哈尔滨工业大学计算机科学与技术学院

大作业文档说明

课程名称：数据结构与算法

课程类型：必修

大作业名称：hash一致性

项目成员：1170300520郭子阳、1170300503张越

1. **实验内容**：

Hash一致性问题

1. **算法实现原理：**

先构造一个长度为2^32的整数环（这个环被称为一致性Hash环），根据节点名称的Hash值（其分布为[0, 232-1]）将服务器节点放置在这个Hash环上，然后根据数据的Key值计算得到其Hash值（其分布也为[0, 232-1]），接着在Hash环上顺时针查找距离这个Key值的Hash值最近的服务器节点，完成Key到服务器的映射查找。

1. **算法改进：使用虚拟节点改善一致性Hash算法：**

引入虚拟节点解决负载不均的问题，基本原理是：将一个物理节点拆分为多个虚拟节点，并且同一个物理节点的虚拟节点尽量均匀分布在Hash环上。采取这样的方式，就可以有效地解决增加或减少节点时候的负载不均衡的问题。

1. **使用数据类型说明：**

//真实节点

private static LinkedList<String> realNodes = new LinkedList<>();

//虚拟节点

private static TreeMap<Integer, String> virtualNodes = new TreeMap<>();

//每个真实节点对应的虚拟节点的个数

public static final int NUMBER\_OF\_VIRTUAL\_NODES = 10;

final int p = 16777619; //FNV用于散列的质数

int hash = (int)2166136261L; //初始的哈希值

1. **测试结果：**



1. **附源代码：**

import java.util.LinkedList;

import java.util.Scanner;

import java.util.SortedMap;

import java.util.TreeMap;

public class Main {

private static Scanner scanner = new Scanner(System.in);

private static LinkedList<String> realNodes = new LinkedList<>();

private static TreeMap<Integer, String> virtualNodes = new TreeMap<>();

public static final int NUMBER\_OF\_VIRTUAL\_NODES = 10;

public static void main(String[] args) {

addRawNodes();

while(true) {

menu();

}

}

private static void addRawNodes() {

System.out.print("请输入初始服务器数量：");

int number = scanner.nextInt();

for(int i = 0; i < number; i ++) {

String realIp = generateIp();

realNodes.add(realIp);

}

for(String ip : realNodes) {

addVirtualNode(ip);

System.out.println("添加服务器" + ip + "成功！");

}

}

private static String generateIp() {

StringBuilder ip = new StringBuilder();

for(int i = 0; i < 3; i ++) {

ip.append((int)(Math.random() \* 255 + 1) + ".");

}

ip.append((int)(Math.random() \* 255 + 1));

return ip.toString();

}

private static void addVirtualNode(String ip) {

for(int i = 0; i < NUMBER\_OF\_VIRTUAL\_NODES; i ++) {

String virtualNodeName = ip + "##VN" + String.valueOf(i);

int hash = getHash(virtualNodeName);

virtualNodes.put(hash, virtualNodeName);

}

}

/\*\*

\* 使用FNV1\_32\_HASH算法计算Hash值

\*/

private static int getHash(String string) {

final int p = 16777619;

int hash = (int)2166136261L;

for(int i = 0; i < string.length(); i ++) {

hash = (hash ^ string.charAt(i)) \* p;

}

hash += hash << 13;

hash ^= hash >> 7;

hash += hash << 3;

hash ^= hash >> 17;

hash += hash << 5;

if(hash < 0) {

hash = Math.abs(hash);

}

return hash;

}

private static void menu() {

System.out.println();

System.out.println();

System.out.println("1. 服务器路由查询");

System.out.println("2. 增加服务器");

System.out.println("3. 删除服务器");

System.out.println("4. 退出");

System.out.print("请输入选择：");

int choice = scanner.nextInt();

System.out.println();

System.out.println();

switch(choice) {

case 1:

searchNode();

break;

case 2:

System.out.print("请输入待添加的服务器ip：");

scanner = new Scanner(System.in);

String ip = scanner.nextLine();

if(realNodes.contains(ip)) {

System.out.println("该服务器已被添加，请勿重复添加！");

} else {

realNodes.add(ip);

addVirtualNode(ip);

System.out.println("添加服务器" + ip + "成功！");

}

break;

case 3:

System.out.print("请输入待删除的服务器ip：");

scanner = new Scanner(System.in);

String deleteIp = scanner.nextLine();

if(!realNodes.contains(deleteIp)) {

System.out.println("该服务器不在列表中，无法删除！");

} else {

deleteServer(deleteIp);

System.out.println("删除服务器" + deleteIp + "完成！");

}

break;

case 4:

System.exit(0);

break;

default:

System.out.println("选择错误，请重新选择！");

break;

}

}

private static void searchNode() {

if(virtualNodes.isEmpty()) {

System.out.println("当前无任何服务器，请添加一些。");

return;

}

System.out.print("请输入客户端ip：");

scanner = new Scanner(System.in);

String ip = scanner.nextLine();

System.out.println(ip + "的hash值为" + getHash(ip) + "，被路由到服务器" + getServer(ip));

}

private static String getServer(String ip) {

int hash = getHash(ip);

SortedMap<Integer, String> subMap = virtualNodes.tailMap(hash);

Integer i = 0;

if(!subMap.isEmpty()) {

i = subMap.firstKey();

}

String virtualNode = subMap.get(i);

return virtualNode.substring(0, virtualNode.indexOf("##"));

}

private static void deleteServer(String ip) {

realNodes.remove(ip);

for(int i = 0; i < NUMBER\_OF\_VIRTUAL\_NODES; i ++) {

String virtualNodeName = ip + "##VN" + String.valueOf(i);

int hash = getHash(virtualNodeName);

virtualNodes.remove(hash);

}

}

}