第45卷第10期 中国科学技术大学学报 Vol. 45, No. 10
2015年10月 JOURNAL OF UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA Oct. 2015

文章编号:0253-2778(2015)10-0813-09

基于 ADMD 融合策略的海洋大数据索引技术研究

黄冬梅,孙 乐,赵丹枫

(上海海洋大学信息学院,上海 201306)

摘要:海洋数据具有多源、多类、多维、海量等特点,是一种典型的大数据,海洋大数据上的快速查询是该领域各类应用的基本需求.提高查询速度的关键是建立一个完善的索引结构,为此提出了一种基于时间间隔 B+-tree 和 HSP-tree 的多层索引架构 ML-index(multi-layer index),分别制定样本驱动的数据融合机制(adaptive method of data merging strategy)以确定分布式时态数据分区;并基于海洋数据特性、数据单元饱和度等,提出了一种自适应空间划分方法(adaptive space partition),在此基础上建立 HSP-tree 作为辅助索引.实验验证在海洋数据模式下,提出的多层索引结构保证了海洋数据的查询速度,逼近线性的时间复杂度.

关键词:海洋大数据;时间间隔 B+-tree 索引;自适应空间划分;AMDM

中图分类号:TP311 **文献标识码**:A doi:10.3969/j.issn.0253-2778.2015.10.003

引用格式: HUANG Dongmei, SUN Le, ZHAO Danfeng. A composite index strategy for big marine data based on adaptive method of data merging strategy[J]. Journal of University of Science and Technology of China, 2015,45(10):813-821.

黄冬梅,孙乐,赵丹枫. 基于 ADMD 融合策略的海洋大数据索引技术研究[J]. 中国科学技术大学学报,2015,45(10):813-821.

A composite index strategy for big marine data based on adaptive method of data merging strategy

HUANG Dongmei, SUN Le, ZHAO Danfeng

(*College of Information, Shanghai Ocean University, Shanghai* 201306)

Abstract: Marine data fall easily into category of Big Data. A basic requirement for various marine monitoring applications is quick retrieval and the establishment of a sound index structure is of great importance. A multi-layer index (ML-index, for short) with regard to time interval B+-tree and hybrid space partition tree (HSP-tree, for short) was proposed. It employs the adaptive method of data merging strategy to optimize the primary key index (i.e. B+-tree). An adaptive space partition method was also proposed on the basis of data characteristics, and data unit capacity particular, for building secondary index, namely, HSP-tree. The experiment result shows that ML-index saves about 2/3 of the time in comparison with two state-of-the-art index methods.

Key words: marine data value function; time interval B+-tree; adaptive space partition; HSP-tree

收稿日期:2015-08-27;**修回日期**:2015-09-29

基金项目:国家自然科学基金项目(61272098);上海市自然科学基金项目(13ZR1455800).

作者简介:黄冬梅(通讯作者),女,1964 年生,教授,研究方向:数据挖掘. E-mail:dmhuang@shou.edu.cn

0 引言

近年来,信息技术的快速发展,尤其是信息获取技术、互联网、物联网以及社交网络等技术的突飞猛进,导致了各行业数据量的急剧增长,行业大数据已经成为目前研究的热点[1]. 信息化的发展,大数据的增长还将进一步攀升. 根据孟小峰等对当下大数据资料的归纳和总结,大数据的定义可在把握量的(volume,variety,velocity)的基础上仍需考虑其第4V特性(value)[2]. 海洋数据是一种典型的大数据. 国际上已经开展诸多观测计划,包括Argo、海王星、OOI等多个观测计划并发射了多颗海洋观测卫星. 如Argo计划目前在全球共投放浮标1 0231个,其中一个Argo数据中心在过去一年里就处理了657个活跃浮标观测的21 954条剖面数据;"水瓶座"2个月内采集数据量相当于调查船和浮标125年测量的历史记录;截止到2012年底,NOAA管理的年数据量高达30 PB[3-4]. 全方位、多手段的海洋观测现状使得海洋数据量大、增长速度快、类型多样化以及蕴含价值大等特点,无疑给海洋数据的快速获取、访问及应用带来了巨大的挑战.

获取数据、建立数据库并优化,这一传统的集中式数据存储方式对于快速获取目标海洋数据存在一定的局限性,然而加快数据访问速度的关键需要建立一个完善的索引机制,避免消耗大量的I/O冗余操作导致的数据定位弱等问题. 此外,数据的分析及数据空间划分是否合理,也会直接影响索引结构的性能[5]. 由于多源异构的海洋大数据存在着数量庞大、格式不一、质量不高等问题,因此本文在分布式环境下,综合分析海洋数据特征,形式化描述海洋数据,设计数据动态融合算法及数据自适应划分算法,并定制合理的索引框架,以实现海洋大数据集高度可控的管理需求. 主要贡献包括:

(Ⅰ)形式化描述海洋数据,提出一种基于样本驱动的AMDM(adaptive method of data merging)算法,确定及优化数据时态分区.

(Ⅱ)提出一种的新的数据划分策略——自适应空间划分,聚类敏感度及价值相似的数据集.

(Ⅲ)提出一种多层索引架构体系(ML-Index),基于AMDM的时间间隔B+-tree索引作为主键索引;基于自适应空间划分的混合空间划分树(HSP-tree)作为辅助索引.

1 相关工作

为了使分布式存储较好地适用敏感的空间海洋大数据,需根据不同的海洋大数据规模、分布及特征对其进行数据划分,建立适当的索引结构,以此提升超高维海洋大数据的查询效率.

