一种针对结构化并行控制机制的任务调度算法*

张宏莉，方滨兴，胡铭曾

(哈尔滨工业大学 计算机科学与工程系，黑龙江 哈尔滨 150001)

E mail: zhl@pact518.hit.edu.cn

http://www.hit.edu.cn

摘要：缩短程序的执行时间是并行处理的首要目标，有效的任务分配算法是实现这一目标的关键，对机群系统来说更是如此.研究机群系统上针对结构化并行控制机制的任务调度问题，并基于贪心算法、粒度控制、反馈式分派的原则，提出近优的任务调度算法 SSA(sub-optimal scheduling algorithm).实验结果表明，在机群环境下，该算法的并行计算性能与其他算法相比均有所提高.

关键词：任务调度；并行处理；并行编译；机群系统

中图法分类号：TP315 **文献标识码**：A

20世纪70年代以来，针对不同时期硬件环境的特点，人们在任务调度方面开展了大量的研究工作，主要分为3类.第1类，静态调度算法，如文献[1,2]中以最长路径长度为优先级的表调度方法.其优点是运行时调度开销较小，可忽略不计，适合于大规模并行机MPP等较为理想的并行计算环境.第2类，启发式调度算法，如文献[3]中基于branch-and-bound算法、文献[4]中基于任务聚类的算法、文献[5]中基于任务复制的算法等.这是人们在异构环境下探索最优解的重要结果，一般算法的复杂度较高，如文献[3]的复杂度为$O(M^3N^2)$(M为任务个数，N为处理机个数)，具有一定的调度开销.第3类，动态调度算法，又称动态负载平衡，如文献[6]中的双向启动自适应任务分配策略，另外还有轻者启动任务分配策略、重者发送任务分配策略等.这些能够适合于负载变化比较剧烈的多用户共享的网络计算环境.表1对三者进行了比较.

Table 1 Comparison of scheduling algorithms

表1 各种调度算法的比较

	Suitable hardware platform①	Complexity②	Existed algorithms③
Static scheduling algorithm④	Low latency, single user⑤	Less complex⑥	References⑦[1,2]
Heuristic scheduling algorithm⑧	Heterogeneous, high latency, single user⑨	More complex⑩	References [3~5]
Dynamic scheduling algorithm⑪	Heterogeneous, high latency, multiple users⑫	Complex⑬	Reference [6]

①适合的硬件环境，②算法复杂度，③已有算法，④静态调度算法，⑤通信延迟小，独占，⑥复杂度较小，⑦文献，⑧启发式调度算法，⑨异构，高延迟，独占，⑩复杂度较大，⑪动态调度算法，⑫异构，高延迟，多用户，⑬复杂度大.

工作站机群系统具有较好的性能价格比和可扩展性，成为人们最易于得到的并行硬件资源之一.同时，它也存在着不可忽视的缺陷：异构、高通信延迟.这里，我们试图针对此类硬件环境，研究一种复杂度较低、调度开销较小的高效任务调度算法.

* **收稿日期**：1999-12-14；**修改日期**：2000-03-23

基金项目：国家"九五"国防预研基金资助项目(16.1.3.3)

作者简介：张宏莉(1973—)，女，吉林榆树人，博士，讲师，主要研究领域为并行计算，机群网络；方滨兴(1960—)，男，江西万年人，博士，教授，博士生导师，主要研究领域为并行处理，计算机网络；胡铭曾(1935—)，男，江苏江阴人，教授，博士生导师，主要研究领域为并行体系结构，并行处理.

1 结构化与非结构化并行控制机制

纵观各类并行语言,其并行控制机制可归纳为两种.这里,我们沿用串行程序中关于结构化"单入口,单出口"的定义,将它们定义为结构化和非结构化并行控制机制.

定义 1(任务流图 TASK GRAGH). 一个任务流图 TG 是一个二元组,$TG=(V,E)$,其中 V 是程序中所有任务的集合,E 是所有任务间依赖关系的集合.

定义 2(结构化并行控制机制).

(1) 由有且只有一个共同前驱和共同后继的两个或两个以上任务构成的并行控制机制,是结构化并行控制机制.

(2) 由有且只有一个共同前驱和共同后继的两个或两个以上任务或结构化并行控制机制构成的并行控制机制,是结构化并行控制机制.

