

嵌入式多核系统中三维立体调度模型的研究

黄姝娟¹, 朱怡安², 刘白林¹, 肖锋¹

(1.西安工业大学 计算机科学与工程学院, 陕西 西安 710021; 2.西北工业大学 计算机学院, 陕西 西安 710072)

摘要:针对嵌入式多核平台下分区调度算法不能满足系统高效的需求,提出一种三维立体调度模型,该模型根据任务各项参数将任务划分为可调和任务和不可调和任务,并将其执行区域用面积表示,根据区域的特征分为执行区域、干扰区以及空闲区。利用干扰区的特征将不同性质的任务分派到不同的处理器核上运行。实验表明,该方法无论在提高系统利用率还是在吞吐量方面都优于 PEDF。

关键词:嵌入式系统;多核;调度算法;调度模型;实时任务

中图分类号:TP302 **文献标志码:**A **文章编号:**1000-2758(2018)05-1020-06

嵌入式多核平台下,任务调度是系统的核心,如何优化调度算法,充分发挥多核处理器性能优势,是当今研究的主要方向^[1-3]。当前多核实时任务的调度算法主要分为全局调度算法(global scheduling algorithm)^[4-6]和分区调度算法(partitioned scheduling algorithm)^[7-8]2大类。GEDF(global earliest deadline first)^[9]和 PEDF(partitioned earliest deadline first)^[10],是2类调度算法的典型代表。对于全局调度算法来说,如果一个任务不允许抢占,而另外一个任务必须执行,那么就要将该任务迁移到其他核上执行^[11]。这种迁移开销比较大。对于分区调度算法而言,划分在一个范围内的任务只能在同一个处理器核上调度,虽避免了迁移,简化了问题^[12],但对划分的精确度有很高要求,划分不好会直接影响丢失时限的任务数量。很多情况下任务划分方法是很棘手的问题。为此,研究者提出了半分区调度算法(semi-partitioned scheduling algorithm)^[13],该方法将全局调度算法与分区调度算法相结合,旨在减少迁移开销的同时减少进行划分的任务数量,降低划分难度。尽管如此,该类算法仍然存在迁移开销和划分方法如何精确的问题。为此,本文提出一种三维立体调度模型和调度算法,该方法以区域面积作为划分方法,降低划分的难度,提高划分算法的精确度,在提高系统效率的同时,降低丢失时限的任务

数量。

1 三维调度模型

1.1 实时周期任务模型

假设一个包含 n 个实时周期任务的嵌入式多核系统 $I = \{T_0, T_1, T_2 \dots T_n\}$, 其中 T_i 为一个四元组 $T_i(R_i, C_i, D_i, P_i)$, R_i 表示该任务首次发布时刻, C_i 表示该任务的 WCET(worst case execution time), D_i 表示相对时限且 $D_i \geq C_i$, P_i 表示该任务的执行周期且 $P_i \geq D_i \geq C_i$ 。用 $Job_{i,j}(r_{i,j}, c_{i,j}, d_{i,j})$ 表示 T_i 的第 j 次执行, 其中 $r_{i,j}$ 为 $job_{i,j}$ 的发布时间 $r_{i,j} = R_i + (j - 1)P_i$, $c_{i,j}$ 为 $job_{i,j}$ 的执行时间 $c_{i,j} = C_i$, $d_{i,j}$ 为 $job_{i,j}$ 的绝对时限 $d_{i,j} = r_{i,j} + D_i = R_i + (j - 1)P_i + D_i$ 。

1.2 三维调度模型

定义 1 对于一个实时周期任务 $T_i(R_i, C_i, D_i, P_i)$, 如果 $C_i = D_i$ 则称该任务为不可调和任务; 如果 $C_i < D_i$, 则称该任务为可调和任务; 对应的 job 称为可调和 job。

定义 2 对于一个可调和任务 $T_i(R_i, C_i, D_i, P_i)$, 若用 hf_i 表示该任务的调和因子(harmonic factor), 则 $hf_i = D_i - C_i$ 。则 T_i 的任何一个 $job_{i,j}$ 都会有 $hf_i + 1$ 个调和区间 $HF_{i,j} = \{[r_{i,j} + k, r_{i,j} + c_{i,j} + k]\}$

