SolrCloud 课程内容

教学目标

了解集群和分布式

掌握 Zookeeper 服务的部署

掌握 Solr 集群的搭建

掌握 SolrJ 对 Solr 集群的访问

1、认识系统架构

1.1、集群概述

1.1.1、为什么要使用服务器集群

此前我们在进行项目的开发时，使用的标准的 Javaweb 的工程开发模式，这种方式使用一个服务器部署和发布服务器程序，又称为单点服务器，在这种状况下，如果单点服务器出现故障 ,将直接影响系统的使用。而停止服务可能会带来非常严重的问题 ,造成重大损失，所以这种项目架构的方式显然存在较大的风险。

在此基础上出现了，服务器的集群架构模式，这里我们可以总结一下，传统的单点服务器上存在的问题：

1）单服务器的处理能力有限，尤其在高并发环境下，单服务器的并发处理能力的瓶颈很难突破，每台服务器都存在有限的最大连接数。

2）单服务器的容错率低，如果服务器发生故障，那么整个服务将无法访问

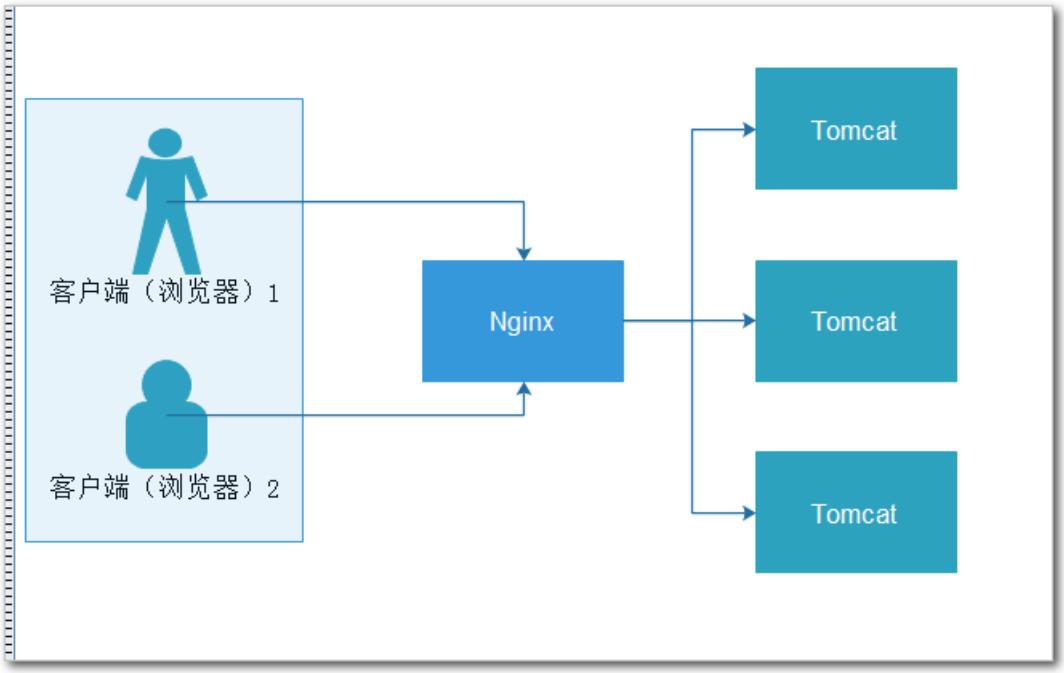
3）单服务器运算能力有限，逐渐无法满足高性能服务器的需求。

以上就是我们在进行项目架构时需要考虑的三高：高并发，高可用和高性能，而这也是我们在实际项目中使用分布式集群的主要原因。

1.1.2、什么是集群

集群就是将多个系统连接到一起，使多台服务器能够像一台机器那样工作或者看起来好像一台机器的一种技术。集群的主要任务是为了提高系统的稳定性和网络中心的数据处理能力及服务能力。

从以上集群的主要看出，集群可以协调和统筹多台服务器的处理，提供高效稳定的服务。那么这是怎么来实现的呢？在集群中实现高并发和可扩展的方案是采用负载均衡技术，而为了提供高可用的服务则采用数据冗余技术。



负载均衡就是通过集群架构多台服务器，通过技术实现对用户并发请求进行均衡操作，以分担单个服务器的压力，常见的负载均衡的实现方式事通过反向代理，有反向代理服务器对用户的请求进行处理，分发给被代理的哥哥服务器进行处理。

