点：大多对collection的处理，作加法优于作减法。作减法意味着再次匹配与查找。

positionIncrementGap的作用：对multivalue里头多个值域进行搜索时。多个value之间我认为应该是保持相互独立的，但是在实际搜索中感觉solr把所有的值域都串在一起，当作一个长value来处理，没有达到我想要的效果。例子：

*Suppose a document has a multi-valued "author" field.   Like this:*

*author: John Doe*

*author: Bob Smith*

*With a position increment gap of 0, a phrase query of "doe bob" would*

*be a match.  But often it is undesirable for that kind of match across*

*different field values.  A position increment gap controls the virtual*

*space between the last token of one field instance and the first token*

*of the next instance.  With a gap of 100, this prevents phrase queries*

*(even with a modest slop factor) from matching across instances.*

四、FieldType 指明analyzer

<fieldType name="text\_ik"class="solr.TextField">

        <analyzer type="index"class="org.wltea.analyzer.lucene.IKAnalyzer" />

        <analyzer type="query"class="org.wltea.analyzer.lucene.IKAnalyzerUseSmart" />

      </fieldType>

*<!-- A text field that only splits on whitespace for exact matching of words -->*

    <fieldType name="text\_ws"class="solr.TextField"positionIncrementGap="100">

      <analyzer>

        <tokenizer class="solr.WhitespaceTokenizerFactory"/>

      </analyzer>

    </fieldType>

*<!-- 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">

      <analyzer type="index">

        <tokenizer class="solr.StandardTokenizerFactory"/>

        <filter class="solr.StopFilterFactory"ignoreCase="true"words="stopwords.txt" />

        <filter class="solr.LowerCaseFilterFactory"/>

      </analyzer>

      <analyzer type="query">

        <tokenizer class="solr.StandardTokenizerFactory"/>

        <filter class="solr.StopFilterFactory"ignoreCase="true"words="stopwords.txt" />

        <filterclass="solr.SynonymFilterFactory"synonyms="synonyms.txt"ignoreCase="true"expand="true"/>

        <filter class="solr.LowerCaseFilterFactory"/>

      </analyzer>

    </fieldType>

五、UniqueKey

<uniqueKey>id</uniqueKey>

uniqueKey 元素用来表示一个文档的唯一值，类似数据库ID。尽管uniqueKey不是必须，但在程序设计过程中还是建议使用。当做索引时，如果id相同，会直接update。

六、指定评分器

<similarity class="org.apache.lucene.search.similarities.DefaultSimilarity"/>

总结一下，修改评分的方式，待更深入了解后，具体再介绍。

1、   Similarity：自定义评分器，用于索引阶段

2、  setBoost：即可在索引的时候设置boost，又可以在查询的时候设置

3、  Dismax/eDisMax：两种queryParse，可以查询时设置，影响得分。

4、Evaluate.xml，对指定query修改结果，影响的是结果，不是评分。

2.2 solrconfig.xml

²  request handlers

²  listeners (processes that "listen" for particular query-related events; listeners can be used to trigger the execution of special code, such as invoking some common queries to warm-up caches)

²  the Request Dispatcher for managing HTTP communications

²  the Admin Web interface

² parameters related to replication and duplication

3、            Data Import Handler

 我们不采用。1、与solr的耦合太强，2、不支持mongodb等其他数据源，3、不适用于复杂数据源的情况，例如Mysql与Mongodb的组合。

4、            Solrj

4.1 solrj Index

我们利用solrj编写index服务，不用DataImportHandler.

****privatefinalstatic**** String *URL* = "http://localhost:8983/solr";

****private**** HttpSolrServer server = ****null****;

    @Before

****publicvoid**** init() {

           server = ****new**** HttpSolrServer(*URL*);

    }

****publicvoid**** index() {

****try**** {

           SolrInputDocument doc = ****new**** SolrInputDocument();

           doc.addField("id","1");

           doc.addField("msg\_title", "这是我的第一个solrj的程序");

           doc.addField("msg\_content","我的solrj的程序究竟能不能跑得起来呢？");

           Date date = ****new**** Date();

           doc.addField("tdate\_field",date);

           doc.addField("date\_field",date);

           server.add(doc);

           server.commit();