如何进行数据的划分是影响拓展性、负载平衡以及系统性能的关键问题,它影响着数据访问速度以及数据利用效率. 现有数据划分方法主要从以下几个角度出发. 从数据敏感和密保级别的不同考虑数据划分的方法有:通过计算数据敏感度实现动态划分[6]、采取物理隔绝[7]及用户访问设限[8]等密文方式分流数据到相应节点. 此外,也有一些基于统计分析理论的数据划分方法,主要有:采取聚类方式对数据进行划分[9],根据数据分布自适应选择划分边界,或采用抽样[10]等方法来减轻海量数据的处理压力,避免数据倾斜,实现数据稳定及动态分布. 其中具有代表性的划分方法有由Zhou[5]等提出的根据关键维筛选大批量数据集,但存在着不能有效地进行数据动态扩充和扩维的劣势.

索引则是影响整个数据库系统效率的关键,是提高数据库系统执行效率的一种有效工具. 海洋大数据主要采用分布式环境,在此基础上的研究包括哈希结构索引[3]、树状索引[11]、以时间为主体的复合索引[12]、基于并行处理技术的优化索引[13]、随数据迁移而动态调节的索引[14]. Innodb、MyISAM等都是具有代表性的复合索引,使用B^+-树类存储结构,在叶子结点上存放索引键的相关信息以及指针. 这类引擎都具有主索引和辅助索引结构,在结构上没有任何区别,只是主索引要求key是唯一的,而辅助索引的key可以重复,辅助索引B+-树的结构难以处理多属性的海洋数据. K-D树及其变体[15]是一种非平衡的树结构,很难设计高效的搜索算法,却在空间数据有序划分上占有一定的优势.

分布式环境下大数据处理模型的基本要求是实时性. 针对海洋中数据的存储索引结构问题,上述一些结构算法都存在一定的不足,主要缺乏对数据的深度认识,尤其面对海洋数据的敏感性、空间性、时效性等属性特征. 本文以海洋数据为样本进行对比分析,融合数据的特性制定相关的服务策略,研究更适用于分布式平台下面向多维海洋数据的复合索引结构.

2　海洋数据的描述与定义

本节将对海洋大数据预处理中的相关概念进行描述与定义，具体包括海洋数据集、海洋数据模式以及样本驱动的数据划分策略等.

2.1　海洋数据定义

海洋数据是一种典型的大数据，其中每一组站点观测的海洋数据都具有时间信息，此外还有站点的经纬度等信息，作为现实世界中的一个海洋对象，其脱离了时间和位置信息，数据就失去了存在的意义. 针对海洋大数据本文给出海洋数据模式的定义如下：

定义 2.1　海洋数据模式. 海洋数据模式为 $R(A_1,A_2,\cdots,A_m)$，且 $(A_1,A_2,\cdots,A_m)$ 是海洋数据的属性集，其中，A_1 表示海洋监测的时间属性.

定义 2.2　海洋数据集. 海洋数据集 A 是一个三元组，$A=(R,N,S)$，其中，R 是海洋数据模式；N 为数据集 A 记录的数量；S 为海洋数据集 A 的大小.

2.2　Sample-driven 数据融合策略

预处理海洋数据，了解数据的分布特征有助于加快数据的查询效率. 故本文以海洋属性数据样本为驱动，设计了 AMDM 算法. 其采取聚类[15]的思想，通过计算时间序列中关键维 A_i（数值属性，如水温、盐度）数据区间的距离差，逐步 merge&split 得到最终稳定的数据划分.

假设一个数值属性值域中包括有 m 个不同的值，$I=\{x_1,x_2,\cdots,x_m\}$，其中属性值 x_1 的出现频次（属性值在值域中出现的次数）为 n_i. 该属性值出现频次的总数 $N=\sum_{i=1}^{m} n_i$. 为了满足属性值域的一致性，对其进行排序，使得 $x_i<x_{i+1}$ $(1\leqslant x\leqslant \mathrm{m}-1)$. $I_1,\cdots,I_m$ 表示为 I 上的一次完全互斥区间划分（区间互不相交），且每一个区间 I_i 都有一个参考中心值 c_i，初始定义为 $x_i(1\leqslant x\leqslant m)$.

假设一个区间 I_k 包含有属性值 $\{x_1,x_2,\cdots,x_k\}$，则 $N_k=\sum_{i=1}^{m} n_i$ 表示为属性值出现频次，且包含一个可控值 c_k，因此区间 I 的平均内部距离差可定义为：

$$\mathrm{Dist}(I_k,c_k)=\frac{1}{N_k}\sum_{i=1}^{k} n_i\mid x_i-c_k\mid \qquad (1)$$

假设有两个邻近区间 $I_i=\{x_1,x_2,\cdots,x_i\}$ 和 $I_{i+1}=\{x_{i+1},x_{i+2},\cdots,x_{i+j}\}$，其属性值出现频次可分别表示为 $N_i=\sum_{p=1}^{i} n_p$ 和 $N_{i+1}=\sum_{p=i+1}^{i+j} n_p$，参考中心值分别为 $c_i=\sum_{p=1}^{i} x_p\,n_p/\,N_i$ 和 $c_{i+1}=\sum_{p=i+1}^{i+j} x_p\,n_p/\,N_{i+1}(1\leqslant i+j\leqslant m)$. 合并区间 I_i 和 I_{i+1}，得到区间 I'，$I'=I_i\cup I_{i+1}$. 区间 I' 共有 $i+j$ 个属性值，其总属性值出现频次，$N_i+N_{i+1}=\sum_{p=1}^{i+j} n_p$，其参考中心值 c 可通过计算 (c_{i+1},n_{i+1}) 和 (c_{i+1},n_{i+1}) 的平均权重得到.

