Journal of Computer Applications ISSN 1001-9081 2015-01-10
计算机应用,2015,35(1):103-107,135 CODEN JYIIDU http://www.joca.cn

文章编号:1001-9081(2015)01-0103-05 doi:10.11772/j.issn.1001-9081.2015.01.0103

基于 HBase 的交通流数据实时存储系统

陆 婷,房 俊*,乔彦克

(北方工业大学 云计算研究中心,北京 100041)

(* 通信作者电子邮箱 fangjun@ncut.edu.cn)

摘 要: 交通流数据具有多来源、高速率、体量大等特征,传统数据存储方法和系统暴露出扩展性弱和存储实时性低等问题。针对上述问题,设计并实现了一套基于 HBase 交通流数据实时存储系统。该系统采用分布式存储架构,通过前端的预处理操作对数据进行规范化整理,利用多源缓冲区结构对不同类型的流数据进行队列划分,并结合一致性哈希算法、多线程技术、行键优化设计等策略将数据并行存储到 HBase 集群服务器中。实验结果表明:该系统与基于 Oracle 的实时存储系统相比,其存储性能提升了 3~5 倍;与原生的 HBase 方法相比,其存储性能提升了2~3 倍,并且具有良好的扩展性能。

关键词: 流数据;多源缓冲区;数据切分;一致性哈希算法;实时存储;HBase

中图分类号: TP311 **文献标志码:** A

HBase-based real-time storage system for traffic stream data

LU Ting, FANG Jun*, QIAO Yanke

(*Research Center for Cloud Computing, North China University of Technology, Beijing* 100041, *China*)

Abstract: Traffic stream data has characteristics of multi-source, high speed and large volume, etc. When dealing with these data, the traditional methods and systems of data storage have exposed the problems of weak scalability and low real-time storage. To address these problems, this work designed and implemented a HBase-based real-time storage system for traffic streaming data. The system adopted the distributed storage architecture, standardized data through front-end preprocessing, divided different kinds of streaming data into different queues by using multi-source cache structure, and combined the consistent Hash algorithm, multi-thread and row-key optimization strategy to write data into HBase cluster in parallel. The experimental results demonstrate that, compared with the real-time storage system based on Oracle, the storage performance of the system has 3 - 5 times increment. When compared with the original HBase, it has 2 - 3 times increment of storage performance and it also has good scalability.

Key words: streaming data; multi-source buffer; data sharding; consistent Hash algorithm; real-time storage; HBase

0 引言

大数据处理等技术的出现为现代城市智能交通系统(Intelligent Transportation System, ITS)提供了新的发展机遇。目前各大城市都建立了专用车辆数据采集网络,采集手段越来越多样化,包括摄像头、地磁棒、射频识别、微波、线圈、全球定位系统(Global Positioning System, GPS)设备等多种形式,所采集的信息也越来越丰富,有浮动车数据、车辆识别数据、交通流数据等。这些不断生成的数据已经逐渐形成了极具价值的大规模交通流数据,对解决交通拥堵问题、提升交通管理水平等具有重要意义。

上述交通数据是一类典型的流数据,除了种类多以外,还具有速率快、总量大的特点。仅以一个大型城市的车辆识别交通数据为例,城市道路上所部署的车辆识别传感器数量为5 000,每个点的高峰采样频率为每秒 1 条,则每秒将产生5 000条车辆识别数据,每天的高峰折算率为 0.33,一年车辆识别数据记录数将超过 500 亿条,数据存储量为 10 TB 级(来自实际项目)。

本文重点关注这一类流数据的实时存储问题。传统的基于关系数据库的数据存储系统在面对交通流数据时存在写入延迟高、水平扩展能力差以及固定数据结构等诸多问题。以 HBase 为代表的 NoSQL 类数据库由于采用了简单数据模型,相对于传统的关系数据库,具有存储速度快、扩展性高、数据结构随机等特点,非常适合作为交通流数据的存储介质。但是在实际运用 HBase 过程中,也暴露出一些问题,如下所示。

1) 行键(row key)冗余设计对存储效率的影响。为了提供高效的查询能力,HBase 往往需要对行键进行冗余设计,比如,交通识别数据可以把行键结构设置为图 1 所示格式。

监测时间 (4比特)	摄像头编号 (6比特)	车牌号 (11比特)	监测地点 (11比特)

图 1 交通识别数据的行键结构示例

这种行键结构提高了查询交通流数据的效率,但是相对

收稿日期:2014-07-18;修回日期:2014-09-09。

基金项目:北京市自然科学基金重点项目(4131001);北京市属高等学校创新团队建设与教师职业发展规划项目(IDHT20130502);北大方正集团有限公司数字出版技术国家重点实验室开放课题;北方工业大学科研启动基金资助项目。