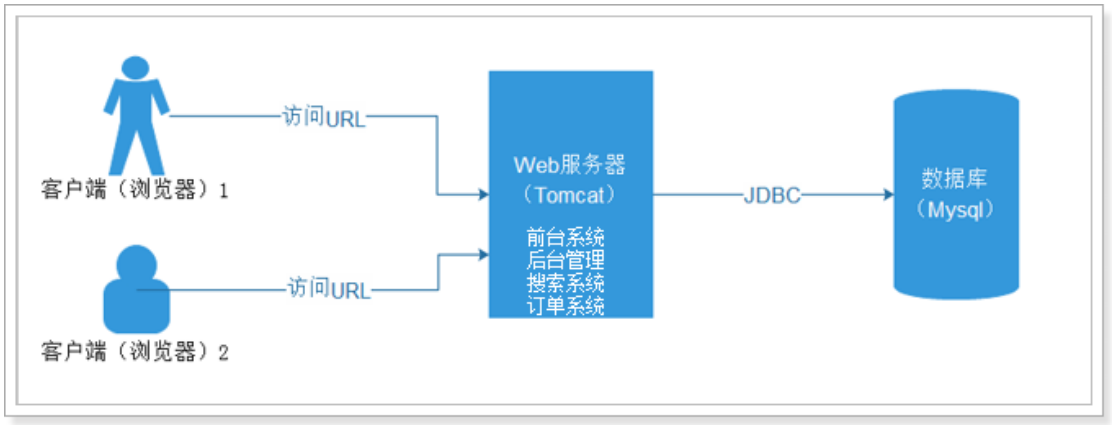
数据冗余技术主要指对可能出现单点故障的服务器进行冗余处理，提供多台服务器，按

照特定的机制进行相互的同步以保持数据的一致性，当一台服务器出现故障之后，可以实现自动的快速切换，提供高可用的服务器支持。

1.2、分布式架构

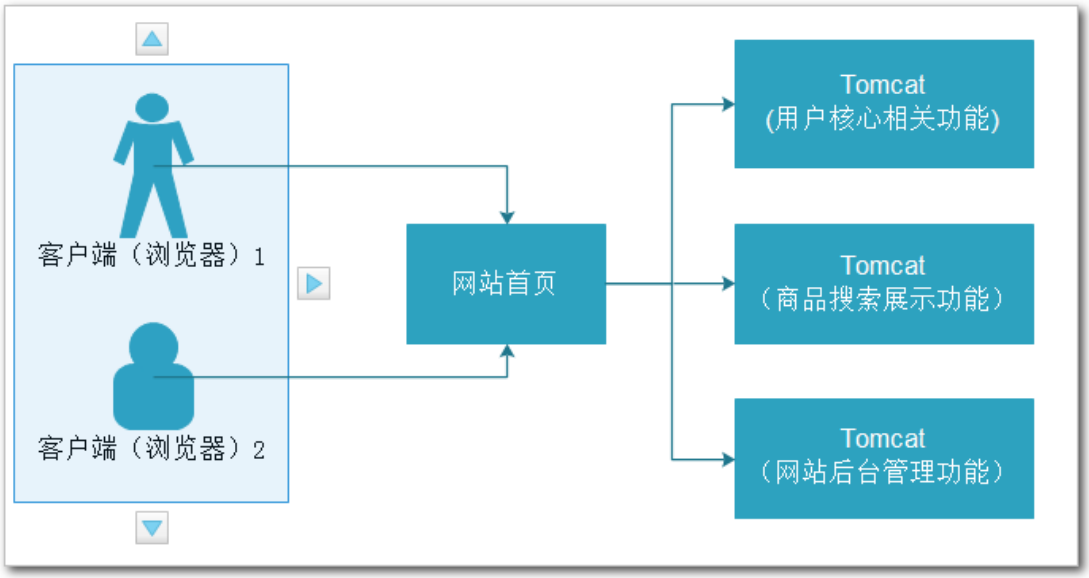
1.2.1、为什么使用分布式

传统的 web 架构采用单个工程进行处理，系统功能比较繁杂，业务之间的耦合性高，不理单个模块的开发和维护，也不利于功能的水平扩展。在集群部署中，也不理于集群的管理。



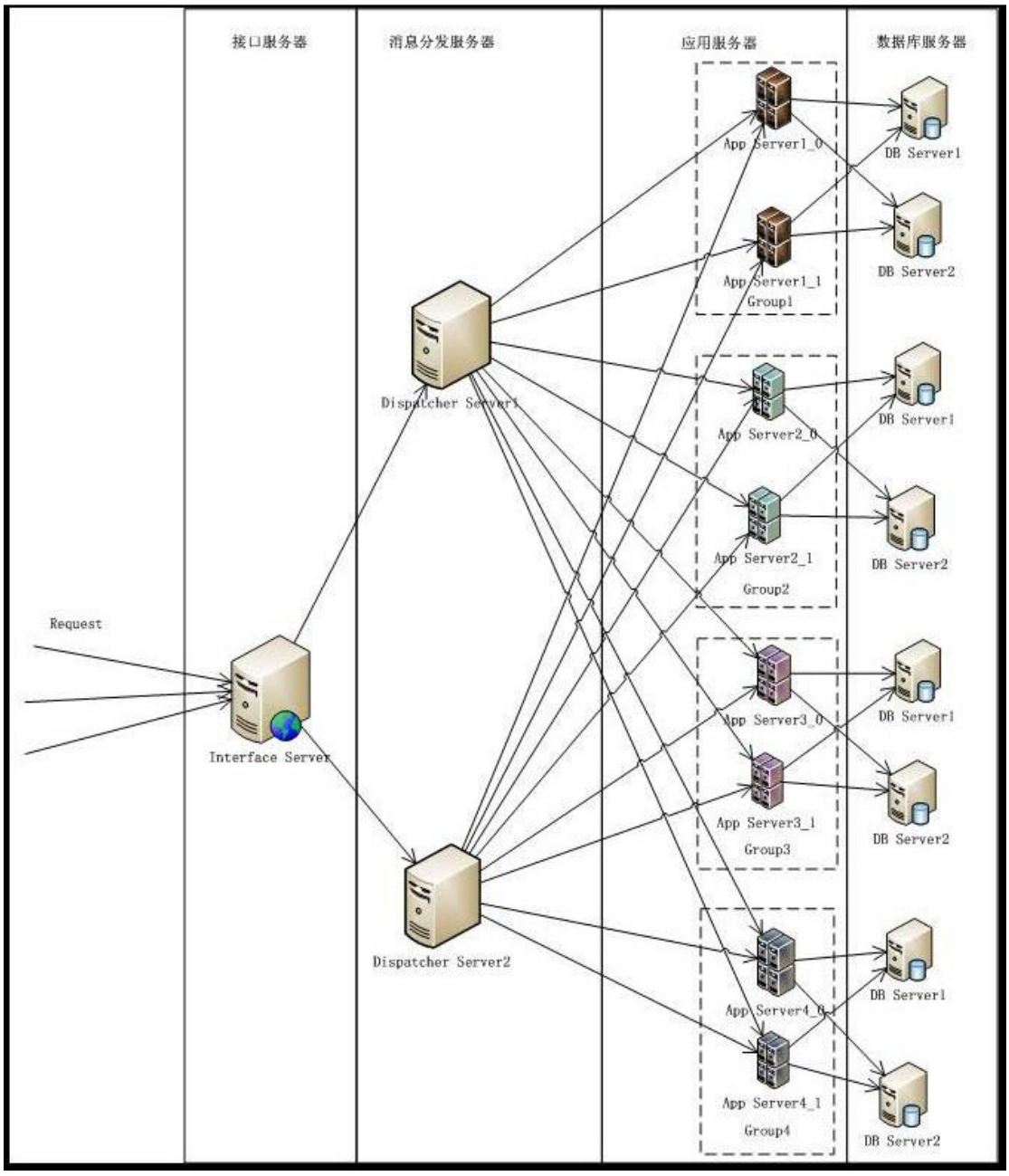
1.2.2、分布式架构

分布式是指将多台服务器集中在一起，每台服务器都实现总体中的不同业务，做不同的事情。并且每台服务器都缺一不可，如果某台服务器故障，则网站部分功能缺失，或导致整体无法运行。存在的主要作用是大幅度的提高效率，缓解服务器的访问和存储压力。



