Solr 7.1 Cloud

1. 课程计划

1、solr集群搭建

2、使用solrj管理solr集群

2. 什么是SolrCloud

SolrCloud(solr 云)是Solr提供的分布式搜索方案，当你需要大规模，容错，分布式索引和检索能力时使用 SolrCloud。当一个系统的索引数据量少的时候是不需要使用SolrCloud的，当索引量很大，搜索请求并发很高，这时需要使用SolrCloud来满足这些需求。

SolrCloud是基于Solr和Zookeeper的分布式搜索方案，它的主要思想是使用Zookeeper作为集群的配置信息中心。

它有几个特色功能：

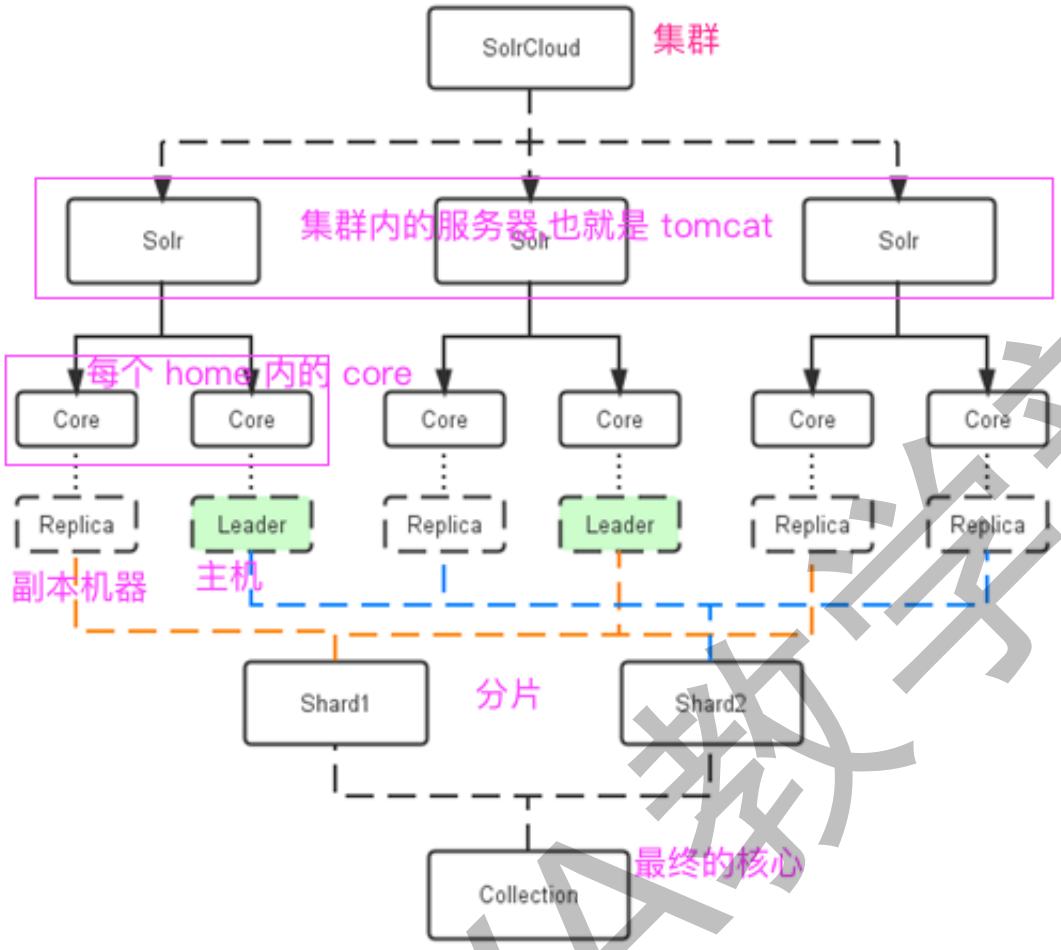
1）集中式的配置信息

2）自动容错

3）近实时搜索

4）查询时自动负载均衡

3. Solr集群的系统架构



3.1. 物理结构

本课程中 使用三个Solr实例（ 每个实例包括两个Core），组成一个SolrCloud。

3.2. 逻辑结构

索引集合包括两个Shard（shard1和shard2），shard1和shard2分别由三个Core组成，其中一个Leader两个Replication，Leader是由zookeeper选举产生，zookeeper控制每个shard上三个Core的索引数据一致，解决高可用问题。