作者简介:陆婷(1990-),女,山东菏泽人,硕士研究生,主要研究方向:流数据的存储优化;房俊(1976-),男,江苏南京人,副研究员,主要研究方向:服务计算、云计算;乔彦克(1989-),男,河南漯河人,硕士研究生,主要研究方向:云数据处理。

于简单短小的行键(row key)结构,存储量大带来了数据存储性能的下降。

2)数据写入中的热点问题。对于row key具有连续性特质的流数据,其会按照row key的字典序依次插入集群中的一个region中,待此region达到一定的阈值后才会向其他region依次插入。这种数据写入热点,降低了HBase数据库的写入性能。

为此本文设计并实现了一种基于HBase的交通流数据实时存储系统。该系统通过前端的多源实时感知数据接入预处理,利用多源缓冲区结构对不同类型的流数据进行队列划分,并在数据写入时采用多线程技术,数据存储时采用一致性哈希算法和HBase行键优化设计等策略将数据均衡分布到HBase集群服务器中,保证了存储的性能和可伸缩性。

1 相关工作

在流数据存储系统方面,文献[1-2]是一类典型的以关系数据库作为持久化存储介质的工作。尤其是文献[2]提出一种基于Oracle的面向流数据的分布式实时存储(Distributed Real-Time Storage, DRTS)方法,该方法依据窗口阈值,不间断地从缓存区接收数据,然后结合一定的分布策略将数据分布存储到数据服务器中。虽然这些工作在一定程度上提升了数据实时存储的效率,但是如引言中所示,其在交通流数据存储方面仍存在许多不足。文献[3]给出了一个基于HBase的大规模无线传感网络数据存储系统。该系统虽然实现了传感数据近实时的存储,但是其忽略了数据在传输过程中的完整性、多元性所带来的存储问题。例如对于不完整的或者是不合规范的传感数据的存储会造成存储空间的浪费和数据冗余;将不同类型传感网络数据直接存储到数据库时,百亿条数据对象无规则性的存储方式一方面会加大对海量数据管理难度,另一方面也会降低针对特定类型数据的查询效率。

在具体优化技术方面,包括数据切分和数据分布策略。在数据切分方面,文献[2]中还提出了车辆流数据切分策略,该方法解决了使用单一队列接受数据时的阻塞问题。但是该方法也忽略了多源数据之间模式不同的问题,使用该方法会将所有类型的数据统一管理存储,在影响数据写入效率的同时也降低了流数据查询的效率。在数据分布策略方面,一致性哈希算法克服了传统Hash数据分布算法的不能够满足单调性,致使某台数据服务器负载过高的问题,已经在实际应用中广泛使用。具体实现可参考文献[4]。

基于多线程技术的并行数据写入也是提升数据存储性能的方法。文献[5]从开发数据通信软件的实际问题入手,提出了多线程技术编程方案,最后示例了多线程技术在数据通信中的应用,较好地解决了数据通信的延时处理问题。

文献[6]中提出了一种基于NoSQL的LaUD-MS系统存储架构,该系统在一定程度上解决了海量数据存储、快速读写响应等难题。本文设计的系统在系统架构的设计上参考了文献[6]所提及的存储架构。

本文在以上研究工作的基础上,设计实现了基于一个HBase的交通流数据实时存储系统,在一定程度上有效地解决了高数据流速的实时存储问题,提高了交通流数据的存储效率。

2 基于HBase的交通流数据实时存储系统

针对交通流数据多源、异构、海量、高速的特点,本文设计和实现了基于HBase的交通流数据实时存储系统,接下来分别从系统架构、关键设计及核心流程三方面对系统进行介绍。

2.1 系统架构

系统架构如图2所示,整个系统由4部分构成,分别如下。

1)数据接入预处理层。该层实现对交通流数据的整合、规范化操作,以保证数据的完整性、有效性。经预处理之后的数据上送至缓冲区进行队列划分。

2)数据缓冲区。为了提高海量流数据批量写入的速度,实现对不同类型流数据的实时处理,该层实现了针对不同类型数据对象的多源缓冲区结构及相应的数据划分策略。

3)数据写入层。数据缓冲区将该段时间内的数据划分完成后,数据写入区接收分片数据。写入层基于多线程技术,采用并行写入的方法存储数据。如何将各个分片数据有序存储到数据存储服务器中,保证海量并发数据的分布存储是至关重要的。

4)数据存储层。基于HBase集群数据库分布式存储写入层发送过来的数据,数据的相关组织和分布信息保存到元数据库中。

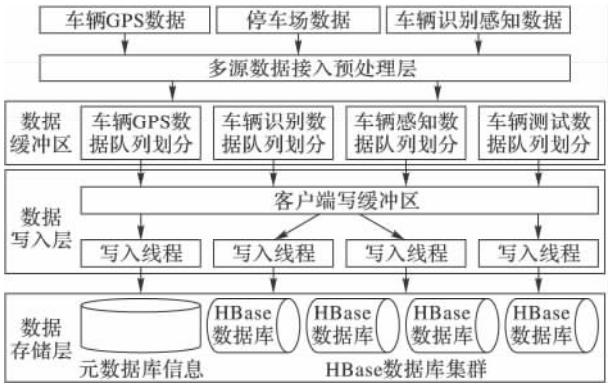

图2 系统架构

本文接下来重点讨论多源缓冲区结构及队列管理策略、数据写入层row key设计及数据分布写入等内容。有关数据接入层的设计可参考文献[7],元数据的设计可参考文献[8]。

