一种改进的优先级列表任务调度算法

李静梅 王 雪 吴艳霞
(哈尔滨工程大学计算机科学与技术学院 哈尔滨 150001)

摘 要 异构多核处理器任务调度是高性能计算领域的重要问题。针对优先级列表调度算法中存在的优先级排序方法失当、调度结果不理想的问题,提出一种改进的优先级列表任务调度算法。该算法对传统优先级列表任务调度中以任务执行时间平均值作为参数的优先级计算方式进行优化,提出一种基于异构核性能差异性、依赖任务特征加权优先级的排序方式。在此基础上,以当前格局下每个任务的向后关键路径执行时间为权值作为任务分配到处理器内核的依据,克服贪心思想在内核选择中带来的局部最优解问题。此外,在任务分配阶段利用任务复制和区间插入技术,缩短任务最早开始时间,提高处理器利用率。实例分析和模拟实验结果表明,该算法可有效降低任务的执行时间,能发挥异构多核处理器优势。

关键词 高性能计算,异构多核,任务调度,优先级列表
中图法分类号 TP302 **文献标识码** A

Improved Priority List Task Scheduling Algorithm

LI Jing-mei WANG Xue WU Yan-xia
(Department of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China)

Abstract Task scheduling on heterogeneous multiprocessor is an important problem in the field of high performance computing. The paper proposed an improved priority list task scheduling algorithm to solve the problem that misconduct priority method exists in the task scheduling algorithm and the scheduling result is unsatisfactory. The algorithm improves the traditional priority list of task scheduling method which takes average execution time as parameter to calculate priority of task and proposes a weighted priority method to order the priority of tasks based on heterogeneous multi-core performance difference and dependent task characteristics. And then, the paper proposed a new way to select processor core for task which takes the current situation and backward critical path execution time as weight and overcame the local optimal problem brought by greedy though. Furthermore, at the task allocation stage, the algorithm takes task duplication and interval insertion technique to optimize schedule process and shorten the task earliest start time and improve the processor utilization successfully. The instance analysis and simulation experiments prove the algorithm can reduce the execution time of tasks effectively and give a full play to heterogeneous multi-core processor advantage.

Keywords High performance computing, Heterogeneous multi-core, Task scheduling, Priority list

多核处理器作为目前占领市场最大份额的典型处理器代表,对其关键技术的研究正逐渐成为计算机体系结构的研究热点。

任务调度是指根据调度算法将任务序列分配到不同处理器内核上执行的问题[1]。同构多核处理器和异构多核处理器上的任务调度问题都是多核系统性能发挥的关键问题。同构多核处理器发展于传统单核处理器,包含多个计算能力相同、地位对等的处理器内核。同构多核处理器的任务调度算法在成形的单核处理器任务调度算法的基础上迅速发展,是目前相对成熟的多核体系任务调度算法。由于异构环境的特点,采用同构环境下成熟的任务调度算法不但难以提升系统性能,甚至会抹杀异构环境的优越性。目前异构多核处理器系统中运用较多的为优先级列表任务调度算法,如支持多核的Linux内核操作系统。优先级列表任务调度算法中普遍采用任务的执行时间平均值作为任务优先级计算参数,试图将异构问题转化为同构问题处理,虽获得了一定的性能提升,但在处理器内核执行差异较大的情况下,将极大地阻碍异构多核处理器性能的发挥。另外,现有的任务调度算法中大多采用贪心思想选择任务执行时所处内核,虽然算法简单有效,但却容易使得调度陷入局部最优解,降低整体调度性能。

本文在充分研究异构多核处理器特征的基础之上,针对两种现有经典的优先级列表调度 HEFT[2](Heterogeneous Earlier Finish Time)算法和 HCNF[3](Heterogeneous Critical Node First)算法存在的缺点和不足,提出一种改进优先级列表的任务调度算法——HIPLTS(Heterogeneous improved priority list for Task scheduling),并通过所设计的实验验证

