能效查询处理与优化可行性初探①

吴 岩, 杨良怀, 潘 建

(浙江工业大学 计算机科学与技术学院, 杭州 310014)
(浙江省可视媒体智能处理技术研究重点实验室, 杭州 310014)

摘 要: 随着数据管理需求的不断增长, 降低与控制数据中心的能耗成为一个挑战性问题. DBMS 是数据中心核心软件, 能效查询处理与优化是其中一个重要议题. 本文提出了新型的能耗代价评估模型, 通过评估查询计划的时间和能耗代价, 考察了不同优化目标在不同硬件条件下对查询处理的影响. 实验表明, 传统硬件下面向性能的优化与面向能耗的优化结果是一致的; 在新硬件条件下, 两者结果则不同, 可以改进数据库系统能效.

关键词: 能耗; 代价模型; 查询优化; 查询处理

Towards Energy Efficient Query Optimization

WU Yan, YANG Liang-Huai, PAN Jian

(School of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310014, China)
(Key Laboratory of Visual Media Intelligent Process Technology of Zhejiang Province, Hangzhou 310014, China)

Abstract: With the growing demand for data management, it is a great challenge to reduce and throttle the energy consumption in data centers. DBMS is a vital software in data center to deal with volumes of data, thus, energy efficient query processing and optimization is one of the critical issues to be solved. We proposed a novel cost model to estimate energy consumption of a query plan, and we investigated the effects of different optimization goals on query optimization under different hardware configurations. Experiements demonstrated that the results of performance-oriented and energy-oriented query optimization are same under the traditional hardware configuration, while the results turned out to be different under the new hardware configuration, which indicates that it is practical to improve the energy efficiency of database systems.

Key words: energy consumption; cost model; query optimization; query processing

1 引言

传统数据库系统中硬盘存取代价占统治地位, 其查询优化面向性能, 没有考虑系统能耗问题. 随着数据管理需求的不断增长, 降低与控制数据中心的能耗成为重要问题; DBMS 是数据中心核心软件, 能效查询处理与优化成为其中一个重要研究议题. 随着存储技术的变革, 如新一代非易失存储级内存(SCM)[1]的出现, 改变了计算机的存储层次, 给系统能效带来了机会. SCM 的初级版固态盘(SSD)在读写性能是微秒级, 而相变内存(PCM)则 100ns 级的, 能耗更小; 预计到 2020 年, 新的存储将被大量使用. 因此, 在新的存储层次中进行面向能效查询优化是充满机遇的.

为解决 DBMS 的能效查询优化问题, 本文构建了新型的能量评估模型, 估算查询处理中各个操作符的能耗; 在计划选取中, 将能耗作为优化因子进行优化. 本文分别在不同存储层次下评估查询处理的时间和能耗, 考察 DBMS 中能效查询优化的可行性.

2 相关工作

早期对便携机硬盘节能进行了广泛研究[2][3], 近来扩展到台式机、服务器的存储子系统[4][5][6]. Tsirogiannis[7]系统地分析了数据库服务器的能耗, 发

① 基金项目:国家自然科学基金(61070042);浙江省自然科学基金(Y13F020110)
收稿时间:2013-09-09;收到修改稿时间:2013-10-04

现系统空闲功率达到了峰值功率的一半, 存在节能的机会.

随着 CPU 能效的提升, 其能耗在整个系统中所占比例下降[8], 伴随内存容量增大, 其能耗变得不可忽视. 内存价格的下降, 数据库使用内存容量不断增大, 内存能耗最多可以占到服务器的 46%[9], 内存将成为能耗最大的部件. 针对内存节能已开展了一些研究, Pisharath 等[10]通过对主存数据库的内存功耗状态动态调度, 最大可节省 40%的内存能耗; 采用热断电方式关闭内存可以节省高达 40%的系统耗能[11][12]. 内存在数据库能效优化中将发挥重要作用[11][12][13][14].

