蚁群算法在教育资源分布式共享系统中的应用①

李 昕, 荆永君

(沈阳师范大学 教育技术学院, 沈阳 110034)

摘 要: 针对教育资源分布式共享系统中资源中心和用户任务之间的调度问题, 建立该类问题的通用调度模型, 讨论基于蚁群算法的任务调度机制, 实现资源中心和用户任务间的优化匹配方案. 仿真实验表明, 该方法具有较好的稳定性和有效性, 减少用户任务完成时间, 提高系统服务效率.

关键词: 分布式; 教育资源系统; 任务调度; 多目标优化问题; 蚁群算法

Application of Ant Colony Algorithm in Distributed and Shared Educational Resources System

LI Xin, JING Yong-Jun

(School of Educational Technology, Shenyang Normal University, Shenyang 110034, China)

Abstract: The schedule between servers and users is a MOP (Multi-objective Optimization Problems), in a distributed and shared educational resources system. Ant colony algorithm has been proved to be a kind of effective algorithm to solve it. A common model of the above schedule was given, and ant colony algorithm was applied to optimize the schedule. The experiment results show that the effect of this optimization is evident.

Key words: distributed; education resources system; mission schedule; MOP; ant colony algorithm

教育资源建设并不是一步到位, 而是一个“缺失—供给—平衡—缺失—供给……”不断循环的动态过程. 随着资源建设研究机构的新资源不断积累和资源用户群的日益增大, 目前教育资源建设存在一个明显的问题: 新资源不能及时更新到资源用户端而资源用户又迫切的需求新资源. 如何实现有效的资源共享已成为当前教育资源建设中一个亟待解决的问题, 是制约教育资源建设和教育信息化进一步发展的瓶颈.

目前基于Internet的资源共享模式, 从资源共享方式上来说, 主要有集中式、分层式和网状式三种[1]. 任务调度策略按照任务的目的划分为: 面向应用、面向系统和面向市场的调度三种; 从调度阶段划分可划分为静态调度和动态调度两种[2]. 集中式和分层式共享系统当用户比较多时, 很难处理中心服务器负载平衡问题, 并且容错能力差. 网状式共享系统虽然具有很强的容错能力和扩展性, 但是, 经调查发现各个资源用户很不情愿把自己的硬件资源贡献于这种模式[3]. 在调度策略上主要采用面向应用、静态的调度策略, 在调度执行前做出所有的决定, 从特定任务的角度来衡量每个可能的调度方案, 从中选择能够最大程度满足特定任务要求的调度方案. 在任务运行过程中不能根据运行时环境动态地调整, 尽可能使系统中各个中心的负载达到基本平衡, 不能从整个系统的角度来衡量每个可能的调度方案, 最大程度提高整个系统的性能.

1 系统设计

为了实现教育资源的有效共享和及时更新, 笔者基于“辽宁省基础教育资源网”, 设计一种基于任务调度服务器的教育资源分布式共享系统(图 1). 任务调度服务器由任务接收器、任务调度器、任务转发器、任务监测器, 以及原始任务队列、优化任务队列、任务日志和信息库组成. 任务调度服务器以 Web Service 形式向资源中心和资源用户提供访问接口, 其的目的是实现资源中心和资源用户间的合理匹配.

① 基金项目:辽宁省教育科学“十二五”规划(JG11DB280)
收稿时间:2012-02-19;收到修改稿时间:2012-03-23

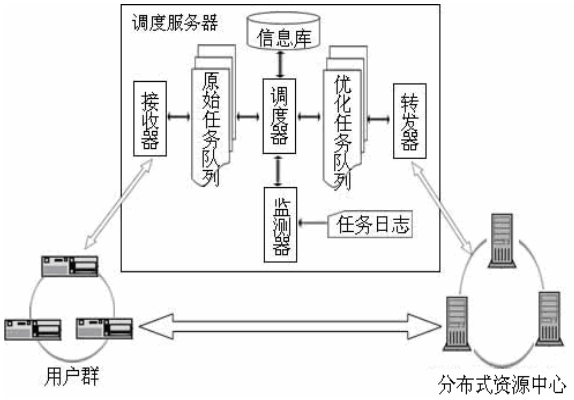

图 1 分布式教育资源共享系统

该系统中有多个平等的、处于不同地理位置的资源中心，每个中心都可以在任务调度程序的支配下，独立向资源用户提供下载服务. 这样既有效地解决了集中式和分层式服务器负载瓶颈问题，也不会像网状式增加资源用户的负担[1].