1.3、分布式集群综合架构

一般分布式中的每一个节点，都可以做集群。这样的系统架构，我们通常称为分布式集群架构。

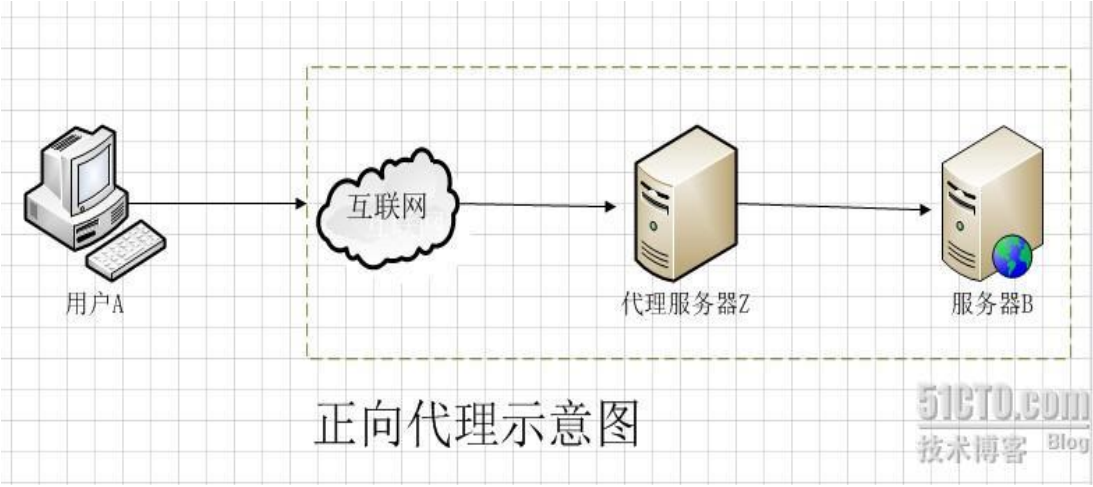


1.3、代理技术

1.4.1、正向代理（Forward Proxy）

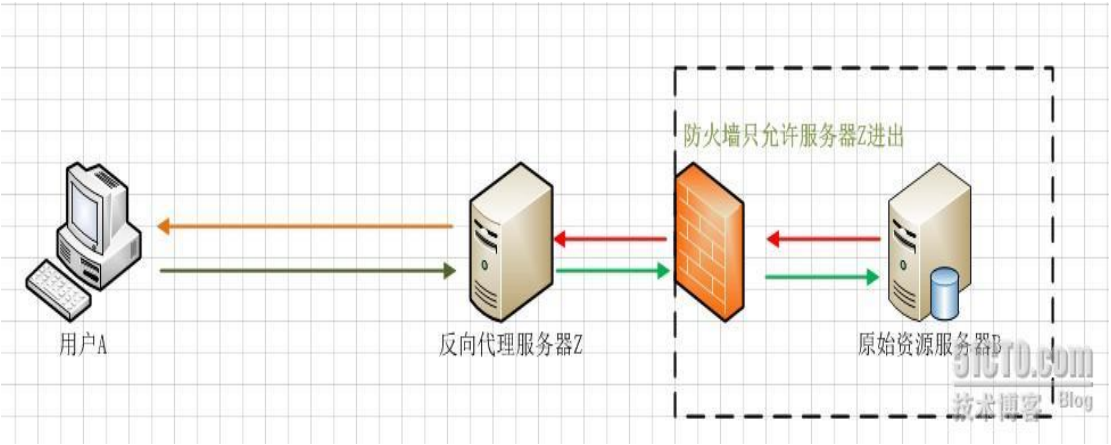
一般情况下，如果没有特别说明，代理技术默认说的是正向代理技术。关于正向代理的概念如下： 正向代理(forward)是一个位于客户端【用户 A】和原始服务器(origin server)【服务器 B】之间的服务器【代理服务器 Z】，为了从原始服务器取得内容，用户 A 向代理服务器 Z 发送一个请求并指定目标(服务器 B)，然后代理服务器 Z 向服务器 B 转交请求并将

