时间序列数据摘要与索引机制

A Time Series Data Summary and Index Mechanism

曾碧贵[1] 叶少珍[1,2] (1 福州大学数学与计算机科学学院,福建 福州 350108;
2 福建省医疗器械与医药技术重点实验室,福建 福州 350002)

摘要:高容量的时间序列可视化分析在金融、离散制造等众多行业中应用越来越普遍,但高容量时间序列可视化在处理可视化延时需求和高摄取率上比较困难。针对该问题,提出一种基于 M4 的概要信息聚合预存储和索引机制。首先,将原始的时间序列数据根据不同压缩比例分层处理,每层处理中数据等距离分割成若干段,并选择每段中时间最大最小和序列值最大最小时对应的 4 个最值点;其次,将聚合后的数据存储在时间序列概要树上;最后,实现基于概要树索引查询算法。通过模拟生成时序数据集进行了实验分析,结果显示该索引机制与原有的 M4 聚合查询方法可达到更好的查询效果。

关键词:时序数据,数据约减,概要树

Abstract:This paper proposes a method that is the M4-based summary information pre-store and index mechanism. First,compressing the original time series data by different compression ratio hierarchical,dividing the data into equidistance segments in each layer,and select the maximum minimum sequence,the largest and most hours value in each period of time.Second,insert the aggregated data into a time sequence summary tree.Last,implementing the index query algorithm based on summary tree.

Keywords:time series data,dimensionality reduction,summary tree

目前,时间序列数据来源于各行各业,主要包括传感器网络、智能电网[1]和金融市场等[2],各行各业都在收集海量的大数据,且大容量收集后时间序列数据存储在不同的数据库文件中。而数据库作为可视化数据分析的重要工具,在数据分析和数据可视化两者不断相互交互,需要保持查询结果与原始时间序列数据一致性。目前比较流行可视化工具包括 Tableau、SAP Lumira 等,这些工具在可视化查询数据集时没有考虑查询返回的结果数据的基数大小,在面对大量数据查询时,查询结果集通常包括上百万条的记录,这样导致查询过程中需要非常大的磁盘开销、网络带宽消耗和等待时间[3]。本文提出了一种概要信息预存储方法,是面向可视化的时间序列概要信息聚合与信息预存储相结合的一种方法。

1 时间序列数据概要树构建

常见时间序列数据归约方法有很多种,最简单分段聚合近似(平均)方法、基于线性简化法等[4]。大部分的时间序列维度约减方法都很一般,而且不是针对可视化设计[5]。M4 聚合方法是目前仅有的面向光栅线可视化的时间序列数据聚合方法,在可视化查询过程中对查询结果集数据进行 M4 聚合后再返回到可视化客户端上。其中 M4 聚合方法对原始时间序列进行分组,每组的 k 个连续的时序数据最终被 min(v)、max(v)、min(t)和 max(t)4 个概要数据点替代。min(v)为该分组中值最小的数据点,max(v)为值最大的数据点,min(t)为时间最小的数据点,max(t)为时间最大的数据点。

表 1 聚合前 16 个数据项

时间	值	时间	值	时间	值	时间	值
T1	2	T5	6	T9	6	T13	5
T2	4	T6	1	T10	8	T14	6
T3	5	T7	3	T11	9	T15	8
T4	6	T8	7	T12	7	T16	6

在时序数据中,k 个在时间上连续的数据项,选择 4 个最值信息和时间戳聚合表示原来的 k 个值。例如,在 t1~t16 时间内连续有 16 个数据项,假设对每 16 个数据项的概要信息进行一个聚合,如表 2 所示。

通过对数据进行聚合,大幅度减少了所需查询的数据量。但当时间序列数据的增多,聚合后的数据也会相应增长,仍不利于快速查询。因此文本在数据聚合的基础上建立概要树索引。

表 2 聚合后 4 个数据项

t1~t16			
t_min=t1	v_min=t6	v_max=t11	t_max=6
2	1	9	6

1.1 M4 数据预存储原理

对原始时间序列数据 T(t,v)根据用户需求以不同分辨率进行预存储,形成数据量由大到小的分层数据存储形式[6],对压缩数据进行存储,在可视化过程中,客户通过选择一个与显示区域最相近的缩放粒。从该缩放粒度对应的概要树查找所需数据,并进行计算和判断,分层数据存储原理如图 1 所示。可以看出,数据的顶层代表的是原始的时间序列数据,从顶层开始每 g(g 为缩放粒度,g>4)个相邻的像素数据经过 M4 重新采样后生成 4个新的数据,代替原先的 n 个数据,依此重复进行,直到数据的存储开销超过预期存储开销则停止。每层数据都有相对应缩放粒度 g,在可视化过程中进行放大或者移动等操作时,可计算出当操作所需缩放粒度和已经建好的数据缩放粒度相匹配,哪层数据层的分辨率最接近用哪层的数据来显示。