结构化并行控制机制的特点是同步性强,并发结构与其后的串行或并行部分之间存在明显的依赖关系,需按一定的先后次序执行,如图 1(a)和(b)所示.通常,这种结构在结束前有一个栅障同步点.CC++[7]中的 par,parfor 结构,Multi-PASCAL[8]中的 cobegin,coend 结构,HPF[9]中的 DOALL 结构等典型多分支并发结构都属于结构化并行控制机制.

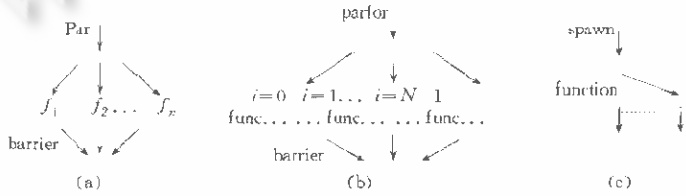

Fig. 1 Parallel control structures
图1 并行控制机制

定义 3(非结构化并行控制机制). 由共同后继结点为 0 的两个或两个以上任务构成的并行控制机制,是非结构化并行控制机制.

这类结构中又以双分支并行控制机制最为常见,如图 1(c)所示.CC++[7]中的 SPAWN、远程过程调用等均属于此类.

本文主要探讨工作站机群系统上针对结构化多分支并行控制机制的任务调度算法.

2 算法 SSA

工作站机群上针对结构化多分支并行控制机制的任务调度问题描述如下:

已知任务集合 $T=\{T_i,0\leqslant i\leqslant M-1\}$,处理机集合 $P=\{P_i,0\leqslant i\leqslant N-1\}$,且已知任务 T_i 的计算量为 $T(i)$,任务 T_i 与 T_j 间的通信量为 $C(i,j)$(通过编译阶段的静态分析及运行时值的代入获得).求一种 $T\to P$ 的分配方案使下式值较小,其中 exc[i]为处理机 P_i 的计算开销,comm[i,j]为处理机 P_i 与 P_j 间的通信开销.

$$\max_{0\leqslant i\leqslant N-1}\{\exp[i]\}+\sum_{0\leqslant i\leqslant N-1}\sum_{0\leqslant j\leqslant N-1}\mathrm{comm}[i,j]$$

首先,我们注意到在工作站机群系统上进行任务调度需坚持以下原则:

(1) 根据实际运行效果,考虑到机群通信延迟较长的特点,在分配前进行任务合并.也就是说,将通信量大于计算量的任务 T_i 与 T_0 合并.由加速比的简单计算公式可推导出它的必要性:如果有

p 个处理器参与并行,每个处理器的计算时间为 T_p,总的通信时间为 T_c,则在负载平衡的情况下,p 个处理器所能得到的并行加速比为

$$S_p=\frac{p*T_p}{T_p+T_c}=\frac{p}{1+T_c/T_p}.$$

当 $T_c\gg T_p$ 时,$SP<1$.

(2) 为易于调节负载,首先根据贪心算法,将任务按由大到小的顺序分派出去;然后,考虑到硬件的异构性,在分配之初为保证各处理机尽早开工,将按通信量升序排列的 $N-1$ 个任务依次分配到通信速度按升序排列的处理机上.

(3) 考虑到任务计算量估算的不准确性和相关性给任务执行时间带来的不确定性,采取反馈式调度:主机分配任务给从机,从机响应执行主机分来的任务,并于执行完毕后向主机发送反馈信息,主机接到反馈信息后再分配、再反馈...直到所有任务分配完毕为止.也就是说,改"一次分完"的静态策略为"多次分配"的动态策略.

(4) 每次分配的任务量 B 根据任务总数和处理机个数而定.B 的大小对整个调度性能影响很大,若太大则会降低负载的调整能力;若太小则会导致调度及通信开销过大,如图 2 所示.我们设

$$B=\frac{\text{任务总数}}{a^*\text{处理机总数}},$$

其中 a 为每台处理机的最低分配次数(通常取 3).当任务的平均通信开销与近平均计算开销相近时,是不宜频繁调度的,这时,$B=$任务总数/(a^*处理机个数);当任务大小相等、粒度较粗时,调整余地较大,这时,B/单个任务的粒度可达到 1.