获得的内容返回给客户端。客户端必须要进行一些特别的设置才能使用正向代理。



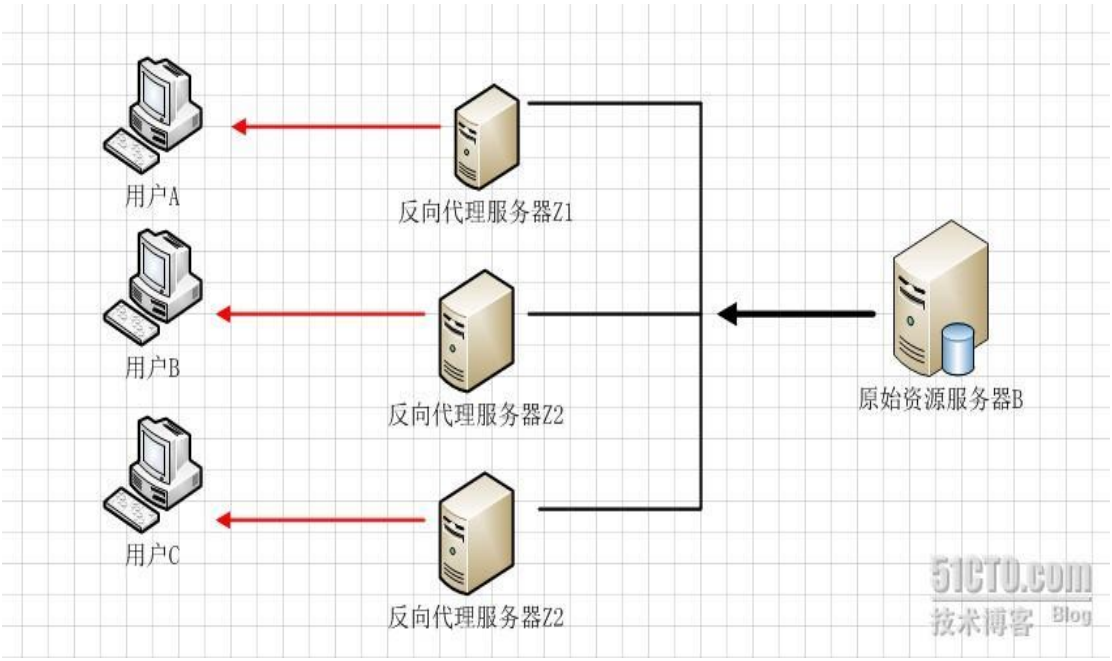
1.4.2、反向代理（reverse proxy）

反向代理正好与正向代理相反，对于客户端而言代理服务器就像是原始服务器，并且客户端不需要进行任何特别的设置。客户端向反向代理的命名空间(name-space)中的内容发送普通请求，接着反向代理将判断向何处(原始服务器)转交请求，并将获得的内容返回给客户端。 使用反向代理服务器的作用如下：



用户 A 始终认为它访问的是原始服务器 B 而不是代理服务器 Z，但实用际上反向代理服务器接受用户 A 的应答，从原始资源服务器 B 中取得用户 A 的需求资源，然后发送给用户 A。由于防火墙的作用，只允许代理服务器 Z 访问原始资源服务器 B。在这个虚拟的环境下，防火墙和反向代理的共同作用保护了原始资源服务器 B，但用户 A 并不知情。

当反向代理服务器不止一个的时候，我们甚至可以把它们做成集群，当更多的用户访问资源服务器 B 的时候，让不同的代理服务器 Z（x）去应答不同的用户，然后发送不同用户需要的资源。而且反向代理服务器像正向代理服务器一样拥有 CACHE 的作用，它可以缓存原始资源服务器 B 的资源，而不是每次都要向原始资源服务器 B 请求数据，特别是一些静态的数据，比如图片和文件。



2、Zookeeper 分布式协调服务

2.1、什么是 Zookeeper

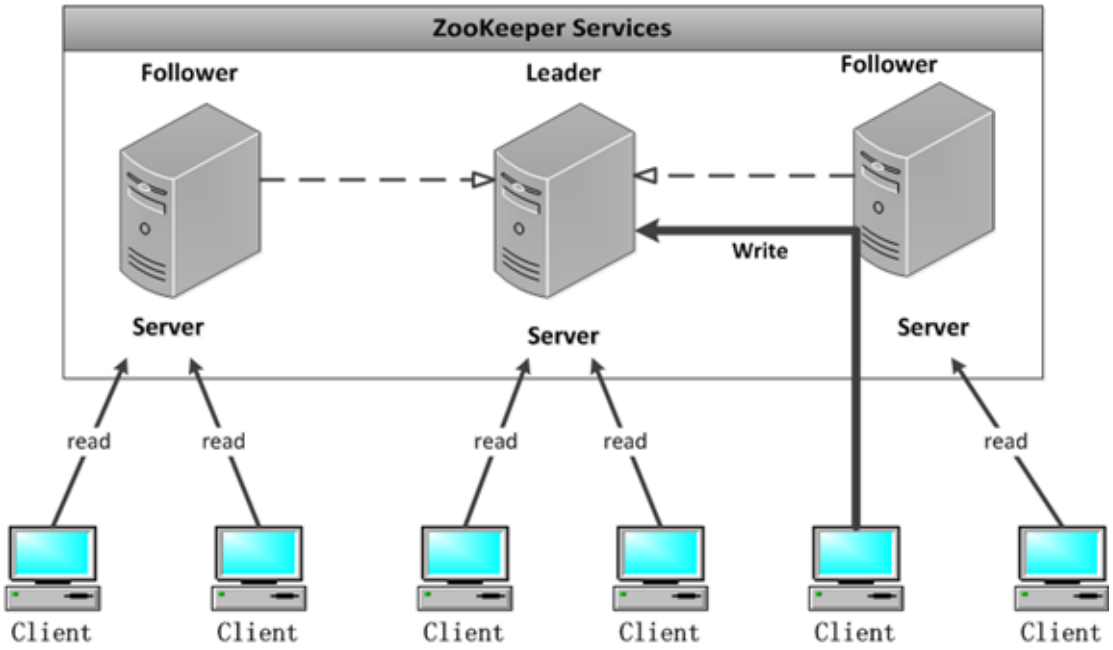
Zookeeper 是集群分布式中大管家，分布式集群系统比较复杂，子模块很多，但是子模块往往不是孤立存在的，它们彼此之间需要协作和交互，各个子系统就好比动物园里的动物，为了使各个子系统能正常为用户提供统一的服务，必须需要一种机制来进行协调——这就是 ZooKeeper。Zookeeper 是为分布式应用程序提供高性能协调服务的工具集合，也是 Google 的 Chubby 一个开源的实现，是 Hadoop 的分布式协调服务。

2.2、Zookeeper 的集群结构

在 ZooKeeper 集群当中，集群中的服务器角色有两种：1 个 Leader 和多个 Follower，具体功能如下：

1）领导者(leader)，负责进行投票的发起和决议，监控集群中的节点是否存活（心跳机制），进行分配资源

2）follower 用于接受客户端请求并向客户端返回结果，在选主过程中参与投票



Zookeeper 集群的特点：

A：Zookeeper：一个 leader，多个 follower 组成的集群

B：全局数据一致：每个 server 保存一份相同的数据副本，client 无论连接到哪个 server，数据都是一致的

C：数据更新原子性，一次数据更新要么成功，要么失败

D：实时性，在一定时间范围内，client 能读到最新数据

E：半数机制：整个集群中只要有一半以上存活，就可以提供服务。因此通常 Zookeeper 由 2n+1 台 servers 组成，每个 server 都知道彼此的存在。每个 server 都维护的内存状态