到稿日期:2013-07-07 返修日期:2013-10-15 本文受国家自然科学基金(61003036),黑龙江省基金项目(F201124),Fundamental Research Funds for the Central Universities(HEUCF100606)资助。

李静梅(1964—),女,博士,教授,主要研究领域为计算机系统结构、多核处理器性能优化,E-mail:lijingmei@hrbeu.edu.cn;王 雪(1989—),女,硕士,主要研究领域为多核处理器任务调度;吴艳霞(1979—),女,博士,副教授,主要研究领域为计算机体系结构。

了 HIPLTS 算法的有效性。

1 任务调度的数学模型

不失一般性，采用有向无环 DAG 图表示任务模型。DAG 图的节点表示任务，有向边的方向表示任务间的依赖关系，边上的权值表示任务间的通信时间开销，用四元组 $G=(V,E,W,C)$ 表示。

$V=\{v_0,v_1,\cdots,v_i,\cdots,v_{N-1}\}$ 表示 DAG 图中任务节点的集合，元素 v_i 表示第 i 个节点，$N=|V|$ 表示总任务数，$0\leqslant i\leqslant N-1$。

E 表示 DAG 图有向边的集合，元素 $e_{i,j}$ 表示任务 v_i 与任务 v_j 的偏序关系，在 DAG 图中表示为节点之间的有向边，即任务 v_i 与任务 v_j 之间存在邻接关系。

W 是一个 $N\times M$ 的矩阵，M 为处理器内核的个数。p_m 表示处理器的第 m 个内核，$1\leqslant m\leqslant M$，集合元素 $w(v_i,p_j)$ 表示任务 v_i 在内核 p_m 上的时间执行开销。DAG 图中，将任务的平均时间执行开销作为节点的权值，任务 v_i 的平均时间执行开销 $\overline{w_i}$ 定义见式(1)。

$$\overline{w_i}=\sum_{m=1}^{M}w(v_i,p_m)/M \tag{1}$$

C 是任务间通信时间开销的集合，元素 $c(v_i,v_j)$ 表示任务 v_i 与 v_j 间的通信时间开销。DAG 图中，将任务的通信时间开销作为边的权值。任务 v_i 与 v_j 间的通信时间开销定义见式(2)。其中，$startup_m$ 为任务在内核 p_m 上的通信启动时间，$data(v_i,v_j)$ 为任务 v_i 和 v_j 之间的通信数据量，$rate(p_m,p_n)$ 为内核 p_m 和 p_n 之间的通信传输速率。

$$c(v_i,v_j)=\begin{cases}startup_m+\dfrac{data(v_i,v_j)}{rate(p_m,p_n)}, & \text{当 } v_i,v_j \text{ 分配到不同的核心}\\ 0, & \text{当 } v_i,v_j \text{ 分配到同一个核心}\end{cases} \tag{2}$$

任务 v_i 与 v_j 间的平均通信时间开销 $\overline{c(v_i,v_j)}$ 定义见式(3)。其中，$\overline{startup}$ 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。DAG 图中，将任务的平均通信延迟开销作为边的权值。

$$\overline{c(v_i,v_j)}=\overline{startup}+\frac{data(v_i,v_j)}{\overline{rate}} \tag{3}$$

另外，为方便任务调度问题的描述，给出任务前驱节点集合、任务后继节点集合、最早开始时间、最早完成时间等变量的定义。$Prec(v_i)$ 表示节点 v_i 的直接前驱节点集合，当前任务只有在它全部前驱节点的通信数据到达后才能执行。将任务 v_i 的关键前驱节点记为 $fprec(v_i)$。$Succ(v_i)$ 表示节点 v_i 的直接后继节点集合。没有前驱节点的节点为入口节点。没有后继节点的节点为出口节点。若一个任务图中有两个或两个以上入口节点，则添加一个执行时间为 0 的伪入口节点，并分别用开销为 0 的边指向入口节点，生成新的具有唯一入口节点的 DAG 任务图。出口节点同理。

