一种多核系统可靠性加强的任务调度方法

徐 超[1,2,3],何炎祥[1,3],陈 勇[1,3],刘健博[1,3],吴 伟[1,3],李清安[1,3]
(1.武汉大学计算机学院,湖北武汉 430072;2.徐州工业职业技术学院,江苏徐州 221000;3.软件工程国家重点实验室,湖北武汉 430072)

摘 要: 多核系统已经被广泛应用于各行各业,其稳定性和可靠性也越来越受到人们的关注.在分析了现有芯片可靠性模型的基础上,增加温度和工作负载对多核芯片可靠性的影响,建立了对多核系统的可靠性评估模型.该模型以处理器为依托,从指令到任务,逐层构建可靠性评估指标,以便于定量分析影响多核系统可靠性的因素.同时,以该模型为指导,设计了一种面向多核系统可靠性的任务调度方法,该方法通过对评估指标值的计算,选择评估指标值尽可能高的调度策略对多核系统中的任务进行调度,以减少由于芯片本身可靠性而导致的错误.通过模拟实验可以看出,该任务调度算法能有效减少系统的出错率45%左右,为系统的稳定运行提供了可靠的保证.

关键词: 多核系统; 任务调度; 稳定性; 出错评估模型
中图分类号: TP309.1 **文献标识码:** A **文章编号:** 0372-2112 (2013)05-1019-06
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2013.05.031

A Task Scheduling Method to Increase the Reliability of the Multicore System

XU Chao[1,2,3], HE Yan-xiang[1,3], CHEN Yong[1,3], LIU Jian-bo[1,3], WU Wei[1,3], LI Qing-an[1,3]
(1. *Wuhan University Computer of School, Wuhan, Hubei* 430072, *China*; 2. *Xuzhou College of Industrial Technology, Xuzhou, Jiangsu* 221000, *China*; 3. *State Key Laboratory of Software Engineering, Wuhan, Hubei* 430072, *China*)

Abstract: The stability and reliability of multicore systems have been gotten more attentions with the widely used in all kinds of fields. After analyzing the current reliability model for chips, a reliability evaluating model considering the temperature and workloads for the multiprocessor is proposed. Based on construction characteristic of the multiprocessor, the model hierarchically constructs the evaluating index from instruction level to the task level that enables to quantitatively evaluate the reliability of the multiprocessor. To check the effectiveness of this model, a stability-oriented task scheduling algorithm is designed to enhance the stability of multicore system. It calculates the evaluating index for each scheduling scheme and chooses the scheduling scheme with highest evaluating index to obtain the low error rate caused by multicore. According to the results of simulation experiment, we can see that this model and method are effective that have the potential to reduce 45% of the error in multicore system.

Key words: multicore system; task scheduling; stability; error evaluating model

1 引言

随着信息时代的快速发展,多核嵌入式系统应用越来越广泛,从个人手持PDA到涉及国家安全的航天航空方面,多核嵌入式产品随处可见,多核嵌入式系统具有内核小、集成度高、可移植性、实时性强、速度快等特点.然而随着多核嵌入式产品的技术缩放,倾向于小的面积、集成度高的特性,那么低的阈值电压和严格的噪声容限,势必会增加高性能处理的软件错误[1].软件错误通常是指由于内部电流干扰或者外部辐射导致的暂时错误,这些错误通常使得某些bit位发生翻转而使得程序执行或者数据错误[2].由于硬件设备本身的缺陷,就无法保证软件执行时获得完全正确的结果,因此,需要相关措施以保证系统的正确运行[3].

在多核嵌入式系统中,核心部分是任务调度模块,任务调度主要目的是将任务合理分配到各个处理器上,使得任务完成时间最短.多核嵌入式系统的可靠性很大程度上取决于任务调度.虽然调度的主要目的是为了分配处理机,但随着任务分配方式的不同,系统可靠性也会有较大不同[4,5],因此,在多核系统中,如何使系统高效、可靠地运行与具体的任务调度方法有着紧密的联系[6,7].有效的任务调度方法将会大大提高多处理器系

收稿日期:2012-11-13;修回日期:2013-02-18
基金项目:国家自然科学基金重点项目(No.91118003);国家自然科学基金面上项目(No.61170022);江苏省高校"青蓝工程"优秀青年骨干教师培养对象资助;江苏省高校优秀中青年教师和校长境外研修项目资助

统的计算能力,降低不必要的能耗,提高系统的可靠性.

