

**SolrCloud**

**集群搭建**

**讲师：传智.入云龙**

# 什么是SolrCloud

## 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

**zookeeper是个什么玩意？**

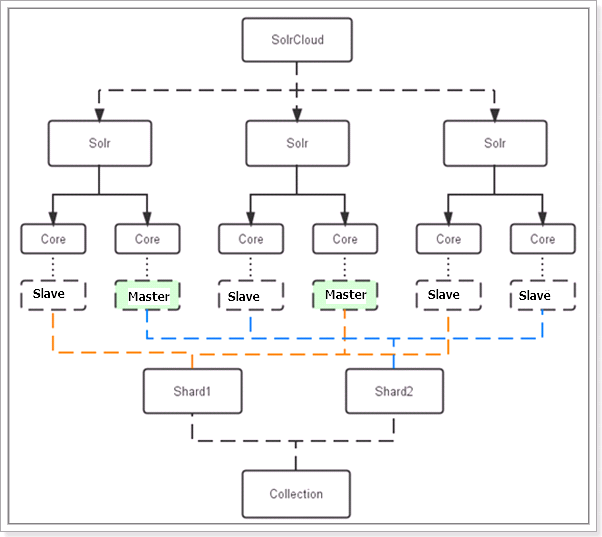
顾名思义zookeeper就是动物园管理员，他是用来管hadoop（大象）、Hive(蜜蜂)、pig(小猪)的管理员， Apache Hbase和 Apache Solr 的分布式集群都用到了zookeeper；Zookeeper:是一个分布式的、开源的程序协调服务，是hadoop项目下的一个子项目。

## SolrCloud结构

SolrCloud为了降低单机的处理压力，需要由多台服务器共同来完成索引和搜索任务。实现的思路是将索引数据进行Shard（分片）拆分，每个分片由多台的服务器共同完成，当一个索引或搜索请求过来时会分别从不同的Shard的服务器中操作索引。

SolrCloud需要Solr基于Zookeeper部署，Zookeeper是一个集群管理软件，由于SolrCloud需要由多台服务器组成，由zookeeper来进行协调管理。

下图是一个SolrCloud应用的例子：



对上图进行图解，如下：



### 物理结构

三个Solr实例（ 每个实例包括两个Core），组成一个SolrCloud。

### 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。

#### 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

#### 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。

#### 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。

#### 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。

# SolrCloud搭建

本教程的这套安装是单机版的安装，所以采用伪集群的方式进行安装，如果是真正的生成环境，将伪集群的ip改下就可以了，步骤是一样的。

SolrCloud结构图如下：



## 环境准备

CentOS-6.4-i386-bin-DVD1.iso

jdk-7u72-linux-i586.tar.gz

apache-tomcat-7.0.47.tar.gz

zookeeper-3.4.6.tar.gz

solr-4.10.3.tgz

## 环境安装

### CentOs 6.4安装

略

### jdk7安装

略

### zookeeper集群安装

**第一步**：解压zookeeper，tar -zxvf zookeeper-3.4.6.tar.gz将zookeeper-3.4.6拷贝到/usr/local/solrcloud下，复制三份分别并将目录名改为zookeeper1、zookeeper2、zookeeper3

**第二步**：进入zookeeper1文件夹，创建data目录。并在data目录中创建一个myid文件内容为“1”（echo 1 >> data/myid）。

**第三步**：进入conf文件夹，把zoo\_sample.cfg改名为zoo.cfg

**第四步**：修改zoo.cfg。

修改：

dataDir=/usr/local/solrcloud/zookeeper1/data

clientPort=2181（zookeeper2中为2182、zookeeper3中为2183）

添加：

server.1=192.168.25.150:2881:3881

server.2=192.168.25.150:2882:3882

server.3=192.168.25.150:2883:3883

|  |
| --- |
| # 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/solrcloud/zookeeper1/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  server.1=192.168.25.150:2881:3881  server.2=192.168.25.150:2882:3882  server.3=192.168.25.150:2883:3883 |