镜像以及持久化存储的事务日志和快照。为了保证 Leader 选举能过得到多数的支持，所以 ZooKeeper 集群的数量一般为奇数。对于 2n+1 台 server ,只要有 n+1 台(大多数)server 可用，整个系统保持可用

2.3、Zookeeper 的作用

Zookeeper 包含一个简单的原语集，分布式应用程序可以基于它实现命名服务、配置维护、集群选主等：

命名服务：注册节点信息，形成有层次的目录结构（类似 Java 的包名）。

配置维护：配置信息的统一管理和动态切换

集群选主：确保整个集群中只有一个主，其它为从。并且当主挂了后，可以自动选主

3、SolrCloud 概述

3.1、什么是 SolrCloud

单点的 Solr 服务的问题：

A：能存储的数据量有限，如果是海量数据，无法存储

B：容易出现单点故障

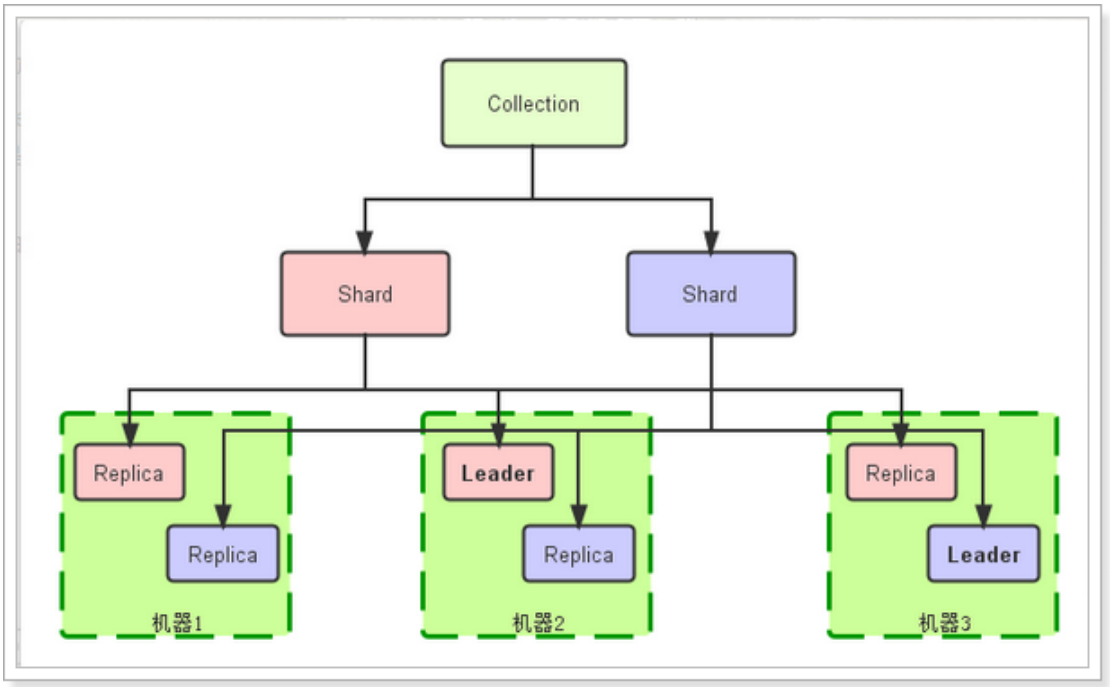
C：对高并发的处理能力差

如果我们的需要处理海量数据、应对高并发请求，并且让我们的服务实现高可用。那么就必须要搭建 SolrCloud 了。反之，如果数据量很少，请求并发低的情况下，是不需要 SolrCloud 的，单点 Solr 就够了。

SolrCloud(solr 云)是 Solr 提供的分布式搜索方案，可以解决海量数据的 分布式全文检索，因为搭建了集群，因此具备高可用的特性，同时对数据进行主从备份，避免了单点故障问题。可以做到数据的快速恢复。并且可以动态的添加新的节点，再对数据进行平衡。

3.2、SolrCloud 结构简介

为了实现海量数据的存储，我们会把索引进行分片（Shard），把分片后的数据存储到不同 Solr 节点。为了保证节点数据的高可用，避免单点故障，我们又会对每一个 Shard 进行复制，产生很多副本（Replicas），每一个副本对于一个 Solr 节点中的一个 core。用户访问的时候，可以访问任何一个会被自动分配到任何一个可用副本进行查询，这样就实现了负载均衡。



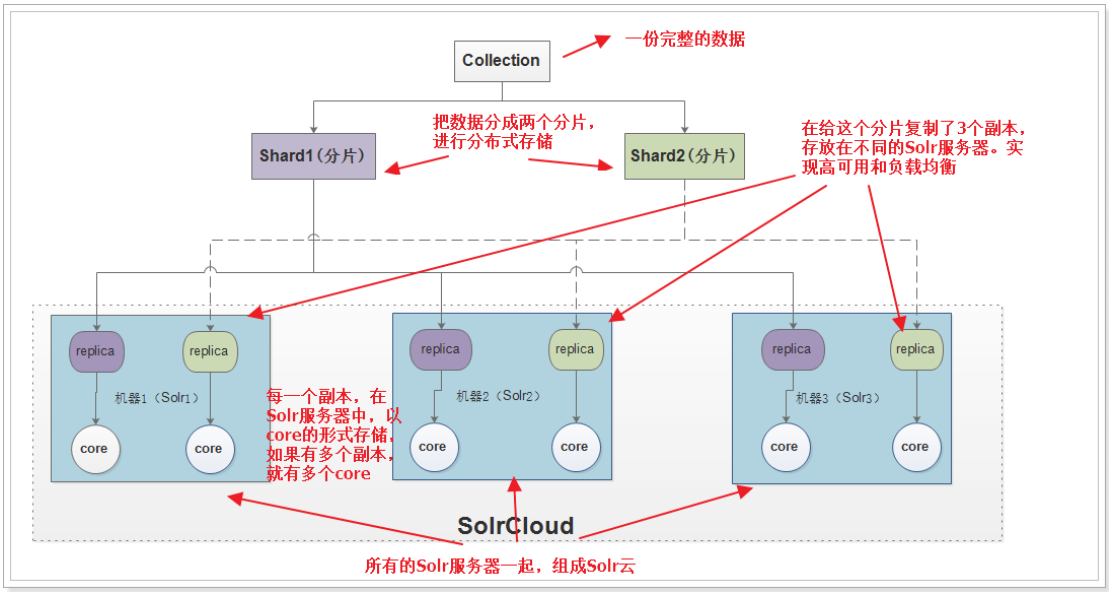
Collection：在 SolrCloud 集群中逻辑意义上的完整的索引。一般会包含多个 Shard（分片），如果大于 1 个分片，那么就是分布式存储。

