基于融合数据库的海量传感器信息存储架构

类兴邦[1] 房 俊[2,3]
(山东科技大学信息科学与工程学院 青岛 266590)[1] (北方工业大学云计算研究中心 北京 100144)[2]
(大规模流数据集成与分析技术北京市重点实验室 北京 100144)[3]

摘 要 在物联网、工业监控等系统中,庞大规模的传感器每时每刻都在产生大量的数据。实时数据库在处理高时效性数据方面具有较强的优势,但是在处理大规模传感器数据方面存在着存储量低、不便于扩展的弊端。而HBase在处理海量数据方面具有高读写性能、高扩展性、高可靠性和高存储量的优势。通过将实时数据库与HBase相结合,设计并实现了基于融合数据库的传感器信息存储架构。该架构采用多租户机制,对HBase写入进行了优化,将原来分散的传感器数据集中式存储,并把传感器元数据与历史数据分离存储,同时维持了实时数据库原有的查询、数据组织结构的特点。经过实验验证,该架构具有较高的读写性能以及良好的可扩展性,有效避免了Region写入热点,实现了集群负载均衡。

关键词 实时数据库,HBase,传感器数据,集中式存储,写入优化
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.6.014

Mass Sensor Information Storage Infrastructure Based on Fusion Database

LEI Xing-bang[1] FANG Jun[2,3]
(School of Information Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China)[1]
(Research Center for Cloud Computing, North China University of Technology, Beijing 100144, China)[2]
(Beijing Key Laboratory on Integration and Analysis of Large-scale Stream Data, Beijing 100144, China)[3]

Abstract In the internet of things, industrial control and other systems, large-scale sensor generates a large amount of data all the time. The Real-time database has an advantage in terms of time-sensive data processing, while it has a problem in storage capacity and scalability. On the contrary, the HBase has the advantage of high read and write performance, high scalability and high reliablity. Through the combination of real-time database and HBase, we designed and implemented the sensor information storage architecture based on fusion database. The architecture uses a multi-tenant mechanism to optimize HBase writes, the original sensor data are centrally stored, and the sensor metadata and historical data are stored separately, while maintaining the original real-time database queries, data structure characteristics. Our experiments verify that the system has high read and write performance and good scalability. And it effectively avoids the region write hot and achieves the objective of the cluster load balancing.

Keywords Real-time database, HBase, Sensor data, Centralized storage, Write-optimized

1 引言

在物联网、工业监控等系统中,庞大数量的传感器每时每刻都在产生大量实时变化的数据,将这些数据收集起来,其规模将是海量的。为了处理传感器实时产生的数据,实时数据库应运而生,其主要用来处理高时效性、高吞吐量的数据,并根据这些数据对设备进行实时监控、预警。对实时数据库中的历史数据进行分析、挖掘,能够为企业决策等提供依据,因此历史数据具有十分重要的价值。目前实时数据库处理历史数据的传统方法是高效地对数据进行压缩,采用文件进行存储;或者与关系型数据库进行结合,将数据存储到关系型数据库中。但这两种方式都存在一定弊端:采用高效的数据压缩方式,一定程度上降低了数据查询的速度;存储到关系数据库中的方法在处理海量数据时,存在无法大规模扩展、不善于处理非结构化数据以及高并发查询时效率低下的问题。同时由于实时数据库处理能力的限制,每个企业、单位往往需要独立部署,从而呈现出数据库在各地离散分布的特点,这就使得大量数据不能集中统计,无法进行有效的数据挖掘。

为解决实时数据库的上述问题,结合NoSQL类型数据库在处理海量数据方面具有的高扩展性、高性能、高可靠性、稀疏性、能够存储大容量数据的优势,本文设计并实现了基于融合数据库的海量传感器信息存储架构。针对实时数据库原

到稿日期:2015-06-29 返修日期:2015-09-27 本文受北京市属高等学校创新团队建设与教师职业发展计划基金资助项目(IDHT20130502),北京市教育委员会科技计划重点项目(KZ201310009009)资助。
类兴邦(1989-),男,硕士生,主要研究方向为流数据存储优化,E-mail:leixingbang@163.com;房 俊(1976-),男,副研究员,主要研究方向为云数据管理,E-mail:fangjun@ncut.edu.cn。

