运行时消除指针别名歧义方法的加速比分析*

乔 林[1] 汤志忠[1] 张赤红[1] 苏伯珙[2]

[1](清华大学计算机科学与技术系 北京 100084)

[2](William Paterson 大学计算机科学系 美国)

摘要 采用软硬件结合的运行时消除指针别名歧义方法 SHRTD(software/hardware run-time disambiguation)适用于不可逆代码,同时,它的代码空间受到限制,不存在严重的代码可重入性问题.文章详细分析了 SHRTD 方法的指令级并行加速比,给出了发生地址冲突后的并行加速比与平均并行加速比以及发生地址冲突的依概率并行加速比.文章引入的三类理论加速比对指令级并行编译技术的研究和评测有重要的实际意义.

关 键 词 指令级并行性,超长指令字,指针别名,歧义相关性,加速比.

中图法分类号 TP302

当前的超长指令字(very-long instruction word,简称 VLIW)编译器都采用静态代码调度和软件流水的方法开发程序的指令级并行性(instruction-level parallelism,简称 ILP)[1].这两种方法最大的局限是存在内存访问的歧义相关性(ambiguous dependence)问题.编译器即使能够处理数组静态别名分析,也不能够很好地处理指针别名(pointer aliasing)分析.为解决指针别名问题以获得更高的隐含指令级并行处理加速比,文献[2]提出了两种运行时消除歧义性(run-time disambiguation,简称 RTD)的方法,即运行时检查(run-time check)方法和运行时补偿(run-time compensation)方法.将这两种方法应用于软件流水时,运行时补偿方法虽然允许不确定的内存访问,但它只适合那些可逆代码[2];运行时检查方法虽然适用于任何代码,但存在代码可重入性(rerollability)问题.这两种方法共同的缺陷是存在严重的代码空间问题,尤其是在全局软件流水中可能导致巨大的补偿代码空间开销.

因此,文献[3]提出了一种基于软硬件结合的运行时检查方法 SHRTD(software/hardware run-time disambiguation).SHRTD 的基本思想是:(1) 在检测到内存地址冲突时,为延迟不正确的 LOAD 操作及其后继操作,功能单元在歧义的 LOAD 操作之前插入 NOP 操作;(2) 为保证所有延迟操作执行顺序的一致性,编译时就确定执行 NOP 操作的所有功能单元的顺序和 NOP 操作的数目.

SHRTD 方法具有下述 3 个优势:(1) 因为运行时检查方法没有代码重做问题,所以它特别适合任何不可逆代码;(2) 因为任何 SHRTD 只需要一个 SHRTD 控制指令,补偿代码的代码空间并不大;(3) 不存在代码可重入性问题.然而,该文对 SHRTD 方法的分析不够完善,只针对具体两个例子给出了分析结果,没有产生完整的理论分析结论.本文在文献[3]的基础上对 SHRTD 方法展开更深入的研究,给出了 SHRTD 方法较完整的理论分析结果,得出了发生地址冲突的依概率并行加速比和发生地址冲突后的并行加速比与平均并行加速比的理论值.本文的结果对指令级并行编译技术的研究和评测具有重要的实际意义.

* 本文研究得到国家自然科学基金资助.作者乔林,1972 年生,博士生,主要研究领域为计算机并行编译技术,Petri 网,并行程序的形式语义.汤志忠,1946 年生,教授,主要研究领域为计算机并行体系结构,并行算法,并行编译技术.张赤红,1964 年生,副教授,主要研究领域为计算机并行算法,并行编译技术.苏伯珙,1938 年生,教授,博士生导师,主要研究领域为软件流水算法,并行编译技术.

本文通讯联系人:乔林,北京 100084,清华大学计算机科学与技术系

本文 1998-06-22 收到原稿,1998-11-06 收到修改稿

1 SHRTD 方法

SHRTD 方法假设:(1) 所有的操作都只占用一个时钟周期;(2) 所有的 PE 共享一个单一的内存片,且每个 PE 只有 1 个内存读单元、1 个内存写单元和 4 个循环控制单元.

