高技术通讯 2012 年 第22 卷 第3 期:225 ~ 230

doi:10.3772/j.issn.1002-0470.2012.03.001

基于异构多核处理器的高效任务调度算法①

李静梅② 李 静 丁 楠

(哈尔滨工程大学计算机科学与技术学院 哈尔滨 150001)

摘 要 针对现有异构多核处理器任务调度算法效率低的问题,提出了一种综合性的、高效的静态任务调度算法,即聚簇与复制列表优化调度(CDLOS)算法。该算法首先通过对任务图进行聚簇优化,降低某些特殊任务的通信开销;然后从整个任务图的拓扑结构出发计算任务的优先级权值,提高关键任务的优先级;继而采用区间插入和任务复制技术进行调度,降低处理器资源浪费;最后通过优化调度结果,消除冗余任务,减小整个任务的调度长度。实例分析和模拟实验结果表明:与以往算法相比,此新算法较高地提升了多核处理器任务调度的效率,具有更好的应用前景。

关键词 异构多核,任务调度,聚簇,任务复制,列表

0 引 言

异构多核处理器任务调度的好坏直接影响着多核处理器的性能,如果调度不当,甚至会抹煞多核处理器高并行性的优势,降低整个系统的性能[1]。异构多核处理器需要同时满足总任务完成时间最少和任务优先级约束的要求,这会导致不可能将任务都分配到执行效率最高的处理器内核。因此,设计合理的调度方法,在满足任务优先级约束的基础上减少总任务的执行时间,则成为异构多核处理器任务调度研究的重点。研究人员针对这一问题进行了大量研究,Topcuoglu 等提出了异构最早结束时间优先(heterogeneous earliest-finish-time,HEFT)算法和异构关键路径优先(critical-path-on-a-processor,CPOP)算法[2],通过采用表调度方法在一定程度上提高了任务的调度效率;Sai Ranga 提出了异构关键节点优先(heterogeneous critical node first,HCNF)算法[3],通过优先调度关键任务减少了总任务的调度长度。与以往算法相比,尽管 HEFT、CPOP 和 HCNF 算法具有较高的调度效率,但仍存在通信开销过大、关键任务优先级不高、冗余任务过多等问题,严重影响了多核处理器的任务调度性能。为进一步提高多核处理器的系统性能,本文提出了一种综合性的、高效的静态任务调度算法——聚簇与复制列表优化调度(clustering and duplicate list optimization scheduling,CDLOS)算法。CDLOS 算法首先对有向无环图(DAG)进行聚簇优化,然后从整个任务图的拓扑结构出发考虑任务的优先级,在考虑区间插入和任务复制的条件下对任务进行调度,最后通过冗余任务的消除优化任务调度结果,减少总任务的执行时间,降低任务的实际调度长度。

1 任务调度的数学模型

静态任务调度通常采用 DAG 表示任务模型[4]。图中节点表示任务,有向边表示任务间的依赖和通信关系,用 $G = (N,E,W,C)$ 表示。

$N = \{n_1,n_2,\cdots,n_i,\cdots\}$ 表示图中节点的集合,n_i 表示 DAG 中第 i 个节点(应用程序中第 i 个任务),n 表示一个 DAG 中的总任务数。

$E = \{e_{1,2},e_{2,3},\cdots,e_{i,j},\cdots\}$ 表示图中有向边的集合,$e_{i,j}$ 表示第 i 个节点与第 j 个节点间的有向边,e 表示一个 DAG 中有向边的数量[5]。

$\boldsymbol{W}$ 是一个 $n \times p$ 的矩阵,n 为任务数,p 表示系统中处理器内核的数量。矩阵中的元素 $\boldsymbol{W}(n_i, p_j)$ 表示任务 n_i 在处理器内核 p_j 上的执行开销。将节点 n_i 的平均执行开销作为节点的权值,定义如下:

① 国家自然科学基金(60873037,60873138)资助项目。

② 女,1964 年生,教授;研究方向:计算机系统结构;联系人,E-mail: lijingmei@hrbeu.edu.cn

(收稿日期:2010-09-19)

$$\overline{W_i} = \sum_{j=1}^{p} W(n_i, p_j)/p \quad (1)$$

$C(n_i, n_j)$ 表示任务 n_i 与 n_j 间的通信开销，如果将两个任务分配到同一处理器内核，则通信开销为零。通信开销定义如下

$$C(n_i, n_j) = s_m + \frac{data(n_i, n_j)}{rate(p_m, p_n)} \quad (2)$$

其中 S_m 表示任务通信启动时间，$data(n_i, n_j)$ 表示任务 n_i 与 n_j 间的数据通信量，$rate(p_m, p_n)$ 表示处理器内核 p_m 与 p_n 之间的数据传输率。

节点 n_i 与 n_j 间的平均通信开销作为边权值，定义如下：

$$\overline{C(n_i, n_j)} = \overline{S} + \frac{data(n_i, n_j)}{\overline{rate}} \quad (3)$$

其中 $\overline{S}$ 表示任务平均通信启动时间，$\overline{rate}$ 表示处理器内核的平均数据传输速率。

$Pre(n_i)$ 表示节点 n_i 的前驱节点集合，当前任务只有在它全部前驱节点的数据到达后才能执行。$Succ(n_i)$ 表示节点 n_i 的后继节点集合。没有前驱节点的节点为入口节点，记为 n_{entry}。没有后继节点的节点为出口节点，记为 n_{exit}。

关键路径 CP 是指图中从入口节点到出口节点的最长路径。最长路径的值 |CP| 等于关键路径上所有节点的计算开销与通信开销之和。

任务 n_i 在处理器内核 p_j 上的最早开始时间定义如下：

$$EST(n_i, p_j) = \max\{P\_Available[j], \max_{n_m \in Pre(n_i)}\{EST(n_m) + W(n_i, p_j) + \overline{C(n_m, n_i)}\}\} \quad (4)$$

其中 P_Available[j] 表示任务在处理器内核 p_j 上获得的最早可用时间，即已经调度到处理器内核 p_j 上的任务全部执行完成的时间。

任务 n_i 在处理器内核 p_j 上的最早结束时间定义如下：

$$EFT(n_i, p_j) = EST(n_i) + W(n_i, p_j) \quad (5)$$

2 现有任务调度算法分析

异构系统任务调度已被证明是一个 NP 完全问题[6]。现有任务调度算法主要有 HED[7]（heterogonous economical duplication）、DLS[8]（dynamic level scheduling）、MH[9]（mapping heuristic）、BIL（best imaginary level）、TDS（task duplication scheduling）、HEFT、CPOP 和 HCNF 等。其中，以 HCNF、HEFT、CPOP 算法为优，HCNF 算法具有最高的效率，HEFT 算法次之。

CPOP 算法将节点的向上排序值 ranku 和向下排序值 rankd 之和作为节点的权值，若某个节点的权值与入口节点的权值相同，将该节点标记为关键路径节点。分配阶段首先寻找串行执行所有关键路径节点具有最小最早完成时间的处理器 CPP，然后从入口节点开始调度。如果当前节点为关键路径节点，将其分配到处理器 CPP；否则，在考虑区间插入技术的基础上，将当前任务分配给具有最小最早完成时间的处理器。该算法虽然具有比较高效的调度结果，但它仅保证了关键路径节点具有较高的优先级，某些关键节点的优先级并不高，该类节点的调度延迟将会增加整个关键路径节点的执行时间，降低整个应用程序的调度效率。同时，将所有关键路径节点分配到相同的处理器，易产生负载不平衡的现象，进一步延迟了整个应用程序任务的执行时间。

HEFT 算法在排序阶段使用节点的 ranku 作为任务的优先级[2]。分配阶段在考虑区间插入技术的基础上，将任务分配给具有最小最早完成时间的处理器。该算法对含有大量处理器内核的多核处理器具有较高的调度效率[10]。但该算法使用向上排序值 ranku 作为任务的权值，并没有从整个任务图的拓扑结构考虑任务在整个任务图中的位置，某些关键任务的优先级可能不高，延长了整个任务的调度长度。其次，该算法使用区间插入技术来提高处理器资源的利用率，但满足区间插入条件的任务并不多，且没有降低任务的通信开销，对具有较大通信开销的应用程序任务调度效率不够理想，仍需进一步优化。

