计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2019.20180670
56(9): 1821-1831, 2019

# 面向绿色数据中心的能耗有效查询优化技术

邢宝平[1] 吕梦圆[1] 金培权[1,2] 黄国锐[3] 岳丽华[1,2]
[1](中国科学技术大学计算机科学与技术学院 合肥 230027)
[2](中国科学院电磁空间信息重点实验室 合肥 230027)
[3](中国人民解放军 31002 部队 北京 100081)
(lmys@mail.ustc.edu.cn)

# Energy-Efficiency Query Optimization for Green Datacenters

Xing Baoping[1], Lü Mengyuan[1], Jin Peiquan[1,2], Huang Guorui[3], and Yue Lihua[1,2]
[1](*School of Computer Science and Technology, University of Science and Technology of China, Hefei* 230027)
[2](*Key Laboratory of Electromagnetic Space Information, Chinese Academy of Sciences, Hefei* 230027)
[3](*Unit* 31002, *People's Liberation Army of China, Beijing* 100081)

**Abstract** Reducing energy consumption and building green datacenters has been one of the major needs of modern large-scale datacenters. In green datacenters, a key research issue is how to lower the energy consumption of database systems while keeping stable performance. This issue is called energy efficiency, and has become a new research frontier recently. Energy efficiency of database systems is defined as using little energy to accomplish as many operations as possible. High energy efficiency means that database systems can use less energy while processing a fixed number of operations. In other words, it uses less energy but achieves the same performance. In this paper, we propose a method for energy-efficient query optimization. First, an operator-level power model is established based on the regression analysis method, which can accurately predict average power consumption during query execution for a given query. Next, a new cost model is proposed for query optimizer, which considers both energy and performance aspects. The new cost model uses a new factor to obtain a better tradeoff between performance and energy costs. A testbed is built for measuring energy consumption of database systems, and the TPC-H and TPC-C benchmarks are used to evaluate the performance of our proposal. The results show that the proposed power model achieves higher precision than existing methods. In addition, the proposed performance-degrade factor can provide flexible trade-offs between performance and energy. Moreover, by setting up an appropriate performance-degrade factor, better energy efficiency can be achieved than the original PostgreSQL.

**Key words** green datacenter; energy efficiency; query optimization; cost model; power model

**摘 要** 降低能耗开销、建设绿色数据中心,已经成为目前大规模数据中心的重要需求.在绿色数据中心,如何使数据库系统在满足性能需求的前提下尽量地节约能耗,即如何提高数据库系统的能耗有效性,是目前研究的重点.数据库系统中的能耗有效性旨在使用更少的电能来提供相同的服务.能耗有效性越高,说明数据库系统可以用更少的能耗就能够响应同样数量的操作,换句话说,可以用更少的能耗

**收稿日期**:2018-09-26;**修回日期**:2019-06-14
**基金项目**:国家自然科学基金面上项目(61672479)
This work was supported by the General Program of the National Natural Science Foundation of China (61672479).
**通信作者**:金培权(jpq@ustc.edu.cn)

达到同样的性能.据此提出了一种面向绿色数据中心的能耗有效查询优化方法.该方法首先利用回归分析建立操作符层的功耗预测模型,从而可以准确地预测给定查询在执行过程中的平均功耗.接着,在PostgreSQL查询优化器中扩充了结合预测能耗成本和时间成本的新的查询执行代价计算模型,并引入性能退化度因子调节性能和能耗的权重.最后构建了数据库系统能耗测试平台,在PostgreSQL上基于TPC-H和TPC-C基准测试进行了实验.结果表明:所提出的功耗预测模型比已有方法准确度更高.同时,提出的性能退化度因子为数据库系统提供了性能和能耗之间的灵活折中方案,并且通过设置适当的性能退化度因子,可以实现比原始PostgreSQL更高的能耗有效性.

**关键词** 绿色数据中心;能耗有效性;查询优化;代价模型;功耗模型

**中图法分类号** TP311

自哥本哈根大会之后,建设低碳社会已经成为全球共识.面向可持续发展的低成本、低能耗的新型计算系统、模型和应用的研究,或称为绿色计算,已成为未来信息技术领域面临的重大挑战[1].在当前以数据为中心的计算模式下,研究节能的低能耗数据库系统不仅具有显著的应用价值和社会意义,同时对于推动数据库领域的发展也具有重要的科学意义.

低能耗数据库系统的主要目标是实现数据库系统的低能耗,同时兼顾性能.传统的数据库系统通常以高性能为目标进行设计,没有充分考虑数据存储与操作时的能耗有效性(energy efficiency).所谓能耗有效性[2],通常指使用更少的电能来提供相同的服务,例如缓冲区管理、查询执行等.由于能耗在现有大型数据管理系统(通常是数据中心)中的费用比例逐年升高(目前大约占总能耗的16%左右)[3],不仅给企业带来了沉重的经济负担,产生的碳排放也会带来一系列的社会问题和国际影响.因此,研究能耗有效的数据库系统,降低数据管理的能耗,已经成为政府、企业和学术界普遍关心的焦点问题之一[4-7].

数据中心的能耗总体上受到2个方面的因素影响:第1是单个服务器节点的能耗;第2是由于分布式环境带来的能耗代价,例如网络通信、任务调度、分布式执行等带来的能耗开销.本文主要针对单个服务器节点的能耗问题.传统的数据库服务器以性能为导向,各模块实现时并没有考虑到能耗因素.作为数据库系统核心模块的查询优化器目前均基于性能(即时间代价的估计)来进行执行计划的选择.已有研究表明[8],现有查询优化器选中的执行计划通常是性能最优,但并不是能耗/性能比最优的.但已有工作[8]仅考查了有限的物理操作符的能耗,而且物理操作符的能耗采用了经验性估计方法,存在较大的误差(例如,没有剔除统计量装载等能耗代价).此外,也没有在查询处理器中提供灵活的能耗和性能折中的方案.

本文提出了一种能耗有效的数据库查询优化新方法.该方法通过在执行计划生成过程中对时间代价进行预测的同时,引入计划执行的能耗代价,进而做出更好的计划选择策略.同时,由于不同的应用对性能和能耗有着不同的要求,因此我们的方法也允许用户在性能和能耗之间进行灵活选择.总体而言,本文的主要工作和贡献可总结为3点:

1) 提出了一个基于操作符的执行计划功耗模型.我们结合数据库应用层信息和操作系统层信息,基于回归分析方法构建了操作符功耗模型,进而给出了整个执行计划的功耗模型.