1.1 硬件基本结构

一个完整的指令级并行计算机加速系统主要由主机、采用超标量体系结构的单处理机和采用 VLIW 体系结构的 8 个处理单元(PE)串联的多处理机三大部分组成.每个 PE 包含 11 个功能单元,即 2 个 ALU、2 个乘法器和 2 个内存访问端口和 4 个循环控制单元.该 VLIW 处理器能够在一个时钟周期中处理 4 个整数操作、2 个内存访问操作和 4 个分支操作.SHRTD 的硬件支持环境包括:(1) 在指令存储器上添加了一个大小为 $D\times W$ 的存储延迟操作的指令缓冲区,W 是 VLIW 指令的宽度,$D=d_{\max}+1$,这里,$d_{\max}$ 是大多数流行程序中的最大值;(2) 一个从指令缓冲区或正常的指令存储器选择操作的多路选择器 MUX,MUX 的数目等于 VLIW 指令字的操作域数目;(3)SHRTD 控制指令缓冲区和 SHRTD WORD 只读寄存器.

1.2 相关定义和定理

本节引入相关定义和定理,有关定理的证明请参阅文献[3].

定义 1(安放距离). 设 op_1 和 op_2 是程序中已安放的两个操作,且在原始串行代码中操作 op_1 在操作 op_2 之前.若安放后它们之间间隔的 VLIW 操作数目为 N,则这两个操作的安放距离

$$d(op_1,op_2)=\begin{cases}N+1, & \text{如果}op_1\text{在}op_2\text{之前};\\ 0, & \text{如果}op_1\text{和}op_2\text{在同一个VLIW中};\\ -N-1, & \text{如果}op_1\text{在}op_2\text{之后}.\end{cases} \tag{1}$$

定义 2(代码补偿量). 设 op_1 和 op_2 分别是程序中两个已安放的歧义 STORE 和 LOAD 操作,且它们的安放距离 $d(op_1,op_2)<0$.当检测到地址冲突时必须补偿一些空操作以延迟不正确的 LOAD 及其后继操作;我们称这些补偿的空操作数目为代码补偿量(code compensation measure).

定理 1. 若 op_1 和 op_2 安放在不同的内存端口,则相应的代码补偿量 $\Omega=|d(op_1,op_2)|+1$.

给定一个循环程序,其中的每一个操作在不同的迭代上都有一个安放位置,同一个操作在不同迭代上的安放位置是不同的.因此,下面我们定义体内安放距离和体间安放距离的概念,用来描述操作在不同迭代上的具体安放信息.

定义 3(体内安放距离和体间安放距离). 对任意一个迭代次数为 n 的循环中的操作 op_1 和 op_2,设 $op_1^{(k)}$和 $op_2^{(j)}$分别表示 op_1 和 op_2 的第 k 次和第 j 次迭代,$1\le j\le n$,$1\le k\le n$.如果 $j\ne k$,称安放距离 $d(op_1^{(k)},op_2^{(j)})$为体间安放距离;如果 $j=k$,称安放距离 $d(op_1^{(k)},op_2^{(j)})$为体内安放距离.考虑到操作 op_1 和 op_2 在循环体不同迭代的体内安放距离是相同的,故可将体内安放距离简记为 $d_{\text{inn}}(op_1,op_2)$.

循环程序的软件流水算法必须在循环调度前确定循环的体间启动间距 I,即相邻两次循环迭代的第 1 个操作之间的体间安放距离.一旦确定了循环体间启动间距 I,则有以下定理.

定理 2. 给定循环的体间启动间距 I.设 $op_1^{(k)}$和 $op_2^{(j)}$分别是循环程序中两个已安放的歧义 LOAD 和 STORE 操作,且它们的体内安放距离为 $d_{\text{inn}}(op_1,op_2)$,体间安放距离 $d(op_1^{(k)},op_2^{(j)})<0$,$j<k$.若 $op_1^{(k)}$和 $op_2^{(j)}$安放在不同的内存端口,则一次迭代需要插入的 SHRTD 操作个数 $p=\left\lfloor\dfrac{d_{\text{inn}}(op_1,op_2)}{I}\right\rfloor$.

定理 3. 给定循环的体间启动间距 I.设 $op_1^{(k)}$和 $op_2^{(j)}$分别是循环程序中两个已安放的歧义 LOAD 和 STORE 操作,当 SHRTD 检测到地址冲突时,相应的代码补偿量

$$\Omega=\left|d(op_1^{(k)},op_2^{(j)})\right|+1=d_{\text{inn}}(op_1,op_2)-(k-j)\times I+1. \tag{2}$$

2 SHRTD 方法的并行加速比分析

本节从理论上讨论 SHRTD 方法的并行加速比.因为代码并行执行时具有不确定性,要精确地分析最终代码的复杂度和代码空间大小是非常困难的,这里,我们使用概率论来分析 SHRTD 的并行加速比.

