软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2012,23(8):2223−2234 [doi: 10.3724/SP.J.1001.2012.04139]

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

基于双曲线边界的多处理器实时任务可调度性判定*

王洪亚[1+], 尹 伟[1], 宋 晖[1], 徐立群[2,3], 王 梅[1]

[1](东华大学 计算机科学与技术学院,上海 201620)

[2](成功大学 会计系,中国 台湾)

[3](长荣大学 资讯与工程学院,中国 台湾)

Schedulability Test for Multiprocessor Scheduling Based on Hyperbolic Bound

WANG Hong-Ya[1+], YIN Wei[1], SONG Hui[1], SHU Lih-Chyun[2,3], WANG Mei[1]

[1](School of Computer Science and Technology, Donghua University, Shanghai 201620, China)

[2](Department of Accountancy, Cheng Kung University, Taiwan, China)

[3](College of Information and Engineering, Chang Jung Christian University, Taiwan, China)

+ Corresponding author: E-mail: hywang@dhu.edu.cn

Wang HY, Yin W, Song H, Shu LC, Wang M. Schedulability test for multiprocessor scheduling based on hyperbolic bound. *Journal of Software*, 2012,23(8):2223−2234 (in Chinese). http://www.jos.org.cn/1000-9825/4139.htm

Abstract: The utilization bound for multiprocessor systems using the rate-monotonic scheduling algorithm and first fit allocation policy proposed by Lopez, *et al*. offers the best performance among all $O(m)$ complexity schedulability tests. In this paper, a utilization bound is derived for the same target problem. The main difference between these two bounds lies in the technique to verify the schedulability of task sets on uniprocessors; the schedulability test here is performed based on the hyperbolic bound proposed by Bini, *et al*. The new bound surpasses the existing one under quite a lot parameter settings, and the combination of these two schedulability test methods, which are compatible with each other, can significantly improve the number of schedulable task sets with little extra overhead.

Key words: hyperbolic bound; rate-monotonic scheduling algorithm; first fit allocation policy; schedulability test

摘 要: Lopez 等学者求解出基于单调速率算法和首次适应分派策略的多处理器实时任务可调度性判定边界.该边界在所有 $O(m)$复杂度的判定边界中是最优的.基于 Bini 等学者针对单处理器提出的双曲线可调度性判定方法,给出了一种多处理器实时任务可调度性判定边界.新边界在相当数量的利用率分布下明显优于已有边界.新边界与已有边界具有相容性,所以虽然新边界无法在所有情况下超越已有边界,但在实际应用中可联合两种边界进行判定,在不增加计算复杂度的同时全面提高可调度任务集的数量.

关键词: 双曲线边界;单调速率算法;首次适应分派策略;可调度性判定

中图法分类号: TP316 **文献标识码**: A

多处理器实时任务调度是实时系统中重要的研究问题之一.根据实时任务是否能够在处理器间动态迁移,

* 基金项目: 国家自然科学基金(60903160, 61103046); 上海市自然科学基金(11ZR1401200)
收稿时间: 2011-02-17; 定稿时间: 2011-11-03

实时任务调度策略分可为全局的和划分的两大类[1].由于划分策略具有更小的调度开销和实现代价,因而得到了学术界和产业界的普遍关注[2,3].在划分策略下,多处理器实时任务调度分两个步骤:

(1) 根据某种任务分派策略将实时任务分派到不同的处理器上,分派完成后任务将一直在指定的处理器上执行.

(2) 按照选定的实时调度算法对单个处理器上的任务实施调度.单处理器实时调度算法已较为成熟,因此,学术界研究的重点集中在设计和分析不同的任务分派策略上.

确定最优的任务分派策略是 NP-Hard 的,因此实用的任务分派策略都基于某种启发式规则.文献[1]首次使用性能指标 N_{Heu}/N_{opt} 来评估一个任务分派策略 *Heu* 的优劣,其中,N_{opt} 是最优分派策略所需的处理器数量,N_{Heu} 是 *Heu* 所需的处理器数量.虽然 N_{Heu}/N_{opt} 可用于判定某个启发式分派策略的优劣,但它无法判定在给定实时任务集、任务调度算法和分派策略的情况下,该任务集是否可以成功调度.N_{Heu}/N_{opt} 不可用的原因有两个:(1) N_{opt} 的求解是 NP-Hard;(2) 尽管 N_{opt} 已知,但文献[4]的分析表明,N_{Heu}/N_{opt} 的判定效率过低,即大量可调度的任务集会被判定为不可调度.

多处理器实时任务调度边界不等于单处理器可调度边界的简单累加.为了有效地解决多处理器实时任务的可调度性判定问题,文献[4]第一次给出了基于单调速率实时调度算法(rate monotonic,简称 RM)和首次适应分派策略(first fit,简称 FF)的多处理器实时任务调度边界 $U(n)$,如公式(1)所示.其中,n 为处理器的个数且 $n\geq 2$.

$$U(n)=n(2^{1/2}-1) \tag{1}$$

公式(1)表明,当一个实时任务集的利用率小于等于 $n(2^{1/2}-1)$时,应用 FF 和 RM 在 n 个处理器上一定可以调度该任务集.

Lopez 等学者在 2003 年给出了一种考虑任务集中最大任务利用率的可调度性判定边界 $U(m,n,\rho)$,如公式(2)所示.其中,m 为实时任务的个数,ρ为单个处理器上能够成功调度最大利用率任务的数量.

$$U(m,n,\rho)=(n-1)(2^{1/(\rho+1)}-1)\rho+(m-\rho(n-1))(2^{1/(m-\rho(n-1))}-1) \tag{2}$$

文献[2]分析表明,$U(m,n,\rho)$下可调度任务集空间是 $U(n)$下可调度任务集空间的超集.就我们所知,在 RM+FF 策略下,$U(m,n,\rho)$是目前最优的 $O(m)$复杂度判定边界.

文献[2,4]都采用 Liu 和 Layland 提出的单处理器可调度性边界,即 $\sum_{i=1}^{m}u_i\leqslant m(2^{1/m}-1)$[3],来判定单处理器上实时任务的可调度性.为了便于表述,我们将这两个边界所基于的调度策略简记为 FF+RM+Liu&Layland.

