一种传感数据的压缩和高效存储方案

祁 兰,毛燕琴,沈苏彬
(南京邮电大学 计算机学院、软件学院,江苏 南京 210003)

摘 要:传感网络数据经旋转门压缩后是零散分布的,压缩后的数据直接在云中存储,将导致其难以统一管理和查询,集群为了负载均衡会频繁移动数据。MongoDB 数据库是一种新型的非关系数据库,其存储结构灵活,查询效率高,适合传感数据的存储和管理。通过研究 MongoDB 的存储特点和空间分配机制,利用云平台的计算能力,设计针对传感数据的压缩和高效存储方案。先将传感器压缩后的传感数据在云中使用最小二乘法拟合进行解压缩处理,得到时间粒度上的完整数据集,设计一种针对时域性传感数据的通用存储格式,将数据集存储到高性能的 NoSQL 数据库 MongoDB 中。实验结果表明,该方案可以更好地恢复有损压缩后的数据并提高数据的查询效率,体现了 MongoDB 灵活的存储模式在传感数据中应用的显著优势。

关键词:传感数据;云计算;MongoDB;压缩算法
中图分类号:TP392 **文献标识码**:A **文章编号**:1673-629X(2016)11-0177-05
doi:10.3969/j.issn.1673-629X.2016.11.039

A Compressed and Efficient Storage Scheme of Sensor Network Data

QI Lan, MAO Yan-qin, SHEN Su-bin
(School of Computer Science & Technology, School of Software, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

Abstract: Since compressed wireless sensor network data based on Swing Door Trending (SDT) is scattered, unified management and query are difficult to handle, meanwhile the data would be moved frequently for the goal of loading balancing. MongoDB, as a new non-relational database, is famous for flexible storing configuration and high query efficiency, which is suitable for storage and management of sensor data. By studying the MongoDB storage features and space allocation mechanism, taking advantage of the ability to cloud computing platform, a compressed and efficient storage scheme of sensor network data is proposed. The sensing data compressed by sensors is decompressed in the cloud using the least squares fitting, getting the complete data set on time granularity, and a common storage format applies to the time domain senor data is designed to put the data into the MongoDB, a high-performance NoSQL database. Experimental results show that the scheme recoveries compressed data better and promotes the database query efficiency, and the flexible storage mode of MongoDB plays a significant advantage in sensing data applications.

Key words: sensing data; cloud computing; MongoDB; compression algorithm

0 引 言

物联网是继互联网和移动互联网后世界信息产业的第三次浪潮,2001 年,NASA 第一次定义了传感器网络[1]。2005 年,开放地理空间信息联盟(Open GIS Consortium,OGC)制定了一套传感网络标准。数据显示,到 2020 年,全球接入物联网的传感器终端将达到 500 亿个,在物联网行业的应用日益广泛[2]。

传感数据不仅可以在实时处理和监控中使用,历史数据还可以预测未来的新趋势[3],但由于传感器收集的数据量大,无线传输耗能大,数据种类和分布复杂,传感器精度有限,部分传感器误差较大,传统的关系型数据库在处理传感数据时面临严重挑战。使用云存储技术不仅能够解决传感数据的扩展和管理问题,而且还能提高数据存储的可用性、稳定性和安全性。选取合适的统一完整的存储模式有利于传感数据的统一管理和高效查询[4]。

对于传感数据的传输和存储,一般采用的解决方案是将传感数据压缩后存储,以降低传感器网络的功

收稿日期:2016-01-16 **修回日期**:2016-04-21 **网络出版时间**:2016-10-24
基金项目:江苏省未来网络前瞻性研究资助项目(BY2013095-1-08);南京邮电大学自然科学基金(NY211115)
作者简介:祁 兰(1991-),女,硕士研究生,研究方向为计算机网络;沈苏彬,博士生导师,研究方向为计算网络、下一代电信网及网络安全。

耗。但是如果采用有损压缩,那么得到的数据集是零散分布的。然而云存储在存储大数据的核心思想为化零为整、预先分配、统一管理,这与有损压缩后数据的零散性相矛盾,所以文中提出先在传感器上压缩数据,后在云中解压缩,然后使用嵌套格式统一存储的方案。