Lang 与 Patel[13]在能效数据库方面开展了一些实验, 其中实验 1 在 DBMS 中通过处理器电压/频率控制(PVC)使其进入低功耗状态(p-states)来平衡能耗与性能; 实验 2 是通过等待一定数量查询到达后进行批处理(进行显式延迟以改进查询能效), 利用共享公共子表达式来实现多查询优化, 藉此减少每个查询的平均能耗. 多查询并行处理有利于能效的提升, 在其所用工作集中可节能 45%. Lang 等[14]采用查询操作的 CPU 指令数、内存访问次数以及固态盘的页面访问数来估算能耗, 指出能效最高的查询计划不一定是时间最短的, 在查询中存在用较少的时间延迟换取较大能量节省的可能. 如他们通过在固态盘上进行哈希连接和归并连接用 1%延迟换取多达 10%以上的能量节省.

Xu[15][16]在 PostgreSQL 的代价模型的基础上构建了功耗代价模型, 功耗评估模型通过计算单个元组、单个带索引元组以及硬盘页面的功耗参数来评估查询处理的功耗以及相应的时间, 其查询优化的代价公式采用 $C = PT^n = ET^{n-1}$, 其中 P 代表功率、E 代表能量、T 代表时间, n 根据用户的实际需求进行选择. 当 n = 0 时, 代价只考虑功耗, 即功率最小化; 当 n = ∞时, 代价只考虑时间, 即性能最大化; 当 n = 1 时, 代价变为能量, 优化成为最小化能量, 对功率与时间同等对待. 其实验表明, 选择不同的 n 可以进行能量与性能的折衷. CPU 读、写数据页操作(如调用 fread/fwrite)存在一定时间开销, 也会产生 CPU 能耗, 在 Xu[16]所提方法中没有提及; 以功耗评估模型中顺序扫描功率代价为例, 采用每条元组消耗功率乘以元组数加上每页消耗的功率乘以关系的页面数来计算其功率代价, 其物理含义不明晰. 在连接过程中涉及的谓词演算, 如 A.a＝B.b, 需要访问元组的内容, 不同的元组大小也会影响硬盘的 I/O 次数. 所以, 忽略元组大小是不合理的; 无索引元组与带索引元组功耗参数确定为一个固定值也显得过于武断, 不同的数据表、不同索引会有不同的功耗值, 且其代价模型物理含义不合理.

综上所述, 基于能效的数据库查询优化已引起学术界的重视, 提出了不同于传统查询处理代价模型的功耗代价评估模型, 基于功耗代价模型进行查询优化; 一些研究人员通过设计实验验证数据库查询优化的可行性. 在能效数据库系统方面, 对 CPU 和硬盘的能耗关注较多, 但对内存能耗则关注较少.

本文将评估查询计划中 CPU、内存、存储系统的动态能耗和执行时间, 比较不同优化目标下查询结果, 为后续 QoS 数据库的能效查询提供新的思路.

3 代价评估及查询优化算法实现

3.1 代价评估方法

本文代价评估将分别考虑时间和能量代价, 通过代价模型, 评估查询的能耗和时间代价, 优化查询计划, 返回满足用户要求的最小代价计划, 比较两者的异同, 总体流程如图 1 所示.

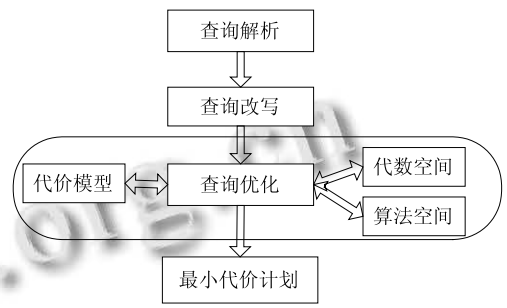

图 1 查询优化流程图

在查询处理代价评估中, 对于时间代价, 本文采取传统硬盘的 I/O 页面数来估计, 而忽略 CPU 耗时, 这里不再细述. 下面讲述能量代价的估算. 首先给出静态功率和动态功率的定义. 静态功率是指数据库服务器处于空闲状态时系统的功率; 动态功率是指数据库服务器执行任务而使系统产生的额外功率. 文中若不特别声明, 出现的能耗代价即指动态能耗代价, 功率即为动态功率. 对于查询过程中产生的中间结果, 优先选择物化在存储系统. 此过程产生的 I/O 时间代价将计算到存储系统的 I/O 时间中. 产生的能耗代价, 将分别计算进内存和存储系统的能耗代价中.