任务 v_i 在内核 p_m 上的最早开始时间为 $EST(v_i,p_m)$，定义见式(4)。其中，$PAT(p_m)$ 为处理器 p_m 最迟可执行时刻，即内核 p_m 上最后一个任务执行完毕的时间。

$$EST(v_i,p_m)=\max_{k\in Prec(v_i)}(PAT(p_m),\max(\max(EFT(v_k,p_n)+c(v_k,v_i),\max(EFT(v_k,p_m)))) \tag{4}$$

任务 v_i 在处理器内核 p_m 上的最早完成时间记为 $EFT(v_i,p_m)$，定义见式(5)。

$$EFT(v_i,p_m)=EST(v_i,p_m)+w(v_i,p_m) \tag{5}$$

2 优先级表调度算法分析

异构多核处理器任务调度已经被证明是一个 NP 完全问题[4]。现有的任务调度算法为启发式任务调度算法，主要包括优先级列表[5,6]、任务复制[7,8]和任务聚簇[9,10]，其中优先级列表调度算法因高性能代价比而获得广大学者的青睐，应用也最广泛。

Haluk Topcuoglu、Salin Hariri 和 Min-You Wu 在参考文献[2]中提出目前公认的任务调度效率较高的 HEFT(Heterogeneous Earliest Finish Time)算法和 CPOP(Critical Path on a Processor)算法。HEFT 算法根据 $rank_u$(从当前任务结点到出口结点的关键路径长度)的值对任务进行优先级排序，并将任务插入到具有最小执行时间的处理器内核上执行；CPOP 算法将 $rank_u$ 和 $rank_d$(从入口结点到当前节点的关键路径长度)的和作为权值来确定任务优先级，将任务分配到具有最小完成时间的处理器内核上。HEFT 算法和 CPOP 算法分别以任务的向前和向后关键路径作为优先级权值参数，这一定程度上起到了全局调度的效果，提升了任务执行效率，且 HEFT 算法优于 CPOP 算法。然而，HEFT 算法和 CPOP 算法均采用任务执行平均参数作为优先级权值计算参数，无法充分发挥异构平台性能，任务在各内核执行差异较大的情况下，势必会降低处理性能。并且这两种算法没有采用对于减少任务间通信时延行之有效的任务复制方法，放弃了获取更短调度时间的机会。

Prashanth C. Sai Ranga 和 Sanjeev Baskiyar 在参考文献[3]中提出关键任务优先调度算法 HCNF，算法的核心思想为：首先标记出任务图的关键路径，在每一步赋予关键路径节点最高的调度优先级，将其调度到使其结束时间最短的处理器上，并采取复制前驱的方法，减少任务之间的通信延迟时间。但 HCNF 算法仅对于关键路径上的任务给予最高优先级，对于非关键路径节点的优先级没有提供有效的排序算法，虽然缩短了关键路径的执行时间，但当非关键路径上任务结点调度失当时，势必增加任务图的整体调度长度，降低任务调度算法整体性能。

3 HIPLTS 算法

针对现有优先级列表任务调度算法中存在的优先级排序方法失当、处理器选择策略效率不高等问题，提出一种改进的优先级列表任务调度 HIPLTS 算法。HIPLTS 算法以表调度为主要思想，将任务调度问题分为 3 个部分。首先，为解决现有表调度算法中用平均值作为优先级参数导致在处理器内核上执行开销差异较大的任务不能优先执的问题，综合依赖任务调度通信特征，提出一种加权优先级排序方式；其次，改进传统任务调度中利用贪心思想选择处理器导致难以获得全局最优解的方式，提出一种根据当前格局计算权值、选择内核的方式；最后，在任务分配过程中，运用任务复制技术和区间插入技术，进一步优化调度过程，达到消减任务通信开销、提高处理器利用率的目的。

