Journal of System Simulation

Volume 31 | Issue 12 Article 16

12-13-2019

Fault-tolerant Method and Simulation of Heterogeneous Multi-core Processor Based on Speculative Mechanism

Shigan Yu
1. State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China; ;2. School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing 100049, China; ;3.National Engineering Laboratory for Advanced Processor Technology, Chengdu 610218, China; ;4.Information Engineering college, Fuyang Normal University, Fuyang 236041, China;

Zhimin Tang
1. State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China; ;2. School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing 100049, China; ;3.National Engineering Laboratory for Advanced Processor Technology, Chengdu 610218, China; ;

Xiaochun Ye
1. State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China; ;2. School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing 100049, China; ;

Dongrui Fan
1. State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China; ;2. School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing 100049, China; ;

Fault-tolerant Method and Simulation of Heterogeneous Multi-core Processor Based on Speculative Mechanism

Abstract

Abstract: Heterogeneous multicore is one of the important branches of processors,but they are still faced with frequent transient failures. TMR(Triple mode redundancy) is the main method to solve transient faults, which has the characteristics of low efficiency and high power consumption, *a high-performance Fault-Tolerant Scheduling Algorithm with Speculative mechanism(FTSAS) is proposed. Each heterogeneous core can execute tasks independently, the state values of the first completed core are recorded, and the first completed core continues to perform the next task with forward speculative method. The results are compared by backward core, the majority consensus principle is adopted to ensure the reliability of the system.* The simulation results show the average performance of the FTSAS is improved by 12.9% compared with state-of-the-art methods. When 200 errors are injected, the FTSAS has a similar fault-tolerant effect, however, FTSAS can achieve 11.4% average efficiency improvement and 15.8% average power consumption decrement.

Keywords

Heterogeneous multicore, Processor, Speculative mechanism, fault tolerance, Scheduling

第 31 卷第 12 期 系统仿真学报© Vol. 31 No. 12
2019 年 12 月 Journal of System Simulation Dec., 2019

基于推测机制异构多核处理器容错方法与仿真

余世干[1,2,3,4]，唐志敏[1,2,3]，叶笑春[1,2]，范东睿[1,2]

（1. 中国科学院计算技术研究所计算机体系结构国家重点实验室，北京 100190; 2. 中国科学院大学计算机控制与工程学院，北京 100049; 3. 先进微处理器技术国家工程实验室，四川 成都 610218; 4. 阜阳师范大学信息工程学院，安徽 阜阳 236041）

摘要：异构多核是处理器重要方向之一，却面临着瞬态故障频发问题，传统 TMR(三模冗余)是主要解决办法，但有效率低，功耗高特点，*提出基于推测机制高性能容错调度算法 FTSAS。各异构核独立执行任务，记录最先完成核的状态值，采用前向推测法继续执行下一任务，采用多数一致原则，由落后的核完成结果比较，保障系统可靠性。*仿真实验表明，FTSAS 比当前容错方法平均性能提高了 12.9%，注入 200 个错误时，具有相近的容错效果，但 FTSAS 平均执行性能提高了 11.4%，平均功耗降低了 15.8%。

关键词：异构多核；处理器；推测机制；容错；调度

中图分类号：TP391.7 文献标识码：A 文章编号：1004-731X (2019) 12-2685-11

DOI: 10.16182/j.issn1004731x.joss.19-FZ0346

Fault-tolerant Method and Simulation of Heterogeneous Multi-core Processor Based on Speculative Mechanism

Yu Shigan[1,2,3,4], *Tang Zhimin*[1,2,3], *Ye Xiaochun*[1,2], *Fan Dongrui*[1,2]

(1. State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China; 2. School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing 100049, China; 3.National Engineering Laboratory for Advanced Processor Technology, Chengdu 610218, China; 4.Information Engineering college, Fuyang Normal University, Fuyang 236041, China)

Abstract: Heterogeneous multicore is one of the important branches of processors，but they are still faced with frequent transient failures. TMR(Triple mode redundancy) is the main method to solve transient faults, which has the characteristics of low efficiency and high power consumption, *a high-performance Fault-Tolerant Scheduling Algorithm with Speculative mechanism(FTSAS) is proposed. Each heterogeneous core can execute tasks independently, the state values of the first completed core are recorded, and the first completed core continues to perform the next task with forward speculative method. The results are compared by backward core, the majority consensus principle is adopted to ensure the reliability of the system.* The simulation results show the average performance of the FTSAS is improved by 12.9% compared with state-of-the-art methods. When 200 errors are injected, the FTSAS has a similar fault-tolerant effect, however, FTSAS can achieve 11.4% average efficiency improvement and 15.8% average power consumption decrement.

Keywords: Heterogeneous multicore; Processor; Speculative mechanism; fault tolerance; Scheduling

引言

高性能处理器过去依赖于单片单核的高性能高集成度晶体管，现已转向单片多核，高集成度的晶体管数量呈指数级的增加[1]，片上多核已经成为处理器的主流。一直以来，同构多核得到了广泛的应用，但在实际中同构大核会导致低优先级低复杂度线程执行效率降低，同构小核会引起单线程执行的吞吐量的减小[2]，因此同构大核或小核都具有不同方面的缺点[3]。异构多核是由不同性能的单核构

收稿日期：2019-04-25 修回日期：2019-07-19;
基金项目：国家重点研发计划(2018YFB1003500), 国家自然科学基金(61732018, 61872335, 61802367), 安徽高校人才支持与科研项目(gxyq2019175, KJ2018A0669);
作者简介：余世干(1982-)，男，安徽定远，硕士，副教授，研究方向为计算机体系结构与容错计算。

成，根据不同核的特点，可以针对不同任务进行动态调整，能够进一步提高系统性能并降低功耗，因此逐步得到了应用推广，例如 ARM 的 big.LITTLE[4]，NVidia 的 Tegra[5]，Intel 的 QuickIA[6]，华为海思 970、980[7]等都是异构多核的产品。

