软件学报 ISSN 1000-9825, CODEN RUXUEW
2024,35(11):5306−5318 [doi: 10.13328/j.cnki.jos.007036] [CSTR: 32375.14.jos.007036]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

并行机器中基于干扰时间的间歇实时任务分区 DM 调度*

刘洪标[1], 宋程昊[2], 王婷煜[1], 姜菁菁[2], 乔 磊[2], 杨孟飞[3]

[1](西安电子科技大学 计算机科学与技术学院, 陕西 西安 710070)

[2](北京控制工程研究所, 北京 100190)

[3](中国空间技术研究院, 北京 100094)

通信作者: 杨孟飞, E-mail: yangmf@bice.org.cn

摘 要: 间歇实时任务的分区 DM (deadline-monotonic) 调度是一个经典的研究问题, 针对约束截止期间歇任务, 提出一种具有更高处理器利用率的多核分区调度算法 PDM-FFD (partitioned deadline-monotonic first-fit decrease). 在 PDM-FFD 中, 首先将任务按照其相对截止期以非递减顺序进行排序, 然后采用 first-fit 策略选择处理器核分配任务, 且在各处理器核上采用 DM 调度策略进行任务调度. 最后通过对任务干扰时间的分析, 得出一种更为紧凑的可调度性判定方法, 并通过该可调度性方法来判定任务的可调度性. 证明 PDM-FFD 的加速因子为 $3-(3\Delta+1)/(m+\Delta)$, 时间复杂度为 $O(n^2)+O(nm)$, 其中 $\Delta=\sum_{\tau_j\in\tau}C_j\times u_j/D_{\max}$, τ_j 为任务集 τ 中的任务, C_j 为该任务最差执行时间, u_j 为该任务利用率, $D_{\max}$ 为 τ 中的最大相对截止期, n 为 τ 的任务数, m 为处理器核数. 该加速因子严格小于 $3-1/m$, 优于已有多核分区调度算法 FBB-FFD. 实验表明, PDM-FFD 算法在 4 核处理器上的处理器利用率比其他算法提高了 18.5%, 且 PDM-FFD 的性能优势随着处理器核数、任务集利用率和任务数的增加而进一步扩大. 由于 PDM-FFD 算法具有高性能特性, 因此该算法可以广泛应用于资源受限的航天器、自动驾驶汽车、工业机器人等典型实时系统中.

关键词: 间歇实时任务; 分区 DM (deadline-monotonic) 调度; 干扰时间; 加速因子; 资源受限

中图法分类号: TP316

中文引用格式: 刘洪标, 宋程昊, 王婷煜, 姜菁菁, 乔磊, 杨孟飞. 并行机器中基于干扰时间的间歇实时任务分区DM调度. 软件学报, 2024, 35(11): 5306–5318. http://www.jos.org.cn/1000-9825/7036.htm

英文引用格式: Liu HB, Song CH, Wang TY, Jiang JJ, Qiao L, Yang MF. Partitioned DM Scheduling for Sporadic Real-time Tasks Based on Interference Time in Parallel Machine. Ruan Jian Xue Bao/Journal of Software, 2024, 35(11): 5306–5318 (in Chinese). http://www.jos.org.cn/1000-9825/7036.htm

Partitioned DM Scheduling for Sporadic Real-time Tasks Based on Interference Time in Parallel Machine

LIU Hong-Biao[1], SONG Cheng-Hao[2], WANG Ting-Yu[1], JIANG Jing-Jing[2], QIAO Lei[2], YANG Meng-Fei[3]

[1](School of Computer Science and Technology, Xidian University, Xi'an 710070, China)

[2](Beijing Institute of Control Engineering, Beijing 100190, China)

[3](China Academy of Space Technology, Beijing 100094, China)

Abstract: Partitioned DM (deadline-monotonic) scheduling of sporadic real-time tasks is a classic research problem. This study proposes a partitioned scheduling algorithm PDM-FFD (partitioned deadline-monotonic first-fit decrease) with higher processor utilization for constrained-deadline sporadic tasks. In PDM-FFD, firstly tasks are sorted in non-decreasing order according to the relative deadline, then the first-fit strategy is utilized to select the processor core to allocate tasks, and each core adopts DM scheduling policy. Finally, a tighter schedulability determination method is obtained by analyzing the task interference time to determine the task schedulability. This study proves that the speedup

* 收稿时间: 2023-01-16; 修改时间: 2023-04-10, 2023-06-06, 2023-06-30; 采用时间: 2023-08-18; jos 在线出版时间: 2023-12-27
CNKI 网络首发时间: 2023-12-29

factor of PDM-FFD is $3-(3\Delta+1)/(m+\Delta)$ and the time complexity is $O(n^2)+O(nm)$. $\Delta=\sum_{\tau_j\in\tau}C_j\times u_j/D_{\max}$ where τ_j belongs to the task set τ, C_j is the worst-case execution time, u_j is the utilization, $D_{\max}$ is the maximum relative deadline, n is the task number, and m is the processor core number. The speedup factor of PDM-FFD is strictly less than $3-1/m$, which outperforms the existing multi-core partitioned scheduling algorithm FBB-FFD. Experiments show that PDM-FFD improves processor utilization by 18.5% compared to other available algorithms on a four-core processor. The PDM-FFD performance improves with the increasing processor core number, task set utilization, and task number. Due to high performance, PDM-FFD can be widely utilized in typical real-time systems such as resource-constrained spacecraft, autonomous vehicles, and industrial robots.

Key words: sporadic real-time task; partitioned deadline-monotonic (DM) scheduling; interference time; speedup factor; resource-constrained

1 引 言

实时系统的正确性不仅取决于逻辑的正确性, 还取决于产生结果的时间. 实时系统广泛应用于航空航天、交通指挥、工业控制、医疗电子等领域[1].

实时系统中的实时任务根据其触发特点可主要分为两类[2].

(1) 由时间触发的任务, 如传感器数据采集任务、雷达跟踪任务等. 这些任务通常在系统定时器中断时释放, 它们是严格的周期性任务 (periodic task), 需要在指定的截止期内完成.

(2) 由系统外部事件触发的任务, 如通信任务、遥控任务等. 这些任务往往在系统产生外部中断时被释放且释放频率不确定, 但相邻任务实例间有一个最小时间间隔. 此类实时任务被称为间歇任务 (sporadic task), 同样也需要在规定的截止期内完成.

对于具有约束截止期的间歇任务 (任务的相对截止期小于等于其任务实例最小到达时间间隔), DM (deadline-monotonic) 调度[3]是一种最优的固定优先级调度策略. 任务优先级的分配取决于其相对截止期大小. 任务的相对截止期越小, 任务的优先级就会被分配得越高, 反之亦然. 由于 DM 调度策略开销低、可预测性强, 因此被广泛应用于解决间歇任务的调度问题.

随着实时系统对计算能力的要求越来越高以及摩尔定律的失效, 多核处理器在实时系统中的应用越来越广泛, 这就带来了间歇实时任务在多核处理器上的调度问题. 多核处理器的调度策略主要包括全局调度 (global scheduling) 和分区调度 (partitioned scheduling). 与全局调度相比, 分区调度[4]可以避免任务迁移带来的系统开销 (如任务上下文迁移带来的核间通信开销、数据传输开销等). 此外, 当处理器核上的任务发生超时 (例如, 实际执行时间超过了预估的最坏执行时间), 只有该处理器核上的任务会受到影响. 因此, 分区调度在航天器等强实时系统中得到了广泛的应用. 现有的多核嵌入式实时操作系统都支持分区调度, 如 FreeRTOS[5]、uCOS[6]、VxWorks[7]等.

