无中心式调度框架下网格作业的节点选择优化

王庆江, 徐建良
(中国海洋大学计算机科学系, 山东青岛 266071)

摘 要: 为优化无中心式调度框架下网格作业的节点选择, 提出了随机多起点爬山算法. 为使多个起点均匀分布于网格, 按随机选择邻居的重复次数的指数增长找出各起点. 为反映合理的用户调度需求, 用平均的并行计算能力加权的有界减慢率衡量节点选择. 灵活调整网格工作负荷, 对随机多起点爬山算法进行了全面评估. 在网格负载不是很轻情况下, 该算法能有效地在网格全局优化节点选择.

关键词: 计算网格; 无中心式调度框架; 节点选择; 随机多起点爬山算法
中图分类号: TP393 文献标识码: A 文章编号: 0372-2112 (2006) 08-1420-04

Optimization of Node Selections for Grid Jobs under De-Centralized Scheduling Frameworks

WANG Qing-jiang, XU Jian-liang
(Department of Computer Science, Ocean University of China, Qingdao, Shandong 266071, China)

Abstract To optimize node selections for grid jobs under de-centralized scheduling frameworks, the algorithm of random multi-start hill climbing is proposed. To make multiple start-points distributed uniformly in grids, each start-point is found according to exponential increase of iteration times of selecting neighbors randomly. To reflect reasonable scheduling requirements from users, the average bounded slowdown weighted by parallel computation capacity is used to evaluate node selections. By flexibly adjusting grid workload, the algorithm of random multi-start hill climbing is evaluated comprehensively. With grid load being not too light, the algorithm can effectively optimize node selections in the whole grid.

Key words computational grid; de-centralized scheduling framework; node selection; random multi-start hill climbing

1 引言

计算网格的作业调度可分为网格层次和本地层次[1]. 网格层次调度接收用户提交的作业, 为作业选择适合的节点. 本地层次调度负责节点内的作业队列管理, 为作业安排处理机资源. 网格层次调度可实现为一个或多个网格调度器, 本地层次调度一般是附加了网格调度器接口的本地调度器. 节点选择是网格层次调度的核心任务, 也是网格聚合分布资源、改善作业执行性能的关键.

在无中心式调度框架[2~5]中, 每个节点上部署一个网格调度器. 网格调度器收到作业时, 查询本地和相邻节点的作业负载或作业预期执行性能, 确定作业放在本地节点, 还是发给相邻节点. 文献 [2]根据节点间的网络性能和本地节点的作业队列状态, 在本地和附近节点中选择负载最轻的节点. 文献 [2]为作业迁移设立了门槛, 当迁移开销超过响应时间的降低幅度时, 作业不会迁移, 但所提算法只能实现在提交点附近搜寻更适合节点. 文献 [3]中调度框架呈树状, 并规定上层节点的计算能力不小于下层节点, 如果本地和兄弟节点的计算能力都不满足作业需求, 则作业交给父节点, 重复此过程, 直到节点计算能力满足作业需求为止. 文献 [3]只考虑寻找能满足作业执行需求的节点, 不考虑实现网格全局的节点选择优化, 且节点组织结构呈树状, 故节点搜寻策略不适用一般的无中心式调度框架. 文献 [4]提出了从本地和相邻节点中寻找最优节点的一组负载平衡算法, 但没有进一步研究在整个网格中如何优化节点选择. 文献 [5]将每个节点及其邻居看成子网格, 提出子网格中的动态调度方法, 即动态优化作业的节点选择, 但没有研究作业提交时如何在网格全局优化节点选择.

收稿日期: 2005-11-30; 修回日期: 2006-03-27

从目前可找到的文献看, 在一般的无中心式调度框架下, 作业提交时节点选择在网格全局的优化还是个未解决的问题. 这里提出随机多起点 (最陡)爬山算法, 该算法在整个网格中按随机选择邻居的重复次数的指数增长选择多个起点, 分别从这些起点用 (最陡)爬山法搜索多个网格局部, 进而实现节点选择在整个网格中的优化.

2　节点选择问题

无中心式调度框架有很好的健壮性和可扩展性, 其主要特征是每个网格调度器只负责在本地节点和相邻节点中优化作业的节点选择, 因而更健壮、更可扩展. 如图 1 作业 1 提交到 A_2, 通过迁移最终指派到 A_8. 作业 2 提交到 A_4, 最终指派到 A_7. 这里的作业迁移是作业提交后立即发生的, 而不是节点本地队列中等待作业的动态迁移.

