DOI：10.7652/xjtuxb201407016

硬实时系统周期任务低功耗调度算法

张忆文[1,2]，郭锐锋[1]
（1.中国科学院沈阳计算技术研究所，110168，沈阳；2.中国科学院大学，100039，北京）

摘要：针对硬实时系统周期任务，提出了动态空闲时间回收算法(DSTRA)。该算法既能利用高优先级任务提早完成的空闲时间，也能利用低优先级任务产生的空闲时间，并且考虑了通用的功耗模型：处理器的动态功耗；处理器的静态功耗。DSTRA算法由两部分组成：在离线状态，确定任务集的静态运行的速度；在在线状态，根据任务集的真实负载，利用高优先级任务提前完成的空闲时间和低优先级所产生的空闲时间，调节处理器速度。实验结果表明：DSTRA算法与DRA(Dynamic Reclaiming Algorithm)和DSRDP(Dynamic Slack Reclamation with Dynamic Procrastination)算法相比节能效果更好，可以分别节约2.03%～27.57%和1.09%～17.04%的能耗。

关键词：实时系统；空闲时间回收；功耗调度
中图分类号：TP316.2 **文献标志码**：A **文章编号**：0253-987X(2014)07-0090-06

A Scheduling Algorithm with Low Power for Periodic Tasks in Hard Real-Time Systems

ZHANG Yiwen[1,2], GUO Ruifeng[1]
(1. Shenyang Institute of Computing Technology, Chinese Academy of Sciences, Shenyang 110168, China;
2. University of Chinese Academy of Sciences, Beijing 100039, China)

Abstract: A dynamic slack time reclaim algorithm (DSTRA) is proposed for periodic tasks in hard real-time systems. The algorithm reclaims the slack time from both the higher-priority tasks and the lower-priority tasks, and takes the common power model into account, that is, both the dynamic power and the static power are considered in the algorithm. The DSTRA algorithm consists of two parts: an off-line calculation to determine the static speed of running tasks, and an on-line speed adjusting mechanism based on the actual workload by using the slack time generated from both the completed higher-priority tasks and the lower-priority tasks. Simulation results and comparisons with the DRA algorithm and the DSRDP algorithm show that the DSTRA algorithm is more efficient, and provides about 2.03%-27.57% and 1.09%-17.04% energy savings, respectively.

Keywords: real-time systems; slack time reclaim; energy scheduling

近年来，随着超大规模集成电路技术的快速发展，CMOS电路的尺寸越来越小，处理器的能耗则越来越高。能耗的显著增长会导致系统能效降低，同时增加了冷却的成本，因此低功耗设计是实时系统的研究热点。动态电压调节(DVS)是一种有效的低功耗和能耗优化技术，它在保证实时任务不错过截止期限的基础上，根据处理器的负载情况动态地调整频率和电压以降低能耗。

研究者将实时调度算法和DVS技术结合起来，取得了丰富的研究成果[1-12]。文献[1]应用DVS技

收稿日期：2013-12-12。 作者简介：张忆文(1988—)，男，博士生；郭锐锋(通信作者)：男，研究员。 基金项目：国家科技重大专项资助项目(2012ZX01029001-002)。

术不仅降低了处理器的能耗，而且还减少了系统缓存的能耗。文献[2]提出了考虑速度切换开销的节能调度算法，该算法考虑了降低能耗对系统可靠性的影响。文献[3]为软实时任务的调度算法，通过预测周期软实时任务的负载，降低系统能耗。文献[4]提出了抢占阀值的固定优先级节能调度算法，该算法通过设置抢占阀值，减少系统的抢占次数，降低系统能耗。文献[5]分析了固定优先级任务的节能调度算法，但该算法不够灵活。文献[6]针对周期任务提出了 DRA(Dynamic Reclaiming Algorithm)算法，该算法在离线状态下确定任务的最佳运行速度，在在线状态下动态地回收系统的空闲时间，是一种高效的节能调度算法，但只考虑了高优先级任务提前完成的空闲时间，没有考虑低优先级任务的空闲时间。文献[7]对文献[6]进行了改进，即采用了任务最早释放优先策略，调度周期任务，利用高优先级任务提前完成的空闲时间，但没有考虑低优先级任务的空闲时间。文献[8]拓展了文献[6]的算法，不仅考虑了处理器的动态功耗，而且考虑了处理器的静态功耗，但该算法只考虑了高优先级任务提前完成的空闲时间，忽略了低优先级任务产生的空闲时间。