$$c=\frac{c_i\sum_{p=1}^{i} n_p+c_{i+1}\sum_{p=i+1}^{i+j} n_p}{\sum_{p=1}^{i+j} n_p}=\frac{c_i\,N_i+c_{i+1}\,N_{i+1}}{N_i+N_{i+1}} \qquad (2)$$

参考中心值 c 所在区间 I 的内部距离差可由式(1)计算得到，当 $x_i\leqslant c\leqslant x_{i+1}$，给定上述参数 c_i，c_{i+1}，$\sum_{p=1}^{i} x_p\,n_p=c_i\,N_i$，$\sum_{p=i+1}^{i+j} x_p\,n_p=c_{i+1}\,N_{i+1}$，合并区间 I' 的距离差计算如下：

$$\begin{aligned}
&\mathrm{Dist}(I_k,c_k)=(N_i+N_{i+1})^{-1}\sum_{p=1}^{i+j} n_p\mid x_p-c\mid=\\
&(N_i+N_{i+1})^{-1}(\textstyle\sum_{p=1}^{i}(c-x_p,\,n_p+\\
&\textstyle\sum_{p=i+1}^{i+j}(x_p-c)n_p)=\\
&(N_i+N_{i+1})^{-1}(\textstyle\sum_{p=i+1}^{i+j} x_p\,n_p-\sum_{p=1}^{i} x_p\,n_p+\\
&(\textstyle\sum_{p=1}^{i} n_p-\sum_{p=i+1}^{i+j} n_p)c)=\\
&(N_i+N_{i+1})^{-1}(c_{i+1}\,N_{i+1}-c_i\,N_i+\\
&(N_i-N_{i+1})\frac{c_i\,N_i+c_{i+1}\,N_{i+1}}{N_i+N_{i+1}})=\\
&2\,N_i\,N_{i+1}\,(N_i+N_{i+1})-2(c_{i+1}-c_i)
\end{aligned} \qquad (3)$$

基于上述公式，相邻区间 I_i，I_{i+1} 之间的距离差 $(\mathrm{Diff}(c_i,c_{i+1}))$ 可定义为 $(N_i+N_{i+1})\mathrm{Disc}(I',c)/2$. 由上述定义可获知，区间距离差首先由各相邻区间的参考中心值的距离差决定；其次可由每个区间内的属性值出现频次作进一步判断. 如若两对区间具有相同的距离差，则每一区间内属性值出现频次高的区间对，其距离差相对较小.

接下来我们将考虑如何选择邻近区间进行合并. 给定 m 个连续区间 $I_1,\cdots,I_m$ 及其各自参考中心值 $c_1,\cdots,c_m$，共产生 $m-1$ 对相邻区间对. 迭代测试每一对相邻区间，每次合并 Diff 差值最小的两个相邻区间 (I_t,I_{t+1})，得到合并区间 $I_t=I_t\cup I_{t+1}$，其对应的参考中心值 c_t 及 N_t 表述为：$c_t=(N_t\,c_t+$

$N_{t+1}\ c_{t+1})/(N_t+N_{t+1})$，$N_t=N_t+N_{t+1}$.

迭代执行 AMDM 算法，进行区间合并，直到满足以下规则则停止其操作：①如果每个数据区间的密度超出磁盘或者单个数据块的空间容量，即 $N>S_i/cs_i$；②每对相邻区间对的参考中心值距离差大，不可能出现在同一组域内，如 $(c_{i+1}-c_i)>3\bar{d}$，其中 $\bar{d}$ 表示为数值属性中相邻值间的平均距离.

定义 2.3 空间单元饱和度. 划分后空间单元 c_i 的饱和度 $\theta=\frac{S_i}{cs_i}$，其中 S_i 为该单元 c_i 当前容纳的数据点的数目，cs_i 表示空间单元 c_i 的可存储空间. S_i 越小，数据的搜索越快响应完成.

通过执行上述程序，数值属性的值域被划分成一组相邻数据区间的集合，其中每一单元内数据量满足了特定的系统负载及用户需求，且每一区域也相互独立，不足以被合并到相邻区域.

算法 2.1 AMDM algorithm

输入：由关键数值属性 attributes 构成的有序序列$\{x_1,x_2,\cdots,x_m\}$

输出：互斥间隔序列$\{I_1,\cdots,I_{m'}\}$

1：for each ($1\leqslant i\leqslant m$) do

2：Let I_i contains x_i；

3：Let $c_t=x_i$ be the representative centre of I_i；

4：End

5：Let $m'=m$；

6：for each interval pair (I_i, I_{i+1}) ($1\leqslant i\leqslant m'$) do

7：Let $\mathrm{Diff}(c_i,c_{i+1})=\frac{N_iN_{i+1}}{N_i+N_{i+1}}(c_{i+1}-c_i)$；

8：End

9：While ($N>S_i/cs_i$ or $(c_{i+1}-c_i)>3\bar{d}$) do

10：Let t be such that $\mathrm{Diff}(c_t,c_{t+1})$ is minimal；

11：Let $c_t=\frac{N_tc_t+N_{t+1}c_{t+1}}{N_t+N_{t+1}}$；

12：Let $N_t=N_t+N_{t+1}$；13：Merge I_t and I_{t+1} into a new interval I_t；

14：Let $m'=m'-1$；

15：Recompute Diff (c_{t-1}, c_t) and Diff (c_t, c_{t+1})；

16：End；

17：Output intervals $I_1,\cdots,I_{m'}$

3 ML－Index 索引的描述与构建

本节首先给出分布式环境下的复合索引框架 ML-Index；其次，分别介绍 ML-Index（multi-layer index）框架下全局索引及局部索引 HSP 的实施步骤及相关算法.