另一方面，随着电路密度的提高，晶体管栅极临界电荷在减小，这增加了带电元件辐射翻转的概率[8]，另外温度偏差不稳定 BTI 和热载流子注入 HCI 等集成电路老化现象[9]，均将诱发处理器的瞬态故障的增加，而瞬时故障是引发处理器 80%以上失效的主要原因[10]。因此无论同构多核还是异构多核处理器均需要提高可靠性，使系统具有容错能力，尤其对于那些处于重要节点上的关键任务，即使执行中出错，也要保证系统的结果在期望范围内。相对于同构多核可靠性研究，异构多核可靠性的研究工作相对较少[8]，尤其面对错综复杂工作环境时，系统故障发生率相对较高。在保障系统可靠性的同时，仍然需要提高任务执行性能、降低系统功耗，本文就是在此背景下对可靠性要求较高的异构多核系统开展容错调度方法的研究工作。

1 相关工作

为了提高系统可靠性，容错技术在计算机发展过程中一直受到不同程度和形式的重视，传统的容错技术有 DMR(Dual Mode Redundancy)[11]，TMR(Three Mode Redundancy)[12]等多模冗余方法；多版本技术[13]；LOCKSTEP 容错[14]；主副版本 PB(Primary & Backup version)容错[15]；同时多线程 SMT 容错[16]，检查点技术容错[17]以及各类冗余容错校验编码等方法。

在这些容错方法中，TMR 的使用时间较长，范围较广，具有检错和容错效果，一次运行或一个阶段能纠正一个错误，是解决处理器瞬态故障的主要方法，DMR 只能检错而无纠错能力。多版本技术采用软件的某一部分的多个版本使用，利用顺序或并发的运行实现容错。LOCKSTEP 技术主要用冗余的硬件重复执行指令，在同一段时间内重复执行需要容错的指令。PB 方法把待处理的任务主版本和副版本分别调整到不同的处理单元，当主版本任务发生错误时，启用副版本的结果作为最终输出，有主动、被动、重叠 3 种工作方式，在执行时，先对任务进行划分，同时需要增加检测模块实现对主模块结果的正确性判断，增加了主副版本任务的调度开销。SMT 容错技术通过同时处理多个线程进行比较从而实现容错的片上多处理器技术[18]。检查点技术在不同执行阶段，保存系统状态为检查点，当执行中发生错误时，卷回到前一个正确的检查点状态重新恢复执行，主要用于对 fail-stop 类型故障[19]进行容错处理。冗余校验位主要用于存储数据检错和纠错，容错应用的领域受到限制。

推测技术在实现串行指令并行性，非规则程序并行性方面提高处理器执行性能起到了重要作用，是处理器微结构执行的一种重要技术。另外在大数据性能优化[20]，航空轨迹判断[21]等多方面也有着广泛应用，但是在容错领域还较少涉及，而 TMR 是处理瞬态故障的重要方法，但是存在低效率、高功耗的特点，本文即是利用推测技术，结合核的异构性对目前常用的 TMR 方法进行优化研究，在保障系统容错功能的同时提高性能、降低功耗。

2 目前的 TMR 容错机制

现有的 TMR 容错结构如图 1 所示，有 U_1，U_2，U_3 三个模块和表决器 Voter 组成。

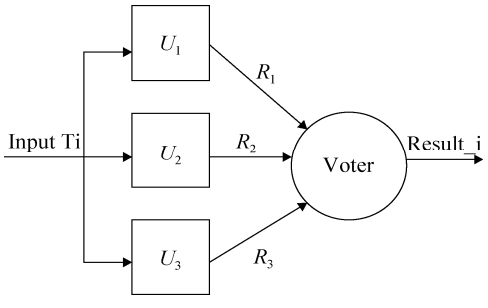


图 1 TMR 基本结构图
Fig. 1 Basic architecture diagram of TMR

任务 Ti 同时输入至 U_1，U_2，U_3，三模块执行完成后分别把各自结果 R_i 传送至 Voter，Voter 完全

DOI: 10.16182/j.issn1004731x.joss.19-FZ0346

接收 3 个模块后，采取多数一致原则进行表决输出。当 R_i 完全相同或者两者一致时，则输出正确结果，在系统执行中即使有一个模块发生错误，系统输出也是可靠的。由于同一时刻或者任务周期内，两个模块同时出现错误是小概率事件，所以三模冗余假设在每一次表决器裁决结果时，最多只有一个模块发生错误，因此 TMR 对任何一个模块出错都有容错能力。

这种同构 TMR 执行机制主要采用空间冗余方式提高处理器系统可靠性，对于每一种任务都是在 3 个同构模块系统执行 3 次，体现出系统执行效率低下、功耗较高特点，没有充分利用任务多样性，而在异构系统中，不同核具有不同的属性，面对不同任务具有不同的执行效率，为了充分结合每个核的特点，本文提出一种面向异构多核的基于推测机制的高性能容错调度方法。

3 系统模型

3.1 系统定义

定义1 可靠性 $R(t)$: 指一个完整系统在正常运行状态下，在时间[0,t]内，达到预期目标的概率。

定义 2 不可靠率 $\lambda(t)$: 表示在某个时间范围[0,t]内，不能正常工作与系统总模块数目之间的比值，对一个模块来说，$\lambda(t)$表示在时间[0,t]内出现错误的概率。根据实际统计可得模块不可靠率 $\lambda(t)$与时间 t 的关系如图 2 所示[22]。

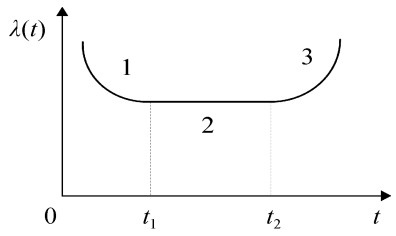


图 2 模块失效率与时间的关系
Fig. 2 Relationship between failure rate and time of module

在图 2 中，使用前应该先对执行模块进行测试，尽可能使其处于图 2 中第二段的稳定而高可靠状态，这能提高系统的可靠性，降低不可控因素，在这一阶段，不可靠率近似为一个常数，可设 $\lambda(t)=\lambda$，则 λ 值通常为系统平均发生故障时间的倒数。根据系统不可靠率定义[23]，可靠度如公式(1)所示。

$$R(t)=e^{-\lambda t} \tag{1}$$

公式(1)表明系统模块可靠度与不可靠率成负指数关系，可以据此来判断系统的可靠性。

定义 3 在异构多核系统中，通常把需要计算的任务定义为有向无环图 DAG 的任务模型[23]。本文定义 DAG 任务模型为 $\Gamma=(M,V,E,T,W)$。其中 $M=\{m_0,m_1,m_2,\ldots,m_n\}$表示处理器集合，$m_i$ 表示整个系统中第 i 个处理单元。$V=\{v_0,v_1,\ldots,v_n\}$是任务的集合，每个 v_i 代表一个需要被处理的任务。$E=\{e_{i,j}\}$是 DAG 中两个节点之间的边，它的值代表两个任务之间的通讯时间，在系统分析设计阶段，这是已知的。$T=\{t_0,\ t_1,\ldots,t_n\}$ 代表每个任务节点可靠性阈值，$W=\{w_{j,k}\}$代表第 j 个任务在第 k 个核上的运行时间，由于异构多核的特点，在不同核上，$w_{j,k}$ 值是不同的。系统 DAG 化的任务流如图 3 所示。