**第五步**：对zookeeper2、3中的设置做第二步至第四步修改。

zookeeper2：

myid内容为2

dataDir=/usr/local/solrcloud/zookeeper2/data

clientPort=2182

Zookeeper3：

的myid内容为3

dataDir=/usr/local/solrcloud/zookeeper3/data

clientPort=2183

**第六步**：启动三个zookeeper

/usr/local/solrcloud/zookeeper1/bin/zkServer.sh start

/usr/local/solrcloud/zookeeper2/bin/zkServer.sh start

/usr/local/solrcloud/zookeeper3/bin/zkServer.sh start

查看集群状态：

/usr/local/solrcloud/zookeeper1/bin/zkServer.sh status

/usr/local/solrcloud/zookeeper2/bin/zkServer.sh status

/usr/local/solrcloud/zookeeper3/bin/zkServer.sh status

**第七步**：开启zookeeper用到的端口，或者直接关闭防火墙。

service iptables stop

## tomcat安装

第一步：将apache-tomcat-7.0.47.tar.gz解压

tar -zxvf apache-tomcat-7.0.47.tar.gz

第二步：把解压后的tomcat复制到/usr/local/solrcloud/目录下复制四份。

/usr/local/solrcloud/tomcat1

/usr/local/solrcloud/tomcat2

/usr/local/solrcloud/tomcat3

/usr/local/solrcloud/tomcat4

第三步：修改tomcat的server.xml

vim tomcat2/conf/server.xml，把其中的端口后都加一。保证两个tomcat可以正常运行不发生端口冲突。

## solr单机部署

### 全新部署

参考solr教案单机部署方法。

## solrCloud部署

### 启动zookeeper

solrCloud部署依赖zookeeper，需要先启动每一台zookeeper服务器。

### zookeeper管理配置文件

由于zookeeper统一管理solr的配置文件（主要是schema.xml、solrconfig.xml）， solrCloud各各节点使用zookeeper管理的配置文件。

将上边部署的solr单机的conf拷贝到/home/solr下。

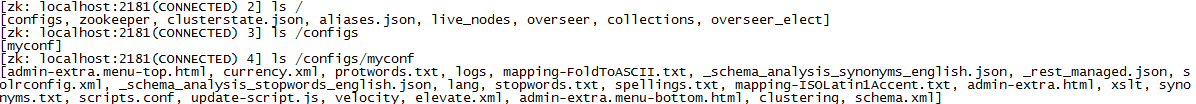
执行下边的命令将/home/solr/conf下的配置文件上传到zookeeper（此命令为单条命令，虽然很长o(╯□╰)o）。此命令在solr-4.10.3/example/scripts/cloud-scripts/目录下：

|  |
| --- |
| ./zkcli.sh -zkhost 192.168.25.150:2181,192.168.25.150:2182,192.168.25.150:2183 -cmd upconfig -confdir /usr/local/solrcloud/solrhome1/collection1/conf -confname myconf |

登陆zookeeper服务器查询配置文件：

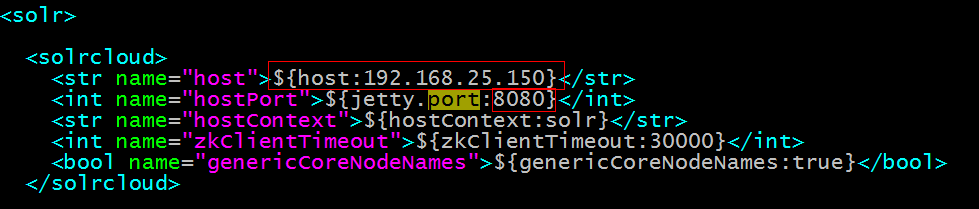
cd /usr/local/zookeeper/bin/

./zkCli.sh



### 修改SolrCloud监控端口为8080：

修改每个solrhome的solr.xml文件。



### 每一台solr和zookeeper关联

修改每一台solr的tomcat 的 bin目录下catalina.sh文件中加入DzkHost指定zookeeper服务器地址：

JAVA\_OPTS="-DzkHost=192.168.25.150:2181,192.168.25.150:2182,192.168.25.150:2183"