针对单处理器实时任务调度,Bini 等学者在 2003 年提出了与 Liu 和 Layland 边界计算复杂性相同的双曲线边界(hyperbolic bound).在双曲线边界下,如果 m 个实时任务满足 $\prod_{i=1}^{m}(1+u_i)\leqslant 2$,则该任务集在单个处理器上是可调度的[5].理论分析结果表明,双曲线边界下可调度任务集空间是 Liu 和 Layland 边界下可调度任务集空间的超集,且两者的比率为 $\sqrt{2}+O(m^{-1})$.

许多实验研究[6,7]表明,在多处理器实时任务调度中,使用双曲线边界代替 Liu 和 Layland 边界能够显著提高可调度任务集的数量.但就我们所知,国内外未见有学者在理论上给出 FF+RM+Hyperbolic-Bound 的可调度边界.随着多核/多处理器系统在实时应用中的日益普及,求解基于 FF+RM+Hyperbolic-Bound 的可调度判定边界具有理论和应用上的双重意义.为此,我们对该问题进行了研究.本文的主要贡献包括:

(1) 给出了基于首次适应分派策略、单调速率算法和双曲线边界的多处理器实时任务可调度性的边界,并对其正确性进行了证明.

(2) 将基于 FF+RM+Hyperbolic-Bound 的可调度性边界与现有边界进行了性能分析.模拟实验结果表明,新边界在相当多的利用率分布下明显优于已有边界.虽然新边界无法全面超越已有边界,但由于两种边界具有相容性,因此在不增加判定复杂度的前提下,联合应用两种判定方法能够全面提高可调度任务集的数量.

本文第 1 节给出系统模型和必要的符号定义.第 2 节完成基于 FF+RM+Hyperbolic-Bound 的可调度性判定

条件的推导.第 3 节对新边界和已有边界的性能进行对比.第 4 节是相关工作.第 5 节给出全文总结.

1 系统模型

本文的系统模型与文献[2]相同.假定系统中有 n 个相同的处理器 $P=\{P_1,P_2,\ldots,P_n\}$.设$\tau=\{\tau_1,\tau_2,\ldots,\tau_m\}$是$m(m>n)$个相互独立的周期性实时任务组成的任务集,其中,任务τ_i 的周期是为 T_i,执行时间是 C_i,$0<C_i<T_i$,任务的截止期与任务周期相同.τ_i 的利用率 $u_i=C_i/T_i$,任务集τ的利用率$U=\sum_{i=1}^{m}u_i$. 任务集中最大的任务利用率用α表示,即$\alpha=\mathrm{Max}\{u_i\}$,处理器 P_j 上已分派的任务集用 $S(P_j)$表示.

一旦τ_i成功分派到处理器P_j上,则τ_i将只在P_j上运行.任务分派策略采用首次适应法,即将τ_i,$1\leqslant i\leqslant m$ 分配到第 1 个能够接纳该任务的处理器P_j上,$1\leqslant j\leqslant n$.单个处理器采用单调速率算法对分派的实时任务进行调度.与文献[2]不同的是,本文中判定处理器 P_j 是否能够接纳τ_i 的条件不是传统的 Liu 和 Layland 边界,而是采用 Hyperbolic 边界,即 $(u_i+1)\prod_{\tau_k\in S(P_j)}(u_k+1)\leqslant 2$. 若$\tau=\{\tau_1,\tau_2,\ldots,\tau_m\}$可以在 $P=\{P_1,P_2,\ldots,P_n\}$上完成分派,我们称任务集τ在处理器集 P 上是可调度的.

2 基于 Hyperbolic Bound 的可调度性判定

定理 1 给出了 FF+RM+Hyperbolic-Bound 策略下,多处理器实时任务的可调度性判定条件.

定理 1. 给定任务集$\tau=\{\tau_1,\tau_2,\ldots,\tau_m\}$和处理器集合$P=\{P_1,P_2,\ldots,P_n\}$,若任务集的利用率满足公式(3)给出的判定条件,则τ在 P 上是可调度的.

$$\prod_{i=1}^{m}(u_i+1)\leqslant 2^{\frac{n\rho+1}{\rho+1}} \tag{3}$$

在公式(3)中,ρ定义为在 Hyperbolic Bound 下,单个处理器能够成功调度利用率为α的任务的最大数量.

证明思路:为了证明定理 1,我们需要首先证明后文中的 4 个引理.其中,引理 1 给出了ρ与α的关系式;引理 2 给出了 $B(n,\rho)$和 $B(n-1,\rho)$的递推关系式;利用该递推关系,引理 3 证明了 $B(n,\rho)\geqslant 2^{\frac{\rho n+1}{\rho+1}}$;引理 4 证明了 $B(n,\rho)\leqslant 2^{\frac{\rho n+1}{\rho+1}}$.综合引理 3 和引理 4 的结论,定理 1 得证. □

引理 1. $\rho=\left\lfloor\frac{1}{\log_2(\alpha+1)}\right\rfloor$.

证明:由α和ρ的定义可知,ρ个利用率为α的任务可以在单个处理器上成功调度.根据 Hyperbolic Bound 的定义,我们有$(\alpha+1)^{\rho}\leqslant 2$,即$\rho\leqslant 1/\log_2(\alpha+1)$.因为$\rho$是整数,所以$\rho\leqslant\lfloor 1/\log_2(\alpha+1)\rfloor$.因为$\rho$为单个处理器上能够成功调度利用率为$\alpha$的任务的最大数量,因此,$\rho+1$ 个利用率为α的任务一定无法在单个处理器上调度,所以我们有$(\alpha+1)^{(\rho+1)}>2$,即$\rho>1/\log_2(\alpha+1)-1$.因为$\rho$是整数,所以 $\rho\geqslant\left\lfloor\frac{1}{\log_2(\alpha+1)}\right\rfloor$.综合上述分析,引理 1 得证. □

由ρ的定义可知,如果 $m\leqslant\rho n$,则此 m 个任务必定可以在 n 个处理器上完成调度.因此,下面我们重点考虑 $m>\rho n$ 的情况.

假设任务集$\tau=\{\tau_1,\tau_2,\ldots,\tau_m\}$的可调度性边界为 $B(n,\rho)$,即 $\prod_{i=1}^{m}(u_i+1)\leqslant B(n,\rho)$ 时 m 个任务都是可以调度的,我们将首先证明下面 3 个引理,并基于这 3 个引理完成定理 1 的证明.