2) 提出了一个性能/功耗均衡的查询代价模型,给出了结合功耗代价与时间代价的总代价计算方法,并引入了性能退化度调节因子,能够方便且灵活地进行性能/功耗的均衡控制.

3) 构建了数据库系统能耗测试平台,在开源数据库PostgreSQL上实现了能耗有效的查询优化器,并基于TPC-H和TPC-C基准测试进行了大量实验.结果表明,我们所提出的功耗预测模型比已有方法准确度更高.同时,提出的性能退化度因子为数据库系统提供了性能和能耗之间的灵活折中方案,可以实现比原始PostgreSQL更高的能耗有效性.

## 1 相关工作

### 1.1 能耗有效的数据库查询处理技术

设计一个能耗有效的数据库管理系统(database management systems, DBMS),需要保证在性能退化较少的情况下大大降低系统的能耗,同时又不影响系统的扩展性和可靠性.文献[7]提出了考虑延迟的能耗有效查询(query energy-efficiency with delays, QED)算法,通过队列缓存一段时间内接收到的查询

请求,达到阈值后通过查询聚合,提取请求中的公共部分.这一显式延迟的方案更适用于多用户场景,如处理搜索引擎的服务器集群.对于单节点数据库系统,反而会使得性能急剧下滑.另一种方式,可以在查询优化器设计时加入能耗因素的考量.这需要在DBMS设计时建立2个模型:1)能耗模型,将它加入查询优化器中,从而能够在评估每个查询计划的代价时考虑到能耗因素;2)总代价模型,结合新加的能耗和性能模型,得出每个查询计划的总代价,从而选择最优的查询计划.

因此,对查询优化的改进,重点就在于建立良好的能耗模型和总代价模型.为此,需要了解系统硬件层面的性能和操作特性,以及能耗与性能间的联系[9].文献[8]利用PostgreSQL中性能模型,通过使用待定系数法,调节参数后建立了能耗模型.尽管该方法通过数学方法确定了系数,但模型的操作符的能耗代价基于经验性估计,与实际代价差异较大(在本文的实验中误差达25%),因此该能耗模型的合理性有待商榷.

### 1.2 数据库系统的资源预测模型

在数据库系统中,估计SQL查询的资源消耗是关乎系统性能方面至关重要的工作.查询优化器中内建的代价模型为给定查询提供了基势估计.这一方面有很多的工作,包括基于采样的方法[10-11]、基于直方图的方法[12]等.

近来的工作都在探讨使用基于机器学习的方法[13-17].文献[13]通过使用混合模型(plan-template models, operator-level models),来预测查询执行时间.但该模型使用的输入特征都是来自于数据库内部数据,没有考虑系统层的影响.文献[15]中研究了在并发环境下如何预测查询的执行时间,使用了多元线性回归模型来捕捉查询间相互作用.文献[16]则是基于一种特殊的循环神经网络——长短期记忆(long-short term memory),能够预测一个特定的查询计划的执行时间区间.文献[17]在单站点数据库服务器上构建了资源单位统一的能耗预测模型,采用多元线性回归模型拟合模型的重要参数。

文献[8]表明,现有查询优化的代价估计选中的执行计划通常是性能最优,但并不是能耗/性能比最优的.文献[8]保留原有的性能模型,初步探索了能耗有效的查询优化研究.依然存在2个问题:

1) 实验中整机功耗的测量使用的是万用表,不仅数据误差大,而且无法具体查看不同操作下各模块的功耗.

2) 使用的功耗模型基于的是标量函数,无法准确地描述系统的功耗代价.在动态环境下,该功耗模型的预测结果较差.

## 2 能耗有效的查询优化框架

不同于已有的基于执行时间预测建模的研究,本文集中于预测查询执行过程中的平均功耗(average power).之所以这样选择,主要考虑了2个方面:

1) 如果我们直接预测能耗,将会是一个十分复杂的问题. 而查询操作的能耗可以通过查询执行时间、执行平均功耗计算得到.由于这2个环节相互之间较为独立,因此通过这种方式可以避免模型估计误差的传递,并将1个复杂问题分解成2个较简单的问题,便于问题的讨论和实现.

2) 数据中心中功耗一直被看作是最重要的优化目标之一.当系统运行在低功耗状态时,冷却系统的耗能也随着大大降低.

因此,本文实现了能耗有效的查询优化器.首先需要建立功耗预测模型,在此基础上,将其融入到现有的查询计划的代价计算模型中,最终选取能耗有效性更好的查询计划.

### 2.1 总体方案

性能模型通过以时间为度量单位的时间模型(time model, TM)表示,用于估计不同查询计划的执行时间.而能耗模型(energy model, EM)是以焦耳(J)为度量单位,用于估计不同查询计划执行的电能消耗大小.功耗模型(power model, PM)是以瓦特(W)为度量单位,用于估计不同查询计划执行中系统功耗大小.PM与TM之间相对独立,但是由于已有的TM无法给定查询计划的实际执行时间,因此在实际使用时并不能简单地通过“能耗=功耗×时间”来度量,而是需要定义合理的参数或值来调节功耗和时间的权重,才能创建更合理的EM.

基于查询优化器的工作原理和流程,图1给出

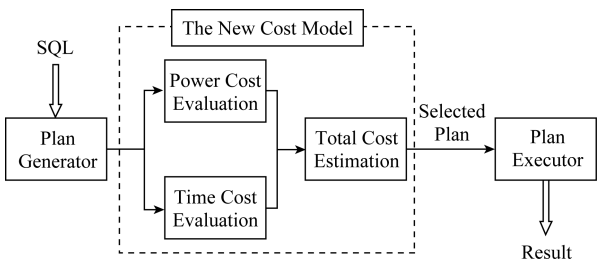


Fig. 1 Design of energy efficient query optimizer

图1 能耗有效查询优化器的设计

了本文的能耗有效查询优化器的设计思路,它主要包含 2 个方面的内容:

1) 功耗代价的计算

功耗代价用于预测给定执行计划在实际运行中的平均功耗.具体可分为 3 个步骤完成:

第 1 步. 确定训练数据集中数据格式和采集方案.对于每类物理操作符,确定在其执行过程中需要记录哪些数据,再设计相应的数据格式和运行负载.在负载运行过程中进行数据采集.