本文根据查询过程中产生的中间关系及其大小以及资源配置来估算查询计划的能耗代价. 利用 CPU、内存和存储系统等不同部件的功率, 乘以各个操作的耗费时间来评估每个具体操作的能耗. 在整机能耗的评估中, 主要考虑 CPU、内存和存储系统的能耗. 即: $E=E_{cpu}+E_{mem}+E_{disk}$, 其中 E_{cpu}、E_{mem}、E_{disk} 分别表示 CPU、内存和硬盘的能耗. 在时间代价的估算中, T_{page} 是页面平均访问延迟, 对于硬盘, T_{page}=寻道时间+旋转延迟+传输延迟, 在实际中, 传输时间相对于寻道时间和旋转延迟来说数值较小, 可以忽略不计, 所以 $T_{page}\approx$寻道时间+旋转延迟; N_{page} 表示访问的页面数; $T_{disk}= N_{page} * T_{page}$ 表示访问 N_{page} 磁盘页所需时间. 在能量代价估算中, P_{disk} 表示硬盘的活动功率, $E_{disk} = P_{disk} * T_{disk}$ 表示磁盘存取 T_{disk} 时间所产生磁盘能耗. 在实验中, 当存储系统为 SSD 或 SCM 时, 本文同样根据功率参数 P_{SSD} 和 P_{SCM} 来估算相应 I/O 过程中的能耗.

在 CPU 代价的评估中, 文献[14]采取的方法是估算 CPU 进行各种操作(散列、连接)时的指令数, 通过指令数目和指令周期的乘积来确定 CPU 执行的时间. 但这种做法存在缺陷: CPU 在进行不同操作的时候, 涉及到底层操作系统函数调用、循环计数、递归等多项操作, 难以较精确地评估不同 CPU 操作的指令数. 因此, 本文直接通过实验测得 CPU 执行不同操作的平均时间代价, 以此来评估 CPU 执行整个查询计划的时间和能耗代价. 本文主要考察 CPU 的哈希连接操作, 哈希操作涉及到 CPU 的操作主要有比较、交换、哈希和保存操作[14]. 哈希过程只是将两个元组的(key, pointer)键值读入缓存进行比较, 并不是将整条元组读入缓存; 而交换则是将整条元组记录进行交换. 本文通过实验发现, 在硬盘的I/O操作时, CPU本身也要耗时(文献[14]中仅考虑 CPU 写操作是不够的), 因此, 在页面 I/O 时需考虑 CPU 的时间和能耗代价. 下面给出 CPU 进行读取、比较、交换、哈希和保存时所对应的时间参数, 分别为 T_{read}、T_{comp}、T_{swap}、T_{hash} 和 T_{save}, 其中 T_{read} 表示从硬盘读入一页数据至缓存时 CPU 耗费的时间, T_{save} 表示从缓存写入一页数据至硬盘时 CPU 耗费的时间, T_{comp} 表示不同关系之间一对键值比较时 CPU 耗费的时间, T_{hash} 表示一次哈希操作时 CPU 耗费的时间.

据此, CPU 哈希连接时的时间代价公式为: $T_{cpu} = 2 *(N_{page}(R) + N_{page}(S))* T_{read} +(2 * N_R + N_S) * T_{hash} + N_S * T_{comp} +(N_{page}(R) + N_{page}(S) + \varphi_{sel} * N_{page}(R) * N_{page}(S))* T_{save}$, 其中 R 是哈希过程中的小表, S 为大表, $N_{page}(X)$表示关系 X 的页面数, N_R、N_S 表示 R 和 S 的元组数目. P_{cpu} 表示 CPU 的活动功率. $E_{cpu}=P_{cpu} * T_{cpu}$ 表示 CPU 在 T_{cpu} 时间内的能耗.