本文根据多核系统的特点,提出一种面向多核系统错误评估模型,同时,以该模型为指导,设计了一种面向多核系统可靠性的任务调度方法.以求减少软件系统的出错率,增加软件系统的可靠性.

2 相关研究

针对于嵌入式系统的可靠性研究,主要分为两个研究方向:(1)主要针对于寄存器文件和指令队列;(2)主要针对冗余数据. Jongeun Lee[8]等针对寄存器文件的脆弱性进行评估,设计了一种快速的,较为精确的,基于等式的寄存器文件脆弱性评估机制,以指导编译器优化生成高效的保护寄存器文件的程序. Jun Yan[9]等也针对寄存器文件导致软件出错的原因,提出了寄存器脆弱因子(RVF)的评估指标,然后根据该指标指导编译生成对应的较为可靠的软件程序. Xin Fu[10]等针对 Issue Queue(IQ,主要用于指令级和线程级的并行性),通过对指令脆弱性 profile 的分析,识别有关可靠性的关键指令,然后利用这些信息指导多线程的指令调度和资源分配,以减少并行程序中的错误 EDDI 通过在编译的过程中使用不同的寄存器和变量为重要指令生成对应的冗余指令,然后利用这些冗余指令检测程序中的错误. Vivek Sarkar[11]等针对嵌套循环的优化进行了研究,提出了一个更详细的成本模型和一个为展开嵌套循环生成紧凑代码的代码生成算法,解决了在嵌套循环中自动选择展开因素,并且为选择的展开因素生成了紧凑的代码. VK Sridharan[12]提出了一种系统脆弱堆栈框架,它根据系统体系结构的堆栈逐层计算对应的体系结构级脆弱因子,这些脆弱因子可以独立的或者联合起来一起对系统级的 AVF 进行评估. Ute Schiffel[13]等评估和比较了运行时开销及冗余和一些算术编码的错误检测能力,在运行成本和安全性之间进行了明确的权衡.算术代码的运行成本与冗余的增加只是线性关系,而获得的安全性则成指数的增加. Paul Lokuciejewski[14]等人针对实时系统提出了一种新的循环展开优化方法,利用最坏情况执行时间的信息有效的减少程序在最坏情况下的行为,利用最大优化的潜力,确定了一个合适的基于精确循环迭代计数的展开因子. Nahmsuk Oh[15]等提出了一个纯软件的技术-重复指令的错误检测(EDDI),检测常用系统操作过程中的错误,EDDI 在袁系统中添加错误检测能力不需要任何的硬件修改. Jie S. Hu[16]等通过对编译器导向的重复指令进行实验,检测出 VLIW 数据通路中的软错误,在这种方法中,编译器通过平衡允许的性能退化和所需的重复程度决定指令调度.

3 面向多核系统错误评估模型

由于硬件设备本身的缺陷问题,造成计算机无法保证软件在其执行时获得完全正确的结果,因此,需要相关措施以保证程序的正确运行.处理器作为计算机的核心部件,对软件的正确执行具有举足轻重的作用,因此,保证处理器的可靠性,对应软件的可靠性十分重要.为从软件的角度加强其可靠性,我们首先需要建立一个有效的评估模型,以指导对应软件优化方法的实施.

对于单核系统,文献[1]建立了基本的指令级错误评价模型,该模型假设错误均匀的分布在处理器资源的各个部分,因此评估硬件出错的可能性直接同面积成正比,即该处理器资源面积越大,其出错概率也就越大.然而在实际系统中,任务的错误指标除与设备本身相关外,同设备运行的环境(如温度、外部辐射等)也有很大的关系.一般情况下,温度越高,设备的不稳定性也就越高,错误的概率也随之增加,而为达到散热的目的,设备制造商通常通过增加该器件与周期器件的距离,从而也就增加了面积,而此时虽然该设备所占面积较大,但其错误概率较其紧凑布局(即小面积分布)时将显著减小.

