基于规范划分集的并行循环计算划分*

黄其军[1+], 杨建武[2], 余华山[1], 许卓群[1]

[1](北京大学 计算机科学技术系,北京 100871)

[2](北京大学 计算机科学技术研究所 文字信息处理国家重点实验室,北京 100871)

A Computation Partition Based on Uniform Partitioning Schemes for Parallel Loops

HUANG Qi-Jun[1+], YANG Jian-Wu[2], YU Hua-Shan[1], XU Zhuo-Qun[1]

[1](Department of Computer Science and Technology, Peking University, Beijing 100871, China)

[2](National Key Laboratory for Text Processing, Institute of Computer Science and Technology, Peking University, Beijing 100871, China)

+ Corresponding author: Phn: 86-10-51863509, E-mail: huangqj@ailab.pku.edu.cn

http://www.pku.edu.cn

Received 2001-11-19; Accepted 2002-05-13

Huang QJ, Yang JW, Yu HS, Xu ZQ. A computation partition based on uniform partitioning schemes for parallel loops. *Journal of Software*, 2003,14(3):362~368.

Abstract: Computation partition is one of the most important problems in parallel compilation and optimization. For dealing with parallel loops with determinated data distribution, a computation partition algorithm based on the subset of uniform schemes is proposed. The method of getting the subset of uniform schemes is given, as well as the algorithm of selecting the most optimized scheme under the consideration of communication and load balance. The experimental results prove that this algorithm is simpler and more effective than several previous algorithms in dealing with parallel loops, and the p_HPF compiler adopted by this algorithm can obtain good speedups and efficiencies. The compiler has been applied in the field of petroleum.

Key words: parallel loop; parallel compilation; computation partition; parallel computation; node program

摘 要: 计算划分问题是并行编译中最为重要的问题之一.针对并行循环,在数据分布确定的情况下,提出了基于规范集的计算划分算法,具体讨论了规范集的获取方法及综合通信与负载均衡的最优方案选取算法.实验表明,在并行循环处理方面,这一算法与以前几种算法相比更加简单、有效;采用这一算法的 p_HPF 编译器对数据并行应用问题可以获得良好的加速比和效率.该编译器已在石油领域得到应用.

关键词: 并行循环;并行编译;计算划分;并行计算;结点程序

中图法分类号: TP314 **文献标识码**: A

高级并行语言往往提供专门指导语句来表达并行循环的语义限制,以便使编译器获得并行优化.在数据并

* Supported by the National Natural Science Foundation of China under Grant No.60173004 (国家自然科学基金)

第一作者简介: 黄其军(1973－),男,安徽明光人,博士,主要研究领域为并行计算,并行编译,信息系统.

© 中国科学院软件研究所 http://www.jos.org.cn

行程序设计语言 HPF 中规定,可以将关键词 INDEPENDENT 加在 Do 循环前面,指出其后的 Do 循环是一个并行循环.

并行循环 L 可以严格定义如下:假设 $R(i)$表示第 i 次迭代所引用的变量集合,$W(i)$表示第 i 次迭代所更新的变量集合;如果对循环 L 的任意两次迭代 $i1$ 和 $i2(i1\neq i2)$,满足:

$$R(i1)\cap W(i2)=\varnothing \text{且 } W(i1)\cap R(i2)=\varnothing \text{且 } W(i1)\cap W(i2)=\varnothing,$$

则称 L 为并行循环.

$n(n\geq 1)$个具有不同循环控制变量的并行循环可以通过完全嵌套*组成 n 重并行循环,在下文中,$n(n\geq 1)$重并行循环统称为并行循环.

对于分布式内存多处理器体系结构,如果把并行循环迭代尽可能均匀地分配到不同处理器上来执行,那么循环内的并行性就可以被开发出来.如何进行这种分配称为计算划分问题.计算划分从两个方面影响并行程序效率:一方面,编译器将计算分配到多个处理器上并行计算,可以缩短整个程序的执行时间;另一方面,计算划分必须考虑各个结点计算所需的非本地数据,需要使用通信语句取到本地,通信将带来通信开销,使整个程序执行时间延长.最佳计算划分方案应该使整个循环在最短时间内完成.

国际并行研究领域从理论上提出幺模变换算法[1,2]、数据和计算分解加障碍消除算法[3]以及仿射计算划分算法[4,5]等循环并行化方法.针对并行循环,我们基于规范集的计算划分算法与这些算法相比更加简单、有效.