算法的实时性能衡量了该算法成功调度实时任务集的能力, 因此通常使用实时性能来体现算法之间的性能优劣. 对于多核调度算法, 任务集可调度率及加速因子常用来评价算法的实时性能. 算法的任务集可调度率定义为对于所有给定的任务集, 其中可被该算法成功调度的任务集占总任务集合的比例; 而加速因子反映了一个算法相较于理论最优调度算法的次优性, 一般来说, 算法的加速因子越小, 其实时性能越优, 详见后续定义.

本文的动机可以分为以下两个方面.

一方面源于实际的工程需求, 航天器等资源受限的实时系统需要一种多核分区调度机制, 用于约束截止期间歇实时任务的调度. 而此类系统的计算资源通常是有限的[8−11], 例如航天器计算机在面对微重力、极端温度、单粒子翻转等恶劣影响因素时, 为了长时间稳定运行, 处理器的时钟速度相对较低, 从而限制了计算资源和存储资源.

另一方面, 间歇实时任务的分区 DM 调度是一个经典的研究问题[12,13], 研究人员正在不断提高算法的任务集可调度率, 减小加速因子. 本文希望能够找到一个实时性能更优的分区 DM 调度算法, 即该算法拥有更高的任务集可调度率和更小的加速因子.

本文主要贡献如下.

(1) 提出了一种基于干扰时间的间歇实时任务多核分区 DM 调度算法 PDM-FFD, 该算法具有更高的处理器利

用率. 实验表明, 特定情况下, 在 4 核处理器中, 该算法的处理器利用率相较其他相关算法提高了 18.5%.

(2) 证明了 PDM-FFD 算法的加速因子为 $3-(3\Delta+1)/(m+\Delta)$, 其中 $\Delta=\sum_{\tau_j\in\tau}C_j\times u_j/D_{\max}$, τ_j 为任务集 τ 中的任务, C_j 为该任务最差执行时间, u_j 为该任务利用率, $D_{\max}$ 为 τ 中的最大相对截止期, m 为处理器核数. 该加速因子严格小于 $3-1/m$, 优于已有的多核分区调度算法 FBB-FFD.

本文第 2 节介绍相关工作. 第 3 节介绍系统模型, 包括处理器模型和任务模型. 第 4 节介绍分区调度算法 PDM-FFD, 又可进一步分为可调度性测试、分区调度算法和时间复杂度分析. 第 5 节是实验比较与结果分析. 最后是对整个研究工作的总结.

2 相关工作

多核实时调度策略[12]可分为分区调度、全局调度、集群调度和半分区调度. 分区调度具有运行开销低、可预测性强、故障隔离性好、单处理器研究成果可重用等优点, 因此在实时系统中得到了广泛应用. Liu 等人[14]提出了 RM (rate-monotonic) 算法和 EDF (earliest deadline first) 算法, 这两个算法对于周期任务分别是最优的固定优先级调度算法和最优的动态优先级调度算法. Leung 等人[3]提出的 DM (deadline-monotonic) 调度算法对于约束截止期间歇任务是最优的固定优先级调度算法. 但上述算法都未考虑多核场景, 因此在将这些算法直接应用于解决多核调度问题时, 算法实时性能较差. RTA (response time analysis)[15]是被广泛采用的精确可调度性测试方法. 该方法能够准确地计算出任务的响应时间, 从而对任务的可调度性作出判断. 由于 RTA 分析方法采用迭代的方式不断计算任务的响应时间, 导致该方法时间复杂度较高, 当时间参数为有理数时, 为 NP-hard 问题. 在后续的研究中, 出现了一些充分性的可调度性测试方法, 如利用率上界分析[16]和干扰时间分析[17], 这些方法牺牲了一定的分析精度, 增加了悲观性, 但降低了算法的运行时开销, 仅为多项式时间复杂度.

任务如何分配是多核分区调度中的关键问题. 众所周知, 任务分配是一个装箱问题 (bin-packing), 找到一个最优结果的时间复杂度是 NP-hard[18]. 因此, 通常采用启发式算法来近似最优分配算法. 广泛使用的启发式分配算法[12]有 first-fit, next-fit, worst-fit 和 best-fit.

Dhall 等人[19]和 Oh 等人[20]提出了针对周期任务的 RMNF (rate-monotonic next-fit) 分配算法. 该算法首先按照任务周期大小的非递减顺序对任务进行排序, 然后采用 next-fit 分配策略将任务分配给处理器核, 并在各个核上采用 RM 算法进行调度, 该分配算法的近似率为 2.67, 时间复杂度为 $O(n^2\times D_{\max})$, 其中 n 为任务集的任务数. FFDU (first-fit decrease-utilization)[21]是 RMNF 算法的变体, 时间复杂度也是 $O(n\times\log n)$. 随后, Burchard 等人[22]针对最大任务利用率小于 0.5 的任务集提出了 RMST (rate-monotonic small-tasks) 分配算法. 该算法首先根据因子 $\alpha(\tau_i)=\log_2 T_i-\lfloor\log_2 T_i\rfloor$ 对任务进行非递减排序, 其中 T_i 为任务 τ_i 的周期, 然后采用 next-fit 策略分配任务. RMST 的近似率为 $1/(1-u_{\max})$, $u_{\max}$ 为任务的最大利用率. 为了支持任务利用率为任意值的情况, 他们还提出了 RMGT (rate-monotonic general-tasks) 分配算法. RMGT 的近似率为 1.75, 时间复杂度为 $O(n^2)$. 这两种算法相较于 RMNF 降低了近似率和算法时间复杂度, 算法的实时性能得到了一定提升. Rothvoss[23]提出了 FFMP (first-fit matching periods) 任务分配算法, 该算法同样按照 $\alpha(\tau_i)$ 以非递减顺序排列所有任务, 然后按照 first-fit 策略分配任务. 该分配算法近似率为 2, 时间复杂度为 $O(n\times\log n)$. 该文章还提出了一种基于匹配的分配算法 RMMatching (rate-monotonic matching). 该算法的近似率仅为 1.5, 但时间复杂度为 $O(n^3)$, 可以看出当算法近似率越低时 (表示该算法与最优算法的差距越小), 其时间复杂度也往往较大. 因此一个分区调度算法如何在保证较低近似率的同时, 避免过大的时间复杂度, 是针对周期任务分区调度问题研究的难点.

由于近似率不能作为任务分配算法的可调度性测试条件, Rothvoss[23]在具有固定核数的处理器中, 当任务分配算法采用 FFS (first-fit scheduling) 时, 分析了周期任务集的处理器利用率上界. 任务集的最小处理器利用率上界可以达到 $m(\sqrt{2}-1)$, 其中 m 为核数. 与前面提到的分配算法不同, FFS 没有对任务进行预先排序, 因此它可以应用于支持任务动态加载的系统中, 该上界可以作为任务集可调度性测试的充分条件. Lopez 等人[24]进一步指出, 在启发式算法中, FFD 和 BFD 的处理器利用率上界最高, 而 WF 的处理器利用率上界最低. 其中 FFD (first-fit dcrease) 分配算法在 m 个核上成功调度 n 个周期任务的充分条件是:

$$\sum_{i=1}^{n} u_i \leqslant (n-1)(\sqrt{2}-1)+(m-n+1)(2^{1/(m-n+1)}-1).$$

上述算法及可调度性测试方法都仅考虑了周期任务, 因此不适用于约束截止期间歇任务.

Fisher 等人[25]针对间歇任务提出了 FBB-FFD (“FBB”是作者姓名首字母的组合, “FFD”代表 first-fit decrease) 分区调度算法. 该算法首先按照任务的相对截止期以非递减顺序对任务进行排序, 然后采用 first-fit 策略选择目标核进行任务分配, 每个核上的任务由 DM 策略进行调度. 其中近似请求上界函数 (approximation request bound function) 被用于判定任务分配到一个处理器核上的可行性. 该算法具有多项式时间复杂度, 针对约束截止期间歇任务和任意截止期间歇任务的加速因子分别为 $3-1/m$ 和 $4-2/m$. 随后, Chen 等人[26]表明, 对于约束截止期间歇任务, 分区 DM 调度的加速因子至少为 2.843 06. 此外, 他们还证明了对于任意截止期间歇任务, 一种贪心映射策略的加速因子为 $3-1/m$, 而如果忽略所使用的任务分配策略, 即使采用精确的可调度性测试方法也无法减小加速因子.