       } ****catch**** (MalformedURLException e) {

           e.printStackTrace();

       } ****catch**** (SolrServerException e) {

           e.printStackTrace();

       } ****catch**** (IOException e) {

           e.printStackTrace();

       }

    }

4.2 solrj Search

****publicvoid**** search() {

****try**** {

           SolrQuery query = ****new**** SolrQuery("\*");

           QueryResponse resp = server.query(query);

           //查询出来的结果都保存在SolrDocumentList中

           SolrDocumentList sdl = resp.getResults();

           System.*out*.println(sdl.getNumFound());

****for****(SolrDocument sd:sdl) {

                  String id = (String)sd.getFieldValue("id");              System.*out*.println((Date)sd.getFieldValue("date\_field"));

                          System.*out*.println((Date)sd.getFieldValue("tdate\_field"));

           }

       } ****catch**** (SolrServerException e) {

           e.printStackTrace();

       }

    }

4.3 solr常用查询参数

强调一点，solr是基于http的，restful。

²  q - 查询字符串，必须的。

²  fl - 指定返回那些字段内容，用逗号或空格分隔多个。

²  start - 返回第一条记录在完整找到结果中的偏移位置，0开始，一般分页用。

²  rows - 指定返回结果最多有多少条记录，配合start来实现分页。

²  sort - 排序，格式：sort=<field name>+<desc|asc>[,<field name>+<desc|asc>]… 。示例：（inStock desc, price asc）表示先 “inStock” 降序, 再 “price” 升序，默认是相关性降序。

²  wt - (writer type)指定输出格式，可以有 xml, json, php, phps, 后面 solr 1.3增加的，要用通知我们，因为默认没有打开。

²  fq - （filter query）过虑查询，作用：在q查询符合结果中同时是fq查询符合的，例如：q=mm&fq=date\_time:[20081001 TO 20091031]，找关键字mm，并且date\_time是20081001到20091031之间的。

5、            Solr Cluster

5.1 利用ssh/rsync

Solr 1.2引入。（过时）

SSh/rsync 可手动同步或定时任务同步。  
无法实现实时同步。需要额外配置文件。

5.2  Master/Slave

5.2.1 Features 特性

<!--[if !supportLists]-->²  <!--[endif]-->  replication无需额外的配置脚本

<!--[if !supportLists]-->²  <!--[endif]-->  仅须在solrconifg.xml配置

<!--[if !supportLists]-->²  <!--[endif]-->  同时replicate配置文件

<!--[if !supportLists]-->²  <!--[endif]-->  相同配置，跨平台使用

<!--[if !supportLists]-->²  <!--[endif]-->  不依赖OS

<!--[if !supportLists]-->²  <!--[endif]-->  与solr紧密结全，管理界面提供精细的控制

5.2.2 角色

Master

Slave

Repeater

如果slave从远方的数据中心下载索引，就可能导致下载消耗过多的带宽。为了避免类似的性能降级情况，你可以将一个或多个slave配置为repeater。简单地说，一个repeater即是一个slave也是一个master。

<requestHandler name="/replication"class="solr.ReplicationHandler" >

       <lst name="master">

           <str name="enable">${enable.master:false}</str>

         <str name="replicateAfter">commit</str>

         <str name="replicateAfter">startup</str>

*<!--<str name="backupAfter">optimise</str>-->*

*<!--If configuration files need to be replicated give the names here, separated by comma -->*

         <str name="confFiles">${CONF\_FILES}</str>

*<!--The default value of reservation is 10 secs.See the documentation below . Normally , you should not need to specify this -->*

          <str name="commitReserveDuration">00:00:10</str>

         </lst>

       <lst name="slave">

           <str name="enable">${enable.slave:false}</str>

           <str name="masterUrl">${MASTER\_CORE\_URL}</str>

            <str name="pollInterval">${POLL\_TIME}</str>

         </lst>

  </requestHandler>

How does it work? 工作原理

该功能是依赖于Lucene的IndexDeletionPolicy特性实现的。通过该API,Lucene暴露了IndexCommits作为每次 commit/optimize的回调。 一次[IndexCommit](http://hudson.zones.apache.org/hudson/job/Lucene-trunk/javadoc/core/org/apache/lucene/index/IndexCommit.html) 调用暴露了每次commit相关的文件。这便使得我们得以确认哪些文件是需要被复制的。