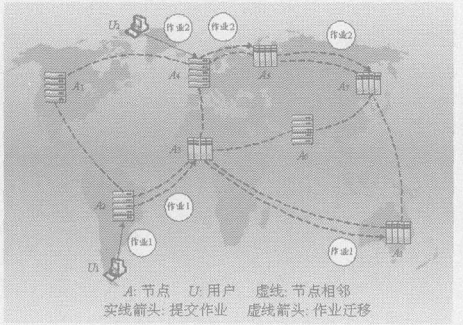

图 1　无中心式调度框架

若 J_i 的提交时刻、在节点 A_k 上的开始运行时刻和运行时间分别为 t_i^a、t_{ik}^s和 t_{ik}^e, 则响应时间 $t_{ik}^r = t_{ik}^s - t_i^a + t_{ik}^e$. 响应时间和运行时间的比值称作作业的减慢率[6], 减慢率被广泛用于评估节点的本地调度. 然而, 网格节点的并行计算能力往往不等, 在不同节点上作业的运行时间不同, 作业响应时间和减慢率的变化可能不一致, 即响应时间缩短时减慢率未必下降, 故平均减慢率不能用于衡量网格中的节点选择.

节点 A_k 的并行计算能力可用一些测试软件 (如 NPB[7])获得, 令为 c_k. J_i 在 A_k 上的运行时间为 t_{ik}^e, 不妨假设 t_{ik}^e与 c_k 成反比. $c_k t_{ik}^e$称作 J_i 的纯运行时间, 其含义是不依赖指派的"纯粹 "的并行计算资源需求.

假设 A_k 的并行计算能力为 c_k, J_i 在 A_k 上的运行时间和响应时间分别为 t_{ik}^e和 t_{ik}^r, 则$\frac{t_{ik}^r}{c_k t_{ik}^e}$称作 J_i 在 A_k 上的并行计算能力加权的减慢率.

纯运行时间越短, 即作业"纯粹 "的计算资源需求越少, 从用户角度看, 作业的响应时间应越短. 显然, 作业响应时间与并行计算能力加权的减慢率的变化一致, 故并行计算能力加权的减慢率合理表达了用户的节点选择需求, 适合用于衡量节点选择.

纯运行时间很短的作业对平均的并行计算能力加权的减慢率有过大的贡献, 故提出 J_i 在 A_k 上的并行计算能力加权的有界减慢率 d_{ik}, 如式 (1)

$$d_{ik} = \frac{t_{ik}^s - t_i^a + t_{ik}^e}{\max\{c_k t_{ik}^e,\ t\}} \quad (1)$$

这里, t是纯运行时间是否很短的门槛值. 平均的并行计算能力加权的有界减慢率记为 $\bar{d}$, $\bar{d}$更适用于衡量节点选择的有效性.

若一段时期里提交给网格的作业依次为 $\{J_l, J_{l+1}, \cdots, J_m\}$, 网格节点分别为 A_1、A_2、…、A_N. 节点选择问题就是将 J_i ($i \in [l, m]$)指派到 $\{A_1, A_2, \cdots, A_N\}$中的哪个节点, 才能尽量降低 $\bar{d}$.

3　随机多起点 (最陡)爬山法

并行计算能力加权的有界减慢率与响应时间的变化一致, 故可按最短响应时间选择节点. 在无中心式调度框架中, 没有维护网格全局信息的"服务器 ", 故网格调度器只能沿着节点相邻关系搜寻适合的节点. (最陡)爬山法可实现这一过程, 但只能搜寻网格局部. 为此, 在网格全局随机选择若干个节点, 分别以它们为起点, 用 (最陡)爬山法搜寻适合作业的节点, 然后在这些局部解中找出最优解, 这种方法可称作随机多起点 (最陡)爬山法, 记为 RMSHC (Random Multi-Start Hill Climbing)或 RMSMSHC (Random Multi-Start Most Steep Hill Climbing).

当网格规模很大时, 起点数应适当增加. 此外, 这些起点应分布在尽可能多的优化收敛域, 这是随机多起点 (最陡)爬山法实现在网格全局优化节点选择的关键. 为此, 这里提出按随机选择邻居的重复次数的指数增长选择各起点.

若 J_i 被提交到 A_j, 用 x 个起点的随机多地点 (最陡)爬山法为 J_i 选择节点, 步骤如下. 以响应时间预测为基本操作, 则该算法的复杂度为 O (x ×N).