尽管多核分区调度理论经过多年的发展, 已经涌现出大量优秀的研究成果, 但针对约束截止期间歇任务的分区 DM 调度策略还有待进一步研究, 仍然可以通过改善可调度性测试方法的悲观性来降低算法加速因子, 提高任务集可调度率, 从而提升算法的实时性能.

3 系统模型

本文考虑的所有任务均为强实时任务 (strong real-time task), 且任务数量及任务基本信息在系统离线 (offline) 阶段就已确定, 不存在任务在系统运行时动态载入的情况, 任何通过可调度性测试的实时任务都不允许错过其截止期, 否则可能产生灾难性后果. 任务的优先级根据 DM 调度策略进行分配, 即任务的相对截止期越小, 分配的优先级越高, 反之越低. 任务在多核中按照分区调度策略进行调度, 即任务一旦被分配到某个处理器核上就不再发生核间迁移, 该任务的所有任务实例都只在当前处理器核上执行.

3.1 处理器模型

系统运行平台 P 为一个同构 m ($m \geqslant 2$) 核处理器, $p=\{p_1,p_2,\ldots,p_m\}$, 其中 p_k ($1 \leqslant k \leqslant m$) 表示第 k 个处理器核. 多核处理器虽然与多处理器物理结构不同, 但是调度理论是相通的. 假设核间通信和任务上下文等开销都已计入任务最差执行时间.

3.2 任务模型

间歇 (sporadic) 实时任务模型通常用一个三元组表示, $\tau_i=(C_i,D_i,T_i)$, 其中 C_i 表示最差执行时间, D_i 表示相对截止期, T_i 表示任务实例的最小到达间隔. 根据 D_i 和 T_i 的关系, 间歇任务可以细分为 3 类任务: 1) 如果 $D_i=T_i$, 称为隐式截止期 (implicit-deadline) 间歇任务; 2) 如果 $D_i \leqslant T_i$, 称为约束截止期 (constrained-deadline) 间歇任务; 3) 如果 D_i 与 T_i 无关, 称为任意截止期 (arbitrary-deadline) 间歇任务.

本文考虑约束截止期间歇实时任务, 满足 $C_i \leqslant D_i \leqslant T_i$, 该任务模型如图 1 所示. 任务的相关参数解释如下.

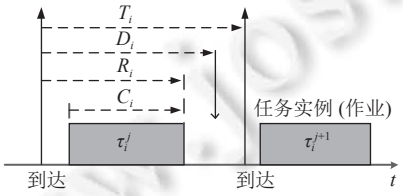

图 1 约束截止期间歇任务模型

- C_i: 任务 τ_i 的最差执行时间.
- D_i: 任务 τ_i 的相对截止期, 即 τ_i 的每个任务实例都需要在 D_i 内完成.
- T_i: 任务 τ_i 的连续两个实例到达的最小时间间隔.
- τ: n 个约束截止期间歇任务组成的集合, $\tau=\{\tau_1,\tau_2,\ldots,\tau_n\}$.

- τ_i^j: 任务 τ_i 的第 j 个任务实例 (也被称为作业), τ_i^j 和 τ_i^{j+1} 的到达时间最小间隔应大于等于 T_i.
- R_i: 任务 τ_i 的最坏响应时间, 它表示 τ_i 的释放时刻与完成时刻之间的时间差.
- u_i: 任务 τ_i 的处理器利用率, 其值为 C_i/T_i.
- U: 任务集 $\tau=\{\tau_1,\tau_2,\ldots,\tau_n\}$ 的处理器利用率, 其值为 $\sum_{i=1}^{n}u_i$, 即该任务集中所有任务的处理器利用率之和, 其中 n 代表任务集的任务数.

若 $R_i \leqslant D_i$, 则 τ_i 称为可调度的, 即任务满足截止期要求. 若 $\forall\tau_i\in\tau$, τ_i 是可调度的, 则任务集 τ 是可调度的. 任务集中的任务按照优先级从高到低排序, 即 τ_i 的优先级高于 τ_j 当且仅当 $i<j$. 在后文分析中均考虑最坏情况[14], 即所有任务同时到达, 此时高优先级任务对低优先级任务抢占量达到最大, 若该情况下任务可调度, 则任意情况下任务都可调度. 每个任务的所有作业一旦到达就立刻被释放, 任务之间是时序独立的, 且无共享资源访问.

4 基于干扰时间的分组调度方法

分区调度算法主要可分为 3 步: 1) 确定任务优先级; 2) 选择目标核; 3) 测试分配到目标核的可行性, 即进行任务集的可调度性测试. 由于第 1 步和第 2 步已经有最优策略, 所以核心在于第 3 步的可调度性测试方法. 已有的分区调度算法[25−27]往往结合处理器需求上界, 利用率上界或者响应时间上界等方法来进行可调度性测试. 而本文是结合干扰时间上界方法来进行可调度性测试, 因为能得到更加紧凑的分析结果. 在推导过程中, 首先引入了干扰时间的定义以及相关概念, 然后推导出一个干扰时间近似函数, 并对其更加紧凑的性质进行了证明, 以表明其有效性. 最后根据该近似函数的性质, 推导了本文分区调度算法的加速因子, 并证明了比算法 FBB-FFD 的加速因子更小, 从而从理论上说明了本文的分区调度算法更优.

4.1 基本概念

定义 1. 任务需求上界函数 (demand bound function) $DBF(\tau_i,\Delta t)$ 表示在任意长度为 Δt 的时间区间内, τ_i 的所有作业 (到达时刻和截止时刻都在此时间区间内) 产生的最大处理器执行时间需求量[27].

$$DBF(\tau_i,\Delta t)\overset{\text{def}}{=}\max\left\{0,\left(\left\lfloor\frac{\Delta t-D_i}{T_i}\right\rfloor+1\right)\times C_i\right\} \tag{1}$$

定义 2. 任务集需求上界函数 $DBF(\tau,\Delta t)$ 表示 τ 的所有任务在任意长度为 Δt 的时间区间内对处理器时间需求量的总和.

$$DBF(\tau,\Delta t)\overset{\text{def}}{=}\sum_{\tau_i\in\tau}DBF(\tau_i,\Delta t) \tag{2}$$

定义 3. 任务集最大负载函数 $LOAD(\tau)$ 表示 τ 的需求上界 $DBF(\tau,\Delta t)$ 归一化后的最大值[12,27].

$$LOAD(\tau)\overset{\text{def}}{=}\max_{\Delta t>0}\left\{\frac{DBF(\tau,\Delta t)}{\Delta t}\right\} \tag{3}$$

当任务集 τ 可调度时, 显然有:

$$DBF(\tau_i,\Delta t)\leqslant\Delta t\Rightarrow LOAD(\tau)\leqslant 1.$$

当处理器核的运行速度为 α 时, 任务 τ_i 的最差执行时间变为 C_i/α, 对于单核处理器, 有:

$$LOAD(\tau)\leqslant\alpha.$$

而对于 m 核处理器, 对 m 个核进行相加, 可以得到:

$$LOAD(\tau)\leqslant m\times\alpha \tag{4}$$

定义 4. 任务请求上界函数 (request bound function) $RBF(\tau_i,\Delta t)$ 表示在任意长度为 Δt 的时间区间内, τ_i 的所有作业 (到达时刻在该区间内即可) 产生的最大处理器执行时间请求量:

$$RBF(\tau_i,\Delta t)\overset{\text{def}}{=}\left\lceil\frac{\Delta t}{T_i}\right\rceil\times C_i \tag{5}$$

定义 5. 请求近似函数 $RBF^*(\tau_i,\Delta t)$ 表示对 $RBF(\tau_i,\Delta t)$ 函数的线性逼近[28].

$$RBF^*(\tau_i,\Delta t)\overset{\text{def}}{=}u_i\times\Delta t+C_i \tag{6}$$

引理 1. 对任意长度为 Δt 的时间区间, 其中 $\Delta t\geqslant 0$, 有 $RBF(\tau_i,\Delta t)\leqslant RBF^*(\tau_i,\Delta t)$.