为了充分利用系统的空闲时间以取得更好的节能效果，本文提出了动态空闲时间回收算法(DSTRA)。

1　模型和研究动机

1.1　系统模型

考虑有 n 个周期任务的实时任务集 $T=\{T_1, T_2, \cdots, T_n\}$，并且使用最早截止期优先(EDF)调度策略[13]调度该任务集，每个周期任务 T_i 用二元组 (P_i, C_i) 表示，其中 P_i 是任务 T_i 的周期，C_i 是任务 T_i 最坏情况下的执行时间，并且假设任务的相对截止期限等于 P_i。假设用 A_i 表示任务 T_i 的真实执行时间，r_i 表示任务 T_i 的就绪时间，d_i 表示任务 T_i 的截止期限，$T_{i,j}$ 表示任务 T_i 的第 j 次调用，$U_{\text{tot}}=\sum_{i=1}^{n} C_i/P_i$ 表示在最大运行速度下，任务集的总利用率。同时，假设任务是相互独立且在 0 时刻就绪。

假设处理器提供连续的频率和电压，对频率进行归一化处理，速度的取值范围为 $[S_{\min}, 1]$。尽管目前的商用处理器提供的速度是离散的，但文献[9]提供的算法能够有效地解决处理器提供离散速度的情形，因此这个假设是合理的。忽略速度转换的一切开销，并且假设任务的执行时间和运行速度成线性关系，即任务 T_i 在速度 S 下的执行时间为 C_i/S。

1.2　功耗模型

处理器的功耗[10-11]主要由静态功耗 P_{s}、与速度无关的功耗 P_{ind}、与速度相关的功耗 P_{dep} 组成。P_{s} 主要来自基本的电路和保持时钟频率所消耗的功耗；P_{ind} 主要来自与速度无关组件产生的功耗，当处理器处于休眠状态时，该功耗忽略不计；P_{dep} 是与速度相关的功耗，可以表示为 $P_{\text{dep}}=C_{\text{ef}}S^m$，其中 C_{ef} 是有效的负载电容，S 为任务的运行速度，m 是与系统无关的常数($2\leqslant m\leqslant 3$)。因此，处理器的功耗模型可以表示为

$$P = P_{\text{s}} + h(P_{\text{ind}} + P_{\text{dep}}) = P_{\text{s}} + h(P_{\text{ind}} + C_{\text{ef}}S^m) \tag{1}$$

其中系数 h 为常数，当有任务执行时 $h=1$，否则 $h=0$。降低任务的运行速度，可以减少与速度相关的功耗，但降低速度会延长任务的执行时间，增加处理器的静态功耗。为了使总的能耗最低，文献[12]提出了关键速度 S_{crit} 的概念，该速度是能耗最优的运行速度。为了不使任务错过截止期限，本文中任务的运行速度不低于关键速度 S_{crit}。