有数据离散分布不便于统计、挖掘，传统的基于 HBase 的数据写入方式负载不均衡、写入效率较低，同时各地实时数据库作为写入前端需要保留原有的数据查询方式等问题，系统做了以下优化和改进。1）数据集中式存储：通过多租户机制将原来离散分布在各地的实时数据库数据集中式存储到 HBase 中，从而能够有效地对历史数据进行统计、挖掘。2）保留实时数据库数据的原有查询的特征，对实时数据库的逻辑结构进行映射，将映射关系保存到关系型数据库中，将传感器产生的大量历史数据保存到 HBase 中，实现了传感器元数据与历史数据的分离，进而保留了原有实时数据库数据逻辑结构的特点，架构针对实时数据库原有的查询，设计了特定查询模块，保留了实时数据库的查询特征。3）提高 HBase 写入性能：通过合理的 RowKey 设计，根据 Rowkey 范围预分区，实现了整个集群的负载均衡；通过多线程并发、批量数据写入提高了 HBase 写入速度；引入多源缓冲策略，将写入失败的数据重新放入缓冲区，保证了数据的完整性。

2 相关工作

文献[1]针对物联网传感器数据的海量、异构、时空敏感等特性，采用了自定义的 IoT-ClusterDB 数据库集群架构，该架构对传感器数据进行预处理，只保留关键部分数据。同时，该架构把监控对象的采样值组织成采样序列进行存储，建立了传感器时空数据模型，提供了能够同时支持“键-值”查询和普通 SQL 的快速查询方式，但是其存在着对传感器信息进行预处理而损失部分有挖掘价值信息的问题。文献[2]根据区域构建存储集群，使得每个区域都有一个 HBase 集群存储数据。在所有区域上设计了一个全局数据存储，用来保存各个区域集群的元数据信息，从而形成了两层存储架构。该方法将各个区域采集的数据存放在区域服务器，解决了网络资源消耗大的问题。但是由于集群分布性存储，其存在不便于维护和对数据进行统计挖掘的问题。文献[4]设计了一种实时数据库历史数据的压缩架构，对缓冲区、磁盘管理以及文件组织结构进行了详细的描述，但是该架构单纯强调了压缩比，未对压缩以后的数据查询进行进一步说明。文献[6]以 HBase 作为数据库存储系统，对车辆 GPS 等传感器 RowKey 和列族的设计方法进行了描述，并对 HBase 的可拓展性进行了测试。本文的 RowKey 设计借鉴了该文献的设计思想。

2.1 实时数据库以及 ThinkDB

实时数据库是数据库与实时系统相结合的一种新型数据库，负责管理有时间限制的数据和事物，整个系统的正确性不仅依赖于逻辑结果，还依赖于逻辑结果产生的时间。ThinkDB 是由中科启信开发的实时数据库系统。如图 1 所示，该数据库系统将一个传感器定义为一个标签点，并将标签点作为最小管理单元。将多个标签点划分为一个逻辑分组“设备站”以方便管理。该系统提出了“数据订阅”的概念，通过主动推送的数据发布形式使得客户端无需主动查询，就可以获取指定标签点的最新值。客户端可以通过“历史查询”的方式查询出指定标签点产生的历史数据。ThinkDB 针对传感器数据的特点，提出了自定义的查询语言 ChinSQL，该查询语言具有特定的查询功能，例如断面查询：查询标签点在某一时刻的数据；统计查询：统计一个标签点的某个时段的个数、最小值、最大值、求和、平均值等。

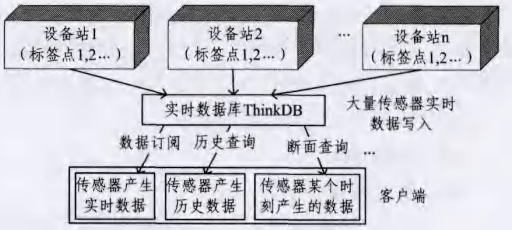

图 1 ThinkDB 实时数据库体系结构

2.2 HBase 数据库

HBase 是架构在分布式文件系统 HDFS 之上的分布式开源数据库。Hadoop 由于具有开源、底层细节透明、可部署到低廉的硬件、易拓展扩容的特点，因此得到了广泛应用。HBase 是面向列、稀疏、分布式、持久化存储的多维映射表。在 HBase 中，具有相同列族的数据将被存放在一起，这使得在进行读写的操作时，可有效地降低磁盘 I/O 的开销，也可以使得数据被更好地压缩，经过压缩后的数据大约只有原数据的 1/4，极大地节省了物理空间。HBase 是为 TB 到 PB 级别的海量数据存储而设计的，可部署到成千上万台普通计算机上，能够被大规模用户高速读写。HBase 的上述特性使得存储海量传感器历史数据具备较强的优势。