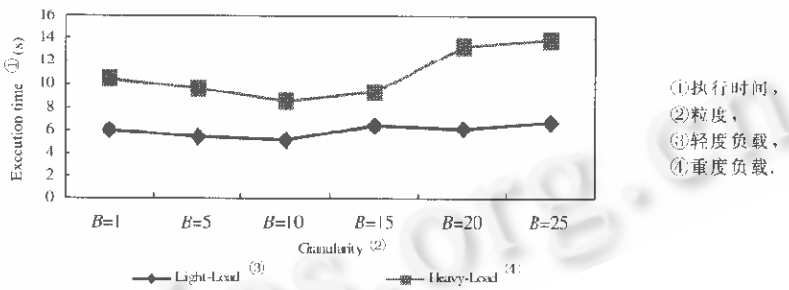

Fig. 2 Relationship of granularity and parallel performance

图 2 粒度与并行性能的关系

基于以上原则,本文得到异构机群系统上针对结构化多分支并行控制机制的调度算法 SSA 如下:

(1) 首先,测定各处理机的计算速度(秒/指令)以及主机到各从机的通信速度(秒/字节),得到两个参数数组 $x[i]$,$c[i]$;

(2) 进行任务合并,将通信量 $C(0,i)$ 和计算量 $T(i)$ 的任务 T_i 与 T_0 合并,然后分配任务 T_0 到主机 P_0;

(3) 选择 $N-1$ 个 $C(0,i)$ 较小的任务按升序依次分配到通信速度按降序排列的处理机 $\{P_i,i>0\}$ 上,保证除 P_0 之外的其他处理机尽早开工;

(4) 选择 N 个 $C(i,M-1)$ 且 $T(i)$ 较小的任务留至最后分配,以保证最后分配的任务带来的延迟最小,尽早收工;

(5) 根据贪心算法,将所有剩余任务按 $T(i)$ 由大到小的顺序排列,求出 B,再将任务按组分配给最早反馈回结束信号的处理机;

(6) 按 $C(i,M-1)$ 由大到小的顺序,将(4)中的 N 个任务分给最早反馈回结束信号的处理机;

(7) 最后将 T_{M-1} 分配给主处理机 P_0.

综上所述,不难看出,当任务数远远大于处理机数时,步骤(2)和步骤(5)将发挥主要作用;当任务数与处理机数相当时,步骤(3)、(4)、(6)将起到一定的调节作用.

该算法的复杂度为$O(M^*N^*\log(M))$,明显低于启发式算法.

3 性能对照实验

3.1 异构环境下的性能对照

在由4台POWER PC组成的工作站机群系统上,我们分别针对任务粒度$M=100M,10M,1M,10K$,在处理机处于不同负载状态的情况下,对SSA与静态算法[10]、动态算法(轻者启动)做了比照实验.当处理机处于轻度负载(4台CPU利用率小于10%)、中度负载(1台CPU利用率50%左右,3台轻度负载)和重度负载(3台轻度负载,1台CPU利用率大于80%)时,并行程序运行时间见表2.由此可以看出,在同构环境下,SSA与静态算法性能相近,说明SSA的调度开销与静态算法相近;异构环境下SSA较动态算法有10%~50%的提高.

Table 2 Comparison of SSA, static algorithm and dynamic algorithm

表2 SSA与静态算法、动态算法的性能比较

Granularity①	Light-Loaded②			Middle-Loaded③			Heavy-Loaded④		
	SSA	Static algorithm⑤	Dynamic algorithm⑥	SSA	Static algorithm	Dynamic algorithm	SSA	Static algorithm	Dynamic algorithm
10^8	169.28	169.32	169.31	153.36	170.56	196.53	206.17	340.22	326.48
10^7	17.01	17.09	21.02	14.5	17.84	39	31.95	33.02	33.62
10^6	1.84	1.97	3.14	2.11	2.69	3.27	3.24	3.85	4.12
10^4	0.174	0.154	0.194	0.194	0.15	0.19	—	—	—

①任务粒度,②轻度负载,③中度负载,④重度负载,⑤静态算法,⑥动态算法.

3.2 b值比较

在由3台POWER PC组成的工作站机群系统上,我们运行了3个典型程序(设$b=B/$单个任务的粒度),其数据量及系统计算的b_{SSA}如下:

(1) Sort(16000):对16 000个元素的数组进行归并排序,任务数为32,$b_{SSA}=\lfloor 32/(3*3)\rfloor=3$;

(2) Fibonacci(1000000):求1 000 000个Fibonacci数,任务数为50,$b_{SSA}=\lfloor 50/(3*3)\rfloor=5$;

(3) Matrix(500*500):两个500*500的矩阵乘积,任务数为100,$b_{SSA}=\lfloor 100/(3*3)\rfloor=11$.

图3列出了不同b值下的程序运行时间,结果表明,SSA确定的b值优于其他点.

