文章编号: 1001-0920(2005)01-0109-04

数据仓库系统中任务调度策略研究

史 捷[1], 鲍玉斌[2], 刘运涛[2], 张 斌[2], 孙焕良[2], 于 戈[2]
(1. 中铁九局集团有限公司 信息中心, 辽宁 沈阳 110013; 2. 东北大学 信息科学与工程学院, 辽宁 沈阳 110004)

摘 要: 针对数据仓库系统中存在众多的数据处理任务, 提出了数据仓库系统中任务调度框架. 通过自动分析任务文本生成任务流图, 并提出了 3 种任务调度策略. 模拟实验结果显示, 该框架可以很好地完成数据仓库系统中的任务调度和管理, 同时发现影响系统性能的因素还应包括磁盘 I/O 性能.

关键词: 数据仓库系统; 任务调度; 工作流; 多处理机
中图分类号: TP31 **文献标识码**: A

On task scheduling strategy in data warehouse system

SHI Jie[1], *BAO Yu-bin*[2], *LIU Yun-tao*[2], *ZHANG Bin*[2], *SUN Huan-liang*[2], *YU Ge*[2]
(1. Information Center, China Railway No. 9 Group Co. Ltd, Shenyang 110013, China; 2. School of Information Science and Engineering, Northeastern University, Shenyang 110004, China. Correspondent: SHI Jie, E-mail: xxzx@stju.com.cn)

Abstract: To data processing tasks in data warehouse system (DWS), a framework of task scheduling in DWS is presented. In the framework, the task flow diagram is generated automatically by analyzing task-texts and three task-scheduling methods are proposed. The simulation results show that the framework can implement task scheduling in DWS perfectly, and that disk I/O performance is a factor having impact on system performance also.

Key words: data warehouse system; task scheduling; workflow; multiprocessor

1 引 言

数据仓库是一个为决策者提供联机分析处理(如决策支持、数据挖掘)所需要信息的仓储. 它是面向主题的、集成的、随时间改变的、持久的数据集合, 主要用于支持经营管理中的决策制定过程[1]. 数据仓库中的数据是按主题组织的, 具有不同的粒度级别, 并支持多种前端分析应用[2]. 因此, 数据仓库中的处理过程众多, 包括建立数据仓库的数据抽取、转换、加载、数据集市生成、数据立方生成过程, 数据仓库管理的更新、归档过程以及应用报表生成程序等. 对于企业级数据仓库系统, 其中的处理程序成千上万, 而这些处理程序之间的关系千丝万缕. 如何有效地调度和管理这些处理任务是数据仓库管理中非常重要的工作, 也是提高数据仓库性能和资源利用率的关键.

关于任务(作业)调度在批处理系统中的讨论已有很多, 提出了许多算法[3]. 然而, 这些算法都是讨论对于给定的一批相互之间没有制约关系的作业, 使它们按什么顺序执行会使得某个指标(如平均周转时间)达到最优. 目前, LSF[4]及其变体在作业调度方面得到了大量的使用. 但是, LSF 工具需要用户给出作业队列, 而且需要用户描述作业之间的同步和互斥关系. 然而, 在数据仓库应用中, 这个工作对用户应该是透明的, 即用户只需提供有关的处理作业和相关信息, 而如何调度它们执行是数据仓库管理员的工作. 但是, 数据仓库系统中作业的数量巨大, 完全由数据仓库管理员手工安排是不现实的. 即使可以安排, 也是粒度很粗的安排. 因此, 不能完全

收稿日期: 2003-08-04; **修回日期**: 2004-06-15.
基金项目: 国家自然科学基金项目(60173051).
作者简介: 史捷(1956—), 男, 江苏南京人, 工程师, 从事数据仓库与数据挖掘等研究; 于戈(1962—), 男, 辽宁大连人, 教授, 博士生导师, 从事数据库理论等研究.

依靠LSF作为数据仓库系统中的任务调度工具,还需要研究在满足某种要求前提下的数据仓库系统中任务的自动调度策略.为此,本文提出基于工作流思想和方法来完成任务流的调度.

2 基本概念和问题描述

数据仓库系统中的各种处理过程以及它们之间的相互依赖关系构成了一个有向图(见图1).下面给出数据仓库系统中任务调度的问题描述.

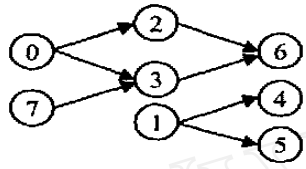

图1 任务流图

定义1(任务) 一个任务是一个在特定环境下运行的一个程序或命令,也称为作业(Job).一个任务有很多描述其调度和执行需求的属性.任务调度系统利用任务的属性和系统的资源信息以及调度策略,决定何时、何地、如何运行一个任务.在一个实际数据仓库系统中,一个任务可以是一个数据集的抽取程序或一个报表的生成程序等.