HCNF 算法在每一步调度中都保证关键路径节点具有最高的优先级，通过将关键路径节点优先调度到具有最小最早完成时间的处理器，减少了整个任务图中所有关键路径节点的完成时间[3]。在任务分配过程中，通过复制任务的关键前驱节点到空闲处理器时间段来减少任务的通信开销，降低了总任务的完成时间。但该算法中处理器的空闲时间段并没有得到足够充分地利用，处理器中存在的剩余空闲时间段仍旧较多，浪费了处理器资源，降低了任务的调度效率。同时，由于该算法采用了任务复制技术，与大多数基于复制的任务调度算法相同，调度结果中存在许多冗余任务，这些不必要冗余任务的多次执行降低了处理器资源的利用率，因此，该算法仍需进一步改进，以消除不必要的冗余任务，提高算法的调度效率。

3 CDLOS 算法

针对以上三种较高效率的任务调度算法存在的不足,为提高算法的调度效率,进一步缩短总任务的执行时间,在综合吸收现有算法各自优点的基础上,本文提出了一种高效的综合性静态任务调度算法——CDLOS 算法,以达到进一步提高处理器性能的目的。该算法主要包括三个阶段:任务图的聚簇优化、任务分配、调度结果优化。

3.1 任务图的聚簇优化

针对现有任务调度算法的某些任务通信开销远远大于任务执行开销,严重影响任务执行效率的问题,CDLOS 算法在任务图的聚簇优化阶段采用线性任务聚簇方法,通过深度优先遍历 DAG 图,将任务图中某些特殊的、通信开销较大的、容易聚簇的任务聚簇到其前驱节点,消除了该类任务的通信开销。

聚簇任务必须同时满足 3 个条件:(1)当前节点只有一个直接前驱节点;(2)当前节点为其直接前驱节点的唯一直接后继节点;(3)当前任务在不同处理器内核上执行的最大时间小于该任务与其唯一前驱节点之间的平均通信开销,即 $\max(W(n_i, p_j)) < \overline{C(n_k, n_i)}$,其中 n_k 为 n_i 的直接前驱节点。

聚簇优化过程如下:

```
1. for DAG 中所有节点 do
2. if 当前节点 n_i 仅有一个直接前驱节点 n_k then
3.    if 节点 n_k 的直接前驱节点只有一个直接后继节点 then
4.       if max(W(n_i, p_j)) < C(n_k, n_i) then
5.          将节点 n_k 聚簇到前驱节点 n_k;
6.       end if
7.    end if
8. end if
9. end for
```

3.2 任务分配

任务分配过程如图 1。针对现有任务调度算法存在的某些关键节点优先级不高,延迟了整个任务的完成时间问题,CDLOS 算法在任务优先级计算阶段试图从整个任务图的拓扑结构出发,在平衡各方面因素对任务调度长度影响的基础上,考虑任务在 DAG 中的位置,将 $\mathrm{Succ\_sum}(n_i)$ 作为任务的优先权权值。优先权权值定义如下:

$$\mathrm{Succ\_sum}(n_i) = \overline{w_i} + \sum_{n_j \in succ(n_i)} (Succ\_sum(n_j) + \overline{C(n_i, n_j)}) \quad (6)$$