目前的内存架构中, 只要有一页数据处在读写状态, 所有的内存都会保持在此活动态下, 处在相同级别的能耗状态[17], 而且在整个查询阶段, 数据不停地通过内存进行交换, 内存几乎一直处在活动态, 内存的能耗变成一个相对固定的数值. 所以在整个查询阶段, 并不能简单的根据每一步操作的内存使用量来评估内存的能耗. 本文采用的方法是: 只计算因查询计划而产生的内存能耗代价, 即对于每一个查询计划, 预先分配固定的内存供其使用, 由此产生的能耗算入查询计划的整体能耗代价中. 在内存能耗的评估中, 内存的功率 P_{mem} 是固定值, 内存的使用时间 T_{mem} 即整个查询处理的时间, 从读取某个关系的硬盘数据开始, 到CPU返回查询结果为止. 所以内存的占用时间 $T_{mem} = T_{disk} + T_{cpu}$, 内存能耗代价为: $E_{mem} = T_{mem} * P_{mem}$.

3.2 查询优化算法实现

本文利用动态规划算法来查找最小代价的查询计划. 动态规划算法的逻辑如下:

1) 设 $R_1, R_2, \cdots\cdots, R_k$ 是需要进行连接的关系, 动态规划算法的任务是找出K元自然连接 $\{R_1, \cdots, R_k\}$ 最小能耗策略, 并产生左深树计划. 初始时, 每一个关系为此左深树的叶结点且能耗皆为 0, 即对于 i=1 to k, $C_{energy}(\{R_i\})=0$;此时的最小能耗策略为其关系本身, $Strat(\{R_i\})=R_i$.

2) 从最底层节点开始, 每一次中间结点的构建, 都需要探测选择最小能耗的组合. 即对于大小为 S(S=2 to k) 的子集 $Z\{R_1, \cdots, R_k\}$. 需要计算 $\min\{C_{energy}(Z-\{W_j\})+C_{energy}(((Z-\{W_j\}),W_j))\}$, 并求 $W_j=\text{argmin}\{C_{energy}(Z-\{W_j\})+C_{energy}(((Z-\{W_j\}),W_j))\}(w_j \in Z)$的最小解, 此步骤中构建的左深树左节点为 $Z-\{W_j\}$, 右节点为 W_j. 由此得到的最小能耗策略为 $Strat(Z)= Strat(Z-\{W_j. \}).W_j$.

3) 在计算 $C_{energy}(Z)$时, $C_{energy}(Z-\{W_j\})$已计算过; 同样, 在计算 Strat(Z)时, $Strat(Z-\{W_j\})$已计算过. 在构建树节点时, 该方法修剪了除 $Strat(Z-\{W_j\})$之外的全部策略.

4 性能评价

4.1 实验环境

本文的查询实验通过仿真实现, 但是实验中所需的参数, 都是通过真实测量得到. 实验的硬件环境如表 1 所示. 在实验中, 本文关闭了系统缓存, 去除了系统缓存给实验带来的影响. 通过实验测得的 CPU 时间, 仅是 CPU 执行指令的时间, 并非整个 CPU 执行过程中的全部耗时(非 CPU“墙上时间”), 结合硬件的参数, 得到以下的时间参数表.

表 1 实验环境参数

设备	型号	峰值功率(w)	动态功率(w)
CPU	Intel(R)Core(TM)i5-2400 3.1GHZ	95	30
内存	kingston DDR3 1600MHZ 2GB	3*2	3*2
硬盘	WD5000AAKX	6	5.3
固态盘	Samsung 830 128G	2.4	2

表 2 CPU 指令时间耗时

时间参数	T_{read}	T_{comp}	T_{swap}	T_{hash}	T_{save}
时间(μs)	2.05	0.1015	0.1067	0.102	5.59

表 3 TPC-H 数据集

表名	part	partsupp	supplier	nation	region	lineitem	orders
字母	A	B	C	D	E	F	H

I/O 时间测量中, 当存储系统为 HDD 时, T_{page} = 5000 μ s/页; 当存储系统为 SSD 时, T_{page} = 104 μ s/页; 当存储系统为 SCM 时, 通过分析论文[19]中的数据得出 T_{page} = 0.1 μ s/页, 功率为 0.1w.