HIPLTS 算法对输入的处理器模型和依赖任务模型进行处理，最后形成任务映射图。该算法原理如图 1 所示。

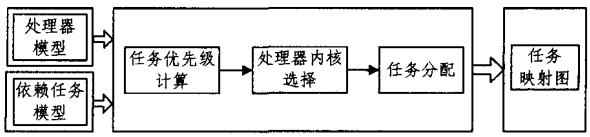

图 1 HIPLTS 算法原理图

3.1 任务优先级计算

本文针对现有异构平台算法中采用任务平均执行时间作为任务优先级参数，将异构问题转化为较简单的同构问题的不当做法，根据方差公式，提出一种以任务在各个处理器上的执行时间方差作为参数的优先级排序方式。任务的执行时间方差越大，说明任务在不同处理器内核上的执行时间差异越大，则应赋予其较高的优先级，以提高获得最优执行时间的概率。否则，赋予其较小的优先级。同时以任务通信平均开销时间作为第二个权值计算参数，通信平均时间开销较大的任务对于其后继任务影响也比较大，应赋予其较高的优先级级别。任务优先级权值 δ_i 定义如式(6)所示，其中 $J=|Succ(v_i)|$，表示任务节点 v_i 的后继节点个数。

$$\delta_i=\sum_{\substack{0\leqslant i,j\leqslant N-1\\1\leqslant m\leqslant M}}\frac{(w(v_i,p_m)-\overline{w_i})^2}{M}+\frac{\sum\limits_{v_j\in Succ(v_i)}\overline{c(v_i,v_j)}}{J} \tag{6}$$

3.2 处理器内核选择

在处理器选择阶段，针对现有算法中利用贪心思想选择处理内核，导致任务调度陷入局部最优解的问题，本文采用权值分配的原则，以获得全局最小任务执行时间为指导思想，利用当前格局下向后关键路径的执行时间为权值，选择权值较小的处理器内核进行分配。任务 v_i 的处理器内核选择权值 Ksp_{im} 定义如式(7)所示。其中，$CP\_KEY(v_i)$ 的值如式(8)所示，$cp(v_i)$ 为任务 v_i 的关键后继节点。

$$Ksp_{im}=\min_{1\leqslant m\leqslant M}(EST(v_i,p_m)+CP\_KEY(v_i)) \tag{7}$$

$$CP\_KEY(v_i)=\min_{j\in cp(v_i)}(CP\_KEY(v_j)+c(v_i,v_j)+w(v_i,p_m)) \tag{8}$$

3.3 任务分配过程及其优化

调度开始时，深度遍历 DAG 图，获得任务图的关键路径 CP，调度的每一步都赋予关键路径上节点最高优先级，对于非关键节点按照 δ_i 值降序进行优先级排序。若存在两个或多个 δ_i 值相等的就绪任务，选择后继结点数目最多的任务优先调度。若仍然不唯一，则随机调度。

选择优先级队列中优先级最高的任务进行调度，获取当前节点到出口节点的向后关键路径，计算处理器选择权值 Ksp_{im}，此时，若存在多个相同的可使 Ksp_{im} 获得最小值的处理器内核，则选择关键节点执行时间最小的处理器内核。

为进一步缩短任务执行时间，提高处理器利用率，算法在任务调度阶段采用任务复制技术和区间插入技术对调度过程进行优化。

1. 任务复制

若将当前任务的最佳前驱节点复制到目标内核上可以提前任务的最早执行时间，则复制最佳前驱节点。循环判断前驱任务节点是否满足任务复制的条件 $EFT(v_i,p_m)'<EFT(v_i,p_m)$，$EFT(v_i,p_m)'$ 如式(9)所示，若满足，则将当前任务节点复制到处理器内核上。

$$EFT(v_i,p_m)'=w(v_i,p_m)+\max(EST(v_{fprec(v_i)},p_m)+w_{fprec(v_i)},\max_{k\neq fprec(v_i)}(EFT(v_i,p_m)+c(v_k,v_i))) \tag{9}$$

复制优化过程如下：

```
1. FOR Prec(v_i) DO
2.   计算 EFT(v_i, p_m)
3.   计算 EFT(v_i, p_m)'
4.     IF EFT(v_i, p_m)'<EFT(v_i, p_m)
5.       复制 fprec(v_i)到 p_m 上
6.       更新 fprec(v_i)
7.       更新 EFT(v_i, p_m)'
8.       更新 EFT(v_i, p_m)
9.     END IF
10.  END FOR
```