2.2 关键设计

2.2.1 多源缓冲区结构

实际交通应用中的多源化数据带来了其结构的多样性,车辆GPS数据、停车场数据以及车辆识别感知数据的组成结构如图3所示。

GPS数据	数据ID	年月日信息	监测时间	经纬度	监测地点	海拔高度	车牌号	标志性建筑NAME	存储文件编号	
停车场数据	数据ID	摄像头编号	车辆颜色	车牌号	车辆类型	位置	监测时间	监测地点	监测部门	存储文件编号
识别感知数据	数据ID	摄像头编号	车辆颜色	车牌号	车辆类型	车速	监测时间	监测地点	监测部门	车辆照片	存储文件编号

图3 各类型数据结构组成

多源数据接入层预处理之后的流数据会上送到多源队缓冲结构中的数据队列划分区。多源缓冲区结构如图4所示。数据队列划分阶段是指在其数据类型对应的缓冲区结构中把该类型数据的监测时间属性通过Hash方法进行转换,进行

Hash 直接取余,最后根据 Hash 值将该条数据放置到对应的队列中。各个队列中数据对象在内存中采用链表结构存储,为了提高划分完成后数据的传输的速率,设定各类型缓冲区中均有 3 个队列等待接收划分完成的数据,如图 5 所示。QueueList 代表单位时间内发送的数据对象,List1,List2,…代表各个队列子对象。例如,某一类型的流数据缓冲区中有 3 个空队列在等待数据,接收到该条数据的记录时间 rTime 为 2012-10-17 00:00:52,将该时间转换成 1970 年以来的毫秒数 *times*,*times* 为 1 350 403 252 000 ms,*times* 值直接 Hash 对 3 取余后,计算出 Hash 值为 1,则会将该条数据放置在 Hash 值为 1 对应的队列中。数据划分组织结构如图 5 所示。

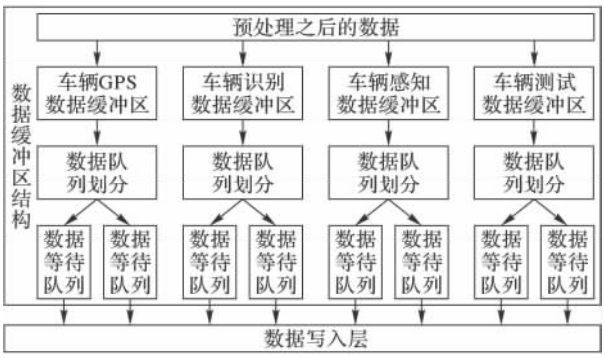

图 4　数据队列划分缓冲区结构

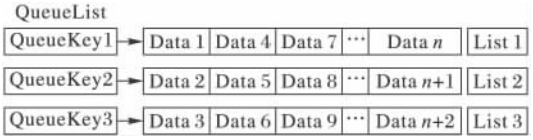

图 5　数据队列划分结构

划分完成后的队列依据窗口阈值向写缓冲区持续发送数据,客户端写缓冲区接受的数据在达到阈值后调用 HBase 多线程写入方法或者是 flushCommits( ) 方法,把数据发送到服务器作永久存储。在实际的项目应用中,面对每秒近 10 万的高流速数据,本文系统将写缓冲区大小设置为 6 MB(1 MB = 10^6 B) ,从而允许客户端更高效地将一定数量的数据组成一组,通过一个远程过程调用请求来执行。

2.2.2　行键(row key) 结构

智能交通系统需具备在线查询的能力,例如获得特定时间段内经过某一监测点的车流量以及车辆信息等。在 HBase 中,行键设计是支持在线实时查询的关键。良好的行键设计不仅仅能克服行键冗余带来的存储性能低下的缺点,也能够一定程度上解决写热点问题[3]。

实际应用中根据交通监管系统的需要,查询操作中常用到的数据属性是: 车牌号、监测点、监测时间,所以依次设计了 3 种行键设计方案。

方案 1　行键最初设计由车牌号、监测时间、监测点(CARNO + rTime + lawNO) 组成。由于车牌号是无序、散列的,所以此种方案设计好处在于可以将流数据较为均匀地实时存储到 HBase 集群数据库中。但是其不足之处在于几乎每次针对特定类型流数据的查询都要扫描整个集群数据库,尤其是对于特定时间段内的车流量的查询,其在扫描集群数据库的过程中使查询效率大打折扣。

