Journal of Computer Applications
计算机应用,2012,32(3):603-605,613
ISSN 1001-9081
CODEN JYIIDU
2012-03-01
http://www.joca.cn

文章编号:1001-9081(2012)03-0603-03
doi:10.3724/SP.J.1087.2012.00603

# 多处理器固定优先级算法的可调度性分析

白 露*,晏 立
(江苏大学 计算机科学与通信工程学院,江苏 镇江 212013)
(*通信作者电子邮箱 bailujs@163.com)

**摘 要**:针对多处理器实时调度中的固定优先级(FP)调度算法,提出了一种改进的可调度性判定方法。引入Baruah的最早截止期优先(EDF)窗口分析框架,将高优先级任务带入作业的最大数量限定为$m-1$($m$为处理器个数),进而对任务的干涉上界进行重新界定,并由此得到一个更加紧密的可调度性判定充分条件。仿真实验结果表明,该方法增加了通过判定任务集的数量,体现出更优的可调度判定性能。

**关键词**:多处理器;实时调度;固定优先级;可调度性判定;干涉
**中图分类号**:TP301.6;TP316.2 **文献标志码**:A

# Analysis on schedulability of fixed-priority multiprocessor scheduling

BAI Lu*, YAN Li
(*School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang Jiangsu* 212013, *China*)

**Abstract:** Concerning the Fixed-Priority (FP) algorithm of multiprocessor real-time scheduling, an improved schedulability test was proposed. This paper applied Baruah's window analytical framework of Earliest Deadline First (EDF) to FP, bounded the max number of higher priority tasks doing carry-in by $m$-1 (with $m$ being the number of processors), and thus got a new upper bound of interference a task suffered. Then, a tighter sufficient condition to determine schedulability was derived. The simulation results show the schedulability test is more efficient by increasing the number of detected schedulable task sets.

**Key words:** multiprocessor; real-time scheduling; Fixed-Priority (FP); schedulability test; interference



## 0 引言

随着多处理器芯片的普及,多处理器平台上的实时应用系统开发也引起越来越广泛的关注。为了保障这些系统的时间约束性和高可靠性,可调度性判定亦成为实时系统调度理论研究的核心问题。自1973年Liu和Layland[1]对多处理器系统的可调度性判定问题进行研究后,越来越多的人开始研究多处理器平台的可调度性判定技术,近年来提出了一些切实有效的方法[2-10]。Baker[2-3]开创性地从任务错过其截止期的角度出发,对全局最早截止期优先(Earliest Deadline First, EDF)和固定优先级(Fixed-Priority, FP)算法进行了研究,分别得到了一个可调度性判定的充分条件;Bertogna等[4]对实现工作保持的调度算法进行研究,得到任务集满足可调度性判定的充要条件,但由于没有理想复杂度的方法计算任务受到的干涉,使用工作负载作为干涉上界进行替代,得到了一个充分的可调度判定——BCL判定[5]。然而,得到的工作负载取值过于悲观,导致了大量的可调度任务集不能通过判定。

本文针对全局FP调度算法的可调度性判定问题,引入文献[6]中的窗口分析框架,分析限定高优先级任务带入作业的最大数量,求得一个更加逼近真实的任务干涉上界,改进了BCL判定方法,由此得到一个更加紧密的可调度性判定充分条件。通过实验将改进后的判定方法和BCL判定方法进行了比较,验证了改进后的方法可以检测到更多的可调度任务集,具有更高的可调度判定性能。

## 1 系统模型及相关定义

讨论的内容基于以下系统模型:

1)实时多处理器系统由$m$个具有相同处理能力的处理器组成,且任务在它们间可以互相迁移。

2)每个任务的不同作业可以在不同的处理器上执行,但单个作业只能在同一个处理器上执行。

3)任务之间是相互独立的,不存在先后次序约束,不存在除处理器外的资源访问冲突;任务之间可抢占,任务上下文切换、迁移和调度的开销忽略不计。

4)任务集中任务按照优先级由高到低的顺序排列,若任务$\tau_i$的优先级高于$\tau_j$,则有$1 \leqslant i < j \leqslant n$;任务除非被抢占;否则不会自挂起。