定义2(任务流图TF) 一个系统中的任务流图TF是一个有向无环图,表示为一个二元组(V, E).其中:$V=\{T_1, T_2, \cdots, T_n\}$是由任务组成的顶点集合;$E=\{e_1, e_2, \cdots, e_m\}$是由任务之间的依赖关系构成的有向边的集合,即一些任务对$\langle T_i, T_k\rangle$的集合.如图1所示,其中一个任务的所有前导任务之间是"与"关系,即只有一个任务的所有前导任务都完成,它才可以执行.令PreCount(T_i)表示任务T_i的前驱任务数.

每个节点(任务)都具有一些属性,如任务号、存储的位置和估计的执行时间(或估计计算量)等.在调度时需要参考任务的估计执行时间.

任务调度就是按照一定的策略将任务按照它们之间的依赖关系分先后调度执行的过程.本文讨论的数据仓库系统中的任务调度问题的关键是如何找出数据仓库系统及其应用中任务之间的时序关系,并按照一定的策略选择一个或一组任务投入"运行",并使得所有任务的执行占用CPU的时间尽可能短,并能够监视和控制任务的执行.

调度的衡量指标包括:1)使得所有任务的总的周转时间尽可能短.所谓一批任务的周转时间是指从它们开始运行到全部运行结束所经历的时间.2)CPU的利用率高且均衡(多CPU情况下),即每个CPU都参与计算,而且负载均衡.

3 任务调度系统框架

数据仓库任务调度系统的基本框架如图2所示.数据仓库管理员(DWA)通过任务登记器登记任务的编号和存放的物理路径等基本信息,这些基本信息存放在数据仓库系统的任务仓中.任务登记器根据这些基本信息,对任务的源码进行扫描,得到每个任务的输入数据集和输出数据集.扫描结束,启动流程生成器.流程生成器根据任务登记器提供的信息,得到每个任务的前驱和后继任务,生成数据仓库系统的任务流图.任务调度器根据任务流图对任务进行调度执行(关于任务调度方案的具体讨论见第4节),将选中的任务交给任务执行器包装成可执行体,提交给系统运行.同时把任务的执行信息和出错信息登记到任务仓.任务监控器监控任务流程的执行,并监视系统中资源的使用情况.它收集任务的执行状态信息,并记录系统中任务的执行情况和系统中资源的使用现状,进行系统资源利用率的统计分析,供DWA优化管理系统时参考.出错处理器按照用户指定的出错处理规则处理错误.

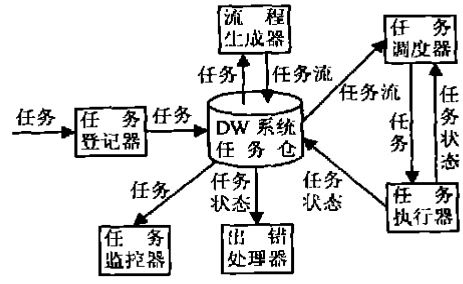

图2 数据仓库任务调度系统结构图

4 任务调度算法

任务调度器是一个独立的进程.它根据任务流图中描述的任务间的时序关系,控制对任务的排序,对任务进行合理分批,并调用任务执行器启动任务执行.目标是使系统的资源利用率尽可能地高,尤其是发挥CPU的效率.

本文提出3种调度方案:动态调度、静态调度和同层划分调度.调度的单位可以是单个任务或一批任务.下面具体讨论各种调度方案.

4.1 动态调度方案

动态调度方案是以单个任务为调度单位的方案.其调度策略是:调度进程首先顺序扫描任务列表,对于那些前驱数为零的任务,直接产生线程并提交给操作系统;然后等待线程结束的信号.任务执行器发现任务执行结束后,更新任务流图中的有关信息,将以该任务为前驱的任务的前驱计数值减1,一旦发现任务流图中某个任务的前驱任务数为零,则创建新的线程执行该任务.

这种调度策略的特点如下:

1) 调度灵活, 不需对图进行排序, 适合于各种类型的任务流图(如连通性强或弱)的调度, 因而便于图的扩展;

2) 在有些运行环境下, 为运行一个任务, 需启动一个应用程序实例(如 SAS 系统), 这便增加了任务执行的附加开销;

3) 因为任务并发执行结束时需要更新任务流图, 因而需解决对任务流图的互斥访问.

4.2 静态调度方案

因为所有待调度的任务是预先知道的, 执行具有相对的稳定性, 因此静态调度也不失为一种不错的选择. 它的具体方法是寻找任务图的连通分支, 对一个连通分支内部的任务进行拓扑排序, 一个连通分支作为一个批整体提交, 排序在前的先执行, 排序在后的后执行. 例如: 有编号分别为 0～7 共 8 个任务节点, 它们之间的时序关系如图 1 所示. 它具有两个连通分量, 它们对应的拓扑序列为(0, 7, 2, 3, 6)和(1, 4, 5).