基于上述问题分析,本文以一种温度为主要参数,将错误分布表示为温度和面积的函数,每条指令 i 执行错误指标描述为如下公式(1)形式:

$$\mathrm{IVI}_i = \frac{\sum_{c\in \mathrm{Proc}} \mathrm{IVI}_{ic} * f(A_c, T_c) * P_{\mathrm{fault}}(c)}{\sum_{c\in \mathrm{Proc}} f(A_c, T_c)} \quad (1)$$

其中,$P_{\mathrm{fault}}(c)$表示处理器资源 c 输出结果出错的概率,它需要综合考虑时序电路、组合电路以及掩码的作用,Proc 是处理器所有资源构成的集合,A_c 表示资源 c 所占的面积,T_c 表示资源 c 当时的温度,f 函数用于根据面积和温度计算其错误分布,采用如下归一化公式作为 f 函数的参考值:

$f(A,T) = A * (T - \mathrm{avg}(T))$,其中 $\mathrm{avg}(T)$表示该部件常温下工作时的平均温度.对于 IVI_{ic},其表示指令 i 在资源 c 中运行的出错指标,其值采用如下公式(2)进行计算.

$$\mathrm{IVI}_{ic} = \frac{Vu\ln \mathrm{erablePeiod}_{ic} * f(\mathrm{Bits}_{\mathrm{ACE}-c}, T_c)}{\sum_{c\in \mathrm{Proc}} f(\mathrm{TotalBits}_c, T_c)} \quad (2)$$

在以上公式的基础上,每个任务 t 的出错指标采用如公式(3)-(5)进行计算:

$$\mathrm{TPI}_{t(\mathrm{Failures})} = \sum_{i\in CI} \mathrm{IVI}_i * P_{T,\mathrm{Failures}}(\mathrm{FaultRate}) \quad (3)$$

$$\mathrm{TPI}_{t(\mathrm{IncorrOP})} = \sum_{i\in nCI} \mathrm{IVI}_i * P_{T,\mathrm{IncorrOP}\text{-}nCI}(\mathrm{FaultRate}) + \sum_{i\in CI} \mathrm{IVI}_i * P_{T,\mathrm{IncorrOP}\text{-}CI}(\mathrm{FaultRate}) \quad (4)$$

$$\mathrm{TPI}_t = \frac{\alpha * \mathrm{TPI}_{t(\mathrm{Failures})} + \beta * \mathrm{TPI}_{t(\mathrm{IncorrOP})}}{I_t} \quad (5)$$

由于考虑到温度因素,因此,$P_{T,\mathrm{Failures}}(\mathrm{FaultRate})$,$P_{T,\mathrm{IncorrOP}-nCI}(\mathrm{FaultRate})$,$P_{T,\mathrm{IncorrOP}-CI}(\mathrm{FaultRate})$分别表示温度为 T 时,关键指令导致的严重错误,非关键指令导致的输出错误,关键指令导致的输出错误的概率. I_t 表示任务 t 中的指令数,$\mathrm{TPI}_{t(\mathrm{Faukures})}$用于评估任务 t 运行错误的概率,$\mathrm{TPI}_{t(\mathrm{IncorrOP})}$用于评估输出结果出错的概率. α 和 β 分别表示程序出错以及错误输出对整个任务的影响因子.

在多核系统中,根据每个核负载的不同,其出错概率也不相同.一般来说,负载高的核其出错的概率高于负载低的核.因此,我们可以在单核系统任务出错指标的基础上,构建如公式(6)的评估模型:

$$TI_{i,j} = \sum_{k \in C_j} \mathrm{TPI}_i * (\lambda m_k) \quad (6)$$

其中:TI_{i,C_j}表示任务 i 在处理器 j 中运行的出错指标,C_j 表示处理器 j 中所有资源的集合,m_k 表示处理器资源 k 中运行的任务数,$\mathrm{TPI}_{i,k}$表示任务 i 在处理器资源 k 中的出错指标,λ 是任务权重参数,用于权衡核中任务数对系统稳定性的影响.

4　面向系统可靠性的任务调度方法

为了能够对任务队列进行较好的调度,使得系统能够更稳定的运行,本文在以上任务出错指标的指导下,设计了如图 1 所示的调度方案.