5)使用离散时间的概念,任意时刻$t$为非负整数且表示整个区间$[t, t+1)$。

定义$\tau = \{\tau_1, \tau_2, \cdots, \tau_n\}$为一个含有$n$个周期或偶发任务的集合,集合中的任务用三元组描述为:$\tau_k = \{C_k, D_k, T_k\}(k = 1, 2, \cdots, n)$。其中$C_k$、$D_k$、$T_k$分别表示任务$\tau_i$的最坏执行时间、相对截止期、周期任务执行周期或偶发任务最小到达周期。满足关系$C_k \leqslant D_k \leqslant T_k$,任务$\tau_k$的利用率表示为$U_k = C_k/T_k$。

任务是由作业构成的无穷序列,用$j_k^i$表示任务$\tau_k$的第$i$个作业,与作业$j_k^i$相关联的时间有释放时间$r_k^i$、计算时间$c_k^i$、

**收稿日期**:2011-09-13;**修回日期**:2011-11-21。
**基金项目**:国家自然科学基金资助项目(61005017);江苏省高校自然科学基金资助项目(10KJB520005)。
**作者简介**:白露(1987-),男,河南南阳人,硕士研究生,主要研究方向:实时系统; 晏立(1951-),男,江苏镇江人,教授,主要研究方向:实时系统、信息安全。

绝对截止期 $d_k^i$ 以及完成时间 $f_k^i$，并满足：

$$r_k^i \leqslant f_k^i \leqslant d_k^i$$

$$r_k^i \geqslant r_k^{i-1} + T_k$$

$$d_k^i = r_k^i + D_k$$

由于任务会受到高优先级任务的干扰，一个低优先级任务的各个作业的完成时间是不同的。任务 $\tau_k$ 的第 $i$ 个作业 $j_k^i$ 的响应时间定义为从释放时间到完成时间的区间 $[r_k^i, f_k^i)$。任务 $\tau_k$ 的最坏响应时间 $R_k$ 为该任务所有作业响应时间中的最大值。若任务 $\tau_k$ 的响应时间不能满足其截止期要求，则称任务 $\tau_k$ 发生超时。如果系统中的每个实时任务都能保证在截止期内完成，这个实时系统称为可调度的。

**定义 1** 干涉 $I_k(a,b)$ 指任务 $\tau_k$ 在区间 $[a,b)$ 中就绪，但因为有更高优先级任务执行而等待处理器的累积时间长度。$I_k^i(a,b)$ 指任务 $\tau_k$ 在区间 $[a,b)$ 中就绪，但因更高优先级任务 $\tau_i$ 执行而等待的累积时间长度。$\overline{I_k}$ 表示任务 $\tau_k$ 在最坏情况下所受到的干涉，与之对应的作业记为 $j^*$。任务 $\tau_i$ 对该作业的最大干涉为 $\overline{I_k^i} = I_k^i(r_k^{j^*}, d_k^{j^*})$。

**定义 2** 工作负载 $W_k(a,b)$ 为任务 $\tau_k$ 在区间 $[a,b)$ 中所有作业执行时间的总和。$\overline{\omega}_k(L)$ 表示在长度为 $L$ 的时间区间上任务 $\tau_k$ 的工作负载上界。

显然，在区间 $[a,b)$ 内干涉和工作负载满足关系：

$$\forall i,k,a,b \; I_k^i(a,b) \leqslant W_i(a,b) \leqslant b-a \tag{1}$$

## 2 可调度性分析

本文研究的可调度性判定基于任务截止期分析。该方法由 Baker[2] 提出并被广泛使用，其主要步骤如下：

1）假设在一时间区间末端任务 $\tau_k$ 的作业 $j_k^i$ 错过其截止期 $d_k^i$ 发生超时，称该区间为问题窗口（problem window）。

2）给出作业发生超时的一个必要条件：在问题窗口内作业 $j_k^i$ 必然受到高优先级任务的干涉，$m$ 个处理器执行高优先级任务的时间都大于 $D_k - C_k$，从而导致超时。