静态调度方案的特点如下:

1) 调度简单, 仅基于图的连通性和拓扑结构进行, 执行序列在执行前便已经确定, 与动态调度方式相比, 不需并发修改任务表;

2) 对图的依赖性很高, 连通分支有可能大小不均以及数量不确定, 因而会造成各个处理机的负载不均衡.

4.3 同层划分调度方案

实际应用中, 任务流图的深度并不深, 而图的宽度却很宽. 另外, 在数据仓库系统环境中, 数据的组织是分层的, 如细节数据层、轻度汇总层、高度汇总层等, 因而可按层调度任务执行. 下面先给出相关的概念:

定义 3(同层任务集)　在任务流图 TF 中, 按照广度优先策略搜索没有前驱的任务, 这样形成一个任务集合, 记作 $ts(TF)=\{t \mid CountPre(t)=0\}$, 简记为 ts. 称这个集合中的任务是位于同一层的任务, 这个集合称为同层任务集. 同时, 用 TF-ts(TF)表示从任务流图 TF 中除掉 ts(TF)中的任务及与其相连的边形成的任务流图.

定义 4(任务流图的同层划分)　对于一个任务流图 TF, 在得到第 1 个同层任务集 $ts_1(TF)$ 之后, 逻辑上将它们从图中删除, 这样将得到在剩余图($TF-ts_1(TF)$)中的另一个同层任务集 $ts_2(TF-ts_1(TF))$, 一直重复这个过程直到剩余的图为空. 这样得到的有序的同层任务集$(ts_1, ts_2, \cdots, ts_n)$称为一个任务流图的同层划分.

例如: 图 1 的同层划分结果是({0, 7, 1}, {2, 3, 4, 5}, {6}). 如果将{0, 7, 1}作为一个调度单位, 则只有在它的 3 个任务都运行结束, 才可能执行任务集{2, 3, 4, 5}.

同层划分调度策略如下: 首先按照广度优先拓扑排序, 对任务流图进行同层划分; 然后对每个同层任务集按照负载均衡的原则分配到各个 CPU. 由于处在同一层的任务都可以并发执行, 它们不必再排序. 但是, 只有前一层上的所有任务都正确执行完成后, 后一层的任务才能提交. 如前所述, 层与层之间需保持同步, 层内各组完全并行.

同层划分调度策略具有如下特点:

1) 该方案有效地提高了任务执行的并发度, 同时减小了对任务图结构的敏感性;

2) 改方案可以将分在同一个 CPU 上的任务再分小组按批提交, 因此可以加大执行的粒度, 降低了任务切换的开销;

3) 该方案在层与层之间需保持同步, 在此期间由于负载的不平衡, 会使一些 CPU 出现空闲现象.

4.4 实验模拟研究

实验中设计了一个具有 41 个任务的任务流, 共 4 个层次, 9 个连通分支. 模拟实验的硬件环境为 PIII 933 Hz CPU, 512 M 内存的惠普服务器, 共享磁盘和内存.

模拟方案的主要思想是用进程模拟 CPU, 用 SQL 语句模拟任务(主要为 select 语句, 使用了 avg, sum 等聚集函数来模拟 I/O 型和计算型任务, 最大数据集有 100 多万条记录). 在每个进程中再创建多个线程模拟在一个处理机上的多个并发执行的任务. 每个线程运行一个(或一批)任务, 计算每个任务的时间开销, 每个任务处理的数据元组数大于等于 1 000 条. 可以测得每个线程(任务)实际占用处理机的时间(不包括因等待 I/O 而挂起的时间). 各进程模拟的处理机, 在多处理机环境下可以认为它们在多个 CPU 上的并行计算时间是重叠的, 但 I/O 时间不重叠. 这里只模拟任务流的周转时间. 实验的原则是每一组任务启动一个进程, 在进程内给每个任务创建相应的一个线程. 通过 API 函数(getThreadTimes)得到每个线程的创建时间、退出时间以及 CPU 占用时间. 一个组内所有线程的占用 CPU 时间之和就是这一组任务的 CPU 占用时间, 所有线程中最晚退出的时间减去所有线程中最早创建时间就是这组任务的运行时间, 它与 CPU 占用时间之差就是这组任务的 I/O 时间. 通过扫描任务的执行日志, 可得到每个任务的大致执行时间. 因此, 实验开始时, 在单任务环境下可测得各个任务的

CPU 时间和 I/O 时间. 另外, 模拟方案中不考虑创建线程和调度线程的时间开销. 图 3 显示了一个具有 41 个任务的任务流周转时间与 CPU 个数的关系.