1.3　动机实例

考虑有 3 个任务的周期任务集 $T=\{T_1, T_2, T_3\}$，任务的参数如下：$P_1=2, C_1=1, A_1=0.5$；$P_2=4, C_2=1, A_2=0.6$；$P_3=8, C_3=2, A_3=1$。任务集 T 用文献[6]所提出的 DRA 算法调度，调度的结果如图 1 所示。任务 $T_{1,0}$ 在相对执行时间 $t=0.5$ 时完成，留下 0.5 个单位的空闲时间；任务 $T_{2,0}$ 在相对执行时间 $t=0.5$ 时调度，以 1/1.5 的归一化速度(下文简称速度)执行，在 $t=1.4$ 时完成，并且留下 0.6 个单位的空闲时间；任务 $T_{3,0}$ 在 $t=1.4$ 时调度，以 2/2.6 的速度执行，在 $t=2$ 时被任务 $T_{1,1}$ 抢占，$T_{1,1}$ 在 $t=2.5$ 时完成执行，并且留下 0.5 个单位的空闲时间，同时 $T_{3,0}$ 以原来的速度恢复执行，在 $t=3.2$ 时完成，留下 1.3 个单位的空闲时间；$T_{1,3}$ 在 $t=4$ 时以 1/2 的速度执行，在 $t=5$ 时完成执行，留下 1 个单位的空闲时间；$T_{2,1}$ 在 $t=5$ 时以 1/2 的速度执行，在 $t=6.2$ 时完成执行，留下 0.8 个单位的空闲时间；最后 $T_{1,4}$ 在 $t=6.2$ 时以 1/1.8 的速度执行，在 $t=7.1$ 时完成执行。从图 1 可以看出：DRA 算法只利用了高优先级任务提早完成的空闲时间，而没有利用低优先级任务产生的空闲时间。例如任务 $T_{3,0}$ 被抢占时已经有部分完成执行，任务 $T_{1,1}$ 可以利用 $T_{3,0}$ 产生的空闲时间。

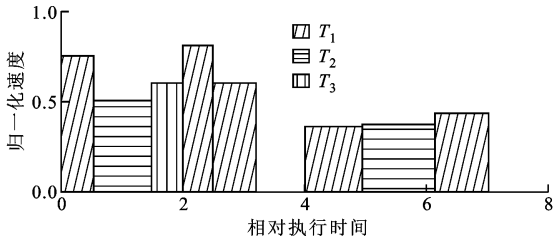
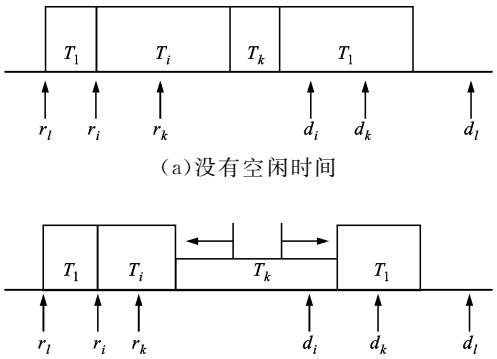


图 1 DRA 算法调度图

2 动态空闲时间回收算法

2.1 基本思路

算法的基本思路：一个任务既可以利用高优先级任务产生的空闲时间，也可以利用低优先级任务产生的空闲时间。任务 T_k 的空闲时间来源如图 2 所示。



(a)没有空闲时间

(b)高优先级和低优先级任务的空闲时间

图 2 任务 T_k 的空闲时间来源

图 2a 中任务没有可利用的空闲时间，任务 T_l 的优先级最低被任务 T_i 和 T_k 抢占，任务 T_k 完成执行，任务 T_l 恢复执行。图 2b 中最高优先级任务 T_i 提前完成执行，任务 T_k 回收空闲时间。同时，由于低优先级任务 T_l 已经有部分执行，所以在恢复执行时，仍然以最坏情况下的执行时间分析调度算法的可行性。因此，任务 T_k 可以回收其已经执行部分的时间。