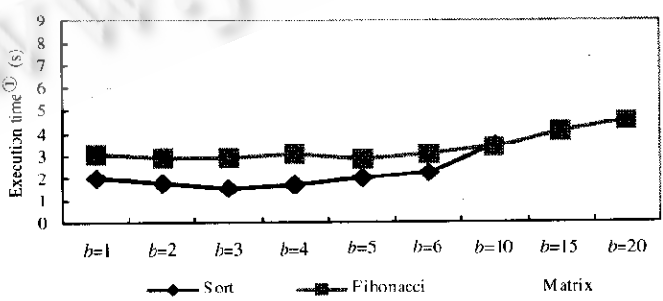

Fig. 3 The execution time of typical problems

图3 典型问题的实验结果

4 小 结

本着"知己(任务图)知彼(处理机)"的思想,本文研究了异构机群系统上结构化并行控制机制的任务调度算法,基于贪心算法、粒度控制、反馈式分派的原则,提出任务调度算法 SSA. 实验表明,在异构环境下,其性能与其他算法相比有 10%～50%的提高.

References:

[1] El-Rewini, H., Lewis, T.G. Scheduling parallel program tasks onto arbitrary target machines. Journal Parallel and Distributed Computing, 1990,9(2):138～153.

[2] Bokhari, S.H. A shortest tree algorithm for optimal assignments across space and time in a distributed processor system. IEEE Transactions on Software Engineering, 1981,7(6):583～589.

[3] Chern, M.S., Liu, P. An LC branch-and-bound algorithm for module assignment problem. Information Processing Letters, 1989,32(2):61～71.

[4] Yang, T., Gerasoulis, A. DSC: scheduling parallel tasks on an unbounded number of processors. IEEE Transactions on Parallel and Distributed Systems, 1994,5(9):951～967.

[5] Darbha, S., Agrawal, D.P. Optimal scheduling algorithm for distributed-memory machines. IEEE Transactions on Parallel and Distributed Systems, 1998,9(1):87～95.

[6] Mao, Guo-jun, Yang, Ming-sheng, Wang, Xiu-kun, *et al*. An adaptive co-direction starting task scheduling algorithm in distributed systems. Chinese Journal of Computers, 1996,19(7):514～519 (in Chinese).

[7] Foster, I. Compositional parallel programming languages. ACM Transactions on Programming Languages and Systems, 1996,18(4):454～477.

[8] Liu, Hong-wei, Li, Xiao-ming. Multi-Pascal: an effective research tool for parallel programming. Technical Report, PACT-TR-94-025, Department of Computer Science and Engineering, Harbin Institute of Technology, 1994 (in Chinese).

[9] Koelbel, C., Loveman, D., Schreiber, R., *et al*. The High Performance Fortran Handbook. Cambridge, MA: MIT Press, 1994.

[10] Bodin, F., Beckman, P., Gannon, D.B. Distributed pC++: basic ideas for an object parallel language. In: Proceedings of Supercomputing. 1991. 273～282. http://www.extreme.indiana.edu/sage.

附中文参考文献:

[6] 毛国君,杨名生,王秀坤,等.分布式系统中的双向启动自适应任务分配算法.计算机学报,1996,19(7):514～519.

[8] 刘宏伟,李晓明.Multi-Pascal:一个经济有效的并行程序设计研究工具.科技报告,PACT-TR-94-025,哈尔滨工业大学计算机科学与工程系,1994.

An Algorithm on Task Scheduling in Structural Parallel Control Mechanism*

ZHANG Hong-li, FANG Bin-xing, HU Ming-zeng

(*Department of Computer Science and Technology, Harbin Institute of Technology, Harbin* 150001, *China*)

E-mail: zhl@pact518.hit.edu.cn

http://www.hit.edu.cn

Abstract: Reducing the execution time of program is a main goal of parallel processing, while an efficient task allocation algorithm is a crucial step, especially on NOW (network of workstations). In this paper, the problem of task scheduling in structural parallel control mechanism on heterogeneous net of workstations is studied. Based on principles of greedy algorithm, granularity controlling and feeding-back-liked assignment, an algorithm on scheduling tasks is put forward under structural parallel mechanism: SSA (sub-optimal scheduling algorithm). The experimental results show that it can get better performance than other algorithms on heterogeneous NOW.

Key words: task scheduling; parallel processing; parallel compiling; NOW (network of workstations)

* Received December 14, 1999; accepted March 23, 2000

Supported by the Defence Pre-Research Project of the 'Ninth Five-Year Plan' of China under Grant No. 16.1.3.3