方案 2　为了简化行键的格式,提高流数据插入和查询的效率,将行键设计为车牌号与监测点(CARNO + lawNO) 的组合,而把监测时间作为时间戳由数据写入时自动添加。这样设计的好处是: 1) 将行键的长度缩减为原来的一半,当大规模的流数据插入 HBase 集群数据库时其所占空间可以大大减少。2) 大范围从数据库中查询时,其监测时间作为时间戳出现,简洁易得,查询效率更高。但是这种设计降低了对数据属性监测时间的可控制性。

方案 3　针对以上设计的不足,进一步对行键进行了优化设计,即将监测时间放到行键起始端。但是在实际的测试中发现,由于流数据插入过程中 row key 是按照字典序存储的,所以行键以监测时间、车牌号、监测点(rTime + CARNO + lawNO) 组合的形式向 HBase 集群数据库中载入数据时,会出现热点。即流数据会按照监测时间(rTime) 的字典序依次插入集群中的一个 region 中,待此 region 达到一定的阈值后才会向其他 region 依次插入。毫无疑问此种方法使 HBase 集群数据库的插入性能大大下降。

为了消除热点,本文系统在已设计好的行键前端加一个单字节前缀,这种方法不仅解决了热点问题,同时保留了获取数据开始和结束行键的能力。具体地,行键表示为: 单字节前缀(0 ~ 15) + 监测时间(rTime) + 车牌号(CARNO) + 监测点(lawNO) 。此处单字节前缀的取值与2.2.4节中提到的预分区紧密相关,不同时间段中的流数据会根据哈希之后 row key 的取值存储到对应的已设定好起始 row key 范围的预分区中。

2.2.3　数据分布策略

数据写入区解析某个数据对象时,根据信息标示行键(row key) 来保证数据的快速定位。利用行键(row key) 以及已知的数据服务器列表,根据哈希计算将某个数据对象存储到对应的数据服务器上。一般通常使用的方法是使用 row key 对服务器数直接取余数来决定当前这个 row key 的内容发往哪一个服务器。但是使用这种方法不能够满足单调性,致使某台数据服务器负载过高。基于以上描述,本文使用一致性哈希算法来使每个服务器节点可以对等均匀地分布数据。

在流数据存储过程中使用一致性哈希计算时,数据写入线程将数据写入到哪台 HBase 集群数据库服务器不仅仅依赖信息标示 row key 本身而是将数据服务器的配置(在这里为 IP 地址或者机器名) 也进行 Hash 运算。集群数据库中服务器配置哈希前后在环空间的位置如图 6(a) 、(b) 所示。接下来使用如下算法定位数据访问到相应服务器,计算过程如下。

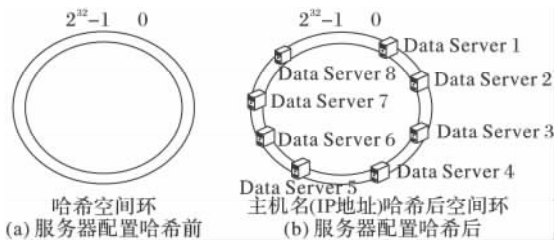

图 6　一致性哈希算法

1) 把数据库服务器 IP 地址(主机名) 进行 MD5 哈希,取其部分作为一个无符号整型。

2) 把数据子对象的唯一标示行键(row key) 同样作 MD5 哈希,得到一个整数。

3) 数据服务器的整数会根据大小形成一个数字环,而信息标示行键(row key) 的哈希则会分布在这些数字上或中间。

4) 如果信息标示行键(row key) 的哈希等于数据服务器的哈希,则把信息标示行键(row key) 数据片段存放在该数据

服务器上;否则,寻找第一个大于 row key 哈希的数据服务器,用于存放信息标示行键(row key)数据对象,从而节点的加入或退出仅影响相邻的节点。单个数据服务器节点只需要专注于自身的数据接收、发送与存储工作,从而实现将复杂的分布式架构逻辑与单个服务节点数据处理相分离,最大限度地利用现有的数据库存储技术。

2.2.4 数据服务器预分区

海量高并发数据的分布存储中使用一致性哈希算法可以使不同类型的车辆流数据在每个服务器节点上实现对等均匀地分布。但是在服务器节点中的 region 存储数据时,默认情况下,在创建 HBase 表的时候会自动创建一个 region 分区,当导入数据的时候,所有的 HBase 客户端都向这一个 region 写数据,直到这个 region 足够大了才进行切分,这样容易引起单节点负载升高,从而影响入库性能。而且实际运用中发现交通流数据速率与时间有一定的对应关系,比如每日的00:00—07:00 以及 21:00—24:00 中车流量相对于其他时间明显减少,每日的 07:00—21:00 车流量分布则相对均匀。