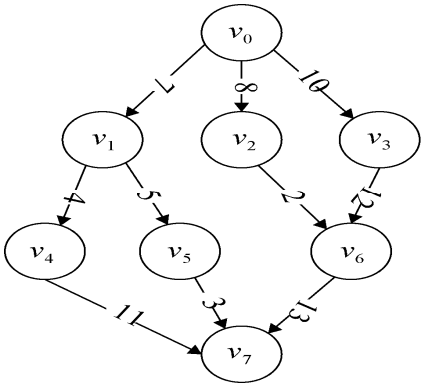


图 3 系统 DAG 任务模型
Fig. 3 DAG task model of the system

系统执行任务时需对 DAG 任务节点进行排序，采用降序排列法[23-25]，计算方法如公式(2)~(3)所示。

$$sort(v_{exit})=c_{exit} \tag{2}$$

$$sort(v_i)=c_i+\max\{c_{i,j}+sort(v_j)\} \tag{3}$$

式中：c_{exit}，c_i 分别为尾节点任务、第 i 个任务节点任务的处理时间，任务 v_j 是任务 v_i 的后继任务。

3.2 系统可靠性

λm_i 代表第 i 个处理器核的失效率，第 i 个任务在第 j 个处理器上执行的可靠性 $R(v_i,m_j)$可根据公式(4)计算得出，其中 k 表示执行任务 v_i 的处理器核的序号。当第 i 个任务被执行时，整个系统可靠性可由公式(5)计算得出。如果高于可靠性阈值 t_i，则满足要求，否则需要对系统执行过程进行检查。

$$R(v_i,m_j)=\exp(-\lambda_{m_i}\cdot m_{i,k}) \tag{4}$$

$$R(\Gamma_i)=(\prod_{pre(i)\in\Gamma}R(v_i,m_k))\cdot R(v_i,m_j) \tag{5}$$

4 系统实现

4.1 系统假设

本系统工作时首先定义合理性前提，假设在每一次任务被执行时，3 个核最多只容许发生一个错误，否则需要增加冗余单元，核间可以相互通信、发送和接受执行状态。整个系统若多于 3 个核，可以选用其中 3 个核处于本文的工作状态之中。

4.2 总体思路

由于推测技术在串行任务并行化从而在提高系统执行效率方面具有重要作用，目前 TMR 解决处理器瞬态故障时，效率低下和功耗较高主要由于 3 个核始终同时工作，在每个核都执行完毕后，由表决器进行表决，然后进行下一任务执行，在异构多核中，由于每个核的结构和性能有所区别，如果在执行 TMR 时，执行任务速度较快的核执行完成后会停留等待落后的核，直到落后的核执行完成再进行表决，这不能利用异构多核的特点。为充分发挥不同核的异构性优势，在保障系统可靠性基础上，提出了本文的容错调度方法。

首先把待运行任务按照 DAG 模型划分为一个有序任务序列 $v_1,v_2,\ldots,v_n$，按照公式(2)~(3)确定优先级顺序。任务启动时，赋予 3 个核同样的任务 v_i $(0\le i\le n)$，同时开始执行，v_i 在不同核中的执行速度有所区别，记执行最快的核为 C_1，在最先执行完成时，暂时保存结果为 R_1，然后采用前向推测执行方法，继续执行下一个任务 v_{i+1}，记第二完成的核为 C_2，保存执行结果为 R_2。如果 R_1 与 R_2 相等，则立即终止执行最慢的核 C_3，使其同步到 R_2 状态，同时开始执行 v_{i+1}；如果 R_1 与 R_2 不相等，则需等到 C_3 执行完成，保存结果记为 R_3，若 R_3 与 R_1 相等，则提交 R_1 的值，若 R_3 与 R_2 相等，则提交 R_2 的值，若 R_1，R_2，R_3 互不相等，则撤回 C_1 执行状态，使 3 个核 C_1，C_2，C_3 退回至 v_i 任务并重新同时执行(这是极小概率事件)，若再一次执行，仍得不到正确结果，则说明处理器硬件发生故障，需要对硬件进行检修或者更换新的执行模块，按此工作方式直至所有任务执行结束。推测执行流程如图 4 所示，具体实现过程如算法 1 所示。

算法 1：Fault-Tolerant Scheduling Algorithm with Speculative method (FTSAS)

输入: 任务流 v_i

输出: 每一个任务的执行结果 Result_i

1. 初始化每个任务和各核模块的可靠度，并按公式(2)~(3)排序;

2. 按公式(1)检查每个核的可靠度，若达不到预期，则更换新核重新检测直到达到可靠度要求;

3. 设置若干任务固定保存时间点 $P_j(1\le j\le m)$;

4. 任务 v_i 赋给每个核，再同时执行;

5. 记录最先执行到任务保存点的核为 C_1，保存执行结果为 R_1，然后赋予下一任务 v_{i+1}，让其不等待，继续向前推测执行;

6. 记录第二个执行到任务保存点 P_j 的核为 C_2，保存当前执行结果为 R_2，然后立即赋予下一任务 v_{i+1}，让其不做等待，继续执行;

7. 如果 R_1 与 R_2 相等，则直接终止落后第三个核 C_3 的执行状态，同步 C_3 到与 C_2 状态，直接从 P_j 点开始执行下一任务 v_{i+1};

8. 如果 R_1 与 R_2 不相等，再继续让 C_3 执行到 P_j 点，然后保存其执行结果为 R_3：

8.1. 如果 R_1 与 R_3 相等，则撤回 C_2 至当前 P_j 点，并同步到 C_3 状态，并同时执行下一任务 v_{i+1};