实验中的数据来自 TPC-H 标准数据库基准测试, 为了方便实验结果的展示, 本文用英文字母代替 TPC-H 的表名, 其对应关系如表 3 所示.

实验中的查询语句 Q_1 和 Q_2 为:

Q_1: select * from A, B, C, D, E where A.partkey = B.partkey and B.suppkey = C.suppkey and C.nationkey = D.nationkey and D.regionkey = E.regionkey .

Q_2: select * from A, B, C, D, F, H where A.partkey = F.partkey and B.suppkey = F.suppkey and C.nationkey = D.nationkey and C.suppkey = F.suppkey and H.orderkey = F.orderkey .

实验针对每个查询改变存储介质和数据集大小进行相应查询优化. 存储介质有 HDD、SSD 以及虚拟的 SCM, 采用数据集规模有 2G、5G 和 10G.

4.2 实验结果

在实验结果中, 时间代价以查询结果的页面数来表示, 能耗代价以查询消耗的能量来表示, 单位为焦耳. 结果如表 4 所示.

表 4 Q_1 的实验结果

数据大小(G)	存储介质	最小时间连接顺序	最小时间代价(页)	最小能耗连接顺序	最小能耗代价(焦)
2	SSD	D C B A E	59854	E D C B A	104
2	HDD	D C B A E	59854	D C B A E	4562.5
2	SCM	D C B A E	59854	E D C B A	65
5	SSD	D C B A E	149650	E D C B A	260.4
5	HDD	D C B A E	149650	D C B A E	11420.3
5	SCM	D C B A E	149650	E D C B A	162.8
10	SSD	D C B A E	299304	E D C B A	521.5
10	HDD	D C B A E	299304	D C B A E	22833.2
10	SCM	D C B A E	299304	E D C B A	326.1

从实验结果中可以看到: 1.若存储介质为 HDD, 则最小时间代价计划即为最小能耗代价计划. 这是由于硬盘传输速度较慢, 耗时较多, 导致功耗较高, 在整个查询中, 硬盘功耗占主导地位; 以 2G 大小的查询 Q_1 为例, 其 I/O 传输时间达到 824 秒, I/O 传输的功耗占总功耗的 95%. 这预示着对于查询 Q_1 和 Q_2, 大部分的时间和能量都消耗在数据传输上面, 表明基于硬盘的数据库能效的调节, 空间比较小.

若存储介质为 SSD, 查询 Q_1 的最小时间计划和最小能耗计划不同, 本文进一步考察两种不同计划下的 CPU 耗时和 I/O 耗时, 得到表 5 所示数据.

表 5 Q_1 不同查询结果 CPU 和 I/O 时间对比

数据大小(G)	存储介质	最小时间代价下 CPU 时间	最小时间代价下I/O时间	最小能耗代价下 CPU 时间	最小能耗代价下I/O时间
2	SSD	2.39	17.1	2.16	18.2
2	SCM	2.39	0.26	2.16	0.28
5	SSD	5.99	42.9	5.39	45.4
5	SCM	5.99	0.66	5.39	0.7
10	SSD	11.99	85.95	10.8	91
10	SCM	11.99	1.32	10.8	1.4

从表中可以看出, 最小能耗计划 EDCBA 的 I/O 时间与最小时间计划 DCBAE 的 I/O 时间相比, 都略大. 所以在传统数据库中, 基于时间的优化目标下, 会返回 DCBAE 的查询结果. 但是在新型的基于能耗的优化目标下, 由于计划 EDCBA 的 CPU 使用时间较

小, 导致 CPU 耗能较少, CPU 耗能的减少幅度大于 I/O 时间增长带来的能耗. 于是返回 EDCBA 的查询结果.