针对以上问题,结合实际应用中交通流数据的发送速率与监测时间线性相关的特点,考虑通过预先创建一些空的 regions 来提高批量写入速度,这样当数据写入 HBase 时,会按照 region 分区情况,在集群内作数据的负载均衡。本文实现的系统采用建表时预分区的方法,在一定程度上提升了流数据写入 HBase 集群数据库时的性能。基于 HBase 的交通流数据实时存储过程中,结合 2.2.2 节中行键的设计将预分区设为 16 个,且制定了每个分区的起始与结束行键的范围。

2.3 数据写入流程

一个完整的数据写入流程如图 7 所示,流程如下。

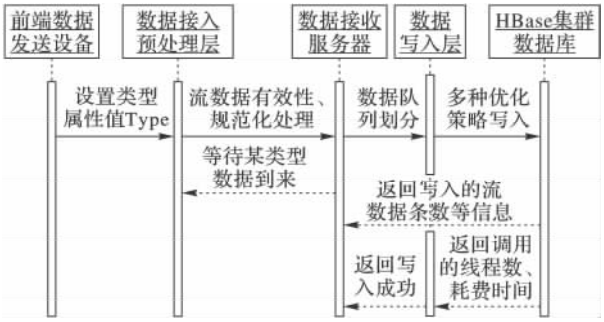

图 7 数据写入流程

1) 为不同类型源数据设定唯一标示,如设置代表其类型的属性值 Type,并配置其值,即拥有相同 Type 值的数据流为同一类型源数据。

2) 多源数据接入层预处理即对接收到前端设备发来的数据进行预处理,包括舍弃不完整的数据对象、对不合规范的数据对象进行标点或者格式整合等。

3) 多源缓冲区结构中的操作如下:

a) 数据队列划分接收服务器(以下简称数据接受服务器)一直监听预处理之后上送的数据,监测到数据开始被发送时,则立即接收数据并初始化存放数据的队列;

b) 如果某一类型的数据接收服务器没有监测到经预处理之后此类数据的到来,则一直等待,直到经预处理之后的该类型数据已发送;

c) 数据接收服务器对接收到的当前数据根据其时间戳属性计算出该条数据发往的对应队列,然后将该条数据临时存放到相应的队列中;

d) 如果某个队列数据到达已设定好的时间或者是队列阈值,数据接收服务器均给该数据对象配置标示其队列属性的信息 key,然后将其发送到客户端写入缓冲区中。

4) 多源数据写入层与存储层中的操作如下:

a) 客户端写入缓冲区接收到各类流数据的数目达到其阈值后,开始调用数据写入区线程为其设置唯一行键标示 row key;

b) 数据写入区获取该数据对象,根据信息行键标示 row key 和 HBase 集群数据库服务器主机名(IP 地址)列表,进行一致性哈希计算,根据最终的 Hash 值将该数据对象存放到对应的数据库服务器中。

3 实验设计及结果

实验目的 在 Oracle 集群数据库环境下,实验 DRTS 方法的实时存储性能;在 HBase 集群数据库环境下,实验基于 HBase 的交通流数据实时存储系统的存储性能。对相关结果进行对比与分析,期望验证本文系统在性能和扩展能力上具有先进性。

实验环境 4 台双核 3.0 GHz CPU 和 4 GB 内存的 Load Runner 服务器上安装 Load Runner 9.0,使用 Load Runner 9.0 模拟数据流发送端;在 1 台 2×4 核 2.4 GHz CPU,16 GB 内存的服务器上安装数据预处理程序,作为数据接入预处理服务器;在 1 台 2×4 核 2.4 GHz CPU,16 GB 内存的机器上安装存储处理程序;在 9 台 2×4 核 2.4 GHz CPU,8 GB 内存,1 TB RAID5 磁盘阵列的机器上安装 HBase-0.94,选中其中一台作为 Master 节点,使用 HBase 集群数据库作为持久化存储介质;在 8 台 2×4 核 2.4 GHz CPU,16 GB 内存,1 TB RAID5 磁盘阵列的机器上安装 Oracle 10g,使用 Oracle 10g 作为持久化存储介质。网络连接采用 1 Gb/s 以太网光纤和交换机。系统测试平台架构如图 8 所示。

