doi:10.3772/j.issn.1002-0470.2023.02.002

大规模时空图数据存储和分析的优化方法①

丁梦苏②*** 杨慕乔 *** 陈世敏③***

(*中国科学院计算技术研究所 北京 100190)

(**中国科学院大学计算机科学与技术学院 北京 100190)

(***College of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh 15213)

摘 要 时空图数据在数据量和数据更新速率两方面具有独特的特征,可以用来优化存储和查询分析。然而,现有的成熟的大数据存储和分析系统提供统一化的支持,没有考虑结合数据特征和查询特征做针对性的优化,因而无法很好地应对大规模数据的挑战,存储和分析能力都有待加强。本文利用时空图数据的数据特征,提出了针对不同类型的顶点和边的差异化存储方案;利用时空图数据的查询特征,提出了差异化的存储布局和基于此的查询执行优化方案。实验结果表明,和现有方案相比,本研究提出的优化方法能减少 1.7~5.4 倍的存储空间,查询性能可以提高 1~4 个数量级。

关键词 时空图存储;时空图查询;大数据存储;大数据分析

0 引 言

大数据对存储和分析系统的挑战主要体现在数据量大(volume)、更新速度快(velocity)和种类繁多(variety)这 3 个方面。不同应用场景产生的大数据在上述 1 个或多个方面具有不同的数据特征,给大数据存储和分析系统带来不一样的挑战和机遇。

图数据是大数据类型之一,被广泛用于表示现实世界的实体和关系。现实世界的图数据通常包含由各种硬件设备(如传感器、交通摄像头、POS 机)和软件系统(如 Web 服务器)生成的时空信息,并且这些信息随着时间演变。例如,人在某一时刻访问过某一位置,可以被基站、二维码(如健康宝)、商店结账机、交通摄像头等记录下来,形成一条包含对象(如人)、时间、位置三要素的数据,本文称这种类型的数据为时空图数据。时空图数据分析可以给政务、电商等带来巨大的价值,促进社会和经济的发展。

时空图数据在数据量和数据更新速率两方面具有更为鲜明、独特的特征,给大数据存储和分析系统带来不一样的需求、挑战和研究机遇。本研究主要针对时空图数据的数据特征和查询特征,研究大规模时空图数据存储和查询分析的优化方法。

传统大规模数据存储和分析系统对数据提供统一化的支持[1-4],没有考虑结合时空图数据的特征做针对性优化,因而无法很好地应对数据量、数据更新速率、查询处理等方面的挑战,存储和分析能力都有待加强。本文主要有 3 方面的贡献。

(1)提出基于确定的顶点类型和边类型的时空图模型,该模型适用于表达时空图数据的数据特征。

(2)提出基于数据特征和查询特征的针对性的大规模时空图数据存储和分析优化方法:利用顶点和边在数据特征上的差异,设计差异化的存储方法,并采用列式压缩存储策略,显著降低存储空间;基于查询特征设计差异化的数据布局方案,并提供高效的查询执行方案,从而减少网络通信和计算开销,实

① 国家自然科学基金(62172390),华为创新项目(HO2017050001B5)和王宽诚教育基金资助项目。

② 女,1993 年生,博士生;研究方向:大数据处理,数据管理系统;E-mail:dingmengsu@ict.ac.cn。

③ 通信作者,E-mail:chensm@ict.ac.cn。

(收稿日期:2021-12-08)

现提升查询处理性能的目的。

(3)实现了原型系统,并实验验证本研究方案的高效性。和现有的解决方案(ST-Hadoop[5]、JanusGraph[1]、Greenplum[2]和 Spark[3])相比,本研究方案在查询性能上能取得 1 ~4 个数量级的提升,同时能减少 1.7 ~5.4 倍的存储空间。

1 大规模时空图存储和分析背景介绍

1.1 典型的大规模时空图应用场景

应用场景 1:新冠疫情病例跟踪和防控。时空图数据可以帮助分析和防控新冠疫情[6-7]。这些时空图数据可以是被基站、健康宝二维码和刷卡机等记录下来的,表示人在某一时刻访问过某一位置(如蜂窝基站、商店覆盖的区域)。利用时空图数据可以发现时空伴随者、确定中/高风险地区,从而帮助做针对性的筛查。对于高风险人员,在必要的情况下,可以建议采取预防措施(如病毒检测、隔离)。

应用场景 2:用户行为分析和挖掘。理解用户行为有助于分析欺诈行为并提供个性化推荐。例如,信用卡公司可以通过分析客户刷卡的时间和地理位置来进行盗刷检测,互联网公司可以通过分析用户的行为数据进行感兴趣的地方(points of interest,POI)推荐[8-9]。在相邻的时间访问相似地点的人通常具有相似的兴趣,因此,可以根据时空图数据来发现具有相似活动的人群。

应用场景 3:套牌车检测。套牌车指的是与合法汽车具有相同品牌、型号、颜色、车牌号等信息的汽车。套牌车的车牌是不法分子伪造或非法套取的。基于“望、摸、问、查”的传统检测方式[10]很难识别套牌车,而时空图数据分析可以降低套牌车的检测难度,提高检测的准确性[11]。当汽车经过交通摄像头时,摄像头会拍照并自动识别车牌号码。然后,可以参考道路行驶的正常速度,来判断一辆车(车牌)是否存在套牌车的可能性。如果摄像头记录的同一车牌在短时间范围内出现在位置相距很远的地方,而汽车不可能在如此短的时间内行驶这么长的距离,则可判断该汽车存在套牌车的可能。