证明: 为了得到两式大小关系, 令它们做减法:

$$RBF^*(\tau_i,\Delta t)-RBF(\tau_i,\Delta t)=u_i\times\Delta t+C_i-\left\lceil\frac{\Delta t}{T_i}\right\rceil C_i\geqslant u_i\times\Delta t+C_i-\left(\frac{\Delta t}{T_i}+1\right)C_i=u_i\times\Delta t+C_i-u_i\times\Delta t-C_i=0.$$

因此, $RBF^*(\tau_i,\Delta t)\geqslant RBF(\tau_i,\Delta t)$, 证明完成.

引理 2. 对任意长度为 Δt 的时间区间, 其中 $\Delta t\geqslant D_i$, 有 $RBF^*(\tau_i,\Delta t)\leqslant DBF(\tau_i,\Delta t)+u_i\times\Delta t$.

证明: 由公式 (6), $RBF^*(\tau_i,\Delta t)\overset{\text{def}}{=}u_i\times\Delta t+C_i$, 有:

$$DBF(\tau_i,\Delta t)+u_i\times\Delta t-RBF^*(\tau_i,\Delta t)=DBF(\tau_i,\Delta t)-C_i.$$

结合定义 1 可知, $\Delta t\geqslant D_i$ 时, $DBF(\tau_i,\Delta t)\geqslant C_i$, 有:

$$DBF(\tau_i,\Delta t)+u_i\times\Delta t-RBF^*(\tau_i,\Delta t)\geqslant 0.$$

因此, $RBF^*(\tau_i,\Delta t)\leqslant DBF(\tau_i,\Delta t)+u_i\times\Delta t$, 证明完成.

定义 6. 任务干扰时间 (interference bound function) $IBF(\tau_i,\Delta t)$ 表示任务 τ_i 在任意长度为 Δt 的时间区间内的作业对低优先级任务产生的最大干扰量[17].

$$IBF(\tau_i,\Delta t)\overset{\text{def}}{=}\left\lfloor\frac{\Delta t}{T_i}\right\rfloor C_i+\min\{C_i,\Delta t \bmod T_i\} \tag{7}$$

定理 1. 如果 $D_i-\sum_{j<i}IBF(\tau_j,D_i)\geqslant C_i$, 则任务 τ_i 可调度.

证明: 所有更高优先级任务 $\{\tau_j|j<i\}$ 对 τ_i 在其截止期前产生的最大干扰时间为 $\sum_{j<i}IBF(\tau_j,D_i)$. 如果 $D_i-\sum_{j<i}IBF(\tau_j,D_i)\geqslant C_i$, 表明 τ_i 的执行请求在其截止前能得到满足, 因此任务 τ_i 可调度. 证明完成.

定义 7. 干扰时间近似函数 $IBF^*(\tau_i,\Delta t)$ 表示对 $IBF(\tau_i,\Delta t)$ 函数的线性逼近.

$$IBF^*(\tau_i,\Delta t)\overset{\text{def}}{=}u_i\times\Delta t+C_i-C_i\times u_i \tag{8}$$

引理 3. 对任意长度为 Δt 的时间区间, 其中 $\Delta t\geqslant 0$, 有 $IBF(\tau_i,\Delta t)\leqslant IBF^*(\tau_i,\Delta t)$.

证明: 令 $k\in N$, Δt 的取值可以分两种情况讨论:

情况 1: $kT_i\leqslant\Delta t<kT_i+C_i$, 有:

$$IBF^*(\tau_i,\Delta t)-IBF(\tau_i,\Delta t)=u_i\times\Delta t+C_i-C_iu_i-\left(\left\lfloor\frac{\Delta t}{T_i}\right\rfloor C_i+\min\{C_i,\Delta t \bmod T_i\}\right)=$$

$$kC_i+(\Delta t-kT_i)u_i+C_i-C_i\times u_i-kC_i-(\Delta t-kT_i)=(1-u_i)(C_i+kT_i-\Delta t)>0.$$

情况 2: $kT_i+C_i\leqslant\Delta t<(k+1)T_i$, 有:

$$IBF^*(\tau_i,t)-IBF(\tau_i,\Delta t)=u_i\times\Delta t+C_i-C_iu_i-\left(\left\lfloor\frac{\Delta t}{T_i}\right\rfloor C_i+\min\{C_i,\Delta t \bmod T_i\}\right)=$$

$$kC_i+C_i\times u_i+(\Delta t-kT_i-C_i)u_i+C_i-C_i\times u_i-kC_i-C_i=u_i(\Delta t-kT_i-C_i)\geqslant 0.$$

综上, $\forall\Delta t\geqslant 0, IBF(\tau_i,\Delta t)\leqslant IBF^*(\tau_i,\Delta t)$, 证明完成.

推论 1. 对任意长度为 Δt 的时间区间, 其中 $\Delta t\geqslant 0$, 有 $IBF^*(\tau_i,\Delta t)=RBF^*(\tau_i,\Delta t)-C_i\times u_i$.

证明: 根据公式 (6) 和公式 (8), 有:

$$RBF^*(\tau_i,\Delta t)=u_i\times\Delta t+C_i,$$

$$IBF^*(\tau_i,\Delta t)=u_i\times\Delta t+C_i-C_i\times u_i.$$

因此, 有:

$$IBF^*(\tau_i,\Delta t)=RBF^*(\tau_i,\Delta t)-C_i\times u_i.$$

证明完成.

引理 4. 对任意长度为 Δt 的时间区间, 其中 $\Delta t\geqslant D_i$, 有 $IBF(\tau_i,\Delta t)\leqslant DBF(\tau_i,\Delta t)+u_i\times\Delta t-C_i\times u_i$.

证明: 结合引理 3 和推论 1, 可知:

$$IBF(\tau_i,\Delta t) \leqslant IBF^*(\tau_i,\Delta t) = RBF^*(\tau_i,\Delta t) - C_i \times u_i \tag{9}$$

结合引理 2, 可知 $\Delta t \geqslant D_i$ 时,

$$RBF^*(\tau_i,\Delta t) \leqslant DBF(\tau_i,\Delta t) + u_i \times \Delta t.$$

结合公式 (9), 有:

$$IBF(\tau_i,\Delta t) \leqslant DBF(\tau_i,\Delta t) + u_i \times \Delta t - C_i \times u_i.$$

证明完成.

4.2 分区调度算法

本文提出的分区调度算法命名为 PDM-FFD (partitioned deadline-monotonic first-fit decrease). 该算法中, 首先将任务集中的任务按照其相对截止期以非递减顺序进行排序, 然后采用首次适应 (first-fit) 策略进行目标核分配, 其中每个任务在目标核上的可调度性通过定理 1 进行判定, 最后, 采用 DM 调度算法对每个核进行任务调度. PDM-FFD 在进行可调度性测试时, 采用了更紧凑 (tighter) 的上界函数 IBF . 相比于 FBB-FDD 使用的请求上界函数 RBF , IBF 在计算高优先级任务实例在一段时间区间内对低优先级任务产生的抢占量时更加精准, 从而降低了分析的悲观性. 提升了算法的任务集可调度率. PDM-FFD 的详细步骤描述如下所示.