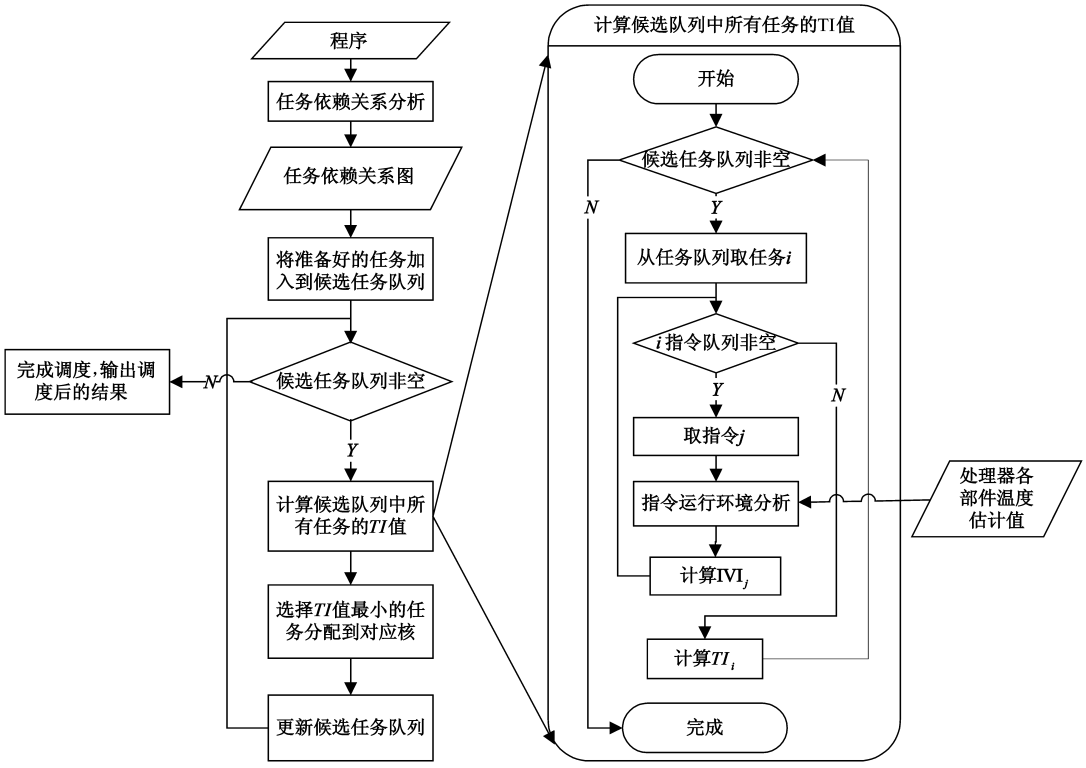

图1　面向系统可靠性的任务调度方法

在该方案中,首先对程序进行任务依赖关系分析,获得各任务依赖关系图.任务依赖关系图用于描述各任务执行的先后顺序关系,它的每个节点表示一个任务,每条有向边($u \rightarrow v$)表示任务 u 必须先于任务 v 执行.在获得了任务依赖关系图之后,将那些没有前驱任务,且其运行所需的各种处理器资源都准备好的任务加入候选任务队列,以供调度.在进行任务调度时,我们依次从候选任务队列中选择任务,计算其在每个核执行时的 TI 值,选择其中 TI 值最小的任务分配到对应的核中执行.然后更新候选任务队列,重复以上过程进行调度,直至将所有任务都分配到对应核中.

在该方法中,为有效的计算 TI 值,获得处理器核中各个部件较为准确的温度信息,我们采用静态 profile 的方法,通过将各任务置于单个处理器运行时的温度升高值作为温度增量,然后加上初始温度值,形成最终温度值加以计算,从而保证该调度方案能有效执行(如算法 1 所示).

