(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 109815303 B
(45)授权公告日 2020.10.13

(21)申请号 201811654923.3
(22)申请日 2018.12.29
(65)同一申请的已公布的文献号
申请公布号 CN 109815303 A
(43)申请公布日 2019.05.28
(73)专利权人 哈尔滨工业大学(深圳)
地址 518055 广东省深圳市南山区桃源街道深圳大学城哈尔滨工业大学校区
(72)发明人 廖清 丁烨 漆舒汉 蒋琳 王轩
(74)专利代理机构 广州三环专利商标代理有限公司 44202
代理人 麦小婵 郝传鑫
(51)Int.Cl.
G06F 16/29(2019.01)

审查员 程小梅

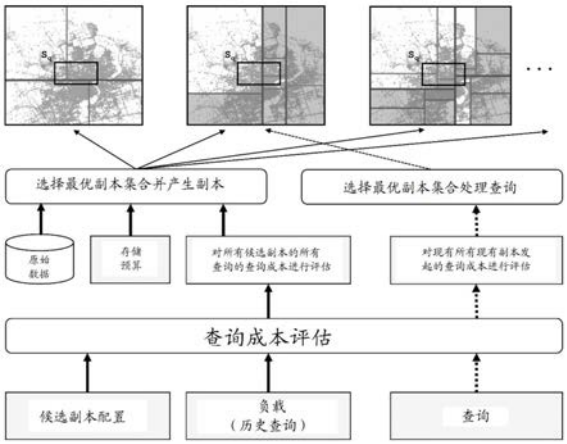
权利要求书2页 说明书13页 附图2页

(54)发明名称

一种基于位置的移动数据存储系统

(57)摘要

本申请涉及一种基于位置的移动数据存储系统及其优化方法,在预设的负载和存储预算的情况下,根据查询范围及存储系统中的原始数据,生成用于查询的多个候选副本,并使用查询成本评估模块对上述多个副本构成的副本集合进行评估,并从中选择出一个成本最低或接近最低的副本集合。该系统用于实现对基于位置的移动大数据分布式存储系统的宽查询范围性能的优化,并在此基础上提出了贪婪算法及线性规划舍入算法进一步对存储系统的性能进行进一步的发掘。



CN 109815303 B

1.一种基于位置的移动数据存储系统,其特征在于,所述存储系统包括:副本产生模块,查询成本评估模块,副本选择模块;

所述副本产生模块,根据查询范围及存储系统中的原始数据,生成用于查询的多个候选副本;

所述查询成本评估模块,在预设的负载、存储预算和输入查询范围情况下,对所述候选副本构成的所有副本集合的查询成本进行评估;

所述副本选择模块,在预设的负载和存储预算的情况下,设置一个空的副本集合R*;

在存储预算被用完,或向副本集合R*中添加单个副本但负载成本ρ(W,R*)不再降低之前,遍历所述副本产生模块产生的所有副本,并使用下式对单个副本r进行评分,将该次遍历中评分最高的副本增加到副本集合R*中:

$$\frac{\rho(W,R^*)-\rho(W,R^*\cup\{r\})}{\eta(r)};$$

当副本集合R*不在再增加副本时,R*内所有的副本即为副本产生模块需要产生的副本集合;

其中,ρ(W,R*)为查询集合W对副本集合R*查询时所需的成本,ρ(W,R*∪{r})为查询集合W对向副本集合R*增加副本r查询时所需的成本,η(r)为副本r的存储空间。

2.如权利要求1所述的存储系统,其特征在于:所述存储系统中的记录以下面的格式进行存储:(OID,TIME,LOC,A_1,……,A_m);

其中,OID为目标ID,TIME为时间戳,LOC是目标ID在某个时间点上所处的位置,A_1……A_m为根据数据库的应用场景变化的通用属性。

3.如权利要求2所述的存储系统,其特征在于:所述存储系统采用分布式的大数据存储构架。

4.如权利要求3所述的存储系统,其特征在于:所述副本的编码方案相同或不同。

5.如权利要求3或4所述的存储系统,其特征在于:所述存储系统可以为:TrajStore、PIST、CloSTridium、SpatialHadoop或SATO中的任意一种。

6.一种如权利要求1-5任一项所述的存储系统的优化方法,其特征在于:

所述副本选择模块,通过以下步骤选择最优或接近最优的副本集合:

1)设置一个空的副本集合R*;

2)在存储预算被用完,或向副本集合R*中添加单个副本但负载成本ρ(W,R*)不再降低之前,遍历所述副本产生模块产生的所有副本,并使用下式对单个副本r进行评分,将该次遍历中评分最高的副本增加到副本集合R*中:

$$\frac{\rho(W,R^*)-\rho(W,R^*\cup\{r\})}{\eta(r)};$$

3)当副本集合R*不再增加副本时,R*内所有的副本即为副本产生模块需要产生的副本集合;

其中,ρ(W,R*)为查询集合W对副本集合R*查询时所需的成本,ρ(W,R*∪{r})为查询集合W对向副本集合R*增加副本r查询时所需的成本,η(r)为副本r的存储空间。

7.一种如权利要求6所述的优化方法,其特征在于:

所述副本选择模块，还可以通过以下步骤选择最优或接近最优的副本集合：

10)使用下方程组作为副本选择问题的整数线性规划方程组：

$$\sum_{i=1}^{n}\sum_{j=1}^{m} w_i \cdot c_{ij} \cdot y_{ij},$$

上式的约束为：

$$\sum_{j=1} \eta(r_j) \le b,$$

$$\sum_{j=1}^{m} y_{ij} = 1, \forall i \in \{1,2,\cdots\cdots,n\},$$

$$\sum_{i=1}^{n} y_{ij} \le n \cdot x_j, \forall j \in \{1,2,\cdots\cdots,m\},$$

其中，w_i为查询q_i特定负载情况下的权重值，x_j为表示副本r_j是否在最优副本集合中的副本中的0-1变量，$c_{ij}=\rho(q_i,r_j)$，是指查询q_i在副本r_j上的查询成本，y_{ij}为表示查询q_i是否需要在副本r_j上进行操作的0-1变量，n为查询集合W中的查询的个数，m为副本集合R中副本r的个数，b为存储系统的存储预算；

11)将约束放松为$x_j \leqslant 1$和$y_{ij} \geqslant 0$，并使用MIP求解器求解放松约束后的整数线性规划方程；

12)舍入步骤11)求解出线性规划方程的部分解，将其组合成为一个完整的解。

8.一种如权利要求6或7所述的优化方法，其特征在于：根据查询输入的空间-时间范围对查询进行分组，副本生成模块根据上述分组生成多个候选副本。

9.一种如权利要求6或7所述的优化方法，其特征在于：在所有生成的副本中，若第一副本的存储空间大于第二副本，且在应答存储系统的输入查询时，第一副本的成本低于第二副本的成本，则将第二副本从候选副本中删除。

