一种应用代价评估的推测多线程路径预测方法

李远成，赵银亮，阴培培，韩博
（西安交通大学计算机科学与技术系，710049，西安）

摘要：推测多线程技术对于自动并行化非规则程序是有效的，然而基于控制流图和分支预测方法的线程划分方法，不可避免地受到划分路径上存在的控制依赖和数据依赖制约.针对现有的路径预测方法在考虑控制依赖影响的同时却不能有效地综合考虑数据依赖影响的问题，提出一种新的基于代价评估的路径预测方法，通过引入数据依赖模型，综合评估控制和数据依赖两种影响因素，寻求一条具有近似最小推测开销的推测划分路径.实验结果表明，文中提出的路径预测方法能够计算出代价更小的推测划分路径，并取得了更好的加速比性能，总体上系统可以得到2.43%的加速比性能提升.

关键词：推测多线程；代价评估模型；路径预测技术；数据依赖模型
中图分类号：TP302 **文献标志码**：A **文章编号**：0253-987X(2010)12-0022-06

A Cost Estimation Based Speculative Path Prediction Method for Speculative Multithreading

LI Yuancheng, ZHAO Yinliang, YIN Peipei, HAN Bo
(Department of Computer Science and Technology, Xi′an Jiaotong University, Xi′an 710049, China)

Abstract: Speculative multithreading (SpMT) technology is an effective mechanism for automatic parallelization of irregular programs. However, just generating speculative threads based on the control flow graph which only contains branch probability information, it is inevitable that there may be excessive constraints resulting from control and data dependence in practice. Therefore, it is very important to understand the trade-offs between different speculative paths. In this paper, by introducing the data dependence model and discussing the trade-offs between different speculative paths, we propose a novel cost estimation based speculative path prediction method which comprehensively takes account of control- and data-dependence. By this method, we attempt to seek a speculative path which has the minimum cost overhead. The experimental results show that there are interesting trade-offs between different speculative paths and we can indeed get better performance. On average, we achieve 2.43% performance improvement.

Keywords: speculative multithreading; cost estimation model; path-based profiling technology; data dependence mode

挖掘并行性是提高程序执行性能的有效方法之一.随着指令级并行(ILP)遇到越来越多的瓶颈以及片上多处理器(CMP)的迅速发展，线程级并行(TLP)逐渐成为一个更佳的选择.推测多线程(SpMT，speculative multithreading)技术[1-3]作为一种能有效开发非规则程序并行性的线程级并行技术，得到了迅速发展.SpMT技术在允许存在大量的控制和数据依赖的情况下，以激进的方式发掘线程级并行性.在执行的过程中，每一个推测线程执行程序的不同部分，并在运行时由执行模型检测控制和数据等依赖的发生，如果出现依赖违规，采取撤销并重新运行等硬件手段来保证程序执行的正确性.显然，推测多线程技术在试

收稿日期：2010-06-13. 作者简介：李远成(1981—)，男，博士生；赵银亮(联系人)，男，教授，博士生导师. 基金项目：国家高技术研究发展计划资助项目(2008AA01Z136).

图通过激进的方式发掘线程级并行性的同时,控制和数据等依赖成为制约加速比性能提高的重要因素.

目前,研究者已经提出了多种推测多线程路径预测技术. Jayanth 等[4]提出利用分支历史信息,通过硬件实现来进行分支预测的方法. 但是,基于分支历史信息的分支预测的局限性在于当程序执行面临经常撤销时,其预测错误率将显著增加. 为了解决这个问题,Liu 等[3]采用了基于路径分支概率信息的预测技术,通过动态运行程序并计算路径的分支概率,选取分支概率值较大的路径作为推测划分的路径. 现有的这些路径预测方法的一个共同特点,就是在进行路径预测时仅仅考虑了控制依赖的因素,并在确定推测线程之后采取数据值预测的技术[5-6]和线程同步机制来减少线程间的数据依赖,以增加程序的并行性. 但是,这些预测技术都恰恰忽略了不同路径推测执行时的数据依赖的影响,而根据程序的特点可知,不同的路径将会导致不同的控制和数据依赖. 因此,评估不同路径所带来的控制和数据依赖的开销并进而选取最优推测路径至关重要.