算法 1

1. **Input:** 被调度的任务序列的集合 $P = \{p_i \mid i = 1,2,\cdots,n\}$,处理器数目 k
2. **Output:** 调度好的任务序偶的集合 $Q = \{(p_i, c_i) \mid i$

$=1,2,\cdots,n\}$

```
3.  Begin:
4.      PG(V,E):=ConstructDFG(P)
5.      for each n∈V
6.        if pre(n) is null
7.          add k to candidate list Cnd
8.        endif
9.      endfor
10.     while Cnd! =ϕ
11.       TI_min = +∞
12.       for each m∈Cnd
13.         for each i<k
14.           TI:=CaculateTI(m,i)
15.           if TI<TI_min
16.             TI_min=TI, <p,c>:=<m,i>
17.           endif
18.         endfor
19.       endfor
20.       add <p,c> to Q
21.       remove p from Cnd
22.       update PG(V,E) and Cnd
23.     endwhile
24.     return Q
25.     end
```

在该算法中,首先根据任务调度序列以及程序的数据依赖关系,构建任务依赖关系图 $PG(V,E)$(第 4 行);然后将根据任务依赖关系图,选择准备好的任务(即该任务 n 的依赖的任务 pre(n)为空)加入到候选队列 Cnd(第 5~9 行);接着,依次遍历候选队列中的任务序列,通过对 TI 值的计算,每次选择候选任务列表中 TI 值最小的<任务,处理器编号>加入到最终的调度序偶 Q 中,并更新任务依赖关系图 $PG(V,E)$和 Cnd(第 10~23 行);最后返回调度好的任务调度序偶 Q.由于该方法是静态调度范围,因此该调度算法本身不会增加系统运行时开销.

5 模拟实验及结果分析

(1)**检测该方法的有效性**

我们构建了如表 1 所示的实验环境,其中错误分布函数主要是以平均值函数为基础,当该部件的面积及温度的总值大于该处理器面积及温度的平均值时,则其分布函数值为 1,应该添加错误,否则为 0,表示不添加错误.具体实验过程中,依次调度各任务到核内运行,指导发现第一个错误为止,根据公式(7)计算出错率.其中 t_{err} 表示发现第一次出错的时间,t_s 表示无错误运行时需要的总时间.

$$Err = 1 - \frac{t_{err}}{t_s} \tag{7}$$

表 1 实验环境描述信息

模拟器使用操作系统	Federal Linux 12
模拟核数目	4
错误分布函数	$P(A_c,T_c)=\begin{cases}1, & A_c/\sum_{k\in proc}A_k + T_c/\sum_{t\in detT}t_c > avg(A)+avg(T)\\ 0, & \text{otherwise}\end{cases}$

(2)**测试用例集**

测试用例选取了通用嵌入式测试用例集 Mibench 中的三个用例:"quick_sort","fft"以及"basicmath",将其转换为并行程序并构建相应的任务调度序列图,使其能够方便的被模拟程序调度,修改后的程序其任务数如表 2 所示.

表 2 修改后的测试用例集

测试用例	分解的任务数
quick_sort	29
fft	41
basicmath	37

(3)**具体实验主要分为三个步骤**

(a)**注入错误** 采用随机函数,在任务运行的时间内,根据错误概率函数 $f(A,T)$对每个处理器插入错误,即当 $f(A,T)$为 1 时,通过如果随机值(在[0,1]的区间内)大于 0.5,则添加错误,否则不添加错误.

(b)**错误检测** 根据每个核的运行结果与预期结果的对比,判断该错误是否导致整个程序出错.

(c)**实验结果统计** 根据错误检测结果,评估该方法的有效性.主要通过同负载均衡调度算法[17]以及性能优先调度算法[18]在不同外界温度情况下程序出错率,在相同外界温度下,各种算法由于核内温度,核内负载不同而产生错误数.不同外界温度下,该调度算法导致的系统出错数如图 2 所示.

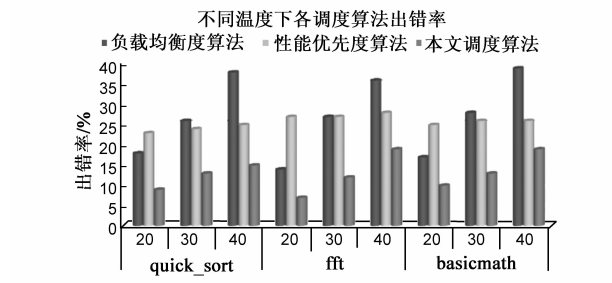

图2 不同环境温度下各调度算法的出错数分布图

根据以上实验结果可以看出,在外界温度变化时,各种算法程序出错的概率各不相同,但无论是哪种算

法,随着外界温度的增加,程序出错的概率也有不同程度的增多.由此可见,温度对于程序出错的概率有较大程度的影响.为更好的验证其影响的程度以及各种算法下系统的稳定程度,我们在外界温度为 30 时,对各种任务调度算法进行了比较,其实验结果如图 3 ~ 5 所示.