3 基于融合数据库的海量传感器信息存储架构

3.1 实时数据库数据模型到关系型数据库的映射

为了方便对数据进行管理并充分对传感器数据进行挖掘，该架构引入了多租户机制，将传感器数据集中式存储。为了保证用户在多租户下仅能访问到自己的数据，传统的做法是将租户 ID 和传感器 ID 都存储到 HBase 的 RowKey 中。这就使得每一条传感器数据都会保存租户 ID，从而导致大量数据冗余且查询效率较低。同时 HBase 是一个分布式存储系统，在多连接查询方面存在较大缺陷，会给以后的数据挖掘、分析带来不便。为解决上述问题，该架构采用传感器的元数据信息和历史数据分离的策略，将传感器历史数据存储到 HBase 中，而将传感器元数据信息、多租户与传感器之间所属关系等存储到关系型数据库中。图 2 示出从实时数据库的数据模型到关系型数据库的映射，其中 TennatInfo 存储着所有租户的信息，TeanantToSensor 存储着传感器与租户之间的所属关系，SensorInfo 存储着传感器的描述信息，Station 代表上文提到的 ThinkDB 逻辑分组“设备站”，而 SensorToStation 代表传感器与设备站之间的所属关系。

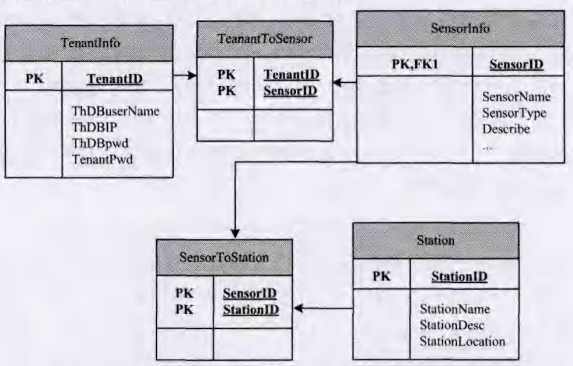

图 2 实时数据库逻辑结构到关系型数据库的映射

系统根据租户注册的实时数据库的 IP、用户名、密码信

息，通过“标签点信息查询”方式获取传感器元数据信息，并将其保存到关系型数据库的上述表中。上述映射和存储方式有效地降低了数据冗余度，既保持了实时数据库原有的数据模型的特点，又实现了系统为多租户服务。

3.2 传感器数据存储

3.2.1 传感器数据写入

如图 3 所示，实时数据库离散分布在各地单位，并保存着本单位内的传感器产生的、具有一定时效的数据。存储系统根据注册信息，向已注册的实时数据库发送“数据订阅”请求。实时数据库会把传感器产生的实时数据通过主动推送的形式传送到存储系统中。传感器类别的差异使得数据结构呈现多元化，存储系统首先根据传感器类别对数据进行分类，然后将分类后的数据存入缓冲区中。

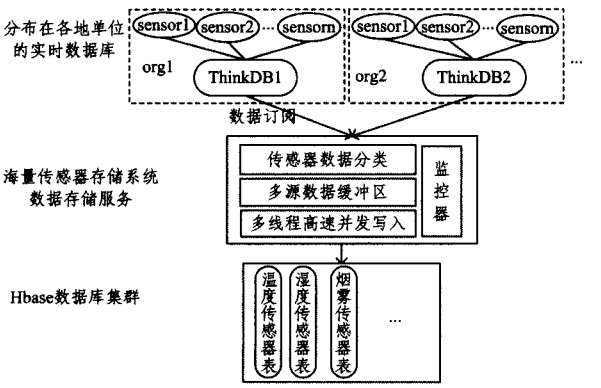

图 3 数据存储服务详细设计图

在大量数据写入 HBase 过程中，会触发 Region 的 split 操作，将一个 region 分裂为两个。在 split 的过程中，HBase 会对 Region 加锁，对该 Region 的访问请求会被 block。HBase 对 Region 的 spilt 操作会导致 Region 在RegionServer 上分布不均，紧着接会触发 HBase 的 Region Balance 操作，此过程会导致 Region 下线，如果客户端仍然向已下线的 Region 写入大量数据，将会导致异常出现。因此 HBase 写入性能具有一定的不稳定性。为解决上述问题，引入多源数据缓冲区功能模块。