$k = 0, 1, 2, \dots, hf_i$ 。对应的每个调和区间衍生出来的 job 称为衍生 job, 记为 $job_{i,j,k}(r_{i,j,k}, c_{i,j,k}, d_{i,j,k})$, 其中, $r_{i,j,k} = r_{i,j} + k$, ($k = 0, 1, 2 \dots hf_i$), $c_{i,j,k} = c_{i,j} = C_i$, $d_{i,j,k} = d_{i,j}$ 。而对于可以被抢占性的 job 在区间 $[R_i, D_i]$ 之间可以衍生出来 $l = D_i$ 个 $job_{i,j,k}(r_{i,j,k}, c_{i,j,k}, d_{i,j,k})$ 其中, $r_{i,j,k} = r_{i,j} + k$, ($k = 0, 1, 2 \dots d_{i,j} - 1$), $\sum_{k=1}^l c_{i,j,k} = c_{i,j} = C_i, d_{i,j,k} = d_{i,j}$ 。那么把这 l 个 job 整体称为衍生 l -job。

定义 3 对于任意实时调度系统 $I = \{T_0, T_1, T_2 \dots T_n\}$, 必然存在一个三维调度空间 $\Omega = \{x, y, z\}$, 其中 x 表示时间轴, y 表示当前运行在该平面的任务数, z 表示处理器核数。对于任何一个实时周期任务 T_i 的任何一次执行 $job_{i,j}$, 该调度空间上必然会存在该 job 执行的一个矩形区域 $\Phi_{i,j}$, 该区域的 4 个顶点坐标分别为 $A_{i,j}(r_{i,j}, 0, z_{i,j}), B_{i,j}(r_{i,j} + c_{i,j}, 0, z_{i,j}), C_{i,j}(r_{i,j}, y_{i,j}, z_{i,j}), D_{i,j}(r_{i,j} + c_{i,j}, y_{i,j}, z_{i,j})$ 其中 $z_{i,j}$ 表示给 $job_{i,j}$ 分配的处理器号, $y_{i,j}$ 表示在第 $z_{i,j}$ 平面上所分配的任务的个数; 把该矩形区域 $\Phi_{i,j}$ 称为 $job_{i,j}$ 对应在该三维调度空间 Ω 上的执行区域, 简单记为 $\Phi_{i,j} = \{(r_{i,j}, r_{i,j} + c_{i,j}, y_{i,j}, z_{i,j})\}$ 。

对应于该矩形 $\Phi_{i,j}$ 区域上的面积称为执行区域面积即 $S_{i,j} = c_{i,j} * y_{i,j}$ 如图 1 所示。

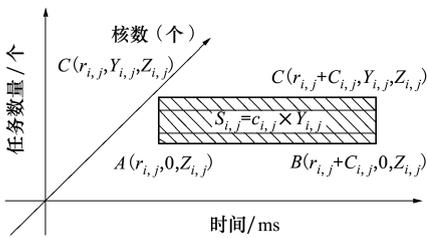


图 1 $job_{i,j}(r_{i,j}, c_{i,j}, d_{i,j})$ 的执行区域

定义 4 对于在同一 Z 平面上任何 2 个 $job_{i,j}$ 和 $job_{v,w}$, 如果它们的执行区域面积没有任何交叉覆盖情况, 则称 $job_{i,j}$ 和 $job_{v,w}$ 相互独立。如果在某段时间 t 内, 2 个任务 T_i 和 T_j 在该时间段内所有需要执行的 job 都相互独立, 则称这 2 个任务在时间段 t 内相互独立。如果 t 为无穷大都能满足, 则 T_i 和 T_j 称为永久相互独立任务。

定义 5 对于任何 2 个 $job_{i,j}$ 和 $job_{v,w}$ 的执行区域面积存在交叉覆盖情况, 则称这 2 个 job 具有互干扰性。交叉覆盖的区域称为干扰区。相对应的干扰区面积 S^A 为交叉覆盖面积, 相对应的 $job_{i,j}$ 对 $job_{v,w}$ 的干扰因子为 $\xi_{i,j/v,w} = S^A/S_{v,w}$ 而相对应的 $job_{v,w}$ 对 $job_{i,j}$

的干扰因子为 $\xi_{v,w/i,j} = S^A/S_{i,j}$ 。

定义 6 对于任何 2 个具有干扰区的 job, 如果至少有一个是可调和 job, 且其衍生出来的任意一个 job 或者是 l -job 中存在执行区域与另外一个 job 或其任意一个衍生 job 或者是 l -job 相互独立, 则称这干扰区为可调和干扰区, 否则称为不可调和干扰区。