本文基于项目组开发的 Prophet 并行编译器平台[7],提出一种基于代价评估的路径预测方法,通过综合评估控制和数据依赖两种重要影响因素,寻求一条具有最小推测开销的推测划分路径,然后推测划分该开销代价最小的路径,以得到更好的加速比性能.

1　SpMT 执行模型

在 Prophet 并行编译器中,串行程序被划分成多个推测线程,每个推测线程执行程序的不同部分. 在并行推测执行中,有且只有一个是非推测线程,该线程可以提交其执行结果,代表程序当前确定执行的状态. 其他线程为推测线程,由串行程序代码片段及其预计算片段(pre-computation slice,P-slice)构成. P-slice 是由编译器根据程序切片技术生成的一小段代码,用来对推测线程使用的 live-ins 变量(线程体使用但并非由该线程定义的值)进行值预测. 如图 1 所示,SP(Spawn Point)和 CQIP(Control Quasi-independent Point)指令唯一确定一个激发线程对. 当程序执行到 SP 时,如果现有资源允许激发,则将激发一个新的推测线程. CQIP 是线程的分界点,当确定线程(线程 0)执行到 CQIP 时,将验证其直接后继线程(线程 1)在 P-slice 中产生的 live-ins 数据,若验证正确,则确定线程提交其执行结果,然后将确定执行的权限传递给其直接后继线程;若验证失败,则撤销其所有推测子线程,然后跳过 P-slice 片段,确定执行其直接后继线程.

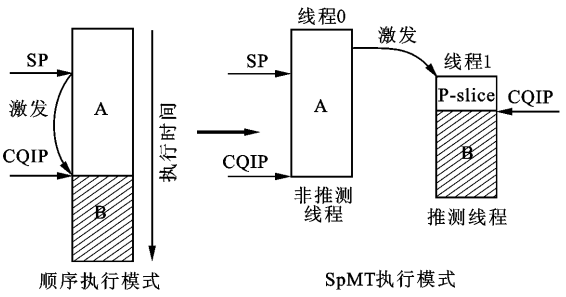

图 1　顺序执行模式和 SPMT 执行模式

2　基于代价评估的路径预测模型

2.1　数据依赖模型

程序段之间的数据依赖是影响线程划分结果的重要因素,因此,为了评估数据依赖因素对线程划分结果的影响,本文尝试引入度量和计算程序间数据依赖程度的数据模型,以寻求对数据依赖因素进行有效的评估. 首先,对程序中的所有变量,确定其引用-定值链. 引用-定值链(简称为 ud(use-defined)链)是关于变量数据流信息的稀疏表示. 一个变量的 ud 链是一个列表,假设在程序中某点 u 引用了变量 a 的值,则将能够到达该引用的 a 的所有定值点的全体称为 a 在引用点 u 的 ud 链[8]. 也就是说,ud 链连接一个变量的引用到所有可能流经到该引用的定值,如图 2 所示. 在求解 ud 链时应该考虑所有可能路径,尤其是在出现循环时,例如变量 j 在 d_4 处的 ud 链应包含 d_2、d_4、d_5 这三个定值点,虽然从形式上看 d_5 处的定值位于 d_4 处的引用之后,但由于循环控制流的影响,d_4 和 d_5 在位于两次不同的迭代时会出现 d_5 中的定值位于 d_4 中的引用之前的情况. 此外,变量 j 在 d_7 处的 ud 链也需考虑来自所有可能路径上的定值点.