1.2 大规模时空图应用的通用特征

(1)数据维度:包含对象维度、空间维度和时间维度。时空图应用程序反映对象在时间和空间上的活动。除了和应用程序关联的其他信息,时空图数据包含 3 个通用的信息维度,即对象维度(如人、汽车)、空间维度(摄像头等记录下的位置)和时间维度。

(2)空间维度:有限的位置。手机通信的基站、携带位置信息的二维码(如健康宝二维码)、交通摄像头和商店登记机等可以产生空间信息的设备数量是有限的。上述应用场景代表了一系列基于有限固定位置信息的时空图数据应用场景。有限的位置为优化时空数据存储和分析的设计提供了机会。

(3)对象维度:对象比位置多得多。对象(如人、车牌)的数量可能比位置的数量大几个数量级(详见 1.3 节)。这可以差异化地对待对象数据和位置数据,由此可以大幅降低数据存储开销。传统大规模时空数据分析系统(如 ST-Hadoop[5])将对象数据和空间数据一视同仁,没有利用这两者的差异进行优化处理,因而,在存储和查询分析两方面的性能均存在改进空间。

(4)时间维度:高速流入的新边。假设全球 78 亿人在高峰时期每人每天访问约 100 个地点,那么,每天都有上万亿的新边发生。边数据具有庞大的数据量和高速的数据更新速度,对存储和查询分析性能起着至关重要的作用。

(5)查询:涉及 1 个或多个维度的数据分析。时空图数据分析通常需要处理对象、时间、空间这 3 个数据维度中的 1 个或多个。系统需要分布式的解决方案来应对数据量和数据更新速度带来的挑战。但是,如果没有精心的设计,查询很容易带来高昂的跨机器通信开销。

1.3 优化目标和优化思想

本研究提出的面向大规模时空图存储和分析系统的优化方法,希望在实现以下目标的情况下,提供更加高效的存储和查询性能。

数据量:支持 100 亿对象顶点、1000 万个位置顶点和 100 万亿条时空边。首先,100 亿对象的需求是基于世界上大约有 78 亿人的事实。其次,根据 booking.com,全球约有 190 万家酒店和其他住宿场所。假设商店和餐馆的数量是酒店的 5 倍,那么位

置总数可以达到 1000 万个。最后,假设一个对象平均每天看到大约 10 个新事件。如果将事件保存 3 a,则大约可以有 100 万亿个事件,每个事件都由唯一时间戳标识,即时间维度的数据基数可达 100 万亿。

本文考虑时空图数据在对象、时间和位置维度上的最大的数据量。现实的时空图应用场景可能产生规模更小的时空图数据。

查询:最小化跨机器通信开销。时空图数据的总数据量十分庞大,查询分析产生的跨机器通信开销是性能的主要瓶颈。理想情况下,查询处理中唯一的通信是发送查询语句和返回查询结果的通信。查询通常涉及 1 个或多个维度,不同维度构成的查询带来的通信开销可能不一样,这主要由数据存储布局和查询执行计划决定。良好的数据存储布局和查询执行计划可以显著降低参与跨机器通信的数据量。

本研究主要从以下 2 方面考虑面向大规模时空图的大数据存储和分析的优化方法。

(1)利用对象、时间、空间 3 个维度在数据量和数据更新速率等方面的巨大差异,设计差异化的存储方法,来减少存储空间、提高查询性能。

(2)根据时空图查询涉及的多个维度,设计有效的数据布局,实现在不同维度的情况下都能有理想的查询性能,解决传统的多维划分方案的瓶颈问题。

2 相关工作

分布式图数据库系统。考虑使用分布式图数据库系统(如 JanusGraph[1])来支持时空图数据存储和查询分析。图数据库系统将顶点和边存储在键值存储系统(如 Apache Cassandra、Apache HBase),并利用搜索平台来加速数据访问。图数据系统可以有效地处理简单的图遍历查询。然而,大规模时空图数据的查询效率低下,主要因为:(1)它们不支持在时间维度和空间维度上进行数据过滤。但是,基于对象、时间、空间这 3 个维度的数据过滤对时空图数据查询分析性能起着至关重要的作用。(2)它们需要扫描所有图数据,然后调用大数据分析系统(如 Spark)进行复杂查询,这个过程会产生巨大的 I/O 开销。

大规模并行分析(massively parallel processing, MPP)数据库。考虑使用 MPP 关系型数据库来支持时空图数据存储和查询分析。时空图数据可以表达成关系表,存储在 MPP 数据库系统(如 Greenplum[2]),并使用结构化查询语言(structured query language, SQL)进行查询。Greenplum 支持在多个数据维度上进行分区,从而加速基于不同数据维度的数据访问。数据首先按第 1 个分区维度进行分区,然后将第 2 个分区维度应用于每个一级分区,得到一组二级分区,以此类推。所有维度的分区形成一棵树。MPP 数据库系统执行时空图数据分析的性能可能十分低下,因为:(1)不支持空间维度分区,从而导致基于空间维度的查询可能需要处理大量不必要的数据;(2)当查询语句在第 1 个分区维度上没有过滤谓词的情况下,查询仍需要遵循树从根级别开始访问数据,这可能会带来大量的中央处理器(central processing unit, CPU)、磁盘 I/O 和通信开销。