第 2 步. 建立操作符层功耗模型.基于第 1 步采集的数据,提取特征数据,利用回归分析,建立操作符层功耗模型.

第 3 步. 建立执行计划的功耗模型.基于第 2 步建立的操作符层功耗模型,通过单个物理操作符的功耗预测来计算整个执行计划的平均功耗.

2) 计划间代价比较及执行计划选择

功耗模型建立后,对于每个查询的执行计划都会得到 2 个代价值:估计执行时间 $T$ 以及预测执行的平均功耗 $P$.而后将其融入到已有的代价计算架构中,选择能耗有效性最好的查询计划.

### 2.2 数据采集方案

已有研究发现,不同操作符之间有着不同的功耗[7-8]. 即使在 CPU 使用率相同的情况下,功耗差异最大可达 60%.因此对于单个数据库物理操作的执行功耗,不仅要考虑 CPU 和内存的使用情况[18-19],还需要考虑数据库的运行状态.最终,要收集的数据由 3 部分数据组成:系统当前运行状态的表征数据、操作符执行时可能读取的页面数目等数据和执行的功耗数据.在建立的功耗模型中,将系统和数据库内部的表征数据作为输入来预测功耗.

图 2 显示了我们所设计的数据采集平台架构,主要由 2 个模块构成:

1) 内部监控器(internal monitor).在能够提供数据库服务的测试机上实现内部数据采集.在 PostgreSQL 中,通过 explain analyze+SQL 查询语句的查询执行方法的方式,能够输出结果的同时输出代价计算值和相关估计参数值.通过修改 explain analyze 的代码,可以提取数据库参数.通过修改查询执行中的代码,在对应的物理执行计划执行前,查询系统运行状态(CPU 使用率等).组织后得到内部采集数据,格式为(操作符类型标识,数据库参数,系统参数,开始时间,结束时间).以顺序扫描操作符的执行为例,采集的数据格式为($seq\_scan$,读写页面数目,处理元组数目,CPU 使用率,开始时间 $t_1$,结束时间 $t_2$).

2) 外部监控器(external monitor).用于监控测试机部件的实时功耗,采集的功耗数据格式为(采集时间,CPU 功耗,磁盘功耗,整机功耗). External monitor 通过我们自行设计的数据库系统能耗测试平台来实现.在 3.2 节中将具体介绍该平台的实现架构.

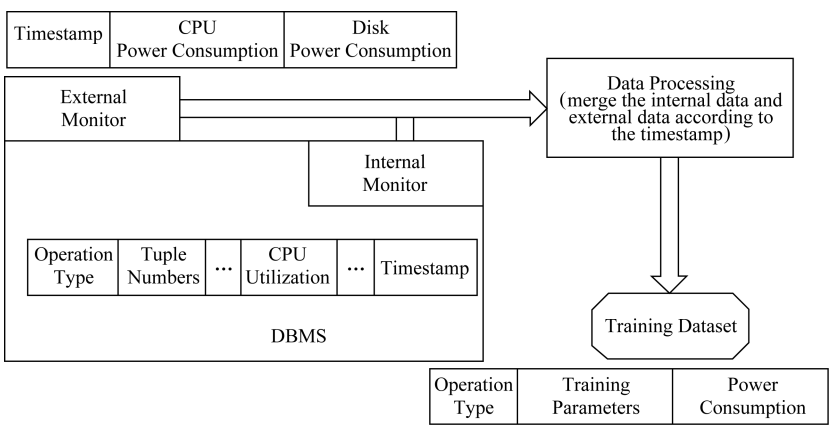


Fig. 2 Data acquisition platform and data format design

图 2 数据采集平台及采集数据格式设计

对于系统信息,可以收集的信息有:CPU 使用率、内存使用率、系统热设计功耗(thermal design power, TDP)值. 其中,系统 TDP 值是系统运行功耗上限,是常量值.CPU 使用率和内存使用率可以实时获取.具体地,在 Linux 系统中,可以通过读取文件/proc/stat/proc/meminfo 中内容来分别确定当前 CPU 的使用率和内存使用率.stat 文件中包含了所有 CPU 活动的信息,该文件中所有值都是从系统启动开始累积到当前时刻.执行 cat/proc/stat 后显示内容如表 1 所示:

**Table 1 CPU Running Information**

**表 1 CPU 运行信息**

| CPU | *User* | *Nice* | *Sys* | *Idle* | *Iowait* | *Irq* | *Sofirq* |
|---|---|---|---|---|---|---|---|
| $CPU_{total}$ | 814 | 131 | 435 | 88 449 | 2 071 | 0 | 3 |
| $CPU_0$ | 217 | 9 | 150 | 22 326 | 270 | 0 | 1 |
| $CPU_1$ | 203 | 1 | 118 | 22 370 | 257 | 0 | 0 |
| $CPU_2$ | 218 | 120 | 98 | 21 144 | 1 408 | 0 | 0 |
| $CPU_3$ | 175 | 0 | 68 | 22 607 | 134 | 0 | 1 |

通过选择 2 个时刻 $t_1$,$t_2$,分别读取文件的内容,经过计算得到这段时间内的 CPU 使用率,即

$$\text{CPU 使用率}=100\%\times\frac{(User_{t_2}+Sys_{t_2}+Nice_{t_2})-(User_{t_1}+Sys_{t_1}+Nice_{t_1})}{total_{t_2}-total_{t_1}}. \quad (1)$$

内存使用率需要从 meminfo 文件中提取 2 个数据,分别是当前内存的使用量($cmem$)以及内存总量($umem$),计算公式为

$$\text{内存使用率}=100\%\times\frac{cmem}{umem}. \quad (2)$$

对于数据库信息,表 2 和表 3 通过分类方式列出了在数据库系统内部可以取得的参数及相应含义.