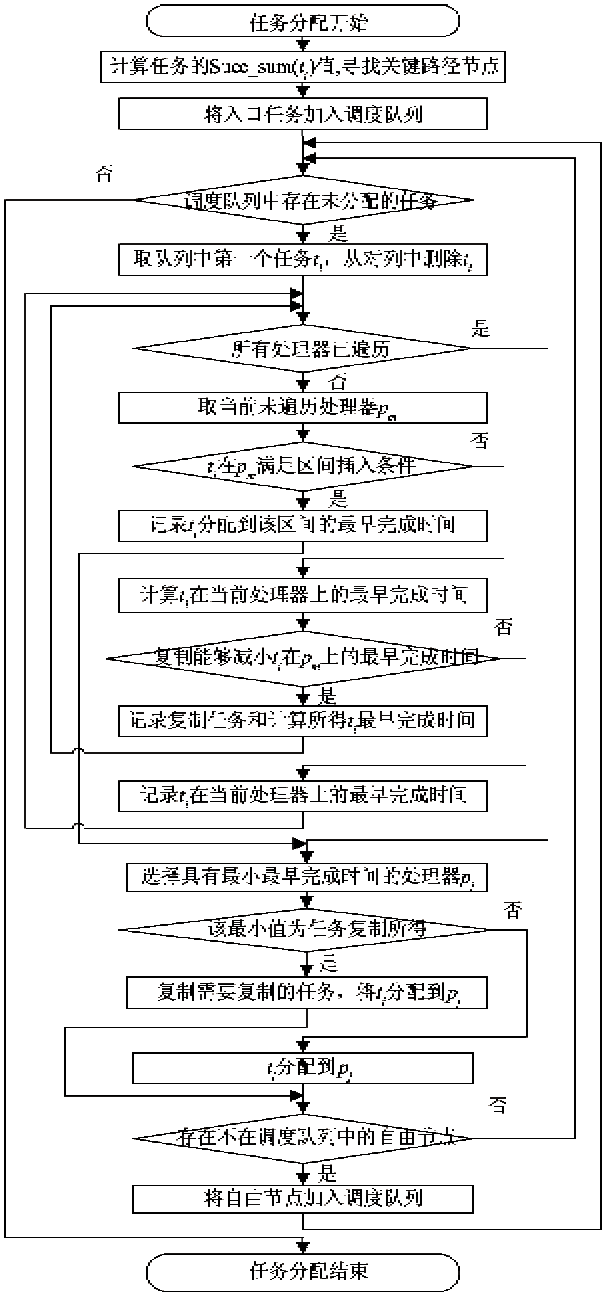

图 1 任务分配过程

聚簇优化阶段只是针对某些特殊的、通信开销较大的任务进行优化,现有算法存在的处理器资源利用不足以及任务通信开销过大的问题并没有得到较大范围的解决,仍需进一步优化。为解决这一问题,CDLOS 算法在任务分配阶段采用了区间插入技术和任务复制技术,每一步调度都赋予关键路径节点以最高的优先级,其余节点按照节点的优先权权值排序。算法首先计算任务的优先权权值,识别关键路径节点,将关键路径节点标记为 CP 节点。然后将入口节点作为自由节点加入调度队列。在调度过程中,如果调度队列不为空,取队列中第一个任务 n_i,如果能够将 n_i 插入到处理

器内核的空闲时间段,计算该任务在该空闲时间段的最早完成时间,并将该任务分配到该空闲时间段;否则,计算任务 n_i 在每个处理器内核上的最早完成时间,选择能够使任务 n_i 具有最小最早完成时间的处理器内核 p_j,如果该最小最早完成时间是通过任务复制得到,复制对应的任务到 p_j,并将任务 n_i 分配到处理器内核 p_j;如果该最小最早完成时间不是通过任务复制得到,将任务 n_i 直接分配到处理器内核 p_j,完成任务 n_i 的分配过程。最后,将任务 n_i 调度完成后的所有自由节点插入到调度队列,如果自由节点中存在 CP 节点,首先将自由 CP 节点加入调度队列的开始,然后非 CP 自由节点按照节点优先权值从大到小的顺序加入调度队列,递归执行以上调度过程,直到所有任务调度完成。

3.3 调度结果优化

针对现有任务复制算法中存在的冗余任务过多,严重浪费处理器资源问题,CDLOS 算法在最后的调度结果优化阶段对冗余任务进行了优化处理。

调度结果优化阶段首先进行冗余任务的删除。算法寻遍所有处理器内核,将复制任务及其相关信息存储到任务复制列表 DL,将 DL 中的任务按最早完成时间从大到小排列。查找任务复制列表 DL,如果 DL 不为空,取 DL 中的第一个任务 $n_{i,j}$(表示分配到处理器内核 p_j 上的任务 n_i),并从 DL 中删除任务 $n_{i,j}$。如果删除处理器内核 p_j 上的任务 n_i 不会延长 n_i 后继节点的完成时间,则任务 $n_{i,j}$ 为冗余任务,将该任务从对应的处理器内核上删除;如果该任务的删除延长了其后继节点的完成时间,则该任务不是冗余任务,不能从调度结果中删除,重复执行以上判断过程,直到任务复制列表 DL 为空。

调度结果优化阶段最后进行结果的优化处理。查找所有处理器内核,记录每个处理器内核的空闲时间段,在考虑区间插入的条件下重新调整每个任务在该处理器内核上的分配。在进行区间插入技术时,寻找开始时间大于任务最早开始时间的空闲时间段,如果找到,判断任务插入对整个调度长度的影响,如果能够减少任务调度的长度,则将任务重新分配到该空闲时间段。