用户发起索引请求分别从shard1和shard2上获取，解决高并发问题。

3.2.1. collection

Collection在SolrCloud集群中是一个逻辑意义上的完整的索引结构。它常常被划分为一个或多个Shard（分片），它们使用相同的配置信息。

比如：针对商品信息搜索可以创建一个collection。

collection=shard1+shard2+....+shardX

3.2.2. Core

每个Core是Solr中一个独立运行单位，提供 索引和搜索服务。一个shard需要由一个Core或多个Core组成。由于collection由多个shard组成所以collection一般由多个core组成。

3.2.3. Master或Slave

Master是master-slave结构中的主结点（通常说主服务器），Slave是master-slave结构中的从结点（通常说从服务器或备服务器）。同一个Shard下master和slave存储的数据是一致的，这是为了达到高可用目的。

3.2.4. Shard

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

3.3. 需要实现的solr集群架构

3.3.1 集群管理工具

Zookeeper作为集群的管理工具。

1、集群管理：容错、负载均衡。

2、配置文件的集中管理

3、集群的入口

3.3.2 zookeeper 集群

需要实现zookeeper 高可用。需要搭建集群。建议是奇数节点。本案例需要三个zookeeper服务器。

本案例搭建solr集群需要7台服务器。

需要三个zookeeper节点,搭建 zookeeper 集群

需要四个tomcat节点。搭建 solr 集群

建议虚拟机的内容1G以上。

4. 环境准备

```
CentOS-6.8
jdk1.8
tomact8.5
zookeeper-3.4.11
solr-7.10
```

5. 安装步骤

5.1. Zookeeper集群搭建

> 第一步：需要安装jdk环境。

> 第二步：把zookeeper的压缩包上传到服务器。

> 第三步：解压缩。

> 第四步：把zookeeper复制三份。

```
[root@localhost ~]# mkdir /usr/local/solr-cloud
[root@localhost ~]# cp -r zookeeper-3.4.11 /usr/local/solr-cloud/zookeeper01
[root@localhost ~]# cp -r zookeeper-3.4.11 /usr/local/solr-cloud/zookeeper02
[root@localhost ~]# cp -r zookeeper-3.4.11 /usr/local/solr-cloud/zookeeper03
```

> 第五步：在每个zookeeper目录下创建一个data目录。

> 第六步：在data目录下创建一个myid文件，文件名就叫做“myid”。内容就是每个实例的id。例如1、2、3

```
    [root@localhost data]# echo 1 >> myid
    [root@localhost data]# ll
    total 4
    -rw-r--r--. 1 root root 2 Apr  7 18:23 myid
    [root@localhost data]# cat myid
     1
```

> 第七步：修改配置文件。把conf目录下的zoo_sample.cfg文件改名为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=/usr/local/solr-cloud/zookeeper01/data
# the port at which the clients will connect
clientPort=2181
# the maximum number of client connections.
# increase this if you need to handle more clients
#maxClientCnxns=60
#
# 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=3
# Purge task interval in hours
# Set to "0" to disable auto purge feature
#autopurge.purgeInterval=1
# 1,2,3代表 service 的 id, 端口中的第一个代表zookeeper 内部通讯端口,第二个代表投票时候的端口
server.1=10.0.135.131:2881:3881
server.2=10.0.135.131:2882:3882
server.3=10.0.135.131:2883:3883
```

数据目录

```
server.1=10.0.135.131:2881:3881
server.2=10.0.135.131:2882:3882
server.3=10.0.135.131:2883:3883
```

> 第八步：启动每个zookeeper实例。

启动

```
zkServer.sh start
```

查看zookeeper的状态：

```
zkServer.sh status
```

5.2. Solr集群的搭建

> 参考单机版创建四份 solr

5.2.1修改 solrhome 下的 solr.xml

```
<solr>

  <solrcloud>

    <str name="host">10.0.135.131</str>
    <int name="hostPort">8180</int>
    <str name="hostContext">${hostContext:solr}</str>

    <bool name="genericCoreNodeNames">${genericCoreNodeNames:true}</bool>

    <int name="zkClientTimeout">${zkClientTimeout:30000}</int>
    <int name="distribUpdateSoTimeout">${distribUpdateSoTimeout:600000}</int>
    <int name="distribUpdateConnTimeout">${distribUpdateConnTimeout:60000}</int>
    <str name="zkCredentialsProvider">${zkCredentialsProvider:org.apache.solr.common.cloud.DefaultZkCredentialsPr
    <str name="zkACLProvider">${zkACLProvider:org.apache.solr.common.cloud.DefaultZkACLProvider}</str>