**Table 2 Common Features of All Physical Operators**

**表 2 所有物理操作符的共同特征**

| Operator | Description |
|---|---|
| COUT | Number of Output Tuples |
| SOUTAVG | Average Size of Output Tuples |
| CIN | Number of Input Tuples |
| SINAVG | Average Size of Input Tuples |

**Table 3 Features of Specific Operators**

**表 3 基于特定操作符的特征**

| Operator | Description | Operator Type |
|---|---|---|
| TSIZE | Tuple Numbers of Input Table | Seek/Scan |
| PAGES | Page Numbers of Input Table | Seek/Scan |
| TCOLUMNS | Column Numbers of Tuple | Seek/Scan |
| INDEXDEPTH | Index Depth | Seek |
| HASHOPAVA | Cost of Hash Operation per Tuple | Hash |
| CHASHCOL | Column Numbers of Hash Operation | Hash Agg |
| CINNERCOL | Column Numbers of Inner Join | Joins |
| COUTERCOL | Column Numbers of Outer Join | Joins |
| SSEEKTABLE | Tuple Numbers of Inner Table | Nested Loop |
| CSORTCOL | Column Numbers of Sort Operation | Sort |
| SINSUM | Total Data Size of Multiway Merge Join | Merge Join |

在 PostgreSQL 中具体实现时,需要修改内核中的 costsize.c,createplan.c,explain.c,plannodes.h,relation.h 等文件.通过修改 plannodes.h 和 relation.h 中对于结构体 plan,path 的定义添加相应数据类型的字段来记录信息.已知在 PostgreSQL 数据库中可以通过(explain+SQL 查询)的方式来显示给定 SQL 查询的执行计划树.explain enalyze 会显示带有代价的执行计划树且执行查询.因此通过修改程序中 explain 执行操作的源码,借由系统中本身自有的查询执行树的解析程序来完成记录信息的输出保存.具体的修改主要集中在 PostgreSQL 的 src\backend\commands 目录下 explain.c 文件中的 explain_outNode 函数上.

数据采集工作需要有相应的负载,在查询负载运行过程中进行数据的收集.由于 TPC-H 基准测试中的查询都是复杂查询,其执行树由一系列物理操作符组合而成,因此在实际数据采集中会遇到 2 个问题:1)尽管能耗测试平台(external monitor)能够实时监控测试机的运行功耗.但是由于复杂查询下物理操作符执行时序的不确定性,很难确定每个物理操作的执行时段,进而去确定它的运行功耗.2)对于单个操作符,系统信息采集时误差大.这是由于短时间内频繁地读取/proc/stat,/proc/meminfo 文件造成的.

为此,我们提出了一套简单负载设计原则,通过合理有效地使用简单查询负载来达到数据收集的目的,同时保证收集数据能够较好地反映物理操作符的真实功耗情况.在设计运行负载时,我们使单个

查询尽可能涉及较少的物理操作符、不同查询间尽可能不存在操作符交集、整个负载要尽可能地包括所有类型的物理操作符.最终,设计的运行负载为

SELECT * FROM $R$; (Scan)

SELECT * FROM $R$ ORDER BY $R.a$; (Sort)

SELECT * FROM $R$ WHERE $R.A<a$; (Select)

SELECT COUNT( * ) FROM $R$; (Aggregate)

SELECT * FROM $R$,$S$; (Product)

SELECT * FROM $R$,$S$ WHERE $R.a=S.a$; (Join)

利用 TPC-H 基准测试方法,我们将上述查询实现为 TPC-H 中扩展的查询模板.

### 2.3 执行计划的功耗模型

基于采集的训练数据集,我们采用回归分析的方法来建立操作符层功耗模型.对于单个物理操作符,为了能够得到预测功耗模型,我们使用 2 个特征数据:1)在 DBMS 内部,采集到的物理操作符执行时可能处理的元组数目 $T$、读取页面数目 $N$、选择度 $\sigma$ 等信息;2)当前系统的 CPU 使用率 $C$.单个操作符的预测功耗 $P$ 定义为

$$P=F(T,N,\sigma,C). \tag{3}$$

然后,对 $F$ 用训练数据集进行回归分析.考虑到实际使用中计算复杂度问题,我们使用迭代线性回归的训练方法,以总误差率值 $f$(训练数据误差率 $err$ 的累加,误差率 $err$ 的定义为

$$\text{误差率 } err=\frac{|\text{预测功率}-\text{实际功率}|}{\text{实际功率}}\times 100\%. \tag{4}$$

作为目标函数,回归过程为

步骤 1. 根据初始输入特征 $T,N,\sigma,C$ 和训练集中功耗 $W$,线性回归得到 $F_1$,并计算总误差率值 $f_1$.

步骤 2. 选取单个特征进行简单变换(如 $T\rightarrow T^2$)作为新特征,再次进行线性回归,得到 $F_2$ 和 $f_2$.

步骤 3. 比较 $f_1,f_2$,若 $f_2<f_1$,则将步骤 2 中生成的新特征保留,保存 $F_2$,重复步骤 2;若 $f_2>f_1$,且无新特征,则算法停止.

实现时使用了 GSL(GNU scientific library)工具. GSL 是开源的 C/C++统计学习工具库,能够方便地完成对于多维数据的回归学习.

下一步,由得到的单个物理操作的预测功耗来估计查询执行的平均功耗.任意一个查询 $Q$ 可以看成是由一系列物理操作符 $O$ 组成的执行计划树.每个合法的执行计划树都是由一组物理操作符有序执行的结果.在传统代价估计中,$T_Q$ 为执行计划的时间估计,$t_O$ 为操作符的时间估计,如式(5)所示.已有研究并没有特别强调 $P_Q$ 如何计算,为此我们自然地认为 $P_Q$ 也是如此计算.

$$T_Q=\sum_i t_{o_i}. \tag{5}$$

然而,时间累加代表总时间,功耗的累加并不具有实际意义.本文通过对每个操作符的功耗进行加权累加,而不是直接的叠加方式,最终得到整个执行计划的平均功耗(即查询的功耗),即:

$$Tmp=\sum_i t_{o_i}p_{o_i}, \tag{6}$$

$$P_Q=\frac{Tmp}{T_Q}=\frac{\sum_i t_{o_i}p_{o_i}}{\sum_i t_{o_i}}. \tag{7}$$

其中,$Tmp$ 是中间变量,用来计算每个操作符的时间、功耗乘积.

### 2.4 总的代价模型

通过 2.3 节的功耗模型,对于每个执行计划最终可以得到 2 个值:1)估计执行时间 $T$,无单位,并不与真实执行时间量级上有关.但是在传统数据库系统中能够很好地指导最优计划的选取.2)预测执行平均功耗 $P$(单位 W).