本文首先介绍了并行编译中计算划分问题的产生及计算划分的基本思路;接着提出了基于规范集的计算划分算法;最后给出了测试结果,使用 FFT 程序和 SPEC92 的 nasa7 基准测试程序检验了我们算法的效果,并指出进一步工作的设想.

1 基于规范集的计算划分思想

1.1 计算划分问题

1.1.1 问题的产生

HPF 程序的设计思路是:首先程序员根据程序计算模型的特点,把数据在处理器间进行分布;然后将注意力集中在程序的算法实现上,像单个程序那样直接地、全局性地操作所有数据,完成计算.HPF 程序员不必关心计算发生在哪个处理器上.计算与处理器的对应(即计算划分问题)由 HPF 编译器负责.

在科学研究性质的 HPF 编译器中,国际上具有代表性的有 RICE 大学的 dHPF,Stanford 大学的 SUIF 等;商业 HPF 编译器有 IBM 的 pHPF,PGI 的 pgHPF 等.这些编译器在并行循环处理的计算划分方面采用了不同的策略,可分为 3 类:第 1 类是拥有者计算原则,即根据数据元素的赋值语句,数据元素的新值由拥有该数据元素的处理器负责计算.这是一类简单策略,仅仅适用于相对整齐的循环,但一般说来优化性能不好.第 2 类是让循环内各个语句拥有各自的计算划分方案,各语句的计算划分方案共同构成整个循环的计算划分方案;dHPF 属于这一类,这是在较细粒度上开发并行性,意味着可能获得更大程度的并行;但其缺点在于无法评估负载均衡性(dHPF 不进行负载平衡)[6].第 3 类是循环内各语句拥有统一的计算划分方案,循环迭代被作为整体来分配,并行粒度大.我们提出的基于规范集的计算划分算法和 Stanford 大学的 SUIF 中使用的算法属于第 3 类,虽然在计算划分方案上丧失了某些一般性,但迭代被作为整体来分配,编译器可以通过比较各个结点所分配的迭代数来评估负载均衡性,进行负载平衡.

1.1.2 计算划分的基本思路

计算划分问题可理解为:对要处理的并行循环,怎样建造一个计算划分方案集(CPset),并且在 CPset 中,评估每个方案的优劣,选出最佳方案.即建造方案集合 CPset 和在 CPset 中挑选最佳方案.理论上讲,CPset 应该一般性地把各种可能的方案都包括在内,这样才能保证选出的方案是最优的.

* 完全嵌套是指内层循环是外层循环内惟一语句的循环嵌套形式.

© 中国科学院软件研究所 http://www.jos.org.cn

为了保持数据访问的规则性,我们一般不直接对计算空间进行划分,而是根据计算所涉及的数据(通常是数组)的位置,间接确定计算的位置,所以我们特别关注并行循环体中出现的数组访问.

例 1:

```
!HPF$ INDEPENDENT
      DO  I=2,99
!HPF$ INDEPENDENT
      DO  J=3,99
        A(I,J)=B(I+1,J-2)                    !S1
        D(I-1,J-1)=E(J,I)+F(A(I,J),c)        !S2
      END DO
      END DO
```

注:这是一个两重并行循环,其中 F 是函数调用,A,B,D 和 E 是数组,都按第 1 维在处理器间进行块(block)分布.c 为标量.

对于例 1,若把并行循环体中出现的数组访问组成的集合记为 K,则

$$K=\{A(I,J),B(I+1,J-2),D(I-1,J-1),E(I,J)\}.$$

集合 K 的元素可一般地记为 Ku.

1.2 规范集的构造

定义 1(方案全集). 假设并行循环 S 中有 n 条语句 $S1,S2,...,Sn,S$ 的数组访问集合 K 有 m 个元素,S 的方案全集 CPset^s 就有 m^n 个元素,每个元素是一个数组访问元组:

$$\mathrm{CPset}^s=\{\langle \mathrm{CP}1,\mathrm{CP}2,\ldots,\mathrm{CP}n\rangle|\mathrm{CP}i\in K,1\le i\le n\}.$$

各 CPi 分别对应语句 Si,作为 Si 的计算划分标准,使得 Si 的计算划分按照 CPi 的数据划分进行.