定义 7 k 重干扰区 如果在某段时间 t 内, 存在 k 个 job 的执行区都覆盖了同一干扰区且该干扰区对于任何一个 job 都不可调和, 那么称该干扰区为 k 重干扰区。

定义 8 如果在 $Z = z_i$ 平面上存在某个区域既不是执行区也不是干扰区, 则该区域称为闲置区。

定义 9 给定一个三维空间 $\Omega = \{x, y, z\}$ 在某段时间 t 内, 假设 x_t, y_t 和 z_t 分别为三维空间在该时间段内 3 个坐标轴所能取到的最大值, 则在任意的 $Z = z_i (i = 1, 2, \dots, t)$ 的平面内, 所有单位面积之和称为可调度空间面积, 记为 $S = x_t * y_t$ 。

1.3 可调度性证明

由上述定义可知, 调度空间面积为执行区面积和闲置区面积之和。对于 $Z = z_i$ 的平面, 任何一个区域无非为干扰区、执行区和闲置区三者之一, 其中干扰区必定是执行区。

推论 1 如果系统 I 中所有任务之间在任何时间段不产生不可调和干扰区, 则该系统所有任务可以被调度在同一个处理器上。

证: 如果在任何时间段内不产生干扰区, 就说明所有任务都有自己的执行区且执行区并不被相互覆盖, 那么所有任务在任何时间段内产生的 job 都能在时限之前被调度完成。故可以被分配到同一个处理器上。

如果在任何时间段内产生的干扰区都是可调和干扰区, 就说明所有任务都有自己的执行区且执行区通过调和和区间衍生 job 或者 l -job 调和后并不被相互覆盖, 那么所有任务在任何时间段内产生的 job 都能在时限之前被调度完成。故可以被分配到同一个处理器上。

推论 2 若在某段时间 t 内, 系统 I 产生 k 重干扰区, 则在该时间段内必须有 k 个处理器核才能完成调度。

证: 因为产生了 k 重干扰区, 说明有 k 个 job 的执行区域都覆盖在同一区域, 且没有任何可调和空间, 若采用小于 k 个处理器核来调度, 则必然还会产生

无法调和的干扰区,一定会存在一些 job 无法在时限之前调度完成,因此必须采用 k 个处理器核来调度。

可以根据区域覆盖情况来判断 2 个 job 是否有干扰区。如果任务是可以互相抢占的,可以根据任务的可调和性来判断干扰区是否为不可调和干扰区。下面讨论都是可抢占的实时任务。

定理 假设在不考虑上下文切换和中断等额外开销的情况下,任何 2 个 $job_{i,j}(r_{i,j},c_{i,j},d_{i,j})$ 和 $job_{v,w}(r_{v,w},c_{v,w},d_{v,w})$ 能够被调度在同一平面 z 上的充分必要条件是区域 $\Phi = \{ \min(r_{i,j},r_{v,w}), \max(d_{i,j},d_{v,w}), 2,z \}$ 的面积大于等于 $job_{i,j}$ 的执行区域面积与 $job_{v,w}$ 的执行区域面积之和。

证明 先证充分条件:

$job_{i,j}(r_{i,j},c_{i,j},d_{i,j})$ 和 $job_{v,w}(r_{v,w},c_{v,w},d_{v,w})$ 的 $r_{i,j}, r_{v,w}, d_{i,j}, d_{v,w}$ 有如下 4 种情况:① $r_{i,j} \leq r_{v,w}$ 且 $d_{i,j} \leq d_{v,w}$;② $r_{i,j} > r_{v,w}$ 且 $d_{i,j} > d_{v,w}$;③ $r_{i,j} > r_{v,w}$ 且 $d_{i,j} \leq d_{v,w}$;④ $r_{i,j} \leq r_{v,w}$ 且 $d_{i,j} > d_{v,w}$