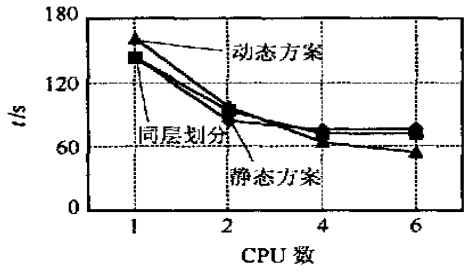

图 3 任务流的周转时间与 CPU 数目的关系

从图 3 可以发现, 增加 CPU 数目会缩短一批任务的总的周转时间, 尤其从一个增加到两个时. 但是, 再增加 CPU 数, 时间缩短的幅度不大. 分析发现, 在该批模拟的任务中有一个任务的时间开销很长, 总的时间主要由该任务决定. 在实际环境中, 因为任务数很多, 所以会弥补这方面的问题. 另外, 从该测试可以发现, CPU 数目增加到一定数量后(如图中的 3 或 4), 作业流的时间开销主要由 I/O 时间决定. 因此, 数据仓库服务器的配置在考虑扩展 CPU 数目的同时, 还应考虑磁盘的配置和数据的分配技术. 如果采用共享磁盘, 当 CPU 数目达到一定值后, 系统的 I/O 能力便成为系统性能的瓶颈. 采用 RAID 磁盘, 可以缓解 I/O 压力.

在本实验样本环境下, 处理机数量增加时, 动态调度算法体现出优越性, 比其他两种的性能好. 在不考虑调度开销的情况下, 动态调度可以使 CPU 利用率最高.

5 结 语

本文利用工作流思想调度数据仓库系统中的任务. 通过自动分析任务文本发现每个任务的输入和输出, 形成任务的描述表; 然后根据各个任务的输入输出数据集之间的关系, 得到数据仓库系统中任务间的任务流图. 针对任务流图提出了 3 种任务调度策略. 从实验结果看, 3 种算法的定量性能类似, 随着处理机数量的增加, 动态算法优于其他两种方法. 从算法的灵活性和可扩展性方面考察, 动态调度和同层划分调度优于静态调度; 然而从算法实现的角度, 静态算法最简单. 模拟实验结果表明, 影响作业流周转时间的因素不仅有 CPU 的数目和调度算法, 而且还有磁盘 I/O 的效率. 因此, 基于此任务流的调度方法可以为数据仓库的配置提供量化决策依据, 即通过构造周转时间与 CPU 数目的关系曲线, 便可确定合理的 CPU 数目.

参考文献(References)

[1] Inmon W H. *Building the Data Warehouse* [M]. 2nd Edition. New York: Wiley, 2003.

[2] Wu M C, Buchmann A P. Research issues in data warehousing [A]. *Proc of BTW '97* [C]. Ulm, 1997: 61-68.

[3] William Stallings. *Operating Systems Internals and Design Principles* [M]. 4th Edition. New Jersey: Prentice Hall, Inc, 2001.

[4] Platform Inc. Load Sharing Facility (LSF) [EB/OL], http://www.platform.com/products/wm/LSF/, 2002/10/15.

(上接第 105 页)

5 结 论

上述研究结果表明, 与以往[1,5]采用的扩展卡尔曼滤波器相比, 模型简单, 系统运算量降低, 实时性较好. 将 GPS 定位误差视为一阶马尔科夫过程, 扩充为状态变量, 利用自适应卡尔曼滤波器对车辆位置和速度信息进行估计是可行的, 效果良好. 为了改善滤波器的动态性能, 采用加权自适应卡尔曼滤波算法, 仿真结果表明效果较好. 从而说明了本文的 GPS 车辆导航动态模型, 能够有效地降低 GPS 定位信号的随机干扰, 提高车辆动态定位精度, 降低车辆导航定位的成本, 提高车辆导航定位系统的实用性.

参考文献(References)

[1] 万德钧, 房建成, 王庆. GPS *动态滤波的理论、方法及其应用*[M]. 南京: 江苏科学技术出版社, 2000.

[2] Singer R A. Estimating optimal tracking filter performance for manned maneuvering targets [J]. *IEEE Trans on Aerospace and Electronic Systems*, 1970, 6(4): 473-483.

[3] Zhou H R, Kumar K S P. A current statistical model and adaptive algorithm for estimating maneuvering targets[J]. *J of Guidance, Control and Dynamics*, 1984, 7(5): 596-602.

[4] 袁信, 俞济祥, 陈哲. *导航系统*[M]. 北京: 航空工业出版社, 1993.

[5] Moussa R. Making the best with GPS in car application [A]. *Proc of ION GPS-95* [C]. California, 1995: 1819-1823.