文章编号:1671-4229(2021)03-0069-11

面向时序大数据的数据库性能研究

李庆宇，王松波，林伟伟*
(华南理工大学 计算机科学与工程学院，广东 广州　510006)

摘　要：在大数据背景下,物联网技术的广泛应用产生了大量的时序数据。如何合理选择最适合的数据库来存储时序大数据是一个重要的研究内容。然而现有的数据库性能对比研究没有考虑数据的具体应用场景,缺乏特定场景下的性能对比实验。为了在存储时序大数据时能够从不同存储结构的数据库中选择最适合当前场景的数据库,文章对关系型数据库、本地 NoSQL 数据库以及公有云 NoSQL 数据库在燃气大数据的应用场景下进行了定量定性的实验与分析。实验结果表明,相比于关系型数据库,NoSQL 数据库更加适合存储时序大数据。然后,进一步提出了针对不同应用场景下时序大数据的数据库选型建议。

关键词：时序大数据；NoSQL；TSDB；公有云
中图分类号：TP 311.13　　**文献标志码**：A

Research on database performance for time series big data

LI Qing-yu, *WANG Song-bo*, *LIN Wei-wei**
(School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China)

Abstract: In the age of big data, the widespread application of the Internet of Things (IoT) technology has produced a large amount of time series data. How to choose the most suitable database to store time series data is an important research content. However, the existing database performance comparison studies do not consider the specific application scenes of data, and lack performance comparison experiments in specific scenes. In order to choose the most suitable database for the current scene from the databases with different storage structures when storing time series big data, we conduct quantitative and qualitative experiments and analyses on relational databases, local NoSQL databases and public cloud NoSQL databases in the application scenes of natural gas big data. Experimental results show that NoSQL databases are more suitable for storing time series big data than relational databases. Furthermore, the database selection suggestions for different application scenarios are proposed.

Key words: time series data; NoSQL; TSDB; public cloud

过去在传统领域中,关系型数据库凭借其结构简单易懂、使用操作方便、查询灵活和易于维护等优点得到了广泛的应用。但是随着物联网及相关技术[1]的快速发展,工业生产中比如电力、能耗和燃气等领域由于 24 h 不间断的生产运行,产生了大量的时序数据[2]。时序数据即时间序列数据,其最大的特征为数据按照时间先后排列。常见的时序数据有 CPU 使用率、股票价格和各类工业传感器的读数,时序数据通常单条数据长度不大,但是数据量很大,每天采样数超过百万,并且存储每年的采样数据所需要的容量接近 TB 级别。对于部分中小型企业,仍需将这部分数据存储在关系型数据库当中。

关系型数据库是以表形式组织的数据项集

基金项目：国家自然科学基金资助项目(62072187,61872084)；广东省基础与应用基础研究重大资助项目(2019B030302002)；广州市科学研究计划重点资助项目(202007040002)

作者简介：李庆宇(1996—),男,硕士研究生. E-mail:201921041974@mail.scut.edu.cn
* 通信作者. E-mail:Linww@scut.edu.cn

引文格式：李庆宇，王松波，林伟伟. 面向时序大数据的数据库性能研究[J]. 广州大学学报(自然科学版),2021,20(3):69-79.

合,从中可以以多种不同的方式访问或重新组织数据[3]。在表中的每行包含对应数据类别的唯一数据实例。当创建一个关系型数据库时,数据之间的约束与关系,使其成为“关系”表。大多数关系型数据库都使用结构化查询语言(SQL)来查询和修改存储在数据库中的数据。关系型数据库在设计时通过范式设计,可以合理地将数据库冗余缩小,但是,范式处理也使得稍微复杂一点的查询需要使用大量的联表查询(即连接操作),这反而导致了查询性能的下降。

随着时序数据的爆发式增长,数据量也随之达到千万级乃至更高水平。面对这些时序大数据,传统的关系型数据库所暴露出来的性能问题越来越严重,开始难以满足用户在数据读写时的低延迟需求。在生产环境中,从关系型数据库中查询一条特定的时序数据经常需要耗费数 10 s 的时间,随着数据量的增加,这个时间还会进一步增加。

为了脱离“关系”的桎梏,获得更好的性能,NoSQL 数据库应运而生[4]。NoSQL 数据库抛弃了“关系”的特点,它在许多重要方面与关系型数据库有很大的不同,其中最重要的是 NoSQL 数据库不再使用关系表作为其存储结构。除此之外,NoSQL 数据库还支持水平扩展,但不再使用 SQL 作为其查询语言,具有无法执行连接操作、不保证 ACID 的属性。2000 年 Eric Brewer① 提出了著名的 CAP 理论,即在分布式系统中无法同时满足一致性、可用性和分区容错性这 3 大特征,只能满足其中 2 者。

(1)根据 CAP 理论以及 NoSQL 数据库侧重点的不同,可以将 NoSQL 数据库分为 3 类[5]:

1)满足“CA”,即不关心分区容错性,主要保证一致性和可用性的数据库。

2)满足“CP”,即将数据存储在分布式节点中,保证数据的一致性和分区容错性,但是对于可用性的支持不够好。HBase、MongoDB 等数据库属于此分类。

3)满足“AP”,即通过牺牲一致性来确保可用性和分区容错性。Cassandra、CouchDB 等数据库属于此分类。

(2)根据 NoSQL 数据库存储结构的不同,NoSQL 数据库也可以分为以下 4 类[6]:

1)键值对数据库。键值对数据库存储的值对应于特定的键,键可以是合成的或自动生成的,而值可以是任何对象类型。存储时使用唯一的键和指向特定数据项的指针组成的哈希表,键是检索存储数据的唯一方法。常见的键值对数据库包括 Redis 和 Dynamo。

2)列存储数据库。列存储数据库将同一列的数据存在一起,适合处理存储在非常大的集群上的大量数据,因为可以非常有效地对数据进行分区。常见的列存储数据库包括 Cassandra 和 HBase。