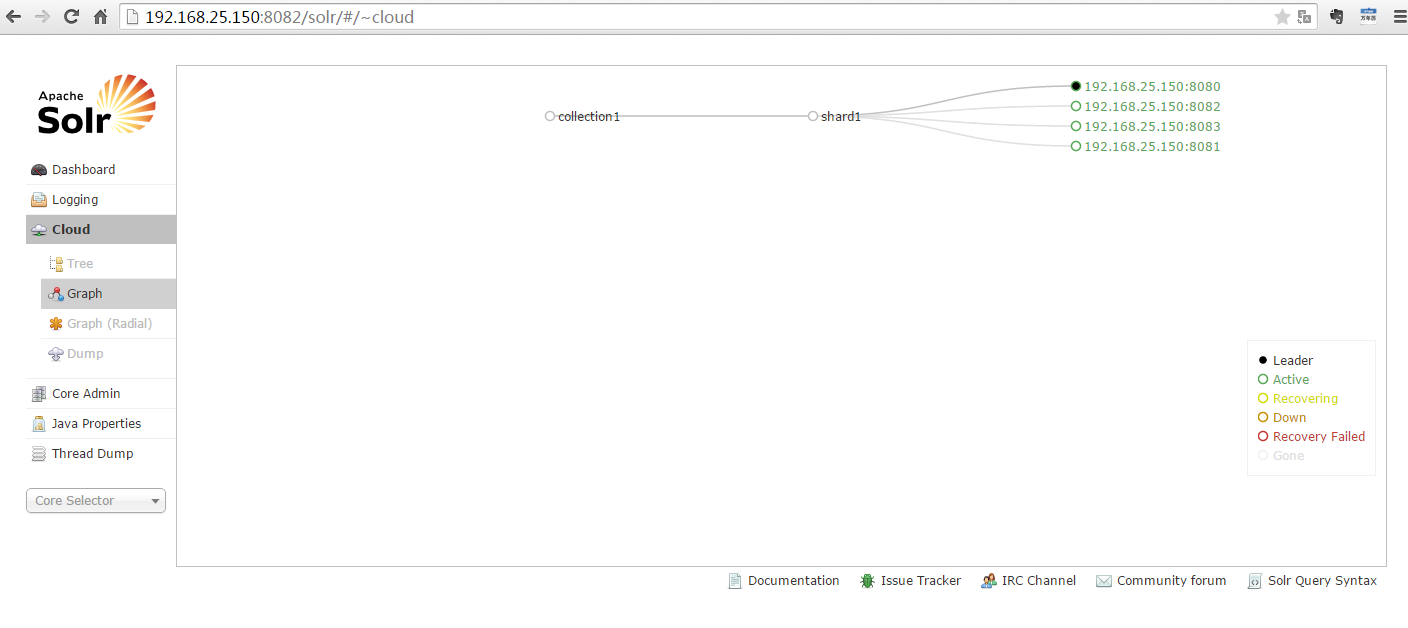
（可以使用vim的查找功能查找到JAVA\_OPTS的定义的位置，然后添加）

### 启动所有的solr服务

启动每一台solr的tomcat服务。

### 访问solrcloud

访问任意一台solr，左侧菜单出现Cloud：

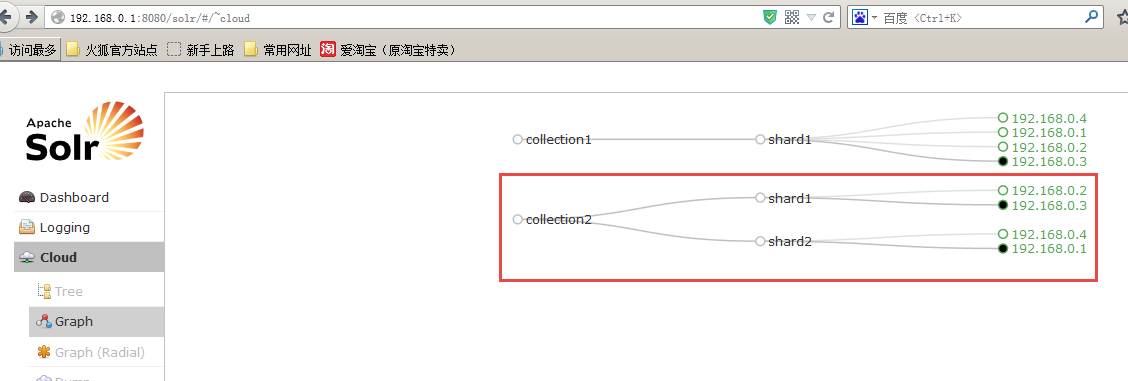


### SolrCloud集群配置

上图中的collection1集群只有一片，可以通过下边的方法配置新的集群。

如果集群中有四个solr节点创建新集群collection2，将集群分为两片，每片两个副本。

http://192.168.25.150:8080/solr/admin/collections?action=CREATE&name=collection2&numShards=2&replicationFactor=2



删除集群命令；

http://192.168.25.150:8080/solr/admin/collections?action=DELETE&name=collection1

执行后原来的collection1删除，如下：



更多的命令请参数官方文档：apache-solr-ref-guide-4.10.pdf

### 启动solrCloud注意

启动solrCloud需要先启动solrCloud依赖的所有zookeeper服务器，再启动每台solr服务器。

# solrJ访问solrCloud

**public** **class** SolrCloudTest {

// zookeeper地址

**private** **static** String *zkHostString* = "192.168.25.150:2181,192.168.25.150:2182,192.168.25.150:2183";

// collection默认名称，比如我的solr服务器上的collection是collection2\_shard1\_replica1，就是去掉“\_shard1\_replica1”的名称

**private** **static** String *defaultCollection* = "collection1";

// cloudSolrServer实际

**private** CloudSolrServer cloudSolrServer;

// 测试方法之前构造 CloudSolrServer

@Before

**public** **void** init() {

cloudSolrServer = **new** CloudSolrServer(*zkHostString*);

cloudSolrServer.setDefaultCollection(*defaultCollection*);

cloudSolrServer.connect();

}

// 向solrCloud上创建索引

@Test

**public** **void** testCreateIndexToSolrCloud() **throws** SolrServerException,

IOException {