引理 2. $B(n,\rho)\geqslant 2^{\frac{\rho}{\rho+1}}\times B(n-1,\rho)$.

证明:为了证明引理 2,我们只需证明当$\prod_{i=1}^{m}(u_i+1)\leqslant 2^{\frac{\rho}{\rho+1}}\times B(n-1,\rho)$时,所有 $m(m>\rho n)$个任务都可以成功调度,其中,$B(n-1,\rho)$是只有 $n-1$ 个处理器时任务的可调度性边界.下面分两种情况加以证明.

- 情况 1:前 $m-\rho$个任务的利用率小于等于 $B(n-1,\rho)$,即 $\prod_{i=1}^{m-\rho}(u_i+1)\leqslant B(n-1,\rho)$. 由 $B(n,\rho)$的定义可知,前 $m-\rho$个任务肯定可以在 $n-1$ 个处理器上调度.由ρ的定义可知,后ρ个任务必定可以在第 n 个 CPU 上完成调度.因此,该 m 个任务都可以在 n 个处理器上完成调度.
- 情况 2:前 $m-\rho$个任务的利用率大于 $B(n-1,\rho)$,即 $\prod_{i=1}^{m-\rho}(u_i+1)>B(n-1,\rho)$. 在情况 2 下,我们假定任务集的 $\prod_{i=1}^{m}(u_i+1)=B(n-1,\rho)\times\Delta,\Delta\leqslant 2^{\frac{\rho}{\rho+1}}$, 并证明在该假设下 m 个任务都可以完成调度.

由情况 2 的定义可知,在前 $m-\rho$个任务中,必定存在一个任务τ_k,$1\leqslant k\leqslant m-\rho$,使公式(4)得以成立.

$$\prod_{i=1}^{k-1}(u_i+1)\leqslant B(n-1,\rho)<\prod_{i=1}^{k}(u_i+1) \tag{4}$$

容易看出,前 $k-1$ 个任务可以在 $n-1$ 个 CPU 上完成调度.为了证明的需要,定义 $\delta_k=B(n-1,\rho)\Big/\prod_{i=1}^{k-1}(u_i+1)$. 由公式(4)可知,前 k 个任务无法在 $n-1$ 个处理器上的完成调度,即 $B(n-1,\rho)<(u_k+1)\prod_{i=1}^{k-1}(u_i+1)$, 所以我们有$\delta_k<u_k+1$, 即$\delta_k-1<u_k$.值得注意的是,对$\tau_{k+1}$ 到τ_m 的所有任务,其利用率也必定都大于δ_k-1.否则,根据首次适应的分配法则,我们总可以将τ_{k+1}到τ_m中利用率小于等于δ_k-1 的任务分配到前 $n-1$ 个 CPU 上,直到没有这样的任务为止.

现在我们证明剩下的 $m-k+1$ 个任务可以在最后一个处理器上成功调度,首先由δ_k和Δ定义有:

$$\prod_{i=k}^{m}(u_i+1)=\delta_k\times\Delta \tag{5}$$

由于对τ_k到τ_m中的所有任务,其利用率都大于δ_k-1,因此我们有:

$$(\delta_k)^{m-k+1}<\delta_k\times\Delta<\delta_k\times 2^{\frac{\rho}{\rho+1}}.$$

由上式可以得到:

$$\delta_k<2^{\frac{\rho}{\rho+1}\times\frac{1}{m-k}}.$$

由公式(5)可得:

$$\prod_{i=k}^{m}(u_i+1)<2^{\frac{\rho}{\rho+1}\times\frac{1}{m-k}}\times 2^{\frac{\rho}{\rho+1}}=2^{\frac{\rho}{\rho+1}\times\frac{m-k+1}{m-k}}.$$

因为 $m-k>\rho$,有 $2^{\frac{\rho}{\rho+1}\times\frac{m-k+1}{m-k}}<2$, 即后 $m-k+1$ 个任务在 $\prod_{i=1}^{m}(u_i+1)=B(n-1,\rho)\times\Delta,\Delta\leqslant 2^{\frac{\rho}{\rho+1}}$ 的条件下可以在最后一个处理器上完成调度.因为 $\Delta\leqslant 2^{\frac{\rho}{\rho+1}}$, 综合上述分析有:当 $\prod_{i=1}^{m}(u_i+1)\leqslant 2^{\frac{\rho}{\rho+1}}\times B(n-1,\rho)$ 时,所有 $m(m>\rho n)$个任务都可以成功调度,从而对可调度性边界 $B(n,\rho)$有:

$$B(n,\rho)\geqslant 2^{\frac{\rho}{\rho+1}}\times B(n-1,\rho) \tag{6}$$

引理 2 得证. □

引理 3. $B(n,\rho)\geqslant 2^{\frac{\rho n+1}{\rho+1}}$.

证明:根据公式(6)蕴涵的递归关系,有:

$$B(n,\rho)\geqslant 2^{\frac{\rho}{\rho+1}}\times B(n-1,\rho),$$

$$B(n,\rho)\geqslant 2^{\frac{\rho}{\rho+1}}\times 2^{\frac{\rho}{\rho+1}}\times B(n-2,\rho),$$

…

$$B(n,\rho)\geqslant B(1,\rho)\times 2^{\frac{\rho(n-1)}{\rho+1}}.$$

由于 $B(n,\rho)\leqslant 2$,因此得到:

$$B(n,\rho)\geqslant 2^{\frac{\rho(n-1)}{\rho+1}}\times 2=2^{\frac{\rho n+1}{\rho+1}} \tag{7}$$

引理 3 得证. □

引理 4. $B(n,\rho)\leqslant 2^{\frac{\rho n+1}{\rho+1}}$.

证明:为了证明引理 4,我们只需证明必定存在一个任务集$\tau=\{\tau_1,\tau_2,\ldots,\tau_m\}$,当$\prod_{i=1}^{m}(1+u_i)=2^{\frac{n\rho+1}{\rho+1}}\times(1+\varepsilon)$,$\varepsilon\to 0^+$时,该任务集无法在 n 个处理器上完成调度.