在最终合理的计划集中,能耗有效的优化器要选取的执行计划与传统优化器中选择的执行计划就代价估计而言,常常要选取的计划在功耗上有优势,但是在性能上有所退化.为此,我们定义了总代价模型:

$$Cost=P^{\alpha}T^{1-\alpha}. \tag{8}$$

假设现有 2 个执行计划:Plan1 和 Plan2,并假设 Plan2 的功耗低于 Plan1,我们可以计算出当两者代价相等时 Plan2 的性能 $T^*$,具体过程为

Plan1:已知平均功耗 $P$ 以及执行时间 $T$;

Plan2:已知平均功为 $np$($0<n<1$),推导其执行时间 $T^*$.

$$P^{\alpha}T^{1-\alpha}=(nP)^{\alpha}T^{*1-\alpha}\Rightarrow$$

$$\alpha\lg P+(1-\alpha)\lg T=\alpha(\lg n+\lg P)+(1-\alpha)\lg T^*\Rightarrow$$

$$(1-\alpha)\lg T=(1-\alpha)\lg T^*+\alpha\lg n\Rightarrow$$

$$\lg T^*=\lg T-\frac{\alpha}{1-\alpha}\lg n\Rightarrow$$

$$T^*=\frac{1}{n^{\frac{\alpha}{1-\alpha}}}T.$$

通过上述推导过程,我们发现不同的执行计划的功耗和性能之间存在着可计算的相关性.换句话说,如果查询优化器最终选中比 Plan1 功耗低的 Plan2,参数 $\alpha$ 决定了 Plan2 的性能退化率,即由 $T$ 退化为 $T^*$.

因此,我们将 $\alpha$ 定义为性能退化度调节因子.假设 Plan2 在功耗上有 20% 的收益($n=0.8$),通过

外部的 $\alpha$ 值设置,可以控制能耗有效的查询优化器在选择 Plan2 所允许的最大的性能退化程度(相比传统查询计划选择策略).比如:$\alpha=0.2$ 时,$T^*=1.06T$,允许的性能最大退化度为 6%.因此,在实际系统中,我们可以通过设置合适的 $\alpha$ 值,有效地控制查询计划选择时的性能与功耗之间的折中.$\alpha=0$ 表示不允许性能退化;$\alpha=1$ 表示允许性能可以无限退化,达到功耗最低.表 4 给出了不同退化因子下的性能退化度.

**Table 4 Degenerated Factor $\alpha$ ($n=0.8$)**

**表 4 退化因子 $\alpha$(设 $n=0.8$)**

| $\alpha$ | $1/n^{\frac{\alpha}{1-\alpha}}$ | Degree of Performance Degradation |
|---|---|---|
| 0 | 1 | 0 |
| 0.2 | 1.06 | 0.06 |
| 0.5 | 1.25 | 0.25 |
| 0.7 | 1.69 | 0.69 |
| 0.9 | 7.45 | 6.45 |
| 1.0 | $\infty$ | $\infty$ |

## 3 实验结果

### 3.1 实验设置

实验的软件环境为 PostgreSQL 8.3.3 和 Ubuntu 12.04(内核 3.5.0).硬件设置以及相应的功耗参数如表 5 所示.测试负载使用了 TPC-H 和 TPC-C.

**Table 5 Experimental Environment and Parameter Settings**

**表 5 实验环境和功耗参数设置**

| Hardware | Max Power Comsumption/W |
|---|---|
| CPU i3-3220 | 55 |
| Mainboard and GPU | 16 |
| Memory DDR3 4 GB | 3 |
| WD Disk 1 TB | 6 |
| Others(CD driver, mouse, keyboard) | 15 |
| Complete Machine | 95 |

### 3.2 能耗测试平台

系统运行时,测试平台能够实时对计算部件和存储部件的各路供电线路进行电流和电压采样,并将采样所得的模拟信号经过模电转换器件转换后传输给数据处理系统进行数据的处理和保存.

图 3 给出了系统架构设计.系统由硬件和软件 2 部分构成,其中软件部分由 2 个部分构成:

1) 运行在测试机上的各类测试负载;

2) 运行在数据分析机上的数据收集和分析软件.

硬件部分由 5 部分构成:

1) 能耗测试设备;

2) 计算机 1(作为测试机,提供数据库服务);

3) 计算机 2(数据采集和分析);

4) A/D 转换器;

5) 机箱电源.

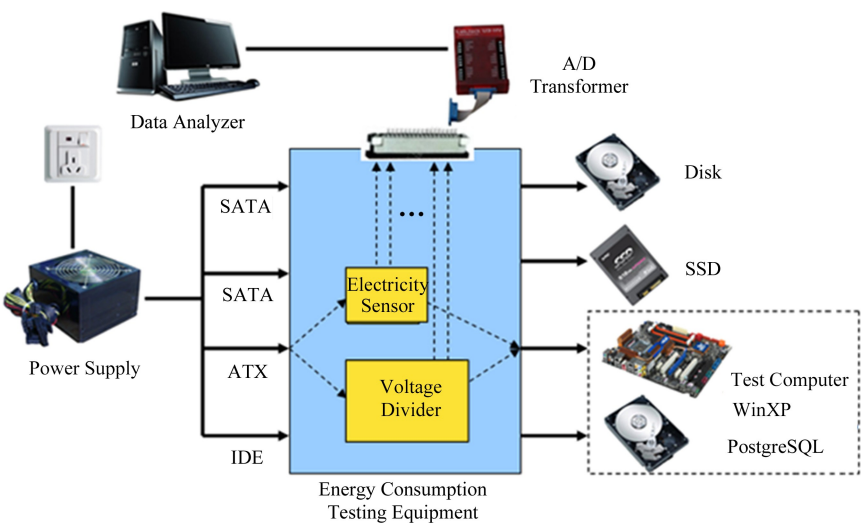


Fig. 3 Energy consumption testing platform architecture

图 3 能耗测试平台架构

### 3.3 TPC-H 基准测试结果与分析

1) 预测精度

我们分别在 2 种不同系统环境下进行了 TPC-H 基准测试.在基准测试生成的 10 GB 数据集上,分别运行 TPC-H 测试查询语句,利用能耗测试平台统计并计算查询执行的平均功耗,进而计算模型精度.

我们测试了静态运行环境和竞争运行环境下的性能.静态运行环境是指系统上只运行数据库服务.

