

多核处理器中任务调度与负载均衡的研究

彭蔓蔓, 黄 亮

(湖南大学 计算机与通信学院, 湖南 长沙 410082)

摘 要: 针对多核处理器系统的特点, 对任务分配及调度模型进行改进, 提高各处理器相对均衡负载度, 并在此基础上提出一种均衡种群遗传算法(BPGA). 算法在任务节点的高度约束条件下, 达到任务节点在处理器上随机分配, 而任务节点数均衡分配. 采用随机生成图法进行模拟实验, 与其他算法相比, BPGA 算法有更小的调度长度和较少的执行时间.

关键词: 多核处理器系统; 任务调度; 负载均衡; 遗传算法

中图分类号: TP312

文献标识码: A

文章编号: 1000-7180(2011)11-0035-05

Task Allocation and Load Balance on Multi-core Processors

PENG Man-man, HUANG Liang

(Department of Computer & Communication, Hunan University, Changsha 410082, China)

Abstract: According to the characteristics of multi-core processor system, this paper improves task allocation and scheduling model to improve the relative balance of the load on the processors, and puts forward a balanced population genetic algorithm (BPGA). The algorithm under the constraints of the height of task nodes can allocate task nodes randomly and the number of nodes balanced on each processing core. The experiment used randomly generated DAG graph, compared with other algorithms, the BPGA has less makespan and less execution time.

Key words: multi-core processors system; task scheduling; load balancing; genetic algorithm

1 引言

随着多核处理器在大规模并行系统中的应用, 计算性能的提高将更多地依赖于处理器核的数量增加. 为了充分地利用这些数量庞大的处理器核, 应用程序的进程/线程个数也将比过去多得多. 同时, 多核处理器也使得系统层次结构变得更为复杂.

任务分配和调度问题通常被认为是 NP-complete 问题^[1]. 任务调度通常分为静态调度和动态调度; 在动态调度中, 事先无法知道任务执行的信息, 根据当前的任务运行情况对任务进行调度; 而在静态调度中, 调度方法安排在任务运行之前且无法改变, 将一串任务集合分配到指定数目的处理器上, 达到最优的参考标准, 例如使得调度长度最小, 通信代价最低及 CPU 的利用效率最大化等. 目前, 已有许

多研究者提出了大量的任务调度算法, 其中主要分为启发式算法和随机搜索算法, 启发式算法具有较高的精确性, 但随着问题规模的扩大将会耗费大量的时间, 典型的如表调度算法; 随机搜索算法是近来调度研究中比较青睐的方法, 它具有快速搜索次优解的特性和执行效率, 典型的如遗传算法.

文中主要研究任务的静态调度. 首先对文献[2-3]中任务分配模型进行改进, 综合考虑复合节点平均执行时间和进程间通信代价两个因素, 并充分考虑任务分配的负载均衡性; 其次, 在每个任务子集分配对应的处理器后, 调用均衡种群遗传算法 (Balanced Population Genetic Algorithm, BPGA), 该算法将遗传算法种群采用均衡构造, 有效地提高了初始种群的质量, 做到任务节点随机分配, 进一步提高了处理器间的负载均衡度, 为遗传算法提供了一个

收稿日期: 2011-01-18; 修回日期: 2011-02-15

基金项目: 国家自然科学基金项目(60973030)

较好的迭代起点. 实验结果表明, BPGA 算法优于其他同类算法.

2 任务分配模型

2.1 问题定义

假设一个多核处理器并行系统由 N_{proc} 个处理器 $P_0, P_1, \dots, P_{\text{proc}-1}$ 组成, 并且每个处理器内有 N_{core} 个内核 $C_0, C_1, \dots, C_{N_{\text{core}}-1}$; 并假设系统中有 N_{task} 个任务进程 $PT_0, PT_1, \dots, PT_{N_{\text{task}}-1}$, 且任务进程 PT_i 由 TT_i 个线程 $TD_0, TD_1, \dots, TD_{TT_i-1}$ 组成, 任务集合采用 DAG(Directed Acyclic Graph) 表示, 即有向无环图可以表示一个四元组: $\Pi = (V, E, RT, CT)$, 其中:

① V 为顶点的集合 $\{V_i\}$. V_i 对应一对二元组 (TD_i, PT_i) , 其中 TD_i 是相应的线程, PT_i 是对应的进程;

② E 为有向边的集合 $\{E_i\}$, 同时线程 TD_i 和 TD_j 可用 $(V_i, V_j) \in E$, 即 TD_i 是 TD_j 的一个前驱, TD_j 是 TD_i 的一个后继;

③ RT 为运行时间的集合 $\{RT(TD_i)\}$, $RT(TD_i)$ 表示线程 TD_i 估计运行时间;

④ CT 为边 (V_i, V_j) 的通信时间的集合 $\{CT_{i \rightarrow j}\}$. $CT_{i \rightarrow j}$ 值越大表示两个线程之间通信越频繁, 通信的代价越大, 反之亦然. $CT_{i \rightarrow j} = 0$ 表示 TD_i 和 TD_j 没有通信. 而属于同一个进程的线程, 由于共享存储单元, 边 (V_i, V_j) 可设为最大值;

⑤ $\text{Com}[N_{\text{task}}][N_{\text{task}}]$ 为存储进程通信量的二维数组;

$\text{Gene}[\text{MaxH}][\text{MaxW}]$ 为存储子图节点高度的二维数组, MaxH 为子图最大高度值, MaxW 为每层中最大任务节点数;

PT_{argv} 为整体所有进程平均执行时间;

CT_{argv} 为整体所有进程间平均通信时间;

LF_{domain} 为整体负载均衡阈值;

PTN_{comp} 为复合节点数;

$LF_{i \rightarrow j}$ 为 PT_i 和 PT_j 之间的实际任务负载度;

ET_{argv} 为 DAG 图分配到处理器上每个核的平均执行时间;

ET_i 为第 i 个处理核上所有任务节点执行时间, ET_j 同理;

$\text{Fit}(S)$ 为适应度函数, $F_Execute(S)$ 是调度个体的执行时间值.

⑥ 各处理器为均衡同构处理器, 任务执行顺序

不可剥夺.

如图 1 为一个 DAG 图. 一共包含 3 个进程, PT_0, PT_1 和 PT_2 ; 属于 PT_0 的线程有 TD_0, TD_1 和 TD_2 ; 属于 PT_1 的线程有 TD_3 和 TD_4 ; 属于 PT_2 的线程有 $TD_5, TD_6, TD_7, TD_8, TD_9, TD_{10}$ 和 TD_{11} ; 图中的边表示线程间的通信开销, 任务节点执行时间没有标注.

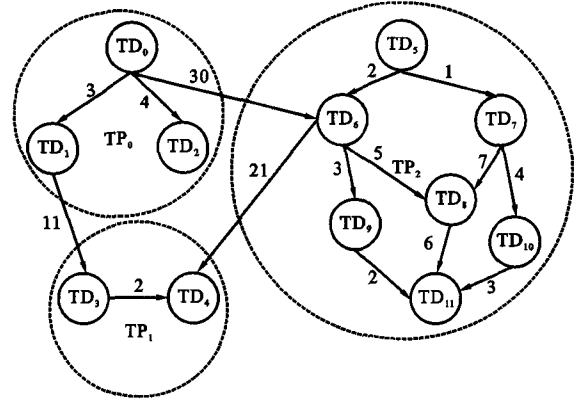


图 1 任务关系图

2.2 改进任务分配

任务分配原则^[2]有:

① 将通信频繁的进程分配给同一个处理节点, 减少通信开销;

② 属于同一个进程的线程分配在一起, 避免频繁的通信开销;

③ 尽量不将多个进程分配给同一处理器核, 避免频繁切换开销;

④ 各个处理器节点及处理器核的负载基本均衡.