按照solr传统，所有的操作都是通过Restful API完成的。

## How does the slave replicate?slave如何复制

Master是无法感知slaves的，salve会持续轮循master(依据'pollInterval参数')检测当前mater的索引版本。如果slave发现master上有更新版本的索引，便发起repliaction过程。

·         Slave发起filelis命令，获得文件列表。该命令返回文件名的同时还会返回metadate(size,lastmodifyed,alias)。

·         Slave会检查本地的索引。接着会下载缺少的文件（命令名为“filecontent”）。这会使用一种定制的格式（类似HTTP chunked encoding）每个文件的全部或部份内容。如果连接断开，下载会在上一次失败的点继续。在任意点，会尝式5次，如果都失败，就会放弃整个replication.

·         文件会被下载至一个临时文件夹。所以如果slave或者master在replication过程中宕机了，不会造成任何损失，只是停止了当前的replication。

·         当下载完成后，所有的新文件被移至slave的活动索引目录，而且文件的时间戳会与master上的一致。

·         Slave的 ReplicationHandler 会发起‘commit’命令，新的索引被加载。

## How are configuration files replicated?配置文件如何被复制

·         需要复制的文件被必须被'confFiles'参数显式指明。

·         只有conf文件夹下的solr实例会被复制。

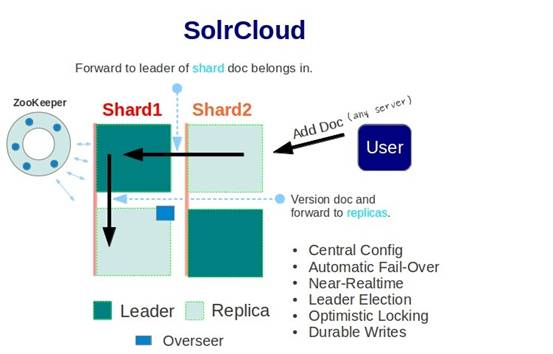
·        配置文件只会跟随索引被复制。意味着即使master的一个文件被改变了，只有等到master上一次新的commit/optimize，这些文件才会被复制。

·         不同于索引文件，配置文件没有时间戳可用，它们会通过校验和被比较。如果 master和slave上的schema.xml的校验和相同，则会被视为相同文件。

·         配置文件也会在被移至目的文件夹前，被先下载到临时文件，老文件会被重命令，ReplicationHandler 不会自动清理配置文件。

·         如果replication包含了最新版本的conf文件，对应的solr核会被重新加载，而不是发起一条"commit"命令

6、            Solr Cloud



## 概念

·         ****Collection****：在SolrCloud集群中逻辑意义上的完整的索引。它常常被划分为一个或多个Shard，它们使用相同的Config Set。如果Shard数超过一个，它就是分布式索引，SolrCloud让你通过Collection名称引用它，而不需要关心分布式检索时需要使用的和Shard相关参数。

·         ****Config Set****: Solr Core提供服务必须的一组配置文件。每个config set有一个名字。最小需要包括solrconfig.xml (SolrConfigXml)和schema.xml (SchemaXml)，除此之外，依据这两个文件的配置内容，可能还需要包含其它文件。它存储在Zookeeper中。Config sets可以重新上传或者使用upconfig命令更新，使用Solr的启动参数bootstrap\_confdir指定可以初始化或更新它。