为了更好地理解本文算法，重新执行图 1 的任务集，执行结果如图 3 所示。

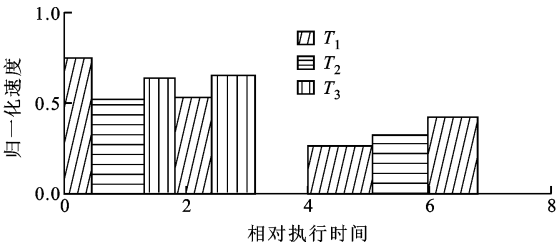


图 3 DSTRA 算法调度图

任务 $T_{3,0}$ 在 $t=1.4$ 时调度，以 2/2.6 的速度运行，在 $t=2$ 时被任务 $T_{1,1}$ 抢占，任务 $T_{3,0}$ 的剩余执行时间低于最坏情况下的执行时间，因为 $T_{3,0}$ 已经有一部分完成执行。通过计算可知 $T_{3,0}$ 在最坏情况下的相对剩余执行时间为 1.54，而其最坏情况下的相对执行时间为 2，会产生 0.46 个单位的空闲时间。因此，任务 $T_{1,1}$ 在 $t=2$ 时以 1/1.46 的速度执行，在 $t=2.73$ 时完成执行，留下 0.73 个单位的空闲时间。任务 $T_{3,0}$ 在 $t=2.73$时以原来的速度恢复执行，并且在 $t=3.43$ 时完成执行，留下 1.3 个单位的空闲时间。任务 $T_{1,3}$ 在 $t=4$ 时以 1/2.4 的 6 速度执行，在 $t=5.23$ 时完成执行，留下 1.23 个单位的空闲时间。任务 $T_{2,1}$ 在 $t=5.23$ 时以 1/2.23 的速度执行，在 $t=6.56$ 时完成执行，留下 0.89 个单位的空闲时间。最后 $T_{1,4}$ 在 $t=6.56$ 时以 1/1.89 的速度执行，在 $t=7.51$ 时完成执行。在此，使用文献[12]的功耗模型，该功耗模型可近似表示为 $0.08+1.52S^3$，关键速度 $S_{\text{crit}}=0.3$，处理器处于休眠状态的功耗为 0.085。通过计算可知：DSTRA 算法比 DRA 算法节约大约 13.37%的能耗。

2.2 离线状态任务的调度

在离线状态，周期任务都是以最坏情况下的负载运行。根据文献[13]，EDF 调度任务集是可行的，充要条件是系统的利用率 $U_{\text{tot}}\leqslant 1$。为了充分利用离线状态的空闲时间，提出离线状态的运行速度 S_{offline}，该速度可由定理 1 确定。

定理 1：保证所有任务的截止期限并且使总能耗最小的离线状态的运行速度为

$$S_{\text{offline}}=\max\{S_{\text{crit}},U_{\text{tot}}\}$$

在采用 EDF 调度策略调度时，利用速度 S_{offline} 可以充分利用处理器的资源并且保证调度是可行的。

证明：当关键速度 $S_{\text{crit}}\leqslant U_{\text{tot}}$ 时，$S_{\text{offline}}=U_{\text{tot}}$，这时充分利用了处理器的空闲时间，使得处理器的利用率为 1。当关键速度 $S_{\text{crit}}>U_{\text{tot}}$ 时，$S_{\text{offline}}=S_{\text{crit}}$，根据文献[12]，关键速度是使系统能耗最低的速度。因此，离线状态的运行速度为 S_{offline}。

2.3 计算空闲时间

为了有效地估计空闲时间，建立 α 队列[6]来记录提早完成的任务。α 队列的具体实现可以参见文献[6]。α 队列的元素是根据 EDF 策略排序的。$\text{rem}_i(t)$ 表示在速度 S_{offline} 下任务 T_i 在时刻 t 的剩余执行时间，$w_i(t)$ 表示在速度 S_{offline} 下任务 T_i 在时刻 t 的最坏情况下的剩余执行时间。

任务 T_x 在时刻 t 可利用的时间由 3 部分组成：

高优先级任务在时刻 t 提前完成所产生的空闲时间总和$S_H(T_x,t)$；任务 T_x 的剩余执行时间$\mathrm{rem}_x(t)$；低优先级任务部分完成所产生的空闲时间$S_L(T_x,t)$。