通用大数据分析系统。通用大数据分析系统(如 Hadoop[4]、Spark[3])支持对存储在底层存储系统中的数据进行大规模计算,这些底层存储系统包括分布式文件系统(如 Apache HDFS)、分布式键值存储系统(如 Apache Cassandra)。由于分布式键值存储系统提供根据键(key)进行快速查找的能力,因此,可以考虑将时空图数据存储在分布式键值存储系统,用以加速基于对象维度的查询。但是,通用大数据分析系统通常需要在执行计算之前加载所有数据,这将带来高昂的磁盘 I/O 开销。因此,通用大数据分析系统不适合支持涉及少量数据计算的时空图数据查询。此外,由于系统不支持时间维度和空间维度的数据过滤,因而复杂的时空图查询可能会因读取所有数据而产生大量的磁盘、计算和通信开销。

时空大数据存储和分析系统。时空大数据存储和分析系统在通用大数据分析系统的基础上,增加了时间和空间索引,能提供更高效的查询性能。ST-

Hadoop[5]是该类系统的典型代表,它是 Hadoop[5]和 SpatialHadoop[12]的扩展。然而,ST-Hadoop 具有局限性。首先,ST-Hadoop 牺牲存储空间来换取查询性能。ST-Hadoop 将其两级索引复制到具有不同时间粒度(如日、月、年)的多个层中。在每一层内,整个数据集都被复制和划分。其次,不支持索引对象标识符(identifier, ID)。因此,简单的查询可能会因读取大量数据而产生庞大的 I/O 开销。最后,ST-Hadoop 基于 MapReduce 框架,中间结果写入磁盘,可能会产生大量磁盘 I/O 开销。

对象、时间和空间这 3 个数据维度对时空图查询性能起着至关重要的作用。采用数据划分的方式来创建数据布局,是数据库常用的一种优化手段。有效的划分和数据布局可以帮助系统高效地支持基于相关维度的查询,减少磁盘、CPU 计算和网络通信开销。表 1 总结了上述描述的系统针对时空图数据在对象维度、时间维度、空间维度上的查询支持情况。由于这些系统无法对所有维度提供有效的支持,因而,时空图数据分析性能可能表现得十分低下。

表 1　现有的解决方案与本研究方案

系统	查询关联的数据维度		
	对象	时间	空间
JanusGraph	√	×	×
Greenplum	√	√(不友好)	×
Cassandra	√	×	×
ST-Hadoop	×	√	√
本研究方案	√	√	√

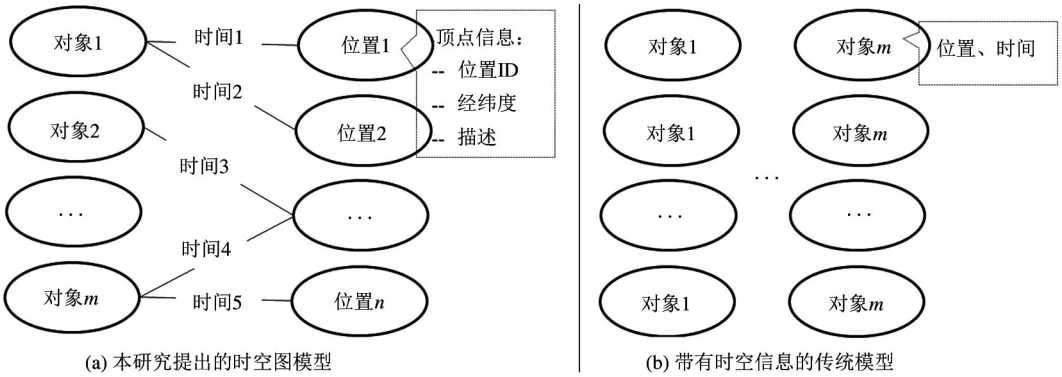
3　时空图模型

本研究提出如下时空图模型。

时空图 $G = (V_L, V_O, E)$。V_L 是有限的位置顶点集合,每个位置顶点都包含位置相关的属性。V_O 是有限的对象顶点集合,每个对象顶点也可能具有其他属性。V_L 和 V_O 的每个顶点都被分配了一个全局唯一的 ID。E 是无向边的集合,集合中的每条边都连接一个对象顶点和位置顶点,并包含时间属性。值得一提的是,对象顶点、位置顶点、边可能包含除对象 ID、位置 ID、时间之外的和应用程序相关的属性。

图 1(a)展示了本研究提出的时空图模型。图 1(a)中的椭圆代表顶点,图中包含对象顶点和位置顶点。位置顶点的示例包括用户在新冠疫情病例跟踪和防控、用户行为分析和挖掘等应用中访问的位置和套牌车检测应用中的交通摄像头。对象顶点的示例包括用户行为分析和挖掘应用中的人、套牌车检测应用中的车牌以及物流追踪应用中的包裹。每个位置顶点都包含一个位置属性,这样给定 2 个位置顶点 u 和 v, 它们的距离 dist(u, v)是确定的。例如,如果应用程序关注地理位置,则位置属性由位置顶点经纬度组成。

上述描述的时空图本质上是无向二分图。本文不考虑对象顶点之间的边和位置顶点之间的边,因为现有的技术可以解决这些问题。本研究聚焦时空图,关注时空图查询优化的关键技术。传统的二分



(a) 本研究提出的时空图模型　　(b) 带有时空信息的传统模型

图 1　位置建模成顶点及位置建模成属性

图数据划分[13-14]侧重于利用谱聚类的思想将相邻的顶点聚在一起。然而,时空图数据具有十分庞大的顶点和边数目,如果不考虑时空图数据的数据特征和查询特征做针对性的设计,容易产生巨大的存储和计算开销。