3)文档型数据库。在文档型数据库中,数据依然使用键值对进行存储,但是数据被压缩为文档。文档存储没有任何模式限制,支持对具有不同类型键值对的记录进行多属性查找。常见的文档型数据库包括 MongoDB 和 CouchDB。

4)图形数据库。图形数据库使用图结构存储数据,包括节点、边和属性,节点和边由带有嵌入键值对的对象组成。图形数据库专门用于高效管理、高度连接的数据,如社交网络、推荐系统等。常见的图形数据库包括 Neo4j 和 HyperGraphDB。

在众多 NoSQL 数据库中,快速发展的时序数据库为管理时序大数据提供了一个良好的解决方案[7]。时序数据库是针对时间序列数据进行存取优化的数据库,时序数据库以其对时间序列数据的独特支持而被广泛应用于互联网、物联网等领域,并且其应用场景和范围还在不断扩大。针对时序数据写入频率远高于查询频率、数据更新少、查询多基于时间段等特点,时序数据库大多提供了监控、降采样和聚合等功能,方便对时序数据进行分析计算与可视化。

目前,虽然已经有较为成熟的数据库实现对时序数据的优化存储[8-10],但部分企业因缺乏对时序数据库的使用场景及其在存储时序大数据时性能优势的了解,依旧使用关系型数据库来存储和分析时序大数据。在现有的关于数据库性能对比研究中,有对关系型数据库和 NoSQL 数据库进行性能对比的,也有对各种不同存储结构的 NoSQL 数据库进行性能对比的。如 Li 等[11]将 MongoDB、Cassandra 等多种 NoSQL 数据库和关系型数据库 Microsoft SQL Server Express 进行性能对比,分析这些数据库在进行增、删、读等操作的性能差异。Abramova 等[12]对比了 HBase、Redis 等多种 NoSQL 数据库在增、删、改、查等操作的性能,但是

① CAD 理论由 Eric Brewer 于 2000 年的一次邀请谈治 Towards robust distributed systems 中提出。

这些文献缺少在具体场景下的数据库性能对比实验,因此,针对特定场景下的数据库选型并没有很大的参考价值。Rautmare 等[13]研究了在小型物联网应用场景下 MySQL 数据库和 MongoDB 数据库的性能差异,Klein 等[14]研究了在大型医疗保健系统应用场景下 MongoDB 数据库、Cassandra 数据库和 Riak 数据库的性能差异。尽管这些文献实现了特定场景下不同数据库的性能对比试验,但是所进行比较的数据库类别较少,目前的相关研究也很少综合对比并分析关系型数据库、本地 NoSQL 数据库和公有云 NoSQL 数据库这几类数据库在存储和分析时序大数据时的性能差异[15]。因此,本文设计并实现了基于燃气大数据应用场景下不同种类数据库的性能对比实验,通过比较和分析这几类数据库在数据写入和查询的性能差异,进而提出几点关于面向时序大数据的存储、分析与计算的数据库选型建议。

(3)本文的主要贡献如下:

1)介绍了现有的主流数据库性能测试工具,如 SysBench、HammerDB、YCSB 与 Apache JMeter 的使用和原理,充分分析了它们之间的优势和劣势,并从中选择一款合适的性能测试工具以实现本文所设计的对比实验。

2)使用 Apache JMeter 工具对 MySQL、HBase、OpenTSDB 和阿里云 TSDB 数据库进行了性能测试,实验分析了这 4 类数据库在基于不同数据量与线程数时数据写入与查询的吞吐量。实验结果表明,在数据量较大的情况下,相比于关系型数据库,NoSQL 数据库更加适合存储时序大数据。

3)最后根据数据库性能对比结果,针对不同应用场景下时序大数据的存储提出了相应的数据库选型建议。

本文后续章节结构安排如下:第一节主要介绍关系型数据库和 NoSQL 数据库常用的性能评测工具并针对本文提出的场景设计一种性能评测方案。第二节为数据库性能测试实验,主要测试 4 类数据库在基于不同数据量和启动线程数的性能表现,并针对不同应用场景给出数据库选型建议。最后总结本文的工作并提出未来改进的方向。

1　基于时序大数据的数据库性能评测方案

性能评测的主体思路是围绕以 MySQL 为代表的关系型数据库、以 HBase① 为代表的本地 NoSQL 数据库、以 OpenTSDB② 为代表的时序数据库和以阿里云 TSDB③ 为代表的公有云时序数据库,并依托燃气企业监控大数据,设计若干组对比实验,探究时序大数据在不同数据库上的存取性能差异,并提出不同场景下的数据库选取方案建议。

1.1　主流数据库性能测试工具

(1)SysBench④

SysBench 是一个基于 LuaJIT 的脚本化多线程基准测试工具[16-17]。它最常用于数据库基准测试,但也可用于创建其他任意的复杂工作负载。SysBench 包括类 OLTP 数据库基准测试集合、文件系统级基准测试、CPU 基准测试、内存访问基准、基于线程的调度程序基准和 POSIX 互斥量基准共 6 大类基准测试。

SysBench 使用非常简单,通过在用户提供的 Lua 脚本中实现预定义的钩子,可以轻松创建新的基准测试,也可以使用通用的 Lua 解释器,只需要将脚本中的路径修改即可。SysBench 能够每秒生成和跟踪数亿个事件,即使有数千个并发线程,开销也很低。统计方面,SysBench 可以以百分位数和直方图的形式提供有关速率和延迟的统计数据。SysBench 支持 MySQL 数据库与 PostgreSQL 数据库。

(2)HammerDB⑤

HammerDB 是一个数据库负载测试和基准测试工具。使用 HammerDB 可以创建测试模式,加载数据,并针对事务性和分析性场景针对数据库模拟多个虚拟用户的工作负载。然后通过工作负载获取 HammerDB 提供的监控指标 TPM,即吞吐量,单位为每分钟的事务数。HammerDB 是可视化的基准工具,拥有图形用户界面,从 3.0 版本起引入了命令行版本。