SolrInputDocument document = **new** SolrInputDocument();

document.addField("id", "100001");

document.addField("title", "李四");

cloudSolrServer.add(document);

cloudSolrServer.commit();

}

// 搜索索引

@Test

**public** **void** testSearchIndexFromSolrCloud() **throws** Exception {

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

query.setQuery("\*:\*");

**try** {

QueryResponse response = cloudSolrServer.query(query);

SolrDocumentList docs = response.getResults();

System.*out*.println("文档个数：" + docs.getNumFound());

System.*out*.println("查询时间：" + response.getQTime());

**for** (SolrDocument doc : docs) {

ArrayList title = (ArrayList) doc.getFieldValue("title");

String id = (String) doc.getFieldValue("id");

System.*out*.println("id: " + id);

System.*out*.println("title: " + title);

System.*out*.println();

}

} **catch** (SolrServerException e) {

e.printStackTrace();

} **catch** (Exception e) {

System.*out*.println("Unknowned Exception!!!!");

e.printStackTrace();

}

}

// 删除索引

@Test

**public** **void** testDeleteIndexFromSolrCloud() **throws** SolrServerException, IOException {

// 根据id删除

UpdateResponse response = cloudSolrServer.deleteById("zhangsan");

// 根据多个id删除

// cloudSolrServer.deleteById(ids);

// 自动查询条件删除

// cloudSolrServer.deleteByQuery("product\_keywords:教程");

// 提交

cloudSolrServer.commit();

}

}