Journal of Computer Applications
计算机应用,2012,32(7):1920-1923,1977
ISSN 1001-9081
CODEN JYIIDU
2012-07-01
http://www.joca.cn

文章编号:1001-9081(2012)07-1920-04
doi:10.3724/SP.J.1087.2012.01920

基于 HBase 的大规模无线传感网络数据存储系统

陈庆奎,周利珍*
(上海理工大学 光电信息与计算机工程学院,上海 200093)
(*通信作者电子邮箱 zhoulizhenzhen@gmail.com)

摘　要:无线传感网络(WSN)存在分布的跨区域性,随着无线传感网络的扩张,传感器数目增多,将产生大规模的传感数据。针对存储大规模无线传感网络数据的问题,提出了一个两层分布式存储架构,使用分布式数据库 HBase 存储跨区域的无线传感网络数据和全局数据存储管理目录,实现一个近实时的存储系统。实验结果证明,该系统有良好的扩展性、存储和查询效率。

关键词:云计算;HBase;分布式存储;无线传感网络;物联网
中图分类号:TP274;TP393.07　　**文献标志码**:A

HBase-based storage system for large-scale data in wireless sensor network

CHEN Qing-kui, ZHOU Li-zhen*
(*School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China*)

Abstract: Wireless Sensor Network (WSN) spreads widely, and with the expansion of WSNs there will be a large number of sensors which produce massive sensor data. To store the data from large-scale WSNs, this paper proposed a two-tier storage architecture based on HBase storing sensor data from different regions and global data management directory, which achieved a near real-time storage system. The experimental results show that this system with high scalability, storage and query efficiency can solve the massive sensor data storage.

Key words: cloud computing; HBase; distributed storage; Wireless Sensor Network (WSN); Internet of Things (IoT)

0　引言

随着物联网[1-3]在中国的展开,信息技术界面临海量信息存储[4]问题。无线传感网[5]作为物联网末端信息存储,分布区域越来越广,规模越来越大,产生信息量飞速增长。举例来说,若传感器采样频率为 5 s,那么一天一个传感器采样数据量有 17 280 条,1 000 个传感器一天的采样数据量就有近 2 千万条,一年的数据量有 63 亿条。海量信息存储的方法很多,Oracle 和 EMC 提供了存储系统及硬件,比较经典的是 RAC(Real Application Cluster)[6],RAC 已经在工业和信息界得到广泛应用。然而这些数据库都是基于关系和对象模型的,对复杂数据存储有较高的表现能力,但是存储代价及系统消耗比较大。近年来,随着云计算技术的发展,学术界对海量数据存储进行了深入探讨;以 Hadoop[7]集群为代表的分布式计算技术以及 HBase[8]存储系统(以列存储数据)。HBase 是 BigTable[9]的开源实现。传感器数据类型单一,是按时间分布的单列数据,HBase 是以列存储数据,在处理传感器这种单列类型数据上比关系数据库有更好的效率。虽然无线传感网络(Wireless Sensor Network, WSN)对数据的实时要求比较高,然而实际应用中不是所有无线传感网络都有严格的实时性要求,比如农业生产。此外,由于部署的传感网络多,成千上万的传感器数据会并发同时被处理,而 HBase 集群就是处理大量的并发数据操作。因此 HBase 是适合无线传感网络的信息存储。根据农业传感器分布广、数目多的特点,用 Hadoop 构建集群存储管理系统,通过 HBase 分布式存储传感器数据。HBase 具有很好的扩展性和负载均衡性,可以任意添加和减少集群节点个数并可以保持负载均衡。为解决跨区域的数据存储问题,设计 2 层存储架构,根据区域构建存储集群,每个区域都有一个 HBase 集群存储数据,在所有区域之上设计一个全局数据存储,用来保存各个区域存储信息,不保存传感器采集的数据。