时空图模型和传统图模型的差异主要体现在对位置维度信息的处理。受有限位置的启发,时空图模型明确地将位置维度信息建模成顶点,而传统图模型没有明确这一点。通过将位置维度信息建模成顶点,可以显著降低时空图数据存储的开销。

为便于理解时空图模型给数据存储带来的优势,本节考虑与不将位置维度信息建模成顶点的传统图模型(图 1(b))进行对比。在该模型中,对象维度信息被建模成顶点,时间维度信息和位置维度信息被建模成和对象顶点关联的属性(分别为图 1(b)每个对象顶点关联的时间和位置信息)。由于同一对象在不同的时间可能访问相同或不同的位置,因此,图 1(b)呈现了多个相同的对象顶点。实际上,在图 1(b)代表的图模型中,对象顶点的基数数目和时间维度的基数数目是一致的。

不失一般性,本文只考虑直接与对象维度、时间维度和空间维度关联的属性。假设对象维度、时间维度和空间维度信息的基数分别为 M、N 和 K,即整个数据集反映 M 个对象在 N 个时间点上访问了 K 个位置的数据。根据 1.2 节描述的数据量,M、N 和 K 之间存在以下数量级关系。

$$\frac{N}{M} = 10^4, \quad \frac{M}{K} = 10^3$$

假设对象维度关联的所有信息(如 ID、姓名、电话号码等)占 $b1$ 字节,时间维度关联的所有信息占 $b2$ 字节,空间维度关联的属性(如 ID、经度、维度、位置描述等)占 $b3$ 字节。若用 a 表示存储一个对象 ID 和一个位置 ID 占用的字节数,那么 a 可能远小于 $b1 + b3$。

表 2 呈现了不同的模型存储数据所需的字节数。时空图模型所需的字节数,由对象顶点的字节数 $M \times b1$、位置顶点的字节数 $K \times b3$ 和边占用的字节数 $N \times (b2 + a)$ 组成(存储的边信息包含对象 ID 和位置ID)。而在传统图模型中,每个顶点同时包含对象维度、时间维度和位置维度数据,因此,每个顶点占用的字节数为 $(b1 + b2 + b3)$,顶点的数目和时间维度的基数一样,即 N 个顶点,故总体占用 $N \times (b1 + b2 + b3)$ 字节。由于 $N >> M >> K$ 并且 a 远小于 $b1 + b3$,因此本研究提出的时空图模型可以大幅减少数据的冗余度,节省大规模数据的存储开销。

表 2 本研究提出的时空图模型和传统图模型关于存储数据所需字节数的对比

	时空图模型(本研究)	传统图模型
顶点数	M, K	N
边数	N	-
字节数	$M \times b1 + N \times (b2 + a) + K \times b3$	$N \times (b1 + b2 + b3)$

注:$N >> M >> K$;a 远小于 $b1 + b3$

4 存储和查询分析优化方法

4.1 基于差异化特性的时空图数据存储

传统图划分方法的研究工作主要可分为 2 类:基于顶点的划分和基于边的划分。基于顶点的划分方法[15]将顶点视为基本的划分单元,顶点及其入射边将被分配给同一分区,以便最小化跨分区边数目。然而,现实生活中的图通常具有大量的大度顶点,无论如何分配,都会产生大量的跨分区边。为了解决这个问题,基于边的划分方法[16]将边分配给分区。如果某个(大度)顶点的入射边在 $k > 1$ 个分区中,则该方案选择一个分区来存储该顶点的主副本,并在每个其他分区创建一个该顶点的副本。这样一来,跨分区边的数量将减少至 $k - 1$(即副本顶点到主顶点之间的边)。

受基于边的划分方法的启发,本研究为时空图提出差异化的划分和存储方法。不同于现有方法,本研究设计方案考虑了位置顶点、对象顶点和边的不同特征,以及时空图数据的查询特征。利用不同顶点和边在数据量、数据更新速率上的差异,设计差异性的存储方法(如表 3 所示),可以有效地减少存储数据的磁盘开销。与此同时,结合查询特征,对数据进行有效地划分,可以达到提高查询性能的目的。

表 3　本研究提出的差异化时空图存储方案

	位置顶点	对象顶点	边
划分方式	复制	哈希	差异化划分
存储介质	内存	磁盘	磁盘

4.1.1　位置顶点存储

时空图数据存储方案采用复制和基于内存的存储方式来存储位置顶点数据,因为存储所有的位置顶点占用的空间非常小。假设一个位置顶点相关的数据信息(含经纬度信息、位置描述信息等)占用 10 ~100 字节,那么存储 1000 万个位置顶点相关的数据需要大约 100 MB ~1 GB 空间。现在的中端服务器通常配备 100 GB ~1 TB 的主内存。因此,所有位置顶点都可以轻松放入一台机器的主内存中。

时空图数据查询通常需要根据位置 ID 访问位置信息,采用这样的存储方式,查询只需要访问本地存储的信息,避免了跨机器的通信开销和磁盘访问开销,因而可以大幅提高查询分析的性能。在系统初始化阶段,工作节点将本地存储的所有位置顶点数据加载到主内存中。当位置更新时,系统更新所有工作节点。然而,更新成本相对来说是非常小的,因为位置顶点(如基站、商店、酒店、交通摄像头位置等)的变化相对比较缓慢。