3）计算问题窗口内任务受到的总干涉量。由于目前没有合理的方法来精确计算任务受到的干涉量，只能用任务受到的干涉上界来替代，显然这个上界越逼近真实就越精确。

4）得出干涉上界和发生超时所需时间的关系，从而导出可调度性判定充分条件。

基于上述分析方法，Bertogna 等[5] 根据式(1)将高优先级任务的工作负载作为干涉上界。假设每个高优先级任务均产生带入作业，求得任务的干涉上界，进而导出 BCL 判定。对比其他判定方法尽管可调度判定性能显著提高，但依然存在着不足：该判定方法在分析任务受到的干涉时，假设每个高优先级任务都存在带入作业，这是一个过于悲观的假设。下面将对之进行分析说明。

### 2.1 带入作业数量分析

针对多处理器调度的带入作业数量问题，将文献[6]中针对 EDF 调度算法的窗口分析框架应用到 FP 上来：假设任务 $\tau_k$ 为任务集中第一个发生超时的任务。其对应超时作业 $j_k$ 在时刻 $f_k$ 发生超时，$r_k$ 为该作业的释放时刻，$t_o$ 为 $r_k$ 时刻前处理器的最后空闲时刻（在区间 $[t_o - 1, t_o)$ 内至少有一个处理器不被高优先级任务占用，而在区间 $[t_o, r_k)$ 内所有的处理器都被高优先级任务占用）。这样要研究的问题窗口就扩展为 $[t_o, f_k)$。根据时刻 $t_o$ 的定义，在问题窗口 $[t_o, f_k)$ 内我们得到以下定理。

**定理 1** 在使用全局 FP 算法进行调度的多处理器系统中，若任务 $\tau_k$ 的作业在 $t_o$ 时刻释放，则 $\tau_k$ 在问题窗口受到高优先级任务的干涉达到最大值。

证明 假设任务 $\tau_k$ 的超时作业在时刻 $x(x \neq t_o)$ 释放，在 $x$ 代表的区间 $[x, x+1)$ 内，可以分为以下两种情况：

1）所有的处理器均被高优先级任务占用。根据时刻 $t_o$ 的定义，在区间 $[t_o, x]$ 内 $m$ 个处理器都被占用。若任务 $\tau_k$ 作业的释放时间由 $x$ 前移至 $t_o$，则作业的完成时间不会改变。因此，作业在时刻 $t_o$ 比在时刻 $x$ 释放具有更大的响应时间，相应的受到的干涉量也会更大。

2）所有的处理器并不都在被高优先级任务占用。显然，在区间 $[x, t_o]$ 内至少有一个处理器可以用来执行任务 $\tau_k$ 的作业。将任务 $\tau_k$ 作业的释放时间由 $x$ 后移至 $t_o$，该作业的完成时间至少增加 $t_o - x$。因此，作业在时刻 $t_o$ 释放的响应时间至少等于在时刻 $x$ 释放的响应时间，相应的受到的干涉量也至少相等。

综上所述，任务 $\tau_k$ 的作业在 $t_o$ 时刻释放受到高优先级任务的干涉达到最大。 证毕。

根据 $t_o$ 的定义，在 $t_o - 1$ 时刻最多有 $m-1$ 个高优先级任务的作业正在执行。又由定理 1 可以得到如下推论：

推论 1 任务 $\tau_k$ 在问题窗口上受到最大干涉时，在问题窗口的开始时刻最多有 $m-1$ 个高优先级任务的带入作业。

通过推论 1 可知当任务受到最大干涉时高优先级任务带入作业的数量为 $m-1$，对比 BCL 判定的每个高优先级任务都产生带入作业大幅降低了任务干涉上界计算的悲观程度。

### 2.2 干涉上界分析

在长度为 $L$ 的问题窗口上，高优先级任务对 $\tau_k$ 的干涉可以分为两种情况：无带入作业的情况和有带入作业的情况。

#### 2.2.1 无带入作业情况的任务干涉

若 $\tau_i$ 没有带入作业，当 $\tau_i$ 第一个位于窗口内的作业在时刻 $t_o$ 释放并执行，随后的作业尽可能地接近其释放时刻执行时，该任务在区间上就会执行尽可能多的作业。如图 1 所示。