为了完成证明,我们构造如下 m 个任务,其中,前 $m-\rho n$ 个任务构成第 1 个子任务集,该子任务集中每个任务的利用率满足公式(8):

$$1+u_i=2^{\frac{1}{(\rho+1)(m-n\rho)}} \tag{8}$$

后ρn 个任务构成第 2 个子任务集,该子任务集中每个任务的利用率满足公式(9):

$$1+u_i=2^{\frac{1}{\rho+1}}\times(1+\varepsilon)^{\frac{1}{\rho n}} \tag{9}$$

首先要证明构造出的两个任务利用率的合法性,即 $u_i\leqslant\alpha$.由ρ和α的定义可知,$\alpha>2^{\frac{1}{\rho+1}}-1$,因此对前 $m-\rho n$ 个任务有$u_i=2^{\frac{1}{(\rho+1)(m-n\rho)}}-1<2^{\frac{1}{\rho+1}}-1<\alpha$, 对后$\rho n$ 个任务有$u_i=2^{\frac{1}{\rho+1}}\times(1+\varepsilon)^{\frac{1}{\rho n}}-1$. 当$\varepsilon\to 0^+$时,根据$\alpha>2^{\frac{1}{\rho+1}}-1$和实数的连续性可知,必定可以找到一个正实数$\varepsilon$,使得$\alpha>2^{\frac{1}{\rho+1}}\times(1+\varepsilon)^{\frac{1}{\rho n}}-1=u_i$, 我们证明了构造任务的合法性.

由任务集的构造方式可知,第 1 个子任务集中的所有任务($m-\rho n$ 个)加上第 2 个子任务集中的前ρ个任务无法在第 1 个处理器上完成调度,这是因为

$$\prod_{i=1}^{m-\rho n}(1+u_i)\times\prod_{i=m-\rho n+1}^{m-\rho n+\rho}(1+u_i)=2^{\frac{1}{\rho+1}}\times 2^{\frac{\rho}{\rho+1}}\times(1+\varepsilon)^{\frac{1}{n}}=2(1+\varepsilon)^{\frac{1}{n}}>2.$$

然而,如果只取第 2 个子任务集的前$\rho-1$ 个任务,则当$\varepsilon\to 0^+$时,前 $m-\rho(n-1)-1$ 个任务在第 1 个处理器上可以完成调度,这是因为

$$\prod_{i=1}^{m-\rho n}(1+u_i)\times\prod_{i=m-\rho n+1}^{m-\rho n+\rho-1}(1+u_i)=2^{\frac{1}{\rho+1}}\times 2^{\frac{\rho-1}{\rho+1}}\times(1+\varepsilon)^{\frac{\rho-1}{\rho n}}=2^{\frac{\rho}{\rho+1}}(1+\varepsilon)^{\frac{\rho-1}{\rho n}}<2.$$

下面我们证明,后$\rho(n-1)+1$ 个任务当$\varepsilon\to 0^+$时无法在后 $n-1$ 个处理器上完成调度.

由于每个处理器至少可以调度ρ个任务,因此我们给后面每个处理器分配ρ个任务,直到最后一个处理器和最后$\rho+1$ 个任务,这$\rho+1$ 个任务的利用率当$\varepsilon\to 0^+$时为

$$\prod_{i=m-\rho}^{m}(1+u_i)=\left(2^{\frac{1}{\rho+1}}\times(1+\varepsilon)^{\frac{1}{n\rho}}\right)^{\rho+1}=2(1+\varepsilon)^{\frac{\rho+1}{n\rho}}>2.$$

即$\rho+1$ 个任务在最后一个处理器上无法完成调度(请注意,这也表明前面 $n-1$ 个处理器中的任意一个都无法调度第 2 个子任务集中的$\rho+1$ 个任务).

综上所述,当$\varepsilon\to 0^+$时,$\prod_{i=1}^{m}(1+u_i)=2^{\frac{1}{\beta+1}}\times 2^{\frac{\beta n}{\beta+1}}\times(1+\varepsilon)\to 2^{\frac{\beta n+1}{\beta+1}}$. 因此,对构造的任务集,即使$\varepsilon\to 0^+$,该任务集仍然无法在 n 个处理器上完成调度.所以我们有 $B(n,\rho)\leqslant 2^{\frac{\rho n+1}{\rho+1}}$, 引理 4 得证. □

由引理 4 还可以得出如下的推论:

推论 1. $B(n,\rho)=2^{\frac{\rho n+1}{\rho+1}}$ 是任务集$\tau=\{\tau_1,\tau_2,\ldots,\tau_m\}$可调度性判定的紧边界.

3 性能分析

可调度性边界的性能分析主要有两种方法:一种是在理论上推导出不同边界下可调度任务集空间的大小,如文献[5]对单处理器 Liu&Layland 边界和 Hyperbolic Bound 边界的对比;另一种是通过大规模实验来验证不同边界的性能.虽然理论分析方法可以精确地比较出不同边界的优劣,但正如文献[8]所指出的,许多情况下精确的理论分析是不可行的.在我们尝试从理论上对比基于 FF+RM+Liu&Layland 和 FF+RM+Hyperbolic-Bound 两种边界的过程中,出现了难以用解析方法求解的、具有复杂约束的多重积分公式.为此,我们遵循多处理器实时调度中广泛认可的实验方法学[8,9],通过大规模实验对两种边界进行了对比,并对实验结果给出了定性的理论分析.为了便于表述,以下将边界 $U(n)=n(2^{1/2}-1)$简记为 LL1,边界 $U(m,n,\rho)$简记为 LL2,本文求解出的新边界简记为 HB.

3.1 实验设计

本文的实验设计以文献[9]为基础,并在参数设置上进行了扩充.为了评估 3 种边界,本文为每组实验随机产生 1 000 000 个实时任务集,任务集中实时任务的利用率遵循下面 3 种分布:

1. 均匀分布:在 $\left(0,2^{\frac{1}{\rho}}-1\right)$ 的区间上随机产生实时任务的利用率,ρ取 1~20.不同的区间大小决定了所能产生任务的最大利用率,从而影响边界计算中ρ的取值.
2. 双峰分布:取(0,0.5),(0.5,1)两个均匀分布区段,当要生成一个新任务时,其利用率落在(0,0.5)区段的概率分别取 0.25,0.33,0.67 和 0.75.调节任务利用率落在不同区段的概率,可以控制大利用率任务与较小任务的比例.
3. 指数分布:利用率的均值分别取 0.1,0.25 和 0.5.