但是实际研究发现,CPset^s 中很多划分方案采用了"循环内各语句采用不同的计算划分标准"的策略,并不实用:

(1) 如果并行循环内各语句使用的计算划分标准不同,一次迭代的执行语句流被拆分到不同处理器上执行,由于存在迭代内数据相关性,势必造成循环内通信和同步.一方面,同步和通信语句开销大,放在循环内会导致结点程序效率大大降低;另一方面,结点间同步和相关数据通信代码生成复杂,特别是在多条语句之间存在复杂数据相关的时候.

(2) 在各语句使用不同的计算划分标准时,对某个处理器来说,每条语句分配到该处理器的迭代空间可能各不相同.必须插入必要条件语句屏蔽非本机迭代,这样,一方面引入新的条件语句带来额外开销,另一方面代码生成也较为复杂.

(3) 结点负载无法估算,无法比较各个方案的负载均衡性.由于迭代分割,每个处理器所分得的计算是不同的语句序列.比较负载就要估算各个语句的计算量,这在运行前是无法实现的.

为了克服上述缺点,我们将 Cpset 候选方案集加以挑选过滤,求得它的一个子集,形成规范集:让并行循环内的各个语句使用统一的计算划分标准,候选方案集的一个元素就是一个数组访问,而不是前面的一个数组访问元组;并行循环中的所有语句都以这个数组访问为标准进行计算划分.候选方案集由并行循环中出现的各个数组访问组成.我们把这样的候选方案集称为规范集,记为 CPset^k.

定义 2(规范集). 假设 K 为并行循环 S 中出现的数组访问集合,并行循环语句 S 的规范集 CPset^k 为

$$\mathrm{CPset}^k=\{\langle \mathrm{CP}\rangle|\mathrm{CP}\in K\}.$$

规范集 CPset^k 仅是方案全集的子集,基于规范集的优化计算划分只能得到近似最优解.但它的优点是显著的:

(1) 它将一次迭代作为一个整体分给某个处理器,而不对一次迭代本身进行分割,这使得迭代中的相关性被限制在本地,不会导致结点间的同步和通信.

(2) 并行循环内的语句可以共用一个条件语句来屏蔽非本机迭代.好处有:一方面,可以将起屏蔽作用的条件语句提到循环以外,降低开销.另一方面,从结点角度看,并行循环内各语句执行共同的一些迭代,这就可以让

© 中国科学院软件研究所 http://www.jos.org.cn

各结点分别求出各自的迭代集,直接控制循环体执行,消除冗余迭代,进一步提高效率.

(3) 可以通过比较各结点所分得的迭代数量来估算负载,考察负载均衡性,并在各个方案之间进行比较.

(4) 带来了结点程序生成的极大简化,结点程序代码简洁,效率高.

2 计算划分实现策略与算法

2.1 实现策略

通常在方案全集中进行最优方案的选取时仅估算通信开销,不能进行节点负载均衡的估算;但在基于规范集的最优方案选取时,可将负载均衡与通信开销进行综合考虑.

为在上述规范集 CPset 中挑选最佳方案(CP),可采用如下 3 个步骤:

首先,作为候选 CP 的数组访问在处理器间的分布情况反映了 CP 的负载均衡性,可以粗略地认为并行循环各迭代具有相等的计算量,所以 CP 分布的均匀性可视为 CP 的负载均衡性.在 p-HPF 编译器中,我们通过 CP 的结点最大迭代数来计算负载均衡性.一个候选 CP 在每个处理器上分配一定数量的循环迭代,在所有处理器中这个数量的最大值就是结点最大迭代数.结点最大迭代数越小,负载均衡性就越好.如果编译时无法求出所有 CP 的结点最大迭代数,则改用跨越处理器个数来计算负载均衡性.所谓"跨越处理器个数"就是作为 CP 的数组访问分布在几个处理器上.

其次,对 CPset 中的每个元素 CP,将并行循环中其他数组访问与它进行比较,检测通信需求,并从各数组访问的通信形式和通信数据量上估算通信开销,评估整个并行循环的通信开销.我们通过通信的类型和次数来估算通信开销.

最后,综合考虑每个 CP 的负载均衡性和通信开销,相互比较,选出最优 CP.这一步是难点,因为编译时不能把一次迭代的计算量用时间准确地表示出来,也不能把通信开销用时间准确地表示出来,无法在 CP 之间进行准确比较.我们采用抓住主要因素步步逼近的方法求出近似最优 CP.由大量实际测试结果可以看出,影响效率最关键的是 CP 带来的 remap 通信次数,其次是 CP 的负载均衡性,最后是 CP 的 shift 通信次数.算法按以上顺序,一次考虑一个方面,逐渐缩小 CPset,最后得到近似最佳 CP.