文中首先介绍了传感数据的压缩技术、云存储技术和 MongoDB 数据库,然后讨论了各需求之间的矛盾和取舍理由,并详细介绍了方案设计的实现过程,最后对方案结果进行了测试和分析。

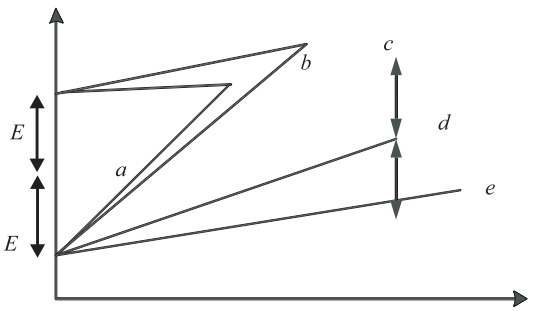
1 相关技术分析

1.1 传感数据的压缩技术

文献[2]中研究表明,传感器节点在数据传输上的能耗比在数据运算上的能耗大得多,在同等条件下,发送 1 位数据所需要的能量甚至是执行 1 次加法运算所需要的能量的 500 倍左右,所以传感器节能问题是传感网络设计需要考虑的一个重要问题,传感器节点上的压缩算法需要压缩率高但又不能太复杂。由于传感器节点采集数据时容易受外界环境偶然因素的影响,并且现有的传感设备的精度非常有限,所以通常不是以传感器节点某一时刻的数据作为判断依据,而是以一个时间段的数据序列作为判断依据,所以线性回归压缩适用于这一类时序性数据的压缩。

针对传感数据,研究者们在经典压缩算法的基础上提出了一些针对传感数据的压缩方法,主要有小波压缩算法、分段常量近似算法、感知压缩算法等[5]。其中旋转门压缩算法比较简单且压缩性能好,最适合传感数据的压缩。

旋转门压缩算法如图 1 所示[6]。在数据 a 点上下有两个像门轴一样的点,这两个点与当前点组成两条线段 E,就像绕着这两个点的两个门渐渐打开,每个门都只能向外打开而不能向内关。当两扇门张开的角度之和不超过 180°时,旋转门继续向外开;若超过 180°,保留上一个数据点,以保存的点为起点开始下一轮的旋转门压缩,起点与终点之间的点全部舍弃。图 1 中执行旋转门压缩后,以 a 与 d 的连线替代 a 到 d 这 4 个点的变化趋势。



万方数据 旋转门压缩算法原理

但是旋转门解压缩过程过于简单,直接用直线连接两点的线性插值法来还原中间数据,不能很好地还原数据的变化趋势,而还原阶段数据已传输到云上,可以利用云的计算能力,能量消耗问题不再是瓶颈,所以可以使用更好的最小二乘法拟合来改进旋转门压缩算法的解压缩部分。

1.2 云存储技术

云计算已成为一种流行的商业模式,它是作为一个实用程序或服务,为互联网上的客户提供计算资源或存储资源等,目的是为了实现资源的虚拟化。云存储是云计算的服务之一,它为客户提供虚拟化的存储需求。供应商可以使用低成本的物理存储器设备灵活扩展,实现更多的功能,而不需要购买额外的设备[7]。

云存储成为处理大数据的首要选择,是因为它能够按照用户需求定制存储空间,而且还具有扩展灵活、管理方便、价格低廉、稳定安全等优点,为数据存储和管理提供了新方式和新思路。物联网中的传感数据使用云存储便于扩展和统一管理,为挖掘物联网大数据的价值提供了条件。

云存储技术综合了虚拟化技术、并行处理、网格计算、集群应用、分布式文件系统等,将各种不同类型的物理存储设备组织到一起,共同对应用层提供服务,通过软件技术实现不同存储设备的统一管理和调度,对外提供存储服务[8]。

其中非关系数据库(NoSQL)是一类典型的云存储,它是为了解决数据的海量性和异构性,主要体现在:

(1)高并发读写;

(2)海量数据的高效存储和访问;