一种基于位置的移动数据存储系统

技术领域

[0001] 本发明涉及移动通信领域,具体涉及了一种基于位置的移动数据存储系统。

背景技术

[0002] 随着数据采集能力的发展,通过数以亿计的电子设备,例如手机、平板电脑、车载GPS导航及多种类型的传感器,采集人或物的巨量位置的移动数据变得更加简单。但是,存储这些位置的移动数据带来了两个挑战:1)怎么有效的处理大量的地理位置移动数据的查询;2)如何降低存储服务的成本。

[0003] 位置移动数据通常具有三个共同点:

[0004] 1.所有的数据都具有三个核心属性:目标ID、时间戳和位置信息;

[0005] 2.这些数据库的查询通常在通过特定的空间范围和时间范围信息进行查询;

[0006] 3.在处理空间-时间范围查询时,特别当查询到的结果特别多时即宽查询范围时,主流的大数据存储和管理系统不适合存储和处理这些位置移动数据。因为,这些系统不能根据时间和空间的临近性物理地聚类记录,这会导致主流系统触发非常多的低效率的随机读取。

[0007] 为了解决随机读取多的问题,TrajStore和PIST通过根据时间-空间代理来共同定位数据,并使用相对较大的分区,但TrajStore和PIST不是采用分布式构架的系统,都不能扩展为TB量级;CloST和SpatialHadoop是两个基于Hadoop的系统,旨在提供可扩展的分布式存储和并行查询处理的大型基于位置的移动数据系统;SATO是一个空间数据分区框架,可以快速分析和分区空间数据,并能够提供一个大小可变的查询处理的最有空间分区策略。

[0008] BLOT系统,是一种系统级的抽象,表示那些用于存储基于位置的大型移动数据的专用存储系统。上述的TrajStore、PIST、CloSTridium、SpatialHadoop和SATO都可以视为BLOT系统的具体例子。附图1展示了BLOT系统中的管理数据和查询数据的过程。

[0009] 在BLOT系统中存储了大量的基于位置的移动数据,每条记录都以下面的格式进行存储:(OID,TIME,LOC,A_1,……,A_m)。其中:OID为目标ID,TIME为时间戳,LOC是目标ID在某个时间点上所处的位置,A_1……A_m是根据数据库应用场景变化的通用属性。我们将前三个属性成为核心属性,其他属性称为公共属性。

[0010] BLOT系统通过核心属性将大数据集分割成相对较小的分区。例如,在TrajStore和CloST中,记录先按照位置(LOC)进行分区,之后再按照时间进行分区。同一分区中的记录一起被存储在一个存储单元中,该存储单元为顺序读取进行了优化。通常,存储单元大于一个磁盘页的,从数百KB到及兆字节。在BLOT系统中,记录通常是按顺序访问的,因此可以有效的处理较宽的时间-空间的范围查询;存储单元的数量走狗小,这样我们可以方便的维护分区索引。

[0011] 数据分区可以存储任何格式的记录,通常将每个分区都存储为CSV文件,每行指定一个记录。该格式虽然易于处理,但存储利用率很低。对于大型数据集,尤其是使用云存储

时，使用上述方法的成本较高。为了减小存储的大小，BLOT系统使用各种压缩技术对分区中的记录进行编码：1)使用二进制格式代替文本格式；2)应用通用的压缩算法对整个分区进行压缩；2)以列方式管理数据，然后应用列向编码方案(如增量编码和运行长度编码)。

[0012] 在BLOT系统中进行范围查询时，需要首选搜索相关的分区，即其范围与查询范围相交的分区；接下来读取并解压所有涉及的分区并提取所有的记录；最后，检查提取的记录并输出查询范围内的记录。需要注意的是，可以通过使用同时扫描多个分区的手段，实现并行查询处理。

[0013] 一般来讲，查询相关分区的成本有两部分组成：1)扫描成本，包含提取和过滤过程；2)额外成本，包含程序初始化、定位分析、加载解码器及清理过程等。在一个典型的BLOT系统中，扫描成本通常与分区中的记录的多少成正比，而额外成本在编码方案确定之后为一个常数。因此，对于特定的查询，查询成本由要扫描的记录的总数和设计的分区的总数决定。

[0014] 下面举例说明扫描成本在BLOT系统中使用不同策略的情况下的区别。在使用了多副本的BLOT系统中，当查询数据涉及3个分区的情况下，使用附图2上所示的不同策略，得到了不同的成本。可以得出，中间的情形2的查询成本最低，因为扫描成本和额外成本都是最低的，但是无法比较出左边的情形比右边的情形的成本是高还是低。

[0015] 从附图2，我们也可以得出，查询的成本会随则不同的分区方案发生很大的变化。现有的大多数BLOT系统可以在查询历史的基础上自适应地优化物理存储的管理配置，例如空间分区和时间分区的大小。但是，在查询的范围发生较大变化的情况下，其总体的查询性能在所述的优化配置情况下仍然不能让人满意。使用不同的物理层面实现的多副本可以缓解上述问题，但是这是典型的以存储空间换取性能的优化方法。

[0016] 其查询效率低的最根本的原因是，上述BLOT系统只使用一套配置参数去管理(例如分区和压缩)数据。但很显然的是，单一的配置无法针对所有的查询进行优化。

发明内容

[0017] 为了解决上述问题，本发明的目的是提供一种基于位置的移动数据存储系统及其优化方法。

[0018] 所述存储系统，其特征在于，所述存储系统包括：

[0019] 副本产生模块，查询成本评估模块，副本选择模块；

[0020] 所述副本产生模块，根据查询范围及存储系统中的原始数据，生成用于查询的多个候选副本；

[0021] 所述查询成本评估模块，在预设的负载、存储预算和输入查询范围情况下，对所述候选副本构成的所有副本集合的查询成本进行评估；

[0022] 所述副本选择模块，在预设的负载和存储预算的情况下，从所有的副本集合中选择出一个成本最低或接近最低的副本集合。

[0023] 优选的，所述存储系统中的记录以下面的格式进行存储：(OID，TIME，LOC，A_1，……，A_m)；

[0024] 其中，OID为目标ID，TIME为时间戳，LOC是目标ID在某个时间点上所处的位置，A_1……A_m为根据数据库的应用场景变化的通用属性。

[0025] 优选的，所述副本产生模块仅以查询输入范围作为副本集合产生的依据。

[0026] 所述优化方法，其特征在于：

[0027] 所述副本选择模块，通过以下步骤选择副本或副本集合：

[0028] 1)设置一个空的副本集合R*；

[0029] 2)在存储预算被用完，或向副本集合R*中添加一个副本但负载成本ρ(W,R*)不再降低之前，遍历所述副本产生模块产生的所有副本r，并使用下式对副本r进行评分，将该次遍历中评分最高的副本增加到副本集合R*中：