4.1.2　对象顶点存储

时空图数据存储方案采用哈希划分和基于磁盘存储的方式来存储对象顶点数据。这样的设计主要出于两方面原因。一方面,对象顶点的数据量十分庞大,中端服务器的内存不适合或无法存储如此大规模的数据。假设一个对象顶点相关的数据信息占用 10 ~100 字节,那么存储 100 亿个对象顶点相关的数据需要大约 100 GB ~1 TB,也就是说,需要占用中端服务器的所有内存开销,这将导致没有额外或足够的内存供其他计算使用,系统可能宕机或是执行查询分析的性能表现得十分低下。另一方面,时空图数据查询需要根据对象 ID 访问对象信息,通过哈希划分,同一对象的数据尽可能在同一台工作节点上,由此可以避免跨机器通信开销,提高查询性能。

4.1.3　边存储

时空图数据存储方案采用差异化划分和基于磁盘的存储方式来存储时空边数据。这样的设计主要出于两方面原因。一方面,边数据的数量级十分庞大(可达百万亿级);另一方面,时空图数据查询涉及由对象、时间和空间等一个或多个维度构成的查询模式,差异化划分方法有助于在不同的组合维度的情况下实现理想的查询性能。这是传统的多维度划分方法无法实现的。

为了进一步减少时空边数据的存储开销,对时空边数据进行列式存储压缩。列式存储压缩是数据库常用的一种存储优化方法,通过将具有同一类型的数据进行列式存储并压缩,可以避免访问与查询无关的数据域,并且提高压缩率、降低磁盘的存储开销。

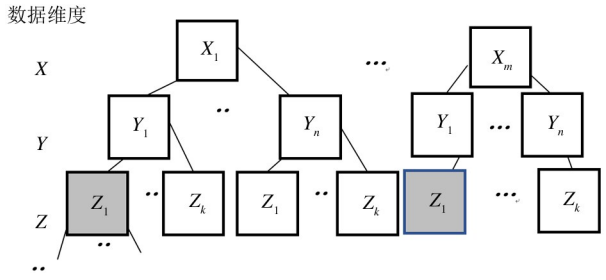
4.2　时空边的差异化存储布局

数据划分是一种常用的构建数据存储布局的方法,通过将数据根据某些数据维度进行重新组织,并在分布式环境下,将数据分配到具体的工作机器,可以在查询过程中有效地避免访问和传递与查询结果无关的数据,减少磁盘开销和网络通信开销。

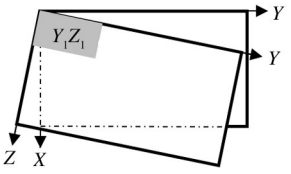
现有的多维数据存储布局的方法大多基于层次结构,如图 2(a)所示。该方案先按照维度 X 对数据进行划分,然后按照维度 Y 对每个一级分区进行划分,从而得到一组二级分区,以此类推到更多的维度。然而,为了检索分区 Y_1Z_1 中的数据,系统必须访问 m 个一级分区,也就是需要读取 m 个 Y_1Z_1 中的数据(图 2(a)的阴影区域)。这通常会导致系统将在多个工作节点上进行 m 次随机访问。

针对现有方法存在的问题,本研究提出基于不同备份采用不同的划分方法来构建差异化的存储布局的优化方法,来实现各个维度的性能平衡。分布式系统通常采用多备份的策略来支持容错恢复,因而,本方案不会给原有系统带来额外的存储开销。图 2(b)展示了差异化存储布局的主要思想。一个备份先按照维度 X(对象),然后按照维度 Y(时间)划分。另一个备份先按照维度 Z(空间),然后按照维度 Y(时间)划分。通过使用第 2 个备份,系统可以在单台工作节点上只执行一次顺序访问磁盘 I/O

的操作来读取一个 Y_1Z_1 分区，和图 2(a)呈现的现有方法相比，本方案可以大幅减少数据访问成本。



(a) 基于传统方案的数据存储布局



(b) 差异化的数据存储布局

(X:对象维度;Y:时间维度;Z:空间维度)

图 2　传统方案的数据存储布局及本研究设计方案

4.3　基于差异化存储布局的查询执行优化

单个时空图查询操作可以组合多个差异化的存储布局来实现更好的查询性能。为了便于理解，不失一般性，以查询语句“给定一个对象和一个时间范围，列出和给定对象在时间和空间上具有交集的其他对象”为例，分析查询执行过程。该查询涉及到复杂的连接操作(笛卡尔积 cross join)，通信开销庞大。系统可以通过采用类似 broadcast join 的思想来减少通信开销。尽管如此，基于单一存储布局的传统方法仍然可能导致系统在查询过程访问大量不相关的数据，加重网络通信开销。

然而，基于差异化存储布局，系统可以大幅减少磁盘 I/O 开销。具体表现为，系统利用对象-时间备份，查找对象的边数据并找到所有位置，根据这些位置计算存在交集的位置分区，并将这些位置发送给存储相关空间分区的工作节点。然后，每个工作节点在给定的分区内查找在时间和空间上存在交集的边数据。在查询的不同阶段利用不同的备份，对象-时间备份支持高效检索对象的行为数据，而空间-时间备份有效地修剪了大量不相关的数据(因为一个人访问的位置通常位于所有空间区域的一个小子集中)。

5　实　验

本实验主要从查询性能和存储性能着手，验证本研究方案的有效性。

5.1　实验配置

实验运行在 11 台机器组成的集群上(1 台协调节点和 10 台工作节点)。每台机器都是戴尔 PowerEdge 刀片服务器，每台服务器配备 2 颗 CPU，型号为 Intel(R) Xeon(R) E5-2650 v3 2.30 GHz(10 核/20 线程，25.6 MB 缓存)，同时配有 128 GB 内存、1 TB机械硬盘和 180 GB 固态硬盘。服务器运行 64 位 Ubuntu 16.04 LTS 系统，系统内核是 4.4.0-112-generic Linux 版。服务器之间通过 10 Gbps 以太网连接。本实验使用 Oracle Java 1.8、Cassandra 2.1.9、JanusGraph 0.2.1、Greenplum 5.9.0 和 Spark 2.2.0。