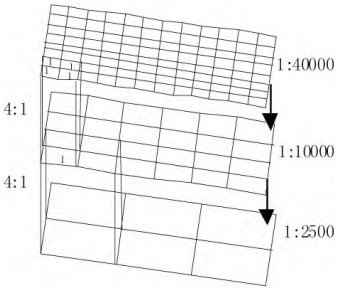

图 1 分层结构原理(g=16)

1.2 概要树

B+树是一颗 m 阶动态搜索树,每个结点最多含有 m 个关键字和 m 个指向记录的指针,所有的叶子结点彼此链接构成一个有序的链表。本文提出了基于 B+树的概要树,概要树主要由数据叶子结点和中间结点构成。由多个数据项聚合后的概要信息加上特定的时间戳信息构成了叶子结点。当树的阶为 M 时,由 m 个叶子节点或 m 个中间结点加上特定的时间戳信息构成了中间结点或根结点。在建立概要树索引时,所有的数据点依据时间排好序依次添加到树中。

1.3 算法实现

设原始数据为 T(t,v),对于原始数据通过 M4 聚合原理生

成多层子数据文件 ChildrenData(t,v),并将聚合后的新数据写入相对应的概要树中。我们的时间序列预存储算法主要包括概要数据子层的创建和概要树创建两个步骤。因为存储多层子数据文件的会增加数据库的膨胀率,为了能够快速查询,以一定的数据膨胀率为代价,预先存储概要信息。Smax(D)即最大额外存储开销,当超过这个值,将停止计算;否则,数据集继续聚合。缩放粒度 g 即数据压缩比例,如 g=20,则时间戳为 20s,每层数据的缩放粒度 g 值不一样,按一定比例增大。g_indexs[]依次为从上到下每层聚合数据的压缩粒度集合。

1.3.1 子层数据构建实现

在构建子层数据时,对父亲数据集进行遍历,连续的 g 个数据放入一个 Alist 中,对 Alist 进行按值排序和按时间排序,并取出值最大的数据结点 Vmax,值最小的数据结点 Vmin,时间最大的数据结点 Tmax,时间最小的数据结点 Tmin,并放入子数据集中,数据量由原先的 g 个减少到了 4 个。

1.3.2 概要树构建实现

概要树的插入仅在叶子结点上进行,当第一个叶子结点插入时,即根结点就是叶子结点,直接插入。接下来每插入一个时序数据时,都要判断结点中的子树棵树是否超出范围 m。当插入后结点中的子树棵数大于 m 时,需要将叶结点分裂为两个结点,它们的关键码分别为(m+1)/2 和(m+1)/2。它们的双亲结点中应同时包含这两个结点的最小关键码和结点地址。

2 概要树查询

根据数据分层存储的特点,可视化查询过程中首先根据查询语句 Q、查询时间 t_1 和 t_2、可视化屏幕宽度 w 计算,确定所需数据在哪一层;再根据原始查询语句查询所需所需结果集,判断查询结果集的数量并与可视化参数屏幕像素宽度 w 比较,是否满足可视化显示范围内,如果满足,直接返回原始查询 Q;否则,对原始查询重写 QR。以查询时间 t_1~t_2 对应数据项的概要数据为例,具体查询步骤如下:

1)计算数据层索引。假设原始时序数据是等时间间隔的,每秒连续有 50 组时序数据,查询一天内时间序列数据时间 t_1 和 t_2 将会有 30 * 60 * 60 * 24=2592000 条数据记录,这远超过可视化屏幕可显示的范围,所以需从预先存储聚合后的概要 B+树查找。假设屏幕 w=200,则屏幕最多可显示 4 * w=800 个数据点。则 2592000 到 800,需经过 3240 倍的压缩。预先存储的数据层由于膨胀率可能不存在压缩率 index=3240 对应的文件,则对 g_index [] 进行遍历,查找与 3240 差最小且小于 3240 的 index。则 index 对应的概要树即所需概要树;