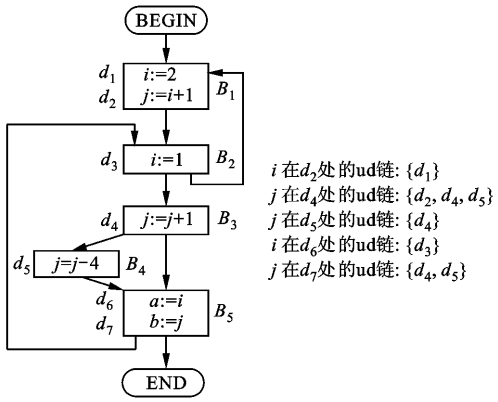

图 2　引用-定值链的构造

由于线程划分以基本块为单位,每个线程都由一到多个基本块组成,因此程序段之间的数据依赖关系

转化为两个程序段所包含的基本块集合之间的数据依赖关系. 为了评估基本块之间的数据依赖关系,将这里的线程之间的 ud 链转化为依赖弧. 依赖弧定义为一个有向弧,每一条依赖弧连接一个变量的引用-定值位置偶对. 从程序段 $T1$ 流入程序段 $T2$ 的依赖弧表示 $T1$ 中引用(读)了 $T2$ 中定值(写)的某个变量,说明两个程序段之间发生读写依赖,依赖弧的数目则反映了两个程序段之间数据依赖的程度. 因此,两个程序段之间的数据依赖程度问题可以用二者之间的依赖弧数目来进行评估.

2.2 路径预测模型

为了进一步定量分析和评估不同推测路径的开销,本节将综合控制依赖和数据依赖两种因素的影响,建立路径预测模型并确立一套预测规则. 以图 3 为例,本文详细分析如何对不同的路径进行评估. 图 3 所示的是一个程序的超级控制流图(super control flow graph, SCFG)的子图,该子图是在加权控制流图(weighted control flow graph, WCFG)的基础上,通过结构化分析[9-10]将循环归结为单入口、单出口的超级块构建而成的抽象控制流图. 图中 A、B 等大写字母标识的是基本块或者抽象的超级块. 对于块 A,块 F 是其最近后向控制无关点.

为了评估控制依赖的因素,对 A-B-C-D-F 路径和 A-E-F 路径引入分支概率,假设分别为 P_{AB} 和 P_{AE}. 同时,假设 $I(AB)$ 和 $I(AE)$ 分别表示沿路径 A-B-C-D-F 和路径 A-E-F 执行的动态指令数目,假设每条指令周期为 T. 另外,为了评估数据依赖因素的影响,假设在相同的软硬件环境下,当控制推测成功时,程序段可以获得的加速比性能和该程序段的数据依赖程度成反比. 假设 m 和 n 分别表示 A-B-C-D-F 路径和 A-E-F 路径所指示块向上的依赖弧个数,则可以简单认定两条路径在推测成功时获得的加速比分别为 $\frac{S}{m}$ 和 $\frac{S}{n}$,此处 S 表示基于相同的软硬件环境且无任何控制和数据依赖的情况下,程序段可以获得的加速比性能. 接

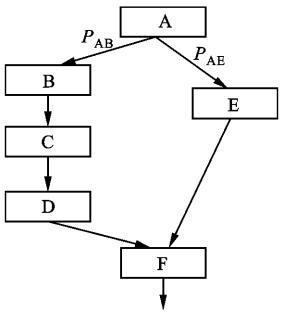

图 3 一个超级控制流图的例子

下来,按照推测路径 A-B-C-D-F 和 A-E-F 路径两种情况分析各自的推测开销.

(1)推测路径 A-B-C-D-F. 此时,路径 A-B-C-D-F 将会被划分并被推测并行执行. 考虑实际运行时发生的两种情况:当在分支概率 P_{AB} 下沿该路径运行时,路径 A-B-C-D-F 被成功推测执行;当在分支概率 P_{AE} 下,由于 A-E-F 路径没有被推测划分,因此,该路径上的程序代码将串行执行. 假设 T_{AB} 表示该段代码的等价执行时间,则