步骤 1: 使用快速排序算法[29]对任务集中的任务按照相对截止期非递减顺序进行排序.

步骤 2: 根据任务集中任务的新顺序逐个选择任务进行多核分配. 如果所有任务都已分配完成, 则返回 true 并且分配算法结束, 即 PDM-FFD 成功调度该任务集.

步骤 3: 对于任务 τ_i , 根据首次适应 (first-fit) 策略, 逐个选择处理器核进行分配. 如果所有处理器核都无法成功分配, 则返回 false 且分配算法结束, 即 PDM-FFD 调度该任务集失败.

步骤 4: 对于处理器核 p_k , 按照 IBF 函数 (见公式 (7)) 计算干扰时间的总和 IBF_{sum} .

步骤 5: 如果 $D_i - IBF_{\text{sum}} \geqslant C_i$, 则任务 τ_i 可调度 (见定理 1), 此时任务 τ_i 被分配到 p_k 上并且转步骤 2, 否则转步骤 3.

分区调度算法 PDM-FFD 的伪代码如算法 1 所示.

算法 1. Boolean PDM-FFD(τ, P) .

输入: 待分区调度的间歇实时任务集 τ ; 多核处理器平台 P ;
输出: 任务集 τ 分区调度结果. true: 成功; false: 失败.

```
1.  Quick_sort(τ);                    //步骤 1, 使用快速排序算法对任务集进行排序
2. for τ_i : τ do                     //步骤 2, 逐个选择任务
3.   for p_k : P do                   //步骤 3, 逐个选择处理器核进行分配
4.     IBF_sum = 0;
5.     for τ_j : τ(p_k) do            //步骤 4, 计算干扰时间总和
6.       IBF_sum += IBF(τ_j, D_i);
7.     end for
8.     if D_i − IBF_sum ⩾ C_i then    //步骤 5, 可调度性判定
9.       τ(p_k) = τ(p_k) ∪ τ_i;
10.      BREAK;
11.    end if
12.  end for
13.  return false;                    //任务分配失败, 任务集不可调度
14. end for
15. return true;                      //任务分配成功, 任务集可调度
```

定理 2. 分区调度算法 PDM-FFD 返回 true, 则任务集 τ 是可调度的.

证明: 由算法 1 可知, 首先将任务按照截止期非递减顺序进行排序, 由于采用了 DM 调度策略, 因此任务的优先级依次降低. 对于任意一个核, 当前分配的任务的优先级总是低于已经被分配在该核的任务 (因为 PDM-FFD 是离线调度算法, 不存在高优先级任务在系统运行时动态载入的情况), 不会对已分配任务的可调度性造成影响, 因此只用测试当前任务的可调度性即可, 这转为一个单核可调度性测试问题. 结合定理 1 可知, PDM-FFD 算法的第 5 步表明当前分配的任务可调度. 而 PDM-FFD 分区调度算法返回 true 表明所有任务都通过了可调度性测试, 因此任务集 τ 是可调度的. 证明完成.

定理 3. 分区调度算法 PDM-FFD 可调度的充分条件是:

$$LOAD(\tau) \leqslant m - U - (m-1) \times dens_{\max}(\tau) + \Delta,$$

其中, $dens_{\max}(\tau) = \max_{\tau_i \in \tau}\{C_i/D_i\}, \Delta = \sum_{\tau_j \in \tau} c_j \times u_j / D_{\max}, D_{\max} = \max_{\tau_i \in \tau}\{D_i\}$.

证明: 由算法 1 可知, 任务分配失败, 一定有:

$$D_i - \sum_{\tau_j \in \tau(p_k)} IBF(\tau_j, D_i) < C_i.$$

求和 m 个核, 有:

$$\begin{aligned} & mD_i - \sum_{\tau_j \in (\tau \setminus \{\tau_i\})} IBF(\tau_j, D_i) < mC_i \\ \Leftrightarrow\ & mD_i - \left(\sum_{\tau_j \in \tau} IBF(\tau_j, D_i) - C_i \right) < mC_i \\ \Leftrightarrow\ & \sum_{\tau_j \in \tau} IBF(\tau_j, D_i) > mD_i - (m-1)C_i. \end{aligned}$$

由于任务集采用 DM 调度策略, 因此有 $D_i \geqslant D_j$, 结合引理 4, 有:

$$\begin{aligned} & \Rightarrow \sum_{\tau_j \in \tau} (DBF(\tau_j, D_i) + u_j \times D_i - C_j \times u_j) > mD_i - (m-1)C_i \\ & \Leftrightarrow \sum_{\tau_j \in \tau} DBF(\tau_j, D_i) > mD_i - (m-1)C_i - \sum_{\tau_j \in \tau} u_j \times D_i + \sum_{\tau_j \in \tau} C_j \times u_j \\ & \Leftrightarrow \frac{\sum_{\tau_j \in \tau} DBF(\tau_j, D_i)}{D_i} > m - (m-1)\frac{C_i}{D_i} - \sum_{\tau_j \in \tau} u_j + \frac{\sum_{\tau_j \in \tau} C_j \times u_j}{D_i}. \end{aligned}$$

由于 $dens_{\max}(\tau) = \max_{\tau_i \in \tau}\{C_i/D_i\}$, $D_{\max} = \max_{\tau_i \in \tau}\{D_i\}$, 有:

$$\Rightarrow LOAD(\tau) > m - (m-1)dens_{\max}(\tau) - U + \frac{\sum_{\tau_j \in \tau} C_j \times u_j}{D_{\max}}.$$

取其逆否命题, 可得:

$$LOAD(\tau) \leqslant m - (m-1)dens_{\max}(\tau) - U + \Delta.$$

则任务集 τ 一定可调度, 其中 $\Delta = \sum_{\tau_j \in \tau} C_j \times u_j / D_{\max}$. 证明完成.

定义 8. 加速因子 (speedup factor)[12] S_A 表示一个调度算法 A 与理论最优调度算法的调度能力差距. 若一个最优算法在处理器速度为 α 的平台上能成功调度任务集 τ, 那么算法 A 在速度为 $\alpha \times S_A$ 的处理器平台上也能成功调度该任务集. 显然, 加速因子越小则调度算法越优, 理论最优调度算法的加速因子则为 1.

定理 4. 分区调度算法 PDM-FFD 的加速因子为 $S_{\text{PDM-FFD}} = 3 - (3\Delta + 1)/(m + \Delta)$, 其中 $\Delta = \sum_{\tau_j \in \tau} C_j \times u_j / D_{\max}$.

证明: 假设当处理器速度为 $\alpha = 1/(3 - (3\Delta + 1)/(m + \Delta)) = (m + \Delta)/(3m - 1)$ 时, 最优调度算法可以调度任务集 τ, 根据加速因子定义, 则要证明当处理器速度提升至 $3 - (3\Delta + 1)/(m + \Delta)$ 倍时, PDM-FFD 算法仍然可以调度该任务集.

根据上述假设和公式 (4), 有:

$$LOAD(\tau) \leqslant m \times \frac{m+\Delta}{3m-1} \tag{10}$$

$$U \leqslant m \times \frac{m+\Delta}{3m-1} \tag{11}$$

$$dens_{\max}(\tau) \leqslant \frac{m+\Delta}{3m-1} \tag{12}$$

显然下述恒等式成立:

$$\begin{aligned} & m+\Delta \leqslant m+\Delta \\ \Leftrightarrow & \frac{m+\Delta}{3m-1}(3m-1) \leqslant m+\Delta \\ \Leftrightarrow & m\frac{m+\Delta}{3m-1} \leqslant (1-2m)\frac{m+\Delta}{3m-1} + m + \Delta \\ \Leftrightarrow & m\frac{m+\Delta}{3m-1} \leqslant m-(m-1)\frac{m+\Delta}{3m-1} - m\frac{m+\Delta}{3m-1} + \Delta. \end{aligned}$$

结合公式 (10)–公式 (12), 可以得出:

$$LOAD(\tau) \leqslant m-(m-1)\times dens_{\max}(\tau) - U + \Delta.$$

根据定理 3 可知, 任务集 τ 可被 PDM-FFD 算法成功调度, 证明完成.