$P_H(T_x,t)$是时刻 t 之前已经完成的优先级比任务 T_x 高的任务集合。高优先级任务的空闲时间可表示为

$$S_H(T_x,t)=\sum_{T_i\in P_H(T_x,t)}\mathrm{rem}_i(t) \tag{2}$$

本文采用与文献[14]相似的方法来估计低优先级任务产生的空闲时间 $S_L(T_x,t)$，在时刻 t，任务 T_x 以速度 $w_x(t)/(S_H(T_x,t)+\mathrm{rem}_x(t))$执行，在时刻 $t_a=t+S_H(T_x,t)+\mathrm{rem}_x(t)$，任务 T_x 将完成执行。如果在时刻 t_a 之前有更高优先级的任务就绪，将抢占任务 T_x 的执行，否则任务 T_x 以原来的速度恢复执行。这时假设任务 T_x 的可利用时间仍为$S_H(T_x,t)+\mathrm{rem}_x(t)$，如果在时刻 t_a 之前没有更高优先级的任务抢占执行，任务 T_x 可以利用低优先级任务产生的空闲时间。在时刻 t_a 就绪队列为空的话，任务 T_x 的可利用执行时间扩展到最近的任务的到来时刻 t'_a。在时刻 t'_a 时 α 队列有空闲时间的话，任务 T_x 还可以利用空闲时间。T_β 表示在时刻 t_a 之前没有完成的优先级最高的任务，$P_L(T_x,t)$表示在时刻 t 之前已经完成的优先级比任务 T_x 低的任务集合，则

$$P'_L(T_\beta,t_a)=\{T_i\mid T_i\in P_L(T_x,t)\ \text{and}\ d_i\leqslant d_\beta\},$$

$$S_L(T_x,t)=\mathrm{rem}_\beta(t)-w_\beta(t)+\sum_{T_i\in P'_L(T_\beta,t_a)}\mathrm{rem}_i(t) \tag{3}$$

由式(3)计算的低优先级任务的空闲时间只有在时段$[t_a,t_a+S_L(T_x,t)]$或$[t'_a,t'_a+S_L(T_x,t)]$没有任务就绪时才有效。如果期间存在就绪的任务 T_λ，r_λ 是任务 T_λ 的到达时间，此时 $S_L(T_x,t)=r_\lambda-t_a$。任务的空闲时间可以由 caculate_salck_time 算法计算得出，描述如下。

caculate_salck_time 算法//计算 T_x 的空闲时间

{1. $S_L(T_x,t)=0$；t_x^h 和 t_β^h 分别为优先级高于 T_x 和 T_β 的最早到达任务实例的时间。//初始条件

2. 计算 $S_H(T_x,t)=\sum_{T_i\in P_H(T_x,t)}\mathrm{rem}_i(t)$；

3. 假如 $t+S_H(T_x,t)<t_x^h$　{

4. 设置 T_x 的速度为 $w_x(t)/(S_H(T_x,t)+\mathrm{rem}_x(t))$；

5. 如果在时刻 t_a 就绪队列只有任务 T_x

6.　$t_a=\max(t_a,t'_a)$；

7. $S_L(T_x,t)=\mathrm{rem}_\beta(t)-w_\beta(t)+\sum_{T_i\in P'_L(T_\beta,t_a)}\mathrm{rem}_i(t)$

8. $S_L(T_x,t)=\max(S_L(T_x,t),t_\beta^h-t_a)$；}

9. $\mathrm{slack\_time}=\min(S_H(T_x,t)+S_L(T_x,t),d_x-t)$。}

2.4　DSTRA 算法的伪代码

DSTRA 算法如下。

{1. 计算离线状态的速度 S_{offline}；

2. 用空链表初始化 α 队列。

3. 每个新任务 T_i 到达时，设置 $\mathrm{rem}_i(t)=w_i(t)$，根据 EDF 算法将任务插入到 α 队列的适当位置。

4. 当任务执行或完成时，α 队列队头的任务的 $\mathrm{rem}_i(t)$和$w_i(t)$逐渐减少，减少的时间为上次调度点到现在时刻所经过的时间。当 $\mathrm{rem}_i(t)$小于上次调度点到现在时刻所经过的时间，将任务从 α 队列移除，同时更新上次调度点到现在时刻所经过的时间。队列的下一个元素重复这个过程，直到队列为空。

5. 任务 T_i 在时刻 t 调度，在 α 队列中计算出任务可利用的空闲时间 slack_time 并且设置执行速度为 $w_i(t)/(\mathrm{slack\_time}+\mathrm{rem}_i(t))$。

6. 当任务 T_i 完成时，设置 $w_i(t)=0$；任务被抢占时设置 $w_i(t)=w_i(t)-\Delta_i S_i$，其中 S_i 为 T_i 的运行速度；执行时 $w_i(t)=w_i(t)-\Delta_t$，Δ_t 是任务在先前的调度中已经执行的时间。}