[0030] $$\frac{\rho(W,R^*)-\rho(W,R^*\cup\{r\})}{\eta(r)};$$

[0031] 3)当副本R*不在增加时，R*内所有的副本即为副本产生模块需要产生的副本。

[0032] 其中，ρ(W,R*)为查询集合W对副本集合R*查询时所需的成本，ρ(W,R*∪{r})为查询集合W对向副本集合R*增加副本r查询时所需的成本，η(r)为副本r的存储空间。

[0033] 优选的，所述的优化方法，其特征在于：

[0034] 所述副本选择模块，还可以通过以下步骤选择需要生成的副本：

[0035] 10)使用下方程组作为副本选择问题的整数线性规划方程组：

[0036] $$\sum_{i=1}^{n}\sum_{j=1}^{m} w_i \cdot c_{ij} \cdot y_{ij},$$

[0037] 上式的约束为：

[0038] $$\sum_{j=1}\eta(r_j)\le b,$$

[0039] $$\sum_{j=1}^{m} y_{ij}=1,\forall i\in\{1,2,\cdots\cdots,n\},$$

[0040] $$\sum_{i=1}^{n} y_{ij}\le n\cdot x_j,\forall j\in\{1,2,\cdots\cdots,m\},$$

[0041] 其中，w_i为查询q_i特定负载情况下的权重值，x_j为表示副本r_j是否在最优副本集合中的副本中的0-1变量，$c_{ij}=\rho(q_i,r_j)$，是指查询q_i在副本r_j上的查询成本，y_{ij}为表示查询q_i是否需要在副本r_j上进行操作的0-1变量，n为查询集合W中的查询的个数，m为副本集合R中副本r的个数，b为存储系统的存储预算；

[0042] 11)将约束放松为$x_j\le 1$和$y_{ij}\ge 0$，并使用MIP求解器求解放松约束后的整数线性规划方程；

[0043] 12)舍入步骤11)求解出线性规划方程的部分解，将其组合成为一个完整的解。

[0044] 在大数据存储系统中，为了解决容错问题会使用数据副本。当数据发生容错的情况下，可以使用多副本来替换特定的副本，从更好的利用存储空间。在此基础上，本发明提出了利用这些多副本来优化查询性能。因此，本发明在提升查询性能的情况下并不会增加存储成本。

[0045] 所以，本申请提出的一种基于位置的移动数据存储系统极其优化方法，不仅具有大数据存储系统使用多副本就拥有的提升数据有效性和耐久性的特点，还能进一步实现宽查询范围的性能的提升。并且使用多副本的效果是双重的：首先，数据可以使用不同的分区和压缩方法，这样，不同的查询可以选择最合适的配置实现最少的处理时间；其次，由于数

据拥有相同的逻辑视图，多副本能够在错误发生的时候彼此恢复。由于利用的是已经存在的副本内容，所以查询性能并不需要使用更多的存储空间。

附图说明

[0046] 图1是BLOT系统的概览图；

[0047] 图2是使用多副本的BLOT系统；

[0048] 图3是查询在BLOT系统中的分布示意；

[0049] 图4是计算$\varepsilon_S(q,r)$的不同情况。

具体实施方式

[0050] 下面结合附图和实施例，对本发明的具体实施方式作进一步详细描述。以下实施例用于说明本发明，但不用来限制本发明的范围。

[0051] 为了详细说明副本选择模块如何选择一组最优的不同副本集合，我们先定义一个基于位置的移动数据集D，本发明希望能够选择出一组不同的副本，这些副本符合存储约束，并针对给定的工作负载优化增提性能。

[0052] 首先给出一些定义，并以数学公式的方式给出他们之间的关系。

[0053] 定义1：分区方案。以U标记所述移动数据集D的时间-空间形成的矩形的边界，时间-空间分区方案$P=\{p_1,p_2,\cdots\cdots,p_n\}$是U的一个时间-空间分区。其中：

[0054] $$\bigcap_{s_i \in S} s_i = \mathrm{A}$$

[0055] $$p_i \cap p_j = \varnothing,\ \forall i,\ j \in \{1,2,\ \cdots\cdots,\ n\},\ i \neq j$$

[0056] 其中，p_i是U中第i个时间-空间分区。

[0057] 定义2：数据分区。对于一个给定的分区方案P，其中的任何分区pi∈P，其对应的数据分区di是pi中的所有的包含时间-空间数据的记录。另外：

[0058] $D(p_i)=d_i$

[0059] $P(d_i)=p_i$

[0060] $D(P)=\{d_i \mid P(d_i) \in P$

[0061] 根据分区方案的定义，我们可以得出：

[0062] $$\bigcup_i d_i = D$$

[0063] $$d_i \cap d_j = \varnothing,\ \forall i, j \in \{1,2,?\ \cdots, n\},\ i \neq j$$

[0064] 在本领域中，经常使用术语分区来指时间-空间分区(例如，p_i)和数据分区(例如，d_i)。另外，使用μ(p)和μ(d)分别标记空间分区p的时间-空间范围和数据分区d的时间-空间范围。

[0065] 定义3：编码方案

[0066] 在一个给定的数据分区d中，编码方案E是产生物理存储数据d的方法。

[0067] 定义4：副本和副本集合

[0068] 副本$r=\{D,P,E\}$表示，在D的所有记录中，使用了P分区方案进行了分区且使用了E编码方案的物理层次的数据。副本集合是多副本(即副本之间是独特的、唯一的)的集合，用

R＝{r_1,r_2,……,r_m}表示。P(r)和E(r)表示副本r使用了分区方案p和编码方案e。需要注意的是,上述定义需要所副本r中的所有数据分区都使用了相同的编码方案。但是在BLOT系统中,每一个分区使用不同的编码方案的理论分析与所有数据分区都使用相同的编码方案的类似,很容易得出。

[0069] 定义5:存储大小

[0070] 副本r的存储大小标记为η(r),为存储副本r中所有已编码分区的的存储空间的大小。数据集R的存储大小标记为η(R),是所有R中副本的存储大小的总和。$\eta(R)=\sum_{r_j\in R}\eta(r_j)$

[0071] 定义6:查询和负载

[0072] 范围查询q是将D所构成的大小标记为{δ_x,δ_y,δ_t},中心标记为{x,y,t}提取记录的过程。负载W＝{(q_1,w_1),(q_2,w_2),……(q_n,w_n)}是一组每一个查询都带有加权信息的独特查询的集合。

[0073] 类似于μ(p)和μ(d),使用μ(q)表示q的时间-空间范围。负载的查询加权可以解释为查询重要性(例如频率,有限程度等)。在一些情况下,加权可以被归一化为:$\bigcup_i d_i = D$。