1　相关原理及架构设计

1.1　HBase 介绍

HBase 是分布式、面向列的存储系统,提供实时读写和随机访问大数据集。HBase 自动把表横切成不同的区域(region),每个区域包含表的所有行的一个子集。HBase 由一个主节点(Master)协调一个或多个区域服务器(region server)组成。HBase 主节点负责引导初始安装、分配区域给区域服务器,恢复区域服务器的故障。主节点负载较轻。区域服务器负责 0 到多个区域,响应客户端的读写请求。HBase 的实现依赖于 Zookeeper[10]来协调管理,Zookeeper 负责选取一个节点为 Master,剩下的节点为 region server。

HBase 表由行和列组成。表的单元格是行和列坐标的交集,它们是有版本号的。在默认情况下,版本号是在单元格插入时由 HBase 自动分配的时间戳(Time Stamp)。表的单元格内容是一个未解释的字节数组。表行的键也是字节数组,可以是 string 或 long 类型的二进制表示,甚至是序列化的数据结构。表的行键是表的主键,在默认情况下按照字典序升序排列。每行的列被分组,形成列簇(column families),所有的

收稿日期:2011-12-13;修回日期:2012-02-08。　基金项目:国家自然科学基金资助项目(60970012;61003031);上海信息技术领域重点科技攻关项目(09511501000);上海重点科技项目(09220502800);上海市重点学科建设项目(S30501)。

作者简介:陈庆奎(1966-),男,上海人,教授,博士生导师,主要研究方向:集群计算、并行计算;　周利珍(1987-),女,安徽安庆人,硕士研究生,主要研究方向:物联网。

列簇成员都有相同的前缀,组内列通过标识符(qualifier)区分,因此,每列就表示为 column family:qualifier。

1.2 大规模传感网络

无线传感网络由一组传感器和汇聚节点组成,它们之间通过无线网络通信。大量传感器节点部署在监测区域内部或附近,能够通过自组织方式构成网络。一个无线传感网络有一个或多个汇聚节点专门收集所有传感器节点采集的数据,汇聚节点通过网关实现数据和应用程序的交互。无线传感网络的数据存储分为内部数据存储和外部应用程序存储,本文讨论数据从传感器采集经汇聚节点到外部的存问题。

传感网络的大规模性包括两方面含义:传感器分布在很大的地理区域内,如在原始大森林采用传感器网络进行森林防火和环境监测,需要部署大量的传感器节点;另一方面,传感器节点部署很密集,在一个面积不是很大的空间内,密集部署了大量传感器节点。

为了获取精确信息,在监测区域通常部署大量传感器节点,传感器节点数量可能达到成千上万,甚至更多。在实际应用中,传感器的分布广、规模大,部署是有区域性的,一些传感器聚集在一个区域中。

1.3 系统架构

根据传感网络分布的区域性,设计物联网数据存储架构如图 1 所示。它包含三个层次,如下所示。

1)网关层。传感器数据的接入层,不同传感器类型都有与之对应的网关来接收并处理传感器数据。传感器采集的数据通过网关被应用程序获取。

2)LM(Local Manager)层。管理传感数据的存储,从网关获取传感器采集数据,并实时存储这些数据。

3)GM(Global Manager)层。管理并记录全局信息,分配传感器数据存放位置。

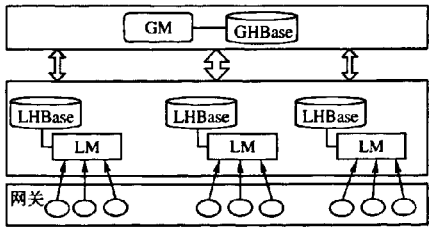

图 1　系统架构

由于大规模传感器分布存在区域性,如果将所有区域传感器采集数据存放到一个 LHBase(Local HBase)集群数据库必然导致网络资源消耗大,费时长,因此,将各个区域采集的数据存放在区域数据库服务器有利于减少网络资源消耗,提高数据存储和访问的实时性。因此,本文将网关按区域划分,每个区域都有 LHBase 用来存放网关收集的实时传感器采集数据,GHBase(Global HBase)中保存全局信息,记录所有 LHBase、LM、网关及传感器等信息。当新添加一个网关时,首先通过本地 LM 向 GM 申请添加新的网关,GM 为网关和传感器分配一个唯一的标识符,为每个传感器指定存储表名和表结构,GM 保存这些信息在 GHBase,维持一个全局信息。LM 从 GM 获取数据存放表名和表结构后,写入数据到 LHBase。