(3)高可扩展性和可用性。

NoSQL 种类繁多,Strauch 等[9]根据数据模型将 NoSQL 数据库分为三种类型:键值存储、文档数据库和列式数据库。其中,MongoDB 属于文档数据库,但是它的功能非常丰富,使用的 BSON 存储格式具有非常灵活的扩展性。

1.3 MongoDB 数据库

自 1970 年 Edgar F. Codd 提出关系数据模型以来,它就成为了存储数据的主要方式。到了 20 世纪 90 年代中期,互联网数据越来越多,结构也发生了变化,1996 年提出的 XML(eXtensible Markup Language)是复杂网状结构和无模式数据的一种解决方案。

然而 XML 对于半结构化数据的建模和处理仍不够理想,2001 年,Douglas Crockford 提出了一个比 XML 更简单的变体:JavaScript Object Notation (JSON)[10]。JSON 是序列化格式,不是逻辑数据模型,它能对复杂的网状结构和无模式数据的记录建模。和 XML 不同

的是,它只能对记录建模,不能对人类生产的文档建模,所以 JSON 更简单、更高效。

MongoDB 公司将 JSON 又进一步优化,产生了另一种数据格式:BSON(Binary JSON)。BSON 拥有更好的性能,主要表现为遍历速度更快、操作更简单、数据类型更丰富。

与关系数据库的扁平结构不同,MongoDB 是通过对键值对进行不同程度的扩展,通过嵌套(embedded)、引用链接(reference link)等方式组成更为复杂的存储模式[11]。嵌套是将一个文档包裹一个子文档,引用链接则是使用 MongoDB 的 DBRef 对象或自定义外键来建立文档和文档之间的引用关系。MongoDB 提供了两种复制模式,主从复制(master-slave replication)和复制集(replica set)。分片技术则可以实现数据库的水平扩展,而对上层应用是透明的,不影响集群中的其他服务器。MongoDB 提供自动分片和负载均衡的功能,这是它相比于关系数据库的另一大优势。

MongoDB 对物联网数据存储和扩展有着很好的支持,为了提供更快的分析查询能力,在 MongoDB 中使用预聚合文档可以直接获得一些常用的统计数据[12]。预聚合文档(pre-aggregated documents)是指在存储时不仅存储每个数据值,还附加存储一些简单常用的聚合值,如各类平均值、最大值和最小值等。使用预聚合文档来查询平均每分钟/小时/天更容易且更快,增强了更新和查询性能。在 MongoDB 中,为了提高查询效率,避免文档在分片间的频繁移动,一般采取预分配的方式存储,查询和存储效率都会大幅提高,但是在传感器端为了降低传感器的能耗,数据是经过压缩的,所以不能确定 MongoDB 预分配空间大小,必须有一套完整的数据集才能确定预分配空间。

2　方案设计

由于传感器采用无线传输,能量有限,数据通信能量消耗远高于其数据运算的能量消耗,所以传感器节点的数据必须在传输前进行压缩处理。旋转门压缩是一种有损压缩,但比无损压缩的能量消耗小得多,且传感数据是一种时序性数据,用户关心的是数据的整体变化过程,所以采用旋转门压缩更合适。

MongoDB 数据库采用时间粒度的嵌套存储查询效率高,但是需要时间粒度上一个完整的数据集,而经旋转门压缩后的数据已经缺失了一些数据,所以在存入 MongoDB 之前需要先进行解压缩处理。旋转门压缩算法的解压缩直接采用线性差值法,这是为节省能量的一种简单处理,不能有效剔除某些因素引起的异常值,而利用云的计算能力,采用最小二乘法拟合能更好地反映传感数据时间的整体走势。图 2 所示的方案设计能同时满足以上需求,在传感器上采用旋转门数据压缩能降低传感器能量消耗,在云服务平台采用最小二乘法拟合对数据进行解压缩处理后可以得到时间粒度上的完整数据集,最后以时间粒度上的嵌套模式将完整数据集存入 MongoDB 数据库中。