8.2. 如果 R_2 与 R_3 相等，则撤回 C_1 至当前 P_j 点，并同步到 C_3 状态，并同时执行下一任务 v_{i+1};

9. 按公式(5)检查执行到当前节点时系统可靠性，若达到要求，则输出结果 Result_i，否则跳至第 4 步重新执行；

10. 若任务流中仍然存在任务没有执行完成，则跳至第 4 步继续执行；否则，执行结束。

在算法 1 中，执行到第 7 步时，采用的办法是先判断相对较快的 C_1 和 C_2 两个核执行结果 R_1 与 R_2 是否相等，若相等直接跳过第 8 步。这是考虑到对于系统正常执行任务时，在大概率情况下 R_1 与 R_2 是相等的，可以直接终止 C_3 执行过程和跳过进一步相互比较的过程，这能保证系统可靠性的同时具有较高的执行性能。

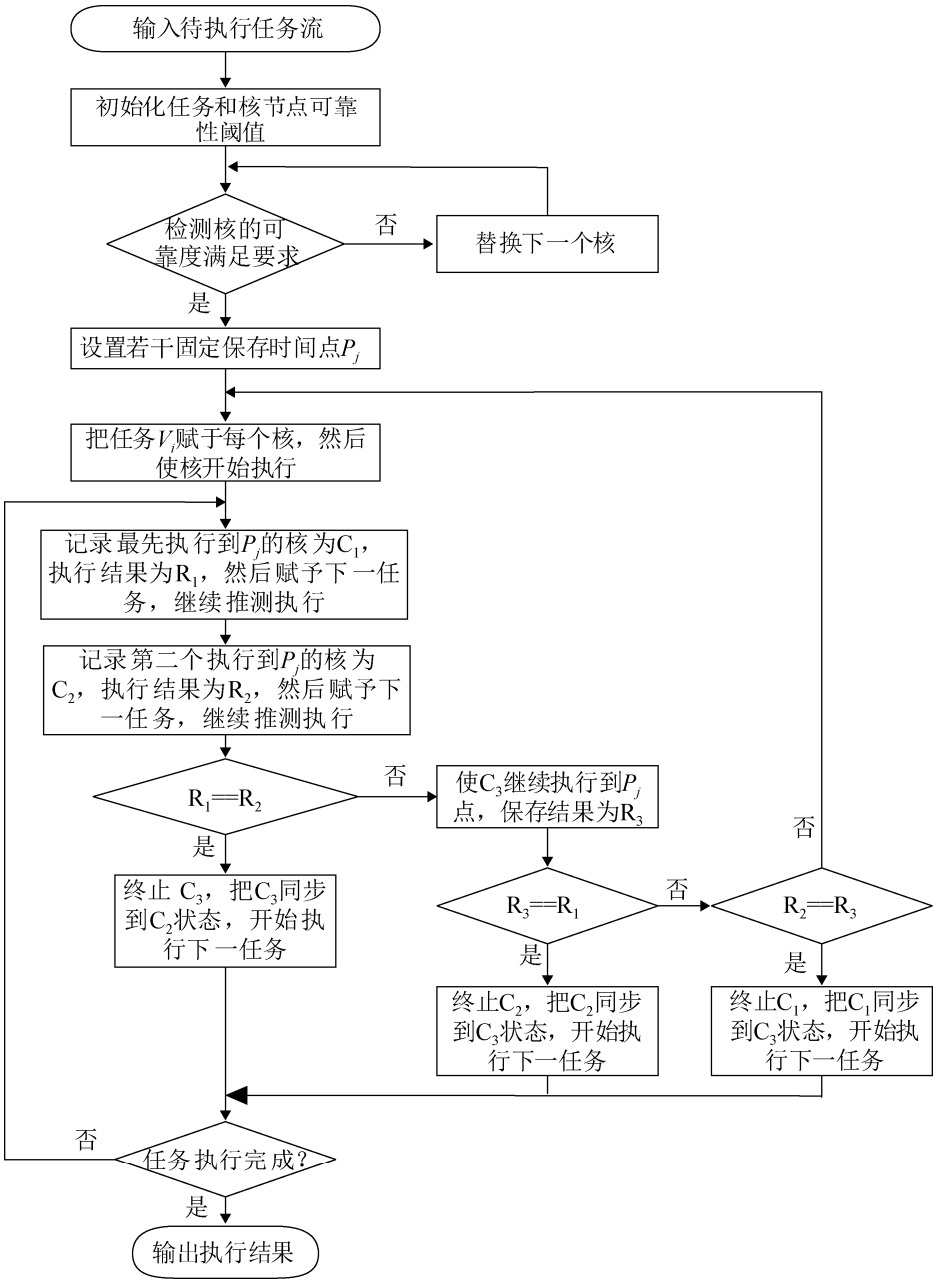


图 4 算法执行流程图
Fig. 4 Algorithm execution flow

5 系统可靠性分析

本文假设每个模块具有相近的不可靠率 λ 和可靠度 R_0，则 R_0 与 λ 之间关系由公式 4 计算可得，系统可靠工作时间为 M，根据定义，M 可由 R_0 在有效工作时间内的积分计算得出[26]。

5.1 传统的 TMR 可靠性分析

现有 TMR 系统正常工作或者某个模块失效，系统都是可靠的，可靠度为 $R_{TMR(t)}$与平均可靠运行时间为 M_{TMR}，可由公式(6)~(7)计算得出。

$$R_{\mathrm{TMR}}(t)=R_0{}^3+C_3^1R_0(1-R_0)R_0= 3R_0{}^2-2R_0{}^3=3e^{-2\lambda t}-2e^{-3\lambda t} \quad (6)$$

$$M_{\mathrm{TMR}}=\int_0^{+\infty}R_{\mathrm{TMR}}(t)=\int_0^{+\infty}(3e^{-2\lambda t}-2e^{-3\lambda t})dt=\frac{5}{6\lambda} \quad (7)$$

5.2 本文提出的容错算法可靠性分析

系统运行区间[0, t]内，本文提出的容错算法可靠度为 $R_S(t)$与系统平均可靠运行时间为 M_S，任意两个核正常工作，或者对于两个核中任一个发生错误则要启动第三个核开始工作，以保证系统正常工作，则可计算出 $R_S(t)$与 M_S，如公式(8)~(9)所示。

$$R_S=R_0{}^2+C_3^2C_2^1R_0(1-R_0)R_0=7R_0{}^2-6R_0{}^3 \quad (8)$$

$$M_S=\int_0^{+\infty}R_S(\mathrm{t})\mathrm{dt}=\int_0^{+\infty}(7e^{-2\lambda t}-6e^{-3\lambda t})dt=\frac{3}{2\lambda} \quad (9)$$