受场景、隐私等方面的限制，本实验无法获取真实场景下满足大规模时空图特征的数据集。因此，本实验结合真实的位置数据，合成满足应用场景需求的大规模数据。本研究实现支持 100 亿个对象顶点、1000 万个位置顶点目标是针对具有 1000 台机器的集群设计的。鉴于真实实验环境的机器规模，本实验将对象顶点和位置顶点的数量都缩小 100 倍。因此，本实验希望生成 1 亿个对象顶点和 10 万个位置顶点。数据生成过程包含以下几个步骤。首先，从某酒店预订网站抓取国内的酒店位置，这些地点分布在大约 1100 个地区(市/县/区)。然后，从这些酒店随机选择 10 万个位置。接着，产生 1 亿客户。由于位置的数量和一个地区的人口通常是相关的，因此可以将客户分配到指定区域，以便客户数量与一个地区的位置数量成正比。接着，生成一个涵盖 2 ~ 3 a 时间段的数据集。假设 40% 的人经常购物，60% 的人不常购物。频繁购物者和不常购物者每周分别以 0.8 和 0.2 的概率访问他/她所在地区的购物地点(这里基于每周来模拟访问概率是为了控制生成的数据总量，以在集群存储容量范围内。在动态数据注入的实验中，系统会尽快发送产生的数据，以便使系统达到饱和状态)。最终，实验产生的时空图数据包含 570 亿条边，时间范围覆盖 800 d。

5.2 查询语句

基于典型的大规模时空图应用场景,本实验考虑以下 4 种查询。

(1)查询 1(Q1):查找对象的行为。给定 1 个对象和 1 个时间范围,列举该对象在指定时间范围内访问过的位置(含对象、时间、位置信息)。该查询可用于查找新冠确诊病例在 14 d 内访问过的地方,从而帮助排查、控制疫情传播;也可以用于检测客户在指定时间段内的活动(访问的景点、餐馆等),帮助改善基于时空的个性化推荐服务。

(2)查询 2(Q2):计算对象的相似度。给定 2 个对象和 1 个时间范围,计算该时间范围内 2 个对象行为的相似性。考虑对象 o1 的数据(*object*1、*time*1、*location*1)和对象 o2 的数据(*object*2、*time*2、*locaton*2)。如果 $|time2 - time1| \leqslant$ TH _ time,并且 dist(*location*2,*location*1) ≤ TH _ dist,则认为这 2 个对象的行为相似。其中,TH _ time 和 TH _ dist 分别是时间和位置距离的预定义阈值。2 个对象的相似度用相似行为发生的频率来衡量。该查询也可以用于检测新冠确诊病例和待筛查人员在相邻时间和空间上的活动交集,来估计待筛查人员的感染风险。

(3)查询 3(Q3):发现相似的对象。给定 1 个对象和 1 个时间范围,列出和给定对象具有相似行为的其他对象,并按照相似度由高到低的顺序排列。其中,相似度的衡量可通过查询 2 判定。该查询可用于发现与新冠确诊病例密接的人员,也可在用户行为分析和挖掘应用程序中发现具有相似兴趣的人群。

(4)查询 4(Q4):检测不合理的对象。给定时间范围,根据对象在指定时间段的行为,检测不合理的对象。根据对象在时空维度产生行为的速率来判断合理性。如果一个对象的 2 个活动行为(*object*、*time*1、*location*1)和(*object*、*time*2、*location*2)不满足(dist(*location*2,*location*1))/(|*time*2 − *time*1|) ≤ TH _ velocity,那么同时发生这 2 条行为被视为不合理。该查询可以用来检测套牌车,还可用于信用卡公司检测盗刷信用卡的情况。

这 4 个查询语句几乎涵盖数据库常用的分析算子(选择、过滤、连接、聚合、排序等)。其中,查询 1 和查询 2 是简单的查询,和其他相比,它们带来的数据访问和计算开销都比较小。

实验统计的查询时间,是在(同一)查询执行 5 次的运行时间的平均值。每个查询语句的关联的对象是随机设置的。在同一查询下,不同的对象不会影响实验结果。

5.3 查询性能

5.3.1 和现有解决方案的性能比较

本实验在全部数据存入系统的情况下,统一比较所有系统的查询时间(查询的时间范围设置为 32 d。查询 2 和查询 3 涉及的时间和位置距离的阈值分别设置为 7 h 和 100 m。查询 4 涉及的速度阈值设置为120 km/h)。图3展示了各个系统的运行