该加速因子也刚好是调度算法 PDM-FFD 的加速因子下界, 原因如下:

假设 PDM-FFD 的加速因子为 S, 当前处理器速度为 1, 那么最优算法可调度的处理器速度为 $1/S$, 有:

$$LOAD(\tau) \leqslant m/S, U \leqslant m/S, dens_{\max}(\tau) \leqslant 1/S.$$

结合任务集可被 PDM-FFD 调度的充分条件 (见定理 3) $LOAD(\tau) \leqslant m - U - (m-1)\times dens_{\max}(\tau) + \Delta$, 有 $m/S \leqslant m - m/S - (m-1)/S + \Delta$, 解得 $S \geqslant 3-(3\Delta+1)/(m+\Delta) = S_{\text{PDM-FFD}}$.

推论 2. 分区调度算法 PDM-FFD 的加速因子小于分区调度算法 FBB-FFD 的加速因子.

证明: 从文献 [25] 中可知, FBB-FFD 的加速因子为 $S_{\text{FBB-FFD}} = 3-1/m$, 结合定理 4, 有:

$$S_{\text{FBB-FFD}} - S_{\text{PDM-FFD}} = 3 - \frac{1}{m} - \left(3 - \frac{3\Delta+1}{m+\Delta}\right) = \frac{3\Delta+1}{m+\Delta} - \frac{1}{m} = \frac{\Delta(3m-1)}{m^2+m\Delta} > 0,$$

其中, $\Delta = \sum_{\tau_j \in \tau} C_j \times u_j / D_{\max} > 0$, 证明完成.

4.3 时间复杂度

任务集 τ 包含 n 个任务, 处理器 P 包含 m 个核.

第 1 步, 采用快速排序方法对任务集 τ 按照截止期非递减顺序进行排序, 时间复杂度为 $\mathrm{O}(n\lg n)$.

第 2 步, 分配任务 τ_i 时, 已经有 $i-1$ 个任务分配在处理器 P 上, 即 $|\tau(p_1)|+|\tau(p_2)|+\ldots+|\tau(p_m)| = i-1$.

第 3 步, 最坏情况, 每个核都需要进行尝试.

第 4 步, IBF_{sum} 的计算时间复杂度为 $|\tau(p_k)|$, 其中 $IBF(\tau_j, D_i)$ 可在常数时间内计算出结果.

第 5 步, 可调度性测试时间复杂度为 $\mathrm{O}(1)$.

因此, 第 3–5 步总的时间复杂度为 $|\tau(p_1)|+1+|\tau(p_2)|+1+\ldots+|\tau(p_m)|+1 = i-1+m$. 第 2–5 步总的时间复杂度为 $\sum_{i=1}^{n} i-1+m = \mathrm{O}(n^2)+\mathrm{O}(nm)$. 第 1–5 步总的时间复杂度为 $\mathrm{O}(n\lg n)+\mathrm{O}(n^2)+\mathrm{O}(nm)$.

综上, 分区调度算法 PDM-FFD 的时间复杂度为 $\mathrm{O}(n\lg n)+\mathrm{O}(n^2)+\mathrm{O}(nm) = \mathrm{O}(n^2)+\mathrm{O}(nm)$.

5 实验及结果分析

实验中随机产生大量约束截止期间歇实时任务集. 本文的 PDM-FFD 算法与相关的分区调度算法进行了性能对比, 对比算法包括 FBB-FFD[25], CHL-FFD[26], BNRB-FFD[30], 这些算法均按照作者首字母命名. 它们的时间复杂

度也是 $O(n^2)+O(nm)$. 这几个算法的调度过程是类似的, 差别在于结合的可调度性测试方法不同, 其中 FBB-FFD 结合的是处理器需求上界方法; CHL-FFD 是结合利用率上界方法; BNRB-FFD 是结合响应时间上界方法; 而本文的 PDM-FFD 算法是结合了更紧凑的干扰时间上界分析方法.

自变量为核数 m、任务集利用率 U、截止期范围 d、任务数 n.

性能指标包括可调度率 r 和平均处理器核数 a (任务集成功分配所需要的核数).

通过 6 个实验对上述分区调度算法的性能进行对比. 每个实验的参数如表 1 所示.

表 1 6 个对比实验的参数

序号	m	n	U	d	性能指标	实验结果图
实验1	2	30	[0.2, 2]	0.5	r	图2
实验2	4	60	[0.5, 4]	0.5	r	图3
实验3	8	150	[1.5, 8]	0.5	r	图4
实验4	4	60	2.6	[0, 1]	r	图5
实验5	4	[8, 150]	2.9	0.5	r	图6
实验6	—	[20, 400]	[0.4, 16]	0.5	a	图7

实验 1, 2, 3 分别进行了处理器核数变化时的可调度率对比, 因为嵌入式系统中常用的多核处理器一般是 2, 4, 8 核, 例如 Xilinx 的 ZYNQ 双核板卡, 宇航用的 SOC2012 四核板卡等. 实验 4 对比了不同算法在截止期范围变化时的可调度率, 而实验 5 则考虑了任务数变化时的可调度率差异, 实验 6 对不同算法在任务数和任务集利用率变化时的平均处理器核数进行了对比.

任务集利用率: 按照 UUnifast 算法[31]随机产生, 使得每个任务的利用率服从均匀分布, 当出现任务利用率大于 1 时则重新产生整个任务集.

截止期范围: 任务的截止期在指定区间内随机产生, 该区间可被表示为 $[C_i+(1-d)\times(T_i-C_i),T_i]$. 其中 $d\in[0,1]$ 用于指定范围, 当 $d=0$ 时, 截止期所属区间为 $[T_i,T_i]$, 即为隐式截止期间歇任务. 当 $d=1$ 时, 截止期所属区间为 $[C_i,T_i]$, 即为约束截止期间歇任务.

● 实验 1–3. 如图 2–图 4, 当任务集的利用率较低时, 各算法的可调度率都为 1. 随着任务集利用率的增加, 可调度率开始下降. 其中 FBB-FFD 最先开始下降, 其次是 BNRB-FFD 和 CHL-FFD, 而 PDM-FFD 最后开始下降. 当核数为 4 时, 已有的最高可调度率算法 CHL-FFD 在任务集利用率为 2.7 时开始下降, 而本文算法 PDM-FFD 的可调度率在任务集利用率为 3.2 时才开始下降, 而且这几个算法的可调度率几乎垂直下降, 因此在此情况下算法 PDM-FFD 将处理器利用率提高了 18.5%. 随着处理器核数的增加, 分区调度算法 PDM-FFD 的可调度率优势进一步扩大, 使得处理器得到更充分的利用. 这是因为分区调度算法 PDM-FFD 是采用干扰时间分析法, 由推论 1 可知, 干扰上界函数 IBF 是一个更紧凑的上界函数, 具有更小的悲观性, 因此能更充分地利用处理器.

● 实验 4. 如图 5 所示, 当截止期范围等于 0 时, 此时的任务模型退化为隐式截止期间歇任务模型, 各算法的可调度率都很高. 随着任务截止期范围的增加, FBB-FFD 和 BNRB-FFD 的可调度率开始下降, 而 CHL-FFD 和 PDM-FFD 的可调度率直到截止期范围等于 0.83 时才开始轻微下降. 而 PDM-FFD 的可调度率仍然是最高.