$$T_{AB} = T\left\{ I(AE)P_{AE} + \frac{m\, I(AB)}{S} P_{AB} \right\} \tag{1}$$

(2)推测路径 A-E-F. 此时,路径 A-E-F 将会被划分并被推测并行执行. 同样,考虑实际运行时发生的两种情况:当在分支概率 P_{AE} 下沿该路径运行时,路径 A-E-F 被成功推测并行执行;当在分支概率 P_{AB} 下,由于 A-B-C-D-F 路径没有被推测划分,因此,该路径上的程序代码将串行执行. 假设 T_{AE} 表示该段代码等价执行时间,则

$$T_{AE} = T\left\{ I(AB)P_{AB} + \frac{n\, I(AE)}{S} P_{AE} \right\} \tag{2}$$

进一步,假设

$$f(\cdot) = T_{AB} - T_{AE} \tag{3}$$

则由式(1)~式(3)可得

$$f(\cdot) = T\left\{ E_1(\cdot) + \frac{1}{S} E_2(\cdot) \right\} \tag{4}$$

其中

$$E_1(\cdot) = I(AE)P_{AE} - I(AB)P_{AB} \tag{5}$$

$$E_2(\cdot) = m\, I(AB)P_{AB} - n\, I(AE)P_{AE} \tag{6}$$

接下来,根据式(3)~式(6),分下面 3 种情况来讨论 $f(\cdot)$ 值的正负符号.

(1)$E_1(\cdot)E_2(\cdot) \geqslant 0$,且 $E_1(\cdot)+E_2(\cdot) \geqslant 0$,则有 $f(\cdot)=T_{AB}-T_{AE} \geqslant 0$ 且 $T_{AB} \geqslant T_{AE}$.

(2)$E_1(\cdot)E_2(\cdot) \geqslant 0$,且 $E_1(\cdot)+E_2(\cdot) < 0$,则有 $f(\cdot)=T_{AB}-T_{AE} < 0$ 且 $T_{AB} < T_{AE}$.

(3)$E_1(\cdot)E_2(\cdot) < 0$. 此时,分两种情况,当 $|E_1(\cdot)| \geqslant |E_2(\cdot)|$ 时,由于 S 的值大于等于 1,则 $f(\cdot)$ 值的正负符号由 $E_1(\cdot)$ 决定;当 $|E_1(\cdot)| < |\exp_2(\cdot)|$ 时,由于 S 的具体值未知,因此无法精确确定 $f(\cdot)$ 值的正负,也无法精确确定 T_{AE} 和 T_{AB} 的大小关系.

由于 T_{AB} 和 T_{AE} 分别表示推测路径 A-B-C-D-F 和路径 A-E-F 所带来的时间开销,因此,根据上述的分析,对超级控制流图中的一个节点 A 来说,可以容易地确定一条推测开销较小的路径. 决策规则(decision rules)可以归纳如下.

(1) 如果 $E_1(\cdot)E_2(\cdot)\geqslant 0$,且 $E_1(\cdot)+E_2(\cdot)\geqslant 0$,则选择路径 A-E-F;

(2) 如果 $E_1(\cdot)E_2(\cdot)\geqslant 0$,且 $E_1(\cdot)+E_2(\cdot)<0$,则选择路径 A-B-C-D-F;

(3) 如果 $E_1(\cdot)E_2(\cdot)<0$,当 $|E_1(\cdot)|\geqslant|E_2(\cdot)|$ 时,若 $E_1(\cdot)>0$ 选择路径 A-E-F,若 $E_1(\cdot)<0$,选择路径 A-B-C-D-F;当 $|E_1(\cdot)|<|E_2(\cdot)|$ 时,则直接选取分支概率较大的路径.