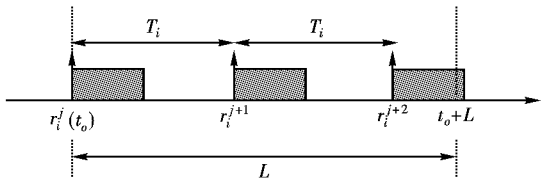


图 1 无带入作业的工作负载情况

任务 $\tau_i$ 位于区间内作业的最大数量为 $\left\lfloor \frac{L}{T_i} \right\rfloor$，则 $\tau_i$ 在该窗口的一个工作负载上界为：

$$\overline{\omega}_i^{\mathrm{NC}}(L) = \left\lfloor \frac{L}{T_i} \right\rfloor C_i + \min\left(C_i, L - \left\lfloor \frac{L}{T_i} \right\rfloor T_i\right)$$

其中 NC 表示无带入作业的情况。那么在长度为 $D_k$ 的窗口上任务 $\tau_k$ 受到无带入作业的高优先级任务 $\tau_i$ 的干涉上界为：

$$\overline{I_k^{i\,\mathrm{NC}}} = \min(\overline{\omega}_i^{\mathrm{NC}}(D_k), D_k - C_k + 1) \tag{2}$$

#### 2.2.2 有带入作业情况的任务干涉

若高优先级任务 $\tau_i$ 有带入作业，就会产生 BCL 判定方法指出的最坏情况。如图 2 所示。

在长度为 $L$ 的问题窗口上高优先级任务 $\tau_i$ 的一个工作负载上界为：

$$\overline{\omega}_i^{\mathrm{CI}}(L) = N_i(L)C_i + \min(C_i, L + D_i - C_i - N_i(L)T_i)$$

其中：$N_i(L)$ 是该任务在区间内所所执行作业的最大数量，CI 表示有带入作业的情况。

$$N_i(L) = \left\lfloor \frac{L + D_i - C_i}{T_i} \right\rfloor$$

那么在长度为 $D_k$ 的窗口上任务 $\tau_k$ 受到产生带入作业的高优先级任务 $\tau_i$ 的干涉上界为:

$$\overline{I}_k^{i\,\mathrm{CI}} = \min(\overline{\omega}_i^{\mathrm{CI}}(D_k), D_k - C_k + 1) \tag{3}$$

再由推论 1 及全局 FP 调度算法的工作保持特性,可以得出在长度为 $D_k$ 的区间上任务 $\tau_k$ 的一个干涉上界:

$$\overline{I}_k = \frac{1}{m}\left(\sum_{i<k} \overline{I}_k^{i\,\mathrm{NC}} + \sum_{i\in \max(k-1,m-1)} \overline{I}_k^{i\,\Delta}\right) \tag{4}$$

其中:$\overline{I}_k^{i\,\Delta} = \overline{I}_k^{i\,\mathrm{CI}} - \overline{I}_k^{i\,\mathrm{NC}}$,$\max(k-1, m-1)$ 表示最多 $m-1$ 个可以使 $\sum \overline{I}_k^{i\,\Delta}$ 取得最大值的高优先级任务的集合。可以看出式(4) 求得的干涉上界不大于 BCL 判定方出的干涉上界 $\frac{1}{m}\sum_{i<k} \overline{I}_k^{i\,\mathrm{CI}}$。

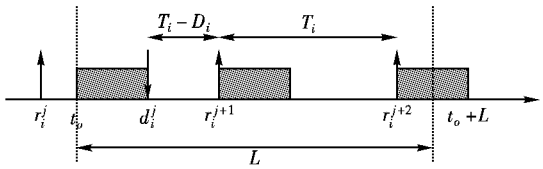


图 2　有带入作业的工作负载情况

### 2.3　可调度性判定

根据上面的分析,可以导出一个具有更加紧密干涉上界的充分判定条件。

**定理 2**　在由 $m$ 个相同处理器组成的多处理器平台上,若任务集 $\tau$ 中的每个任务 $\tau_k \in \tau$ 都满足:

$$\overline{I}_k < D_k - C_k + 1$$

则该任务集可使用全局 FP 算法进行调度。其中干涉上界 $\overline{I}_k$ 由式(4) 给出。

证明　任务 $\tau_k$ 受到高优先级任务 $\tau_i$ 的干涉分为两种情况:

当 $\tau_i$ 有带入作业时,对 $\tau_k$ 的干涉满足关系:

$$\overline{I}_k^{i\,\mathrm{CI}} = I_k^i(r_k^{j^*}, r_k^{j^*} + D_k) \leqslant W_i(r_k^{j^*}, r_k^{j^*} + D_k) \leqslant \overline{\omega}_i^{\mathrm{CI}}(D_k)$$

当 $\tau_i$ 无带入作业时对 $\tau_k$ 的干涉满足关系:

$$\overline{I}_k^{i\,\mathrm{NC}} = I_k^i(r_k^{j^*}, r_k^{j^*} + D_k) \leqslant W_i(r_k^{j^*}, r_k^{j^*} + D_k) \leqslant \overline{\omega}_i^{\mathrm{NC}}(D_k)$$

又根据推论 1 和式(4),受到所有高优先级任务的干涉满足关系:

$$\frac{1}{m}\sum_{i<k} \min(\overline{I}_k^i, D_k - C_k + 1) \leqslant$$

$$\frac{1}{m}\left(\sum_{i<k} \overline{I}_k^{i\,\mathrm{NC}} + \sum_{i\in \max(k-1,m-1)} \overline{I}_k^{i\,\Delta}\right) < D_k - C_k + 1$$

则有:

$$\sum_{i<k} \overline{I}_k^{i\,\mathrm{NC}} + \sum_{i\in \max(k-1,m-1)} \overline{I}_k^{i\,\Delta} < m(D_k - C_k + 1)$$

接下来的证明同文献[5]中 BCL 定理的证明。　证毕。

定理 2 限定了高优先级任务带入作业的数量,求得的任务干涉上界不大于 BCL 判定方法的干涉上界,比 BCL 判定更加逼近真实值。因此改进后的判定方法可以支配 BCL 判定方法,即可以被 BCL 判定方法检测到的可调度任务集同样也可以被改进后的方法检测到;反之则不然。

## 3　实验验证及结果分析

为了验证改进后判定方法的性能,分别对 BCL 判定和改进后的判定方法进行仿真对比。实验编程实现任务集的生成、判定方法及判定过程,最后绘制利用率/可调度任务集数量曲线图。

实验按照以下方式随机生成任务:任务的利用率 $U_k$ 服从以平均利用率为期望的指数分布,生成过程中排除 $U_k > 1$ 的情形;周期 $T_k$ 服从[1,1000] 内的均匀分布;相对截止期 $D_k$ 服从[$C_k$, $T_k$] 内的均匀分布。

实验流程如下:

1) 生成一个具有 $m+1$ 个任务的任务集,任务集计数加 1。

2) 对该任务集使用文献[7] 提出的任务集可行性必要条件进行检验。若任务集具有可行性,则使用给定的方法对该任务集进行可调度性判定;否则重复上一步重新生成任务集。

3) 若该任务集被判定为可调度任务集,则向该任务集中增加一个新任务以增大其总利用率,得到一个新的任务集,重复上一步的检验。

根据上述流程当任务集数量达到 $10^5$ 时停止。可以得到判定方法的利用率 / 可调度任务集数量的数据。我们改进的判定方法与 BCL 判定方法进行了比较。如图 3 显示了处理器数量 $m = 2$,任务集的平均利用率 $\sigma_u = 0.2$,运用 DM 算法进行调度的仿真结果。另外,还使用 $m = 4, 8, 16$ 和 $\sigma_u = 0.25, 0.3, 0.5$ 以及 RM 算法等条件重做了实验,得到的结果与图 3 类似,不再一一给出。

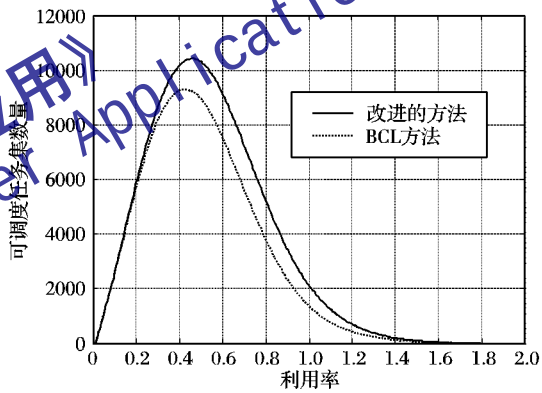


图 3　$\sigma_u = 0.2$ 和 $m = 2$ 的实验数据

由图 3 可看出:改进后的判定方法比 BCL 判定方法检测到更多的可调度任务集,体现出更高的可调度判定性能。根据文中的分析可以预见,当每个任务集中任务的数量远远大于处理器的个数($n \gg m$) 时,把高优先级任务带入作业的总数量限定在 $m-1$,对比 BCL 判定方法的所有高优先级任务都存在带入作业情况,可调度性能的提高将更加显著。