(1)用 S 表示局部解集合, 并令 S = Φ;

(2)重复 x 次执行下面的操作;

(a)令 $A_y = A_j$, 在第 k 次时, 重复 2^k 次执行: 随机选择 A_y 的一个邻居 A_n, 令 y = n.

(b)当采用爬山法时, 若存在 A_y 的一个邻居 A_x, 可使 J_i 的响应时间更短, 则令 y = x, 并重复本步骤; 否则, $S = S \cup A_y$;

(c)当采用最陡爬山法时, 在 A_y 的邻居中, 若邻居 A_x 上 J_i 的响应时间最短, 且比 A_y 上的响应时间短, 则令 y = x, 并重复本步骤; 否则, $S = S \cup A_y$.

(3)在 S 中, 按最短响应时间为 J_i 选择一个节点, 即为最适合 J_i 的节点.

假设节点共 16个, 初始的调度框架只包含 A_1, 其他节点依次加入. 每个节点加入时选择一个节点与之相邻, 得最后的调度框架如图 2 这里, 每个节点加入时没有选择与更多节点相邻, 目的是要突出调度框架的无中心性.

若作业提交到 A_{12}, 第 k 次寻找起点时, 各节点被作为起点的概率如表 1 由表 1, 随着 k 的增大, 有更多的节点可

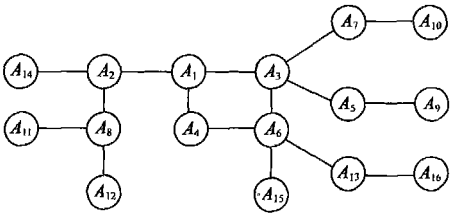

图2 无中心式调度框架

能成为起点,且这些可能的起点分布在网格全局.因此,按随机选择邻居的重复次数的指数增长确定各起点,可使起点分布于尽可能多的优化收敛域,使随机多起点(最陡)爬山算法有可能实现节点选择在网格全局的优化.

表1 第k次寻找时,各节点被作为起点的概率

	A_2	A_3	A_4	A_9	A_{10}	A_{11}	A_{12}	A_{13}	A_{15}
k=1	0.33	0	0	0	0	0.33	0.33	0	0
k=2	0.41	0.04	0.04	0	0	0.26	0.26	0	0
k=3	0.39	0.10	0.08	0.01	0.01	0.19	0.19	0.02	0.02

注:其他节点被作为起点的概率为0.

4 模拟实验

4.1 模拟系统

JSSS[8]是基于离散事件的调度模拟系统.离散事件有三种,即作业提交、开始运行和完成.设置虚拟时钟,每个时刻检查是否有事件发生.事件发生时,执行作业调度或完成后处理.通过模块化的实现,JSSS可灵活模拟网络拓扑、资源相对性能、调度框架、网格层次和本地层次调度、网格工作负荷(workload)等.在JSSS上,可对随机多起点(最陡)爬山算法进行全面评估.

假设调度框架如图2,各节点的处理机数均为128,本地调度采用装填法[6].不同节点可有不同的并行计算能力,不妨令$c_k = 1/k, k \in [1, 16]$.

模拟系统所需的网格工作负荷由模型产生,模型基于并行计算机工作负荷模型[9]构建.作业并行度服从两阶段均匀分布,纯运行时间符合超伽玛分布,到达间隔服从伽玛分布.为缩短模拟实验的周期,作业的纯运行时间需要缩短.若模型产生的J_i纯运行时间为l_i^0,则实际的J_i纯运行时间(l_i)缩短为$1.5^{\ln l_i^0}$.网格负载通过在到达间隔上乘以一个因子来调整,该因子称作到达间隔扩张因子,记为s.s越大,网格负载越轻.

若一个网格调度器收到用户提交的作业J_i,则该网格调度器称作J_i的提交点.为表示提交点分布的不均匀性,假设节点的随机序列为($A_{14}, A_7, A_{13}, A_{15}, A_4, A_6, A_5, A_9, A_8, A_{11}, A_{16}, A_{12}, A_3, A_{10}, A_1, A_2$),提交点服从比率为α的几何分布.如提交到$A_{14}$的概率为q,则提交到$A_7$的概率为$q \times \alpha$,提交到$A_{13}$的概率为$q \times \alpha^2$,依次类推.α=1时,提交点服从均匀分布;α越大,分布越不均匀.