由于测试查询依次单个执行,相当于独占系统.竞争运行环境中则存在其他的应用程序,不是数据库服务器独占系统资源,在实验中我们通过运行 Web 服务程序来模拟.测试结果如图 4 和图 5 所示:

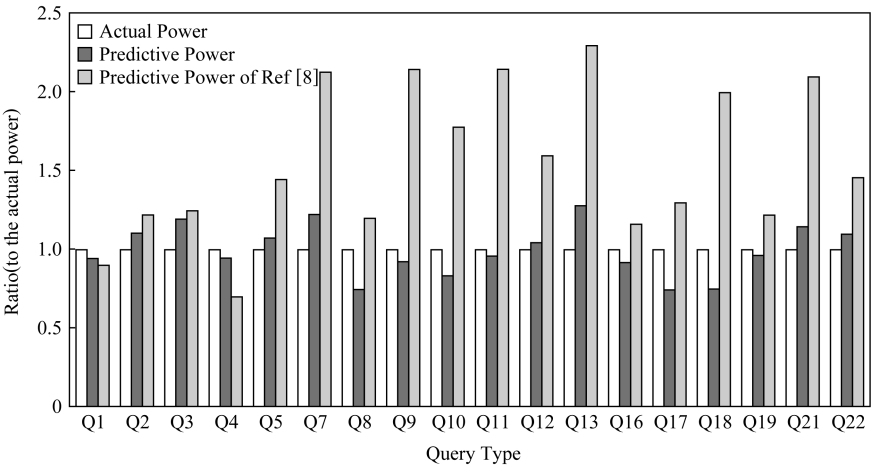


Fig. 4 Accuracy of power consumption prediction of TPC-H in static environment

图 4 TPC-H 在静态环境下功耗预测的精度

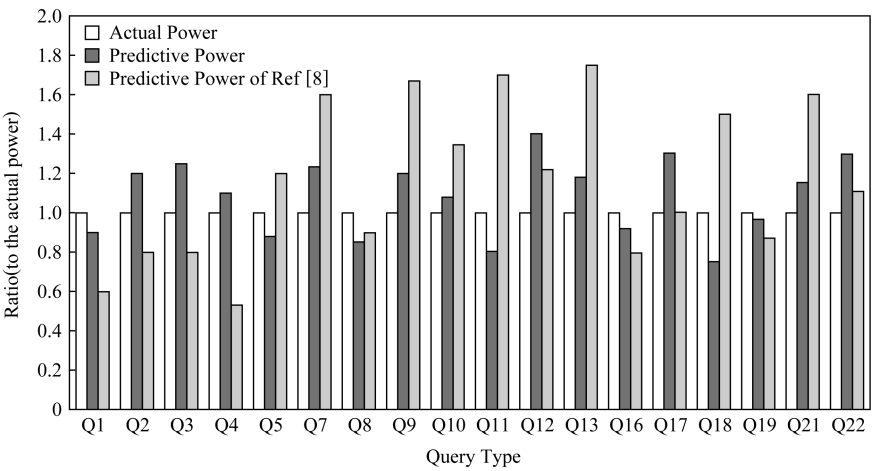


Fig. 5 Accuracy of power consumption prediction of TPC-H in competitive environment

图 5 TPC-H 在竞争环境下功耗预测的精度

在静态运行环境下,本文模型比文献[8]中的功耗模型预测精度更高.文献[8]中普遍存在预测过高的问题.在竞争运行环境下,由于考虑了当前系统运行状态,因此能够对运行功耗进行更精度的估计.

2) 额外代价

本文模型在查询优化阶段获取数据库信息的同时,还要获取当前系统运行信息.对计算得到的代价,还需要进行加权累加,在查询运行时带来了额外代价.为此,我们通过实验来评测查询优化器的额外代价.实验分别对比了 1 GB 和 10 GB 数据集下,未加入能耗代价的 PostgreSQL 和添加了能耗代价模型的 PostgreSQL 的性能和能耗表现.实验中设置 $\alpha=0$,以保证 2 个数据库选择相同执行计划.

设 $t_1$ 为原始的 PostgreSQL 的查询执行时间,$t_2$ 为增加能耗代价模型后的查询执行时间,我们将额外代价定义为 $t_2-t_1$.图 6 和图 7 显示了 TPC-H 查询的实验对比结果,其中 $Y$ 轴为 $(t_2-t_1)/t_1$ 的比值,代表了增加能耗优化后带来的额外时间代价比例.

在 1 GB 的数据量能耗有效的查询优化带来的额外代价较高,说明这一机制在小数据量或者简单查询下额外代价大.主要原因是此类查询本身的执行时间短,生成的候选计划空间小,花费较长时间用于能耗有效的查询优化,反而得不偿失.但是,在 10 GB 数据集上,额外代价相对较小,平均每个查询所增加的额外时间为 3.78%.