2 系统实现

2.1 通信机制及数据传输

在如图 1 的系统架构中,存在着 3 种通信关系,如下。

1)网关到 LM。传感器数据传输,数据实时写入 LHBase。

2)LM 到 GM。新添加、更新或删除网关或传感器节点时,申请注册、更新或删除注册记录。

3)GM 到 LM。响应用户查询请求,由于传感器数据保存在 LHBase,而 GHBase 记录所有 LHBase 中保存数据的数据信息。因此,首先 GM 查询 GHBase 获取所查数据所在 LHBase 地址,然后与 LM 通信获取数据返回给用户。

本文选用 RPC(Remote Procedure Call)[11] 来实现这些通信,RPC 的实现有很多种,Hadoop 中的 RPC 是 Hadoop 系统内部的通信机制,采用客户端/服务器的模式,它的协议是一个 Java 接口,服务端实现这个接口,客户端获取服务端的实例,调用接口在服务端的实现。通过接口的参数和返回值实现数据的传输。协议规定所有参数和返回类型都必须是 Java 原生数据类型(boolean, byte, char, short, int, long, float, double, void),String,Writable。参数和返回类型在调用和响应之前都被会序列化成二进制,提高传输效率。

从前文描述的 RPC 通信可知,支持的传输数据格式有限,因此,需要将某些复杂传输对象转化为可传输类型。JSON(JavaScript Object Notation)是一种轻量级的数据交换格式,把复杂对象转化为 String 类型,易于机器解析和生成。Google 的 Gson[12] 提供了两种方法:1)toJson(),将 Java 对象转化为 JSON 格式;2)fromJson(),将 JSON 格式转化为 Java 对象。例如一个 Sensor 对象,包含的属性有 id = 01, city = shanghai, district = yangpu, sensorType = light, dataType = int, sampleRate = 60,转换成 JSON 格式就是:{"id":"01","city":"shanghai","district":"yangpu","sensorType":"light","dataType":"int","sampleRate":"60"}。

2.2 存储表结构设计

本文设计的存储架构需要保存 2 类数据表:全局信息表和传感器数据表。其中全局信息表包括保存所有已经配置的 LHBase 表、来自各个区域的网关的信息表、所有传感器信息 Sensor 表。这些表都只有一个 column family,qualify 就是存放属性值。

1)LHBase 表。

LHBase(info:gid, info:ip, info:port, info:region servers, info:avaCapacity, info:location)

其中:gid 表示唯一标示符,ip 表示 Master IP 地址,port 表示 Master 端口,region server 表示分配的 HBase 节点,avaCapacity 表示可用存储空间,location 表示所在地理位置。

2)网关表(info:gid, info:name, info:location, info:LHBase gid)。

其中:gid 表示唯一标示符,name 表示网关名称,location 表示所在地理位置,LHBase gid 表示网关数据存放的 LHBase 唯一标示符。

3)Sensor 表(info:gid, info:id, info:location, info:sensor type, info:data type, info:sample rate, info:storage table, info:netgw_gid)。

其中:gid 表示唯一标示符,id 表示传感器节点本地编号,location 表示所在地理位置,sensor type 表示传感器类型,data type 表示传感器采集数据类型,sample rate 表示采样频率,storage table 表示存放采集数据的表名,netgw_gid 表示其接入网关的标示符。

4)每个传感器都有一个表用来存放采集的数据,称为传感器数据表(info:time, info:data),分别表示采样时间和采样

值。

这4种表在数据库中的分布如图2、3所示，HBase集群节点实现分布式存储服务，GHBase中数据表LHBase表、网关表、Sensor表分布在所有region server节点上，而LHBase中所有传感器采集的数据存放表也分布在所有region server节点上。