由于本文的可调度性判定方法只与实时任务的利用率相关,因此在模拟生成任务集时没有考虑具体的任务执行时间和截止期等参数.当每组实验开始时,对每个任务集初始生成 $n+1$ 个任务,剔除利用率大于 n 的任务集并进行补充,直到新的任务集数量达到 1 000 000.随后,不断地向每个任务集添加新任务,每添加一个新任务后,采用 3 种边界对该任务集进行可调度性判定.当所有任务集的利用率都大于 n 后,停止本组实验.为考察不同处理器数量下 3 种边界的表现,我们分别取 n=4,8,12,16.限于篇幅,本文没有列出所有实验结果.

实验的性能指标是可调度任务集比率(success ratio,简称 SR),即通过可调度性测试的任务集数量除以生成的任务集总数量.为了更细致地比较几种边界,我们将 0~n 划分为间隔为 0.01 的区段,并在每个区段统计 SR 的值.此外,我们还统计了 LL2 和 HB 能调度任务集的数量($Num(LL)$和 $Num(HB)$),以及 LL2 可以调度但 HB 不能调度任务的数量 ($Num(LL,\overline{HB})$) 与 HB 可以调度但 LL2 不能调度任务的数量 ($Num(\overline{LL},HB)$).

3.2 实验结果与分析

图 1~图 8 给出了利用率服从均匀分布时的实验结果(n=8,16),其中,x 轴表示任务集利用率,y 轴表示可调度任务集比率.当ρ较小(ρ=1,2)时,HB 的性能始终优于 LL2.LL1 由于没有考虑任务利用率大小这一因素,因此不如前两者.当ρ不断变大时,HB 的优势开始下降,并且在可调度任务总数上还略低于 LL2.这可以用以下两个观察来解释:

- 观察 1:假定 $u_i\in[0,1]$,若 $\sum_{i=1}^{m}u_i$ 为一个常数,则 $\prod_{i=1}^{m}(u_i+1)$ 的值在 u_i 相等时最大;u_i 的分布差异越大,$\prod_{i=1}^{m}(u_i+1)$ 的值变得越小.
- 观察 2:假定 $u_i\in[0,1]$,$m\geqslant n\rho+1$,若 $\sum_{i=1}^{m}u_i$ 为一个常数,则当 u_i 相等时,$U(m,n,\rho)$恒大于等于 $m\times\left(2^{\frac{\rho n+1}{m(\rho+1)}}-1\right)$ (HB 可调度任务集中所有任务利用率的总和).

当ρ较小(ρ=1,2)时,任务的最大利用率分别为 1 和$\sqrt{2}-1(\approx 0.414)$.由于任务利用率服从均匀分布,因此 u_i 的分布差异较大.由观察 1 可知,此时较小的$\prod_{i=1}^{m}(u_i+1)$值使得许多在 LL2 下不可调度的任务集在 HB 下变得可以调度;反之,当ρ较大(ρ>2)时,任务的最大利用率变得越来越小,且 u_i 的分布也趋于一致,因此$\prod_{i=1}^{m}(u_i+1)$的值不断增加.由观察 2 可知,在这种情况下,HB 可调度任务集的利用率很有可能小于 LL2 可调度任务集的利用率,因此,HB 边界的优势随着ρ的增加逐步下降,并最终略逊于 LL2.

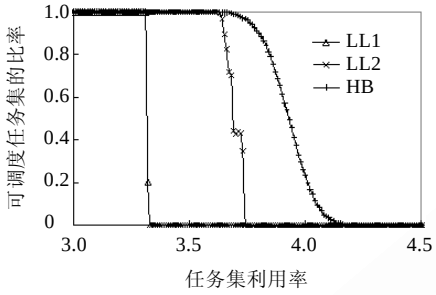


Fig.1 Success ratio (n=8, ρ=1)

图 1 n=8,ρ=1 时可调度任务集比率

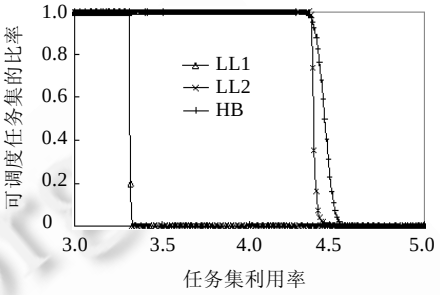


Fig.2 Success ratio (n=8, ρ=2)

图 2 n=8,ρ=2 时可调度任务集比率

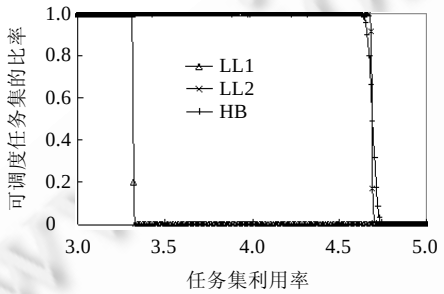


Fig.3 Success ratio (n=8, ρ=3)

图 3 n=8,ρ=3 时可调度任务集比率

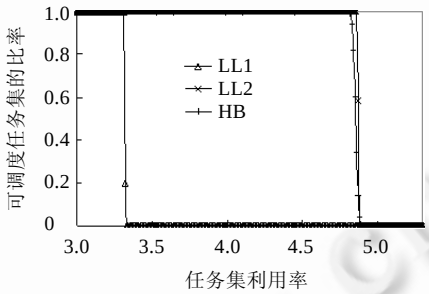


Fig.4 Success ratio (n=8, ρ=4)

图 4 n=8,ρ=4 时可调度任务集比率

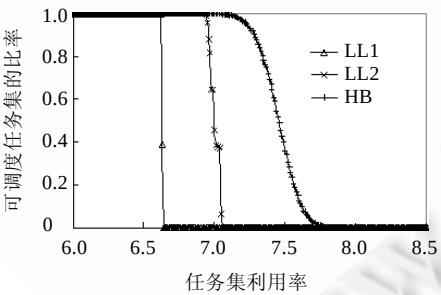


Fig.5 Success ratio (n=16, ρ=1)