2)查询概要树。根据步骤 1)的索引,查找对应的概要树。在概要树,采用顺序查找的方法,从根结点开始,根据给定是查询时间 t_1 与结点关键码 k_i 比较,若 $t_1<k_i$,则在查找该中间结点的下一个关键码或者父结点的兄弟结点;否则,$t_1>k_i$,则继续在当前关键对应的叶子节点查找,并开始遍历直到新的叶子结点的关键码 $k_i>t_2$ 结束,同时将满足条件($t_1<k_i<t_2$)的 k_i 相匹配的数据加入结果集 Q 中,并返回;

3)判断结果集的数量。在步骤 1)中提到由于数据膨胀率的限制可能不存在绝对理想的聚合数据集,所以须对步骤 2)返回的数据集进行判断。计算结果集的数量 Count(Q),比较 Count(Q)和 4 * w,若 Count(Q)<4 * w,返回 Q;否则,对 Q 重新查询;

4)Q 重新查询 M4。M4 查询重写本质上市对原始数据依据 w(屏幕像素宽度)等距时间跨度进行分组,每个组最后对应于可视化屏幕中的一个像素列。对于每个分组,M4 只选择 4 个概要信息表示每个分组,重新聚合后的结果集为最终所需数据集。

3 实验结果与分析

下面本文采用的虚拟数据集为 6 万条记录的时间序列数据为例,来说明基于 M4 的时间序列概要信息预存储算法的有效性。为了验证实验效果,实验环境处理器为 i5 的内核处理器,操作系统为 64 位 Windows 10 系统,4G 内存。

表 3 M4 和 M4_PS 查询数据量比较

	查询 1	查询 2	查询 3	查询 4	查询 5
原始查询(个)	18178	21778	25498	30372	39998
预存储查询(个)	3637	4357	5101	1230	1598
M4(ms)	375	497	588	604	674
M4_PS(ms)	398	255	354	107	91

表 3 第一行为用户查询原始数据集个数,如果在预存储后的文件中查询,可以减少查询的数据量即第二行数据,同时,当用户在处理后的文件查询效率远高于从原始数据中查询。随着查询数据量的增加,M4 的查询消耗时间相应提高,但是变化不明显,在一定范围之间波动,相对稳定。M4_PS 方法的查询消耗时间随着查询数据量的增加查询时间减少。

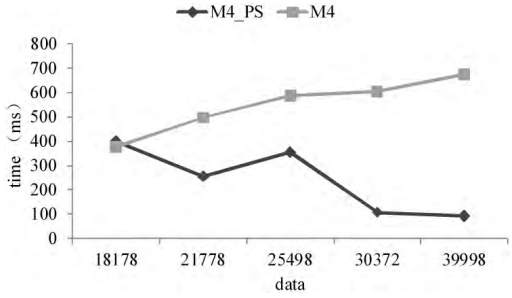

图 2 M4 与 M4_PS 查询效率对比图

从图 2 可以得出结论,当查询数据量越多时候,M4 算法所消耗的时间呈增长的趋势,而 M4_PS 呈下降的趋势,主要是 M4 花费大量时间在查询原始文件数据以提取符合条件的数据集,反之,查询时间少,效率提高,当查询数据量越大,M4_PS 可以提高几十倍的效率。

4 结束语

该方法为高容量时间序列可视化查询面临的低效率,高带宽消耗提供了新的思路,下一步将继续开发相应的查询引擎和概要树索引构建优化,使两者可以更好地结合。

参考文献

[1]Office of Electricity Delivery & Energy Reliability: Smart Grid (2014) [EB/OL].http://energy.gov/oe/technology –development/smart–grid.

[2]Jerzak Z, Heinze T, Fehr M, et al. The DEBS 2012 grand challenge [C]//Proceedings of the 6th ACM International Conference on Distributed Event –Based Systems. ACM, 2012: 393–398

[3]Navathe,Ramez Elmasri,Shamkant B.Fundamentals of database systems[M].6th ed. Pearson Education.2010:652–660

[4]Shi W, Cheung C K. Performance evaluation of line simplification algorithms for vector generalization[J].The Cartographic Journal, 2006, 43(1):27–44

[5]Fu T.A review on time series data mining[J].Engineering Applications of Artificial Intelligence,2011,24(1):164–181

[6]Kraak M J, Ormeling F. Cartography: visualization of spatial data[M].Guilford Press, 2011

[收稿日期:2016.9.23]