对比公式(7)与(9)，可发现对于不同核具有相近的不可靠性参数 λ 时，本文算法的系统正常可靠运行时间大约是现有 TMR 的 1.8 倍(M_S/M_{TMR} =1.8)，由此从理论上分析出本文容错调度方法的优越性，下面通过实验说明本文提出的方法优势。

6 实验与仿真

6.1 仿真平台搭建

SimpleScalar 模拟器是由威斯康星大学 Austin 与 Burger 设计且源代码公开的计算机处理器体系结构重要的模拟器[27-28]，支持 PISA，ARM，X86 等多种指令集，因此本文构建以 SimpleScalar 模拟器为基础的异构多核仿真实验平台，如图 5 所示。

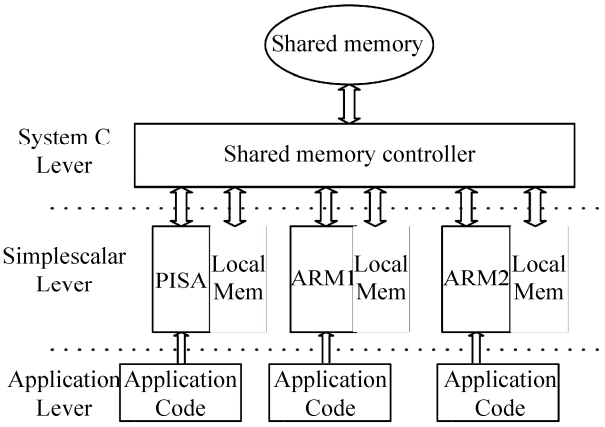


图 5 异构多核仿真模拟结构图

Fig. 5 Heterogeneous multi-core simulation architecture

选用 PISA、ARM1 和 ARM2 (ARM1 与 ARM2 采用相同的指令集，不同性能配置) 组成异构平台，采用 SystemC 作为开发工具，仿真内核，同时定义软硬件成分，可对硬件进行建模描述，核间通过共享存储单元进行通信并实现同步，采用 SimOutorder 进行功能仿真，本文中 PISA、ARM1、ARM2 在模拟器 SimpleScalar 中具体的配置如表 1 所示[29-31]。

表 1 异构多核处理器配置表

Tab. 1 Heterogeneous multi-core processor configuration

CORE Type	PISA	ARM1	ARM2
Fetch/Issue/commit	4/4/4	4/4/4	4/2/2
ROB/LSQ Entries	128/64	128/64	64/32
Int/Fp units	4/4	4/4	2/3
RUU Size	16		
Pipeline width	5		
FUs	3int add, 1int mult, 1int div 1fp add, 1fp mult, 1fp div		
ITLB	16-way, 4096 bytes page, 4-way LRU, 30 cycle miss penalty		
DTLB	32-way, 4096 bytes page, 4-way LRU, 30 cycle miss penalty		
Branch Prediction	Gshare:9, PHT:4096, BTB:512, 2-way, group-mapped, Random		
L1 Icache	64KB, 2-way group-mapped, Random		
L1 Dcache	64KB, 2-way group-mapped, Random		

6.2 仿真测试用例与方法

为了考虑本文算法在执行任务时多样性的优

越性，在实验时选用 SPEC2000、MediaBench、排序算法作为测试用例，其中 SPEC2000 选用整型(记为 SPEC2000_int) 进行测试取平均值，对于 MediaBench 应用，选用自适应差分脉冲编码调制 G.721 为测试对象，排序算法针对 600 个数进行排序(记为 SORT600)。由于 SPEC2000_int、G.721、SORT600 三种测试用例的任务多样性是逐步递减的，这比较有利于测试出本文的方法在执行不同应用任务时的性能差异。

选用传统 TMR、DMR 和 PB 容错调度方法的执行性能作为对比分析。把各个测试用例执行时间进行标准化处理，以本文提出的算法执行完成的时间作为基准，以 TMR、DMR 和 PB 方法执行时间分别与基准进行比较，分别计算相对执行时间，从而得出同一种测试用例在不同方法中执行的效率，对于 PB 方法，选用目前多核或多机系统中容错调度方法具有代表性的 TPFTRM[32]算法来执行各种应用任务进行性能对比分析。

6.3 功耗评价模型

Wattch 是由 Princeton 大学布鲁克斯教授提出的处理器功耗评价模型，是在 SimpleScalar 基础上开源的功耗分析工具，可以用来评估处理器功耗，用户可以根据实际灵活修改和完善，嵌入到 SimpleScalar 之中，使用范围非常广泛。由于动态功耗是系统总功耗中绝对主要组成部分，因此在实验中用 Wattch 模拟出系统动态功耗，通过电容、电压和时钟频率以公式 $P=\alpha\cdot C\cdot {V_{dd}}^2\cdot f$ 获取，其中 α 是动态功耗因子，C 表示电容，V_{dd} 是电压，f 是时钟频率[33-34]。

6.4 实验结果与分析

图 6 反映了执行本文提出的容错算法 FTSAS 与当前的 TMR、DMR、PB 容错算法时的性能差异。在执行 SPEC2000_int、G.721、SORT600 测试用例时，FTSAS 算法比 TMR，DMR，PB 平均综合性能分别提高了 14.9%、13.4%、10.3%。