## 4　结语

本文对多处理器 FP 调度算法的可调度性进行研究,将文献[6] 中针对 EDF 算法的窗口分析框架应用到 FP 算法上,对高优先级任务带入作业的数量进行限定,得到更为精确的干涉上界,进而提出一个更为紧密的充分判定条件。实验结果显示本文的改进方法是行之有效的,在可调度性判定上表现出更优的性能。

**参考文献:**

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

[2] BAKER T P. Multiprocessor EDF and deadline monotonic schedulability analysis [C]// Proceedings of the 24th IEEE Real-Time Systems Symposium. Piscataway, NJ: IEEE Press, 2003: 120 - 129.

(下转第 613 页)

个数据包为单位,分了 5 组实验,每组实验进行 10 次,表 1 记录了 10 次实验测得的多核处理器的处理时间,最后对每组的统计结果求平均值。

3.2.1　CPU 执行效率分析

图 6 是 CPU 处理时间直方图,从中可看出:当数据包个数为 $8\times10^4$ 时,减少时间几乎为 0,而随着数据包个数的增加,在处理时间上均有减少,而且减少的幅度是和数据包个数的增加成正比的。

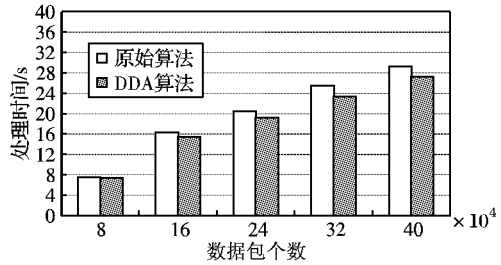


图 6　CPU 处理时间直方图

3.2.2　算法性能提升分析

算法改进前后节约时间及提升百分比如表 2 所示。

表 2　DDA 算法性能提升统计表

| 数据包个数 | 节约时间/s | 提升百分比/% |
|---|---|---|
| $8\times10^4$ | 0.146 | 2.0 |
| $16\times10^4$ | 1.017 | 6.2 |
| $24\times10^4$ | 1.374 | 6.7 |
| $32\times10^4$ | 2.060 | 8.2 |
| $40\times10^4$ | 1.948 | 6.7 |

从表 2 可看出:当数据包个数增加到 $16\times10^4$ 以后,这时无论是节约的绝对时间还是提升的百分比,都有明显增加。当数据包个数达到 $40\times10^4$ 时,这是算法性能的提升率趋向于稳定。通过上面的分析可知,整个算法经历了规则初始学习(性能提升很少)、规则学习逐步完成(性能提升明显)、规则学习逐步稳定(性能提升稳定)的过程。

## 4　结语

本文对 L7-Filter 的工作机制做了简要介绍,并说明了文献[4]对传统单核服务器上 L7-Filter 的改进方法及存在的不足。通过规则所采用的运输层协议对规则链进行分类,具体到每条规则链上利用网络数据流的流量统计模型与动态适应相结合的方法使得规则局部有序。在实验论证的基础上,证明了这种改进的模式匹配算法在 CPU 利用率和数据包的处理性能上的优越性。同时,由于整个规则的学习需要大规模网络流量的支持,所以本文算法在网络流量较小的情况下性能不理想,甚至较以往有一定的损耗,这部分损耗主要是核心间为了互斥并发访问规则链所带来的加/解锁操作。另外,为了降低核心间互斥并发访问规则链所带来的加/减锁操作影响,需要使每两个核心共享一条规则,这相对于改进前全局公用一条规则链而言,增加了算法的空间复杂度。以上两点的不足,也正是以后研究的一个重点。



**参考文献:**

[1] 林闯,王元卓,任丰原. 新一代网络 QoS 研究[J]. 计算机学报,2008,31(9):1525 - 1535.