HammerDB 支持 Oracle、Microsoft SQL Server、

① https://hbase.apache.org/

② http://opentsdb.net/overview.html

③ https://help.aliyun.com/document_detail/55652.html

④ https://github.com/akopytov/sysbench

⑤ https://hammerdb.com/

IBM Db2、PostgreSQL、MySQL 以及 MariaDB 等关系型数据库。NoSQL 数据库中 HammerDB 支持 Hadoop 上的 Redis 和 Trafodion SQL,但是由于 HammerDB 设计的工作负载基本上专门为关系型数据库设计,因此,没有支持更多的 NoSQL 数据库。

(3)YCSB①

YCSB(Yahoo! Cloud Serving Benchmark)是由雅虎公司研发的一款开源的、面向新一代云数据服务系统性能比较的基准工具[18],其最大的特点在于其可扩展性强,即用户可以便捷地实现对各种数据库系统进行基准测试,同时也可以实现自定义具体的负载内容。目前,该基准工具涵盖了一些常见的 NoSQL 数据库,如 HBase、MongoDB、Redis 等数据库的测试,也包括了对 MySQL 等关系型数据库的测试。同时,YCSB 基准工具支持多线程工作,实现对测试对象的并发测试。

(4)JMeter②

Apache JMeter 是一款开源的、基于 Java 语言进行开发的压力测试工具[19-20],它可用于模拟在服务器或者服务器集群、网络和数据库等对象上的负载,并测试在不同负载类型或者负载强度下测试对象的性能表现。对于数据库的性能测试,用户只需要利用不同数据库所提供的读写接口,自定义相应的负载,即可完成对不同种类数据库的性能测试。Apache JMeter 工具内部目前已集成有 JDBC 请求、HTTP 请求等模块,因此,用户也能较为方便地实现负载的定义,并完成对 MySQL、HBase 等数据库的性能测试。

上述4种数据库性能测试工具的分析与对比如表1所示.

表1 数据库性能测试工具对比

Table 1 Comparison of database performance testing tools

项目	SysBench	HammerDB	YCSB	JMeter
系统兼容	Linux、macOS	Linux、Windows	Linux、Windows	Linux、Windows
数据库支持	仅支持 MySQL、PostgreSQL	支持 Oracle、MySQL 等多种关系型数据库,NoSQL 数据库仅支持 Redis 与 Trafodion SQL	支持可使用 JDBC 的关系型数据库,支持 MongoDB、Redis 等多种 NoSQL 数据库	支持可使用 JDBC 的关系型数据库,支持 HBase、MongoDB 等多种 NoSQL 数据库
图形化界面	否	是	否	是
是否开源	是	是	是	是

表1中可见 SysBench 与 HammerDB 主要支持关系型数据库,对 NoSQL 数据库的支持较弱。YCSB 在性能测试时,虽然可以在配置文件中自由设置增、删、改、查各种操作的数量和比例,但是用于测试数据本身默认的 YCSB 是随机生成的。而 JMeter 既支持本文选取的多种数据库性能测试,相比与 YCSB 又可以比较方便的自己设定存储与查询时使用的数据,故本文最后选择使用 JMeter 进行 MySQL 数据库、HBase 数据库、OpenTSDB 数据库和阿里云 TSDB 数据库的性能测试。

1.2 实验数据介绍

实验数据使用的是来自某燃气公司所采集的燃气泄露浓度的监控数据。每条数据包括以下主要字段,具体如表2所示。

表2 燃气监控数据字段表

Table 2 Natural gas monitoring data fields

字段	含义	备注
id	主键	只有关系型数据库含有此字段
collector_id	采集器 id	该条记录是哪个采集器采集到的
manager_id	业主 id	采集器所属业主
area	区域	采集器所在区域
state	状态	0 为正常,1 为气体泄漏
created_at	时间	采集时间,以时间戳表示

对于表2中的数据字段设计,id 为关系型数据库表的主键;collector_id 唯一表示一个特定的燃气数据采集器;manager_id 唯一表示采集器所属业主;area 唯一表示所在区域;state 代表采集器所在燃气管道的状态,“0”表示正常(即未检测燃气

① https://github.com/brianfrankcooper/YCSB

② https://jmeter.apache.org/

泄露),“1”表示气体泄漏;created_at 则表示这条记录所发生的时间。对于一个特定的采集器,每 30 s 触发一次数据采集,则每个采集器每天将产生 2 880 条时序数据,而在实际生产环境中,所部署的采集器数量往往以万计,故每天后台系统所实际存储的数据量是巨大的。

1.3　实验工具介绍

为了实现对上述 4 类数据库的性能测试实验,本文使用 Apache JMeter 工具进行测试,使用版本为 5.4.1。对于 Apache JMeter 工具的基本使用步骤具体如下:

(1)创建测试计划(Test Plan)。通常来说,一个测试计划包括了对测试对象进行性能测试的所有功能实现。Apache JMeter 每次启动都会默认创建一个测试计划;

(2)创建线程组(Thread Group)。Apache JMeter 提供了一个完整的多线程框架,允许实现多个线程并发测试。用户通过线程组,可以自定义测试时所启动的线程数大小;

(3)创建取样器(Sampler)。取样器实现了模拟用于测试的特定操作请求,如“HTTP 请求”取样器、“JDBC 请求”取样器以及“Java 请求”取样器等。

(4)创建监听器(Listener)。监听器实现了对每一次请求的延迟时间(ms)、吞吐量等性能指标的查看。

通过上述操作,可以实现对特定对象进行性能测试工作。

1.4　实验环境介绍

为了保证在实验过程中消除其他无关服务与进程对数据库性能的影响,以及不同数据库之间的相互影响,本实验中的本地数据库均部署并运行在资源配置相同的多个独立的虚拟机上。因此,本实验搭建了 7 台 Linux(CentOS7/x86)虚拟机,并分别在上面部署了 MySQL 数据库、HBase 数据库以及 OpenTSDB 数据库,其中,具体的部署情况为:1 台虚拟机部署 MySQL 数据库,3 台虚拟机部署 HBase 数据库,3 台数据库部署 OpenTSDB 数据库。每一台虚拟机的具体资源配置如表 3 所示。