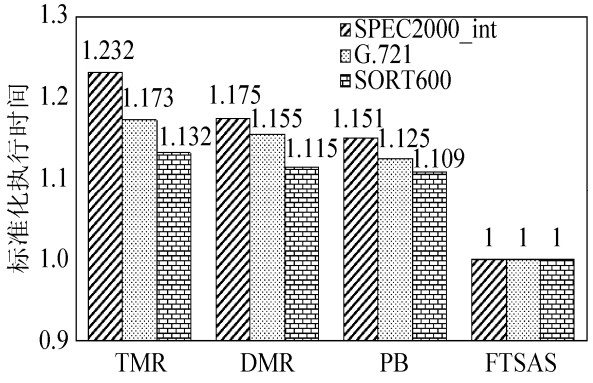


图 6 FTSAS 与其他算法性能对比

Fig. 6 Performance comparison of FTSAS and others

由图 6 可以发现，在执行各种测试用例时，相对 DMR、PB 方法，FTSAS 的性能比 TMR 方法提高幅度最大，由于 TMR 需要等 3 个模块都执行完当前任务并进行比较后才可以执行下一任务，这降低了 TMR 算法性能。FTSAS 执行 SPEC2000_int 时，比执行 G.721 和 SORT600 两种应用任务时能够取得更高的性能，这是因为 SPEC2000_int 包含多样性的整型任务，而在执行 G.721 和 SORT600 时，每一次任务类型多样性较弱，执行 SPEC2000_int 时更容易发挥异构多核的特点，当任务与核特点匹配时，那么这个核执行速度将会得到加快，这说明本文提出的算法 FTSAS 适合在执行差异性、多样化程度较大的任务，能充分发挥异构多核的优势。

为了能够真实模拟仿真系统发生错误的环境，采取的办法是在存储空间中以一定的概率随机的对数据修改，模仿处理器的瞬态故障。在注入 200、2 000、6 000 个错误数后，分别执行 SPEC2000_ int、G.721、SORT600 测试用例，对比各种容错调度算法的系统可靠性。通过实验测试出 DMR 只有检错不具备容错能力，因此只对比 FTSAS 与 TMR、PB 调度性能。预先设定可靠性目标为 0.99，注入错误后执行任务可得 TMR、PB、FTSAS 的可靠性，如图 7 所示，TMR、PB、FTSAS 容错调度算法的可靠性均在预设可靠性目标之上，且具有相近的容错能力。

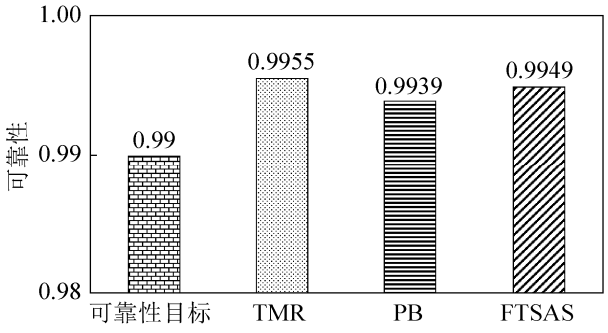
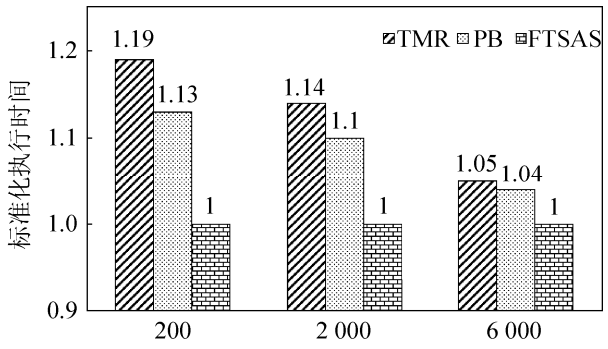


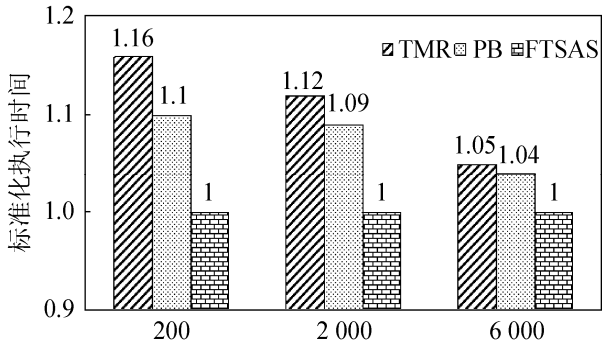
图 7 各个算法可靠性对比

Fig. 7 Reliability comparison of each algorithm

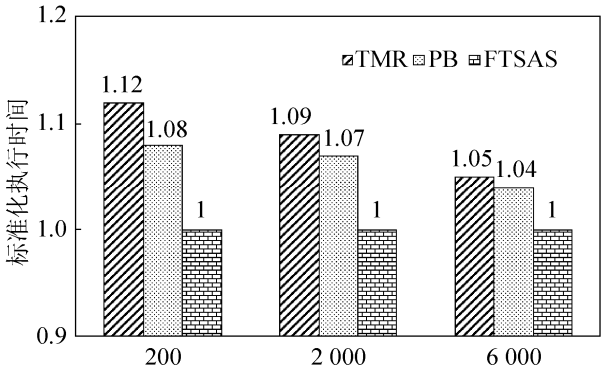
由图 8 可知，在注入 200、2 000、6 000 个错误后，分别执行 SPEC2000_int、G.721、SORT600 时，可以发现，FTSAS 比 TMR、PB 平均综合性能分别提高了 11.4%、9.2%、4.3%。由此可以发现，随着发生错误数的增加，FTSAS 的平均性能优势在减小，这主要由于推测机制的作用在减弱，尤其当注入错误数达到 6 000 时，平均性能优势降低到 4.3%，这种情况在实际中非常罕见，因为一旦在系统执行中发生如此多的错误，应该是硬件模块发生了错误，需要排查硬件模块，检修或更换问题单元。



(a) 注入错误后执行 SPEC2000_int 的性能对比



(b) 注入错误后执行 G.721 的性能对比



(c) 注入错误后执行 SORT600 的性能对比

图 8 注入错误后，不同算法性能对比

Fig. 8 Performance comparison of different algorithms after error injection

6.5 执行功耗分析

为了更好显示 FTSAS 的综合优势，需要在相同执行环境下，对 FTSAS 与 TMR、PB 方法的功耗进行分析。在 SimpleScalar 基础上利用 Wattch 功耗评价模型，结合 Xu[27]和郑凯[35]方法，修改 SimpleScalar 模拟器，分别计算 FTSAS、PB 与 TMR 执行任务时的功耗。在未发生错误时，FTSAS 的功耗大约为 TMR、PB 功耗的 70.4%、81.5%，这主要由于执行 FTSAS 算法时，当有两者一致时，立即终止第 3 个模块，同时领先的核继续执行，这样从系统总体角度分析，提高了执行效率和降低了功耗。分别注入 200、2 000、6 000 个错误后，在执行 SPEC2000_int、G.721、SORT600 测试用例后，通过计算发现，FTSAS 比 TMR、PB 平均功耗分别降低了 19.3%、12.3%，如图 9 所示。