从逻辑上分析两种不同的查询计划, 发现 E 表的数据量非常小, 全表只有 5 条记录, 所以在最小能耗代价中, E 表先做连接, 只会带来非常小的 I/O 时间增长, 而对于本身功率不大的 SSD 来说, 此部分能耗增长较小. 但是在 CPU 的哈希操作中, 此计划却节省了较多的 CPU 使用时间, 节省了较多的 CPU 能耗, 对于整体计划来说, 占主导地位.

Q_2 的实验结果中, 最小时间代价和最小能耗代价计划, 都为 DCFHAB, 两者没有差别. 这是因为在 Q_2 中每个表的数据量都较大, 非最小时间计划会带来较大的 I/O 时间增长, I/O 产生的能耗在能耗主体中占主导地位. 此时最小时间和最小能耗代价计划优化目标一致, 会返回相同的查询结果.

若存储介质为 SCM, 由于它代表一种传输速度更快, 能耗更小的新硬件, 代表一种未来存储系统的发展趋势; 在这种仿真环境下, 通过实验发现最小时间连接策略和最小能耗连接策略是不同的, 这意味着未来数据库能效优化是可行的.

5 小结

本文研究了数据库查询处理的能耗问题, 建立了新型的能量评估公式, 从整体到部分进行了详细的能耗参数评定, 通过实验验证了数据库中基于能量进行查询优化的可行性, 为能效数据库的未来研究提供了新的思路.

参考文献

1 Freitas RF, Wilcke WW. Storage-class memory: The next storage system technology. IBM Journal of Research and Development, 2008, 52(4/5): 439–448.

2 Greenawalt P. Modeling power management for hard disks. Proc. of the 2nd International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems. 1994. 62–66.

3 Douglis F, Krishnan P, Marsh B. Thwarting the power hungry disk. Proc. of Winter USENIX Conference. 1994. 293–306.

4 Carrera E, Pinheiro E, Bianchini R. Conserving disk energy in network servers. Proc. of ACM International Conference on Super Computers. 2003. 86–97.

5 Colarelli D, Grunwald D. Massive arrays of idle disks for storage archives. Proc. of the 16th International Conference on Supercomputing. 2002. 1–11.

6 Gurumurthi S, Sivasubramaniam A, Kandemir M, Franke H. DRPM: Dynamic speed control for power management in server class disks. Proc. of the Symposium on Computer Architecture. 2003. 169–179.

7 Tsirogiannis D, harizopoulos S, Shah MA. Analyzing the energy efficiency of a database server. Proc. of SIGMOD. 2010. 231–242.

8 Deng Q, Meisner D, Ramos L. Active low-power models for main memory with memscale. IEEE Micro, 2012, 32(3): 60–69.

9 Kumar K, Doshi K, Dimitrov M, Yung-Hsiang L. Memory energy management for an enterprise decision support system. Proc. of the International Symposium on Low Power Electronics and Design. 2011. 277–282.

10 Pisharath J, Choudhary A, Kandemir M. Reducing energy consumption of queries in memory-resident database systems. Proc. of the International Conference on Compilers. 2004. 35–45.

11 Ayoub R, Indukuri KR, Rosing TS. Energy efficient proactive thermal management in memory subsystem. Proc. of International Symposium on Low Power Electronics and Design. 2010. 195–200.

12 Tolentino ME, Turner J, Cameron KW. Memory MISER: Improving main memory energy efficiency in servers. IEEE Trans. on Computers, 2009, 3(58): 336–350.

13 Lang W, Patel JM. Towards eco-friendly database management systems. Proc. of Conference on Innovative Data Systems Research (CIDR), 2009.

14 Lang W, Kandhan R, Patel JM. Rethinking query processing for energy efficiency: Slowing down to win the race. IEEE Data Eng. Bull. 2011. 12–23.

15 Xu Z. Building a power-aware database management system. Proc. of the SIGMOD PhD Workshop on Innovative Database Research (IDAR). 2010. 1–6.

16 Xu Z, Tu Y, Wang X. Exploring power-performance tradeoffs in database systems. Proc. of 26th International Conference on Data Engineering. 2010. 485–496.

17 Yuhui D. What is the future of disk drives, death or rebirth? ACM Computing Surveys(CSUR), 2011, 43(3): 1–27.