Shard: Collection 的逻辑分片。每个 Shard 被化成一个或者多个 replicas（副本），通过选举确定哪个是 Leader（主），其它为从

Replica: Shard 的一个副本，存储在 Solr 集群的某一台机器中（就是一个节点），对应这台 Solr 的一个 Core。

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。



4、SolrCloud 的搭建

4.1、基本环境

我们需要三台服务器，也就是三台虚拟机。分别是：

192.168.8.10

192.168.8.11

192.168.8.12

每台及其上都需要部署以下环境：

JDK：基本 Java 运行环境

Tomcat：装载 Solr 服务

Solr-4.10.2：Solr 服务

Zookeeper：对 Solr 云进行管理

4.2、单机部署

4.2.1、Solr 安装

将 Solr 的工程放入 tomcat 的 webapps 目录下并进行解压：`unzip -oq solr.war -d solr`，上传依赖包到 solr/WEB-INF/下。

修改 tomcat 的 bin 目录下的 catalina.sh 文件，添加启动的参数，指向 solr 的索引文件夹。

启动 tomcat，访问：http://192.168.8.10:8080/solr 查看单点 Solr 部署情况。

4.2.2、zookeeper 安装

上传 zookeeper 的安装包

解压缩安装包

```
drwxr-xr-x. 12  501 games      4096 11月 11 08:23 zookeeper-3.4.5
-rw-r--r--.  1 root root   16402010 6月  15 12:20 zookeeper-3.4.5.tar.gz
```

修改配置文件

A：进入 zookeeper/conf 目录

B：复制模板文件

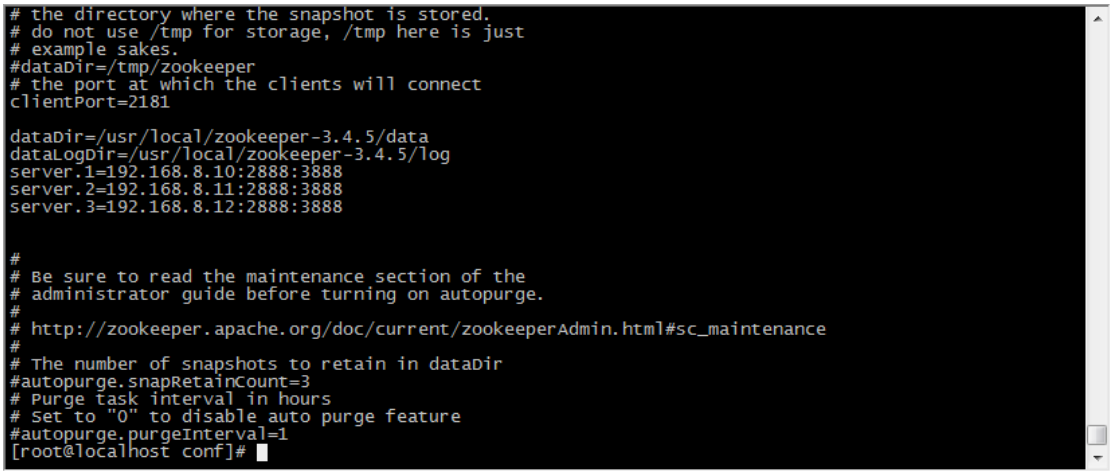
```
cp zoo_sample.cfg    zoo.cfg
```

C：修改配置文件信息 vi zoo.cfg，添加以下内容

要添加的内容：

```
dataDir=/usr/local/ zookeeper-3.4.5/data

dataLogDir=/usr/local/ zookeeper-3.4.5/log

server.1=192.168.8.10:2888:3888

server.2=192.168.8.11:2888:3888

server.3=192.168.8.12:2888:3888
```

模板文件中已经有一个 dataDir 参数，我们一定要把这个删除，或者在这个基础上修改




```
dataDir：数据目录
dataLogDir：日志目录
server.1=x.x.x.x:port1:port2 指定所有 zookeeper 的节点信息
server 后的数字是节点 ID，port1 是心跳端口，port2 是数据端口
```

在 zookeeper 中创建数据目录和日志目录

A：先进入 zookeeper 目录

B：创建目录

```
mkdir -m 755 data
```

```
mkdir -m 755 log
```

添加节点 ID 信息，进入 data 目录，创建文件 myid，并且写上 ID 信息：1，注意，其它节点的 ID 必须与配置文件中的 ID 一直，分别是 2 和 3。

配置 zookeeper 的环境变量，可以在任何位置使用 zookeeper 命令

A：vi /etc/profile(修改文件)

B：添加内容：

```
export ZOOKEEPER_HOME=/usr/local/zookeeper-3.4.5

export PATH=$PATH:$ZOOKEEPER_HOME/bin
```

C：重新编译文件：source /etc/profile

zookeeper 命令：

```
启动 zookeeper:    zkServer.sh start
停止 zookeeper:    zkServer.sh stop
查看状态:          zkServer.sh status
zkServer.sh start-foreground
```