3.1 ML-index 分布式多层索引架构

Hadoop 是一种主从结构，本文在此基础上设计的索引模式包括全局索引以及局部索引，其基本原理是通过全局索引找到计算结点；再由局部索引树检索得到最终的数据对象. 文中使用时间间隔 B^+-tree 作为全局索引，HSP 树作为局部索引，该结构可满足大批量查询服务下的并行运算处理，如图 1 所示.

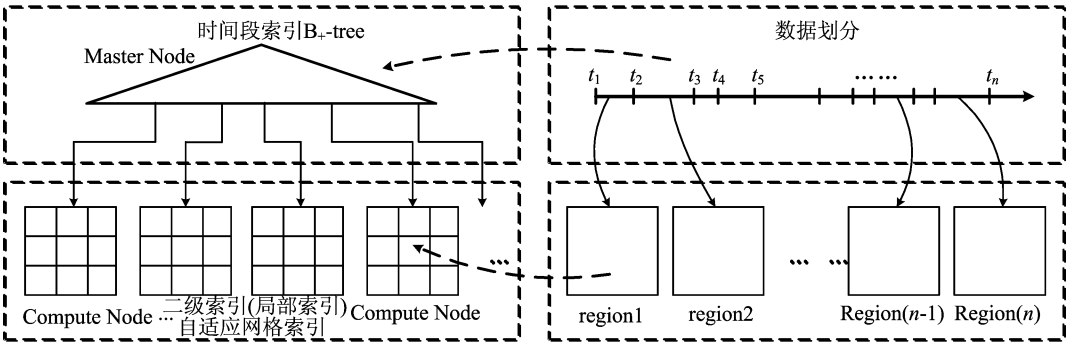


图 1 ML-index 多层索引框架

Fig. 1 Mutli-layer index frame

在图 1 框架上的查询处理主要分为两个步骤. 第一步，查询接收器通过全局索引找到所有与之相关的节点，并建立查询链接. 第二步，采取并行的方式处理查询并将最终的信息返回到客户端.

3.2 时间间隔 B^+-tree 索引

海洋时间序列时态可控，因此可将其划分成若干个数据子集，TIS=$\{[t_0,t_1]$，$[t_1,t_2]$，…，$[t_{i-1},t_i]\}$，且彼此互不相交. 基本操作如下：①提取数据

集，以时间为关键维等距划分；②挖掘关键数值属性（numeric attribute）维度，经由 AMDM merge&split 操作后获取最终 TIS 数据集；③使用 B^+-tree 索引该时间划分结果作为全局索引.

例 3.1　给定数据集 A，其关键数值维为水温，经由 AMDM 得到其数据区间划分 $I_1,\cdots,I_4$，其值域分别为 [0,21.5)，[21.5,25)，[25,28)，[28,～)．数据集 A 根据上述边界值逐步递进分割，获得如图 2 所示的时间间隔划分，并构建 B^+-tree.

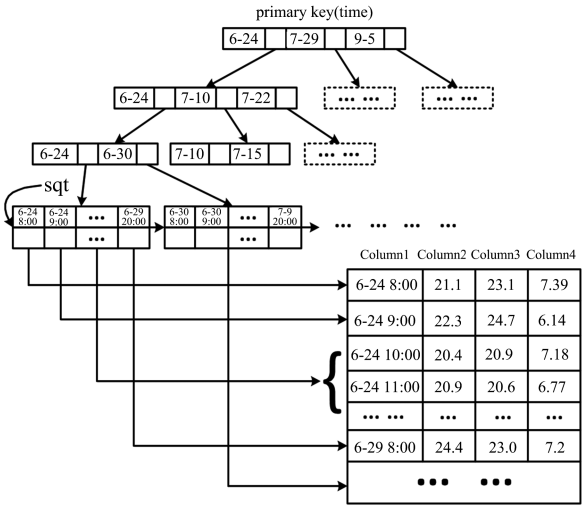


图 2　时间间隔 B^+-tree 索引图

Fig. 2　B^+-tree index for time intervals

3.3　局部索引

如上述 ML-index 索引结构，每一个从节点表示为一个计算节点，且为每一节点由两部分内容组成，Node＝combine(HashObjID, HSP)．①计算节点由网络的形式组织，并对每一个节点其进行一致性哈希编码，得到一个固定的 HashObjID 作为其局部索引的标识符，如图 3 所示；②为每一节点建立对应的局部索引 HSP 树，以实现数据的高效管理以及查询响应．每一个计算节点都享有一个独立存储空间，主节点（master node）负责发送及接受查询请求.

当查询请求响应时，该结构通过全局索引找到数据存储位置．为了进一步加快查询计算节点查询响应能力，本文设计了基于自适应空间划分（adaptive space partition）的辅助索引结构 HSP-tree.