[0074] 另外,Q(W)用来标记W中的所有查询的集合,Q(W)＝{q_1,q_2,……,q_n}。

[0075] 定义7:查询成本和负载成本

[0076] 对于一个给定的副本r∈R和q∈Q(W),对于r中q的查询成本标记为ρ(q,r)。可以得出:

[0077] $$\rho(q,R)=\min_{r_j\in R}\rho(q,r_j)$$

[0078] 而且

[0079] $$\rho(W,R)=\sum_{(q_i,r_j)\in R} w_i\bullet\rho(q_i,R)$$

[0080] 基于上述的内容,我们可以判断如何找到多副本中的最优副本集合。

[0081] 定义8:副本选择问题

[0082] 对于给定的数据集合D、负载W＝{(q_1,w_1),(q_2,w_2),……,(q_n,w_n)},候选副本集合中的一个集合R＝{r_1,r_2,……,r_m},存储预算b,如何找到一个副本集合R*并满足以下条件:

[0083] R∈R*,

[0084] η(R*)≤b,

[0085] 对于所有的$R`\subseteq R$,ρ(W,R*)≤ρ(W,R`)以致η(R`)≤b,

[0086] 对于绝大多数情况,R包含所有可能的副本,例如,如果系统中存在m_P个分区方案和m_E个编码方案的情况下,m＝m_P*m_E。

[0087] 为了找到最优的副本集合R*,首选需要了解q_i∈Q(W)及r_j∈R中所有的查询成本ρ(q_i,r_j)和存储大小η(r_j)。对于η(r_j),由于压缩率在几乎所有的情况下都是稳定的,可以选择使用对应的编码策略E(r_j)的压缩率来评估。对于ρ(q_i,r_j),本发明还提出了一个非常精准的成本模型,在不需要生成实际副本的情况下评估查询成本。

[0088] 副本选择模块的实现

[0089] 首先,根据上述内容,可以通过一系列的计算来证明副本选择问题是一个非确定性多项式难题(NP-Hard)。

[0090] 可以通过使用最小加权集覆盖问题来简化副本选择问题来证明副本选择问题是NP-Hard。尤其是，当给定一个n个元素的集合A＝{a_1,a_2,……,a_n}，并且一个m元素的集合S＝{s_1,s_2,……,s_n}。

[0091] 这里：

[0092] $s_i \subseteq A, \forall s_i \in S$

[0093] 并且

[0094] $\bigcap_{s_i \in S} s_i = \mathrm{A}$

[0095] 最小加权集覆盖问题是找到一个集合S*属于S并且满足：

[0096] |S*|≤|S|

[0097] 并且

[0098] $\sum_{i=1}^{n} y_{ij} \leq n \bullet x_j, \forall j \in \{1,2,\cdots\cdots, \mathrm{m}\}$

[0099] 此时S*的成本是最低的，此时S*的成本可以表示为：

[0100] $\sum_{i=1}^{|S^*|} c_i$

[0101] 其中c_i是集合s_i的成本(加权)。而最小加权集覆盖问题是一个公知的NP-Hard问题。

[0102] 对于A，可以构造一个负载函数W＝{(q_1,1),(q_2,1),……,(q_n,1)}，其中，所有的加权值都被设置为了1。对于S，可以构造一个候选副本集合R＝{r_1,r_2,……,r_m}，其中，η(r_j)∈R都被设置为了0。这样，可以得到查询的成本为：

[0103] 如果ρ(q_i,r_j)＝0，则ρ(q_i,r_j)＝ρ(q_i,R)

[0104] 如果ρ(q_i,r_j)≠ρ(q_i,R)，则ρ(q_i,r_j)＝+∞

[0105] 根据查询成本和负载成本的定义，ρ(q_i,r_j)＝0是r_j的应答q_i的最低查询成本，而ρ(q_i,r_j)＝+∞是r_j的应答q_i的最高查询成本。为了方便起见，我们分别使用α和β分别标记最小加权集合覆盖问题和对应的副本选择问题。

[0106] 假设，在问题β中，最优的副本集合为R*，进而，可以构造出问题α中对应的集合S*＝{s_j|其中j为所有的r_j∈R*}。而问题α是否可解，需要讨论两种情形：

[0107] 第一中情形，如过在问题β中，ρ(W,R*)＝0，那么所有Q(W)中的查询都能立即被R*中的某个副本应答。根据由问题α构造问题β的过程，A中的任一元素都要被S*中的某些副本集合覆盖。在这种情况下，问题α可解是被保证的。

[0108] 第二中情况下，如果在问题β中，ρ(W,R*)＝+∞，可以通过矛盾的方式证明问题α无解。假设S**是问题α的解，那么能够构造一个问题β的副本集合R**＝{s_j|其中j为所有的s_j∈R*}，这种情况下，ρ(W,R**)＝0很容易被证明，这就意味着ρ(W,R**)＜ρ(W,R*)，但这与假设的前提R*是问题β的最优副本集合是矛盾的。

[0109] 因此，当且仅当对应问题β的最优负载成本是0的情况下，问题α才有解。基于此，副本选择问题是与最低加权集合覆盖问题难度等同的问题。

[0110] 上述结论证明了以多项式的形式找到最优副本集合是不可能的，但当输入相对较小的情况下的解依然是很有用的。

[0111] 本发明提供一个解决方案是将最初的问题建模成一个0-1混合整数规划(Mixed Integer Programming,MIP)问题,将问题交给MIP求解器处理。此处的挑战是,如何将问题妥善的建模从而实现0-1MIP问题的最优解同样也是最初问题的最优解。

[0112] 设置n=|W|,即n为查询集合W中的查询的个数;m=|R|,即m为副本集合R中副本r的个数;对于任何i∈{1,2,……,n}和任何j∈{1,2,……,m},设置x_j为表示副本r_j是否属于最优副本集合中的副本的0-1变量,y_{ij}表示查询q_i是否需要在副本r_j上进行操作的0-1变量;b为存储空间的大小的约束:

[0113] 使用特定的副本来处理每一个查询:

[0114] 所有被选中的副本都至少处理一个查询,且在R中出现。

[0115] $$\eta(R)=\sum_{j=1}\eta(r_j)\le b$$

[0116] $$\sum_{j=1}^{m} y_{ij}=1,\forall i\in\{1,2,\cdots\cdots,n\}$$

[0117] $$y_{ij}\le x_j,\forall i\in\{1,2,\cdots\cdots,n\},j\in\{1,2,\cdots\cdots,m\}$$

[0118] 上式可以分解为n×m个约束。MIP问题若存在较多约束的话,求解是极其困难的,所以倾向减少约束对其求解。所以,使用下述m个约束来的替换上述n×m个约束(虽然约束被轻微的放松了,但不不影响最优解的过程)。