● 实验 5. 如图 6 所示, 随着任务集任务数的增加, FBB-FFD 和 BNRB-FFD 的可调度率逐渐开始下降, CHL-FFD 的可调度率先升高再下降, 而 PDM-FFD 的可调度率先升高然后不变. 任务数变化对 PDM-FFD 的可调度率影响较小. 由于实验的随机性, 导致算法的可调度率产生了波动.

● 实验 6. 如图 7 所示, 随着任务集利用率的提高和任务数的增加, 任务集分配成功所需要的核数几乎线性增加. FBB-FFD 增加最快, 其次是 BNRB-FFD 和 CHL-FFD. 而 PDM-FFD 所需要的核数最少. 当任务集利用率等于 15 时, 理论最优调度算法需要 15 个核, 而 PDM-FFD 仅需要 18 个核, CHL-FFD 需要 21 个核. 因此 PDM-FFD 调度算法更接近最优调度, 符合第 4 节中加速因子的理论分析结果.

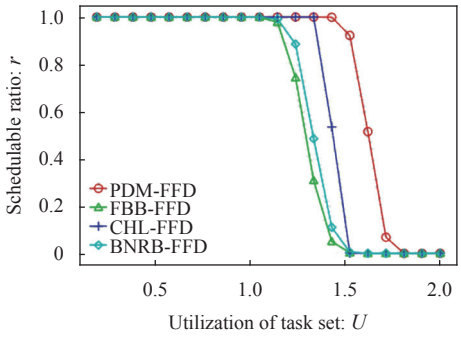

图 2 在 2 核处理器下任务集利用率变化时的可调度率比较

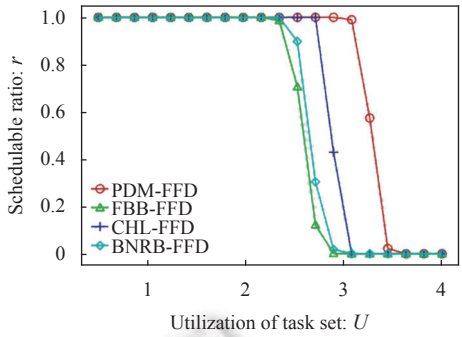

图 3 在 4 核处理器下任务集利用率变化时的可调度率比较

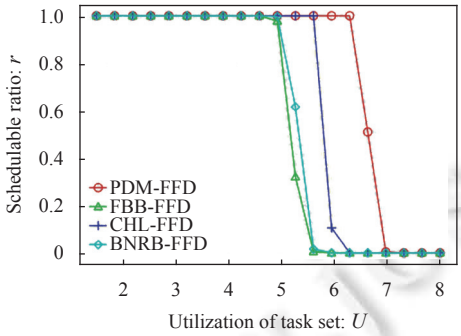

图 4 在 8 核处理器下任务集利用率变化时的可调度率比较

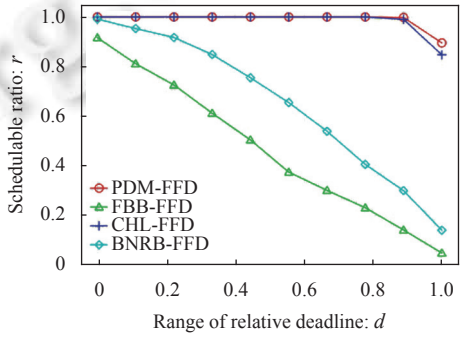

图 5 相对截止期范围变化时的可调度率比较

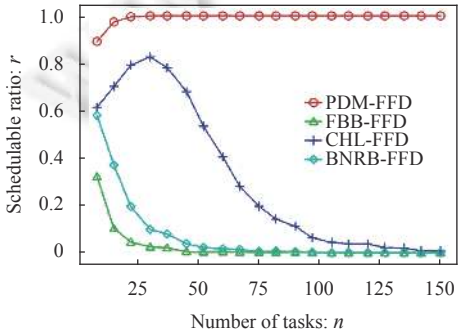

图 6 任务数量发生变化时的可调度率比较

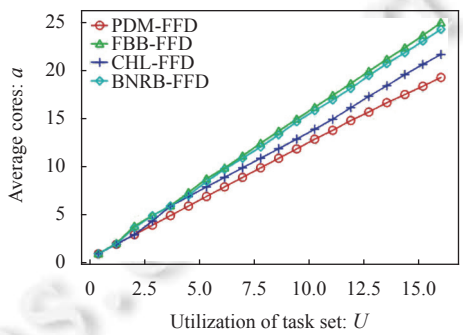

图 7 任务集利用率变化时的平均核数对比

6 总 结

随着实时系统对计算能力要求的不断提高和摩尔定律的失效, 多核处理器在实时系统中得到了广泛应用. 由于间歇任务是系统中最常见的任务, 因此需要在多核处理器中考虑其调度问题. 本文针对约束截止期间歇任务的多核分区 DM 调度问题, 提出了一种处理器利用率更高的分区调度算法 PDM-FFD. 本文首先分析了任务的干扰时间, 得到了一个更紧凑的可调度性测试方法. 然后结合该可调度性测试方法, 给出了算法 PDM-FFD 和对应的可调度性证明. 该算法主要分为 3 步: 1) 将任务按照其相对截止期以非递减顺序进行排序; 2) 采用首次适应策略 (first-fit) 选择处理器核进行分配, 其中任务分配的可行性由上述可调度性测试方法进行判定; 3) 在每个核上采用 DM 策略进行任务调度. 其次, 证明了 PDM-FFD 的加速因子为 $3-(3\Delta+1)/(m+\Delta)$, 时间复杂度为 $O(n^2)+O(nm)$,

其中 $\Delta=\sum_{\tau_j\in\tau}C_j\times u_j/D_{\max}$, τ_j 为任务集 τ 中的任务, C_j 为该任务最差执行时间, u_j 为该任务利用率, $D_{\max}$ 为 τ 中的最大相对截止期, n 表示任务数, m 为处理器核数, 该加速因子优于已有的多核分区调度算法 FBB-FFD. 最后, 实验结果表明, 特定情况下, PDM-FFD 算法在 4 核处理器上的处理器利用率比其他算法提高了 18.5%, 且 PDM-FFD 的性能优势随着处理器核数、任务集利用率和任务数量的增加而进一步扩大. PDM-FFD 算法面向最普遍的间歇任务, 具有较低的时间复杂度和较高的处理器利用率, 使其在航天器、自动驾驶汽车、工业机器人等典型实时系统中具有广泛的应用前景. 本文当前考虑的是多核处理器中任务的分区调度问题, 下一步将考虑全局调度问题, 同样研究出一个实时性能更优的调度算法.

References:

[1] Akesson B, Nasri M, Nelissen G, Altmeyer S, Davis RI. An empirical survey-based study into industry practice in real-time systems. In: Proc. of the 2020 IEEE Real-time Systems Symp. Houston: IEEE, 2020. 3–11. [doi: 10.1109/RTSS49844.2020.00012]

[2] Buttazzo GC. Hard Real-time Computing Systems: Predictable Scheduling Algorithms and Applications. New York: Springer, 2011. [doi: 10.1007/978-1-4614-0676-1]

[3] Leung JYT, Whitehead J. On the complexity of fixed-priority scheduling of periodic, real-time tasks. Performance Evaluation, 1982, 2(4): 237–250. [doi: 10.1016/0166-5316(82)90024-4]

[4] Burmyakov A, Nikolić B. An exact comparison of global, partitioned, and semi-partitioned fixed-priority real-time multiprocessor schedulers. Journal of Systems Architecture, 2021, 121: 102313. [doi: 10.1016/j.sysarc.2021.102313]

[5] Hagens F, Chen KH. Assessment of efficient dispatching in FreeRTOS. In: Proc. of the 17th Annual Workshop on Operating Systems Platforms for Embedded Real-time Applications. Vienna, 2023. 7–9.