表 3　虚拟机配置信息

Table 3　Virtual machine configuration information

硬件配置信息	详情
OS System	CentOS7
CPU Cores	3
Base Memory	4 096 MB
Disk Size	50 GB

对于阿里云 TSDB 数据库,本文选择版本型号为“基础版Ⅰ”的数据库实例,该实例支持时间线数为 240 万条,且支持每秒最多 1 万个数据点的写入。

1.5　实验方法

本实验通过使用 Apache JMeter 工具对 2.3 小节所述的 4 类数据库的数据写入和查询性能分别进行测试,分别如下:

(1)对于在 JMeter 上实现对 MySQL 数据库的性能测试方法,主要是通过创建“JDBC 请求”取样器,即通过 JDBC 连接实现对 MySQL 的访问。

(2)对于在 JMeter 上实现对 HBase 数据库的性能测试方法,主要是通过创建“Java 请求”取样器,并通过自行编写 Java 程序实现对实验数据进行过 Put 和 Scan 操作。

(3)对于在 JMeter 上实现对 OpenTSDB 数据库的性能测试方法,主要是通过创建“HTTP 请求”取样器,调用 OpenTSDB 提供的 HTTP API 实现对数据的写入和查询操作。

(4)对于在 JMeter 上实现对阿里云 TSDB 数据库的性能测试方法,主要是通过创建“HTTP 请求”取样器,调用阿里云 TSDB 提供的 HTTP API 实现对数据的写入和查询操作。

2　实验与结果

本节将对 MySQL、HBase、OpenTSDB 和阿里云 TSDB 这 4 类数据库,分别在不同数据量和并发数下的数据写入与查询进行性能测试实验。

2.1　具体实验方案

(1)写入实验

分别测试与记录 4 类数据库在不同写入数据量和不同线程数下的吞吐量,其中,写入数据量分别为 10 000、100 000、390 000、500 000 和 1 000 000 条,启动线程数分别为 1、5、10、20、50 和 100。同时,需要额外说明的是,实验中写入数据是累计存储的,即每次数据写入之后并不对已写入数据进行清除,而是采取累积写入的方式,因此,每次写入后数据库中的总数据量分别为 10 000、110 000、500 000、1 000 000 和 2 000 000。同时,为了实现对写入数据量的控制,实验将数据以 CSV 格式文件进行存储,并通过 JMeter 中的“CSV Data Set Config”配置元件,读取 CSV 文件中的数据,写入到特定的取样器中。

(2)查询实验

分别测试 4 类数据库在不同总数据条数和不同

线程数下执行1 000次数据查询时的吞吐量。其中,数据量分别为10 000、110 000、500 000、1 000 000和2 000 000条,启动线程数分别为1、5、10、20、50和100。每次查询的内容将基于燃气公司的实际业务需求灵活设计,具体查询内容可见3.2.2小节。

2.2 实验结果与分析

2.2.1 数据库写入性能测试

数据库写入性能测试实验总共分为6组,每组实验将实现启动不同的线程数,把存储在本地的燃气监控数据写入到MySQL、HBase、OpenTSDB和阿里云TSDB数据库中。每组实验又分为5组子实验,实现依次写入10 000、100 000、390 000、500 000和1 000 000条燃气监控数据到上述4类数据库中,直到5组子实验全部完成后,每个数据库均存储有2 000 000条燃气监控数据。

下面将展示4类数据库在启动不同线程数下时,吞吐量与数据写入量的关系,具体如图1所示。

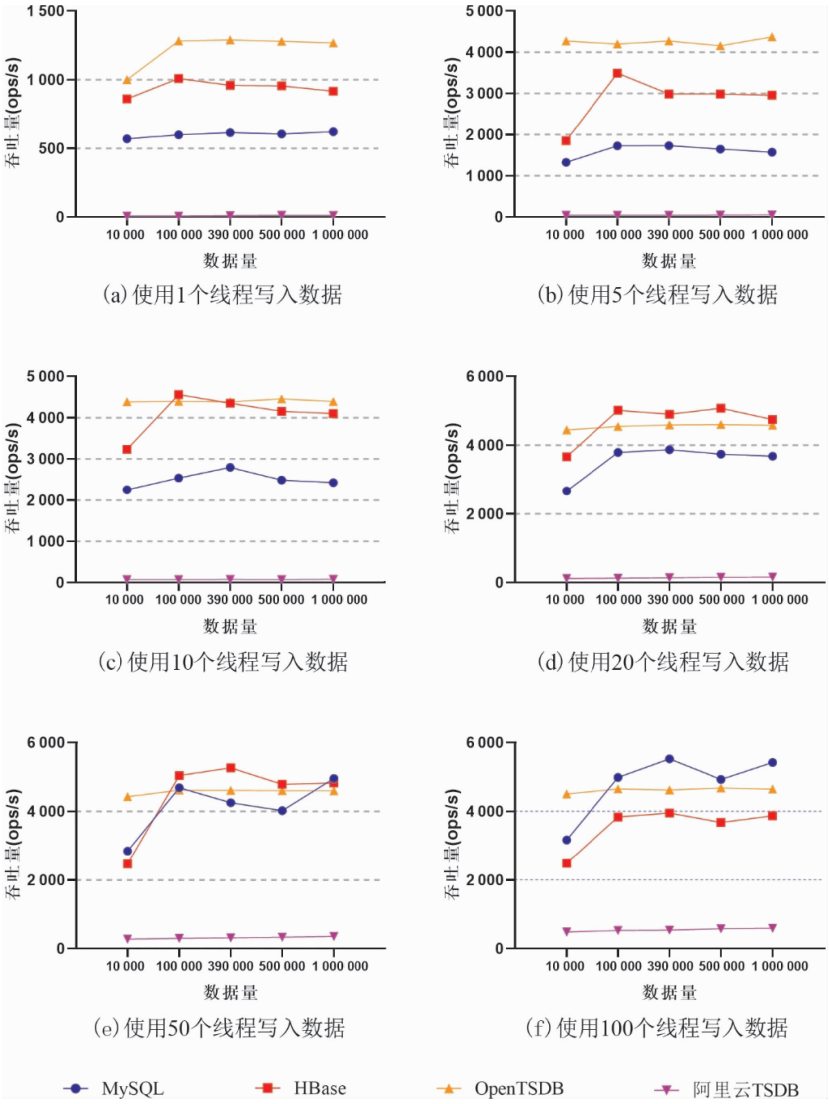


图1 不同线程数下吞吐量与数据写入量的关系

Fig. 1 Relationship between throughput and data write volume under different thread numbers