先证明(1) 因为 $d_{v,w} - r_{i,j} \geq d_{v,w} - r_{v,w} \geq c_{v,w}$ 且 $d_{i,j} - r_{v,w} \geq d_{i,j} - r_{i,j} \geq c_{i,j}$ 那么由条件知 $\Phi = \{ (r_{i,j}, d_{v,w}, 2, z) \}$ 的区域面积 $S = (d_{v,w} - r_{i,j}) \times 2 \geq S_{i,j} + S_{v,w}$ 其中 $S_{i,j} = c_{i,j} \times 2; S_{v,w} = c_{v,w} \times 2$ 分别是 $job_{i,j}$ 和 $job_{v,w}$ 的执行面积,如果没有干扰区,则根据推论 1 在时间段 $[r_{i,j}, d_{v,w}]$ 内 $job_{i,j}$ 和 $job_{v,w}$ 可以调度在同一个处理器核上。若有干扰区如图 2a) 所示,则可以在 Ω 中先满足 $job_{i,j}$ 的执行区域,保证 $job_{i,j}$ 在 $d_{i,j}$ 之前执行完毕。剩余的区域划出与 $S_{v,w}$ 等量的区域作为 $job_{v,w}$ 的执行区域,则能保证 $job_{v,w}$ 在 $d_{v,w}$ 之前完成,如图 2b) 所示。那么 $job_{i,j}$ 和 $job_{v,w}$ 就可以调度在同一个处理器核上。

同理可以证明 ②③④。

证明必要条件

根据推论 1 可知,如果可以调度在同一个处理器上则说明不存在干扰区或存在可调和的干扰区,如果不存在干扰区,则在 $[\min(r_{i,j},r_{v,w}), \max(d_{i,j},d_{v,w})]$ 的时间段内 $Job_{i,j}$ 和 $Job_{v,w}$ 的执行区不存在区域覆盖的情况,则 $Job_{i,j}$ 和 $Job_{v,w}$ 的执行区域面积之和必定小于等于 $\Phi = \{ \min(r_{i,j},r_{v,w}), \max(d_{i,j},d_{v,w}), 2,z \}$ 的区域面积。

如果存在的是可调和干扰区,则可以通过调和区间在时间段 $[\min(r_{i,j},r_{v,w}), \max(d_{i,j},d_{v,w})]$ 内化解为不互相覆盖的区域,则它们的执行区必定存在该时间段的区域内, $Job_{i,j}$ 和 $Job_{v,w}$ 的执行区域面积

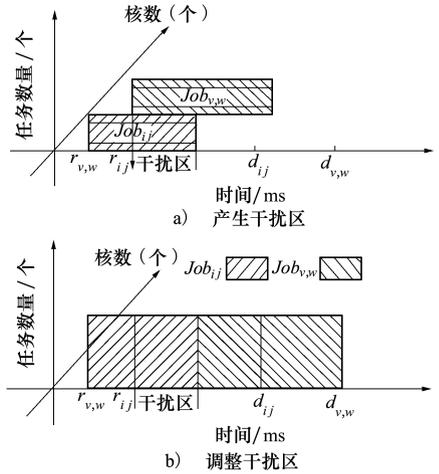


图 2 干扰区产生 / 调整图

之和必定小于等于 $\Phi = \{ \min(r_{i,j},r_{v,w}), \max(d_{i,j},d_{v,w}), 2,z \}$ 的区域面积。

得证。

推论 3 假设在不考虑上下文切换和中断等额外开销的情况下,任何一个 $Job_{i,j}(r_{i,j},c_{i,j},d_{i,j})$ 能够被调度在平面 z 上的充分必要条件是将其和平面上已经分配的 y 个 job 满足区域 $\Phi = \{ \min(r_{i,j}, r_{v,w}, \dots, r_{p,q}), \max(d_{i,j}, d_{v,w}, \dots, d_{p,q}), y + 1, z \}$, 的面积大于等于 z 平面上的所有 job 以及 $job_{i,j}$ 在内的执行区域面积之和,即 $S_{\Phi} \geq \sum_{i=1}^y S_i + S_{i,j}$ 。

2 调度算法

本文所涉及的分区调度算法是根据实时任务的各项参数特征,计算出所有实时任务的公共周期。在该公共周期内找出实时任务所要执行的 job。由文献[8]可以知道,如果该系统所有实时任务的 job 在公共周期内可以被调度成功,那么这些实时任务在所有周期内都能被调度成功。

系统开始初始化 x, y, z 3 个参数代表 3 个坐标轴,分别表示时间、任务个数以及处理器个数;第一次找出发布时间最近的 1 个 job 分配到 $z = 1$ 的平面上,按照定义 3 计算出该 job 的执行区域,再依次找出相应的 job,根据定理和推论 3 从 $z = 1$ 的平面开始判断新加入的 job 是否能被调度到该平面上,直到将所有新加入的 job 分配到相应的平面上,程序结束。具体算法流程图如图 3 所示。