例 3.2　给定时间间隔 $[t_1,t_2]$，该计算节点内的数据对象组成若干个聚集区域，计算每一单元的空间饱和度（定义 2.3），根据每一维迭代划分，最终

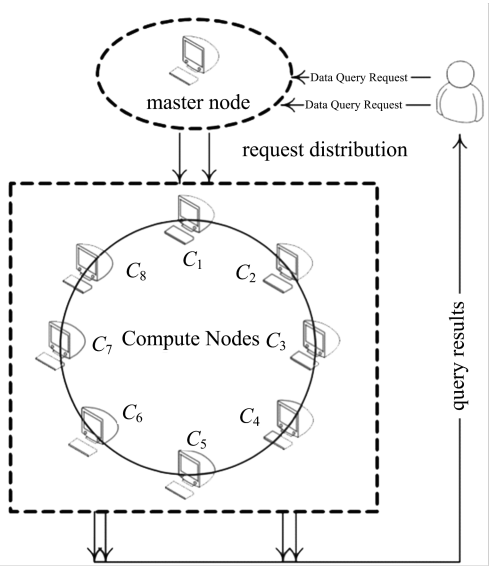


图 3　节点布局

Fig. 3　Node layout

得到特定空间和时间内的数据划分结果，我们把这种方法称之为“自适应空间划分”．在此基础上，索引其划分结果并建立 HSP 树，如图 4 所示.

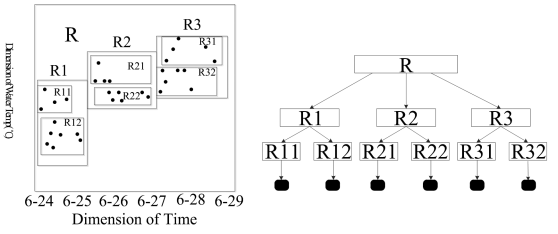


图 4　自适应空间划分及其遍历树

Fig. 4　Adaptive space partition and its index tree

3.3.1　Adaptive Method of Space Partition

数据空间的划分直接影响索引的查询性能[5]．本小节将描述自适应空间划分方法的实现原理，并基于此构建辅助索引树.

定义 3.1　等距划分．给定海洋数据模式 R 下 M 维属性空间 $S=D_1\times D_2\times\cdots\times D_M$，其中 D_i 为 S 中的第 A_i 维的数据域．假设现有参数 k 表示为将 S 中的每一维划分为 k 个相等间隔段，形成 k^M 个互不相交的超矩形单元，则称这种方法为等距划分.

如若对数据集进行等距划分，随着维度的增长，空白块将成倍增长，造成存储利用率低，因而遍历空间消耗大．本文将根据每一维中数据规模和数据分布特征划分的方法称为自适应划分方法，具体步骤如下：

Step 1　获取海洋数据模式 R 中各属性（维度）A_i 的数据域（D_i），并投影到数据空间；

Step 2 如定义3.1所述将各个维度A_i划分成k个等间距的小区间;

Step 3 计算每一维中的每一个区间中数据记录个数,以柱状图的形式直观表示;

Step 4 设定一个阈值th,从左到右将相邻区间的区间值在阈值以内的进行合并,获得各维数据的区间划分$R[i,j]$(第i维第j个区间的范围);

Step 5 如果最终划分合并得到一个单元,也就是个数等于1,数据相应维均匀分布,继而等间距划分;如若不是,则根据Step 4获取的划分情况作为相应维的初步自适应区间划分.

Step 6 获取初步划分结果,迭代判断每一个单元的空间单元饱和度.当经合并后的单元饱和度$\theta > \mu$时,则继续根据维度交替执行自适应划分,直至低于单元饱和度阈值,最后形成若干个数据聚集区域.

3.3.2 自适应划分算法

本文设计的初自适应空间划分的具体算法如下:

算法3.1 GetAdaptiveSP(A, k, th)

输入:A=(R,N,S),$|A_i|$表示第i维A中记录的个数(粒度)、初始区间划分个数k_i以及合并阈值th_i;

输出:区间划分结果SP

1: for each dimension A_i do

2: Partition $|A_i|$ into intervals r_m, m=1,2,…, k_i

3: Compute the histogram for each unit of A_i;

4: if ($val_{m+1} \in [val_m(1-th_i), val_m(1+th_i)]$) then

5: Merge r_m and r_{m+1} into r_m;

/*从左到右将相邻区间的区间值在阈值th_i以内的进行合并*/

6: if (number of bins == 1) then

/* 如果合并后只有一个单元,则数据集A_i均匀分布*/

7: Equal partition dimension A_i;

/*迭代自适应划分单元如若*/

8: else for each i (i <C_i) do

9: for each j (j <C_i) do

/*j表示海洋数据集A上第C_i上的数据集*/

10: if ($\theta=\frac{C_i}{cs_i}$) then

11: Further adaptively partition sub-dataset

12: end

13: end

15: Get the final SP ;

16: end

该算法是一种基于空间和数据的划分,利用海洋数据相关性特点实现数据规律性分布.其计算压力关键在于对每一个单元格内记录个数进行计算,将海洋数据单元的个数记为C_i,则该算法时间复杂度为$O(n)$;如果其数量较大,将其视为n,则该算法的时间复杂度为$O(n^2)$.

3.4 HSP划分索引树

通过上述根据空间和数据分布特征的空间划分方法,得到一种相对稳定,符合某海域海洋数据特征的数据划分.本文采用混合空间划分索引树(HSP-tree)存储上述时间段的最终数据划分结构.