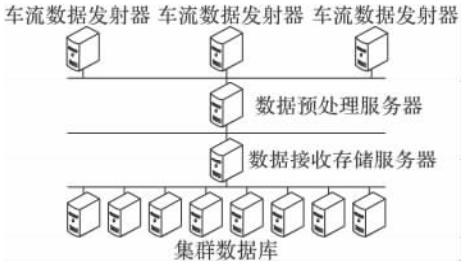

图 8 测试架构平台

实验 1 使用本文设计的系统进行数据存储测试。本文设计的系统对于数据写入区内部的操作采用单线程,行键(row key)设计不采取任何优化,只是监测时间、监测点、车牌号的组合。具体设置如下:使用 Load Runner 服务器模拟数据流发射时,设置不同的数据发送速率与不同大小的缓冲区阈值,分别进行测试,即缓冲区大小取值为 2 MB、4 MB、6 MB、8 MB,数据发送速率(单位是条/秒)取值为 1 000,5 000,10 000,15 000,20 000,25 000,30 000,35 000,40 000,45 000,50 000,…,100 000。窗口时间阈值设定为 2 s,持续运行 10 h。实验中每条数据长度为 100 B,定义项目中每条数据长度为 *Length*(单位为 B),缓冲区大小为 *Zone*(单位为 MB),缓冲区可以存储的数据条数为 *Count*,那么 $Count = Zone/Length$。所以缓冲区大小为 1 MB 时相当于可以存储 10 000 条实际车流量数据。实验结果中出现的延迟时间是指持续发送数据、实

时存储数据 10 h 后,系统未及时处理存储的数据全部存储到集群数据库所需要的时间。

实验结果如图 9 所示。

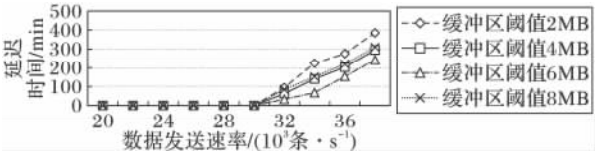

图 9　基于 HBase 的交通流数据实时存储系统测试结果 1

图 9 中,横轴为车辆流数据发送速率,即每秒发送的数据量(单位为条/秒)。纵轴为 HBase 集群数据库实时存储流数据时出现的延迟时间。由图 9 得知,系统中数据写入区内部的操作采用单线程写入且不对行键作任何优化时,其吞吐效率达每秒 30 000 条时出现延迟,且随着数据发送速率的增加,其延迟时间愈来愈长。由不同的缓冲区阈值也可以看出,当阈值设置为 6 MB 时是最佳状态,缓冲区阈值的设定增大或者减小都会使得实时数据的存储延迟时间增加。

实验 2　使用 DRTS 方法和本文设计的系统分别进行数据存储测试。本文设计的系统对于数据写入区内部的操作分别采用单线程、多线程写入,行键(row key)设计依然不采取任何优化,只是监测时间、监测点、车牌号的组合。具体设置如下: 写客户端使缓冲区阈值设置为6 MB,其他实验设置同实验 1。

由图 10 可看出,DRTS 方法在数据发送速率达到每秒 12 000条时,存储数据出现延迟,而且其延迟时间随着数据发送速率的增加而急剧增加,本文系统采用单线程测试时实验结果如实验 1 的图 9 所示,即其吞吐效率达每秒 30 000 条时出现延迟,而采用多线程技术写入时,数据库的吞吐率则得到了很大提升,在数据发送速率达每秒 100 000 条时还未出现延迟。由图 10 可知,HBase 数据库较 Oracle 数据库在数据存储层面有较好的实时存储性能。为了进一步测试本系统的性能,考虑在多线程写入的基础上作进一步优化: 消除行键的写入热点。

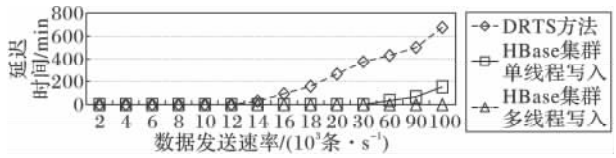

图 10　不同方法延迟测试结果

实验 3　使用基于 HBase 的交通流数据实时存储系统进行更高数据发送级别的存储测试,系统中对于数据写入区内部的操作采用多线程技术,行键设计采取设置单字节前缀的方法消除插入热点优化。缓冲区大小取值为 6 MB,窗口时间阈值设置与程序运行时间同实验 1。数据发送速率(单位是条/秒)取值为 30 000,40 000,50 000,60 000,70 000,80 000,90 000,100 000,110 000,…,140 000。

实验结果如图 11 所示。

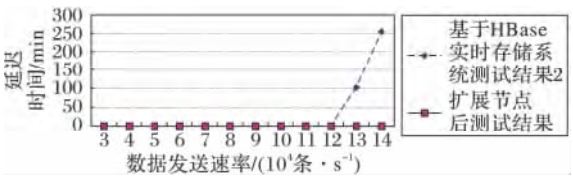

图 11　调优后系统延迟测试与可伸缩性测试结果

由图 11 中代表基于 HBase 实时存储系统测试结果 2 的折线可以看出,在行键优化设计的基础上采用多线程技术并发写入时,数据库的吞吐率大大提升。但是也可看出当数据速率不断提高到每秒 120 000 条的级别时,即使使用 HBase 集群数据库进行优化存储也出现了延迟现象,延迟时间随着数据发送速率的不断提高而增长。