2. 区间插入

在任务分配时，若任务符合以下条件：空闲区间起始时间大于任务的最早开始时间、保证前驱后继关系、在前驱节点之后执行以及空闲区间大于任务的执行时间，则采用区间插入分配方式验证任务是否满足式(10)。若满足则将任务分配到处理器空闲区间执行。Sps、Spe 分别为处理器内核 p_m 空闲区间的起始时间和结束时间。

$$\max(Sp_s,\max(\max_{\substack{v_k\in Prec(v_i)\\n\neq m}}(EFT(v_k,p_n)+c(v_k,v_i),EFT(v_k,p_m))+w(v_i,p_m))\leqslant Sp_e \tag{10}$$

HIPLTS 算法的任务分配流程如图 2 所示。

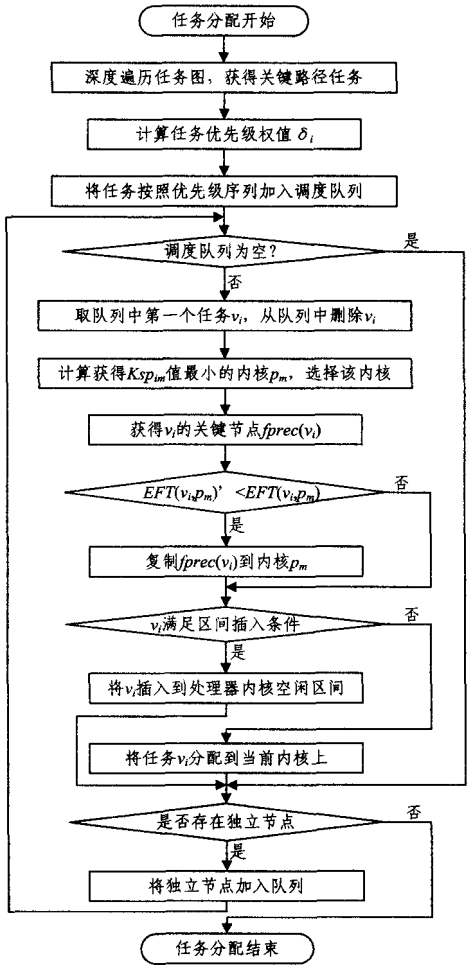

图 2 任务分配流程图

4 性能验证

为了验证 HIPLTS 算法的性能，本文采用实例分析和模拟实验两种方式对算法性能进行验证，并对 HEFT 算法、HCNF 算法以及 HIPLTS 算法的实验结果进行对比分析。

4.1 实例分析

通过具体 DAG 图的实例调度，说明算法的调度性能。图 3 是一个随机生成的任务图，每个圆圈代表一个任务节点，圆圈的中上部分数据表示任务序号，圆圈的中下部分数据表示任务的平均执行延迟时间(t/ms)；边上的数值表示边连接

的两个任务节点之间的通信延迟时间；任务节点 13 是由虚线连接的一个执行时延和通信时延都为 0 的伪出口节点。表 1 是图 3 中任务在异构 3 核全互联结构处理器上的执行时间。HIPLTS 算法对图 3 的调度结果如图 4 所示。