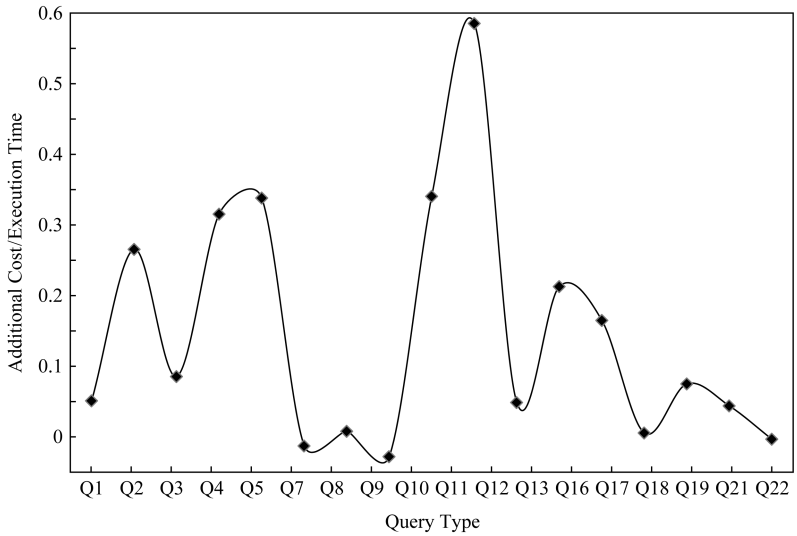


Fig. 6 Additional cost of a single query on the 1 GB data set

图 6 1 GB 数据集上单个查询的额外代价

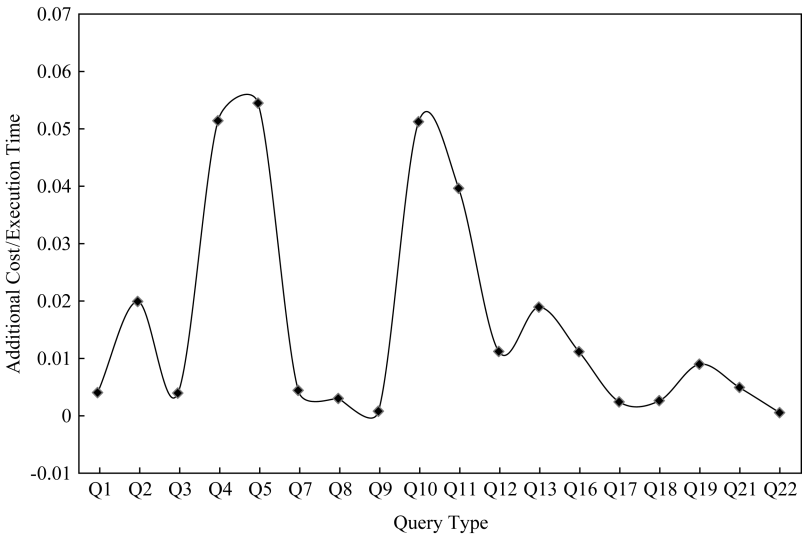


Fig. 7 Additional cost of a single query on the 10 GB data set

图 7 10 GB 数据集上单个查询的额外代价

3) 能耗有效查询优化的总体性能

表 6 给出了不同 $\alpha$ 参数下 TPC-H 总体查询性能(1 GB 数据量)的测试结果.随着 $\alpha$ 的增大,时间性能变差,同时负载执行的平均功耗是降低的,符合我们的预期.通过表 6 可以看出,$\alpha$ 能够有效地调节查询优化器在性能和功耗之间的均衡.如果设置较大

**Table 6 Effect of $\alpha$ on the Overall Query Performance**

**表 6 $\alpha$ 对总体查询性能的影响**

| PostgreSQL | Execution Time/s | Average Power Consumption/W | Energy Consumption/J |
|---|---|---|---|
| Unmodified Version | 974 | 40.7 | 39 642 |
| Modified Version($\alpha$ = 0.1) | 985 | 36.5 | 35 953 |
| Modified Version($\alpha$ = 0.3) | 976 | 40.6 | 39 625 |
| Modified Version($\alpha$ = 0.5) | 1 024 | 38.9 | 39 834 |
| Modified Version($\alpha$ = 0.7) | 1 019 | 38.6 | 39 333 |
| Modified Version($\alpha$ = 0.9) | 1 438 | 35.0 | 50 330 |

的 $\alpha$ 值,意味着总是偏向选择功耗低的执行计划,则会导致查询执行的时间性能下降.在实验中我们发现,$\alpha$ 取值在(0,0.3],可以取得较好的能耗有效性.

### 3.4 TPC-C 基准测试结果与分析

TPC-C 是联机事务处理的基准测试,在本实验中我们使用它来测试所提出的能耗有效查询优化策略对联机事务处理的性能表现.在实验中我们将能耗有效性定义为

$$\text{能效}=\frac{\text{性能}}{\text{能耗}}=\frac{\text{总执行时间}}{\text{平均功耗}}.$$

实验中使用了 1 GB 数据库,模拟开启了 10 个终端,并运行 10 000 个 OLTP 事务.图 8 和图 9 分别给出了执行时间和平均功耗的实验结果.图 10 的能效结果显示能耗有效的查询优化器能够有效提高能效比.结合 TPC-H 中总体测试结果我们可以得出结论:能耗有效的查询优化器,通过添加对功耗代价的考量能够起到能耗优化的目的,但是不能单纯以功耗优化为目的做出忽略性能的计划选择.

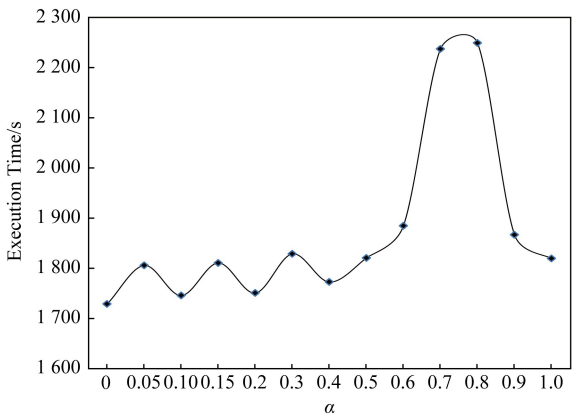


Fig. 8 Execution time of under different $\alpha$ values

图 8 不同 $\alpha$ 值下测试的执行时间

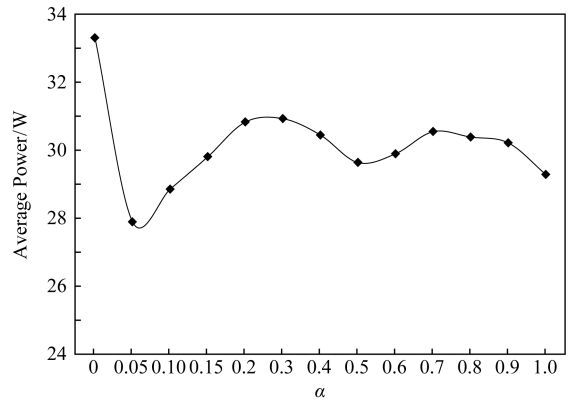


Fig. 9 Average power consumption under different $\alpha$ values

图 9 不同 $\alpha$ 值下测试执行过程中的平均功耗

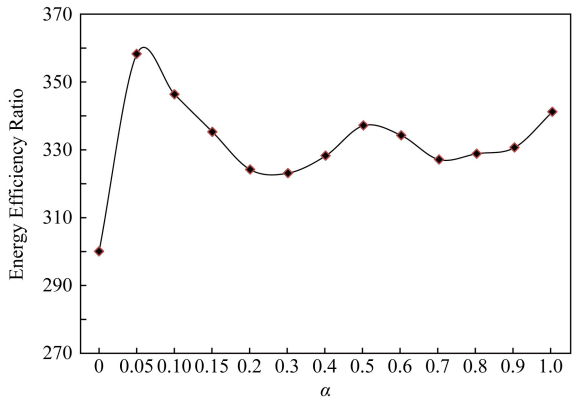


Fig. 10 Energy efficiency ratio under different $\alpha$ values

图 10 不同 $\alpha$ 值下测试执行的能效比

## 4 结束语

本文着眼于在查询优化器中引入能耗代价,提出了基于回归分析的操作符功耗预测方法.此外,通过引入性能退化调节因子,提出了兼顾能耗和性能的查询代价模型,并最终在 PostgreSQL 上实现了能耗有效的查询优化器.基于 TPC-H 和 TPC-C 的实验结果也表明了能耗有效查询优化技术的可行性、有效性以及实用性.