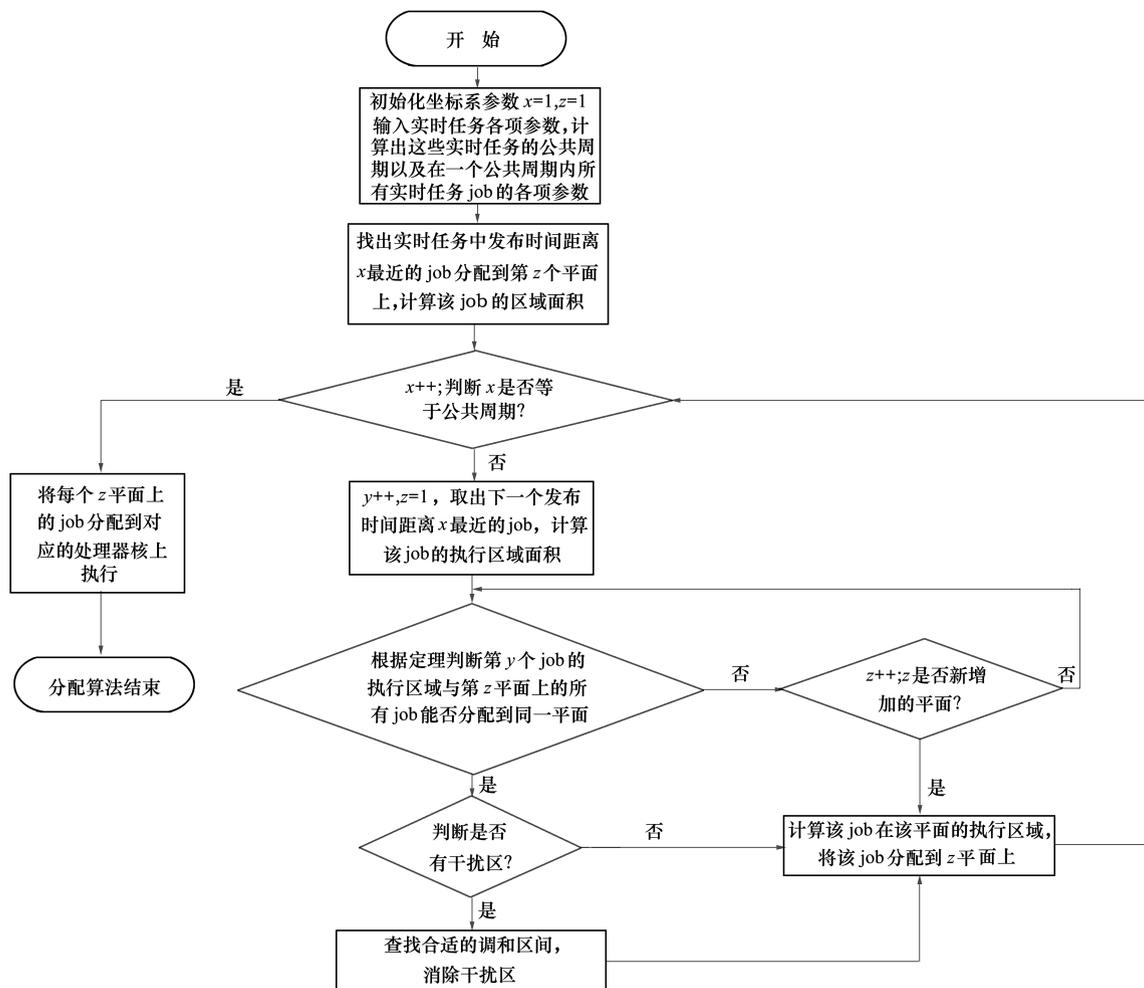


图 3 三维调度算法主流程图

在整个流程中,比较难的步骤是如何找出相应的调和区间以便消除干扰区。

方法如下:

1) 从该平面找出所有可调和 job 和不可调和 job 分别放入不同的队列中,将不可调和任务的执行区间放入集合 H 中;