定理 4. 设循环程序的体间启动间距为 $I=1$,循环的串行代码总长度为 l,循环次数为 n.设 op_1 和 op_2 分别是循环程序中两个已安放的歧义 LOAD 和 STORE 操作,且体间安放距离 $d_{inn}(op_1,op_2)=d$,则某次发生 j_d 次体差为 d,j_{d-1} 次体差为 $d-1$,…,j_1 次体差为 1 的地址冲突后的并行程序加速比

$$S=\frac{l\times n}{n+2l-4+\sum_{x=1}^{d}j_x(d-x+1)}. \tag{3}$$

在一次迭代的过程中,发生一次地址冲突后的算术平均代码补偿量

$$\overline{\Omega(1)}=\frac{d+1}{2}, \tag{4}$$

发生 m 次地址冲突后的算术平均并行加速比

$$\overline{S(m)}=\frac{2l\times n}{2n+md+m+4l-8}. \tag{5}$$

证明:由定理 3 可知,发生体差为 x 的地址冲突时的代码补偿量 $\Omega_x=d-x+1,1\le x\le d$,则在某次发生 j_d 次体差为 d,j_{d-1} 次体差为 $d-1$,…,j_1 次体差为 1 的地址冲突后,总的代码补偿量 $\Omega=\sum_{x=1}^{d}j_x(d-x+1)$.当串行执行该程序时,总的时钟周期为 $l\times n$,并行执行时装入和排空阶段分别需要 $l-1$ 个时钟周期,在没有检测到地址冲突时,流水阶段需要 $n-2$ 个时钟周期.由于在运行时检测到地址冲突,则总的并行执行周期为

$$(n-2)+2(l-1)+\sum_{x=1}^{d}j_x(d-x+1)=n+2l-4+\sum_{x=1}^{d}j_x(d-x+1),$$

因而此时程序的并行程序加速比

$$S=\frac{l\times n}{n+2l-4+\sum_{x=1}^{d}j_x(d-x+1)}.$$

在一次迭代的过程中,发生一次地址冲突后的算术平均代码补偿量

$$\overline{\Omega(1)}=\frac{1}{d}\sum_{x=1}^{d}(d-x+1)=\frac{d+1}{2},$$

因而发生 m 次地址冲突后的算术平均并行加速比

$$\overline{S(m)}=\frac{l\times n}{n+2l-4+m\overline{\Omega(1)}}=\frac{2l\times n}{2n+md+m+4l-8}. \quad \square$$

例 1:设某个循环程序的长度 $l=6$,两个歧义 LOAD 操作和 STORE 操作的安放距离 $d=3$.产生一次地址冲突后的算术平均代码补偿量 $\overline{\Omega(1)}=\frac{d+1}{2}=2$,产生 m 次地址冲突后的算术平均代码补偿量 $\overline{\Omega(m)}=m\times\overline{\Omega(1)}=2m$.

当 $j_1=0,j_2=0,j_3=0$ 时,不存在任何地址冲突,该循环程序的并行加速比当 $n\to\infty$ 时的极限 $\lim_{n\to\infty}S=\lim_{n\to\infty}\frac{l\times n}{n+2l-4}=l=6$;当 $j_1=n,j_2=0,j_3=0$ 时,全部地址冲突的体差都为 1,加速比当 $n\to\infty$ 时的极限 $\lim_{n\to\infty}S=\lim_{n\to\infty}\frac{l\times n}{n+2l-4+3n}=\frac{l}{4}=1.5$;当 $j_1=0,j_2=n,j_3=0$ 时,全部地址冲突的体差都为 2,加速比当 $n\to\infty$ 时的极限 $\lim_{n\to\infty}S=\lim_{n\to\infty}\frac{l\times n}{n+2l-4+2n}=\frac{l}{3}=2$;当 $j_1=0,j_2=0,j_3=n$ 时,全部地址冲突的体差都为 3,加速比当 $n\to\infty$ 时的极限 $\lim_{n\to\infty}S=\lim_{n\to\infty}\frac{l\times n}{n+2l-4+n}=\frac{l}{2}=3$.这些结果与文献[3]的分析一致.