优化过程如下:

1. 将复制任务信息、所在处理器内核信息以及完成时间 EFT 存储到 DL;
2. 将 DL 任务按最早完成时间从大到小排列;
3. while DL 不为空 do
4. 取 DL 中第一个任务 $n_{i,j}$;
5. 计算删除 $n_{i,j}$ 后,其后继任务的最早完成时间 EFT2;
6. if EFT2 不大于 EFT then
7. $n_{i,j}$ 为冗余任务,删除 DL 中的 $n_{i,j}$,从处理器内核 p_j 上删除任务 n_i;
8. else
9. 删除 DL 中的 $n_{i,j}$;
10. end while
11. 按照任务完成时间从小到大将任务信息存储到 CN,空闲时间段信息存储到 KL,总任务的调度长度记为 SL1;
12. while CN、KL 均不为空 do
13. 取 CN 中第一个任务 $n_{i,j}$,从 CN 删除 $n_{i,j}$;
14. for KL 中的空闲时间段 do
15. 取空闲段 Li;
16. if $n_{i,j}$ 的最早开始时间不大于 Li 的开始时间 then
17. if $n_{i,j}$ 的执行时间不大于 Li 的长度 then
18. 将 $n_{i,j}$ 重新调度到处理器内核的空闲时间段 Li,更新 KL;
19. break;
20. else
21. 计算通过后移处理器内核 p_j 上的任务将 $n_{i,j}$ 插入空闲时间段后任务调度的总长度 SL2;
22. if SL2 小于 SL1 then
23. 通过后移任务,将任务 $n_{i,j}$ 重新调度到空闲时间段;
24. break;
25. end for
26. end while

3.4 CDLOS 算法实现

通过具体 DAG 图的调度,说明算法的具体调度性能。图 2 是一个随机生成的任务图,表 1 是图 2 中任务在不同处理器内核上的执行时间,CDLOS 算法对图 2 的调度结果如图 3 所示,表明了各个任务在对应处理器内核上的调度长度。

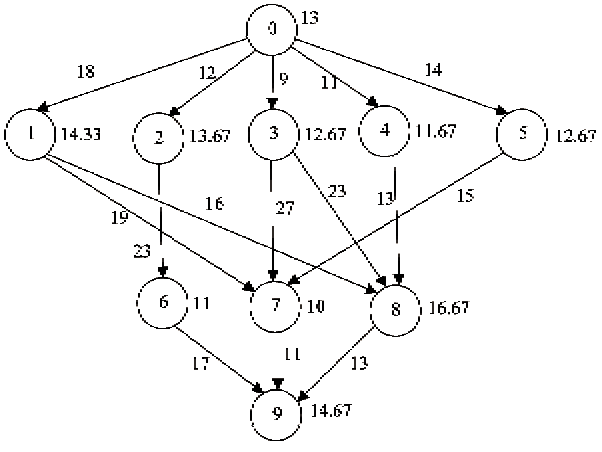

图 2 随机任务图

表1 任务在不同处理器内核的执行时间

Node	P0	P1	P2
0	14	16	9
1	13	12	18
2	9	13	19
3	13	8	17
4	12	13	10
5	13	16	9
6	7	15	11
7	5	11	14
8	18	12	20
9	21	7	16

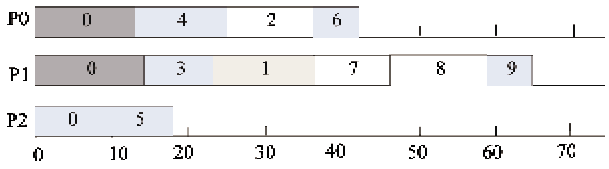

图3 CDLOS 调度结果

与已有算法相比,CDLOS 算法具有更短的调度长度。本文以调度效率较高的 HEFT、CPOP 和 HCNF 算法分别对图 2 进行调度分析,各算法的调度结果为:HEFT 算法调度长度为 81,CPOP 算法调度长度为 86,HCNF 算法调度长度为 73。CDLOS 算法调度长度为 66,与 HEFT、CPOP 和 HCNF 算法相比分别减少 15、20、7 个长度。通过分析可知,在对图 2 的调度中,CDLOS 算法的调度结果最优,具有较高的调度效率。