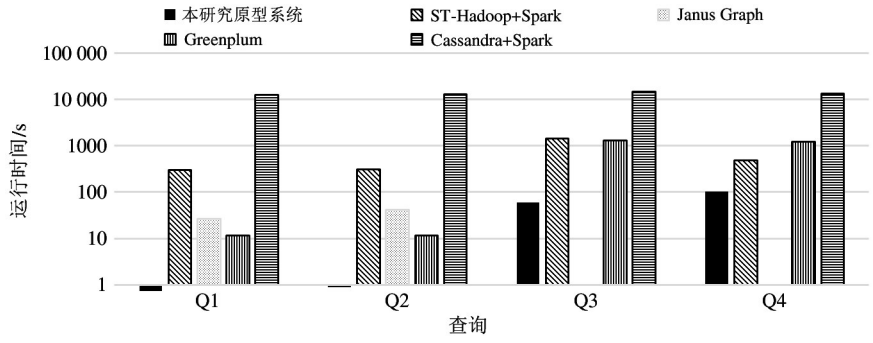


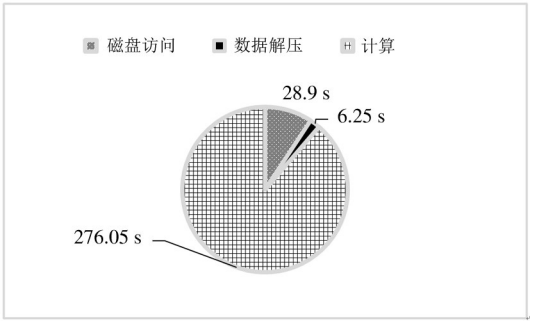
图 3 各系统查询时间

时间,横坐标表示不同的查询,纵坐标表示的是执行时间的对数值。本实验不考虑在 JanusGraph 系统上运行查询 3 和查询 4。JanusGraph 主要关注简单的遍历查询,并使用 Spark 进行复杂的查询,因此 JanusGraph 关于查询 3 和查询 4 的查询性能可以参考 Cassandra + Spark。

与现有的解决方案相比,本研究方案原型系统实现了 1 ~4 个数量级的性能提升。本研究设计方案可以有效减少查询访问的数据量和数据通信开销。Cassandra + Spark 的主要瓶颈在于扫描所有数据的磁盘 I/O。ST-Hadoop + Spark 的性能优于 Cassandra + Spark,因为它利用时空索引,减少了读取的数据量。对于简单查询语句,比如查询 1 和查询 2,Greenplum 的性能明显优于 ST-Hadoop + Spark。这是因为 Greenplum 首先按对象划分数据,然后按时间进一步划分得到子分区。在这种情况下,对象分区很好地满足了查询 1 和查询 2 的需求。相比之下,ST-Hadoop 不支持对象索引,因而前 2 个查询的执行性能不及 Greenplum。

5.3.2　查询时间分解:磁盘访问、解压和计算对总体性能的影响

查询时间主要由 3 部分构成:从磁盘读取查询所需数据涉及的磁盘访问时间、解压读取的数据的时间以及执行查询相关的计算所需的时间。本实验以查询 4 为例,统计查询 4 的各部分用时。从图 4 可以看出,解压的时间占比最小(仅占 2%),相比总时间而言可以忽略不计。因此,数据压缩是一种非常可行的优化方法,既不会给查询带来明显的额外开销,又可以显著降低存储数据所需的空间开销。计算的时间占比最大(高达 89%),其次是磁盘访问的时间。这说明从避免访问查询无关的数据和提供更高效的执行方式这 2 个角度进行查询优化是非常合理的。本研究提出的优化方法可以实现上述目的。



(图中显示的数字为实际用时/s)

图 4　查询时间分解

5.4　存储性能

5.4.1　和现有解决方案所需的存储空间比较

表 4 显示了所有系统存储所有数据花费的存储空间。注意本研究原型系统和 Cassandra 保留 3 份备份。由于磁盘空间的限制,因此 ST-Hadoop、JanusGraph、Greenplum 存储的备份数目都少于 3 份。

表 4　各系统消耗的存储空间

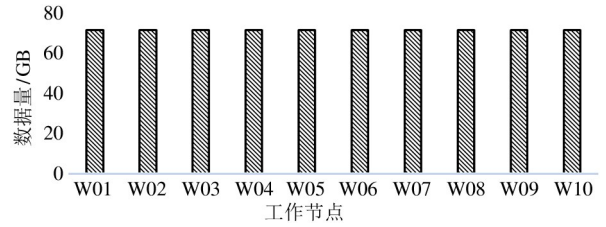
系统	数据量/TB	备份数目
本研究原型系统	1.9	3
ST-Hadoop	3.1	1
JanusGraph	3.4	1
Greenplum	2.8	2
Cassandra	3.2	3

从表 4 可知,与 Cassandra 相比,本研究原型系统节省了 1.7 倍的存储空间。至于其他的系统,尽管本研究原型系统存储了更多的备份,本研究原型系统仍消耗更少的存储空间。如果考虑 ST-Hadoop、JanusGraph 和 Greenplum 存储 3 个备份,那么本研究原型系统和这些系统相比,可以分别减少 4.9 倍、5.4 倍和 2.2 倍的存储空间。JanusGraph 将图数据存储在 Cassandra。Cassandra 和 JanusGraph 都采用 LZ4 压缩。相比之下,本研究原型系统消耗更少的存储空间。这是因为本研究对划分分区中的所有边数据使用列式存储布局,并对一个分区内的数据进行压缩。JanusGraph 比 Cassandra 消耗更多空间,因为它将每条边(边 ID)存储 2 次(JanusGraph 存储顶点的时候会存储关联的边信息)。由于压缩策略不同,本研究原型系统存储单个备份消耗的空间比 Cassandra 和 Greenplum 都少。