图2 表在GHBase中的分布情况 图3 表在LHBase中的分布情况

2.3 无线传感网络数据写入过程

HBase中各个表的操作需要首先获取HTable类的对象，然后使用put、get方法来完成插入和读取数据操作。使用HBaseAdmin对象来完成对表的新建、删除等操作。数据写入是指传感器采集的数据汇集到网关后，需要经过LM的写操作写入到LHBase指定的表中。每个传感器需要分配一个表来存放数据，表的分配是由LM向GM申请的，申请表名和表结构。LM接收到来自各个网关的传感器数据data后，需要将这些数据存放到GM指定的表中，需要经过以下的步骤，如图4所示。

1)判断是否是新加入的网关，若是，转步骤2)；若否，转步骤3)。

2)LM通过RPC调用GM的regist(String TYPE. Net_Gateway,String netGatewayToJson)获取网关gid并保存。

3)判断是否是新的传感器节点，若是，转步骤4)；若否，转步骤5)。

4)LM通过RPC调用GM的regist(String TYPE. Sensor,String sensorToJson)获取传感器gid、存储表名和表结构并保存。

5)LM按照从GM获取的表名和结构写入数据。首先需要判断表是否存在HBaseAdmin. tableExists(tableName)，如果表存在，根据column family和qualify新建一个Put对象，Put. add(column famiy,qualify,data)，然后获取HTable的实例，写入到表中HTable. put(Put)。

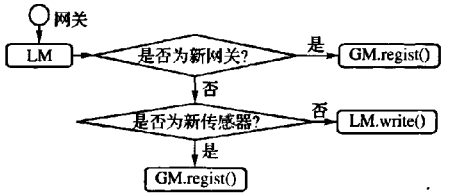

图4 数据写入过程

正如上文描述，GM需要为新添加的数据库LHBase、网关和传感器节点分配gid，并保存这些信息，过程如下：

1)获取regist(String type,String objectToJson)，type表示LHBase、NetGateway、Sensor，根据type，还原传入的JSON字符串为对象；

2)为该对象生成唯一标识符gid；

3)将新添加的对象插入到表中。

2.4 无线传感网络数据访问过程

数据访问是指客户端访问某个地区的传感器采集数据信息，GM维护着全局信息，客户端需要通过GM查询需要访问的数据所在数据库地址，然后GM到对应的数据库服务器上提交请求获取数据并将数据返回，如图5，具体过程如下：

1)客户端首先获取GM的一个实例；

2)通过这个实例RPC调用GM的query(String location,String id)查询sensor type,data type,storage table,netgw_gid以及传感器gid；

3)GM将1)中获得netgw_gid查询表NetGateway，获取LHBase_gid；

4)根据2)中获得的LHBase_gid查询表LHBase，获得ip,port,region servers；

5)GM从3)中获得了LHBase的信息，从1)中获得存储表信息，获得LM的一个实例；

6)通过LM的实例就可以通过RPC调用LM的query(String table,String gid,String sensor_type,String data_type)获得数据，但这个数据是byte类型的，需要根据data_type转换成相应的数据类型。

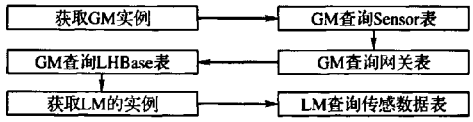

图5 数据查询步骤

2.5 性能分析

HBase是面向列的数据库，使用行作为主键，行在数据写入时默认按照字典序排列，其索引也是在行上建立的。

而该索引是按B^+树存储的，因此，查询的时间复杂度是$\log N$。由于HBase的HMaster负责分配regions给HRegionServer(HRS)，HRS才真正存储数据。因此查询时，客户端首先要请求HMaster找到存放数据的HRS，然后再由这个HRS提供查询结果。如图6所示。