2) 判断可调和队列是否为空? 若是,则返回集合 H 。若不是,则转入步骤 3)。

3) 从可调和任务队列中取出时限最小的 job, 从该 job 的可调和区间找出一个与集合 H 中的所有执行区间不覆盖的可调和区间,放入集合 H 中; 转 2。

具体算法流程如图 4 所示。

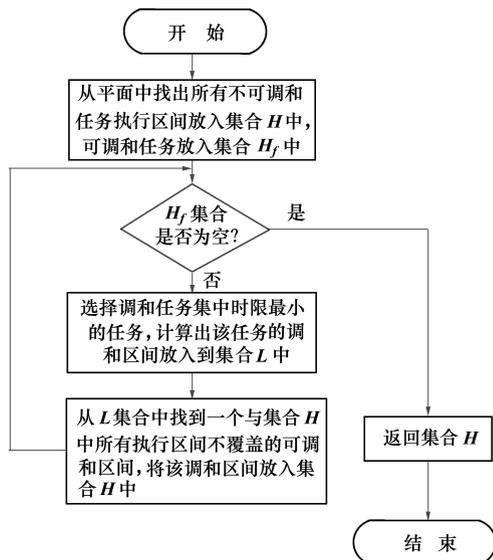


图 4 查找合适的调和区间算法流程图

3 实验结果

本文测试的环境是在一个 Intel (R) Core (TM) 2Quad Q8400 多核平台上。将所提出的三维调度算法和 PEDF 算法进行比较,测试方法是随机产生 1 000 个任务集,每个任务集中产生 50 个参数不等的实时周期任务,所有周期任务都满足时限小于或等于周期,且执行时间小于时限。在整个仿真实验过程中,为了描述算法之间的性能差异,采用多次模拟求平均值的方法按照在某段时间内,系统吞吐率,

丢失时限的任务数量所占总任务数的比例,核利用率 3 个方面进行性能对比,如图 5 和表 1 所示。

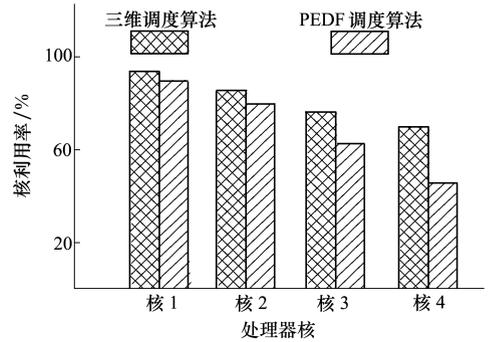


图 5 2 种算法核利用率

表 1 2 种算法的系统利用率和丢失时限任务数所占总任务的比例

随机抽取的 任务集	系统吞吐率		丢失时限的任务数占总任务数的比例/%	
	三维模型调度算法	PEDF 调度算法	三维模型调度算法	PEDF 调度算法
1	0.911	0.892	5.87	11.68
2	0.913	0.903	5.92	11.54
3	0.912	0.893	5.83	11.73
4	0.907	0.895	6.23	11.96
5	0.903	0.892	6.02	11.88
6	0.911	0.854	6.14	11.98
7	0.892	0.883	6.21	12.01
8	0.921	0.911	6.55	12.23
9	0.906	0.890	6.43	12.41
10	0.914	0.892	6.28	12.22

从图 5 中可以看出,随着核数的增加,采用三维调度算法比 PEDF 算法在核利用率方面要更加充分。从表 1 可以看出,在系统吞吐率方面三维调度算法相比 PEDF 算法优势不是很明显。这是因为寻找调和区花费的时间比较长,开销较大,但在丢失时限的任务数比例方面,三维调度算法明显占据优势。这种提高对于解决减少丢失时限的任务数量来说,是能够满足某些实际需要的。由于目前实验条件限制,运行的核数量仅为 4 个,以后将在核数更多的机器上进行实验,并会进一步优化寻找调和区的算法,来验证三维调度算法的优势。

4 结 论

本文设计了一种新的三维调度模型,在该模型中确立了任务之间的相互独立和相互干扰关系,并根据调度面积设定了调度方案,给出了相应的调度算法。通过实验,该算法在核利用率、系统吞吐量以及丢失时限的任务数方面更优于 PEDF 算法。这为以后在异构多核平台下的实时任务调度提供了新的思路。

参考文献:

[1] Tong G, Liu C. Supporting Soft Real-Time Sporadic Task Systems on Uniform Heterogeneous Multiprocessors with No Utilization Loss[J]. IEEE Trans on Parallel and Distributed Systems, 2016, 27(9): 2740-2752

[2] 康鹏,刘从新,沈绪榜. 一种基于分组的多核嵌入式实时调度算法[J]. 微电子学与计算机, 2016, 33(10): 32-35
Kang Peng, Liu Congxin, Shen Xubang. Multicore Embedded Real-Time Scheduling Algorithm Based on Gang Scheduling[J].