现增加数据服务器至 10 台,再次进行同样测试,查看效果。由图 11 代表扩展节点后测试结果的折线所示,当增加两台数据服务器后,数据发送速率提高到每秒 140 000 条的级别时仍未出现存储延迟。由此说明,当数据速率不断提高时,若出现存储延迟现象,则可以适当增加数据服务器以减少或消除延迟。

4　结语

在基于 Oracle 的实时数据的存储方法基础上,本文提出了基于 HBase 的交通流数据实时存储系统。该系统采用分布式存储架构,通过前端的数据接入预处理层对流数据进行整合、规范化,利用多源队列划分缓冲区中的相关操作切分不同类型、不同时间点内的数据,在数据写入层和数据存储层结合一致性哈希算法、多线程技术、行键优化设计等策略将数据并行均衡地存储到 HBase 集群服务器中。该系统基于事实车辆数据进行了一系列的实验,结果证明基于 HBase 的该系统在大部分情况下拥有良好的扩展、存储性能。本文下一步工作将针对 HBase 流数据存储过程中列族和列值字段冗余对集群存储性能的影响作进一步分析,使该系统具有更高的使用价值。同时也将对 HBase 集群数据库服务器增加或者退出时,相应的数据迁移办法进行一定的研究,即增加节点或者减少节点时,在不丢失数据的前提下实现数据在集群数据库中的负载均衡。

参考文献:

[1] QI K, ZHAO Z, FANG J, *et al*. Real-time processing for high speed data stream over large scale data [J]. Chinese Journal of Computers, 2012, 35(3): 477 – 490. (亓开元,赵卓峰,房俊,等. 针对高速数据流的大规模数据实时处理方法[J]. 计算机学报, 2012,35(3):477 – 490.)

[2] SUN Y, FANG J, HAN Y. A distributed real-time storage method for stream data [C]// WISA 2013: Proceedings of the 10th Conference on Web Information System and Application. Washington, DC: IEEE Computer Society, 2013: 314 – 317.

[3] CHEN Q, ZHOU L. HBase-based storage system for large-scale data in wireless sensor network [J]. Journal of Computer Applications, 2012, 32(7):1920 – 1923. (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.)

[4] DEVINE R. Design and implementation of DDH: a distributed dynamic hashing algorithm [C]// Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms. Berlin: Springer, 1993: 101 – 114.

[5] LI Y, CHENG Y, LI B. The application of technology of multithready execute in data communication [J]. Journal of Chongqing University of Posts and Telecommunications: Natural Sciences Edition, 1997, 9(4): 29 – 31. (李毅,程运,李秉智. 多线程技术在数据通信中的应用[J]. 重庆邮电学院学报: 自然科学版,1997,9(4):29 – 31.)

(下转第 135 页)

体现出两者之间的差距,实际的通信需求大多介于这两个极端之间。“软件人”系统中大多数的基本通信如“软件人”的注册注销属于低情景依赖的通信需求,而当真正完成一项功能时却更接近于高情景依赖的通信需求。通信时间越长,情景驱动的“软件人”知识通信语言的优势越能更好体现。

4　结语

不同于分布式系统及多 Agent 系统,“软件人”系统中的个体具有拟人情感,因此“软件人”之间的通信有着区别于传统 Agent 通信特有的需求。传统的“软件人”通信语言在基于消息的基础上已经可以完成基本的通信需求,本文为在通信层为上层“软件人”的模糊推理提供有效的支持,提高通信成功率;在内容层之上细分出了专注于操作知识库的知识层;为了在软件人通信过程中维护情景上下文信息,提高通信质量及密度,引入了情景层,并从情景驱动的软件人知识通信框架层次结构中的消息层、知识层、情景层详尽地阐述了该框架的设计与实现。

参考文献:

[1] ZENG G, TU X, WANG H. SoftMan research and application [M]. Beijing: Science Press, 2007: 9 – 10. (曾广平,涂序彦,王洪泊. “软件人”研究及应用[M]. 北京: 科学出版社,2007: 9 – 10.)

[2] WANG H, ZENG G, TU X. Research on SoftMan systems and its application [J]. CAAI Transactions on Intelligent Systems, 2006, 1(1): 24 – 28. (王洪泊,曾广平,涂序彦. “软件人”系统研究及其应用[J]. 智能系统学报,2006,1(1): 24 – 28.)

[3] WANG H, ZENG G, WANG Z, *et al*. SoftMan group intelligent autonomous coordination model and its application [J]. Acta Automatica Sinica, 2007, 33(9): 924 – 929. (王洪泊,曾广平,王宗杰,等. 软件人群体智能自律协调模型研究与应用[J]. 自动化学报,2007,33(9): 924 – 929.)

[4] RAJI F, LADANI B T. Anonymity and security for autonomous mobile Agents [J]. IET Information Security, 2010, 4(4): 397 – 410.