多源数据缓冲区主要有两个作用：1）缓存批量写入的数据。HBase 写入的最大瓶颈在于网络 IO，数据批量写入将显著地提高写入性能。因此，系统首先将分类后的传感器数据存放在缓冲区中，当缓冲区的数据量达到一定阈值时，便触发一个线程，将数据写入 HBase。2）保存写入失败的数据。触发线程进行写入并不意味着写入数据成功，为保证数据完整性，系统将写入时抛出异常、没有写入成功的数据重新放回缓冲区队列中等待系统调度，再次进行写入，直至写入成功。

HBase 支持高并发写入，为进一步提高写入性能并保证系统稳定，采用线程池技术对多线程进行管理和调度，并使用多线程对数据进行并发写入。

监控器的主要功能是对数据的写入平均速度、瞬时写入速度、总写入量以及在缓冲区缓存的数据总量进行实时统计，并记录到日志文件中。如上文所述，当 Region 已下线时，如果仍然有大量数据执行写入操作，将会出现异常。监控器会实时捕获到异常，并通知触发写入线程的主线程。主线程将采取休眠策略，减少写入线程触发，从而降低写入速度，避免异常频繁出现。

3.2.2 传感器数据存储设计

（1）RowKey 的设计

HBase 的索引是建立在行键 RowKey 的基础上的，用户需要尽量将查询的维度或者信息放在行键中。含有结构信息的整个单元格称为 KeyValue，KeyValue 在存储时先按行键从左到右降序存储[3]。例如行键为：0100，0070，1000，3310，经过排序后的顺序为 0070，0100，1000，3310。在 HBase 中，一张表会随着数据量的增加划分为若干个 Region。Region 存储着其从 Start Key 到 End Key 之间的数据。多台 RegionServer 维护和管理这些 Region。如果 RowKey 不断递增写入，会使得写入热点出现。在此情况下，Region 也会不断分裂，导致 HBase 有一段不可用期，造成写入瓶颈。因此 RowKey 的设计将影响数据插入和查询的性能。

针对上述问题，将用户频繁查询的时间维度和传感器 ID 维度组合作为 RowKey：SensorID＋Time。由于各地的实时数据库是独立部署的，因此可能出现两个不同的单位的传感器 ID 重复的情况。为了便于 Region 预分区，基于行键排序特点，在用户注册时，对所有的传感器 SensorID 重新编号，统一将其设置为长度为 8 位、不足位补 0 的整数，如 00000001，00000002，…，Time 则置为传感器数据产生的时间和 GMT 时间（格林威治时间）所差的毫秒数。

基于上述原则设计的 RowKey 具有明显的优点：首先，随着时间的变更，一个传感器会产生大量的数据，由于行键排序的特点，同一个传感器的数据存储位置会相对集中，同时将查询常用的时间维度和 ID 加入 RowKey，提高了查询的效率；其次，同一时间会有大量不同的传感器数据写入，不会出现写入热点，从而使得集群负载较为均衡；最后，SensorID 统一重新编号，使得 Region 预分区更方便。

（2）列族和列设计

在 HBase 中，列表示为〈列族〉：〈限定符〉。在 Hbase 中，列族是预先定义好的，列族中的列是随意添加的。在各种类型传感器表中只设置一个 dataColFamily 列族，并通过限定符来表示传感器的各种属性。这样设计的好处在于：1）即使传感器类别不同，由于列可以任意添加，也可以通过添加列的形式来解决数据多源异构的问题；2）相同列族的数据在物理上是存放在一起的，相同读写方式的数据放在一起，提高了 HBase 的读写性能。

3.3 系统查询

如相关工作中所述，ThinkDB 实时数据库对传感器数据的特点提出了自定义的查询语言 ChinSQL。如图 4 所示，为了租户仍然能像使用实时数据库一样使用该系统，针对“历史查询”、“统计查询”等特定查询方式增加了 ChinSQL 查询模块。以查询语句“SelectID，NAME，STATION，VALUE，CHINRTCMD HISTORYQ WHEHRE Start _ Time = 1321243200，End_Time＝1321243220，VALUE＞28，TYPE＝‘temp’”为例，查询语句含义为查出在 Start_Time 到 End_Time 时间范围内、类型为温度传感器、值大于 28 的所有传感器的信息与对应的值。特定查询模块首先会对 ChinSQL 语法进行校验，校验完毕后会对语义进行解析，随后校验该用户是否注册了温度传感器数据集，在校验完毕后，向查询适配器发送查询请求。查询配置器屏蔽了底层数据库 HBase 和Oracle的区别。由于传感器的元数据信息和产生的历史数据信