对于一个过程的 SCFG,首先通过等价转换,如插入空指令基本块方法[11]等转换方式将其转化为跳转路径分支不超过两个的等价 SCFG;然后,直接使用上述预测规则即可进行路径预测. 确定推测路径的伪代码如图 4 所示. 算法中,end_block 表示程序的出口块节点,start_block 表示当前处理的块节点,decision rules 即是上述模型确定的决策规则.

```
算法: Path Determination
输入: Transformational SCFG of a procedure;
输出: the denoted path
Path_Determination(start_block,end_block){
  if (start_block == end_block)
   return 1;
  pdom_block==postdominator of start_block
  if(start_block.leftchild==pdom_block)
   {denoting the path between start_block and pdom_block;
     strat_block=pdom_block;
Path_Determination(start_block,end_block);
}
   else {determine the path according to the decision rules;
     start_block=pdom_block;
Path_Determination(start_block,end_block);
     }
}
```

图 4　路径预测伪代码

3　实验结果及分析

3.1　模拟环境及实验结果

本文提出的基于代价评估的路径预测模型已经在 Prophet 并行编译器[7]上实现. Prophet 模拟器[12]采用 MIPS 指令集,通过扩展指令集实现对 SpMT 处理器的模拟. 每个处理单元 PE 有自己的程序计数器、取指令单元、解释指令单元和执行单元,用来从线程中取指令并执行指令. 另外,由于 Prophet 是基于 SUIF/MACHSUIF 编译框架[12]实现的并行编译器原型,而 SUIF/MACHSUIF 编译框架通过对原始控制流图插入空指令基本块的方式,成功实现了对程序控制流图的转换,将其转换成为等价的控制流图. 转换后的等价的控制流图含有若干空指令基本块,同时每个基本块的出度不超过 2. 也就是说,对每个基本块来说,其跳转的路径分支最多为 2 个. 因此,本文提出的路径评估方法可以直接在 Prophet 并行编译器上实现.

选择 Olden 程序包的一个子集对本文提出的改进的推测路径预测模型进行测试. Olden 基准程序[13]是由 Princeton 大学提供的一个测试集,具有复杂的数据结构以及对这些数据结构的操作,例如都是对树和链表进行合并、遍历等操作. 另外,Olden 程序结构大多为递归,具有复杂的线程间依赖关系. Olden 基准程序测试集的这些复杂的控制和数据依赖的特性,非常有利于测试本文提出的基于代价评估的预测路径的有效性. 本文还实现了一个 rook 测试程序. 该测试程序的功能是用来解决有禁区的棋盘上最多放置的棋子个数,主要采用二叉树数据结构,同时,该程序的特点决定了其蕴含较为明显的适合折中的路径,因此,可以更有效地说明本文改进方法的有效性. 图 5 给出了在同一组参数和 4 个处理器核的情况下,基于传统方法[14]和本文改进方法所得的测试程序加速比性能的实验数据.

3.2　实验结果分析

由图 5 可见,有 5 个测试程序都有一定程度的性能提升,特别是 rook 和 perimeter 程序提升较为明显,但 treeadd 和 mst 程序的性能没有变化. 总体上,平均性能提升达到 2.43%.

为了进行进一步的分析,定义

$$\text{性能增长率}=\frac{\text{改进方法加速比}-\text{传统方法加速比}}{\text{传统方法加速比}} \tag{7}$$

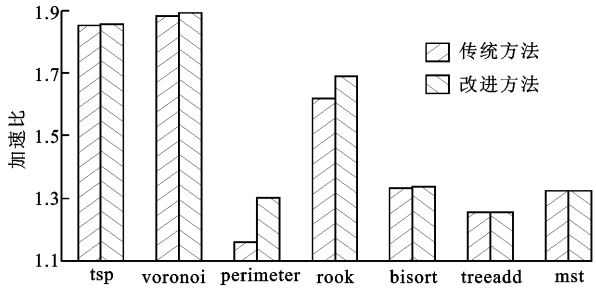

图 5　传统方法和改进方法的加速比性能比较

表 1 给出了基于传统路径预测方法和本文提出的改进路径预测方法的编译器所产生的推测线程的动态信息. 由表 1 可见,对于测试程序 voronoi 和 perimeter,基于改进方法所激发的线程数目均有较为明显的增加,此外,虽然线程激发成功率有一定程度的