[0119] 设置$c_{ij}=\rho(q_i,r_j)$,这样目标函数就可以表示为:

[0120] $$\sum_{i=1}^{n}\sum_{j=1}^{m} w_i\cdot c_{ij}\cdot y_{ij},$$

[0121] $$\sum_{i=1}^{n} y_{ij}\le n\cdot x_j,\forall j\in\{1,2,\cdots\cdots,\ m\},$$

[0122] 上述目标函数可以在下面的约束下直接使用MIP求解器求解:

[0123] $$\sum_{j=1}\eta(r_j)\le b,$$

[0124] $$\sum_{j=1}^{m} y_{ij}=1,\forall i\in\{1,2,\cdots\cdots,n\},$$

[0125] $$\sum_{i=1}^{n} y_{ij}\le n\bullet x_j,\forall j\in\{1,2,\cdots\cdots,m\}。$$

[0126] 存储系统的优化方法

[0127] 首先,对副本选择问题的负载进行优化,即降低问题的规模。

[0128] 通常,求解一个MIP问题的计算时间和问题的大小(例如,决策变量的个数)呈指数增长。在上述式子中,所有的决策变量的个数为m(n+1)在m和n都相对较小的情况下就会变得非常的巨大。例如,在使用20个分区方案、5个编码方案以及1000个查询的给定负载情况下,就会产生10^5个决策变量。虽然,这已经是在实际使用中的常规场景,但这已经使得上述MIP问题几乎不可解(在现有的计算能力下)。所以,需要将上述的方法更加可控,本发明提出了几个能够显著地降低问题大小的有效的解决方案。

[0129] A.降低负载大小

[0130] 如果直接根据记录在查询日志中记录的历史查询生成输入负载,若在系统中经常

性的发布新的查询，m增加地会非常快。为了解决该问题，将每一个q∈Q(W)当作一个类似查询的群组。特别是，使用Q^G标识一组时间-空间范围相同的所有查询。进一步地，将查询和负载中的定义$\mu(q)=\langle x,y,\delta_x,\delta_y,\delta_t\rangle$替换为$\mu(Q^G)=\langle \delta_x,\delta_y,\delta_t\rangle$，即仅以查询输入范围作为副本集合产生的依据。

[0131] 这是基于在真实情况下在同样范围大小的查询经常多次发生的观察的上得出的。例如，用户通常使用相等大小的网格来分解空间，然后对每个网格单元进行简单的统计。特别需要指出的是，对一组查询进行成本评估比对一个查询的成本评估要复杂的多。如果不同范围大小的数量仍旧很大，还可以使用簇算法，例如K-方法将范围大小成簇，仅适用簇的中心来构成输入负载，以此通过对簇数量的控制实现对m值的限制。

[0132] B.降低候选副本的数量

[0133] 如果存在两个副本，$r_1,r_2\in R$，并满足：

[0134] $\eta(r_1)\leqslant\eta(r_2)$

[0135] $$\rho(q_i,r_1)\leq\rho(q_i,r_2),\forall q_i\in Q(W)$$

[0136] 这种情况称为副本r_1主导副本r_2，明显的使用$R\backslash\{r_2\}$替换R作为输入候选副本，并不会改变负载成本ρ(W,R*)。因此，从R中删除r2是安全的。

[0137] 上述内容可以总结为：在所有生成的副本中，若第一副本的存储空间大于第二副本，且在应答存储系统的输入查询时，第一副本的成本低于第二副本的成本，则将第二副本从候选副本中删除。

[0138] 更常见的是，一个副本集合主导一个副本。进而，对于给定的副本$r\in R$，副本集合，如果：

[0139] 可以得出，副本集合R^D主导副本r。

[0140] 虽然理想中，可以找到最小的主导副本集合$R^D\subseteq R$，这样R^D

[0141] $r\notin R^D$，

[0142] $\eta(R)\leqslant\eta(r)$，

[0143] $$\rho(q_i,R^D)\leq\rho(q_i,r),\forall q_i\in Q(W)$$。

[0144] 上述内容可以总结为：在所有生成的副本集合中，若第一副本集合的存储空间大于第二副本集合，且在应答存储系统的输入查询时，第一副本集合的成本低于第二副本集合的成本，则将第二副本集合从候选副本集合中删除。

[0145] 但副本选择问题本身是一个NP-Hard问题，再实际应用当中无法找到最小的R^D。因此，使用一个较为粗犷但有效的启发式算法来找到次优的主导副本集合。

[0146] 因此，本发明提出了几个基于降低问题规模实现选择近于最佳的副本集合的逼近算法。逼近算法适用于那些在降低问题大小之后候选副本的数量依旧很多或负载快速变化造成副本集合需要频繁进行选择的场景。

[0147] 方法(A)：贪婪算法；

[0148] 该算本发明提出的一个用以求解副本选择问题的快速贪婪算法，该算法继承并发展自最小加权集合覆盖算法。需要每次都在副本集合R*中加入一个副本，在存储预算被耗尽或者负载成本ρ(W,R*)不能通过增加剩余的副本而降低的情况下，每增加一个副本都能够使下式最大化。

[0149] $$\frac{\rho(W,R^*)-\rho(W,R^*\cup\{r\})}{\eta(r)}$$

[0150] 在存储空间被沾满之前，每次增加一个副本到副本集合当中，最坏情况需要迭代|R|次。在每次迭代中：

[0151] 为所有还未被加入到R*的候选副本|R\R*|评分；

[0152] 将评分最高的副本增加到R*中。

[0153] 上述步骤可以表示为：

[0154] 1)设置一个空的副本集合R*；

[0155] 2)在存储预算被用完，或向副本集合R*中添加一个副本但负载成本ρ(W,R*)不再降低之前；

[0156] 遍历所述副本产生模块产生的所有副本r，并使用下式对副本r进行评分，将该次遍历中评分最高的副本增加到副本集合R*中：

[0157] $$\frac{\rho(W,R^*)-\rho(W,R^*\cup\{r\})}{\eta(r)}$$；

[0158] 3)当副本R*不在增加时，R*内所有的副本即为副本产生模块需要产生的副本。

[0159] 评分步骤：计算每一个可能会加入到R*的候选副本的收益，这样，在该步骤中所有的$q\in Q(W)$都与当前副本及候选部分的成本进行对比。因此，该步骤需要执行O(|R||Q(W)|)次，产生一个O(log n)的近似比率，这里n是所有查询集合$q\in Q(W)$的大小。该贪婪算法的运行时间是$O(nm^2)$，此处m是候选副本集合的大小。

[0160] 方法(B)：线性规划舍入策略(Linear Programming Rounding Strategy，LP Rounding Strategy)：