4.3、集群部署

4.3.1、修改 tomcat 启动文件，添加 zookeeper 的地址信息

修改：tomcat 文件夹下的 bin 目录中的 catalina.sh 文件，添加以下信息：

```
export JAVA_OPTS="-Dsolr.solr.home=/usr/local/solr-4.10.2/example/solr
-DzkHost=192.168.8.10:2181,192.168.8.11:2181,192.168.8.12:2181"
```

这样 tomcat 启动后，solr 服务就可以到 zookeeper 中注册自己的信息，或者获取其它节点信息。

4.3.2、启动 zookeeper 集群

修改 102 和 103 机器中的 zookeeper 的节点 ID ,修改 zookeeper 目录中 data/myid 的内容，分别为 2 和 3。

分别启动三台虚拟机中的 zookeeper，使用 zookeeper 的客户端控制台，查看 zookeeper 信息。Zookeeper 提供了自己的客户端命令行工具，与 Linux 的命令非常相似。

```
A：启动客户端工具：zkCli.sh -server ip:port
B：显示根目录下、文件： ls / 使用 ls 命令来查看当前 ZooKeeper 中所包含的内容
C：显示根目录下、文件： ls2 / 查看当前节点数据并能看到更新次数等数据
D：创建文件，并设置初始内容： create /zk "test" 创建一个新的 znode 节点“ zk ”以及与它关联的字符串
E：获取文件内容： get /zk 确认 znode 是否包含我们所创建的字符串
F：修改文件内容： set /zk "zkbak" 对 zk 所关联的字符串进行设置
G：删除文件： delete /zk 将刚才创建的 znode 删除
H：退出客户端： quit
I：帮助命令： help
```

4.3.3、修改 Solr 配置文件，配置集群的中每台 Solr 服务的 IP 和端口

进入/usr/local/myapp/solr-4.10.2/example/solr 目录，修改 solr.xml 文件

```
<solrcloud>
    <str name="host">192.168.8.10</str>
    <int name="hostPort">8080</int>
    <str name="hostContext">${hostContext:solr}</str>
    <int name="zkClientTimeout">${zkClientTimeout:30000}</int>
    <bool
name="genericCoreNodeNames">${genericCoreNodeNames:true}</bool>
</solrcloud>
```

4.3.4、将 Solr 配置文件上传到 Zookeeper 中统一管理

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。以后无论创建任何的 core，本地的配置文件都没用了，使用的都是 zookeeper 的配置文件

执行下边的命令将/usr/local/myapp/solr-4.10.2/example/solr/collection1/conf/ 下的配置文件上传到 zookeeper：

```
sh /usr/local/solr-4.10.2/example/scripts/cloud-scripts/zkcli.sh -zkhost
192.168.8.10:2181,192.168.8.11:2181,192.168.8.12:2181 -cmd upconfig
-confdir /usr/local/solr-4.10.2/example/solr/collection1/conf/ -confname
solrconf
```

/usr/local/myapp/solr-4.10.2/example/scripts/cloud-scripts/zkcli.sh ： Solr 提

供的访问 Zookeeper 的脚本文件

-zkhost 192.168.8.10:2181,192.168.8.11:2181,192.168.8.12:2181： 指定 Zookeeper 的地址信息

-cmd upconfig： 指定操作的命令。这里是上传配置

-confdir /usr/local/myapp/solr-4.10.2/example/solr/collection1/conf/ ：指定要上传的配置文件目录，我们上传 Solr 的样例中的配置

-confname solrconf ：指定注册到 Zookeeper 中后配置文件目录名称

4.3.5、启动 SolrCloud

启动每一台服务器中的 Solr 服务，访问 SolrCloud，查看云的状态。

4.4、通过管理界面查看和操作 SolrCloud

4.4.1、SolrCloud 的操作命令：

Solr 采用的是类似 WebService 的 API 接口，采用 Http 方式进行访问，因此，其操作命令都是一些 URL 联接及对应参数

1、创建 core 命令

http://192.168.8.10:8080/solr/admin/collections?action=CREATE&name=myCollection2&numShards=2&replicationFactor=2&maxShardsPerNode=8&property.schema=schema.xml&property.config=solrconfig.xml

参数说明：

name：指明 collection 名称

numShards：指明分片数

replicationFactor：指明副本数

maxShardsPerNode： 每个节点最大分片数（默认为 1）

property.schema：指定使用的 schema.xml，这个文件必须在 zookeeper 上。

property.config：指定使用的 solrconfig.xml，这个文件必须在 zookeeper 上。

2、删除 Collection 命令

http://192.168.8.10:8080/solr/admin/collections?action=DELETE&name=collection1

3、查询所有的 Collection

http://192.168.8.10:8080/solr/admin/collections?action=LIST

4、显示集群的状态

http://192.168.8.10:8080/solr/admin/collections?action=CLUSTERSTATUS

5、分裂 shard

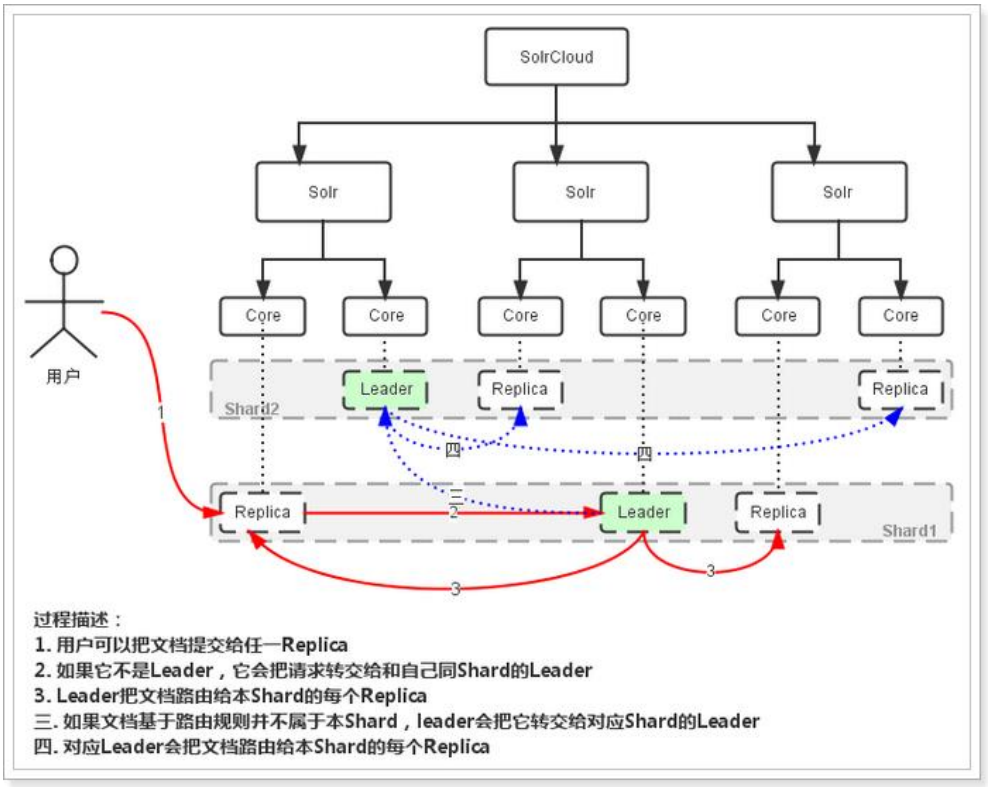
http://192.168.8.10:8080/solr/admin/collections?action=SPLITSHARD&collection=myCollection2&shard=shard2