根据图1所示的实验结果分析可知:

(1)MySQL、HBase和OpenTSDB这3类数据库随着写入数据量的增加,均能稳定地保持较高的吞吐量,而阿里云TSDB数据库的吞吐量明显低于前3类数据库。分析上述实验现象,原因为前3类数据库均部署在本地服务器上,因此,数据传输延迟较低(约4 ms),而阿里云TSDB作为公有云服务,因其数据传输依赖网络状态,故数据传输延迟较高(约79 ms);

(2)MySQL和HBase数据库在写数据量为

100 000 条时的吞吐量,相较于写入数据量为 10 000 条时有一定幅度的提高;而 OpenTSDB 和阿里云 TSDB 数据库随着写入数据量的增加,其吞吐量相较于 MySQL 和 HBase 数据库,没有明显的波动,且更为稳定。

为了更进一步说明在数据写入实验中启动线程数与数据库吞吐量的关系,将每一组实验中写入数据量相同的数据库吞吐量数据集中在一起分析,具体数据可视化结果如图 2 所示。

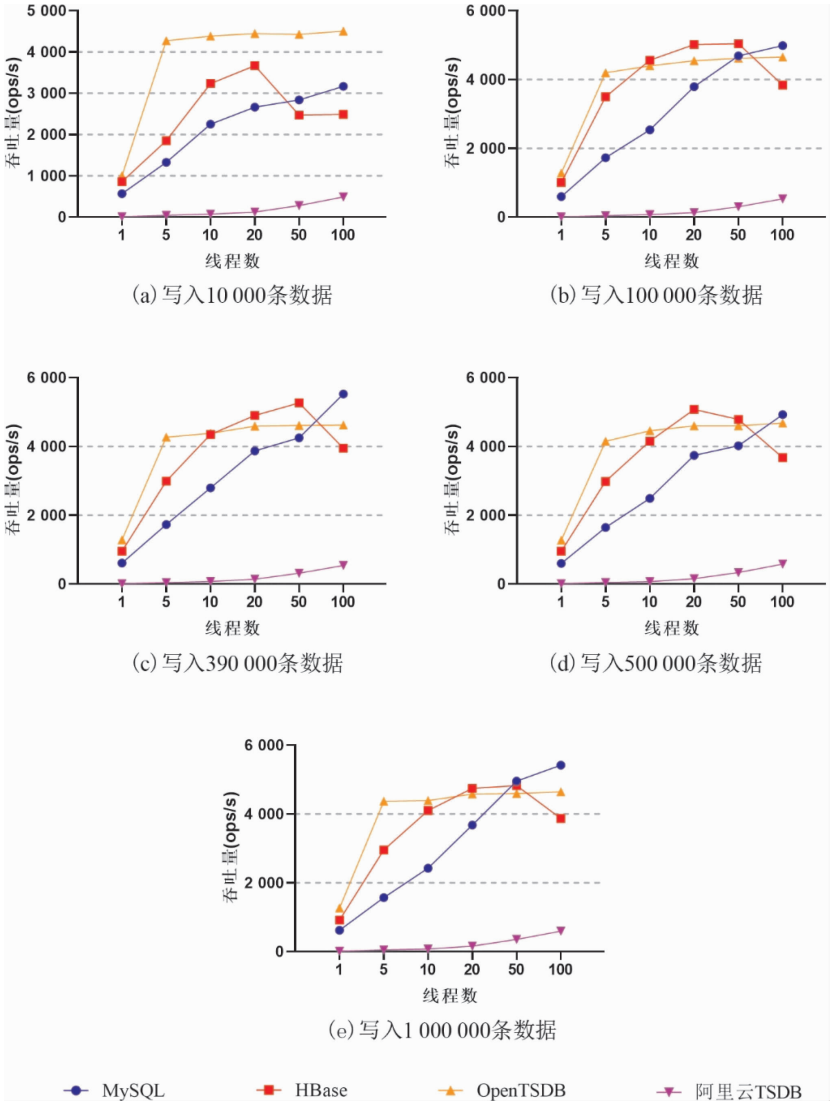


图 2　不同数据写入量下吞吐量与线程数的关系

Fig. 2　Relationship between throughput and number of threads under different data writes

根据图 2 所示实验结果分析可知:

(1) 随着启动线程数的逐渐增加,MySQL、HBase、OpenTSDB 和阿里云 TSDB 数据库的吞吐量均发生不同程度的上升,其中,启动线程数从 1 到 5 时,OpenTSDB 数据库吞吐量的增长幅度最大,MySQL 和 HBase 数据库吞吐量也有一定幅度的增长,阿里云 TSDB 数据库吞吐量则增长不明显;往后随着启动线程数的增加,OpenTSDB 数据库吞吐量增长幅度明显减少,基本上维持在数值大小为 4 600 左右的吞吐量水平,HBase 数据库在启动线程数为 20 或者 50 时,增长幅度明显放缓,甚至出现下降现象;而 MySQL 和阿里云 TSDB 数据库吞吐量则仍保持上升趋势;

(2) 在启动线程数大于 20 时,分析 HBase 数据库吞吐量不稳定的原因为实验所设计的 HBase 表并没有进行合理的 RowKey 设计,导致当线程数增加时,HBase 集群中节点负载不均衡现象加剧,进而影响了 HBase 数据库的写入性能;

(3)随着启动线程数的逐渐增加,阿里云 TSDB 数据库吞吐量增长幅度相较于其他 3 类数据库更为稳定,尽管阿里云 TSDB 数据库的具体资源配置对于用户来说完全透明,但从该现象分析可以推测其云资源是随着负载压力变化而动态调整的,以保证最佳的运行状态。