[0161] 虽然贪婪算法易于实现，而且能够实际实现不错的效果，但是贪婪算法最高的预期是近似对数比率。当查询的数量继续增加的话，其性能也会相应的降低。因此，本发明还提出了基于线性规划舍入的常因子近似算法。该线性规划舍入策略包含三个步骤：

[0162] 10)，生成作为副本选择问题的整数线性规划方程组；

[0163] $$\sum_{i=1}^{n}\sum_{j=1}^{m}w_i\cdot c_{ij}\cdot y_{ij}$$，

[0164] 上式的约束为：

[0165] $$\sum_{j=1}^{}\eta(r_j)\le b$$，

[0166] $$\sum_{j=1}^{m}y_{ij}=1,\forall i\in\{1,2,\cdots\cdots,n\}$$，

[0167] $$\sum_{i=1}^{n}y_{ij}\le n\cdot x_j,\forall j\in\{1,2,\cdots\cdots,m\}$$，

[0168] 其中，w_i为查询q_i特定负载情况下的权重值，x_j为表示副本r_j是否属于最优副本集合中的副本的0-1变量，$c_{ij}=\rho(q_i,r_j)$，是指查询q_i在副本r_j上的查询成本，y_{ij}为表示查询q_i是否需要在副本r_j上进行操作的0-1变量，n为查询集合W中的查询的个数，m为副本集合R中副本r的个数，b为存储系统的存储预算；

[0169] 步骤11),将约束放松为$x_j \leqslant 1$和$y_{ij} \geqslant 0$,并使用MIP求解器求解放松约束后的整数线性规划方程组;

[0170] 步骤12),舍入步骤11)求解出线性规划方程的部分解,将其组合成为一个完整的解。

[0171] 上述步骤是基于前面提到的MIP问题的基础上提出的。在前面的介绍中,已经对可以将副本集合的选择问题简化为MIP问题,并对其进行了详细的介绍,这里就不再赘述。

[0172] 步骤11)中,使用$x_j \leqslant 1$和$y_{ij} \geqslant 0$进一步放松MIP。这样就能够以求解线性规划得到分数的x_j和y_{ij}。

[0173] 步骤12)中,由于常规的舍入技巧无法直接应用于副本选择问题,本发明提出了下面的舍入策略。

[0174] 假设在LP的阶段11)中存在一个最优解,对于任何$q_i \in Q(W)$,定义q_i的邻近范围为:$N_i = \{r_j \in R^* | y_{ij} > 0\}$。所有部分服务于$q_i$的副本$r_j$都是$q_i$的邻近范围。定义簇是以$q_i \in Q(W)$为中心的查询和副本的集合。在LP中,将其标记为$c_i = \sum_{r_j \in R^*} c_{ij} \cdot y_{ij}$。由此可以得到总的查询成本为:$\rho(W,R^*) = \sum_{(q_i,w_i) \in W} w_i \cdot C_i$。

[0175] 使用C_i的升序作为查询q_i的排序依据,在所拥有的查询和副本都被分配到同一个簇中之前,不断地将每一个查询和副本分配到不同的簇中。在每一次迭代中,选择具有最小C_i值的查询q_i。如果对于任何存在的簇中心$q_{i`} \in \Gamma$,$N_i \cap N_{i`} = \varnothing$,则设置一个新的簇i,并将q_i加入到新的簇中并将簇的中心标记为q_i。如果$N_i \cap N_{i`} \neq \varnothing$,则将$q_i$加入到簇i`中。之后,将部分解进行舍入:对每一个簇,对于N_i中选择簇中心为q_i的簇,选择成本最低的副本r_i,并将该簇中的查询都分配给副本r_i,即将得到的部分解组合成为一个完整的解。

[0176] 定理2会对LP舍入策略的接近率进行证明,且所述LP舍入策略为一个三要素逼近算法。

[0177] 假设所述MIP问题的最优解是Θ_0,放松了约束的LP问题的最优解是Θ_1,由于Θ_0可能是Θ_1的一部分,可知$\Theta_0 \leqslant \Theta_1$。在舍入策略下,副本选择模块选择出的副本集合的成本最大为$4\Theta_1$。

[0178] 假设$q`_i$是簇K的中心点,且;对于在簇K中的任意查询其被选择的副本为r_{j_k}。对于q_i,在副本r_{j_k}上付出的查询成本分为3类:

[0179] a)$q`_i$在簇K内,且$c_{ij_k} \leq c_{i`j_k}$;

[0180] b)$q`_i$在簇K内,但$\rho(W,R) = \sum_{i=1}^{n}\sum_{j=1}^{m} w_i \cdot c_{ij} \cdot y_{ij} \leq 3 \cdot \sum_{i=1}^{n} w_i \cdot C_i = 3 \cdot \Theta_1$,由于$N_i \cap N_{i`} = \varnothing$,q_i和$q`_i$具有一些相同的副本。根据三角形的特点,可以得出$c_{i`j} \leq c_{ij_k} + c_{ij} + c_{i`j_k} \leq C_i + 2C_{i`} \leq 3C_{i`}$。后面的不等式之所以成立,是因为$C_i$是按照升序排列的,每次选择的具有最有$C_i$值的查询;

[0181] c)q_i不在簇K以内,将c_{ij_k}设置为无穷大,在该簇内,q_i在任何副本中都查询不到内容。

[0182] 综上,舍入策略求解的成本的总和为:

[0183] $$\rho(W,R)=\sum_{i=1}^{n}\sum_{j=1}^{m}w_i\cdot c_{ij}\cdot y_{ij}\le 3\cdot\sum_{i=1}^{n}w_i\cdot C_i=3\cdot\Theta_1$$

[0184] 该求解成本最多为线型规划的成本的3倍。

[0185] 为实现存储系统的查询成本评估模块，本发明还提出了一种有效的模型，对副本选择问题中的查询成本进行评估。该过程使用查询某个副本的预期运行时间作为评估该询问的成本的标准。由于副本r的每一个分区都有空间范围S(p)和时间范围T(p)组成，下面将从时间方面和空间方面对查询成本的评估进行介绍。

[0186] 根据定义6，可以q当作一个矩形，使用μ(q)标记q的时间-空间范围，例如<x,y,t,δ_x,δ_y,δ_t>。为了清楚的表述，使用S(q)标识q的空间范围，其中S(q)＝<x,y,w,h>,<x,y>是该矩形的左上角，w(q)和h(q)分别是举行的宽度和高度。类似的，对于每一个属于P(r)的分区p，使用w(p)和h(p)来标识这个分区的宽和高。

[0187] 为了找到这些查询需要扫描的分区，考虑到查询是像图3在空间当中均匀分布的。在图3中，w(D)和h(D)分别是地图的宽度和高度。查询的范围使用黑色区域表示，查询范围的左上角的点只能在灰色区域中产生，因为超出地图空间范围的查询可以被看做另一个个更小空间范围的查询。左上角的位置同样在灰色区域当中均匀分布。