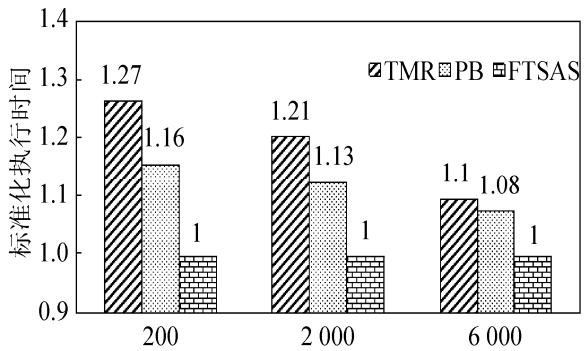


图 9 注入错误后，3 种算法的功耗关系

Fig. 9 Power consumption of three algorithms after error injection

DOI: 10.16182/j.issn1004731x.joss.19-FZ0346

随着发生错误数的增多，FTSAS 在执行中需要等第 3 个核执行完成的次数在增加，因此相应的功耗也随之提高，功耗优势在减弱，当注入 6 000 个错误时，FTSAS 功耗优势平均降至 7%左右，而系统执行时发生 6 000 个错误的概率很小，因此在一般情况下，FTSAS 算法在提高性能的同时，降低了系统总功耗，体现 FTSAS 的性能和功耗优势。

7 结论

推测机制是提高微处理器执行性能的一种重要技术。针对特殊领域，处理器所面临的复杂环境，瞬态故障频发，目前常用的 TMR 在解决处理器瞬态故障时具有速度慢，功耗较高的缺点。异构多核在面对不同任务体现出不同的性能，与核相匹配的任务，执行速度较快，反之速度较慢。基于这些特点，本文提出了基于推测机制的异构多核处理器的容错算法 FTSAS。把待执行任务流按照 DAG 流图倒推法确定优先级，设定任务可靠度和系统可靠度作为判定系统可靠度目标的依据。开始执行时，把待执行任务同时分配至各个核，与任务相匹配的核最先执行完成，然后记录状态，不停留并继续向前推测执行下一任务，第 2 个核执行完成时再与第一个核的结果进行比较，如果一致，则终止第 3 核，与第 1 核同步到下一任务的执行，如果不一致，则需要等到第 3 核任务执行完成，采取多数一致原则判断正确结果，然后再执行下一任务。经过理论分析和仿真实验测试，FTSAS 在执行性能和功耗上都体现了比现有方法具有相应的综合优势。

下一步将在详细分析异构多核处理器的特征之后，然后调整合适任务至相匹配的核执行，在保障系统可靠性的前提下，进一步提高系统执行效率，降低系统执行功耗。

参考文献:

[1] McNairy C. Exascale fault tolerance challenge and approaches[C]. IEEE International Reliability Physics Symposium. Burlingame,USA: IEEE, 2018: 3C.4.1-10.

[2] Chen J, John L K. Efficient Program Scheduling for Heterogeneous Multicore Processors[C]. 46th ACM/ IEEE Design Automation Conference. San Francisco, USA: IEEE, 2009: 927-930.

[3] Baital K, Chakrabarti A. Dynamic Scheduling of Real-Time Tasks in Heterogeneous Multicore Systems[J]. IEEE Embedded Systems Letters (S1943-0671), 2019, 11(1): 29-32.

[4] Greenhalgh P Big. LITTLE processing with ARM Cortex -A15&Cortex-A7. [EB/OL]. [2018-09]. http://www. arm.com.

[5] NVIDIA Corporation. Variable SMP–a multi-core CPU architecture for low power and high performance[EB/OL]. http//www.Nvidia.com. 2018.

[6] Chitlur N, Srinivasa G, Hahn S, et al. QuickIA: Exploring heterogeneous architectures on real prototypes[C]. IEEE International Symposium on High-Performance Computer Architecture (HPCA). New Orleans, USA: IEEE, 2012: 1-8.

[7] 海思麒麟 980 参数[EB/OL]. [2019-04-01]. https: // baike. baidu.com.
Parameters of the kirin 980[EB/OL]. [2019-04-01]. https: // baike.baidu. com.

[8] Naithani A, Eyerman S, Eeckhout L. Optimizing Soft Error Reliability Through Scheduling on Heterogeneous Multicore Processors[J]. IEEE Transactions on Computers (S1557-9956), 2018, 67(6): 830-846.

[9] Raparti V Y, Kapadia N, Pasricha S. CHARM: A check point-based resource manage-ment framework for reliable multicore computing in the dark silicon era[C]. IEEE International Conference on Computer Design (ICCD). Scottsdale, USA: IEEE, 2016: 201-208.

[10] Karlsson J, Liden P, Dahlgren P, et al. Using Heavy-ion Radiation to Validate Fault Handling Mechanisms[J]. IEEE Micro (S1937-4143), 1994, 14(1): 8-23.

[11] Lim H, Kim T, Lee D, et al. LARECD: Low area over-head and reliable error correction DMR architecture [C]. International SoC Design Conference. Seoul, South Korea: IEEE, 2018: 27-28.

[12] Zheng P, Zheng Q, Zeng Z. The Signal Design and Simulation of Triple Modular Redundant (TMR) Computer[C]. International Conference on Cybernetics and Intelligent Systems (CIS) and Conference on Robotics, Auto-mation and Mechatronics (RAM). Ningbo, China: IEEE, 2017: 758-762.

[13] 孙晓星. 面向方面的软件容错模型设计与分析技术[D]. 上海: 华东理工大学, 2012.
Sun Xiaoxing. Aspect-oriented Modeling and Analysis

Techniques for Software Fault Tolerance[D]. Shanghai: East China University of Science and Technology, 2012.

[14] Abate F, Sterpone L, Lisboa C A, et al. New Techniques for Improving the Performance of the Lockstep Architecture for SEEs Mitigation in FPGA Embedded Processors [J]. IEEE Transactions on Nuclear Science (S1558-1578), 2009, 56(4): 1992-2000.

[15] Roy A, Aydin H. Energy-efficient primary/backup scheduling techniques for heterogeneous multicore systems[C]. International Green and Sustainable Computing Conference. Orlando, USA: IEEE, 2017: 1-8.