定理 5. 假设在不同迭代发生地址冲突的概率事件相互独立,且同一次迭代上不同体差的地址冲突的概率事件互斥,并设在一次循环迭代过程中发生体差为 d 的地址冲突的先验概率为 p_d,发生体差为 $d-1$ 的地址冲突的先验概率为 p_{d-1},…,发生体差为 1 的地址冲突的先验概率为 p_1,定理的其他条件同定理 4,则发生 m 次地址冲突的代码补偿量 $\Omega_p(m)$ 是 m 的概率的函数,且

$$\Omega_P(m)=\sum_{\substack{j_1+j_2+\ldots+j_d=m\\0\le j_1,j_2,\ldots,j_d\le m}}\left(\binom{n}{j_1,j_2,\ldots,j_d,n-m}\left(1-\sum_{i=1}^{d}p_i\right)^{n-m}\prod_{i=1}^{d}p_i^{j_i}\sum_{x=1}^{d}j_x(d-x+1)\right). \tag{6}$$

其中

$$\binom{n}{j_1,j_2,\ldots,j_d,n-\sum_{i=1}^{d}j_i}=\binom{n}{j_1}\binom{n-j_1}{j_2}\cdots\binom{n-\sum_{i=1}^{d-1}j_i}{j_d}. \tag{7}$$

此时的并行加速比依概率为

$$S_P(m)=\frac{l\times n}{\Omega_P(m)+n+2l-4}. \tag{8}$$

证明:注意到发生某次 j_d 次体差为 d,j_{d-1} 次体差为 $d-1$,…,j_1 次体差为 1 的地址冲突的概率为

$$\begin{aligned}P(j_1,j_2,\ldots,j_d)&=\binom{n}{j_1}p_1^{j_1}\binom{n-j_1}{j_2}p_2^{j_2}\cdots\binom{n-\sum_{i=1}^{d-1}j_i}{j_d}p_d^{j_d}\left(1-\sum_{i=1}^{d}p_i\right)^{n-\sum_{i=1}^{d}j_i}\\&=\binom{n}{j_1,j_2,\ldots,j_d,n-\sum_{i=1}^{d}j_i}\left(1-\sum_{i=1}^{d}p_i\right)^{n-\sum_{i=1}^{d}j_i}\prod_{i=1}^{d}p_i^{j_i},\end{aligned} \tag{9}$$

则发生 m 次地址冲突的概率

$$P_m=\sum_{\substack{j_1+j_2+\ldots+j_d=m\\0\le j_1,j_2,\ldots,j_d\le m}}P(j_1,j_2,\ldots,j_d)=\sum_{\substack{j_1+j_2+\ldots+j_d=m\\0\le j_1,j_2,\ldots,j_d\le m}}\left(\binom{n}{j_1,j_2,\ldots,j_d,n-m}\left(1-\sum_{i=1}^{d}p_i\right)^{n-m}\prod_{i=1}^{d}p_i^{j_i}\right). \tag{10}$$

因而发生 m 次地址冲突的代码补偿量

$$\begin{aligned}\Omega_P(m)&=\sum_{\substack{j_1+j_2+\ldots+j_d=m\\0\le j_1,j_2,\ldots,j_d\le m}}\left(P(j_1,j_2,\ldots,j_d)\sum_{x=1}^{d}j_x(d-x+1)\right)\\&=\sum_{\substack{j_1+j_2+\ldots+j_d=m\\0\le j_1,j_2,\ldots,j_d\le m}}\left(\binom{n}{j_1,j_2,\ldots,j_d,n-m}\left(1-\sum_{i=1}^{d}p_i\right)^{n-m}\prod_{i=1}^{d}p_i^{j_i}\sum_{x=1}^{d}j_x(d-x+1)\right).\end{aligned}$$

$\Omega_P(m)$ 显然是 m 的概率的函数.因此,发生 m 次地址冲突的并行加速比依概率为

$$S_P(m)=\frac{l\times n}{\Omega_P(m)+n+2l-4}$$

□

当发生不同的地址冲突时,程序的并行加速比并不相同.作为在程序执行前估计并行加速比的一种手段,依概率的并行加速比 $S_P(m)$ 意味着可以期望获得的并行加速比,它是衡量 SHRTD 方法有效性的重要参数.下面的例 2 显示了定理 4 和定理 5 之间的关系.