2.2 算法梗概

(1) 将并行循环中所有数组访问选出来,组成 CPset0.假设得到:

$$\text{CPset0}=\{A(I,J),B(I,J),B(I-1,J),C(I-1,J),D(I,J),E(I-1,J)\}.$$

(2) 分别以 CPset0 中的每个 CP 作为假想的计算划分标准,进行通信检测,统计带来的 remap 通信次数.假设得到:Remap_Num=⟨1,1,1,2,1,1⟩,其中每个数字与 CPset0 中相应位置的元素对应,表示其作为计算划分标准时带来的 remap 通信次数.

记下 remap 通信次数最少的 CP,组成 CPset1,则 $C(I-1,J)$被舍去,得到:

$$\text{CPset1}=\{A(I,J),B(I,J),B(I-1,J),D(I,J),E(I-1,J)\}.$$

(3) 对 CPset1 中的每个 CP,求出结点最大迭代数,挑出结点最大迭代数最小的 CP,组成 CPset2.如果在 CPset1 中存在编译时无法求出结点最大迭代数的 CP,则求出每个 CP 所跨越的处理器个数.假设求出跨越处理器个数为 Proc_Dim=⟨4,2,2,2,4⟩.

类似地,其中每个数字与 CPset1 中相应位置的元素对应,表示其跨越处理器的个数.

挑出所有跨越处理器个数最大的 CP,组成 CPset2,则得到 CPset2=$\{A(I,J),E(I-1,\mathrm{J})\}$.

(4) 分别以 CPset2 中的每个 CP 作为假想的计算划分标准,进行通信检测,统计有多少次 shift 通信.类似地,假设得到:Shift_Num=⟨2,3⟩

找出 shift 通信次数最少的 CP,组成 CPset3,则得到 CPset3=$\{A(I,J)\}$.

(5) 在 CPset3 中任选一个 CP 作为计算划分标准,则得到 Result=$A(I,J)$.

© 中国科学院软件研究所 http://www.jos.org.cn

3 测试和结论

3.1 规范集效果测试

为了测试规范集带来的并行效果,我们搜集了 10 个程序,对这些程序分别用方案全集和规范集进行计算划分方案的优化选取,并进行比较.

所选例程代表了实际计算中对并行循环的各种使用,来源有 3 个方面:(1) HPF 编译器的标准测试程序 HPFBench[7]和 NAS Parkbench[8];(2) 常见计算,比如二维傅里叶变换、矩阵相乘等;(3) HPF 语言规范.

结果发现,对于所有程序,基于规范集所确定的最终计算划分方案与基于方案全集相同.也就是说,规范集对这 10 个程序都能达到最优.结果见表 1.

Table 1 Testing results of the uniform partitioning set

表 1 规范集测试结果

Program	Program source	Effect of uniform partitioning set
Fft	2D Fourier transform	Best
Matrixmul	Matrix multiply	Best
Hough	Hough transform	Best
Gmo	Generalized moveout seismic kernel	Best
Mdcell	Molecular dynamics code	Best
Sweepx	BT application	Best
Spx	SP application	Best
Exam	HPF language specification	Best
Jaccobi	Jaccobi iteration	Best
Explicit_Hydrodynamics	Hydrodynamics	Best

实践证明,使用规范集在绝大多数情况下都能挑选出最优或近似最优的计算划分方案,少数情况下的有限效率降低是可以忽略的.

3.2 算法性能测试

为了检验本文提出的计算划分算法对并行循环的处理效果,实验中我们对 HPFBench[7]和 NAS PARKBench[8]中出现的所有并行循环都进行了测试,获得了很好的效果.下面以二维傅里叶变换程序 FFT 和 SPEC92 的 nasa7 标准测试程序 Cholesky 为例进行讨论.对算法的评价应当包括适应性和效率两个方面,本文提出的算法是针对并行循环的,故不必讨论其适应性问题,为此,对它的评价就集中于它的加速比与效率上.由于受篇幅限制,本文不再给出编译变换后的结点程序,有关细节请参阅文献[9].

测试 FFT 所使用的平台是曙光 2000,由 32 个计算结点组成,每个结点配置了 PowerPC 640e 处理器,256M 本地内存和硬盘.并行计算的通信采用 WRC 连接的高速 Mesh 网进行.操作系统为 AIX V4.2,消息通信使用 MPICH 1.1.如图 1 和图 2 所示.