定义3.2 混合空间划分索引树(HSP-tree).海洋属性数据集A在M维空间划分下的混合空间划分树结构定义如下:

(Ⅰ)它有一个根结点,最优情况下共有$M+1$层;

(Ⅱ)当遍历时,空间树的层与属性相对应,由判定器交替变换维度(属性)进行索引,叶子结点层对应所有非空数据单元;

(Ⅲ)除叶子结点层,非叶子结点包含两部分内容:位置信息以及指向下一层的指针NextLayer.如果为倒数第二层的结点,则指针指向叶子结点.

(Ⅳ)从根结点到叶子结点的一条路径对应一个数据单元.

HSP-tree树不仅有效地保留了数据的空间位置信息,同时保存其相对位置信息,能够更高效地支持空间近邻范围查询;同时,该辅助索引有效地避免了离散空间分布造成的冗余存储信息,减轻了查询遍历时间消耗;此外,由于采用空间索引树,大大减轻了频繁更新和插入造成的时间开销.

3.4.1 HSP-tree时间复杂度分析

(Ⅰ)插入操作.设定总共有N个样本数据,数据的维度数为d,每维被划分成m等分,合并后在极端坏的情况下每个单元都是密集的,保持原有划分,单元格的个数为m^d个;最优的情况下只有一个密集单元格,进行等距划分即可,即k^d,k为对密集单元格划分的个数.通常情况下建立HSP-tree共有$2d$层.每个数据点插入到HSP-tree中需经过$2d$-1层,

每一层最多比较 $\log_2 m$ 次，则每插入一个数据点最多进行 $2d\log_2 m$ 次比较，总共有 N 个数据则需要进行 $N\times 2d\log_2 m$（d，m 为常数），所以此步骤的时间复杂度为 $O(N)$，是一种线性的插入方法.

（Ⅱ）范围查找. 由于在空间划分之前，以时间为关键维高度筛选数据集 A，避免扫描大量不相关数据块，因此关键点在于查找范围覆盖最多的单元格的近邻单元格，且每个单元格又有 $2d$ 个近邻单元格，最坏的情况下每个近邻单元格都包括在内，共执行 $m^d\times 2d$ 次单元格查找操作，然而每次单元格查找操作的时间复杂度为 $d\log_2 m$，所以整体的时间复杂度为：$O=O(2d^2m^d\log_2 m)$. 由于 HSP-tree 仅仅索引 $\frac{C_i}{cs_i}>0$ 的单元格，划分后的数据单元的数目不仅远远小于 m^d 个，非空单元格的最大数量达到 N/k（k 为每个单元格最多能容纳的数据点数），所以式子可改写为：

$$O\leqslant O(\frac{2\ d2\ \log_2 m}{k}\times N).$$

因为 k，d 和 m 都为常数，所以 $O\approx O(N)$，也就是该步的时间复杂度也趋近于线性查找.

（Ⅲ）精确查找. 精确查找也可称为具体点搜寻，只需找到对应的块，因此唯一的磁盘 I/O 操作就是读入该块所需的操作. 最坏的情况下 HSP-tree 经过的层数为 $2d$，每一层需要比较 $\log_2 m$，进行 $2d\log_2 m$ 次比较，所以此步骤的时间复杂度为 $O(N)$，是一种线性的查找方法.

3.4.2　范围查询处理

给定一个二维时间范围查询 $Q(Y_0,Y_i)$，Y_0 表示时间维度 A_0 数据域 D_0 内的时间范围，$Y_0\in\bigcup_{R_0^i}^{R_0^j}\{D_0\}$，同样，$Y_i$ 表示除时间维度 A_0 外第 A_i 维数据域. 首先，用户通过时间间隔 B^+-树索引来找到对应的时间间隔段；对于每一个时间间隔段，用户可以通过 Y_0 对应的辅助索引——混合空索引树找到与 Y_i 相关的区域；接下来在与 Y_0、Y_i 相关的所有候选区域中获取用户想要的数据：①如果该区域的范围正好是查询范围 (Y_0,Y_i) 的子集，那么直接扫描这个区域. ②否则当 (Y_0,Y_i) 与给定的区域的范围的重叠度大于事先定义的阈值，则需扫描整个区域并且使用相应的查询条件筛选数据.

具体实现代码描述如下：

输入：$Q(Y_0,Y_i)$

输出：R_Q/ * 范围查询所得数据集 * /

1：$R_Q\leftarrow I$

2：$S_Q\leftarrow I$/ * 初始化相关区域为空 * /

3：$T_Q\leftarrow$getHSPtree(B^+-tree，Y_0)

4：for each HSP-tree T in T_Q do

5：$S_Q\bigcup$getRegionFromHSPtree(T，Y_i)

5：end / * 从 HSP-tree 获取与 B^+-tree 时间段相关的区域 * /

6：for each region R in S_Q do

7：if abstract(R)$\subseteq$ (Y_0，Y_i) then

8：$R_Q\bigcup$scanRegion(R)

9：else if overlap(abstract(R)，(Y_0，Y_i))＞then

10：for each record r in R do

11：if r$\in$(Y_0，Y_i) then

12：$R_Q\bigcup$r

13：end

14：end

该范围查询的具体性能评估将在实验部分给出，在范围查询的时候，初始阈值 ε 的设定相对较大，采取的是一种启发式的搜索方法，避免 ε 设置过小回到父结点重新搜索.