图 5 n=16,ρ=1 时可调度任务集比率

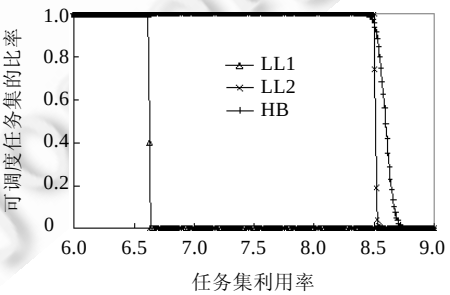


Fig.6 Success ratio (n=16, ρ=2)

图 6 n=16,ρ=2 时可调度任务集比率

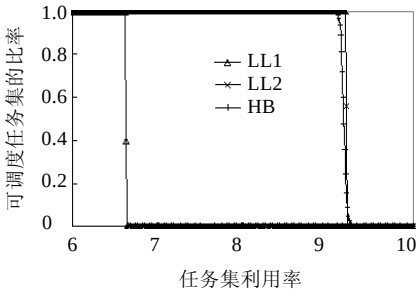


Fig.7 Success ratio (n=16, ρ=3)

图 7 n=16,ρ=3 时可调度任务集比率

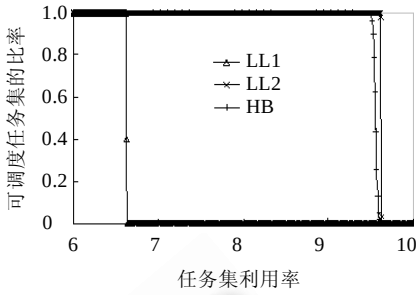


Fig.8 Success ratio (n=16, ρ=4)

图 8 n=16,ρ=4 时可调度任务集比率

表 1 给出了当 n=16 时,$Num(HB)/Num(LL)$随ρ的变化趋势.当ρ较小(ρ=1,2)时,HB 明显优于 LL2;当ρ不断变大时,两者的比值略低于 1,即 LL2 略优于 HB.但当ρ不断增大时,$Num(HB)/Num(LL)$不是不断变小,而是趋向于 1,这可以从下面的观察得到支持:

- 观察 3:当ρ趋于无穷大时,LL2 和 HB 的边界值分别趋于 $n\ln 2$ 和 2^n.即,当任务的利用率变得非常小(都近似相等)时,多处理器调度边界等于单个处理器上调度边界的和(对 LL2 边界而言)或积(对 HB 边界而言).

因为单处理器上 Liu&Layland 边界和双曲线边界在任务利用率相等时是一致的,因此,$Num(HB)/Num(LL)$在ρ趋于无穷大时趋向于 1.这表明 LL2 相对于 HB 的优势不会随着ρ的增加而一直增大,两种边界最终会随着ρ值的增加而趋于一致.

Table 1 Success ratio vs. ρ value (n=16)

表 1 当 n=16 时,可调度任务集比率随ρ值的变化趋势

当 n=16 时,ρ值的变化	ρ=1	ρ=2	ρ=3	ρ=4	ρ=6	ρ=8	ρ=12	ρ=16	ρ=20
$Num(HB)/Num(LL)$	1.757 7	1.015 5	0.995 5	0.991 6	0.991 0	0.991 9	0.993 7	0.994 9	0.995 8

图 9~图 12 给出了双峰分布下,当 n=16 时,HB,LL2 和 LL1 性能随任务集利用率的变化.可以看出,当任务集中任务利用率取(0,0.5)区段的数量不断增加时,HB 相对于 LL2 的优势不断下降.这与均匀分布下的结果是一致的,即任务集中的小利用率任务越多,HB 相对 LL2 的优势越小,同时也再次印证了观察 1 的有效性.由于双峰分布下产生ρ为 1 的任务集的概率非常大,因此,本组实验结果与均匀分布下ρ=1 时的结果近似.

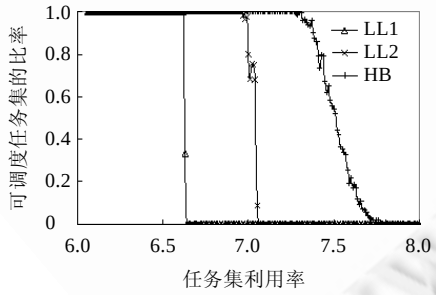


Fig.9 Task utilization falls into (0,0.5) wit Pr=0.25

图 9 任务利用率取(0,0.5)的概率为 0.25

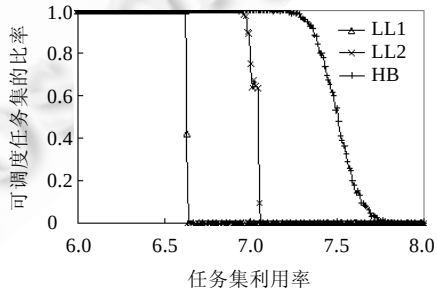


Fig.10 Task utilization falls into (0,0.5) wit Pr=0.33

图 10 任务利用率取(0,0.5)的概率为 0.33

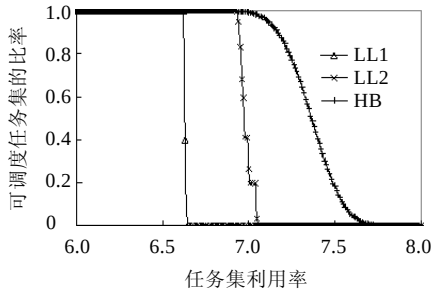


Fig.11 Task utilization falls into (0,0.5) wit Pr=0.67

图 11 任务利用率取(0,0.5)的概率为 0.67

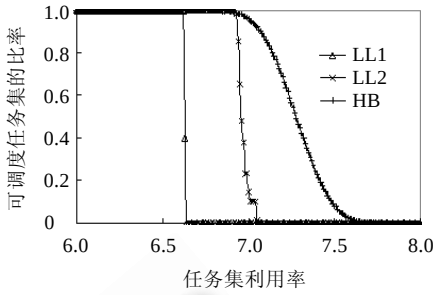


Fig.12 Task utilization falls into (0,0.5) wit Pr=0.75

图 12 任务利用率取(0,0.5)的概率为 0.75

图 13~图 15 给出了指数分布下,当 n=16 时,HB,LL2 和 LL1 的性能随任务集利用率的变化.可以看出,当指数分布的均值较大时(=0.5),HB 全面优于 LL2.这是由于此时任务集中任务利用率的分布差异较大,接近均匀分布.当均值变小时(=0.1,0.25),HB 的性能在许多情况下比 LL2 要差,甚至比 LL1 还要差.造成这种情况的原因有两个:一是小均值使得任务集中大量任务的利用率很小且互相接近,二是指数分布使得任务集中可能存在少量大利用率的任务.请注意,这在ρ为定值的均匀分布下是不会同时出现的.第 1 个原因导致 $\prod_{i=1}^{m}(u_i+1)$ 变得很大;第 2 个原因使得 $B(n,\rho)=2^{\frac{\rho n+1}{\rho+1}}$ 很小(ρ=1,2),并最终导致 HB 性能的下降.