息分别存储在 Oracle 和 HBase 中，因此先向 Oracle 查询出用户具有访问权限的传感器的元数据信息，然后再向 HBase 集群查询这些传感器在对应时间段、VALUE＞28 的历史数据，并将查询后的结果组合、封装，最后返回给用户。

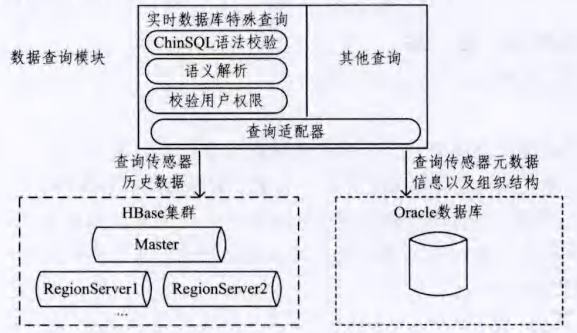

图 4　系统查询模块设计

4　实验设计与结果

文献[12]中对 HBase、Cassandra、Redis 等 NoSQL 类型数据库和 Mysql 关系型数据库在不同集群结点下的吞吐量进行了测试，该测试并未说明写入数据的格式和类型，因此并不具备针对性。本文针对传感器类型的数据设计了特定负载均衡和可拓展性的测试。

4.1　系统高性能读写以及负载均衡测试

实验环境如表 1 所列。

表 1　实验设备表

设备名称	设备配置	设备用途
HBase 数据库集群（3 台）	2×4 核 2.4GHz CPU、8GB 内存、千兆网卡	存储传感器数据，其中 1 台作为 master 节点，另外 2 台作为 RegionServer 结点
Oracle 数据服务器（1 台）	2×4 核 2.4GHz CPU、8GB 内存、千兆网卡	持久化存储传感器数据
Mysql 数据服务器（1 台）	2×4 核 2.4GHz CPU、8GB 内存、千兆网卡	持久化存储传感器数据
海量传感器系统服务器（1 台）	4 核 2.7GHz CPU、4GB 内存 千兆网卡	部署海量物联网传感器信息存储系统，模拟产生大量传感器数据

实验目的：对传统关系型数据库 Mysql、Oracle 与 HBase 的写入性能和查询进行对比，期望验证采用 HBase 高性能读写的特点。

实验步骤：1）数据生成。利用程序模拟订阅分布在多地的实时数据库传感器，根据系统缓冲区数据总量的情况，模拟 5000 个传感器，随机、动态生成数据，能够保证在系统缓冲区一直有大量数据未写入，以测试在当前系统下，不同数据库的最高写入性能。2）向各类型数据库中写入数据。在 Oracle、Mysql 中创建 sensor 表，在 HBase 中根据传感器 ID 范围平均地划分为 10 个预分区，并创建 Sensor 表。分别向 Oracle、Mysql、HBase 写入 5000 万条数据，根据监控器记录写入时间和写入数据量数据。3）统计 HBase 各 Region 请求次数。通过 HBase 的管理工具查询出各 Region 的请求数量，分别记录并统计。4）统计各数据库查询时间。在关系型数据库 Oracle、Mysql 中建立索引。在 5000 个传感器中随机选取了 10 个传感器，分别在各类型数据库中查询其在 2015/6/10 10:53:40 到 2015/6/10 10:53:50 之间的传感器数据。对查询 10 个传感器各数据库所花费的查询时间取平均值，并做记录。