系统工作流程如下:

(1) 用户向调度服务器提出任务需求，任务接收器收集相关信息形成原始任务队列.

(2) 任务调度器负责资源用户和资源中心之间的分配方案，根据原始任务队列和历史信息库相应信息，按照一定的调度策略选取一种优化的分配方案，按照优先级别将任务重新排队，形成优化任务队列.

(3) 任务转发器根据优化任务队列调度相应资源中心为用户任务提供服务.

(4) 任务监测器根据任务日志，激发任务调度器是否需要调整调度策略，重新优化任务队列.

系统的调度策略采用面向系统的、动态的调度策略，从整个系统角度来衡量每一可能的调度方案，均衡每一个资源中心的下载任务，力图避免对于单个中心的过分资源请求. 并在系统运行过程中，不断地根据运行时环境，对系统的任务调度进行动态地调整，尽可能使系统负载达到基本平衡.

任务调度器是此系统中最复杂的一个模块，可在其中设置多种优化调度策略，本文所讨论的基于蚁群算法的调度策略就是其中一种.

2 问题描述

给定系统情况如下: 有 m 个资源中心服务器 $c_1,c_2,\cdots,c_m$，为 n 个用户 $u_1,u_2,\cdots,u_n$ 进行资源共享服务. 第 j 个中心 c_j 连接用户数量上限为 Max_{cj}，第 i 个用户 u_i 下载任务为 M_{ui}，任务 M_{ui} 的物理大小为 SM_{ui}，完成时间为 TM_{ui}，用户 u_i 连接资源中心的数量上限为 Max_{ui}，用户 u_i 和中心 c_j 之间下载速度为 V_{ui-cj}.

系统中优化调度的方式主要考虑下载任务 M_{ui} 的资源中心选择上.

约束条件: 资源中心连接用户上限为 Max_{cj} 和用户连接资源中心的上限 Max_{ui}.

优化的目的: 所有下载任务完成时间趋于最小.

优化调度的数学模型为:

$$\min \quad E=\sum_{i=1}^{n} TM_{ui}$$

$$TM_{ui}=SM_{ui}\Big/\sum_{j=1}^{m} V_{ui-cj}\cdot x_{ui\,cj}$$

$$s.t. \quad \sum_{i=1}^{n} x_{ui\,cj}\le Max_{cj} \qquad (j=1,2,\mathrm{L},m)$$

$$\sum_{j=1}^{m} x_{ui\,cj}\le Max_{ui} \qquad (i=1,2,\mathrm{L},n)$$

$$x_{ui\,cj}=\begin{cases}1 & \text{如果 } c_j \text{ 为 } u_i \text{ 提供服务}\\ 0 & \text{否则}\end{cases}$$

在本系统中，不同的服务器可以对不同的用户进行响应，并且每个用户都希望任务完成时间最小，这是一个典型的多目标优化问题(Multi-objective Optimization Problems, MOP)，并且 MOP 中的各个目标函数间可能是冲突的，一个目标函数的优化往往伴随着其它目标函数的劣化.

3 调度算法实现

进化计算是一种基于种群操作的优化技术，可以隐并行的搜索解空间中的多个解，并能利用不同解之间的相似性来提高其并发求解的效率，因此进化计算非常适合求解 MOP. 蚁群算法是一种基于种群的模拟进化算法，已经被成功的应用于各种领域来求解优化问题，如: 车辆路径问题[4]、Web 服务组合[5]、网格作业调度[6]等. 因此文本采用蚁群算法来实现系统的优化求解.

蚁群算法的关键是把实际问题转化为蚁群网络. 这里把资源中心的分配过程看作 m 个阶段，每个阶段分配一个资源中心提供下载服务，在 c_j 处分别设置 Max_{cj} 个蚂蚁，每个蚂蚁从 c_j 处转移到处 $u_1\sim u_n$，转移概率为[7]:

$$P_{ij}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}\cdot[\eta_{ij}(t)]^{\beta}}{\sum_{s\in allowed_k}^{n}[\tau_{is}(t)]^{\alpha}\cdot[\eta_{is}(t)]^{\beta}} & j\in allowed_k\\ 0 & otherwise\end{cases}$$

其中，$allowed_k$ 表示蚂蚁 k 可以访问的用户集合，每次循环将已经访问的用户从列表中剔除．$\tau_{ij}(t)$ 表示 t 时刻在 u_i 和 c_j 之间的信息素含量，η_{ij} 是指由 c_j 转移到 u_i 的启发程度，参数 α 表示残留信息素的权重，参数 β 表示启发信息的权重．初始时刻，各路径上的信息素量相等，设 $\tau_{ij}(0)=C$ (C 为常数)，η_{ij} 由某种启发式算法确定，本文所使用的启发式算法是:

$$\eta_{ij}=1/TM_{ui}$$

信息素的更新:

$$\tau_{ij}(t+\Delta t)=(1-\rho)\cdot\tau_{ij}(t)+Q\Big/\sum_{i=1}^{n}TM_{ui}$$

其中 ρ 挥发系数，取值范围在 0 到 1 的常数. Q 为一个常量.