4 性能验证

为进一步测试算法的高效性,通过模拟实验对大量随机任务图进行调度,记录各算法的调度长度,通过调度结果分析,比较 CDLOS 算法与已有 HEFT 算法、CPOP 算法和 HCNF 算法的性能。

4.1 性能评估参数

在算法的性能测试中,为更加公正、合理的评价算法的性能,需要对多个应用程序任务进行调度。每个应用程序的任务调度长度不同,为更好地评价新算法的任务调度长度和性能优势,本文选用调度下界比(schedule length ratio,SLR)和加速比 Speedup 作为算法的性能评估参数。

SLR:算法实际调度长度与关键节点执行时间的最小值之比。对于一个 DAG 图,调度算法产生的 SRL 越小,说明算法越高效。SLR 定义为

$$SLR = \frac{makespan}{\sum_{t_i \in cp} \min_{p_j \in P}\{W(t_i, p_j)\}} \tag{7}$$

Speedup:在一个处理器内核上串行执行 DAG 图中的所有任务使用的最少时间与算法实际调度长度的比值。对于一个 DAG 图,调度算法产生的 Speedup 越大,说明算法越高效。Speedup 定义为

$$Speedup = \frac{\min_{p_j \in P}\{\sum_{t_i \in DAG} W(t_i, p_j)\}}{makesapn} \tag{8}$$

4.2 使用随机拓扑图验证结果

对算法性能的测试目前主要有随机任务图和实际应用任务图两种方法。由于实际应用任务图较少,因此本文选用大量随机任务图实现算法的性能测试。随机任务图的产生使用与文献[11]相同的方法。产生随机任务图的参数设置如下:

任务个数 $n = \{20,30,40,50,60,70,80,90,100\}$;

任务图中节点的最大出度 ax _ out = {1,2,5,100};

通信时间与计算时间的比值 CCR = {0.1,0.5,1.0,5.0,10};

任务在不同处理器内核上执行时间的差异度 β = {0.1,0.5,1,1.5,2}。

不同组合将产生 900 组不同类型的任务图,每组类型中随机产生 20 个具有不同边和节点权值的任务图,所以总共产生 18000 个任务图。通过对产生的大量任务图进行调度,消除了特殊拓扑结构图对算法性能的影响,能够更好地对算法的性能进行客观评测。

通过实验结果数据分析,得到的算法调度结果对比如图 4 和图 5 所示。通过图 4 和图 5 的结果分析可知,与 HEFT 算法、CPOP 算法和 HCNF 算法相比,CDLOS 算法的调度下界比 SLR 约分别降低17.54%、18.14%和9.86%,加速比 Speedup 约分别提高16.58%、17.37%和8.46%。CDLOS 算法较高地提升了任务调度的效率。