[0188] 对于给定的负载W，空间分区需要被扫描的概率等于分区的覆盖的查询数量除以负载区域内所有的查询。由于，查询是均匀分布的，被写的概率等于查询所覆盖的分区面积(图4中的粗线围成的矩形)除以所有查询所属的整个面积(图3中的斜线填充的区域)。

[0189] 假设分区p和地图边界之间的具体分别为：west(p)，east(p)，north(p)，和south(p)，那么可以将预期的空间分区定义为：

[0190] 定理3：预期的空间分区(即需要扫描的空间分区)。对于给定的查询q和具有分区p∈P(r)的副本，预期的空间分区的数量 $\varepsilon_S(q,r)=\sum_{p\in P(r)}\varepsilon_S(q,p)$

[0191] 其中：$$\varepsilon_s(q,p)=\frac{(w(q)+w(p)-w(\alpha))g(h(q)+h(p)-h(\alpha))}{(w(D)-w(q))g(h(D)-h(q))}$$

[0192] 其中α是查询的偏移量，并且

[0193] w(α)＝max(0,w(q)-west(p))+max(0,w(q)-east(p))，

[0194] h(α)＝max(0,h(q)-north(p))+max(0,h(q)-south(p))

[0195] 下面证明$\varepsilon_S(q,r)$表达式中的分母是无关紧要的，仅考虑被标记为S的分子(例如在附图4中查询重叠的分区的面积)，在附图4中，查询为被斜线填充的区域，分区被为竖线填充的区域。粗线围成的矩形区域则表示查询与分区重叠的区域。

[0196] 可以分为以下情况：

[0197] (a)在附图4(a)中，分区的范围比查询的范围小。通过观察，可以得出S＝(w(q)+w(p))·(h(p)+h(q))，且w(a)＝h(a)＝0.所以定理3成立。

[0198] 本申请提出了一种有效的评估模型，为解决副本选择问题提供依据。

[0199] (b)如果分区的面积比查询的大，如附图4(b)表示的那样。这与上述情形类似，定理3同样成立。

[0200] (c)分区在一个角落中，像附图4(c)所示的情形，超出了查询允许的范围，通过观察，可以得出S＝(w(q)+w(p)-w(a))·(h(p)+h(q)-h(a))。此时，定理3同样成立。

[0201] (d)如果分区靠近边界，如附图4(d)所示。通过观察，可以得出S＝(w(q)+w(p))·(h(q)+h(p)-h(q))＝(w(q)+w(p))·h(p)，由于

[0202] h(p)+h(q)-h(a)＝h(p)+h(q)-0-(h(q)-0)＝h(p)，所以定理3依然成立。

[0203] (e)分区与两个以上的边界相邻。这种情况是在基于空间分区策略当中不可能发生的情况，因为分区的数量≥4。

[0204] 综上所述，定理3成立。

[0205] 类似于上述的空间分区，时间分区被扫描的概率等于查询可能与分区重叠的范围除以所有查询覆盖的范围得到的商。

[0206] 假设分区p和所有记录T(D)的时间范围之间的间隔为top(p)和bot(p)，可以价格预期的时间分区定义为：

[0207] 定理4：预期的时间分区(即需要扫描的时间分区)。对于给定的查询和与有分区p∈P(r)的副本，预期查询应当扫描的时间分区的数量$\varepsilon_t(q,r)$为：

[0208] $$\varepsilon_T(q,r)=\sum_{p\in P(r)}\varepsilon_T(q,p)$$

[0209] 其中：

[0210] $$\varepsilon_t(q,r)=\frac{T(q)+T(p)-T(\alpha)}{T(D)-T(q)}$$

[0211] 其中α是查询的偏移量，并且：

[0212] T(α)＝max(0,T(q)-top(p))+max(0,T(q)-bot(p))

[0213] 定理4的证明过程与定理3的证明过程类似，这里就不再赘述。

[0214] 预期的查询成本

[0215] 为了应答副本r上的查询q，BLOT系统需要扫描(物理上保存了目标的满足$\mu(p)\cap\mu(q)\neq\varnothing$条件的所有属于P(r)的分区p，并使用μ(p)对所有的记录进行过滤。基于预期的需要扫描的空间和时间分区的数量，可以将他们组合成为一个给定查询q的所需分区的预期值：

[0216] $$\varepsilon(q,r)=\sum_{p\in P(r)}\varepsilon_s(q,p)\cdot\varepsilon_t(q,p)$$

[0217] 对于给定P(r)中的空间分区数目n_s和时间分区数目n_t。可以得到：

[0218] $$\rho(q,r)=\varepsilon(q,r)\cdot\left(\frac{\eta(r)}{n_s\cdot n_t\cdot\zeta(r)}+\xi(r)\right)$$

[0219] 其中，ζ(r)、ξ(r)都表示扫描速度(即单位时间内扫描的记录的数量)，在给定编码方案E(r)情况下，ζ(r)表示扫描过程完成之前的时间，而ξ(r)表示扫描过程完成能够之后的时间。例如，如果每个分区都连续的存储了在本地磁盘上的一个常规文件，ξ(r)就代表了寻址该文件文件头的寻址时间，ζ(r)代表了硬盘的传输速率(假设CPU总是等待I/O操作)。另外举例，如果在Amazon S3上存储了一个目标，查询在Amazon EMR(Elastic MapReduce)上执行，此时ξ(r)表示初始化映射任务的时间与开始扫描分区前的寻址S3目标时间的和。ζ(r)的值取决于编码方案E(r)。在真实的应用场景中，高压缩率通常会导致第扫描速度。

[0220] 假设所有的候选分区方案都只产生无偏斜的数据分区，换句话说，在所有$p_i\in P(r)$中的每个$D(p_i)$中的记录的数目是完全相同的。无偏斜的分区是分区并行处理(例如

MapReduce)的理想特性。使用这种分区方案的一个例子是使用K-D树来建立分区，在这些分区中每当数据被细分一次，其数据也被平均分配一次。

[0221] 根据上述的数学式，就可以计算出在O(|P(r)|)时间内在副本r上任何查询所需的成本。这样就可以计算所有的查询成本，其为：

[0222] $$O(|W|\bullet|R|\bullet\max_{r_j\in R}|P(r_j)|)$$

[0223] 以上所述仅是本发明的优选实施方式，应当指出，对于本技术领域的普通技术人员来说，在不脱离本发明技术原理的前提下，还可以做出若干改进和替换，这些改进和替换也应视为本发明的保护范围。