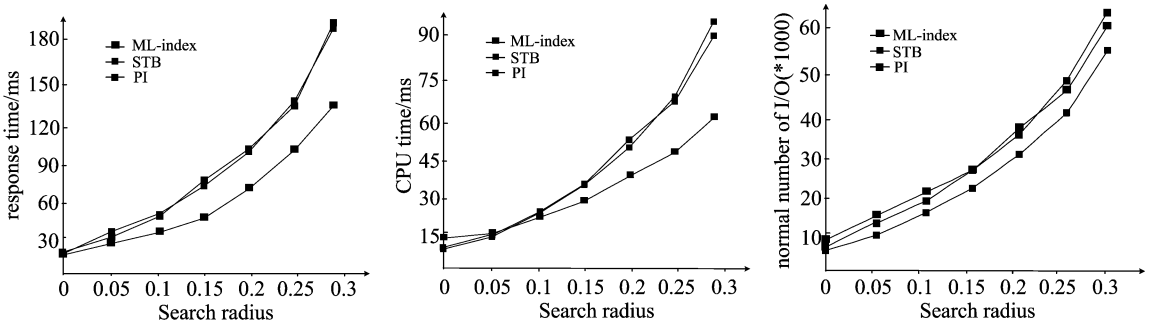
4　实验

本文采用的真实数据集为某海域海水浴场监测 1973～2012 年的海洋数据集，从原始数据集中提取海洋水温、气象监测要素数据集，只考虑一些常用的特征属性（考虑到海洋监测数据要素非常多，选择水温、浪高、气温、类大肠菌群等频繁使用的属性.）

海洋数据情景再现查询类型为范围查询和精确匹配查询. 精确匹配查询实际上可以视为一种特殊类型的范围查询，只需将查找范围设为 0 即可. 因此本次实验主要评估多层索引结构（ML-index）、TI[16] 和 STB[17] 的范围查询性能，并对 3 种索引技术性能进行分析比较，并给出实验对比结果. 本次实验使用真实的海洋离散数据，从以下三方面比较它们的性能：①响应时间. ②CPU 耗时. ③I/O 次数. 所有测试使用的软硬件环境中，（Ⅰ）硬件环境：Spark 云环境，6 workers，16G RAM 内存，1TB 硬盘；（Ⅱ）数据库系统：轻量级关系数据库系统；（Ⅲ）编程环境：Microsoft Visual C++ 编译器，python.

实验分别在平台上部署 ML-index（ML，for short）、TI 和 STB 三种索引结构，搭建实验环境. 图 5 展示了 3 种索引结构在范围查询下的性能对比情

况. 由图5可知,当查询半径较小时,本文的多层索引结构比TI和STB需要更少的查询时间和CPU耗时,但却需要更多的I/O次数. 其原因是ML-index采用了启发式原则,在设计范围查询算法时其初始化阈值ε设置较大,过滤的数据较少,且随着查询的进行,ε逐步调整适应到最佳值,其优势逐渐凸显. 当查询半径为0时,由于ML-index通过事先预处理分析得到数据分布特征,因此能快速定位到一个相对较小的数据单元,查找效率较快,而PI逼近于顺序扫描,查询响应速度偏慢. 在同等条件下,STB具有高效的数据过滤机制,但是增加了过多的空间消耗.



(a) 范围查询响应时间 (b) 范围查询CPU耗时 (c) 范围查询读写操作

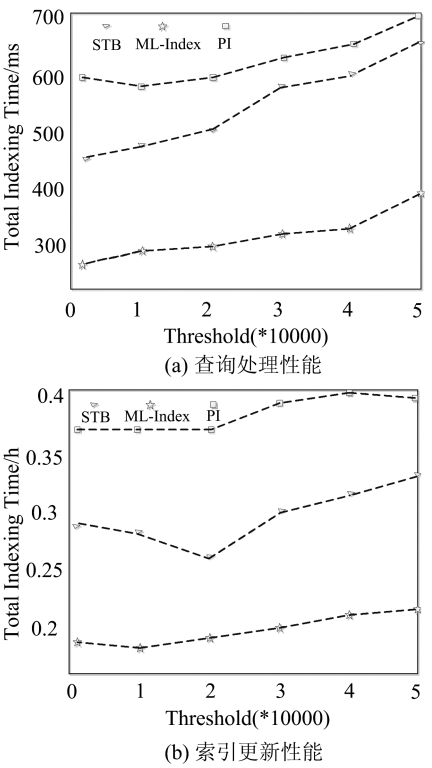
图5 性能曲线对比图

Fig. 5 Comparison of response time

从图5(a)可以发现,随着数据集的增大,ML索引结构的查询响应速度相对较快,优势明显;随着查询半径的增加,PI、STB两者由于是基于关键单维的选择,过滤数据的能力减弱,因此其查询响应耗时较长. 从5(b)性能曲线上可以发现,ML复合索引结构所需CPU时间较少,且随着查询半径的增大,三种索引性能差距增大,PI的处理压力逐步增大. 从图5(c) I/O操作次数来看,ML索引增长较快,随着半径的增大,ML和STB两者I/O性能曲线出现了交叉,但总体趋势上由于数据量的逼近,读写操作变化不大. 此外,PI中叶子结点由于存放了数据访问地址,虽然查询时I/O操作较高,但是具有较好的存储效率,而STB和ML索引结构其叶子结点保存了完整的数据记录,查询时造成了CPU耗时较高. 总体上对范围查询来说,本文提出的索引结构ML-Cloud索引框架是一种更为有效的索引技术.

图6为给定阈值下的三种索引的更新建立耗时以及查询处理耗时. 本文提出的ML-index由于事先对数据进行了分类和划分,因此在给定的阈值条件下表现稍显突出,如当数据单元阈值为1×10^4时,比起PI的0.32小时平均耗时以及STB的0.26 h相比较,ML-index在16G内存下仅需0.15 h完成最后索引的构建,并且能够降低5.7%的查询耗时.