若J_i的纯运行时间为l_i,纯运行时间估计为$l_i \times e(J_i)$,则$e(J_i)$可称作J_i纯运行时间的估计因子.假设$e(J_i)$服从区间$[E_1, E_2]$上的均匀分布.E_2越大,l_i的高估程度可能越大;$E_1 < 1$时,l_i可能被低估.这里只考虑各作业顺利完成的情况,故令$E_1 = 1$,通过调整E_2模拟作业纯运行时间高估的程度.

4.2 结果分析

作业提交到哪个节点上的网格调度器,便在那个节点上运行,这时没有节点选择优化,记为W;用爬山法和最陡爬山法优化节点选择分别记为H和M.令t=10秒,计算J_{300}至J_{799}上的$\bar{d}$,用$\bar{d}$衡量节点选择方法的性能.

首先验证扩张因子s与网格负载的关系.假设$\bar{L}$表示自提交J_{300}到这500个作业全部完成期间节点本地队列长度的平均值,$\bar{L}$越大表示多数节点的负载越重,即网格负载越重.令提交点均匀分布,s对$\bar{L}$的影响如图3,三种节点选择的性能比较如图4.

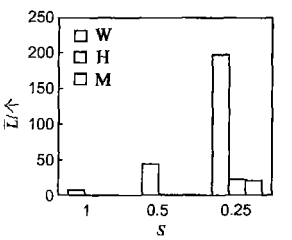

图3 扩张因子对节点本地队列平均长度的影响

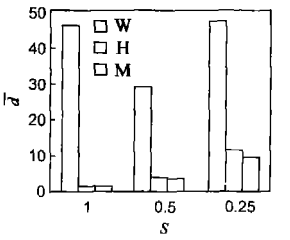

图4 扩张因子不同时三种节点选择的性能比较

由图3,对任一种节点选择,s变小时$\bar{L}$变大,即网格负载变重.对于方法W,并行计算能力较差的节点上有较多的作业等待,故$\bar{L}$较大;而对于方法H或M,作业可被指派到提交点附近最适合的节点,故$\bar{L}$较小.也就是说,对于相同的s,方法H或M下的网格负载比方法W下的网格负载轻.

由图4,对于同样的s,没有节点选择优化(即方法W)时,$\bar{d}$最大,而采用了节点选择优化(如方法H或M)时,$\bar{d}$明显降低.与网格负载较轻相比,负载较重时方法H或M的性能会变差,原因是节点上作业数目变大,作业等待时间普遍变长.

令s=0.25,并选择不同的α,方法O表示在网格全局为作业选择最佳节点,则方法H、M、RMSHC、RMSMSHC和O的性能比较如图5.

由图5,提交点分布越不均匀,方法H的性能越差,原因是方法H只在网格局部优化节点选择;方法M的性能基本稳定,主要原因是网格规模不够大;网格全局最优不受提交点分布的影响,故方法O的性能完全稳定.此外,RMSHC和RMSMSHC可取得比方法H或M更好的性能,进一

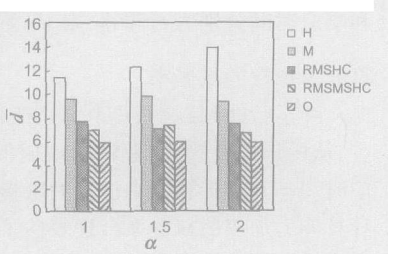

图5 提交点分布不同时五种节点选择的性能

步减小提交点分布对调度性能的影响, 这说明 RMSHC 和 RMSMSHC 一定程度上实现了网格全局的节点选择优化.

令 s= 0.25, α= 1.5. 选择不同的起点数 x, RMSHC、RMSMSHC 和方法 O 的性能比较如图 6, 纯运行时间高估对 RMSHC 和 RMSMSHC 性能的影响如图 7.

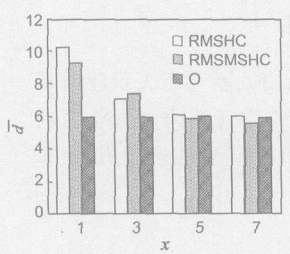

图 6　起点数不同时, 三种节点选择的性能比较

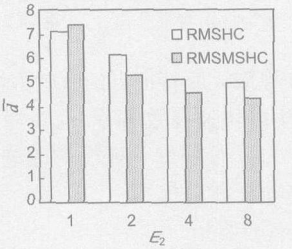

图 7　纯运行时间高估对随机多起点(最陡)爬山法性能的影响

由图 6, x 增大时, RMSHC 和 RMSMSHC 的性能逐步改善. x ≥5时, RMSHC 和 RMSMSHC 的性能趋近方法 O, 这表明 RMSHC 和 RMSMSHC 实现了节点选择在网格全局的优化. x = 7 时, RMSMSHC 的性能比方法 O 略好, 原因是 RMSMSHC 和方法 O 都用于作业提交时立即为之选择节点, 方法 O 在优化一个作业的节点选择时, 可能限制后面作业的节点选择优化.