实验结果：1）图 5 显示 HBase 各 Region 接受请求的次数近似均等，证明 RowKey 设计以及预分区策略有效避免了写入热点，实现了 HBase 集群负载均衡。2）如图 6 所示，在 HBase 集群、Mysql 数据库、Oracle 数据库 3 个写入平台，写入数据总量和写入时间呈现非常明显的线性增长。由两台机器作为 RegionServer 结点的 HBase 集群平均写入速度达到 4.4 万/s，而 Oracle 数据库和 Mysql 数据库平均写入则分别为 2.5 万/s、2.1 万/s。对比传统关系型数据库存储传感器数据，HBase 具有较高的写入性能。3）如图 7 所示，在 Mysql、Oracle 都建立索引的前提下，HBase 的查询速度仍比 Oracle 的提高了近 25%，比 Mysql 的提高了近 53%。在实验中，Mysql 和 Oracle 在 5000 万条数据的规模下，建立索引耗时长达 20 多分钟，并且建立索引将会降低数据库写入性能。因此对比传统的关系型数据库，HBase 具有良好的读写性能。

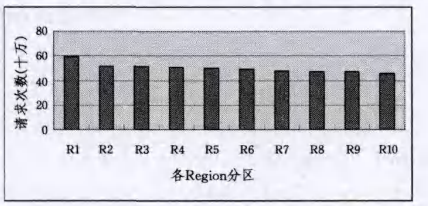

图 5　HBase 各 Region 接受请求次数的统计

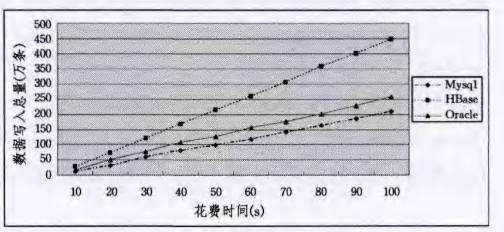

图 6　各数据库写入性能

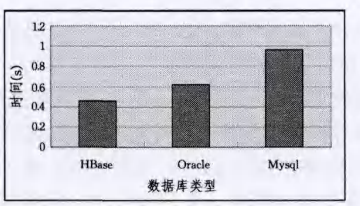

图 7　各数据库查询耗时

4.2　系统可拓展性实验

实验环境：8 台配置为 2×4 核 2.4GHz CPU、8GB 内存、千兆网卡的计算机，其中 1 台作为 master 结点，其他为 RegionServer 结点。

实验目的：测试 HBase 集群在不同数量 RegionServer 结点下的写入性能，统计并分析实验结果，期望验证在增加 HBase 的 RegionServer 结点的情况下，能够提高 HBase 写入速度，验证该架构具有良好的可拓展性。

实验步骤：按照 4.1 节的实验模拟生成传感器数据。在节点数量不同的 HBase 集群上分别写入 1000 万条传感器数据，统计使用时间，求得平均写入速度。

实验结果：从图 8 中可以看出，系统的写入数据的速度随

（下转第 111 页）

ling in Vehicular Sensor Networks[C]// 2013 5th International Conference on Intelligent Networking and Collaborative Systems (INCoS). IEEE, 2013: 448-455

[9] Rasyid M U H A, Lee B H, Sudarsono A. PEGAS: Partitioned GTS Allocation Scheme for IEEE 802. 15. 4 Networks[C]// 2013 International Conference on Computer, Control, Informatics and Its Applications (IC3INA). Nov. 2013: 29-32

[10] Xia Feng, Hao Ruo-nan, Cao Yang, et al. ART-GAS: an adaptive and real-time GTS allocation scheme for IEEE 802. 15. 4[C]// Proc of AINTEC '11. New York: ACM, 2011: 96-103

[11] Chen Jian-xin, Ferreira L, Tovar E. An Explicit GTS allocation algorithm for IEEE 802. 15. 4[C]// Proc of Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conference. Toulouse: IEEE, 2011: 1-8

[12] IEEE 802. 15. 4 standard : Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs) [R]. October 2003: 1-133

[13] Lee D H, Roh H T, et al. Performance Analysis of the IEEE 802. 15. 4 MAC Protocol[C]// 2013 International Conference on ICT Convergence (ICTC). 2013: 398-401

[14] Shen Zhuo-wei. Schedulability Analysis for Non-Preemptive EDF Scheduling Algorithm[J]. Computer Engineering and Applications, 2006, 42(9): 10-12(in Chinese)
沈卓炜. 不可抢占式 EDF 调度算法的可调度性分析[J]. 计算机工程与应用, 2006, 42(9): 10-12

[15] Qiao Guan-hua, Mao Jian-lin, Guo Ning, et al. Research and Improved Design in IEEE 802. 15. 4 MAC Protocol for Service Distinguishing[J]. Computer Science, 2014, 41(10): 149-153(in Chinese)
乔冠华, 毛剑琳, 郭宁, 等. 基于业务区分的 IEEE 802. 15. 4 协议分析及改进[J]. 计算机科学, 2014, 41(10): 149-153