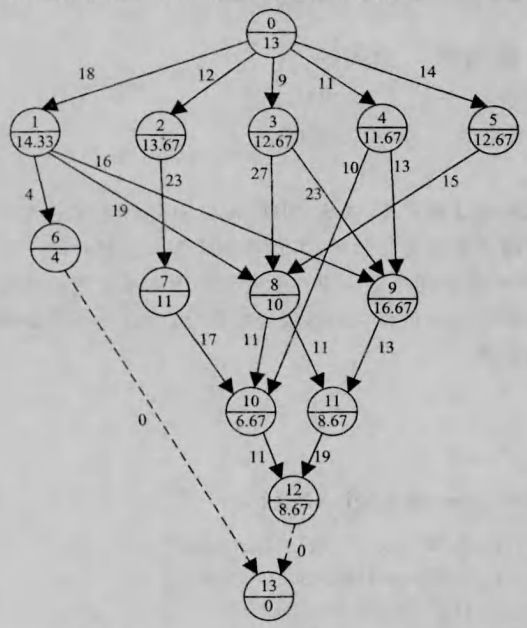

图 3　DAG 任务图

表 1　任务在 3 个不同处理器内核的执行时间

任务编号 \ 内核	p_0	p_1	p_2
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	5	4	3
7	7	15	11
8	5	11	14
9	18	12	20
10	5	8	7
11	9	7	10
12	12	6	8

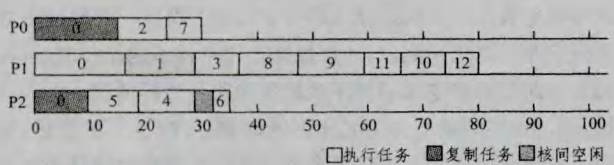

图 4　HIPLTS 算法调度甘特图

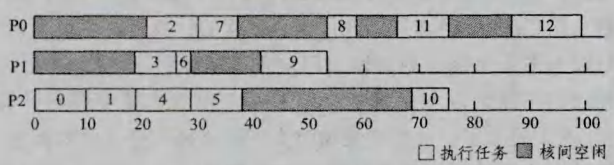

图 5　HEFT 算法调度甘特图

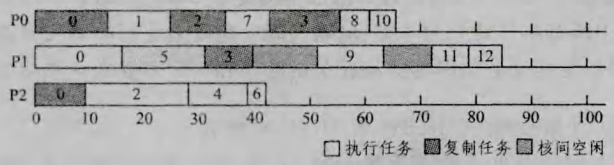

图 6　HCNF 算法调度甘特图

图 5 和图 6 分别为 HEFT 和 HCNF 算法对图 2 所示 DAG 图的调度结果甘特图，甘特图中形象示出任务调度属性以及任务图的完成时间。由图 4、图 5 和图 6 可知，HIPLTS 算法的调度长度为 80、HEFT 算法为 99、HCNF 算法为 85，HIPLTS 算法调度长度最短。另外，HEFT 算法、HCNF 算法在调度过程中，存在多处处理器空闲，处理器利用率较低。经分析，对于图 3 的调度中，HIPLTS 算法相对最优。

4.2　模拟实验

为进一步验证算法性能，利用模拟实验对随机生成的大量任务图进行调度，记录调度中间值和结果，并选用实际调度长度和关键路径任务执行时间之比，即调度下界比（schedule length ratio，SLR）与加速比 Speedup 作为算法的性能评估参数。将 HIPLTS 算法与已有 HEFT 算法和 HCNF 算法分别在不同任务数和不同通信计算比（communication computation ratio，CCR）的情况下进行性能对比分析。

本文采用随机任务图产生器产生 DAG 任务图作为实验测试用例。与此同时，为控制产生的 DAG 任务图的形状和大小等属性，对任务图的参数取值范围进行设定。设定任务个数 $N=\{20,35,50,65,80,95,110\}$；任务图中节点的最大出度 $\alpha=\{1,2,5,10,100\}$；任务的通信时间与执行时间比值参数 $CCR=\{0.1,0.2,0.5,1.0,2.0,5.0,10.0\}$；任务在不同处理器内核上执行时间的差异度参数 $\beta=\{0.1,0.5,1,1.5,2\}$。通过对以上 4 类参数的设定，将产生 1225 组任务图类型。从每组类型中随机选择 10 个任务图，可生成 12250 个任务图。对产生的 12250 个任务图在模拟平台上进行调度，以评测算法性能。

通过对实验结果数据进行分析，得到算法在不同任务数情况下的平均加速比和平均 SLR 结果对比图，分别如图 7 和图 8 所示；不同 CCR 情况下的平均加速比和平均 SLR 结果对比图，分别如图 9 和图 10 所示。