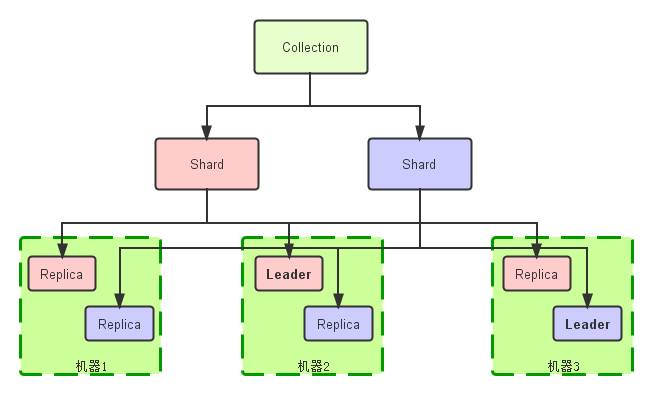
·         ****Core****: 也就是Solr Core，一个Solr中包含一个或者多个Solr Core，每个Solr Core可以独立提供索引和查询功能，每个Solr Core对应一个索引或者Collection的Shard，Solr Core的提出是为了增加管理灵活性和共用资源。在SolrCloud中有个不同点是它使用的配置是在Zookeeper中的，传统的Solr core的配置文件是在磁盘上的配置目录中。

·        ****Leader****: 赢得选举的Shard replicas。每个Shard有多个Replicas，这几个Replicas需要选举来确定一个Leader。选举可以发生在任何时间，但是通常他们仅在某个Solr实例发生故障时才会触发。当索引documents时，SolrCloud会传递它们到此Shard对应的leader，leader再分发它们到全部Shard的replicas。

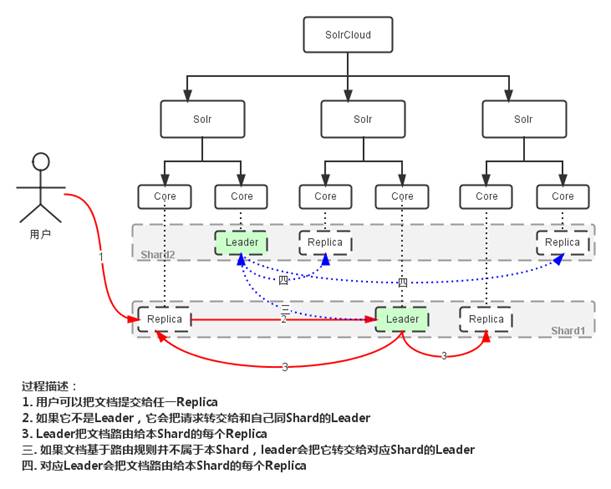
·         ****Replica****: Shard的一个拷贝。每个Replica存在于Solr的一个Core中。一个命名为“test”的collection以numShards=1创建，并且指定replicationFactor设置为2，这会产生2个replicas，也就是对应会有2个Core，每个在不同的机器或者Solr实例。一个会被命名为test\_shard1\_replica1，另一个命名为test\_shard1\_replica2。它们中的一个会被选举为Leader。

·         ****Shard****: Collection的逻辑分片。每个Shard被化成一个或者多个replicas，通过选举确定哪个是Leader。

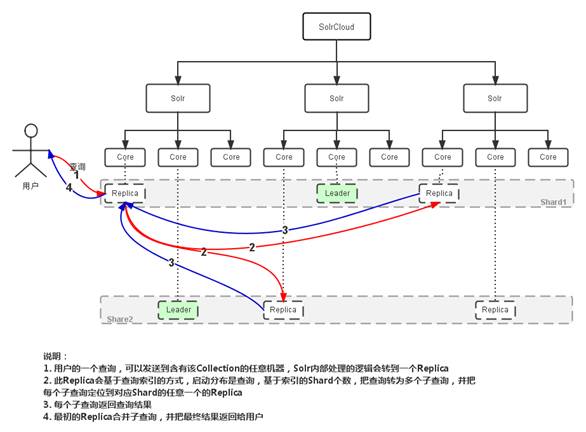
·         ****Zookeeper****: Zookeeper提供分布式锁功能，对SolrCloud是必须的。它处理Leader选举。Solr可以以内嵌的Zookeeper运行，但是建议用独立的，并且最好有3个以上的主机。



**[创建索引过程](http://www.processon.com/view/link/526e0cc10cf2ddc32cf456c5)**



**[检索过程](http://www.processon.com/view/link/526f1b060cf2ddc32cf5707c)**



**[Shard Splitting](http://www.processon.com/view/link/526e0b490cf2ddc32cf45338)**