任务的空闲时间在 α 队列中总是按照任务的优先级排列，DSTRA 算法总是先使用高优先级任务的空闲时间，然后使用自己的空闲时间，最后使用低优先级任务的空闲时间。当任务被抢占时，以原来的速度恢复执行。当就绪队列中没有任务时，处理器进入休眠状态。算法的时间主要用在计算任务的空闲时间上，因此算法的时间复杂度为 $O(n)$。

3　仿真结果

为了评价 DSTRA 算法的节能效果，在 Pentium(R) Dual-core CPU 2.20 GHz、2 GB 的 RAM 的个人计算机上，利用 C 语言实现基于 EDF 调度策略的周期任务调度仿真器。该仿真器基于Intel XScale 270 处理器[15]，该处理器提供的速度范围为[0.15,1]，功耗模型可近似表示为 $P=0.08+1.52S^3$，关键速度 $S_{\mathrm{crit}}=0.3$，处理器处于休眠状态的功耗为 0.085。在该调度仿真器中实现了 4 种算法：NOD-VS 算法，该算法使用 EDF 调度任务集，任务始终以离线状态的速度运行，以它的能耗作为基准，进行归一化；DRA 算法[16]，该算法动态地回收高优先级任务的空闲时间，没有利用低优先级任务所产生的空闲时间；DSRDP 算法[12]，该算法将延迟调度技术和 DVS 技术相结合，但忽略了低优先级任务所产生

的空闲;DSTRA 算法,既考虑高优先级任务的空闲时间,也考虑低优先级任务所产生的空闲时间。

本文考察数控系统周期任务集,根据文献[16],该任务集包括 8 个周期任务,任务的周期为 2 400～9 600 μs。因此,任务的周期 P_i 随机选择,任务最坏情况下的执行时间(W)在[1,P_i]上选择,真实的负载可以通过修改最坏情况下的执行时间与最好情况下的执行时间(B)的比值 W/B 来实现。真实的负载服从[B,W]的均匀分布。每次实验的仿真时间设置为 100 000 个时间片。每个任务集运行 10 次,计算的能耗是 10 次实验结果的平均值。

3.1 真实负载对能耗的影响

将系统利用率 U_{tot} 设定为 0.5,考查真实负载对能耗的影响,如图 4 所示。从图 4 可以看出:任务集归一化的平均能耗依赖于任务集的真实负载。当 $W/B=1$ 时,没有动态空闲时间可以回收,DRA 算法与 DSRDP 算法的能耗相同,但都高于 DSTRA 算法的能耗,这是因为 DSTRA 算法在执行过程中能够利用低优先级任务产生的空闲时间。随着 W/B 值的逐渐增大,DRA 算法、DSRDP 算法以及 DSTRA 算法的归一化能耗波动不大,这是因为所有算法的能耗都以 NODVS 算法的能耗为基准进行归一化。可见:DSTRA 算法的能耗始终低于 DRA 和 DSRDP 算法的能耗,这说明 DSTRA 算法更有效。经过计算可知:DSTRA 算法与 DRA 算法相比可以节约大约 8.65%～14.09%的能耗,与 DSRDP 算法相比可以节约大约 7.88%～14.09%的能耗。