实验验证了数据空间的划分会直接影响索引的查询性能. 同时,海洋业务化平台下,基于80%的查询都聚焦在20%的数据对象上,因此分析海洋大数据的特征和明确更优的数据划分结构能够完善制定



(a) 查询处理性能

(b) 索引更新性能

图6 性能曲线对比图

Fig. 6 Comparison of Query processing performance and index update performance

适用于海洋大数据的索引结构.

5　结论

索引是提高海洋业务化平台查询速度的关键，而数据空间的划分直接影响索引的查询性能. 本文在云环境下研究面向海洋数据的索引结构并分析其数据特征及分布情况. 实验表明，本文方法相对减少了查询数据的时间，并且随着数据量的不断增加，主从结构索引的优势越来越明显；此外，基于自适应划分策略构建的 HSP-tree 局部索引，较好地保证了数据的访问效率以及使用效率，在综合性能上表现出一定的鲁棒性和稳定性.

参考文献(References)

[1] Mayer-Schönberger V, Cukier K. Big data: A Revolution That Will Transform How We Live, Work, and Think[M]. Houghton Mifflin Harcourt, 2013.

[2] 孟小峰，慈祥. 大数据管理：概念、技术与挑战[J]. 计算机研究已发展，2013，50(1)：146-169.
Meng X F, Ci X. Big data management: Concepts, techniques and challenges [J]. Journal of Computer Research and Development, 2013, 50(1): 146-169.

[3] 石绥祥，雷波. 中国数字海洋：理论与实践[M]. 北京：海洋出版社，2011.

[4] Liu Xiansan, Zhang Xin, Chi Tianhe, et al. Study on China digital ocean prototype system[C]// Proceedings of the 2009 WRI World Congress. Piscataway, USA: IEEE Press, 2009: 466-469.

[5] 周项敏，王国仁. 基于关键位的高维空间划分策略[J]. 软件学报，2004，15(9)：1361-1374.
Zhou Xi M, Wang G R. Key dimension based high-dimensional data partition strategy [J]. Journal of Software, 2004, 15(9): 1361-1374.

[6] 黄冬梅，杜艳玲，贺琪. 混合云存储中海洋大数据迁移算法的研究[J]. 计算机研究与发展，2014，51(1)：199-205.
Huang D M, Du Y L, He Q. Migration algorithm for big marine data in hybrid cloud storage[J]. Journal of Integrative Plant Biology, 2014, 50(1): 199-205.

[7] Ren P, Liu W, Sun D, et al. Partition-based data cube storage and parallel queries for cloud computing[C]// Proceedings of the 9th International Conference on Natural Computation. Shengyang, China: IEEE Press, 2013: 1183-1187.

[8] Selvakumar C, Rathanam G J, Sumalatha M R. PDDS-Improving cloud data storage security using data partitioning technique[C]// Proceedings of the 3rd IEEE International Advance Computing Conference. Ghaziabad, India: IEEE Press, 2013: 7-11.

[9] 赵丹枫，金顺福，刘国华，等. DAS 模型下基于查询概率的密文索引技术[J]. 燕山大学学报，2008，32(6)：477-482.
Zhao D F, Jin S F, Liu G H, et al. A cryptograph index technology based on query probability in DAS model[J]. Journal of Yanshan University, 2008, 32(6): 477-482.

[10] 韩蕾，孙徐湛，吴志川，等. MapReduce 上基于抽样的数据划分最优化研究[J]. 计算机研究与发展，2013，50(S)：77-84.
Han L, Sun X Z, Wu Z C, et al. Optimization Study on sample based partition on MapReduce[J]. Journal of Computer Research and Development, 2013, 50(S): 77-84.

[11] Fox A, Eichelberger C, Hughes J, et al. Spatio-temporal indexing in non-relational distributed databases[C]// Proceeding of IEEE Conference on Big Data. Silicon Valley, IEEE Press, 2013: 291-299.

[12] Stantic B, Terry J, Topor R, et al. Indexing temporal data with virtual structure[C]// Proceedings of the 14th east European Conference on Advances in Databases and Information Systems. Springer, 2010: 591-594.

[13] Zhong Y Q, Fang J Y, Zhao X F. VegaIndexer: A Distributed composite index scheme for big spatio-temporal sensor data on cloud[C]// Proceedings of the IEEE Conference on Geoscience and Remote Sensing Symposium. Melbourne, Australia: IEEE Press, 2013: 1713-1716.

[14] Chen S, Ooi B C, Tan K L, et al. ST2B-tree: A self-tunable spatio-temporal b +-tree index for moving objects [C]// Proceeding of Conference on ACM Special Interest Group Conference on Management Of Data. Vancouver, Canada: IEEE Press, 2008: 29-42.

[15] Goil S, Nagesh H, Choudhary A. MAFIA: Efficient and scalable subspace clustering for very large data sets [C]// Proceedings of SIGKDD on Data Mining. San Diego: IEEE Press, 1999: 443-452.

[16] Kaufmann M, Amiri A, Vagenas P, et al., Timeline index: A unified data structure for processing queries on temporal data [C]// Proceedings of the ACM SIGMODInternational Conference on Management of Data. New York, USA: ACM Press, 2013:1173-1184.

[17] Chen S, Mario A, et al: ST2B-tree: a self-tunable spatio-temporal b +-tree index for moving objects. SIGMOD 2008:29-42.