[6] Liu JX, Zhao KB, Du HY, Li M, Wang ZC. Application of uCOS-II and LabVIEW in stirling engine fuel supply control system. Applied Mechanics and Materials, 2011, 143–144: 381–385. [doi: 10.4028/www.scientific.net/AMM.143-144.381]

[7] Wang W, Xia S, Wang LY. Research of interrupt processing in motion control system based on VxWorks. In: Proc. of the 5th Int'l Conf. on Telecommunications and Communication Engineering. Chengdu: Association for Computing Machinery, 2022. 277–280. [doi: 10.1145/3577065.3577115]

[8] Li SF, Qiao L, Yang MF. Memory state verification based on inductive and deductive reasoning. IEEE Trans. on Reliability, 2021, 70(3): 1026–1039. [doi: 10.1109/TR.2021.3074709]

[9] Chen X, Qiao L, Yang MF, Liu HB. Priority ceiling protocol based on avoidance blocking. Ruan Jian Xue Bao/Journal of Software, 2023, 34(7): 3422–3437 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6534.htm [doi: 10.13328/j.cnki.jos.006534]

[10] Liu HB, Xi C, Qiao L, Zhang JK, Yang MF. A schedulability test for sporadic task DM scheduling based on density upper bound. IEEE Access, 2022, 10: 12475–12486. [doi: 10.1109/ACCESS.2022.3143037]

[11] Liu HB, Qiao L, Yang MF, Chen X, Ma Z, Li SF. Real-time task scheduling and analysis method based on virtual zoom out period. Ruan Jian Xue Bao/Journal of Software, 2022, 33(9): 3512–3528 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6394.htm [doi: 10.13328/j.cnki.jos.006394]

[12] Baruah S, Bertogna M, Buttazzo G. Multiprocessor Scheduling for Real-time Systems. Cham: Springer, 2015. [doi: 10.1007/978-3-319-08696-5]

[13] Sun ZY, Guo MY, Liu XW. A survey of real-time scheduling on multiprocessor systems. In: Proc. of the 39th National Conf. of Theoretical Computer Science. Yinchuan: Springer, 2021. 89–118. [doi: 10.1007/978-981-16-7443-3_7]

[14] Liu CL, Layland JW. Scheduling algorithms for multiprogramming in a hard-real-time environment. Journal of the ACM, 1973, 20(1): 46–61. [doi: 10.1145/321738.321743]

[15] Joseph M, Pandya P. Finding response times in a real-time system. The Computer Journal, 1986, 29(5): 390–395. [doi: 10.1093/comjnl/29.5.390]

[16] Chen JJ, Huang WH, Liu C. K2U: A general framework from k-point effective schedulability analysis to utilization-based tests. In: Proc. of the 2015 IEEE Real-time Systems Symp. San Antonio: IEEE, 2015. 107–118. [doi: 10.1109/RTSS.2015.18]

[17] Bertogna M, Cirinei M, Lipari G. Improved schedulability analysis of EDF on multiprocessor platforms. In: Proc. of the 17th Euromicro Conf. on Real-time Systems. Balearic Islands: IEEE, 2005. 209–218. [doi: 10.1109/ECRTS.2005.18]

[18] Garey MR, Johnson DS. Computers and Intractability: A Guide to the Theory of NP-Completeness. San Francisco: W. H. Freeman & Co., 1979.

[19] Dhall SK, Liu CL. On a real-time scheduling problem. Operations Research, 1978, 26(1): 127–140. [doi: 10.1287/opre.26.1.127]

[20] Oh Y, Son SH. Tight Performance Bounds of Heuristics for a Real-time Scheduling Problem. Charlottesville: University of Virginia,

1993.
[21] Sprunt B, Sha L, Lehoczky J. Aperiodic task scheduling for hard-real-time systems. Real-time Systems, 1989, 1(1): 27–60. [doi: 10.1007/BF02341920]
[22] Burchard A, Liebeherr J, Oh Y, Son SH. New strategies for assigning real-time tasks to multiprocessor systems. IEEE Trans. on Computers, 1995, 44(12): 1429–1442. [doi: 10.1109/12.477248]
[23] Rothvoss T. On the computational complexity of periodic scheduling. Lausanne: EPFL, 2009. [doi: 10.5075/epfl-thesis-4513]
[24] Lopez JM, Díaz JL, Garcia DF. Minimum and maximum utilization bounds for multiprocessor rate monotonic scheduling. IEEE Trans. on Parallel and Distributed Systems, 2004, 15(7): 642–653. [doi: 10.1109/TPDS.2004.25]
[25] Fisher N, Baruah S, Baker TP. The partitioned scheduling of sporadic tasks according to static-priorities. In: Proc. of the 18th Euromicro Conf. on Real-time Systems. Dresden: IEEE, 2006. 118–127. [doi: 10.1109/ECRTS.2006.30]
[26] Chen JJ. Partitioned multiprocessor fixed-priority scheduling of sporadic real-time tasks. In: Proc. of the 28th Euromicro Conf. on Real-time Systems. Toulouse: IEEE, 2016. 251–261. [doi: 10.1109/ECRTS.2016.26]
[27] Baruah SK, Mok AK, Rosier LE. Preemptively scheduling hard-real-time sporadic tasks on one processor. In: Proc. of the 11th Real-time Systems Symp. Lake Buena Vista: IEEE, 1990. 182–190. [doi: 10.1109/REAL.1990.128746]
[28] Fisher N, Baruah S. A fully polynomial-time approximation scheme for feasibility analysis in static-priority systems with arbitrary relative deadlines. In: Proc. of the 17th Euromicro Conf. on Real-time Systems. Balearic Islands: IEEE, 2005. 117–126. [doi: 10.1109/ECRTS.2005.1]
[29] Hoare CAR. Algorithm 64: Quicksort. Communications of the ACM, 1961, 4(7): 321. [doi: 10.1145/366622.366644]
[30] Bini E, Nguyen THC, Richard P, Baruah SK. A response-time bound in fixed-priority scheduling with arbitrary deadlines. IEEE Trans. on Computers, 2009, 58(2): 279–286. [doi: 10.1109/TC.2008.167]
[31] Bini E, Buttazzo GC. Measuring the performance of schedulability tests. Real-time Systems, 2005, 30(1): 129–154. [doi: 10.1007/s11241-005-0507-9]

附中文参考文献:

[9] 陈熙, 乔磊, 杨孟飞, 刘洪标. 基于避让阻塞的优先级天花板协议. 软件学报, 2023, 34(7): 3422–3437. http://www.jos.org.cn/1000-9825/6534.htm [doi: 10.13328/j.cnki.jos.006534]
[11] 刘洪标, 乔磊, 杨孟飞, 陈熙, 马智, 李少峰. 基于周期虚拟缩减的实时任务调度和分析方法. 软件学报, 2022, 33(9): 3512–3528. http://www.jos.org.cn/1000-9825/6394.htm [doi: 10.13328/j.cnki.jos.006394]

刘洪标(1995－), 男, 博士, 主要研究领域为嵌入式操作系统, 实时任务调度.

姜菁菁(1996－), 女, 博士, 主要研究领域为嵌入式操作系统, 形式化验证, 文件系统.

宋程昊(1997－), 男, 硕士, 主要研究领域为嵌入式操作系统, 任务调度.

乔磊(1982－), 男, 博士, 研究员, CCF 专业会员, 主要研究领域为操作系统模型设计, 存储管理, 文件系统.

王婷煜(1998－), 女, 博士, 主要研究领域为嵌入式操作系统, 任务调度, 文件系统.

杨孟飞(1962－), 男, 博士, 研究员, 博士生导师, CCF 高级会员, 主要研究领域为空间飞行器嵌入式系统, 控制系统, 总体技术.