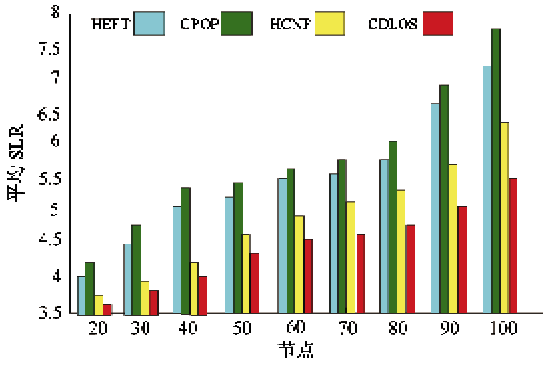

图4 任务数不同时四种算法的 SLR 对比

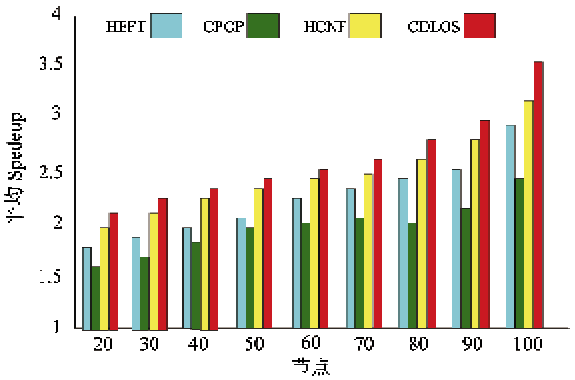

图 5 任务数不同时四种算法的 Speedup 对比

5 结 论

本文针对异构多核处理器任务调度问题提出的综合性高效静态任务调度算法——CDLOS 算法,在不改变传统任务调度模型的基础上,能够提高关键任务调度的优先级,减少任务执行中的通信开销,降低冗余任务对处理器资源的浪费,减少整个任务的调度长度。通过任务实例和模拟实验对此算法进行验证的结果表明,与已有算法相比,此新算法的调度效率约提高 36%,随着任务数量和处理器核数的增多,新算法的优越性更加明显。新算法的提出对异构多核处理器任务调度的研究具有重要的理论意义,对高性能多核处理器的建设具有重要的现实意义。

参考文献

[1] Hoffmann R, Prell A, Rauber T. Dynamic task scheduling and load balancing on cell processors. In: Proceedings of the 18th Euromicro Conference on Parallel, Distributed and Network-Based Processing, Pisa, Italy, 2010. 205-212

[2] Topcuoglu H, Hariri S, Wu M Y. Task scheduling algorithms for heterogeneous processor. In: Proceedings of the Heterogeneous Computing Workshop, San Juan, Puerto Rico, 1999. 3-14

[3] Sai Ranga P C. Algorithms for Task Scheduling in Heterogeneous Computing Environments: [Ph. D dissertation]. Auburn University, 2006. 44-95

[4] PriyaDarshini V N, Sankari P S, Chitra P, et al. Reliable task scheduling for heterogeneous distributed computing environment. In: 2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies, Madurai, India, 2009. 494-496

[5] *Bajaj R, Agrawal D P. Improving scheduling of tasks in a* heterogeneous environment. *IEEE Transaction on Parallel and Distributed Systems*, 2004, 15:107-118

[6] Gallet M, Marchal L, Vivien F. Efficient scheduling of task graph collections on heterogeneous resources. In: Proceedings of the 2009-Proceeding of the 2009 IEEE International Parallel and Distributed Processing Symposium, Lyon, France, 2009. 1-11

[7] Agarwal A, Kumar P. Economical duplication based task scheduling for heterogeneous and homogeneous computing systems. In: 2009 IEEE International Advance Computing Conference, Patiala, India, 2009. 87-93

[8] SIH G C, LEE E A. A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures. *IEEE Transactions on Parallel and Distributed Systems*, 1993, 4(2): 175-186

[9] Rewini H E, Lewis T G. Scheduling parallel program tasks onto arbitrary target machines. *Parallel and Distributed Computing*, 1990, 9: 138-153

[10] Bittencourt L F, Sakellariou R, Madeira E R M. DAG scheduling using a lookahead variant of the heterogeneous earliest finish time algorithm. In: Proceedings of the 18th Euromicro Conference on Parallel, Distributed and Network-Based Processing, Pisa, Italy, 2010. 27-34

[11] 蒋韵联. 并行异构系统任务调度问题研究:[硕士学位论文]. 合肥:中国科学技术大学,2006. 39-41

An efficient task scheduling algorithm for heterogeneous multi-core processors

Li Jingmei, Li Jing, Ding Nan
(College of Computer Science and Technology, Harbin Engineering University, Harbin 150001)

Abstract

In view of the low efficiency problem of present task scheduling algorithms for heterogeneous multi-core processors, a new efficient static task scheduling algorithm, called the clustering and duplicate list optimization scheduling (CDLOS) algorithm, is proposed. Firstly, this new algorithm optimizes task graphs by using clustering to greatly minimize the communication costs of special tasks. Secondly, the task priority value is calculated in the case of the whole topological structure of task graphs is considered, for giving a higher priority to crucial tasks. Then, the techniques of task insertion and task duplication are applied to the process of task scheduling to enhance the efficiency of processor resources. Lastly, the result of task schedule is optimized effectively, so as to reduce the redundant tasks and the makespan of all tasks. The analysis and simulation results show that compared to old scheduling algorithms, this new CDLOS algorithm can enhance the task scheduling efficiency obviously and will have excellent application prospects.

Key words: heterogeneous multi-core, task scheduling, clustering, task duplication, listing