表1 基于传统方法和改进方法的线程激发动态信息的比较

测试程序	激发线程数目		线程激发成功率		成功激发线程数目		性能提升
	传统方法	改进方法	传统方法	改进方法	传统方法	改进方法	
mst	733	733	0.667 12	0.667 12	489	489	0.00%
rook	2 602	2 430	0.517 68	0.566 67	1 347	1 377	4.36%
bisort	16 154	15 996	0.564 13	0.569 33	9 113	9 107	0.04%
voronoi	111 907	113 053	0.918 94	0.910 76	102 836	102 964	0.25%
treeadd	40 963	40 963	0.500 06	0.500 06	20 484	20 484	0.00%
perimeter	9 677	11 769	0.682 55	0.638 63	6 605	7 516	12.29%
tsp	91 922	92 007	0.941 89	0.941 02	86 580	86 581	0.10%

下降,但是成功激发的线程数目都有所增加.这说明,虽然推测成功率微幅降低会增大线程撤销概率而产生额外开销,但是改进的预测方法能够成功激发更多的推测线程,提高程序执行的并行性程度.本例中后项占优,所以最终提高了程序执行的加速比.

对于测试程序 rook 和 bisort,由表1可见,相对于传统方法,基于改进方法的线程激发数目有不同程度的减少,同时,线程激发的成功率却有所增加.这说明,改进的预测方法有效地评估了不同路径分支所导致的数据依赖因素,成功地选取了数据依赖程度更小的路径,并通过激发数据依赖程度更小的线程而提高了推测成功率.虽然成功激发的线程数目降低会影响程序的并行度,但是提高推测成功率将可以更有效地减少线程撤销所带来的时间开销,在此例中后项占优,故最终提升了程序加速比性能.

对于 mst 和 treeadd 程序,从表1可以很清晰地看出,线程激发信息没有任何变化,这说明,在这些测试程序中,改进方法评估的代价最小路径和传统方法预测的路径趋于一致,因此,程序的加速比保持了原来的性能.同时,由表1可见,除了 rook 和 perimeter 程序,其他测试程序性能提升并不太明显,这说明,一方面,这些测试程序并不具有较为明显的适合折中的路径;另一方面,本文所采用的数据依赖模型尚不够精确,可能会导致某些潜在的依赖信息不能被完全提取,因此会导致数据依赖影响因素被弱化,使评估出现一定程度的偏差.但是,这种评估偏差可以通过提高数据依赖模型的精确度来有效地消除,或者降低到可以接受的范围.

总体上,系统得到了2.43%的加速比性能提升.这个结果说明,本文提出的基于代价评估改进的路径预测方法可以有效地对不同的路径进行评估,并可寻求一条具有更小推测开销的推测路径,进而提高系统的加速比性能.

4 结 论

本文提出一种基于代价评估的推测路径预测改进方法.针对传统的路径预测方法仅仅依据控制依赖而忽略数据依赖影响的不足,通过引入数据依赖模型,综合考虑了程序间控制依赖和数据依赖等重要影响因素,建立了一个评估模型来分别评估推测不同路径所带来的开销.然后,确立了一套预测路径选择决策规则,用以寻求具有最小推测开销的最优推测路径.基于 Prophet 编译器平台对测试程序集的测试结果表明,本文提出的基于代价评估的推测路径预测改进方法可以有效地提高程序的加速比性能.

从测试结果也可以看到,由于预测评估所采用的数据依赖模型固有的不精确性,导致了某些数据依赖信息不能完全提取出来,从而影响了一些程序的加速比性能提升.因此,进一步的工作将从更接近机器的目标代码层面上进行寄存器数据依赖和内存数据依赖分析,提高数据依赖模型的精确度,从而提高推测多线程的加速比.

参考文献:

[1] BHOWMIK A, FRANKLIN M. A general compiler framework for speculative multithreaded processors[J]. IEEE Trans on Parallel and Distributed Systems, 2004, 15(8):713-724.

[2] MARCUELLO P, GONZÁLEZ A, TUBELLA J. Thread partitioning and value prediction for exploiting speculative thread-level parallelism[J]. IEEE Transactions on Computer, 2004,53(2):114-125.