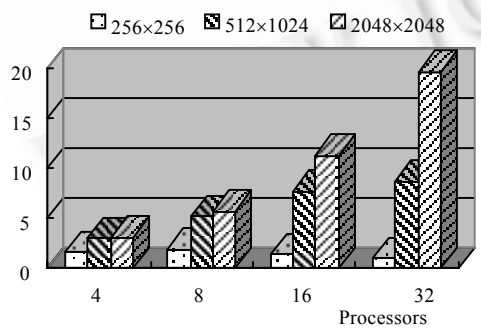

Fig.1 Speedup of FFT program

图 1 FFT 程序加速比图例

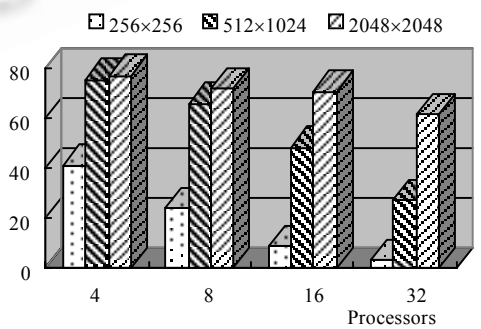

Fig.2 Efficiency of FFT program

图 2 FFT 程序效率图例

以上测试结果表明,在处理器个数一定的情况下,小数据量加速比较小,效率不明显.这是因为在计算量较小时,通信和运行支持函数开销相对较大.当数据达到一定规模时,加速比较高,效率达到 70%左右,并行编译获

© 中国科学院软件研究所 http://www.jos.org.cn

得了良好的并行效果.同时,也表明本文算法找到的编译方案具有很好的效率.对于相当规模的数据量,加速比随处理器个数的增加而显著提高,比如对 2048×2048 的数据,当处理器数为 4,8,16,32 时,加速比分别达到 3.09,5.76,11.30,19.65.表明算法获得的编译方案具有良好的可扩展性.

为了把我们的算法与其他算法进行对比,我们选择 SPEC92 的 nasa7 标准测试程序 Cholesky 进行测试,并与文献[4,5]中的测试结果进行比较.代码由 18 个不规则循环嵌套组成,其中一些计算的嵌套深度达到 4,在文献[4]中作为并行化的难点给出.

测试结果表明,我们获得了很高的加速比,8 处理器达到 7.0.文献[4]中提出的仿射计算划分(affine partition)算法,通过幺模变换、循环融合以及循环交换和融合交替使用等技术,达到与我们相类似的划分方案,加速比略低于我们的方法,8 处理器达到 6.2.相比之下,我们的算法更简单、更有效.改进的仿射计算划分算法对该例的处理结果在文献[5]中已给出,加速比没有太大变化,8 处理器达到 6.3.

其他算法,如幺模变换法(unimodular transform)[1,2]或数据计算分解加障碍同步消除法[3]等对上例的处理都不理想,加速比很低.比如幺模变换法将对程序中 L10 循环进行并行化,因为它是最外层并行循环,但实际上这个方案带来很大的通信开销;对 Cholesky,幺模变换法不能发现最优划分方案.

为了进行比较,我们给出以下 4 种算法对 Cholesky 进行测试的结果:

(1) 幺模变换法;

(2) Anderson 和 Lam 的数据计算分解加障碍同步消除法;

(3) 仿射计算划分(affine partition)算法;

(4) 我们的基于规范集的计算划分算法.

前 3 种算法的测试结果由 Stanford 大学计算机系统实验室在文献[4]给出,测试平台是 Digital Turbolaser,使用 8 个 300-Mhz 的 21164 处理器.我们的测试平台是 100M 以太网连接的 8 个 PC 机,结点机使用 P III 500 处理器.测试平台的不同是因为我们无法找到同样的平台,但测试结果还是具有可比性的:从程序本身来看,我们的计算划分达到了无通信的效果,使得测试结果与并行环境的通信性能无关,这样,平台的差异就表现为处理器处理速度的不同,而处理器的处理速度基本上并不影响程序的加速比.测试结果如图 3 所示.