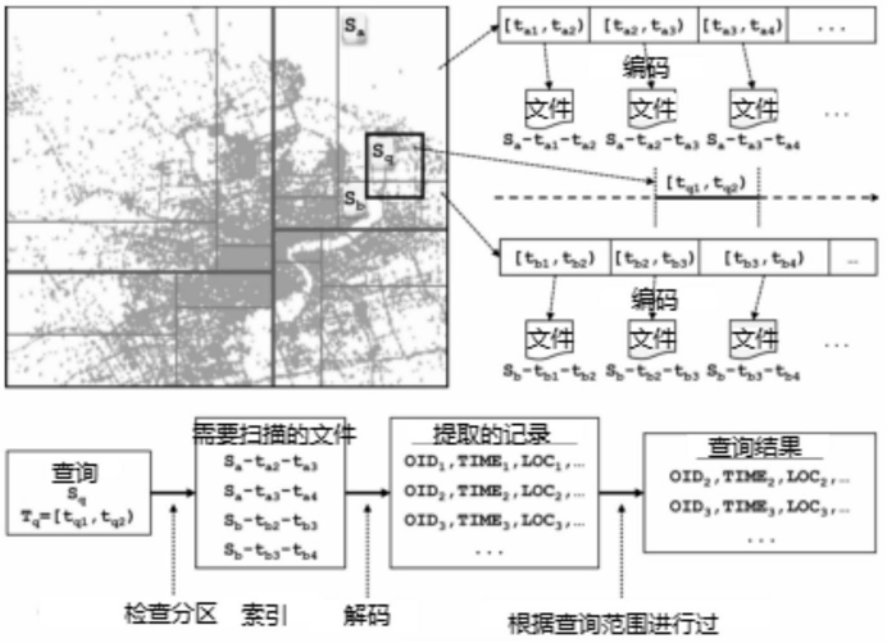

编码
文件
编码
文件
查询
需要扫描的文件
提取的记录
查询结果
检查分区
索引
解码
根据查询范围进行过

图1

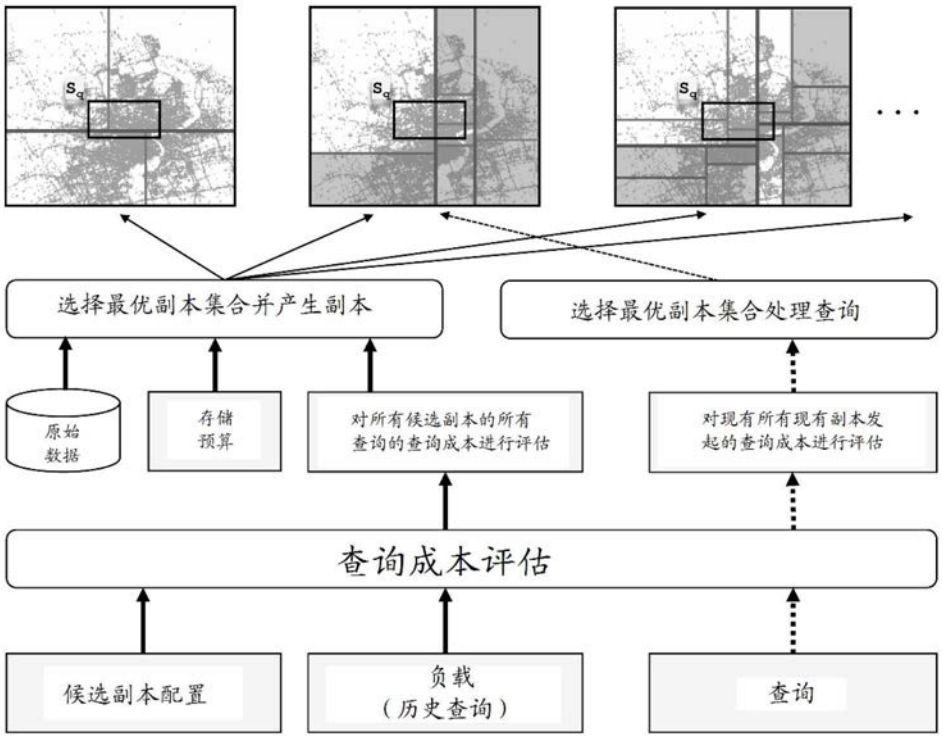

选择最优副本集合并产生副本
选择最优副本集合处理查询
原始数据
存储预算
对所有候选副本的所有查询的查询成本进行评估
对现有所有现有副本发起的查询成本进行评估
查询成本评估
候选副本配置
负载（历史查询）
查询

图2

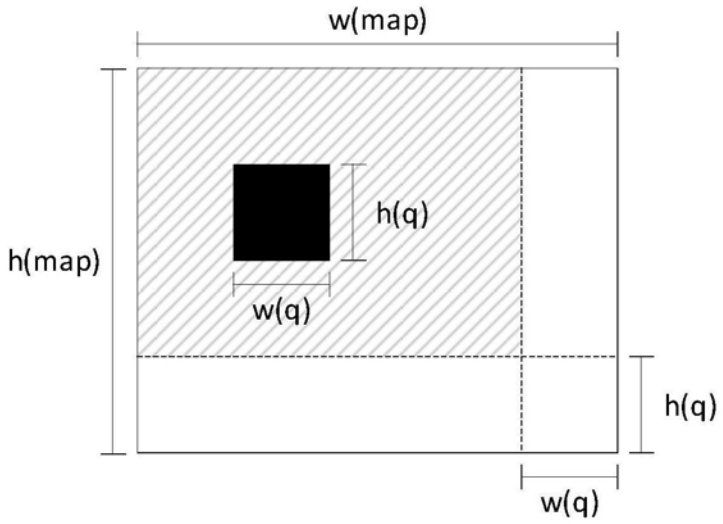


图3

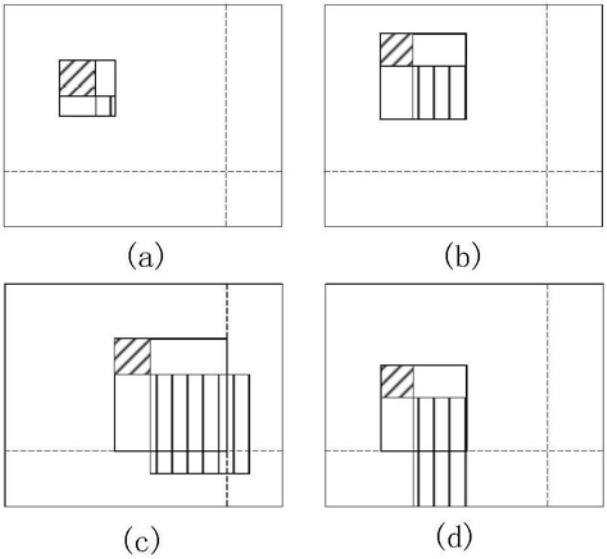


图4