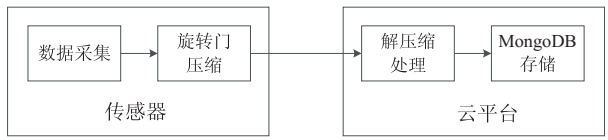


图 2　压缩存储方案示意图

2.1　传感数据的压缩处理设计

旋转门压缩算法(SDT)是一种典型的线性回归压缩算法,原理是对原始数据进行一元线性回归,去掉一些变化趋势不明显的数据,在误差范围内以直线替代原来的走势。旋转门压缩在数据量大且变化比较稳定时可以达到很高的压缩比,计算简单,节省能量,在传感器端可以取得很好的效果。

但是当数据传到云上后,云中可以提供强大的计算力,不再受能量的限制,其线性插值法解压缩原理并不适用于云存储环境。如果直接利用旋转门的线性插值法来解压缩,可能会因为数据中断而出现失真,而且由于现有的传感设备精度非常有限,测量出的传感数据准确度不高,有的点甚至离实际值差距很大,这样的点在工程中称为“噪声”。若用线性插值法直接解压缩,会将“噪声”带进近似函数,不能较好地描述数据整体的变化趋势。解压缩时利用最小二乘法拟合的曲线比用直线连接更贴合实际的数据变化走势。

曲线拟合是指“对一个复杂函数 $f(x)$,求出一个简单的便于计算的函数 $p(x)$,使 $p(x)$ 与 $f(x)$ 的误差在某种度量意义下最小”。它不要求函数必须穿过每个节点,只尽可能反映给定的这些点的基本走势,根据求得的这个函数补全中间的数据点完成解压缩,能更好地反映传感数据整体的走势,从而过滤掉一些经旋转门压缩保留下来的异常点。

2.2　云存储的数据模式设计

传感数据的快速增长迫使 MongoDB 将文档从内存中移动到磁盘,分散的存储将导致访问速度变得更慢,MongoDB 不得不扫描更多的文档,内存与磁盘进行频繁的交换来实现查询,降低效率。如果使用类似关系数据库的扁平存储模式会产生大量相同的域和属性,重复率高,对时间建立索引粒度太小,查询效率也不高。扁平模式存储示例如下:

{"deviceId":"0x3424","temperature":"18","humidity":"43","Time":"2014/12/11 04:17:37"}

MongoDB 中 BSON 的存储模式灵活,可以随意增加或删除文档中的域,也可以在一个文档内嵌入多层

子文档,特别是对时间的嵌套,因为 BSON 的数据结构大小是固定的,扫描时可以根据子文档的大小直接跳过前面的子文档查找,减少了查询引擎检索文档的次数,也减少了对重复时间的存储空间。

对于传感器设备信息这种数据较大、变动较频繁、一致性要求高的数据存储,如果直接嵌套存储,查询和更新的性能都会比较低,而应该使用引用链接来表示文档之间的关系。

综上所述,在 MongoDB 使用以时间为粒度建立的传感数据文档可以很好地解决上述问题。这种文档适用于几乎所有时序性的传感网络数据,合理使用嵌套和引用关系,使文档简洁、层次清晰,便于各层次查询、分配空间和统一管理。以小时为粒度的嵌套存储示例如下:

```
{
"deviceId": ObjectId( $sensor._id) ,
"timestamp": "2014.12.11T04",
"temperature":{
0:{0:18,…,58:18},
…,
59:{0:21,…,59:20},
"avg": arg_val,
"max": max_val,
"min": min_val
}
}
```

3 系统实现

3.1 旋转门压缩和解压缩处理

旋转门压缩算法通过不断计算上下两个门的斜率,同时更新最大上斜率和最小下斜率,当旋转门的最大上斜率大于最小下斜率时,即认为当前点不能拟合到这一段线性函数中,保存当前点的前一个点,并以此点为起点进行下一轮旋转门压缩。

解压缩时采用最小二乘法(Method of Least Squares)进行曲线拟合(Curve Fitting),它是一种用函数逼近的方法拟合,能很好地反映曲线变化,达到更好的解压缩效果。拟合曲线对应的近似值 y_i^* 和实测值 y_i 的差称为残差(Residual):

$$\delta_i = y_i - y_i^*$$

残差的值是衡量拟合好坏的重要指标,通常采用三种标准衡量:

(1)残差的最大绝对值最小;

(2)残差的绝对值之和最小;

(3)残差的平方和最小[13]。

因为准则(1)和准则(2)中都含有绝对值,不便于数学计算,所以准则(3)是实际应用中最合适的衡量指标,用它来确定各个参数是最合适的。用它计算得到拟合曲线的方法称作曲线拟合(或数据拟合)的最小二乘法,问题即为:根据已有的数据对 $(x_i,y_i)(i=1,2,\cdots,n)$,求一个近似函数,使得 $\sum_i^n \delta_i^2 = \sum_{i=1}^n [y_i - \varphi(x_i)]^2$ 最小。

令 $F = \sum_{i=1}^n [y_i - \varphi(x_i)]^2$,即要求出多项式的系数,使 F 取极小值。对 F 各个参数求偏倒,得到一个方程组,问题转化为求解此方程组。当各个参数线性无关时,方程组有唯一解,即 F 有极小值,从而得到拟合的函数 $p(x)$。

3.2 MongoDB 的存储和查询

文中采用 Java 开发,使用 MongoDB 的 Java 驱动操作 MongoDB 数据库,MongoDB 中 BSON 格式的文档模式的转化需要使用它的基本单位 BasicBSONObject 类来修改。由于它的构造函数 BasicBSONObject(String key,Object value)中的 Object 类可以为任意类型,也包括了 BasicBSONObject,它的方法 get(String key)返回的 Object 类也能转化为任意类型,如此通过不断迭代即可完成嵌套。

MongoDB 中的嵌套查询需要使用 MongoDB 的正则表达式“$regex”查询日期和小时,需要返回的域用内部类的方式获取,使用 JavaScript 查询 2015.12.11T11:0:0 的温度的示例如下:

```
>db.collection_name.find({"deviceId":"device1","timestamp":{$regex:"2015.12.11T11"},"temperature.0.0":{"$exists":1}},{"temperature.0.0":1})
```

DBRef 与关系数据库中外键的概念类似,使用 DBRef 引用传感设备信息文档 sensor 时,获取文档的_id 值,与需要嵌入的域名构造 DBRef 对象,代码如下所示:

```
{"deviceId":[new DBRef(device1, sensor._id)]}
```

4 测试和分析

4.1 解压缩结果测试

实验数据为 2015 年 10 月 18 日实验室每隔 1 分钟收集一次的温度数据,使用旋转门压缩算法压缩,参数 E 取 0.2,压缩前为 1 440 个数据,压缩后为 138 个数据,压缩率为 90.4%,最小二乘法拟合阶数取 4。用 Matlab 画出的两种解压缩效果对比图如图 3 所示。其中旋转门压缩用线性插值法还原,曲线以变化幅度大的点为导向,可能会由于某些环境因素影响局部温度,或因传感器的性能和精度而产生“噪声”,而最小二乘法拟合出的曲线则主要反映了传感数据的整体走势,消除了一些经旋转门压缩保留下来的异常点的影响,对于时序性数据更有参考价值。

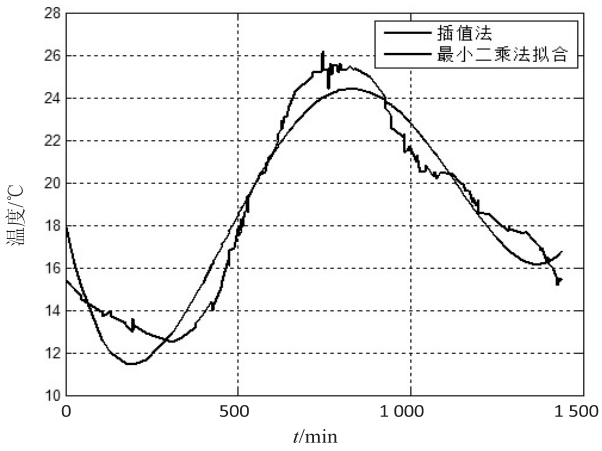


图 3　最小二乘法拟合和线性插值法解压缩对比图

4.2　MongoDB 查询测试

相比于没有嵌套的扁平的形式存储,以对时间粒度嵌套的形式存储查询效率要高很多,使用系统性能工具 benchrRun 测试 5 组数据量较大的随机查询,实验结果如图 4 所示。