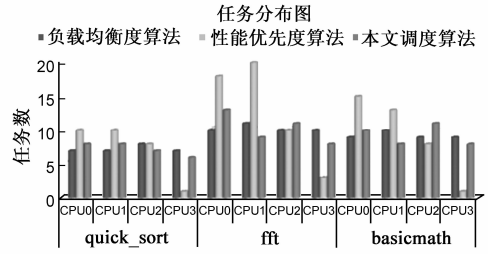

图3　各调度算法下各CPU核的任务分布图

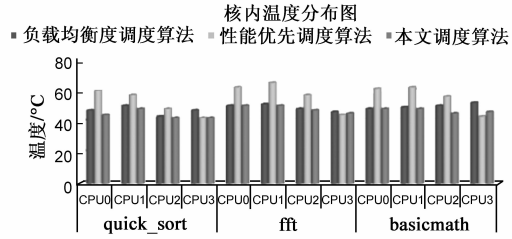

图4　各调度算法下各CPU核的核内温度分布图

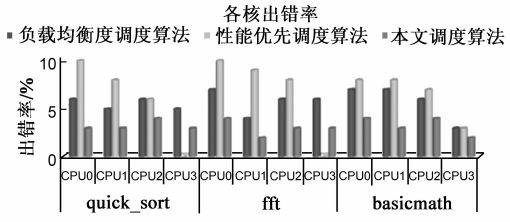

图5　各调度算法下各CPU核的错误分布图

图 3 显示了各种任务调度算法下任务在各个核的分布情况,图 4 显示了各个核内对应的平均温度.由这两个图可以看出,在负载均衡调度算法下,各个核的任务数基本相同,核内温度较低;而在性能优先调度算法下,由于其重点在于提高程序执行的速度,因而为减少核间通信,会出现某些核负载很高的情况,同时核内的平均温度也较高.而本文的调度算法各核之间的任务数与负载均衡调度算法基本相同,可以有效的防止高负载情况下的过高核内温度,以降低高温而引起的程序错误.

图 5 显示了各种调度算法下程序的出错率.由该图可以看出,本文使用的调度算法相对于性能优先算法,其平均出错率可减少 50% 左右,相对于负载均衡调度算法,其平均出错率也可减少 45% 左右,因此,该调度算法能有效减少各个处理器中的实际错误,加强了系统的可靠性.

6　总结

随着多核系统的广泛应用,特别是在各种移动嵌入式设备中的不断普及,其可靠性已经成为人们研究的热点.本文通过对多核处理器中有关系统可靠性关键因素(如温度、面积等)的分析,从指令级到任务逐层建立对应的出错评估指标,同时以该评估指标为基准,设计了对应的任务调度方法,以减少多核系统的出错概率,提高系统的稳定性.通过模拟实验的结果可以看出,相对于传统的任务调度方法,该方法能够减少系统出错率高达 45% 以上,为系统的稳定运行提供了可靠的保证.

此外,由于设备的硬件结构各具特点,不同材质的设备在不同温度下出错的概率各不相同,而本文只是从设备的平均温度出发,构建的评估模型,在某些特殊材质中也许有失精准,因此,在今后的工作中,我们拟根据各种处理器本身对温度的敏感程度动态调整 $f(A, T)$ 函数,进一步提高评估模型的准确性,以便更好的指导任务调度方法,提高系统的可靠性.

参考文献

[1] S Rehman, M Shafique, F Kriebel, J Henkel. Reliable software for unreliable hardware: embedded code generation aiming at reliability[A]. In Proceedings of the seventh IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis[C]. Taipei, Taiwan: ACM, 2011. 237 – 246.