文献[3-4]采用将进程间通信开销从大到小排序, 依次选择两两任务节点进行复合, 形成复合节点. 如果对图 1 采用文献[3-4]中任务分配方法, 假定两个处理器 P_0 和 P_1 ; 由于 PT_0 和 PT_2 通信开销较大, 将其复合成 PT_{02} 分配与 P_0 ; 将 PT_1 分配与 P_1 , 结果将会导致 P_0 处理器负载过重, 而 P_1 处理器负载过轻, 各处理器间相对负载不平衡, 影响执行效率和浪费系统资源. 现提出改进算法如下:

Step1 计算各进程间的通信量, 建立进程间通信量的二维表 $\text{Com}[N_{\text{task}}][N_{\text{task}}]$, 进程间无通信其值设为 0;

Step2 计算整体任务进程的平均执行时间 PT_{argv} 和各进程间平均通信时间 CT_{argv} , $PT_{\text{argv}} = (\sum PT_i) / N_{\text{task}} (0 \leq i \leq N_{\text{task}} - 1)$, $CT_{\text{argv}} = (\sum CT_i) / (N_{\text{task}} * 2) (0 \leq i \leq N_{\text{task}} - 1)$;

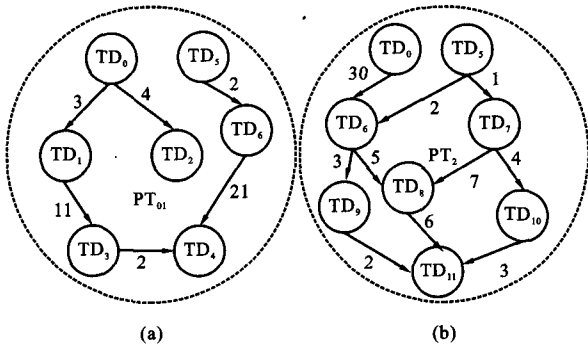
Step3 设定处理器负载均衡阈值 $LF_{domain} = PT_{argv} \cdot \alpha + CT_{argv} \cdot \beta + A^*$; (α 和 β 是比例因子, A^* 是负载动态系数), 并设定 $PTN_{comp} = N_{task} - 1$;

Step4 如果 $PTN_{comp} \leq N_{proc} - 1$; 跳转到 Step 6, 程序结束; 否则, 执行 Step 5;

Step5 在二维数组中选择通信最大的两个进程 PT_i 和 PT_j ; 计算它们之间的负载度 LF_{i+j} , 即 $LF_{i+j} = (PT_i + PT_j + CT_{i+j}) / 2$; 如果 $LF_{i+j} \leq LF_{domain}$, 则进行复合, $PTN_{comp} = PTN_{comp} - 1$, 跳转到 Step 4; 否则, 在二维数组中查询下一条进程间通信量仅次于上一条通信量的两进程进行复合, 并跳转到 Step 5;

Step6 复合节点处理完毕, 程序结束。

经过如上算法, 形成的节点数等于处理器数, 但由于任务节点间存在偏序关系, 为了减少处理器间通信时间开销, 可通过在存在偏序关系的两个复合节点间寻找一条关键路径, 将关键路径上的节点采用复制方法拷贝过来, 这样便形成了各个相互独立的任务子图, 每个任务子图对应一个处理器并在相应处理器上执行调度。假定有两个处理器, 则调整后的任务分配如图 2 所示。



行 Step 5;

Step3 计算 i 行的任务节点数令其为 N_i , 如果 N_i 小于处理核数 N_{core} , 则保证有 N_i 个处理核分配一个任务节点, 即随机选择任务节点 TD_{ij} 分配到随机的一个处理核 C_i 上, 同时已分配的节点和处理核不参与后续操作, 直到任务节点分配完成, $i++$, 跳转到 Step2; 否则执行 Step 4;

Step4 N_i/N_{core} 得到商和余数令其为 d 和 r , 在此需进行 $d+1$ 轮随机选择任务节点和随机选择处理核进行分配, 同样在每轮中已分配的节点和处理核不参与后续操作, 操作结果保证有 r 个处理核分配 $d+1$ 个节点, $N_{\text{core}}-r$ 个处理核分配 d 个节点, 任务节点分配完成, $i++$, 跳转 Step 2;

Step5 所有任务节点分配完成, 进行动态调整, 计算该整个 DAG 图任务节点分配到每个核上的平均执行时间 ET_{argv} , 并设定变量 $i=0$ 和 $j=N_{\text{core}}-1$;