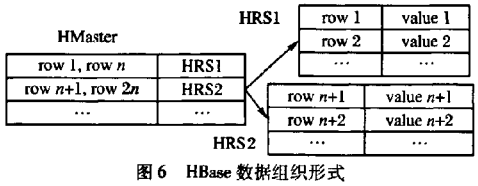

图6 HBase数据组织形式

因此，查询的时间由两部分组成：HMaster查找行所在HRS的时间，设为t_{master}；以及HRS查找行所对应的值的时间，设为t_{HRS}。总时间$t = t_{master} + t_{HRS}$。设插入的行为n，每个region由m行组成，HRS的数量为p，则HMaster维持的目录长度为n/m，HRS保存的数据记录数有$\frac{n}{m \times p}$。于是，$t = t_{master} + t_{HRS} = \log\left(\frac{n}{m}\right) + \log\left(\frac{n}{m \times p}\right) < \log(n)$。

由此可见，HBase大大降低了查询时间。

3 实验环境及实验结果

3.1 实验参数信息

配置两个Hadoop集群，它们的参数都是：5个DataNode节点数目，1个NameNode节点，1个Jobtracker节点，1个Master节点；每个节点处理器分别为Intel Core 2 Duo CPU E7300 2.66 GHz，2 GB内存，320 GB硬盘，操作系统为Ubuntu 10.04，JDK版本为Sun JDK 1.6.0_24，使用Hadoop版本为hadoop-0.20.2，设置数据块为64 MB，副本数是3。使用HBase的版本是hbase-0.20.4，使用Zookeeper的版本是zookeeper-3.3.3。这两个Hadoop集群分别模拟区域存储服务集群和全局存储服务集群。以其中一个集群配置为例，表1列出了配置主机IP地址、主机名和担任角色。

表 1　集群节点及角色

IP 地址	主机名	别名	功能
192.168.6.20	ubuntu20	NameNode	Master
192.168.6.21	ubuntu21	JobTracker	Master
192.168.6.22	ubuntu22	DataNode22	region server
192.168.6.23	ubuntu23	DataNode23	region server
192.168.6.24	ubuntu24	DataNode24	region server
192.168.6.25	ubuntu25	DataNode25	region server
192.168.6.26	ubuntu26	DataNode26	region server

3.1.1　Hadoop 集群配置

配置 Hadoop 集群,需要安装 Java,SSH,hadoop-0.20.2 在上面所有机器上。具体安装过程如下。

1)安装 Java。在 ubuntu 命令行终端输入 sudo apt-get install sun-java6-jre 和 sudo apt-get install sun-java6-jdk,完成 Java 安装。

2)安装 SSH 是为了能够使得远程机器能够无密码通信,在 Hadoop 集群中,需要 NameNode 访问 JobTracker、DataNode,JobTracker 访问 DataNode,因此在它们之间需要配置 SSH。首先是各台主机安装 SSH:命令行终端输入 sudo apt-get install ssh、sudo apt-get install rsync;然后在 NameNode 和 JobTracker 上生成密钥:ssh-keygen -t dsa -P'' -f ~/.ssh/id_dsa,并把其中的公钥存放在 DataNode 主机上 cat id_dsa.pub | ssh hadoop@DataNode 的 IP 'cat >> .ssh/authorized_keys'。

3)解压 hadoop-0.20.2.tar.gz 到各个主机/home 文件夹下,修改 conf 文件夹的 6 个文件:core-site.xml、hdfs-site.xml、mapred-site.xml、hadoop-env.sh、masters、slaves 文件。修改 core-site.xml 的属性 fs.default.name 值为 hdfs://NameNode:9000。修改 hdfs-site.xml 的属性 dfs.name.dir 的值为/home/hadoop/dfs/name。修改 mapred-site.xml 的属性 mapred.job.tracker 的值为 JobTracker:9001。修改 hadoop-env.sh,添加

```
export JAVA_HOME = /usr/lib/jvm/java-6-sun
export HADOOP_LOG_DIR = /home/hadoop/hadoop_logs
export HADOOP_SLAVE_SLEEP = 0.1
export HADOOP_MASTER = namenode: /home/$USER/
  hadoop-0.20.2/
```

修改 masters 文件添加一行 JobTracker。修改 slaves 文件添加 5 行,DataNode22、DataNode23、DataNode24、DataNode25、DataNode26。

3.1.2　HBase 集群的配置

配置好 Hadoop 集群后,就可以配置 HBase 集群了,需要在其集群节点如 Master、region server 上配置 hbase-0.20.4 和 zookeeper-3.3.3。