6、删除 shard

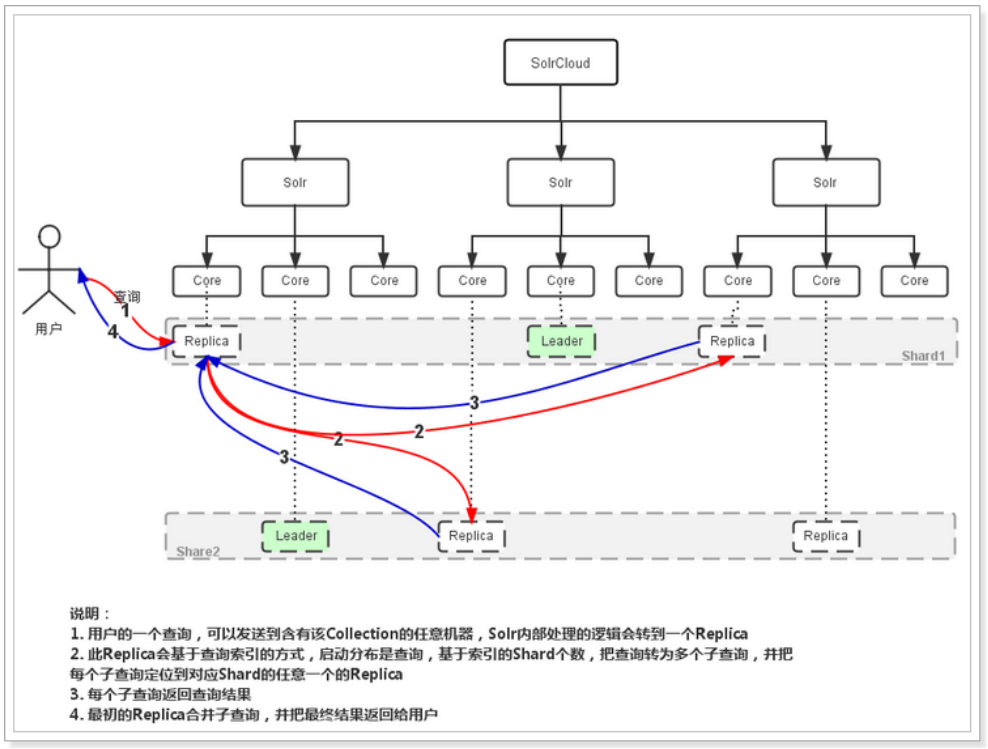
http://192.168.8.10:8080/solr/admin/collections?action=DELETESHARD&collection=myCollection2&shard=shard2

4.4.2、SolrCloud 测试

创建索引：在一台 Solr 上创建的索引，从其它 solr 服务上可以查询到



查询索引：从任意一台 Solr 上查询索引，选择任意的一个分片，都会返回一个完整的结果



5、使用SolrJ访问SolrCloud

与单机 Solr 相比，API 仅仅是在创建 SolrServer 时发生了改变，其它没有变化。单机采用的是：HttpSolrServer，SolrCloud 采用的是：CloudSolrServer。

5.1、添加或修改数据

```
@Test
    public void testWrite() throws Exception{
        // 创建SolrServer
        CloudSolrServer server = new
CloudSolrServer("192.168.8.10:2181,192.168.8.11:2181,192.168.8.12
:2181");
        // 指定要访问的Collection名称
        server.setDefaultCollection("collection1");

        // 创建Document对象
        SolrInputDocument document = new SolrInputDocument();
        // 添加字段
        document.addField("id", "1");
        document.addField("title", "solrcloud写入的数据");

        // 添加Document到Server
        server.add(document);
        // 提交
        server.commit();
    }
```

5.2、删除数据

```
@Test
    public void testDelete()throws Exception{
        // 创建SolrServer
        CloudSolrServer server = new
CloudSolrServer("192.168.8.10:2181,192.168.8.11:2181,192.168.8.12
:2181");
        // 指定要访问的Collection名称
        server.setDefaultCollection("collection1");
        // 根据ID删除
        server.deleteById("1");
        // 提交
        server.commit();
    }
```

5.3、查询数据

```
@Test
    public void testSearch() throws Exception {
        // 创建SolrServer
        CloudSolrServer server = new
CloudSolrServer("192.168.8.10:2181,192.168.8.11:2181,192.168.8.12
:2181");
        // 指定要访问的Collection名称
        server.setDefaultCollection("collection1");

        // 查找数据
        QueryResponse response = server.query(new
SolrQuery("title:solrcloud"));

        // 以document形式解析数据
        SolrDocumentList documentList = response.getResults();
        // 遍历
        for (SolrDocument solrDocument : documentList) {
            System.out.println(solrDocument.getFieldValue("id"));

    System.out.println(solrDocument.getFieldValue("title"));
        }
    }
```