  </solrcloud>

  <shardHandlerFactory name="shardHandlerFactory"
    class="HttpShardHandlerFactory">
    <int name="socketTimeout">${socketTimeout:600000}</int>
    <int name="connTimeout">${connTimeout:60000}</int>
  </shardHandlerFactory>
```

当前 solr 对应的 tomcat 地址

当前 solr 对应的 tomcat 端口

5.2.3 修改每个 solrcore 的配置

> 给每个solrcore 的 core.properties 中额外添加内容

```
name=collection1
#节点名称,每个core不一样
coreNodeName=first
#分片名称,每个保持不一样
shard=s01
```

5.3 使用 zookeeper 管理 solr 配置

> 让zookeeper统一管理配置文件。需要把solrhome/collection1/conf目录上传到zookeeper。上传任意solrhome中的配置文件即可。

> 使用工具上传配置文件：solr-7.1.0/server/scripts/cloud-scripts/zkcli.sh 需要先将solr的压缩包上传

```
./zkcli.sh -zkhost 10.0.135.131:2181,10.0.135.131:2182,10.0.135.131:2183 -cmd
upconfig -confdir /usr/local/solr-cloud/solrhome01/collection1/conf -confname
myconf
## myconf 代表配置的名字
##/usr/local/solr-cloud/solrhome01/collection1/conf代表配置文件的位置,任意选择一个
solrcore 的配置文件即可
```

5.3.1查看配置文件

查看zookeeper上的配置文件：

使用zookeeper目录下的bin/zkCli.sh命令查看zookeeper上的配置文件:

```
[root@localhost bin]# ./zkCli.sh

[zk: localhost:2181(CONNECTED) 0] ls /
[configs, zookeeper]

[zk: localhost:2181(CONNECTED) 1] ls /configs
[myconf]

[zk: localhost:2181(CONNECTED) 2] ls /configs/myconf

[admin-extra.menu-top.html, currency.xml, protwords.txt, mapping-FoldToASCII.txt,
_schema_analysis_synonyms_english.json, _rest_managed.json, solrconfig.xml,
_schema_analysis_stopwords_english.json, stopwords.txt, lang, spellings.txt,
mapping-ISOLatin1Accent.txt, admin-extra.html, xslt, synonyms.txt, scripts.conf,
update-script.js, velocity, elevate.xml, admin-extra.menu-bottom.html, clustering,
schema.xml]
[zk: localhost:2181(CONNECTED) 3]
退出：
[zk: localhost:2181(CONNECTED) 3] quit
```

使用以下命令连接指定的zookeeper服务：

```
./zkCli.sh -server 10.0.135.131:2183
```

5.4 修改 Tomcat 脚本

> 修改tomcat/bin目录下的catalina.sh 文件，关联solr和zookeeper。把此配置添加到该文件中：

```
JAVA_OPTS="-DzkHost=10.0.135.131:2181,10.0.135.131:2182,10.0.135.131:2183"
```

```
#   UMASK           (Optional) Override Tomcat's default UMASK of 0027
#
#   USE_NOHUP       (Optional) If set to the string true the start command will
#                   use nohup so that the Tomcat process will ignore any hangup
#                   signals. Default is "false" unless running on HP-UX in which
#                   case the default is "true"
# -----------------------------------------------------------------------------

# OS specific support.  $var _must_ be set to either true or false.
JAVA_OPTS="-DzkHost=10.0.135.131:2181,10.0.135.131:2182,10.0.135.131:2183"
cygwin=false
darwin=false
os400=false
hpux=false
case "`uname`" in
CYGWIN*) cygwin=true;;
Darwin*) darwin=true;;
OS400*) os400=true;;
HP-UX*) hpux=true;;
esac
```

注意尽量在靠前位置设置

5.5启动每个 tomact

> 启动每个tomcat实例。要保证zookeeper集群是启动状态。

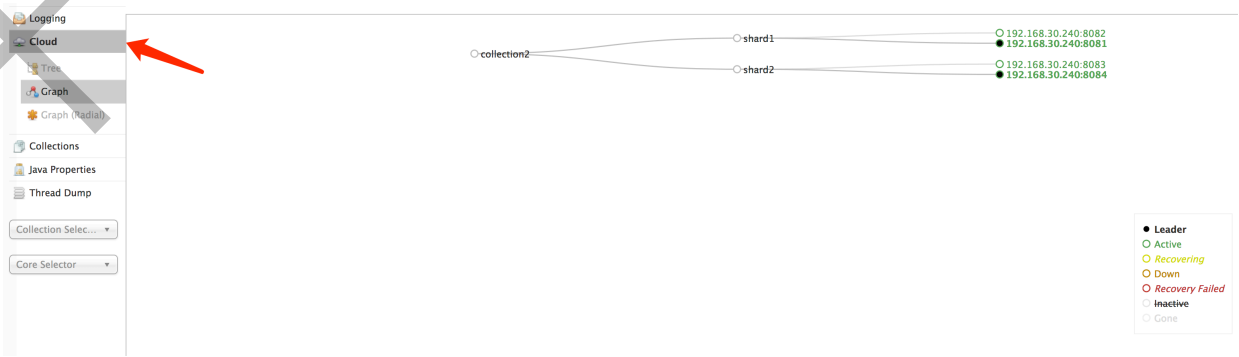
5.6访问任意一个 solr 地址

> 访问集群,但是会提示错误,没有核心,不用管它,执行下一步

5.7开始创建集群

> 创建新的Collection进行分片处理。
>
> 以下指令含义,创建一个名字为collection2的核心,包含两个分片,每个分片两台机器
>
> 最终提示 success 代表创建成功