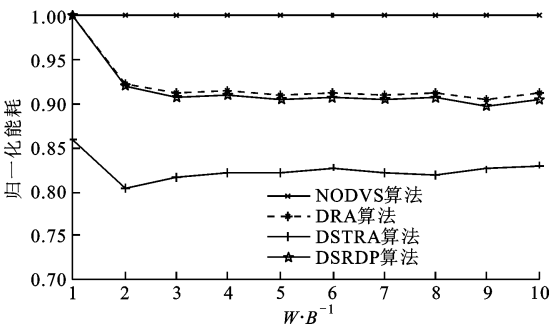


图 4 U_{tot} 为 0.5 时 W/B 对能耗的影响

3.2 利用率对能耗的影响

设置 W/B 的值为 5,考查系统利用率对能耗的影响,如图 5 所示。从图 5 可以看出:算法的能耗依赖于系统的利用率,DSTRA 算法的能耗始终低于其他算法的能耗。随着系统利用率的增大,所有算法的归一化能耗呈现出下降的趋势,这是因为其他算法的能耗增加的速度远低于 NODVS 算法的能耗增加的速度。DSTRA 算法节约的能耗逐渐增加,这是因为系统利用率的增加,使得 DSTRA 有更多的机会使用低优先级任务所产生的空闲时间。经过计算可知:DSTRA 算法与 DRA 算法相比可以节约大约 2.03%～27.57%的能耗,与 DSRDP 算法相比可以节约大约 1.09%～17.04%的能耗。

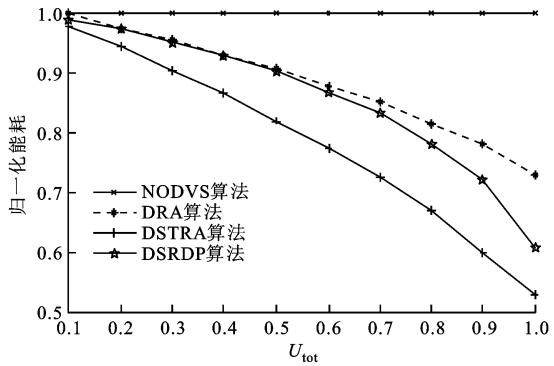


图 5 W/B 为 5 时 U_{tot} 对任务集归一化能耗的影响

4 结 论

目前基于空闲时间分析的 DVS 算法对空闲时间的估计过于保守,而本文提出的 DSTRA 算法不仅有效地估计了系统的空闲时间,而且估计空闲时间的开销非常小,为 $O(n)$,同时还考虑了系统通用的功耗模型,并且使用了关键速度的策略来降低系统层次的能耗。该算法由两部分组成:在离线状态,确定任务集的静态运行速度;在在线状态,不仅回收高优先级任务的空闲时间,而且回收低优先级任务所产生的空闲时间,降低系统的能耗。实验结果表明:DSTRA 算法与 DRA 算法和 DSRDP 算法相比节能效果更好,可以分别节约 2.03%～27.57%和 1.09%～17.04%的能耗。

参考文献:

[1] WANG Weixun, RANKA S, MISHRA P. A general algorithm for energy-aware dynamic reconfiguration in multitasking systems [C]// Proceedings of 24th Annual Conference on VLSI Design. Piscataway, NJ, USA: IEEE, 2011: 334-339.

[2] ZHANG Zhi, LI Fei, AYDIN H. Optimal speed scaling algorithms under speed change constraints [C]// Proceedings of International Conference on High Performance Computing and Communications. Piscataway, NJ, USA: IEEE, 2011: 202-210.

[3] KLUGE F, UHRIG S, MISCHE J, et al. Dynamic workload prediction for soft real-time applications [C]

// Proceedings of 10th IEEE International Conference on Computer and Information Technology. Piscataway, NJ, USA: IEEE, 2010: 1841-1848.

[4] YANG Liu, LIN Man. On-line and off-line DVS for fixed priority with preemption threshold scheduling [C]// Proceedings of International Conferences on Embedded Software and Systems. Piscataway, NJ, USA: IEEE, 2009: 273-280.

[5] CHEN Daren. An efficient DVS algorithm for fixed-priority real-time applications [C]// Proceedings of International Symposium on Parallel and Distributed Processing with Applications. Piscataway, NJ, USA: IEEE, 2010: 29-37.

[6] AYDIN H, MELHEM R, MOSS D. Dynamic and aggressive scheduling techniques for power-aware real-time systems [C]// Real-Time Systems Symposium. Piscataway, NJ, USA: IEEE, 2001: 95-105.

[7] QIAN Dejun, ZHANG Zhe, TIAN Xiaoming, et al. Low power scheduling for periodic real-time systems with dynamic voltage scaling processor [C]// Proceedings of International Conference on Computer Application and System Modeling. Piscataway, NJ, USA: IEEE, 2010: 244-248.