由图 7, 纯运行时间高估较多时, 随机多起点(最陡)爬山法的性能会得到更大改善, 原因是本地调度(装填法)可在运行时间高估情况下改善作业调度[6].

这里的实验没有考虑调度开销, 而网络延迟及调度算法复杂度会拖延作业到节点的指派. 当网格负载较轻, 即作业等待时间很短时, 调度开销会一定程度上抵消作业迁移带来的好处; 当网格负载趋重时, 作业平均等待时间较长, 新提交作业的调度与其他作业的等待或运行大部分时间重叠, 这时, 调度开销对调度效果的影响逐渐可以忽略. 因此, 在实际中本算法在网格负载不是很轻的情况下更有效.

5　结论

节点选择是改善网格作业执行性能的关键. 为实现作业提交时在整个网格中的节点选择优化, 这里提出了随机多起点爬山算法, 该算法按随机选择邻居的重复次数的指数增长选择各起点, 从而可在更多的优化收敛域搜索局部解. 在网格负载不是很轻情况下, 只要起点数足够, 随机多起点(最陡)爬山法可在整个网格中优化作业的节点选择.

下一步将研究在预测响应时间时考虑已产生的调度开销, 即在算法中引入调度开销, 这会提高算法在网格轻载时的可用性. 同时, 研究降低调度开销的方法, 如只从提交点向最适合节点迁移作业, 搜寻节点时只传输作业轮廓信息(即并行度、纯运行时间等), 这可降低网络传输需求, 将多起点的爬山搜索并行化, 也可进一步降低调度开销.

参考文献:

[1] Schwiegelshohn U, Yahyapour R. Attributes for communication between scheduling instances[EB/OL]. http://www.gridforum.org/Documents/GFD/GFD-I-6.pdf, 2004-10-22

[2] Arora M, Das S K, Biswas R. A de-centralized scheduling and load balancing algorithm for heterogeneous grid environments[A]. Parallel Processing Workshops[C]. Washington: IEEE Computer Society, 2002. 499-505.

[3] Cao Junwei, Spooner D P, Jarvis S A, et al. Agent-based grid load balancing using performance-driven task scheduling[A]. Parallel and Distributed Processing Symposium[C]. Washington: IEEE Computer Society, 2003. 49-58

[4] Shan Hongzhang, Leonid O, Rupak B. Job superscheduler architecture and performance in computational grid environments[A]. Supercomputing[C]. Washington: IEEE Computer Society, 2003. 44-58

[5] Wang Qingjiang, Gui Xiaolin, Zheng Shouqi. De-centralized job scheduling on computational grids using distributed backfilling[A]. Grid and Cooperative Computing[C]. Berlin: Springer-Verlag, 2004. 285-292.

[6] Mu'alem A W, Feitelson D G. Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling[J]. IEEE Transactions on Parallel and Distributed Systems, 2001, 12(6): 529-543.

[7] Wolski R, Spring N, Hayes J. The network weather service: A distributed resource performance forecasting service for metacomputing[J]. Journal of Future Generation Computing Systems, 1999, 15(5/6): 757-756.

[8] 王庆江. 计算网格无中心式调度框架下的作业调度方法研究[D]. 西安: 西安交通大学, 2005.
Wang Qingjiang. Study on methods of job scheduling under de-centralized scheduling frameworks in computational grids[D]. Xi'an: Xi'an Jiaotong University, 2005. (in Chinese)

[9] Lublin U, Feitelson D G. The workload on parallel supercomputers: Modeling the characteristics of rigid jobs[EB/OL]. http://citeseer.nj.nec.com/lublin01workload.html, 2005-03-14

作者简介:

王庆江　男, 1968年出生于山东高青, 博士, 中国海洋大学计算机科学系讲师. 研究方向为高性能计算、计算网格中的关键技术等. E-mail: qjwang@ouc.edu.cn

徐建良　男, 1969年生, 博士, 中国海洋大学计算机科学系副主任、教授. 研究方向为并行计算理论.