[3] LIU Wei, JAMES T, LUIS C, et al. POSH: a TLS

compiler that exploit program structure[C]// Proceedings of The ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. New York, USA: ACM Press, 2006:158-167.

[4] JAYANTH G. Branch prediction in multi-threaded processors[C]// Proceedings of the 2000 International Conference on Parallel Architectures and Compilation Techniques. Piscataway, NJ, USA: IEEE, 2000:179-188.

[5] CARLOS M, CARLOS G Q, JESŰS S, et al. Mitosis: a speculative multithreaded processor based on precomputation slices[J]. IEEE Trans on Parallel and Distributed Systems, 2008,19(7): 914-925.

[6] JIMÉNEZ D A. Fast path-based neural branch prediction [C]// Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture. New York, USA: ACM, 2003:243-252.

[7] CHEN Zheng, ZHAO Yinliang, PAN Xiaoyu, et al. An overview of Prophet [C]// LNCS 5574. Berlin, Germany: Springer, 2009:396-407.

[8] AHO A V, SETHI R, ULLMAN J D. Compilers: principles, techniques, and tools[M]. Reading, Massachusetts, USA: Addison-Wesley Publishing Company, 1986.

[9] SHARIR M. Structural analysis: a new approach to flow analysis in optimizing compilers[J]. Computer Languages, 1980,5(3/4):141-153.

[10] MUCHNICK S, Advanced compiler design and implementation[M]. San Francisco, USA: Morgan Kauffmann Publishers, 1997.

[11] HALL M W, ANDERSON J M, AMARASINGHE S P, et al. Maximizing multiprocessor performance with the SUIF compiler[J]. Computer, 1996,29:84-89.

[12] DONG Zhaoyu, ZHAO Yinliang, WEI Yuanke, et al. Prophet: a speculative multi-threading execution model with architectural support based on CMP[C]// International Conference on Embedded Computing. Piscataway, NJ, USA: IEEE, 2009:103-108.

[13] CARLISLE M C. Olden Benchmark Suite [EB/OL]. (1995-01-03)[2007-06-30]. http://www.cs.princeton.edu/~mcc/olden.html.

[14] PAN Xiaoyu, ZHAO Yinliang, CHEN Zheng, et al. A thread partitioning method for speculation multithreading [C]// Proceedings of the 9th International Conference on Algorithms and Architectures for Parallel Processing. Piscataway, NJ, USA: IEEE, 2009:285-290.

[本刊相关文献链接]

网络生存适应性的多目标评估. 西安交通大学学报, 2010, 44(10):1-7.

采用离散粒子群算法的网格任务安全级调度. 西安交通大学学报,2010, 44(6):21-26.

一种适于主-从模式网络计算的事件驱动架构. 西安交通大学学报,2010, 44(2):39-43.

面向 Web 服务总线集成的分阶段优先级事件驱动架构. 西安交通大学学报,2009, 43(12):16-20.

面向变异分析的协议安全测试方法. 西安交通大学学报,2009, 43(12):11-15.

自适应滤波实时网络流量异常检测方法. 西安交通大学学报,2009, 43(12):1-5.

基于系统可靠性评估的机组检修规划模型. 西安交通大学学报,2009, 43(8):80-84.

利用应用程序特征识别降低网格监控负载的研究. 西安交通大学学报,2009, 43(8):11-16.

面向属性约束的自动信任协商模型. 西安交通大学学报,2009, 43(8):1-5.

网络安全协同防卫系统研究与实现. 西安交通大学学报,2008, 42(12):1495-1499.

一种基于性能评估的元任务调度算法. 西安交通大学学报,2008, 42(8):972-976.

广义随机 Petri 网下的组合 Web 服务建模与评价. 西安交通大学学报,2008, 42(8):967-971.

带可信度评估的连续小波分布式拒绝服务攻击检测算法. 西安交通大学学报,2008, 42(8):936-939.

基于反馈的片上多处理器系统层次负载平衡算法. 西安交通大学学报,2008, 42(2):179-183.

(编辑　武红江)