2.2.2 数据库查询性能测试

数据库查询性能测试实验总共分为 6 组,每组实验将启动不同的线程数,并对 MySQL、HBase、OpenTSDB 和阿里云 TSDB 这 4 类数据库分别执行 1 000 次查询,每次查询的具体内容为在时间区间(2021/05/21 18:16:59 - 2021/05/21 18:26:59)内,查询并返回安装在某个物业下的 35 个不同采集器的监控数据。每组实验又分为 5 组子实验,每个子实验执行查询时所基于的数据量分别为 10 000、110 000、500 000、1 000 000 和 2 000 000 条。

下面将展示 4 类数据库在启动不同线程数执行数据查询时,吞吐量与数据量的关系,具体如图 3 所示。

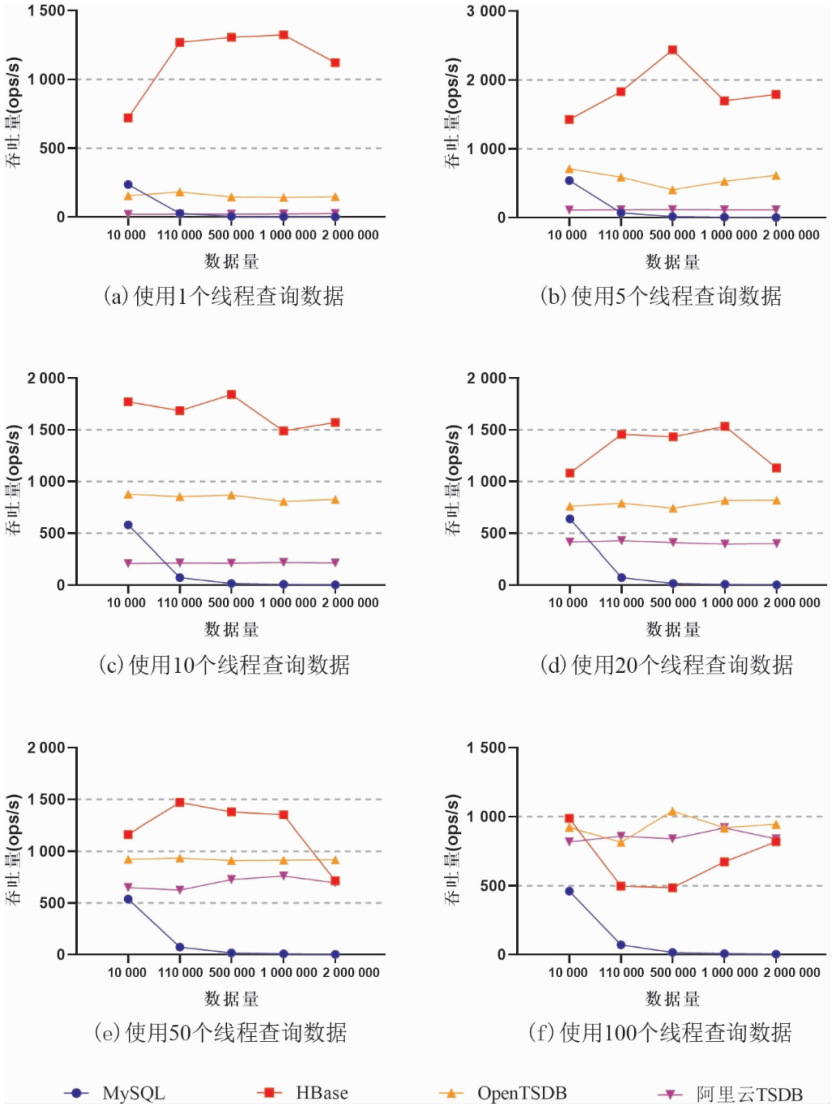


图 3 执行数据查询时吞吐量与数据总量的关系

Fig. 3 Relationship between throughput and total data when querying data

根据图 3 所示实验结果分析可知:

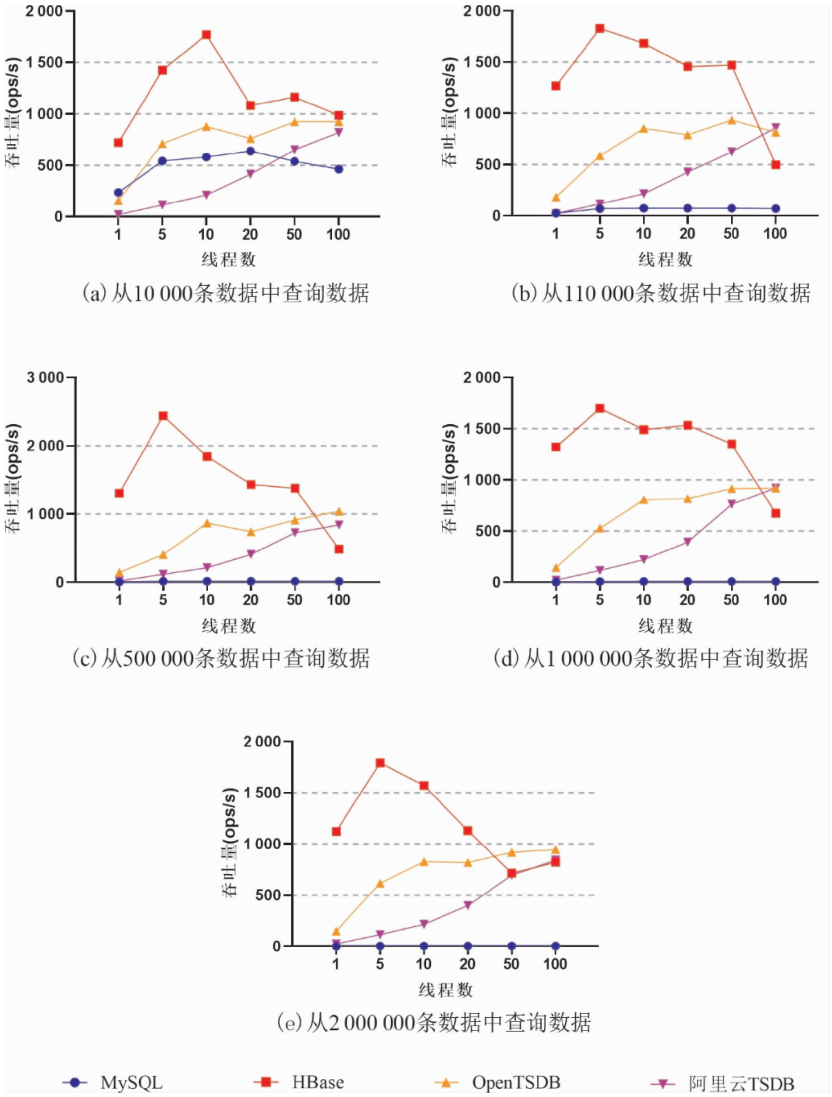
(1)对于 MySQL 数据库,当数据库所存储的数据量较小时执行数据查询操作,吞吐量仍能保持在较高的水平,与 HBase 和 OpenTSDB 数据库相差不大,但随着数据库中所存储数据量的逐渐增加,如大于 110 000 条数据时,吞吐量将快速下降;对于 HBase 数据库,在执行数据查询操作时,吞吐量和数据库所存储的数据量的关系并不固