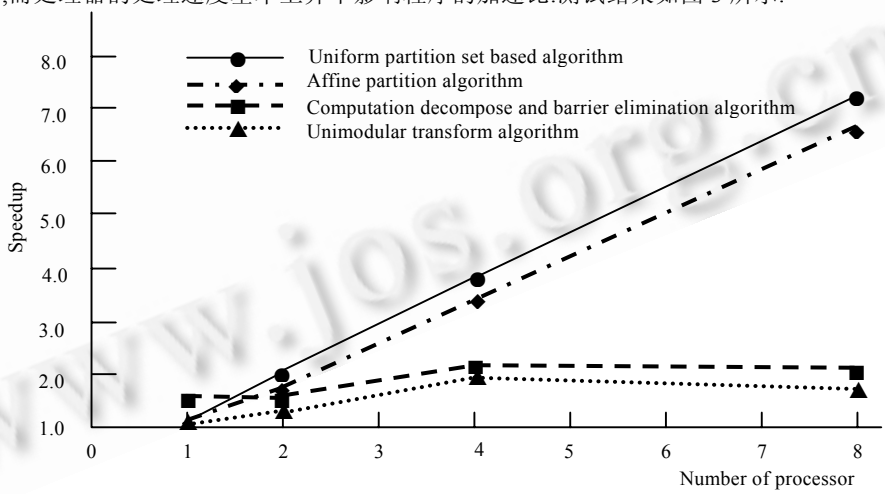

Fig.3 Performance for Cholesky in the SPEC92 nasa7 benchmark

图 3 SPEC92 的 nasa7 标准测试程序 Cholesky 性能

以上对几种算法的测试是就并行循环这种特殊循环来说的,虽然我们的算法在适应性上不如其他几种算法,但针对并行循环问题,我们的算法具有更好的效果,而且,其他 3 种算法只是理论上提出来的,并未真正或完全实现,如仿射计算划分算法只在 SUIF 中实现了基本算法[4],而我们的算法在 p_HPF 中已得到完整的实现.

© 中国科学院软件研究所 http://www.jos.org.cn

4 结束语

为了处理并行循环,我们在理论分析和实践的基础上提出了基于规范集的计算划分算法,具体给出了规范集的获取法及综合通信与负载均衡的最优方案选取算法.测试表明,在并行循环处理方面,该算法与其他几种算法相比更加简单、有效.采用该算法的 p_HPF 编译器不但效率高,而且具有相当强的并行处理能力,可以处理实际应用中复杂的并行计算需求.在石油应用并行化工作中,基于规范集的计算划分算法所实现的并行循环发挥了很大作用,是获得并行效率的关键.今后,我们将在此基础上实现并行循环对非规则数组的支持,进一步满足实际应用的需求.

References:

[1] Banerjee U. Unimodular transformations if double loops. In: Proceedings of the 3rd Workshop on Languages and Compilers for Parallel Computing. 1990. 192~219.

[2] Banerjee U. Loop Transformations for Restructuring Compilers. Norwell: Kluwer Academic Publishers, 1993.

[3] Anderson JM, Lam MS. Global optimizations for parallelism and locality on scalable parallel machines. In: Proceedings of the ACM SIGPLAN'93 Conference on Programming Language Design and Implementation. 1993. 112~125.

[4] Lim AW, Cheong GI, Lam MS. An affine partitioning algorithm to maximize parallelism and minimize communication. In: Proceedings of the 13th ACM SIGARCH International Conference on Supercomputing. 1999. 228~237.

[5] Lim AW, Liao S-W, Lam MS. Blocking and array contraction across arbitrarily nested loops using affine partitioning. ACM SIGPLAN Notices, 2001,36(7):103~112.

[6] http://www.cs.rice.edu/~dsystem/dhpf/dhpf-overview-96/index.html. 1995.

[7] Hu CY, Jin GH, Johnsson SL, Kehagias D, Shalaby N. HPFBench: a hign performance Fortran Benchmark suite. ACM Transactions on Mathematical Software, 2000,26(1):99~149.

[8] CRPC/HPFF/benchmarks/index.cfm. 1995.

[9] Thirumalai A. Code generation and optimization for high performance Fortran [MS. Thesis]. Department of Electrical and Computer Engineering, Louisiana State University, 1995.

[10] Huang QJ. Parallel loop and its compiling and optimizing techniques [Ph.D. Thesis]. Beijing: Peking University, 2002 (in Chinese with English Abstract).

附中文参考文献:

[10] 黄其军.并行循环及其编译优化技术[博士学位论文].北京:北京大学,2002.

© 中国科学院软件研究所 http://www.jos.org.cn