Step6 如果 $i \leq N_{\text{core}}-1$ 且 $j \geq 0$, 则执行 Step 7; 否则执行 Step 10;

Step7 计算第 C_i 处理核任务节点整体执行时间 ET_i , 如果 $ET_i \geq ET_{\text{argv}}(1+\delta)$ 且 $i \leq N_{\text{core}}-1$ ($0 < \delta < 1$) δ 是动态比例因子, 执行 Step 8; 否则 $i++$, 跳转 Step 6;

Step8 计算第 C_j 处理核任务节点整体执行时间 ET_j , 如果 $ET_j \leq ET_{\text{argv}}(1-\delta)$ 且 $j \geq 0$, 执行 Step 9; 否则 $j--$, 跳转 Step 6;

Step9 在处理核 C_i 上随机选择一个任务节点迁到 C_j 上, 同时满足迁移前后的任务节点间的高度约束条件, 跳转到 Step 6;

Step10 动态调整完成, 算法结束.

3.3 适应度函数

适应度函数在遗传算法中通常是优化问题的目标函数, 以染色体适应度大小来确定该任务序列被遗传到下一代群体中的概率, 针对多核处理器系统的任务分配与调度问题, 使用任务执行时间作为评价标准, 则有如下公式:

$$\text{Fit}(S) = \begin{cases} C - F_{\text{Execute}}(S) & \text{if } F_{\text{Execute}}(S) < C \\ 0 & \text{else} \end{cases} \quad (2)$$

其中, C 是公式 (2) 一个足够大的常数, 以保证 $\text{Fit}(S) \geq 0$, $F_{\text{Execute}}(S)$ 表示调度种群中个体 S 所需的执行时间.

3.4 交叉操作

在多核任务调度问题中, 外部交叉容易产生冗

余任务节点被多次执行, 同时将会带来有效任务节点丢弃, 具有不合理性. 文中采用内部单点交叉方式, 具体算法如下:

Step1 依据交叉率随机选取个体 S ;

Step2 随机选择 S 两个执行序列 L_i 和 L_j , 其中 $i \neq j$ 并计算 L_i 和 L_j 最大高度值 (假定 L_i 序列任务节点最大高度值较小);

Step3 在 L_i 内随机选择任务节点 TD_{im} , 计算高度值 $H(TD_{im})$;

Step4 在 L_j 随机选择 TD_{jn} 任务节点, 使得满足 $H(TD_{im}) = H(TD_{jn})$;

Step5 交换 L_i 和 L_j 交叉点之后的序列, 交叉操作结束.

3.5 变异操作

变异操作是增加个体的多样性, 避免形成局部最优解, 具体算法如下:

Step1 依据变异率随机选取个体 S ;

Step2 随机选择 S 两个执行序列 L_i 和 L_j , 其中 $i \neq j$, 计算两序列的执行时间 ET_i 和 ET_j (这里假设 ET_i 较小);

Step3 从 L_j 中随机抽取任一任务节点 TD_{jn} 迁移到 L_i 中, 并满足 $H(TD_{im}) \leq H(TD_{jn}) \leq H(TD_{im+1})$, 变异结束.

3.6 算法终止条件

算法终止采用双重标准:

- ① 算法连续 20 代没有得到进化;
- ② 算法达到规定的最大迭代次数.

4 仿真实验与结果分析

实验采用随机图作为任务调度测试的输入值, 并通过比较调度长度和执行时间来衡量算法的优劣. 为使实验简单, 设定处理器数为 1, 处理器对应的处理核数和任务线程数可变. 实验中所用参数有: 杂交和变异概率分别为 $P_c = 0.8$ 和 $P_m = 0.05$, 种群规模为 100, 算法最大迭代代数为 1000 代, 随机生成 DAG 图, 各任务高度由系统随机生成, 估计运行时间为 1~6 之间的随机数. 实验采用标准 GA 算法、文献 [7] 中 CPGS 的算法和 BPGA 算法比较分析, 实验数据取 20 次平均值, 如表 1 所示.