定,随着数据库中所存储数据量的逐渐增加,尽管其吞吐量相较于其他数据库仍处于一个较高的水平,但存在较大的波动;对于 OpenTSDB 和阿里云 TSDB 数据库,随着数据库中所存储数据量的逐渐增加,其吞吐量变化不明显,比较稳定;

(2)HBase 数据库在执行数据查询操作时,吞吐量在多数场景下相较于其他 3 类数据库均在一个较高的水平上波动,结合 HBase 数据库的架构设计与数据查询原理[21],推测原因是因为 HBase 数据库的读写缓存机制,即在写入数据时,数据并不是直接存储到磁盘中,而是先将数据存储在写缓存(Memstore)中,一旦写缓存达到一定的阈值才会将缓存中的数据刷新到磁盘中,而数据读取则会先在 Memstore 中作查询,若不命中再到读缓存(Blockcache)中进行查询,若再不命中则到磁盘中进行查询,直到将查询数据返回并缓存到 Blockcache 中,以便提高下一次做相同查询时的查询效率。由于 HBase 数据库的读写缓存机制,使其在对热数据的查询效率上存在一定的优势;

(3)在实验过程中,发现 MySQL 数据库在执行查询操作时,对 CPU 资源的消耗情况相比于 HBase 和 OpenTSDB 数据库更为严重,资源瓶颈成为 MySQL 数据库查询性能的关键因素之一。

同样地,为了更进一步地说明在数据查询实验中启动线程数与数据库吞吐量的关系,将每一组实验中查询操作所基于的数据量相同的数据库吞吐量数据集中在一起分析,具体数据可视化结果如图 4 所示。



(a)从10 000条数据中查询数据　(b)从110 000条数据中查询数据

(c)从500 000条数据中查询数据　(d)从1 000 000条数据中查询数据

(e)从2 000 000条数据中查询数据

图 4　执行数据查询时吞吐量与线程数的关系

Fig. 4　Relationship between throughput and number of threads when querying data

根据图4所示实验结果分析可知:

(1)随着启动线程数的逐渐增加,MySQL数据库的吞吐量并没有显著的变化,当数据库存储的数据量为10 000时进行查询,随着线程数的增加,MySQL数据库的吞吐量缓慢上升,并在启动线程数为20时达到最高点,当启动线程数继续增加时,吞吐量反而下降;HBase数据库则在启动线程数逐渐增加时,其吞吐量变化呈现一个先上升后迅速下降的过程,且变化过程也存在一定的波动;而OpenTSDB和阿里云TSDB数据库表现最佳,随着启动线程数的逐渐增加,其吞吐量均逐渐上升,且阿里云TSDB数据库吞吐量的上升速率高于OpenTSDB数据库,当启动线程数为100时,阿里云TSDB数据库吞吐量接近甚至超过HBase和OpenTSDB数据库的吞吐量;

(2)根据HBase数据库的吞吐量变化情况,分析其原因为HBase表的RowKey设计不合理导致HBase数据库在处理并发查询请求时负载不均衡现象加剧,影响数据的查询性能;

(3)阿里云TSDB数据库在启动线程数逐渐增加的情况下,其吞吐量仍能保持较好增长速率,更加体现了云数据库服务的优势所在。

2.2.3 数据库选型建议

根据上述实验分析,给出不同应用场景下存储时序大数据数据库的选型建议:

(1)在MySQL数据库存储时序大数据时,虽然写入吞吐量较高,但是查询吞吐量较低,因此,不建议使用MySQL数据库存储时序大数据;

(2)在HBase数据库存储时序大数据时,写入吞吐量和查询吞吐量均较高,但是由于RowKey没有进行合理设计,使得易于出现负载不均衡,导致吞吐量不稳定。且在数据统计方面,HBase数据库并没有提供数据降采样、聚合等功能。如果要使用HBase数据库实现高效的时序大数据存储,需要确保RowKey根据需求进行合理设计,而数据降采样、聚合等统计功能可通过MapReduce、Spark等技术来实现[22-23]。

(3)在OpenTSDB数据库存储时序大数据时,数据写入和查询吞吐量均保持较高水平且较为稳定。随着线程数的增加,吞吐量也能保持稳定增长。除此之外,OpenTSDB数据库还提供有数据降采样、聚合等统计功能,更有利于数据的统计和分析。在存储时序数据时,尤其是并发压力较高的场景下,推荐使用OpenTSDB数据库。