[16] Ning L, Yao W, Ni J, et al. Fault Tolerance CMP Architecture based on SMT Technology[C]. International Multi-Symposiums on Computer and Computational Sciences. Iowa City, USA: IEEE, 2007: 425-429.

[17] 贾佳, 杨学军, 马亚青. 静态分析面向异构系统的应用级 Checkpoint 设置问题[J]. 软件学报, 2013, 24(6): 1361-1375.
Jia Jia, Yang Xuejun, Ma Yaqing. Static analysis for placement of application-level Checkpoints on heterogeneous system[J]. Chinese Journal of Software, 2013, 24(6): 1361-1375.

[18] 李祖松, 许先超, 胡伟武, 等. 龙芯 2 号处理器的同时多线程设计[J]. 计算机学报, 2009, 32(11): 2265-2273.
Li Zusong, Xu Xianchao, Hu Weiwu, et al. Design of the Simultaneous Multithreading Godson-2 Processor[J]. Chinese Journal of Computers, 2009, 32(11): 2265-2273.

[19] 贾佳, 杨学军, 李志凌. 一种基于冗余线程的 GPU 多副本容错技术[J]. 计算机研究与发展, 2013, 50(7): 1551-1562.
Jia Jia, Yang Xuejun, Li Zhiling. A Redundancy-Multithread-Based Multiple GPU Copies Fault-Tolerance Technique[J]. Chinese Journal of Computer Research and Development, 2013, 50(7): 1551-1562.

[20] 曹英. 大数据环境下 Hadoop 性能优化的研究[D]. 大连: 大连海事大学, 2013.
Cao Ying. Research of Performance Optimization of Hadoop in BigData[D]. Dalian: Dalian Maritime University, 2013.

[21] 陆志伟. 航空器无冲突 4D 航迹推测与生成[D]. 南京: 南京航空航天大学, 2012.
Lu Zhiwei. The Conflict-free 4D Trajectory Prediction and Generation for Aircrafts[D]. Nanjing: Nanjing University of Aeronautics and Astronautics, 2012.

[22] 王丽华. 计算机容错系统的体系结构与安全性研究[D]. 成都: 西南交通大学, 2002.
Wang Lihua. The Research About the Architecture and Safety of the Computer Fault-tolerant System[D]. Chengdu: Southwest Jiaotong University, 2002.

[23] 谢国琪, 李仁发, 刘琳, 等. 异构分布式 DAG 可靠性模型与容错算法[J]. 计算机学报, 2013, 36(10): 2020-2033.
Xie Guoqi, Li Renfa, Liu Lin, et al. DAG Reliability Model and Fault-Tolerant Algorithm for Heterogeneous Distributed Systems[J]. Chinese Journal of computers, 2013, 36(10): 2020-2033.

[24] Benoit A, Hakem M, Robert Y. Fault tolerant scheduling of precedence task graphs on heterogeneous platforms [C]. IEEE International Symposium on Parallel and Distributed Processing. Miami, USA: IEEE, 2008: 1-8.

[25] Benoit A, Hakem M, Robert Y. Contention Awareness and Fault-tolerant Scheduling for Precedence Constrained Tasks in Heterogeneous Systems[J]. Parallel Computing (S0167-8191), 2009, 35(2): 83-108.

[26] 付剑. 星载计算机硬件容错设计与可靠性分析[D]. 长沙: 国防科技大学, 2009.
Fu Jian. The hardware Fault-tolerant Design and Reliability Analysis of On Board Computer[D]. Changsha: National University of Defense Technology, 2009.

[27] Xu J, Zhu Y, Ni J. A Simulator for Multicore Processor Micro-architecture Featuring Intercore Communication, Power and Thermal Behavior[C]. International Conference on Embedded Software and Systems Symposium. Sichuan China: IEEE, 2008: 237-242.

[28] Simplescalar LLC to serve and project[EB/OL]. [2018-09-20]. http:// www.simple scalar.com.

[29] 张福新, 章隆兵, 胡伟武. 基于 Simplescalar 的龙芯 CPU 模拟器 Sim-Godson[J]. 计算机学报, 2007, 30(1): 68-73.
Zhang Fuxin, Zhang Longbin, Hu Weiwu. Sim-Godson: A Godson Processor Simulator Based on SimpleScalar[J]. Chinese Journal of Computers, 2007, 30(1): 68-73.

[30] 陈文智, 姜振宇, 吴帆. 基于 MIPS 体系的扩展指令融合技术[J]. 计算机学报, 2008, 31(11): 1888-1897.
Chen Wenzhi, Jiang Zhenyu, Wu Fan. Instruction Fusion Technology for the MIPS[J]. Chinese Journal of Computers, 2008, 31(11): 1888-1897.

[31] Luo Y, Packirisamy V, Hsu W. Energy efficient speculative threads: Dynamic thread allocation in same-ISA heterogeneous multicore systems[C]. Inter-national Conference on Parallel Architectures and Compilation Techniques. Vienna, Austria: IEEE, 2010: 453-464.

[32] 王健, 孙建伶, 王新宇, 等. 容错多处理机中一种高效的实时调度算法[J]. 软件学报, 2009, 20(10): 2628-2636.
Wang Jian, Sun Jianling, Wang Xinyu, et al. Efficient Scheduling Algorithm for Hard Real-Time Tasks in Primary-backup Based Multiprocessor Systems[J]. Chinese Journal of Software, 2009, 20(10): 2628-2636.

[33] Brooks D, Tiwari V, Martonosi M. Wattch: A framework for Architectural-Level Power Analysis and Optimization[C]. Proceedings of the 27th Annual International Symposium on Computer Architecture. Vancouver, BC, Canada: IEEE, 2000: 83-94.

[34] 倪俊杰. 基于 SimpleScalar 的性能和功耗分析多核模拟器[D]. 上海: 上海交通大学, 2008.
Ni Junjie. A Multiprocessor Simulator with Power Analysis Based on SimpleScalar[D]. Shanghai: Shanghai Jiao Tong University, 2008.

[35] 郑凯. 对数据在异构多核处理器模拟器中进行任务划分的研究[D]. 上海: 上海交通大学, 2008.
Zheng Kai. Evaluation of Partitioning Methods for Application on a Heterogeneous Multicore Processor Simulator[D]. Shanghai: Shanghai Jiao Tong University, 2008.