[2] KUMAR S, TURNER J, WILLIAMS J. Advanced algorithms for fast and scalable deep packet inspection[C]// Proceedings of the 2006 ACM/IEEE Symposium on Architecture for Networking And Communications Systems. New York: ACM Press, 2006: 81 - 92.

[3] 曹折波,李青. 多核处理器并行编程模型的研究与设计[J]. 计算机工程与设计,2010,31(13):2999 - 3003.

[4] GUO DANHUA, LIAO GUANGDENG, BHUAN L N. A scalable multithreaded L7-Filter design for multi-core servers [C]// Proceedings of the 4th ACM/IEEE Symposium on Architecture for Networking and Communications Systems. New York: ACM Press, 2008: 60 - 68.

[5] Windows Hardware Development Center. Receive Side Scaling( RSS) [EB/OL]. [2011-10-20]. http://msdn.microsoft.com/en-us/windows/hardware/gg463253.aspx.

[6] 所光,杨学军. 多核处理机系统 Cache 管理技术研究现状[J]. 计算机工程与科学,2010,32(7):65 - 68.

[7] VALDVOGEL M. Multi-dimensional prefix matching using line search [C]// Proceedings of the 25th Annual IEEE Conference on Local Computer Networks. Washington, DC: IEEE Computer Society, 2000: 200 - 207.

[8] HAMED H, AL-SHAER E. On autonomic optimization of firewall policy organization [J]. Journal of High Speed Networks, 2006, 15(3): 209 - 227.

[9] 丁晶,陈晓岚,吴萍. 基于正则表达式的深度包检测算法[J]. 计算机应用,2007,27(9):2184 - 2193.

[10] MIT DARPA intrusion detection data sets[EB/OL]. [2010-10-10]. http://www.ll.mit.edu/IST/ideval/data/2000/2000_data_index.html.

[11] 杨赞,杨林,王宝林,等. 依据流统计特性的文分类规则动态优化[J]. 计算机应用研究,2011,28(5):1878 - 1882.

[12] (美)约翰逊,(美)威曾格,(美)普拉瓦提. Linux 服务器性能调整[M]. 韩智文,译. 北京:清华大学出版社,2004:23 - 24.

[13] (美) LOVE R. Linux 内核设计与实现[M]. 3 版. 陈莉君,康华,译. 北京:机械工业出版社,2011:143 - 148.

[14] Libnids[CP/OL]. [2010-10-10]. http://libnids.sourceforge.net/.

[15] 徐卫志,宋凤龙,刘志勇,等. 众核处理器片上同步机制和评估方法研究[J]. 计算机学报,2010,33(10):1777 - 1787.

(上接第 605 页)

[3] BAKER T P. An analysis of fixed-priority schedulability on a multiprocessor [J]. Real-Time Systems, 2006, 32(1/2): 49 - 71.

[4] BERTOGNA M . Real - time scheduling analysis for multiprocessor platforms [D]. Pisa: Scuola Superiore Sant'Anna, 2008.

[5] BERTOGNA M, CIRINEI M, LIPARI G. Schedulability analysis of global scheduling algorithms on multiprocessor platforms [J]. IEEE Transactions on Parallel and Distributed Systems, 2009, 20(4): 553 - 566.

[6] BARUAH S. Techniques for mutliprocessor global schedulability analysis [C]// Proceedings of the 28th IEEE International Real-Time Systems Symposium. Washington, DC: IEEE Computer Society, 2007: 119 - 128.

[7] BAKER T P, CIRINEI M. A necessary and sometimes sufficient condition for the feasibility of sets of sporadic hard-deadline tasks [C]// Proceedings of the 27th IEEE International Real-Time Systems Symposium. Piscataway, NJ: IEEE Press, 2006: 178 - 190.

[8] CIRINEI M, BAKER T P. EDZL scheduling analysis[J]. Real-Time Systems, 2008, 40(3): 264 - 289.

[9] BARUAH S, GOOSSENS J. Deadline monotonic scheduling on uniform multiprocessors [C]// Principles of Distributed Systems. Berlin: Springer-Verlag, 2008: 89 - 104.

[10] 石林勇,晏立. 多处理器全局单调比率的可调度性分析[J]. 计算机应用,2010,30(10):2735 - 2737.