```
http://10.0.135.131:8180/solr/admin/collections?
action=CREATE&name=collection2&numShards=2&replicationFactor=2
```



> 如果有不需要的核心要删除,可以使用以下命令 name 代表被删除的核心的名字

```
http://10.0.135.131:8180/solr/admin/collections?action=DELETE&name=collection1
```

6. 使用solrJ

6.1. 添加文档

使用步骤：

> 第一步：把solrJ相关的jar包添加到工程中。
>
> 第二步：创建一个SolrServer对象，需要使用CloudSolrServer子类。构造方法的参数是zookeeper的地址列表。
>
> 第三步：需要设置DefaultCollection属性。
>
> 第四步：创建一SolrInputDocument对象。
>
> 第五步：向文档对象中添加域
>
> 第六步：把文档对象写入索引库。
>
> 第七步：提交。

```
@Test
public void testSolrCloudAddDocument() throws Exception {
// 第一步：把solrJ相关的jar包添加到工程中。
```

```
// 第二步：创建一个SolrServer对象，需要使用CloudSolrServer子类。构造方法的参数是
zookeeper的地址列表。

//参数是zookeeper的地址列表，使用逗号分隔
 CloudSolrClient solrClient = new
CloudSolrClient.Builder().withZkHost("10.0.135.131:2181,10.0.135.131:2182,10.0.135
.131:2183").build();
// 第三步：需要设置DefaultCollection属性。

solrClient.setDefaultCollection("collection2");

// 第四步：创建一SolrInputDocument对象。

SolrInputDocument document = new SolrInputDocument();

// 第五步：向文档对象中添加域

document.addField("item_title", "测试商品");

document.addField("item_price", "100");

document.addField("id", "test001");
// 第六步：把文档对象写入索引库。
solrClient.add(document);
// 第七步：提交。
solrClient.commit();

}
```

6.2. 查询文档

创建一个CloudSolrServer对象，其他处理和单机版一致。

7. 整合 spring

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:context="http://www.springframework.org/schema/context"
     xmlns:p="http://www.springframework.org/schema/p"
xmlns:aop="http://www.springframework.org/schema/aop"
xmlns:tx="http://www.springframework.org/schema/tx"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans4.3.xsd
 http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context4.3.xsd
 http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop4.3.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx4.3.xsd
http://www.springframework.org/schema/util
http://www.springframework.org/schema/util/spring-util4.3.xsd">

   <!--版本更新说明:
    1.从solr5.x开始,HttoSolrServer就变成了HttpSolrClient
    2.查看源码发现,之前的构造方法已经修改,以前的注入方法也不再适用.主要由一个静态类
builder来构造,而builder需要一个baseUrl,
       所以差不多就是之前的单baseUrl的构造方法(源码179,830)
 -->
<bean class="org.apache.solr.client.solrj.impl.HttpSolrClient"
id="httpSolrClient">
    <constructor-arg name="builder" value="solrCore的远程地址">
</constructor-arg></bean>

    <!-- 集群版solr服务
 使用自定义对象,在内部将 cloud 代码封装,提供构造参数,传入 zk 地址和 collection 名字,内部
构造 cloud 对象并返回
-->

</beans>
```

8 错误

8.1 集群副本

> 因为没有给集群中的机器配置副本,所以会提示副本不存在异常,但是不影响使用

8.2 创建集群错误

> 在执行命令创建集群的时候可能会报去烧 DataImporthandler, 将依赖包复制到项目的 lib 目录下即可