由于目前的关系型数据库系统普遍采用了基于代价的查询优化器,因此本文的研究成果理论上也同样适用其他类型的关系型数据库系统.在本文的实验中,我们主要基于开源的 PostgreSQL 开展了性能和能耗方面的测试.目前其他的商用 DBMS 如 Oracle,Microsoft SQL Server,MySQL 等均采用了与 PostgreSQL 类似的基于代价的查询优化技术,因此本文的结果具有一定的普适性,理论上可以推广到其他关系型数据库中.

在未来工作中,我们首先将研究连接、聚集等复杂查询操作符的功耗代价估计方法,并提出针对复杂查询的能耗有效查询优化方法.其次,由于目前数据中心往往以服务器集群的形式部署,因此数据中心的能耗一方面受到每个服务器节点的影响,也受到集群中的分布式调度、数据迁移等问题的影响,在未来工作中,我们将针对数据中心中的分布式查询处理、任务调度等问题研究能耗有效的解决方案.

## 参 考 文 献

[1] Jones A. Green computing: New challenges and opportunities [C] //Proc of the 24th Great Lakes Symp on VLSI 2017. New York: ACM, 2017: 3

[2] Xing Baoping. Research on energy efficient query processing technologies for database systems [D]. Hefei: University of Science and Technology of China, 2014 (in Chinese)
(邢宝平. 能耗有效的数据库查询处理技术研究[D]. 合肥:中国科学技术大学, 2014)

[3] Vaid K. Datacenter power efficiency: Separating fact from fiction[C/OL] //Invited Talk at the 2010 Workshop on Power Aware Computing and Systems. 2010[2016-07-20]. https://www. usenix. org/legacy/event/hotpower10/tech/slides/vaid.pdf

[4] Xie Jiazhuang, Jin Peiquan, Wan Shouhong, et al. EPSCS: Simulating and measuring energy proportionality of server clusters [G] //LNCS 9050: Proc of the 20th DASFAA. Berlin: Springer, 2015: 536-540

[5] Jin Peiquan, Xie Xike, Jensen C, et al. HAG: An energy-proportional data storage scheme for disk array systems [J]. Journal of Computer Science and Technology, 2015, 30(4): 679-695

[6] Yang Puyuan, Jin Peiquan, Yue Lihua. Exploiting the performance-energy tradeoffs for mobile database applications [J]. Journal of Universal Computer Science, 2014, 20(10): 1488-1498.

[7] Lang W, Patel J. Towards eco-friendly database management systems[C/OL] //Proc of the 4th CIDR. 2009[2016-07-20]. http://www-db.c-s.wisc.edu/ cidr/cidr2009/Paper_51.pdf

[8] Xu Zichen, Tu Yicheng, Wang Xiaorui. Online energy estimation of relational operations in database systems [J]. IEEE Transactions on Computers, 2015, 64(11): 3223-3236

[9] Schall D, Hudlet V, Harder T. Enhancing energy efficiency of database applications using SSDs [C] //Proc of the 3rd Canadian Conf on Computer Science & Software Engineering. New York: ACM, 2010: 1-9

[10] Hou Wenchi, Ozsoyoglu G. Statistical estimators for aggregate relational algebra queries [J]. ACM Transactions on Database Systems, 1991, 16(4): 600-654

[11] Xie Jiazhuang, Jin Peiquan, Wan Shouhong, et al. Energy-proportional query processing on database clusters [G] // LNCS 9098: Proc of the 16th Int Conf on Web-Age Information Management. Berlin: Springer, 2015: 324-336

[12] Ioannidis Y. The history of histograms (abridged)[C] //Proc of VLDB. San Francisco: Morgan Kaufmann, 2003: 19-30

[13] Akdere M, Cetintemel U, Riondato M, et al. Learning-based query performance modeling and prediction [C] //Proc of the 28th ICDE. Washington, DC: IEEE, 2012: 390-401

[14] Kandasamy K, Schneider J, Poczos B. Query efficient posterior estimation in scientific experiments via Bayesian active learning [J]. Artificial Intelligence, 2017, 243: 45-56

[15] Li Hui, Hou Xiaohuan, Chen Mei. Performance prediction for concurrent workloads in distributed database systems [C] //LNCS 9531: Proc of the 15th ICA3PP. Berlin: Springer, 2015: 626-639

[16] Bi Liyuan, Wu Sai, Chen Gang, et al. Database query cost prediction using recurrent neural network [J]. Journal of Software, 2018, 29(3): 799-810 (in Chinese)
(毕里缘, 伍赛, 陈刚, 等. 基于循环神经网络的数据库查询开销预测[J]. 软件学报, 2018, 29(3): 799-810)

[17] Guo Binglei, Yu Jiong, Liao Bin, et al. SQL energy consumption forecasting model based on database load status [J]. Computer Science, 2017, 44(1): 208-213 (in Chinese)
(国冰磊, 于炯, 廖彬, 等. 基于数据库负载的SQL能耗预测模型[J]. 计算机科学, 2017, 44(1): 208-213)

[18] Chandrasekharan S, Gniady C. QAMEM: Query aware memory energy management [C] //Proc of the 18th CCGrid. Piscataway, NJ: IEEE, 2018: 412-421

[19] Luo B, Hayamizu Y, Goda K, et al. Modeling query energy costs in analytical database systems with processor speed scaling [G] //LNCS 11030: Proc of DEXA. Berlin: Springer, 2018: 310-317

**Xing Baoping**, born in 1989. Master. His main research interests include energy-efficient task scheduling and green computing.

**Lü Mengyuan**, born in 1995. Master candidate. His main research interests include green computing and big data storage.

**Jin Peiquan**, born in 1975. PhD. Associate professor. His main research interests include databases on new storage, spatial-temporal databases, and Web information retrieval.

**Huang Guorui**, born in 1974. PhD. Associate professor. His main research interests include big data and artificial intelligence.

**Yue Lihua**, born in 1952. Professor. Her main research interests include flash-based databases, spatial-temporal databases, and remote-sensing image processing. (llyue@ustc.edu.cn)