如图 7 所示，在不同任务数目下，HIPLTS 算法表现出较好的加速比性能，且在任务数递增的情况下也保持稳定的性能。如图 8 所示，在不同任务数下，HIPLTS 算法的平均 SLR 相对较低，且在任务数不断增加的情况下增长趋于平缓，这得益于本文对关键节点的优先处理与任务优先级的设定方式。与 HEFT 算法和 HCNF 算法相比，HIPLTS 算法加速比 Speedup 分别提高约 15.63%和 9.98%，调度下界比 SLR 分别降低约 13.51%和 9.39%。

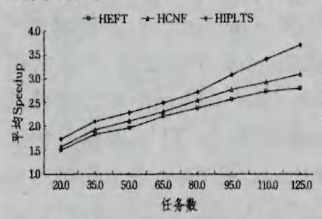

图 7　不同任务数下的加速比对比图

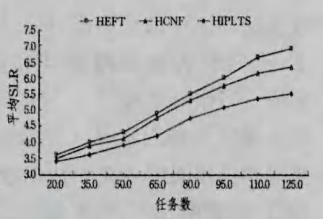

图 8　不同任务数下的 SLR 对比图

如图 9 和图 10 所示，在不同 CCR 下，HIPLTS 算法较高地提升了任务调度效率。在不同 CCR 情况下，随着 CCR 的增大，任务图的通信比重加大，多核处理器下的并行任务调度需要花费更多的处理资源处理任务之间的数据传输，而串行情况下，则不存在数据通信的时间开销，故 3 个算法的加速比均有所下降。相对于其他两种算法，HIPLTS 算法始终保持较高的加速比，具有较强的并行化处理能力。与 HEFT 算法

（下转第 36 页）

法巧妙结合，充分共用同一个解压电路能够达到减少 ROM 开销和降低测试成本的目的。聚类移位压缩解压电路结构简单，易于实现。实验结果显示了聚类移位压缩的有效性。未来将研究容错移位算法，更充分地利用聚类压缩的特性来增加移位压缩的效率。

参 考 文 献

[1] Semiconductor Industry Association(SIA). Test and Test Equipment, International Technology Roadmap for Semiconductors(ITRS) 2011 Update [OL]. http://www.itrs.net/Links/2011ITRS.htm

[2] 李立健，赵瑞莲. 减少多种子内建自测试方法硬件开销的有效途径[J]. 计算机辅助设计与图形学学报，2003，15(6)：662-666

[3] Han Yin-he, Li Xiao-wei, Li Hua-wei, et al. Embedded Test Resource for SoC to Reduce Required Tester Channels Based on Advanced Convolutional Codes[J]. IEEE Transactions on Instrumentation and Measurement. 2006, 55(2): 389-399

[4] Conroy Z, Li Hui, Balangue J. Built In Self Test(BIST) Survey-an industry snapshot of HVM component BIST usage at board and system test [C] // Electronic Manufacturing Technology Symposium(IEMT), 34th IEEE/CPMT International. Melaka, 2010: 1-6

[5] Swaminathan S, Chakrabarty K. On Using Twisted-Ring Counters for Test Set Embedding in BIST[J]. Journal of Electronic Testing, 2001, 17(6): 529-542

[6] Hellebrand S, Liang H-G, Wunderlich H-J. A Mixed Mode BIST Scheme Based on the Reseeding of Folding Counters[C] // Atlantic City. Proceedings IEEE International Test Conference. 2000: 778-784

[7] Liang Hua-guo, Hellebrand S, Wunderlich H-J. Two-Dimensional Test Data Compression for Scan-Based Deterministic BIST [J]. Journal of Electronic Testing, 2002, 18(2): 159-170

[8] Hashempour H, Lombardi F. Compression of VLSI test data by arithmetic coding[C] // Cannes, France. Defect and Fault Tolerance in VLSI Systems. 2004: 150-157

[9] Balakrishnan K J, Touba N A. Relating entropy theory to test data compression[C] // Test Symposium, ETS 2004. Proceedings. Ninth IEEE European. IEEE Press, Corsica, France, 2004: 94-99