[5] WANG L, WANG Z, YANG Y. Framework of multi-Agent system and consultation mechanism [J]. Application Research of Computers, 2012, 29(3): 852 – 855. (王鲁,王志良,杨溢. 一种多 Agent 系统框架与协商机制研究[J]. 计算机应用研究,2012,29(3): 852 – 855.)

[6] KONE M T, NAKAJIMA T. An architecture for a QoS-based mobile Agent system [C]// Proceedings of the 5th International Conference on Real-Time Computing Systems and Applications. Piscataway: IEEE, 1998: 145 – 148.

[7] NIKRAZ M, CAIRE G, BAHRI P A. A methodology for the development of multi-Agent systems using the JADE platform [J]. International Journal of Computer Systems Science and Engineering, 2006, 21(2): 99 – 116.

[8] SINGH M P. Agent communication languages: rethinking the principles [J]. Computer, 1998, 31(12): 40 – 47.

[9] DUAN G. Multi-Agent communication mechanism and strategic analysis [J]. Microcomputer and Its Applications, 2013, 32(2): 57 – 59. (段桂芹. 多 Agent 通信机制与策略分析[J]. 微型机与应用,2013,32(2): 57 – 59.)

[10] WANG L, YU W, HUANG G. Multi-Agent system with information fusion for smart home [J]. Journal of Computer Applications, 2014, 34(9): 2747 – 2751. (王良周,于卫红,黄广超. 基于信息融合的多 Agent 智能家居系统[J]. 计算机应用,2014,34(9): 2747 – 2751.)

[11] HAN Y, LIU F, REN Z. Research on land combat Agent communication mechanism model [J]. Fire Control and Command Control, 2010, 35(Supplement): 60 – 64. (韩月敏,刘非平,任重. 陆战 Agent 通信机制模型研究[J]. 火力与指挥控制,2010,35(增刊): 60 – 64.)

[12] YANG A. Research on communication action of multi-Agent based on fluent calculus [J]. Computer Engineering and Design, 2010, 31(1): 221 – 224. (杨爱琴. 基于流演算的多 Agent 通信动作的研究[J]. 计算机工程与设计,2010,31(1): 221 – 224.)

[13] AN B, LIU G. Predictive controller for networked multi-Agent systems with communication delay and packet loss [J]. Acta Physica Sinica, 2014, 63(14): 140 – 145. (安宝冉,刘国平. 带时延与丢包的网络化多智能体系统控制器设计[J]. 物理学报,2014,63(14): 140 – 145.)

[14] LAI J, CHEN S, LU X. Tracking consensus of nonlinear MASs with asymmetric communication delays in noisy environments [J]. Communications in Nonlinear Science and Numerical Simulation, 2014, 19(7): 2334 – 2344.

[15] XIANG J, LI Y, WEI W, *et al*. Synchronisation of linear continuous multi-Agent systems with switching topology and communication delay [J]. IET Control Theory & Applications, 2014, 8(14): 1604 – 1615.

(上接第 107 页)

[6] ZHONG Y, HUANG X, LIU D, *et al*. NoSQL storage solution for massive equipment monitoring data management [J]. Computer Integrated Manufacturing Systems, 2013, 19(12): 3008 – 3016. (钟雨,黄向东,刘丹,等. 大规模装备监测数据的 NoSQL 存储方案[J]. 计算机集成制造系统,2013,19(12): 3008 – 3016.)

[7] GAO J, DING W, ZHAO Z. A sensory data integrating bus under the intelligent transportation system [C]// WISA 2013: Proceedings of the 10th Conference on Web Information System and Application. Washington, DC: IEEE Computer Society, 2013: 328 – 332.

[8] WANG J. A multi-tenant data management method and its application in intelligent transportation system [D]. Beijing: North China University of Technology, 2014. (王金全. 一种多租户数据管理方法及其在智能交通中的应用[D]. 北京: 北方工业大学,2014.)

[9] GEORGE L. HBase: the definitive guide [M]. DAI Z, LIU J, JIANG J, translated. Beijing: People's Posts and Telecommunications Press, 2013: 339 – 350. (GEORGE L. HBase 权威指南[M]. 代志远,刘佳,蒋杰,译. 北京: 人民邮电出版社,2013: 339 – 350.)

[10] NISHIMURA S, DAS S, AGRAWAL D, *et al*. MD-HBase: design and implementation of an elastic data infrastructure for cloud-scale location services [J]. Distributed and Parallel Databases, 2013, 31(2): 289 – 319.

[11] CHANG F, DEAN J, GHEMAWAT S. Bigtable: a distributed storage system for structure data [J]. ACM Transactions on Computer Systems, 2008, 26(2): Article No. 4.

[12] CATTELL R. Scalable SQL and NoSQL data stores [J]. ACM SIGMOD Record, 2011, 39(4): 12 – 27.