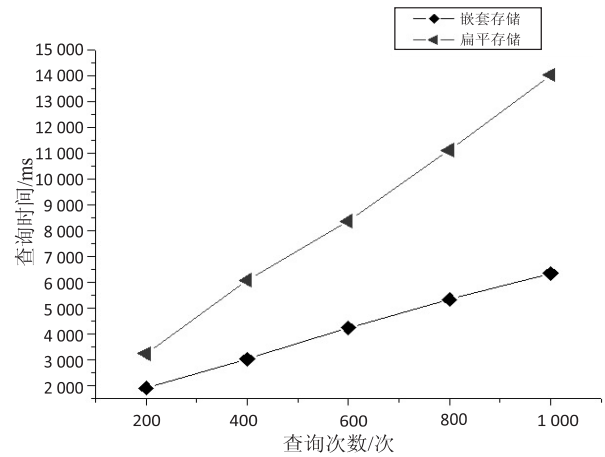


图 4　MongoDB 嵌套和扁平存储的查询效率对比图

MongoDB 使用时间粒度下的多层嵌套,能大大增强查询效率,减少了插入引擎的负荷,虽然增大了更新引擎的负荷,但更新时已分配好空间,所以避免了文档移动。对于数据小但数量多且杂的传感数据,使用这种统一的存储格式有利于查询和管理。

5　结束语

文中对传感数据的存储和查询遇到的挑战进行阐述,分析了 MongoDB 数据库在解决存储大数据时采取的方案。为了在降低传感器能量消耗的同时保证时间粒度的完整数据集,提出先对传感数据有损压缩,再利用曲线拟合进行解压缩,并设计了一种高效存储和查询的存储格式。测试结果表面,该方案可以更好地恢复有损压缩后的数据并提高 MongoDB 的存储和查询效率。

参考文献:

[1] Liu Q,Mao S,Li M,et al. Release and storage of mine gas monitoring data based on sensor web[C]//22nd international conference on geoinformatics. [s. l.]:IEEE,2014:1-6.

[2] Kimura N,Latifi S. A survey on data compression in wireless sensor networks[C]//International conference on information technology:coding and computing. [s. l.]: IEEE, 2005:8-13.

[3] 应蓓华. 用于无线传感网的低能耗数据压缩[D]. 北京:清华大学,2010.

[4] 孙韩林,张　鹏,闫　峥,等. 一种基于云计算的无线传感网体系结构[J]. 计算机应用研究,2013,30(12):3720-3723.

[5] 曾凡文. 云存储环境中传感数据的压缩存储处理研究[D]. 南京:南京理工大学,2012.

[6] 宁海楠. 一种基于 SDT 算法的新的过程数据压缩算法[J]. 计算机技术与发展,2010,20(1):25-28.

[7] 王意洁,孙伟东,周　松,等. 云计算环境下的分布存储关键技术[J]. 软件学报,2012,23(4):962-986.

[8] He Q,Li Z,Zhang X. Analysis of the key technology on cloud storage[C]//International conference on future information technology and management engineering. [s. l.]: IEEE, 2010:426-429.

[9] Kuznetsov S D,Poskonin A V. NoSQL data management systems[J]. Programming & Computer Software,2014,40(6): 323-332.

[10] Fourny G,Florescu D. JSONiq:the history of a query language [J]. IEEE Internet Computing,2013,17(5):86-90.

[11] Mongo DB data modeling introduction[EB/OL]. 2015-12-28. https://docs. mongodb. org/manual/core/data-modeling-introduction/.

[12] Copeland R. MongoDB 应用设计模式[M]. 陈　新,译. 北京:中国电力出版社,2015.

[13] 薛　毅. 数值分析与实验[M]. 北京:北京工业大学出版社,2005.