1)解压 zookeeper-3.3.3.tar.gz 到/home 文件夹下,修改 conf/zoo.cfg 文件,添加

```
tickTime = 2000
initLimit = 10
syncLimit = 5
dataDir = /home/hadoop/zookeeper-3.3.3/var/zookeeper/data
clientPort = 2181
server.21 = ubuntu21: 2888: 3888
server.22 = ubuntu22: 2888: 3888
server.23 = ubuntu23: 2888: 3888
server.24 = ubuntu24: 2888: 3888
server.25 = ubuntu25: 2888: 3888
server.26 = ubuntu26: 2888: 3888
```

2)解压 hbase-0.20.4.tar.gz 到/home 文件夹下,修改 hdfs-site.xml 的属性 hbase.rootdir 值为 hdfs://NameNode:9000/hbase,修改属性 hbase.zookeeper.quorum 的值为 ubuntu22,ubuntu23,ubuntu24,ubuntu25,ubuntu26。

修改 hbase-env.sh,添加:

```
export HBASE_MANAGES_ZK = true
export HBASE_CLASSPATH = /home/hadoop/hadoop-0.20.2/conf
export JAVA_HOME = /usr/lib/jvm/java-6-sun
```

3)将 hbase-site.xml 复制到/home/hadoop/hadoop-0.20.2/conf 文件夹下,完成 HBase 的安装配置。启动时,需要先启动 Hadoop 集群,然后启动 HBase 集群。

3.2　实验结果

为了验证 HBase 的存储和查询效率以及可扩展性,使用 300 个传感器节点,比较单机和配置 5 个节点的集群的写入和读取时间。图 7 显示了单机和集群与传感器节点个数的写入时间关系。从图中可以看出,集群写入时间明显低于单机写入时间,具有高效的写入性。图 8 显示了单机和集群与传感器节点个数的读取时间关系。从图中可以看出,随着传感器数目的增加,集群读取时间明显低于单机读取时间,具有高效的读取性。图 9 显示了集群的扩展性,读取 300 个传感器数据的时间随着集群节点数目的增加而降低,验证了集群的扩展性比较好。

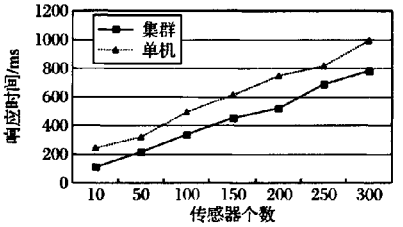

图 7　传感器个数与写入时间关系比较

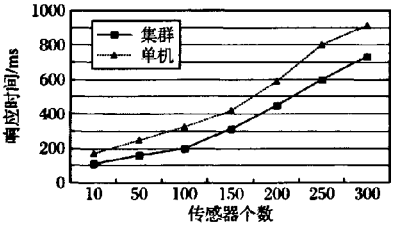

图 8　传感器个数与读取时间关系比较

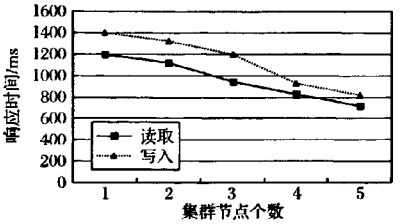

图 9　集群节点个数与写入、查询时间关系

4　结语

本文主要分析了无线传感网络的大规模数据的存储问题,提出一个基于 HBase 的分层存储架构,安装配置多个区域数据存储服务集群 LHBase,每个 LHBase 负责存储其区域范围内的传感器数据。GHBase 保存全局信息。并通过实验验证了 HBase 具有良好的可扩展性、高效的存储效率和访问效率。

(下转第 1977 页)

其他 6 个数据集上本文算法的聚类误差平方和与聚类有效性指标 XB 均优于传统 K-medoids 算法 PAM 和快速 K-medoids 算法。从聚类时间来看,本文算法绝对地优于传统 PAM 算法,与快速 K-medoids 算法在 Iris、Inosphere 两个数据集上持平,然而在其他数据集上的运行时间均比快速 K-medoids 算法短。