(4)在阿里云TSDB数据库存储大数据,在写入数据时,由于网络延迟使得吞吐量低于本地数据库,但是在线程数增长时,其查询吞吐量能够非常稳定的增长,当启动100线程时,吞吐量可以媲美OpenTSDB和HBase数据库,且增速完全没有放缓的趋势。相比OpenTSDB,阿里云TSDB数据库同样也提供了数据降采样、聚合等统计功能,而且阿里云TSDB数据库上还额外提供了多值存储、预聚合等功能。除此之外,相比于本地数据库,阿里云TSDB数据库作为公有云数据库服务,在运维成本开销方面具有极大的优势。因此,存储时序数据时若实际场景对延迟有严格要求,不建议使用通过网络提供服务的阿里云TSDB数据库;但是在其他场景下,阿里云TSDB数据库均能满足存储的需求。推荐对服务可用性、稳定性、数据可靠性等要求高的场景使用阿里云TSDB数据库存储时序大数据。

3 结论与展望

本文针对关系型数据库、本地NoSQL数据库以及公有云NoSQL数据库在时序大数据写入和查询时的性能,设计并进行了多组对比实验,研究了在MySQL、HBase、OpenTSDB和阿里云TSDB这4种数据库环境下,随着数据量和线程数的增加时,写入数据和查询数据的吞吐量和性能的变化情况。实验结果表明:存储时序大数据时,MySQL数据库查询速度相对较慢;HBase和OpenTSDB数据库均能实现较快的写入与查询,但前者不够稳定;阿里云TSDB在高并发场景下表现更好。

在实验的过程中,本文也存在一些不足之处,其中一点是数据量还不够大,未来的研究中将继续探索在更大数据量下的性能对比。同时,在本实验中,并未对HBase数据库的RowKey进行优化设计,未来将对比研究不同RowKey设计下HBase的性能差异。

参考文献:

[1] Lee I, Lee K. The Internet of Things (IoT): Applications, investments, and challenges for enterprises[J]. Business Horizons, 2015, 58(4): 431-440.

[2] Warner R M. Spectral analysis of time-series data[M]. Guilford County: Guilford Press, 1998.

[3] Codd E F. Relational database: A practical foundation for productivity[M]//Readings in Artificial Intelligence and Databases. San Francisco: Morgan Kaufmann, 1989: 60-68.

[4] Jatana N, Puri S, Ahuja M, et al. A survey and comparison of relational and non-relational database[J]. International Journal of Engineering Research & Technology, 2012, 1(6): 1-5.

[5] Han J, Haihong E, Le G, et al. Survey on NoSQL database[C]//2011 6th International Conference on Pervasive Computing and Applications. Piscataway:IEEE, 2011: 363-366.

[6] Makris A, Tserpes K, Andronikou V, et al. A classification of NoSQL data stores based on key design characteristics[J]. Amsterdam:Procedia Computer Science, 2016, 97: 94-103.

[7] Deri L, Mainardi S, Fusco F. TSDB: A compressed database for time series[C]//International Workshop on Traffic Monitoring and Analysis. Berlin: Springer, 2012: 143-156.

[8] Ye F, Liu Z, Zhu S, et al. Research of benchmarking and selection for TSDB[C]//International Conference on Algorithms and Architectures for Parallel Processing. Cham:Springer, 2019: 642-655.

[9] Wang M, Liu S, Wei K, et al. Design of performance benchmark for time series databases[C]//2020 IEEE International Conference on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom). Piscataway: IEEE, 2020: 1394-1399.

[10] Wang C, Huang X, Qiao J, et al. Apache IoTDB: Time-series database for internet of things[J]. Proceedings of the VLDB Endowment, 2020, 13(12): 2901-2904.

[11] Li Y, Manoharan S. A performance comparison of SQL and NoSQL databases[C]//2013 IEEE Pacific Conference on Communication, Computers and Signal Processing (PACRIM), Piscataway: IEEE, 2013:15-19.

[12] Abramova V, Bernardino J, Furtado P. Which nosql database? a performance overview[J]. Open Journal of Databases (OJDB), 2014, 1(2): 17-24.

[13] Rautmare S, Bhalerao D M. MySQL and NoSQL database comparison for IoT application[C]//IEEE International Conference on Advances in Computer Applications. Piscataway: IEEE, 2016:235-238.

[14] Klein J, Gorton I, Ernst N, et al. Performance evaluation of NoSQL databases: A case study[C]//Proceedings of the 1st Workshop on Performance Analysis of Big Data Systems. New York: Association for Computing Machinery, 2015: 5-10.

[15] Li A, Yang X, Kandula S, et al. CloudCmp: Comparing public cloud providers[C]//Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement. New York: Association for Computing Machinery, 2010: 1-14.

[16] Krogh J W. Benchmarking with sysbench[M]//MySQL 8 Query Performance Tuning. Berkeley: Apress, 2020: 19-53.

[17] Kopytov A. Sysbench manual[EB/OL]. [2021-05-28]. https://imysql.com/wp-content/uploads/2014/10/sysbench-manual.pdf.

[18] Cooper B F, Silberstein A, Tam E, et al. Benchmarking cloud serving systems with YCSB[C]//Proceedings of the 1st ACM Symposium on Cloud Computing. New York: Association for Computing Machinery,2010: 143-154.

[19] Nevedrov D. Using jmeter to performance test web services[EB/OL]. [2021-05-28]. https://loadstorm.com/files/Using-JMeter-to-Performance-Test-Web-Services.pdf.

[20] Abbas R, Sultan Z, Bhatti S N. Comparative analysis of automated load testing tools: Apache jmeter, microsoft visual studio (tfs), loadrunner, siege[C]//2017 International Conference on Communication Technologies (ComTech). Piscataway: IEEE, 2017: 39-44.

[21] Vora M N. Hadoop-HBase for large-scale data[C]//Proceedings of 2011 International Conference on Computer Science and Network Technology. Piscataway: IEEE, 2011: 601-605.

[22] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1): 107-113.

[23] Zaharia M, Chowdhury M, Franklin M J, et al. Spark: Cluster computing with working sets[C]//Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing. Berkeley: USENIX Association, 2010: 10.

【责任编辑: 陈　钢】