例 2:仍使用例 1 的参数.根据定理 5,发生 m 次地址冲突时的平均代码补偿量

$$\begin{aligned}\overline{\Omega_P(m)}=\frac{\Omega_P(m)}{P_m}&=\frac{\sum_{\substack{j_1+j_2+j_3=m\\0\le j_1,j_2,j_3\le m}}(3j_1+2j_2+j_3)\binom{n}{j_1,j_2,j_3,n-m}p^{j_1}q^{j_2}r^{j_3}(1-p-q-r)^{n-m}}{\binom{n}{m}(p+q+r)^m(1-p-q-r)^{n-m}}\\&=\frac{\sum_{\substack{j_1+j_2+j_3=m\\0\le j_1,j_2,j_3\le m}}(3j_1+2j_2+j_3)\binom{n}{j_1,j_2,j_3,n-m}p^{j_1}q^{j_2}r^{j_3}}{\binom{n}{m}(p+q+r)^m}.\end{aligned}$$

假设在一次迭代过程中不同体差的地址冲突为等概率事件,即 $p=q=r$,则

$$\sum_{\substack{j_1+j_2+j_3=m\\0\le j_1,j_2,j_3\le m}}(3j_1+2j_2+j_3)\binom{n}{j_1,j_2,j_3,n-m}p^m=\sum_{\substack{j_1+j_2+j_3=m\\0\le j_1,j_2,j_3\le m}}(j_1+2j_2+3j_3)\binom{n}{j_1,j_2,j_3,n-m}p^m,$$

故

$$\overline{\Omega_p(m)}=\frac{\Omega_p(m)}{P_m}=\frac{\sum_{\substack{j_1+j_2+j_3=m\\0\le j_1,j_2,j_3\le m}}(3j_1+2j_2+j_3)\binom{n}{j_1,j_2,j_3,n-m}p^m}{\binom{n}{m}(3p)^m}$$

$$=\frac{\sum_{\substack{j_1+j_2+j_3=m\\0\le j_1,j_2,j_3\le m}}((3j_1+2j_2+j_3)+(j_1+2j_2+3j_3))\binom{n}{j_1,j_2,j_3,n-m}p^m}{2\times\binom{n}{m}(3p)^m}$$

$$=\frac{2mp^m\times\sum_{\substack{j_1+j_2+j_3=m\\0\le j_1,j_2,j_3\le m}}\binom{n}{j_1,j_2,j_3,n-m}}{\binom{n}{m}(3p)^m}=\frac{2mp^m\times 3^m\binom{n}{m}}{\binom{n}{m}(3p)^m}=2m.$$

这个结果同样与使用定理4得到的结果以及文献[3]的分析一致.

引理1.

$$\sum_{m=1}^{n}m\binom{n}{m}x^m(1-x)^{n-m}=nx,\quad 0\le x\le 1. \tag{11}$$

证明:由 $m\binom{n}{m}=n\binom{n-1}{m-1}$,有

$$\sum_{m=1}^{n}m\binom{n}{m}x^m(1-x)^{n-m}=\sum_{m=1}^{n}n\binom{n-1}{m-1}x^m(1-x)^{n-m}=nx\sum_{m=1}^{n}\binom{n-1}{m-1}x^{m-1}(1-x)^{n-m}=nx(x+1-x)^{n-1}=nx.\quad\square$$

定理6. 定理的条件同定理5,则发生地址冲突后的平均代码补偿量

$$\overline{\Omega}=\frac{n\sum_{i=1}^{d}p_i(d-i+1)}{1-\left(1-\sum_{i=1}^{d}p_i\right)^n}. \tag{12}$$

发生地址冲突后的平均并行加速比

$$\overline{S}=\frac{l\times n}{\overline{\Omega}+n+2l-4}. \tag{13}$$

发生地址冲突后的平均并行加速比当 $n\to\infty$ 时依概率收敛于

$$\lim_{n\to\infty}\overline{S}=\lim_{n\to\infty}\frac{l\times n}{\overline{\Omega}+n+2l-4}=\frac{l}{1+\sum_{i=1}^{d}p_i(d-i+1)}. \tag{14}$$

证明:考虑到 P_m 事实上是可能发生或可能不发生地址冲突的一次 n 重 Bernoulli 实验,则有 $P_m=\binom{n}{m}\left(\sum_{i=1}^{d}p_i\right)^m\left(1-\sum_{i=1}^{d}p_i\right)^{n-m}$,即

$$\sum_{\substack{j_1+j_2+\ldots+j_d=m\\0\le j_1,j_2,\ldots,j_d\le m}}\left(\binom{n}{j_1,j_2,\ldots,j_d,n-m}\left(1-\sum_{i=1}^{d}p_i\right)^{n-m}\prod_{i=1}^{d}p_i^{j_i}\right)=\binom{n}{m}\left(\sum_{i=1}^{d}p_i\right)^m\left(1-\sum_{i=1}^{d}p_i\right)^{n-m}.$$