表 5　三种算法在 UCI 数据集上的聚类效果

数据集	聚类误差平方和			聚类有效性指标 XB			聚类时间/s		
	PAM 算法	快速算法	本文算法	PAM 算法	快速算法	本文算法	PAM 算法	快速算法	本文算法
D1	106.57	84.68	83.96	0.66450	0.2809	0.1878	0.163	0.156	0.156
D2	2.5501E+06	2.3889E+06	2.3889E+06	0.17810	0.1773	0.1773	0.192	0.188	0.187
D3	352	324	315	0.66340	0.4055	0.4468	0.087	0.063	0.062
D4	3.4223E+04	4.4537E+04	3.1243E+04	0.39840	0.8177	0.3284	0.319	0.313	0.298
D5	2.8896E+03	2.4415E+03	2.4415E+03	0.65440	0.6500	0.6500	0.397	0.343	0.343
D6	7.8583E+07	7.8148E+07	7.8148E+07	0.09350	0.0805	0.0805	0.649	0.625	0.594
D7	5.5838E+06	5.6279E+06	5.4327E+06	0.17150	0.1722	0.1652	0.823	0.797	0.765
D8	58.774	57.452	54.629	2.18615	1.6832	0.9488	0.609	0.594	0.469
D9	1.8930E+07	1.3707E+07	1.3251E+07	1.46050	0.9229	0.6389	1.569	1.203	1.061

以上实验结果表明,本文提出的基于粒计算的 K-medoids 聚类算法能够得到更优的初始聚类中心,聚类结果更稳定,具有更好的聚类性能。聚类性能优于其他两种 K-medoids 算法。

4　结语

本文将粒计算引入 K-medoids 聚类算法,定义基于粒度的样本相似度函数,基于等价关系的划分产生粒子,根据粒子的势定义粒子密度,选择位于密度较大的前 K 个粒子中心的样本作为初始聚类中心,改进了传统 K-medoids 聚类算法 PAM 的初始聚类中心随机选取以及 Park 等的快速 K-medoids 聚类算法在选取初始聚类中心时,有可能使不同类簇的初始中心位于同一类簇的潜在缺陷。人工模拟数据集和 UCI 机器学习数据库数据集实验共同证明,本文算法优于传统 K-medoids 聚类算法 PAM 和 Park 等的快速 K-medoids 聚类算法。本文算法具有更稳定、更好的聚类结果,适用于大数据集。

参考文献:

[1] 朱明. 数据挖掘[M]. 合肥: 中国科学技术大学出版社, 2002: 208 - 221.

[2] KAUFMAN L, ROUSSEEUW P J. Finding groups in data: an introduction to cluster analysis [M]. New York: Wiley, 1990: 126 - 163.

[3] PARK H S, JUN C H. A simple and fast algorithm for K-medoids clustering [J]. Expert Systems with Applications, 2009, 36(2): 3336 - 3341.

[4] 王国胤, 张清华, 胡军. 粒计算研究综述[J]. 智能系统学报, 2007, 2(6): 9 - 26.

[5] ZADEH L A. Fuzzy sets and information granularity [M]// Fuzzy Sets, Fuzzy Logic and Fuzzy Systems. River Edge, NJ: Word Scientific, 1996: 433 - 448.

[6] 王伦文. 聚类的粒度分析[J]. 计算机工程与应用, 2006, 42(5): 29 - 31.

[7] 卜东波, 白硕, 李国杰. 聚类/分类中的粒度原理[J]. 计算机学报, 2002, 25(8): 810 - 815.

[8] DING S F, XU L, ZHU H, *et al*. Research and progress of cluster algorithms based on granular computing [J]. International Journal of Digital Content Technology and its Applications, 2010, 4(5): 96 - 104.

[9] 夏宁霞, 苏一丹, 覃希. 一种高效的 K-medoids 聚类算法[J]. 计算机应用研究, 2010, 27(12): 4517 - 4519.