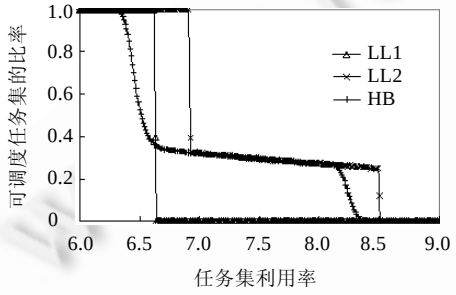


Fig.13 Exponential distribution with *mean*=0.1

图 13 任务利用率均值为 0.1

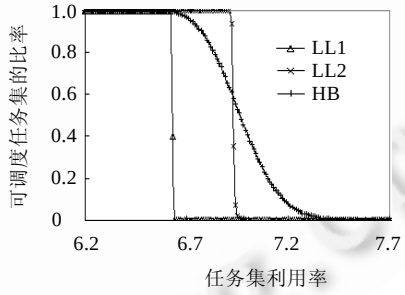


Fig.14 Exponential distribution with *mean*=0.25

图 14 任务利用率均值为 0.25

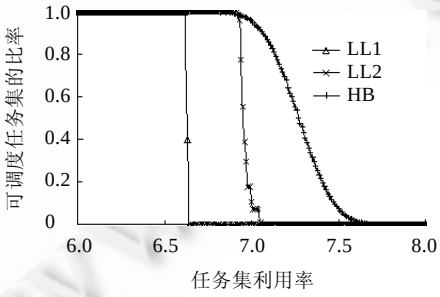


Fig.15 Exponential distribution with *mean*=0.5

图 15 任务利用率均值为 0.5

我们在实验中输出了大量不可调度任务集的构成细节,输出结果验证了我们的分析.在均值为 0.1 的设置

下,在 LL1 或 LL2 可调度;而 HB 不能调度的任务集中,绝大多数的ρ都为 1,且任务集中利用率小于 0.1 的比率超过一半以上.

3.3 HB与LL2的相容性

从实验结果可以看出,HB 和 LL2 分别在某些参数设置下优于对方.通过统计 $Num(LL,\overline{HB})$ 和 $Num(\overline{LL},HB)$ 的值我们发现,它们在绝大多数参数设置下都不等于 0.表 2 给出了均匀分布下,当 m=16,ρ等于 1~4 时,$Num(LL,\overline{HB})$ 和 $Num(\overline{LL},HB)$ 的值.从 $Num(LL,\overline{HB})$ 和 $Num(\overline{LL},HB)$ 的统计结果,我们得出了 HB 和 LL2 可调度任务集空间的关系,如图 16 所示.

按照文献[10]的分类,HB 和 LL2 的关系是不可比的(incomparable),即两种边界都无法全面胜过对方.然而,由于 HB 和 LL2 都基于 FF+RM 调度策略,因此两者的关系是相容而非互斥的.在实际应用中,给定一个实时任务集,可以同时使用两种边界对其进行判定.只要在任何一种(或两种都)边界下通过了判定,该任务集在 FF+RM 策略下就是可以调度的.值得注意的是,由于两种判定的计算复杂度都为 $O(m)$,因此联合判定并未增加计算的复杂度.

图 17~图 19 给出了不同任务利用率分布下(n=16),使用联合判定时可调度任务集数量与单独使用 LL2 时可调度任务集数量的比值.可以看出,联合判定在没有增加计算复杂度的情况下,全面提高了可调度判定的性能.

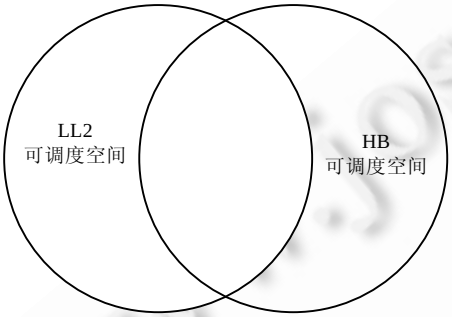


Fig.16 Relation of LL2 and HB

图 16 LL2 和 HB 的关系

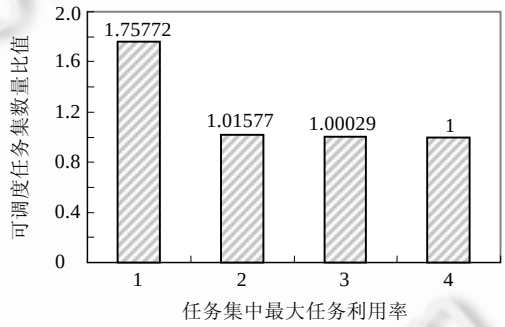


Fig.17 Schedulability improvement vs. ρ value

图 17 任务集可调度性的提升随ρ的变化

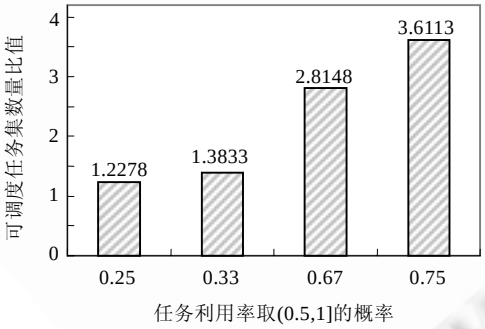


Fig.18 Schedulability improvement under bi-model distribution

图 18 任务集可调度性的提升随双峰分布参数的变化

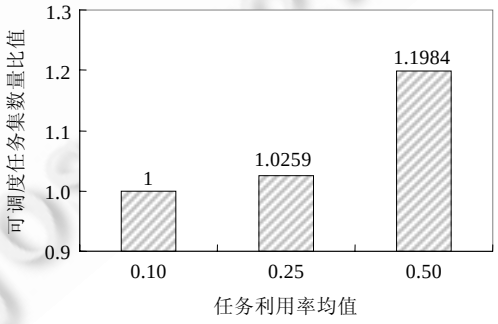


Fig.19 Schedulability improvement under exponential distribution

图 19 任务集可调度性的提升随指数分布均值的变化

Table 2 $Num(LL,\overline{HB})$ and $Num(\overline{LL},HB)$ values (n=16)