具体算法步骤:

(1) 初始化 ρ 、Q、C、 α 、 β 的值;

(2) oc<--0(为外置循环次数);

(3) nc<--0(nc 为内置循环次数);

(4) 在 c_j (j=1, 2, ……m)处放出 Max_{cj} 个蚂蚁，每个蚂蚁按转移概率 P_{ij} 选择下一用户，要求: 同一中心的蚂蚁不能转移到同一个用户处，且某一用户处的蚂蚁数量不能超过用户连接资源中心的上限 Max_{ui}，修改 $allowed_k$ 列表;

(5) 计算本次分配的全部目标的值，比较出较好的结果，放入结果集中, nc<--nc+1;

(6) 若 nc>设定的循环次数，取较好结果按更新方程更新信息素强度，oc<--oc+1; 否则重置列表，跳转到(4);

(7) 若 oc>设定的循环次数，则输出目前最优解，否则转向(3).

4　仿真实验与结果分析

本文以“辽宁省基础教育资源网”中 3 个资源中心为 20 个用户提供资源下载服务为例进行仿真实验. 设定三台服务器最大连接数分别为 6, 8, 6; 每个用户下载的资源大小为 50M, 最大连接数为 10; 每个用户与服务器间的下载速度在 100k/s~900k/s 之间随机产生. 此外，蚁群算法中的常量初始化为: α=2, β=2, ρ=0.1, Q=1, C=0.1, 迭代次数为 200 代[8], 目标函数结果如图 2.

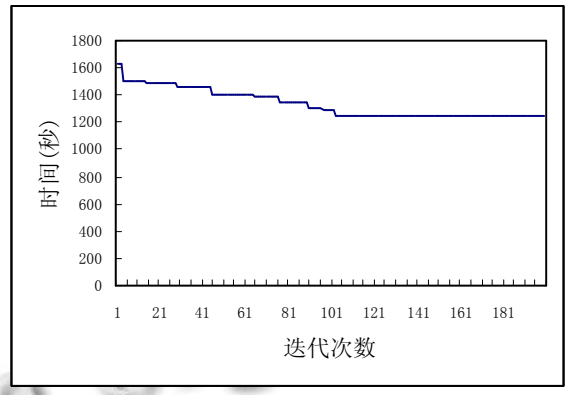

图 2　蚁群算法结果

从图 2 中可以看出，本文算法在第 106 代循环后找到较优解 1383.5 秒，具有很好的收敛性.

为了证明本算法的优化性能，与其他网络负载均衡算法[9](轮转法、随机法、最快响应法和加权轮转法)处理资源中心和资源用户间分配，进行了模拟对比实验，实验结果如图 3.

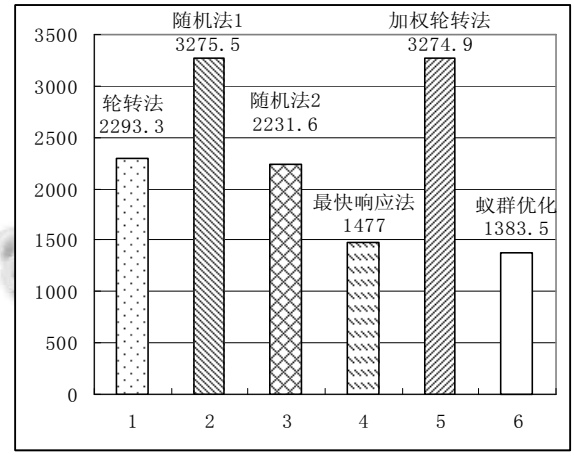

图 3　蚁群算法和其他算法结果对比

从图 3 中可知，基于蚁群算法调度的所有任务完成时间最少; 最快响应法次之; 轮转法和加权轮转法效果不好(因为适用于集群中所有节点的处理能力和性能均相同的情况); 随机法有的效果好，有的效果不好，性能可靠性低.

（下转第 75 页）