[10] 苗夺谦, 王国胤, 刘清, 等. 粒计算: 过去、现在与展望[M]. 北京: 科学出版社, 2007: 143 - 144.

[11] 王国胤, 安秋生, 沈钧毅. 基于信息粒度与 rough 集的聚类方法研究[J]. 模式识别与人工智能, 2003, 16(4): 412 - 417.

[12] XIE X L, BENI G. A validity measure for fuzzy clustering [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1991, 13(8): 841 - 847.

[13] FRANK A, ASUNCION A. UCI machine learning repository [EB/OL]. [2011 - 11 - 02]. http://archive.ics.uci.edu/ml.

(上接第 1923 页)

参考文献:

[1] International Telecommunication Union. ITU Internet Reports 2005: The Internet of Things [R]. UIT, 2005.

[2] RFID WORKING GROUP. Internet of Things in 2020: Roadmap for the future [EB/OL]. [2011 - 05 - 12]. http://www.smart-systems-integration.org/public/internet-of-things.

[3] CONTI J P. The Internet of things [J]. IET Communications Engineer, 2006, 4(6): 20 - 25.

[4] CHAIKEN R, JENKINS B, LARSON P-A, *et al*. SCOPE: Easy and efficient parallel processing of massive data sets [J]. Proceedings of the VLDB Endowment, 2008, 1(2): 1265 - 1276.

[5] YICK J, MUKHERJEE B, GHOSAL D. Wireless sensor network survey [J]. Computer Networks, 2008, 52(12): 2292 - 2330.

[6] Oracle Corporation. Oracle real application clusters [EB/OL]. [2011 - 07 - 01]. http://www.oracle.com/technology/products/database/clustering.

[7] The Apache Software Foundation. Hadoop [EB/OL]. [2011 - 08 - 12]. http://hadoop.apache.org/.

[8] The Apache Software Foundation. Apache HBase [EB/OL]. [2011 - 08 - 04]. http://hadoop.apache.org/hbase/.

[9] CHANG F, DEAN J, GHEMAWAT S, *et al*. BigTable: A distributed storage system for structured data [C]// Proceedings of the 7th Symposium on Operating Systems Design and Implementation. Berkeley, CA: USENIX Association, 2006: 15 - 15.

[10] The Apache Software Foundation. Apache ZooKeeper [EB/OL]. [2010 - 11 - 20]. http://zookeeper.apache.org/.

[11] MARSHALL D, Remote Procedure Calls (RPC) [EB/OL]. [2011 - 05 - 01]. http://www.cs.cf.ac.uk/Dave/C/node33.html.

[12] Google Project Hosting. Google-gson [EB/OL]. [2011 - 08 - 01]. http://code.google.com/p/google-gson/.

基于HBase的大规模无线传感网络数据存储系统

作者： 陈庆奎， 周利珍， CHEN Qing-kui， ZHOU Li-zhen
作者单位： 上海理工大学光电信息与计算机工程学院,上海,200093
刊名： 计算机应用 ISTIC PKU
英文刊名： Journal of Computer Applications
年，卷(期)： 2012,32(7)

参考文献(12条)

1. International Telecommunication Union ITU Intemet Reports 2005:The Internet of Things 2005

2. RFID WORKING GROUP Internet of Things in 2020:Roadmap for the future[外文期刊] 2011

3. CONTI J P The Internet of things 2006(06)

4. CHAIKEN R;JENKINS B;LARSON P-A SCOPE:Easy and efficient parallel processing of massive data sets 2008(02)

5. YICK J;MUKHERJEE B;GHOSAL D Wireless sensor network survey[外文期刊] 2008(12)

6. Oracle Corporation Oracle real application clusters 2011

7. The Apache Software Foundation Hadoop 2011

8. The Apache Software Foundation Apache HBase[外文期刊] 2011

9. CHANGF;DEANJ;GHEMAWATS BigTable:Adistributed storage system for structured data 2006

10. The Apache Software Foundation Apache ZooKeeper[外文期刊] 2010

11. MARSHALLD,Remote Procedure Calls(RPC)[外文期刊] 2011

12. Google Project Hosting Google-gson 2011