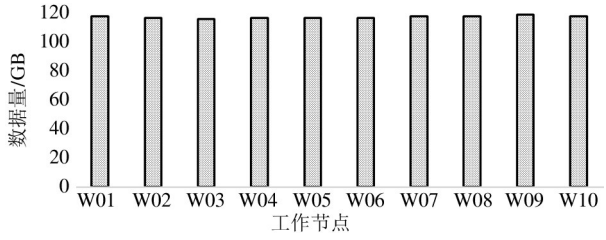
5.4.2 分布式环境下各工作节点的负载情况

本实验旨在探究本文提出的划分方案对各工作节点数据分布的影响。为此,本实验统计各个工作节点上存储的不同的存储布局的数据量,如图 5 所示。横坐标显示的是每个工作节点,纵坐标显示的是数据大小(以 GB 为单位)。由此可见,本研究提出的划分方案实现了数据的均匀分布,即各个工作节点上存储的数据量基本一致。值得一提的是,基于空间-时间划分的存储布局的数据量大约是基于

对象-时间划分的存储布局的数据量的 2 倍。这是因为,在实验过程中存储了 3 份数据(即 3 备份),其中基于空间-时间划分的备份存储了 2 份,而基于对象-时间划分的备份存储了 1 份。



(a) 基于对象-时间划分的备份



(b) 基于空间-时间划分的备份

图 5　不同存储布局的数据在各工作节点上的分布情况

6　结　论

时空图数据是现实世界普遍存在的一种大数据,具有巨大的、社会和经济价值。时空图数据在数据量和数据更新速率两方面具有独特的特征,可以用来优化大规模数据存储和查询分析。然而,现有的成熟的大数据存储和分析系统没有考虑结合数据特征和查询特征做针对性的优化,因而无法很好地应对大规模数据的挑战,存储和分析能力都有待加强。本文针对时空图数据,分析大数据存储和分析系统在存储和查询两方面的需求,并提出高效的、针对性的解决方案。实验结果表明,和现有方案(如 ST-Hadoop + Spark、JanusGraph、Greenplum、Cassandra + Spark)相比,本研究提出的优化方法能减少 1.7 ~ 5.4 倍的存储空间,提高 1 ~ 4 个数量级的查询性能。

参考文献

[1] JanusGraph. JanusGraph[EB/OL]. [2021-12-08]. http://janusgraph.org/.

[2] Greenplum. Greenplum[EB/OL]. [2021-12-08]. https://greenplum.org/.

[3] Spark. Spark[EB/OL]. [2021-12-08]. https://spark.apache.org/.

[4] Hadoop. Hadoop[EB/OL]. [2021-12-08]. https://hadoop.apache.org/.

[5] LOUAI A, MOHAMED M. A demonstration of ST-Hadoop: a mapreduce framework for big spatio-temporal data [C]// Proceedings of the 43rd International Conference on Very Large Data Bases. Munich: VLDB, 2017: 1961-1964.

[6] 吴张峰, 李成仁. 多源时空大数据在疫情防控中的应用[J]. 信息与标准化, 2020, 5: 18-41.

[7] 周松, 沈蕾, 王伟. 从时空大数据的角度分析评价江苏省新冠疫情发展趋势[J]. 现代测绘, 2020,43(3): 5-10.

[8] 张翔宇, 张强, 吕明琪. 基于 GPS 轨迹挖掘的兴趣地点个性化推荐方法[J]. 高技术通讯, 2021, 31(1): 75-83.

[9] GUOHUI L, QI C, BOLONG Z, et al. Group-based recurrent neural networks for POI recommendation [J]. ACM/IMS Transactions on Data Science, 2020, 1(1):1-18.

[10] 百度百科. 套牌车[EB/OL]. [2021-12-08]. https://baike.baidu.com/item/%E5%A5%97%E7%89%8C%E8%BD%A6/5341795.

[11] 李敏茜. 基于轨迹数据的套牌车检测技术研究[D]. 上海:华东师范大学计算机科学与软件工程学院, 2019:1-95.

[12] AHMED E, MOHAMED M. SpatialHadoop: a mapreduce framework for spatial data[C]// The 31st IEEE International Conference on Data Engineering. Seoul: IEEE, 2015:1352-1363.

[13] INDERJIT D. Co-clustering documents and words using bipartite spectral graph partitioning[C]// Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining. New York: Association for Computing Machinery, 2001: 269-274.

[14] BIN G, TIE L, XIN Z, et al. Consistent bipartite graph co-partitioning for star-structured high-order heterogeneous data co-clustering[C]// Proceedings of the 11th International Conference on Knowledge Discovery and Data Mining. Chicago: Association for Computing Machinery,

2005:41-50.

[15] GEORGE K, VIPIN K. A fast and high quality multilevel scheme for partitioning irregular graphs[J]. SIAM Journal on Scientific Computing, 1998, 20:359-392.

[16] JOSEPH G, YU L, HAI G, et al. Distributed graph-parallel computation on natural graphs[C]//The 10th USENIX Symposium on Operating Systems Design and Implementation. Hollywood: USENIX Association, 2012: 17-30.

Optimizing large-scale spatio-temporal graph data storage and analysis

DING Mengsu[*,**], YANG Muqiao[***], CHEN Shimin[*,**]

([*]Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([**]School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100190)

([***]College of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh 15213)

Abstract

Spatio-temporal graph data analysis has huge political, social and economic value. Spatio-temporal graph data has unique characteristics in terms of data volume and data velocity, those characteristics can be used to optimize big data storage and analysis. However, existing solutions provide general support, and do not consider combining data characteristics and query characteristics to make optimizations. This paper analyzes the requirement of big data system for processing spatio-temporal graph data, and proposes efficient solutions to deal with the challenges of data volume, data velocity and query processing. The experimental results show that compared with the existing solutions, this study can reduce the storage space by 1.7× – 5.4×, and the query performance can be improved by 1 – 4 orders of magnitude.

Key words: spatio-temporal graph storage, spatio-temporal graph query, big data storage, big data analysis