- Microelectronics & Computer, 2016, 33(10): 32-35 (in Chinese)
- [3] Giovanni Gracioli, Real-Time Operating System Support for Multicore Application [D]. Universidade Federal de Santa Catarina, 2014
- [4] Yang K, Anderson J H. On the Soft Real-Time Optimality of Global EDF on Multiprocessors: from Identical to Uniform Heterogeneous[C]//Proceedings of the 21st IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, Hong Kong, China, 2015: 1-10
- [5] Baruah S, Bonifaci V, Marchetti-Spaccamela A. The Global EDF Scheduling of Systems of Conditional Sporadic DAG Tasks[C]//Proceedings of the 27th Euromicro Conference on Real-Time Systems, Lund, Sweden, 2015: 222-231
- [6] Zhang Y, Guo Z, Wang L, et al. Integrating Cache-Related Preemption Delay into GEDF Analysis for Multiprocessor Scheduling with On-Chip Cache[C]//Proceedings of the The 14th IEEE International Conference on Embedded Software and Systems Sydney, Australia, 2017: 815-822
- [7] Rhaïem G, Gharsellaoui H, Ahmed S B. A Novel Proposed Approach for Real-Time Scheduling Based on Neural Networks Approach with Minimization of Power Consumption[C]//Proceedings of the 2016 World Symposium on Computer Applications & Research(WSCAR), Cairo, Egypt, 2016: 98-103
- [8] 谷传才,关楠,于金铭,等. 多处理器混合关键性系统中的划分调度策略[J]. 软件学报, 2014, 25(2): 284-297
Gu Chuancal, Guan Nan, Yu Jinming, et al. Partitioned Scheduling Policies on Multi-Processor Mixed-Criticality Systems[J]. Journal of Software, 2014, 25(2): 284-297 (in Chinese)
- [9] Xi Sisu, Xu Meng, Lu Chenyang, et al. Christopher Gill, Oleg Sokolsky, Insup Lee, Real-Time Multi-Core Virtual Machine Scheduling in Xen[C]//Proceedings of the International Conference on Embedded Software(EMSOFT), New Delhi, India, 2014: 1-10
- [10] AbusayeedSaifullah, David Ferry, Jing Li, et al. Parallel Real-Time Scheduling of DAGs[J]. IEEE Trans on Parallel and Distributed Systems, 2014, 12(25): 3242-3252
- [11] Liy Jing, Chenx Jianjia, KunalAgrawaly, et al. Analysis of Federated and Global Scheduling for Parallel Real-Time Tasks[C]//Proceedings of the 26th Euromicro Conference on Real-Time Systems(ECRTS), Madrid, Spain 2014: 85-96
- [12] Saranya N, Hansdah R C. Dynamic Partitioning Based Scheduling of Real-Time Tasks in Multicore Processors[C]//Proceedings of the IEEE 18th International Symposium on Real-Time Distributed Computing, Auckland, New Zealand, 2015: 190-197
- [13] James H Anderson, Jeremy P Erickson, UmaMaheswari C Devi, et al. Optimal Semi-Partitioned Scheduling in Soft Real-Time Systems[C]//Proceedings of the 20th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications(RTCSA), Chongqing, China, 2014: 1-16

Research on Three Dimensional Scheduling Model for Embedded Multi-Core System

Huang Shujuan¹, Zhu Yi'an², Liu Bailin¹, Xiao Feng¹

(1.School of Computer Science and Engineering, Xi'an Technological University, Xi'an 710021, China;)
(2.School of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China)

Abstract: This paper proposes a new three-dimensional scheduling model which can divide the tasks into harmonic tasks and non-harmonic tasks for the high demands of embedded multicore platform. According to the characteristic parameters of the tasks and make the value of the rectangular area as the attribute of the execution region which is divided into executive region, interference region and free region with the characteristic of the area. By using these attributes of the different region, the tasks are allocated to different cores. Experimental results show that the proposed method is more fully optimizing the system utilization and throughput than PEDF.

Keywords: embedded system; multi-core; scheduling algorithm; scheduling model; real-time tasks