[2] 徐超,何炎祥,吴伟,等.基于模拟关系的编译优化实现正确性验证方法[J].电子学报,2012,40(11):2171 – 2176.
XU Chao, HE Yan-xiang, WU Wei. Verifying implementation correctness of compiling optimization based on simulation relation[J]. Acta Electronica Sinica, 2012, 40(11): 2171 – 2176. (in Chinese)

[3] Maheswaran M, H J Siegel. ADynamic matching and scheduling algorithm for heterogeneous computing systems[A]. Heterogeneous Computing Workshop[C]. Orlando, USA: IEEE Computer Society, 1998. 57 – 69.

[4] A J Page, T J Naughton. Dynamic task scheduling using genetic algorithms for heterogeneous distributed computing[A]. International Parallel and Distributed Processing Symposium[C]. Denver, USA: IEEE Computer Society, 2005. 189 – 197.

[5] Y H Yang, S S Yu, X L Bin. A new dynamic scheduling algorithm for real-time heterogeneous multiprocessor systems[A]. Proceedings of the Workshop on Intelligent Information Technology Application[C]. Washington, USA: IEEE Computer Society, 2007. 112 – 115.

[6] Nacul A, Regazzoni F, Lajolo M. Hardware scheduling support in SMP architectures[A]. Proceedings of the design automation

and test in Europe conference[C]. Nice, France: ACM, 2007. 1 – 6.

[7] 江维,常政威,桑楠,等.安全和能量关键的分布式协作任务调度[J].电子学报,2011,39(4):757 – 762.
JIANG Wei, CHANG Zheng-wei, SANG Nan. Scheduling for security and energy-critical distributed collaborative tasks[J]. Acta Electronica Sinica, 2011, 39(4): 757 – 762. (in Chinese)

[8] J Lee, et al. Compiler approach for reducing soft errors inregister file[A]. Conference on Languages, Compilers, and Tools for Embedded Systems[C]. Dublin, Ireland: ACM, 2009. 41 – 49.

[9] J Yan, et al. Compiler guided register reliability improvement against soft errors[A]. International Conference on Embedded Software[C]. Jersey City, New Jersey, USA: IEEE Computer Society, 2005. 203 – 209.

[10] X Fu, W Zhang, T Li, J Fortes. Optimizing issue queue reliability to soft errors on simultaneous multithreaded architectures [A]. International Conference on Parallel Processing[C]. Portland, Oregon, USA: IEEE Computer Society, 2008. 190 – 197.

[11] V Sarkar. Optimized unrolling of nested loops[J]. International Journal on Parallel Programing, 2001, 29(5): 545 – 581.

[12] V Sridharan. Introducing abstraction to vulnerability analysis [D]. Ph. D. Thesis, 2010. 45 – 55.

[13] U Schiffel, et al. Software-implemented hardware error detection: costs and gains[A]. The Third International Conference on Dependability[C]. Venice, Italy: IEEE Computer Society, 2010. 51 – 57.

[14] P Lokuciejewski, et al. Combining worst-case timing models, loop unrolling, and static loop analysis for WCET minimization[A]. Euromicro Conference on Real-Time Systems (ECTRS 09)[C]. Dublin, Ireland: IEEE Computer Society, 2009. 35 – 44.

[15] N Oh, et al. Error detection by duplicated instructions in superscalar processors [J]. IEEE Transaction on Reliability, 2002, 51(1): 63 – 75.

[16] J S Hu, et al. compiler-directed instruction duplication for soft error detection[J]. DATE, 2005, 1(2): 1056 – 1057.

[17] T Li, et al. Efficient and scalable multiprocessor fair scheduling using distributed weighted round-robin[A]. Principles and Practice of Parallel Programming[C]. Raleigh, North Carolina, USA: 2009. 65 – 74.

[18] A Fedorova, et al. Improving Performance Isolation on Chip Multiprocessors via an Operating System Scheduler[A]. Parallel Architectures and Compilation Techniques[C]. Brasov, Romania: IEEE Computer Society, 2007. 25 – 38.

作者简介

徐 超 男,1980 年出生,湖北红安人,博士生,讲师,研究方向为可信软件和嵌入式系统. E-mail: xuch@whu.edu.cn

何炎祥(通讯作者) 男,1952 年出生,湖北应城人,博士,教授,博士生导师,研究方向为可信软件、并行分布处理、软件工程和嵌入式系统. E-mail: yxhe@whu.edu.cn