[8] AYDIN H, DEVADAS V, ZHU Dakai. System-level energy management for periodic real-time tasks [C]// Proceedings of the 27th IEEE International Real-Time Systems Symposium. Piscataway, NJ, USA: IEEE, 2006: 313-322.

[9] GONG M, SEONG Y, LEE C. On-line dynamic voltage scaling on processor with discrete frequency and voltage levels [C]// Proceedings of International Conference on Convergence Information Technology. Piscataway, NJ, USA: IEEE, 2007: 1824-1831.

[10] ZHANG Yiwen, GUO Ruifeng. Power-aware scheduling algorithms for sporadic tasks in real-time systems [J]. The Journal of Systems and Software, 2013, 86(10): 2611-2619.

[11] ZHAO Baoxian, AYDIN H, ZHU Dakai. Energy management under general task-level reliability constraints [C]// Proceedings of the 18th Real Time and Embedded Technology and Applications Symposium. Piscataway, NJ, USA: IEEE, 2012: 285-294.

[12] JEJURIKAR R, PEREIRA C, GUPTA R. Dynamic slack reclamation with procrastination scheduling in real-time embedded systems [C]// Proceedings of the 42nd Design Automation Conference. Piscataway, NJ, USA: IEEE, 2005: 111-116.

[13] LIU C, LAYLAND J. Scheduling algorithms for multiprogramming in a hard real time environment [J]. Journal of ACM, 1973, 20(1): 46-61.

[14] KIM W, KIM J, SANG L. A dynamic voltage scaling algorithm for dynamic-priority hard real-time systems using slack time analysis [C]// Design, Automation and Test in Europe Conference and Exhibition. Piscataway, NJ, USA: IEEE, 2002: 788-794.

[15] CHEN Jian-jia, KUO Tei-wei. Procrastination determination for periodic real-time tasks in leakage-aware dynamic voltage scaling systems [C]// Proceedings of IEEE International Conference on Computer-Aided Design. Piscataway, NJ, USA: IEEE, 2007: 289-294.

[16] KIM N, RYU M, HONG S, et al. Visual assessment of a real-time system design: a case study on a CNC controller [C]// Proceedings of Real-Time Systems Symposium. Piscataway, NJ, USA: IEEE, 1996: 300-310.

[本刊相关文献链接]

张雷，车立新，毕胜山，等. 天然气膨胀预冷混合制冷剂液化流程操作条件优化. 2014，48(2)：111-117. [doi：10.7652/xjtuxb201402019]

代亮，许宏科，陈婷. 无线传感器网络可分负载调度算法. 2012，46(6)：23-28. [doi：10.7652/xjtuxb201206005]

李健，黄庆佳，刘一阳，等. 云计算环境下的大规模图状数据处理任务调度算法. 2012，46(12)：116-122. [doi：10.7652/xjtuxb201212020]

曹仰杰，钱德沛，伍卫国，等. 一种面向多处理器系统的在线低功耗调度算法. 2010，44(8)：15-19. [doi：10.7652/xjtuxb201008004]

曹仰杰，杨海兵，钱德沛，等. 多核编程模型运行时环境的自适应性研究. 2011，45(6)：130. [doi：10.7652/xjtuxb201106023]

张良县，陈模生，彭宗仁，等. 一种单相阶梯接缝变压器铁芯的空载损耗计算方法. 2014，48(4)：52-58. [doi：10.7652/xjtuxb201404010]

李尊朝，罗诚，王闯，等. 部分耗尽异质环栅场效应晶体管阈值电压模型. 2013，47(12)：50-54. [doi：10.7652/xjtuxb201312009]

张鸿，赵桐，贺郁，等. 一种低功耗跨导电阻电容型镜像抑制复数滤波器. 2013，47(8)：80-86. [doi：10.7652/xjtuxb201308014]

王恒利，付立军，揭贵生，等. 采用比例谐振和状态反馈的三相逆变器最优控制. 2013，47(8)：127-132. [doi：10.7652/xjtuxb201308022]

胡周君，胡志刚，李林. 一种基于性能评估的元任务调度算法. 2008，42(8)：972-976. [doi：10.7652/xjtuxb200808010]

（编辑　赵炜）