从而发生地址冲突的总代码补偿量依概率

$$\Omega_P=\sum_{m=1}^{n}\left(\binom{n}{m}\left(\sum_{i=1}^{d}p_i\right)^m\left(1-\sum_{i=1}^{d}p_i\right)^{n-m}\times\frac{m\sum_{i=1}^{d}p_i(d-i+1)}{\sum_{i=1}^{d}p_i}\right).$$

故由引理1,发生地址冲突后的平均代码补偿量

$$\Omega=\frac{\Omega_P}{\sum_{m=1}^{n}P_m}=\frac{\sum_{m=1}^{n}\left(\binom{n}{m}\left(\sum_{i=1}^{d}p_i\right)^m\left(1-\sum_{i=1}^{d}p_i\right)^{n-m}\times\frac{m\sum_{i=1}^{d}p_i(d-i+1)}{\sum_{i=1}^{d}p_i}\right)}{\sum_{m=1}^{n}\binom{n}{m}\left(\sum_{i=1}^{d}p_i\right)^m\left(1-\sum_{i=1}^{d}p_i\right)^{n-m}}$$

$$=\frac{\frac{d+1}{2}\sum_{m=1}^{n}\left(m\binom{n}{m}\left(\sum_{i=1}^{d}p_i\right)^m\left(1-\sum_{i=1}^{d}p_i\right)^{n-m}\right)}{1-\left(1-\sum_{i=1}^{d}p_i\right)^n}\times\frac{\sum_{i=1}^{d}p_i(d-i+1)}{\sum_{i=1}^{d}p_i}$$

$$=\frac{n\sum_{i=1}^{d}p_i(d-i+1)}{1-\left(1-\sum_{i=1}^{d}p_i\right)^n}.$$

发生地址冲突后的平均并行加速比则为 $\overline{S}=\frac{l\times n}{\Omega+n+2l-4}$.发生地址冲突后的平均并行加速比当 $n\to\infty$ 时依概率收敛于 $\lim_{n\to\infty}\overline{S}=\lim_{n\to\infty}\frac{l\times n}{\Omega+n+2l-4}=\frac{l}{1+\sum_{i=1}^{d}p_i(d-i+1)}$. □

此结果与概率有关,它是衡量SHRTD方法平均性能的重要参数.当地址冲突为0概率时,极限值显然为 l.

3 结　论

文献[3]提出的SHRTD方法采用软硬件结合的运行时消除指针别名歧义方法SHRTD,适用于不可逆代码,同时,它的代码空间受限,不存在严重的代码可重入性问题.

本文在此基础上对SHRTD方法展开更深入的研究,详细分析了SHRTD方法的指令级并行加速比,并给出了完整的理论分析结果.本文得到的发生地址冲突的依概率并行加速比和发生地址冲突后的并行加速比与平均并行加速比的理论值对指令级并行编译技术的研究和评测具有重要的实际意义.

参考文献

1 Rau B R, Fisher A. Instruction-level parallel processing: history, overview, and perspective. Journal of Supercomputing, 1993, 7(1):9~50

2 Nicolau A. Run-time disambiguation: coping with statically unpredictable dependencies. IEEE Transactions on Computers, 1989, 38(5):663~678

3 汤志忠,乔林,张赤红等.一种运行时消除指针别名歧义的新方法.软件学报,1999,10(7):685~689
(Tang Zhi-zhong, Qiao Lin, Zhang Chi-hong *et al*. A new run-time pointer aliasing disambiguation method. Journal of Software, 1999,10(7):685~689)

Analyzing Speedups of a Run-Time Pointer Aliasing Disambiguation Method

QIAO Lin[1] TANG Zhi-zhong[1] ZHANG Chi-hong[1] SU Bo-gong[2]

[1](*Department of Computer Science and Technology Tsinghua University Beijing* 100084)

[2](*Department of Computer Science William Paterson University USA*)

Abstract A new run-time pointer aliasing disambiguation method, SHRTD, combined with software and hardware techniques, can be used for irreversible code and has very limited compensation code space and no serious rerollability problem. In this paper, instruction-level parallel speedups of the SHRTD method are analyzed in detail. Speedups and mean speedups where address conflicts occurred and speedups where address conflicts will occur according to their probabilities are given. Three theoretical speedups introduced in this paper are very useful for studying and evaluating instruction-level parallel compiling techniques.

Key words Instruction-level parallelism, very-long instruction word, pointer aliasing, ambiguous dependence, speedup.