(上接第 71 页)

着 RegionServer 结点数量的增加呈间隔性线性增长，证明了系统具有良好的可拓展性。因此在租户以及传感器数量增加的情况下，可以通过增加 HBase 的 RegionServer 结点的数量来提高系统的吞吐率。

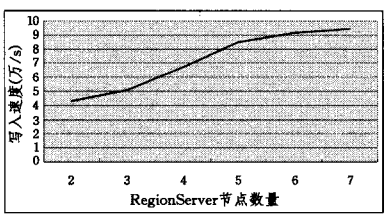

图 8 不同 RegionServer 节点下的写入速度

结束语 本文针对实时数据库在处理大规模历史数据方面的缺陷，鉴于 HBase 在大数据处理方面的优越性，提出了基于融合数据库的海量传感器信息存储架构。该架构将传感器数据集中式存储，采用多租户机制，将原来分散的传感器数据集中式存储，通过将传感器元数据与数据分离，维持了实时数据库原有查询、数据组织结构的特点，并对 HBase 写入进行了优化。经过实验验证，该系统具有良好的可拓展性和读写性能，有效避免了写入热点。

同时，该架构有一定的不足之处，进一步的改进方向包括：1)如何在保证高速并发写入的情况下，使系统有较好的查询性能；2)如何对存储的海量数据信息进行进一步挖掘。

参 考 文 献

[1] Ding Zhi-ming, Gao Xu. A Database Cluster System Framework for Managing Massive Sensor Sampling Data in the Internet of Things[J]. Chinese Journal of Computers, 2012, 35(6): 1175-1191(in Chinese)
丁治明, 高需. 面向物联网海量传感器采样数据管理的数据库集群系统框架[J]. 计算机学报, 2012, 35(6): 1175-1191

[2] Chen Qing-kui, Zhou Li-zhen. HBase-based storage system for large-scale data in wireless sensor network[J]. Journal of Computer Application, 2012, 32(7): 1920-1923, 1977(in Chinese)
陈庆奎, 周利珍. 基于 HBase 的大规模无线传感网络数据存储系统[J]. 计算机应用, 2012, 32(7): 1920-1923, 1977

[3] Lu Ting, Fang Jun, Qiao Yan-ke. HBase-based Real-time Storage System for Traffic Stream Data[J]. Journal of Computer Application, 2015, 35(1): 103-107, 135(in Chinese)
陆婷, 房俊, 乔彦克. 基于 HBase 的交通流数据实时存储系统[J]. 计算机应用, 2015, 35(1): 103-107, 135

[4] Lu Hui-ming, Zhou Zhao, Liao Chang-bin. Historical Data Processing Based On Real-time Database System [J]. Electric Power Automation Equipment, 2012, 29(3): 127-131(in Chinese)
陆会明, 周钊, 廖常斌. 基于实时数据库系统的历史数据处理[J]. 电力自动化设备, 2012, 29(3): 127-131

[5] George L. HBase: The Definitive Guide [M]. 2. Inc, USA: O'Reilly Media, 2013: 339-350

[6] Ku W Y, Chou T Y, Chung L K. The CloudBased Sensor Data Warehouse[C]// International Symposium on Grids and Clouds and the Open Grid Forum. Taipei, Taiwan, 2011: 21-24

[7] Carstoiu D, Cernian A, Olteanu A. Hadoop Hbase-0. 20. 2 performance evaluation[C]// 2010 4th International Conference on New Trends in Information Science and Service Science (NISS). IEEE, 2010: 84-87

[8] The Apache Software Foundation[OL]. http://hadoop. apache. org

[9] Rabl T, Gómez-Villamor S, Sadoghi M, et al. Solving Big Data Challenges for Enterprise Application Performance Management [J]. Proceedings of the Vldb Endowment, 2012, 5(12): 1724-1735

[10] Kallman R, Kimura H, Natkins J, et al. H-store: a high-performance, distributed main memorytransaction processing system [J]. PVLDB, 2008, 1(2): 1496-1499

[11] Lakshman A, Malik P. Cassandra: a decentralized structured storage system[J]. SIGOPS Operating Systems Review, 2010, 44(2): 35-40

[12] http://planetcassandra. org/nosql-performance-benchmarks