从表 1 中可以看出随着处理核和任务数增多, BPGA 算法在调度长度和执行时间上优于标准 GA 和 CPGS 算法. 主要原因在于 BPGA 算法在初始种群中的到了优化, 使 BPGA 算法具有更好的执行效.

图4所示的算法的静态性能曲线.列出了算法在不同进化代数时所找到的最优解,处理器用 m 表示,任务节点数用 n 表示,则图4是40个任务节点在4个处理器分配调度的仿真结果.

表1 仿真结果比较

处理 核数	任务 节点数	调度长度			执行时间/s		
		GA	CPGS	BPGA	GA	CPGS	BPGA
2	20	56	53	50	251	235	207
	40	121	113	107	417	404	373
	80	179	158	129	819	768	722
3	20	47	40	35	235	221	194
	40	83	75	62	391	369	323
	80	117	102	91	767	714	638
4	20	39	33	28	194	176	133
	40	67	58	46	317	282	225
	80	93	73	68	618	531	442

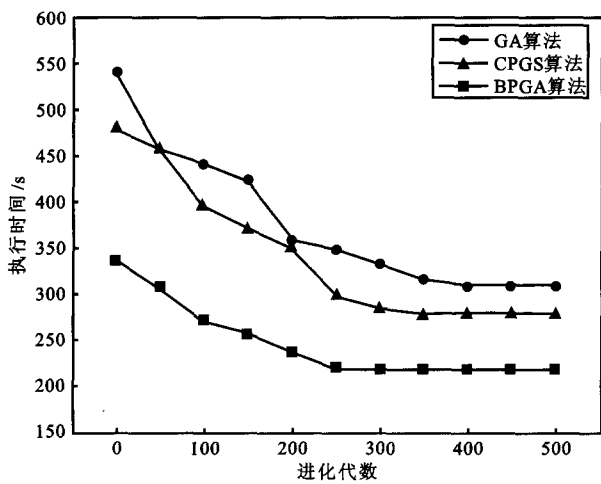


图4 算法静态性能曲线 ($m=4, n=40$)

图4中算法的静态性能曲线显示了BPGA算法具有较好的执行时间和收敛速度,保证了最优解性能不降低.实验结果表明BPGA算法优于标准GA和CPGS算法.

5 结束语

文中对现有任务分配进行改进,即在任务分配同时考虑进程间通信量和合并的两个进程之间的相对平均执行时间,使任务分配更加均衡化.在任务调度方面,对遗传算法初始种群进行改进,使任务随机化分配到处处理核,且同时在每个处理核上的线程任务数相对均衡,进一步考虑了处理核的负载均衡度,优化了种群生成,使算法的执行效率得到提高.

参考文献:

[1] Correa R C, Ferreira A, Rebreyend P. Scheduling multiprocessor tasks with genetic algorithm [J]. IEEE Transactions on Parallel and Distributed Systems, 1999, 10(8):825-837.

[2] Yi Liu, Xin Zhang, He Li, et al. Allocating tasks in multi-core processor based parallel systems[C]// Proceedings of the 2007 IFIP International Conference on Network and Parallel Computing Workshops. Dolard: Cracow, 2007:748-753.

[3] 刘轶, 张昕, 李鹤, 等. 多核处理器大规模并行系统中的任务分配问题集算法[J]. 小型微型计算机系统, 2008, 29(5):972-975.

[4] 袁云, 邵时. 基于多核处理器并行系统的任务调度算法[J]. 计算机应用, 2008(29):280-282.

[5] 苗蕾, 齐勇, 侯迪, 等. 基于遗传算法的片上多处理器任务调度策略研究[J]. 微电子学与计算机, 2007, 24(6):8-11.

[6] 周双娥, 雷辉. 基于改进的遗传-模拟退火的有序任务调度算法[J]. 微电子学与计算机, 2006, 23(10):62-64.

[7] 兰舟, 孙世新. 基于关键路径知识的任务调度遗传算法[J]. 计算机应用, 2008, 28(2):272-274.

作者简介:

彭蔓蔓 女, (1964-), 教授. 研究方向为计算机系统结构.

黄亮 男, (1983-), 硕士研究生. 研究方向为多核任务调度与负载均衡.