[10] Chandra A, Chakrabarty K. System-on-a-chip test data compression and decompression architectures based on Golomb Codes [J]. IEEE Trans. on Computer-Aided Design, 2001, 20(3): 355-368

[11] Chandra A, Chakrabarty K. Frequency-Directed Run-Length Codes with Application to System-on-a-chip Test Data Compression [C] // Proc. VLSI Test Symposium. IEEE Computer Society. California, 2001: 42-47

[12] Gonciari P T, Al-Hashimi B M, et al. Variable-Length Input Huffman Coding for System-on-a-chip Test[J]. IEEE Trans. on computer-Aided Design, 2003, 22(6): 783-796

[13] Lee H K, Ha D S. On the Generation of Test Patterns for Combinational Circuits [R]. Technical Report No. 12_93. Virginia: Virginia Polytechnic Institute and State University, Department of Electrical Engineering, 1997

(上接第 23 页)

和 HCNF 算法相比，HIPLTS 算法加速比 Speedup 分别提高约 13.75%和 8.49%，调度下界比 SLR 分别降低约 14.02%和 9.25%。HIPLTS 算法较高地提升了任务调度效率。

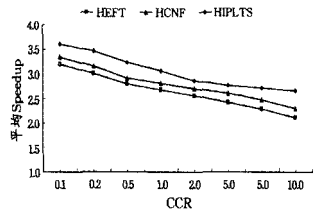

图 9 不同 CCR 下的加速比对比图

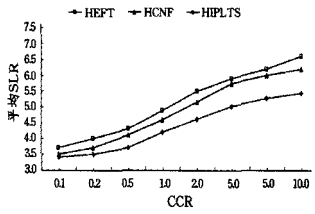

图 10 不同 CCR 下的 SLR 对比

结束语 如何将任务分配到处理器内核上进行处理是能否充分发挥多核处理器性能的首要问题。本文通过对异构多核处理器任务调度问题的深入分析与研究，提出一种改进的优先级列表任务调度算法，该算法采用具有异构平台匹配性的优先级排序方式和全局处理器选择方式，克服现有优先级列表任务调度算法中存在的缺点和不足，并在任务分配时采取任务复制技术和区间插入技术对调度过程进行进一步优化。实验验证结果表明，该算法有效减少了任务调度长度，提高了处理器利用率。

参 考 文 献

[1] 张建军，宋业新，旷文. 基于异构环境的 Out_Tree 任务图的调度算法[J]. 计算机科学，2013，40(4)：107-111

[2] Topcuoglu H, Hariri S, Wu Min-you. Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing [J]. IEEE Transactions on parallel and distributed systems, 2002, 13(3): 260-274

[3] Prashanth C, Sai Ran-ga. Algorithms for task scheduling in heterogeneous computing environments[D]. Alabama: Auburn University, 2006

[4] Ullman J D. Np-complete scheduling problem [J]. Journal of Computer and System Sciences, 1975, 10(3): 384-393

[5] Hagras T, Janecek J. A high performance low complexity algorithm for compile-time task scheduling in heterogeneous systems [J]. Parallel computing, 2005, 31(7): 653-670

[6] Daolud M I, Kharma N. A high performance algorithm for static task scheduling in heterogeneous distributed computing systems [J]. Journal of parallel and distributed computing, 2008, 68(4): 399-409

[7] 何琨，赵勇，黄文奇. 基于任务复制的分簇与调度算法[J]. 计算机学报，2008，31(5)：733-740

[8] Darbha S, Agrawal D P. Optimal scheduling algorithm for distributed memory machines[J]. IEEE transactions on parallel and distributed systems, 1998, 9(1): 87-95

[9] 王小非，方明. 一种基于调度簇树的周期性分布实时任务调度算法[J]. 计算机科学，2007，34(3)：256-261

[10] 曹仰杰，钱德沛，伍卫国，等. 众核处理器系统核资源动态分组的自适应调度算法[J]. 软件学报，2012，23(2)：240-252