表 2 n=16 时,$Num(LL,\overline{HB})$和$Num(\overline{LL},HB)$的值

实验设置	$Num(LL,\overline{HB})$	$Num(\overline{LL},HB)$
m=16, ρ=1	1	353 238
m=16, ρ=2	7 233	432 934
m=16, ρ=3	283 527	17 063
m=16, ρ=4	770 856	16

4 相关工作

从 Dhall 和 Liu 的开创性论文开始[1],学术界对多处理器实时任务调度问题进行了大量研究.文献[10]对该问题做了很好的总结和回顾,指出虽然该领域已经取得了很多的研究成果,但仍有不少 Open Problems 需要继续探究.其中,如何不断提高可调度性边界就是几个待研究问题中的一个.为了有效地解决 RM+FF 策略下多处理器实时任务的可调度性判定问题,文献[4]第一次给出了调度边界 $U(n)$.随后,Lopez 等学者于 2003 年给出了一种考虑任务集最大任务利用率的可调度性判定边界 $U(m,n,\rho)$.该边界是目前最优的 $O(m)$复杂度判定边界[2].

国内的许多学者对单处理器和多处理器实时调度进行了深入研究.文献[11]对单处理器的 RM 可调度性边界进行了对比研究,提出了一般性的应用原则.文献[12,13]研究了多处理器环境下的动态调度问题,分别提出了适合不同应用场景的动态多处理器实时调度算法,这些研究成果与本文工作是互补的.

5 结 论

多处理器实时任务可调度性判定是多处理器实时调度理论研究中的一个重要问题.本文基于单处理器的双曲线可调度性判定方法,给出了一种基于单调速率算法和首次适应分派策略的多处理器可调度性判定边界,新边界在许多参数设置下优于文献[2]提出的可调度性边界.由于新边界与已有边界是相容的,因此可以同时使用两种边界进行可调度性判定,从而大大提高可调度任务的数量.

致谢 感谢审稿人中肯的审稿意见.

References:

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

[2] Lopez JM, Diaz JL, Garcia M, Garcia DF. Utilization bounds for multiprocessor rate-monotonic scheduling. Real-Time Systems, 2003,24(1):5−28. [doi: 10.1023/A:1021749005009]

[3] 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]

[4] Oh D, Baker TP. Utilization bound for n-processor rate monotone scheduling with static processor assignment. Real-Time Systems, 1998,15(2):183−192. [doi: 10.1023/A:1008098013753]

[5] Bini E, Bunazzo GC, Buttazzo GM. Rate monotonic analysis: The hyperbolic bound. IEEE Trans. on Computers, 2003,52(7): 933−942. [doi: 10.1109/TC.2003.1214341]

[6] Lupu I, Courbin P, George L, George L, Goossens J. Multi-Criteria evaluation of partitioning schemes for real-time systems. In: Proc. of 15th IEEE Int'l Conf. on Emerging Technologies and Factory Automation. Dallas: IEEE Computer Society, 2010. 1−8.

[7] Alenawy TA, Aydin H. Energy-Aware task allocation for rate monotonic scheduling. In: Proc. of the 11th IEEE Real-Time and Embedded Technology and Applications Symp. (RTAS 2005). Dallas: IEEE Computer Society, 2005. 213−223. [doi: 10.1109/RTAS.2005.20]

[8] Bini E, Buttazzo GC. Measuring the performance of schedulability tests. Real-Time Systems, 2005,30(1-2):129−154. [doi: 10.1007/s11241-005-0507-9]

[9] Baker TP. A comparison of global and partitioned EDF schedulability tests for multiprocessors. In: Proc. of the Int'l Conf. on Real-Time and Network Systems (RTSN). 2006. 119−130.

[10] Davis RI, Burns A. A survey of hard real-time scheduling for multiprocessor systems. ACM Computing Surveys, 2011,43(4):1−44. [doi: 10.1145/1978802.1978814]

[11] Wang YJ, Chen QP. On schedulability test of rate monotonic and its extendible algorithms. Journal of Software, 2004,15(6): 799−814 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/799.htm

[12] Qiao Y, Wang HA, Dai GZ. Developing a new dynamic scheduling algorithm for real-time multiprocessor systems. Journal of Software, 2002,13(1):51−58 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/13/51.htm

[13] Bin XL, Yang YH, Jin SY. A new dynamic scheduling algorithm for teal time multiprocessor systems based on grouping and properly choosing policies. Chinese Journal of Computers, 2006,17(1):81−91 (in Chinese with English abstract).

附中文参考文献:

[11] 王永吉,陈秋萍.单调速率及其扩展算法的可调度性判定.软件学报,2004,15(6):799−814. http://www.jos.org.cn/1000-9825/15/799.htm

[12] 乔颖,王宏安,戴国忠.一种新的实时多处理器系统的动态调度算法.软件学报,2002,13(1):51−58. http://www.jos.org.cn/1000-9825/13/51.htm

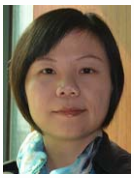
[13] 宾雪莲,杨玉海,金士尧.一种基于分组与适当选取策略的实时多处理器系统的动态调度算法.计算机学报,2006,17(1):81−91.

王洪亚(1976－),男,河南开封人,博士,副教授,主要研究领域为数据库理论与系统,实时计算,移动计算.

徐立群(1961－),男,博士,教授,博士生导师,主要研究领域为实时调度理论,基于位置的查询处理.

尹伟(1985－),男,硕士,主要研究领域为实时计算,数据管理.

王梅(1980－),女,博士,副教授,主要研究领域为数据管理,多媒体,信息检索.



宋晖(1971－),女,博士,副教授,CCF 会员,主要研究领域为 Web 数据挖掘,实时数据处理.