国防科技大学学报
JOURNAL OF NATIONAL UNIVERSITY OF DEFENSE TECHNOLOGY
第13卷第4期 1991年12月 Vol. 13 No. 4

多处理机系统的微任务并行

沈志宇 廖湘科

（电子计算机系）

摘 要 本文讨论微任务技术及其在多机系统上的实现。微任务技术使得一个程序能在循环和语句块一级并行执行，它的显著特点是系统开销较小、效率较高，适应于小粒度并行。

关键词 多处理机系统，多任务，微任务，控制结构，预处理

分类号 TP311

采用多处理机系统结构是超级计算机发展的趋势。多机系统给软件工作者提出了许多新的研究课题，解决好这些课题是充分发挥多机系统性能的关键。多任务技术就是一个重要的课题。多任务（Multitasking）是一种在用户作业地址空间内创建多个非同步的任务，使得一个作业的多个部分能在多处理机系统上并行执行的技术。多任务技术有两种基本的实现方法：宏任务（Macrotasking）方法和微任务（microtasking）方法。宏任务实现的是子程序一级的并行，即大粒度并行。它是由用户将一个程序划分成可以并行执行的多个部分后，在源程序中加入对宏任务库子程序的调用来实现的。宏任务库是一组供用户在语言一级调用的子程序，它提供了并行实体的创建、同步、互斥等功能，用户使用它来组织自己的并行任务[1]。

微任务实现的是循环和语句块一级的并行，即小粒度并行。它是由用户识别出程序中的并行结构后，在源程序中加入微任务预处理器指令来实现的。预处理器指令取FORTRAN程序的注解行的形式。预处理器在编译之前将它们转换成对微任务库子程序的调用，并对程序作适当改写。微任务库由一组实现多处理机的调度、同步、互斥等功能的子程序构成。

美国CRAY公司于1985年底在CRAY X-MP系统上实现了微任务，目前各类CRAY多机系统都支持微任务。这几年来，美国的Alliant Fx系列、Sequent、VAX6000系列、日本的SX-3等都先后实现了微任务。我们对这项技术的研究也已经取得了较好的进展。本文讨论微任务技术的基本概念、功能特性及其在共享存储器多处理机系统上的实现。

* 1990年5月18日收稿

1 微任务技术概述

微任务技术的实现是由微任务预处理器和微任务库来完成的。用户用预处理器来处理插入了预处理器指令的微任务化程序。预处理器将指令转换成对微任务库子程序的调用，并根据微任务化的要求对用户程序作适当的改写。预处理后的微任务化程序可以同FORTRAN 程序一样，由编译程序进行编译。装配程序将用户程序的半目标代码和有关微任务库子程序的半目标代码装配在一起。用户程序运行时，被调用的微任务库子程序将完成处理机的调度、同步、互斥等功能，实现用户程序的并行执行。

介绍几个基本概念。

(1) 共享变量与私有变量：在微任务技术中，变量分为共享的和私有的。共享变量是出现在 COMMON、SAVE、DATA 语句及子程序调用参数表中的变量，它们为执行该程序的所有处理机所共享（有相同的存储单元）。其余的变量则为私有变量，对执行该程序的每个处理机，它们有不同的存储单元，分配在各个处理机的栈中，即每个处理机有它自己的私有变量副本。

(2) 控制结构：控制结构是微任务描述并行的唯一手段，由一个或多个可并行执行的进程组成。控制结构给所有进入该结构的处理机分配工作；在该结构内的所有工作完成以前，禁止任何处理机离开该结构；在该结构内的所有工作完成之后，禁止任何处理机再进入该结构。用户使用微任务预处理器指令来定义控制结构。

(3) 进程：一个进程仅由一个处理机来执行。进程是为处理机分配工作的单位。在一个控制结构中的多个进程可由多个处理机并行执行。

(4) 微任务化子程序：一个微任务化子程序含有一个或多个控制结构，可由多个处理机并行执行。当一个处理机调用一个微任务化子程序时，它激活其它空闲处理机来参与执行该子程序；微任务化子程序返回之前，它再将其他处理机置为空闲状态。

(5) 微任务化程序：一个微任务化程序由一个主程序段、若干个微任务化子程序段和若干个非微任务化辅程序段（亦可以没有）构成。

微任务可以将 DOALL 循环和并发语句块并行化。DOALL 循环是不存在跨迭代依赖关系的循环，它的各个迭代可以并行执行。语句块由一定数量的语句构成。并发语句块是几个互相之间不存在依赖关系的语句块，可以并行执行。(依赖关系的概念参阅文 2)。DOALL 循环和并发语句块的例子如下所示：

例 1　DOALL 循环

```
DO 10 =1,100
  A(I)=B(I)+X
  C(I)=D(I)-A(I)
10 CONTINUE
```

例 2　并发语句块

```
S=SIN(A)+X   }
T=S+Y        }  语句块 1
U=COS(A)+X   }
V=U-Y        }  语句块 2
```

微任务的控制结构不允许嵌套，即不允许在并行中再创建并行。微任务只提供一种隐式同步，即通过控制结构来同步控制结构中所有进程的完成。微任务提供一种临界区方式的互斥，这种互斥也不能嵌套。微任务只支持 DOALL 循环和并发语句块这两种并行结构。由于微任务的功能简单明确、针对性强，我们可以采用一些特殊技术来实现它，从

而达到减少开销、提高效率及适应小粒度并行的目的。由于预处理器指令采用了注解行的形式，微任务化程序保持了良好的可移植性，用户对程序进行微任务化也较为方便。由于微任务实现的是循环和语句块一级的并行，用户对程序进行微任务化时只需在较小范围内进行依赖关系分析，对源程序的修改也较少。

2 微任务预处理器指令

程序员用微任务预处理器指令将程序微任务化。预处理器根据这些指令来对微任务化程序进行预处理。预处理器指令均以 CMIC $ 开始，基本的指令介绍如下：

GETCPUS 为微任务化程序获取多个处理机；

RELCPUS 释放由 GETCPUS 获得的处理机；

GUARD 定义一个临界区的开始；

END GUARD 定义一个临界区的结束

MICRO　　标志一个微任务化子程序

PROCESS　　定义一个 PROCESS 控制结构的开始，并定义该结构中第一个进程的开始

ALSO PROCESS 仅用于 PROCESS 控制结构内，定义上一进程的结束和下一进程的开始

END PROCESS　定义 PROCESS 控制结构内最后一个进程的结束及整个控制结构的结束

DO GLOBAL　　定义一个 DO GLOBAL 控制结构，该结构包含一个循环，循环中的每个迭代为一个进程，所有进程可以并行执行

CONTINUE　　让所有处理机继续并行执行下一个微任务化子程序

一个微任务化子程序以 MICRO 指令开始。对该子程序的一次调用导致所有可用处理机参与执行该子程序。在微任务化子程序内部，每个处理机在控制结构的协调下并行地工作。控制结构分为两种。一种是用 DO GLOBAL 指令来指明的，该指令出现在一个 DOALL 循环之前，指明该循环的每个迭代都是一个独立进程，并且这些进程可以并行执行。

例 3　DO GLOBAL 控制结构

```
CMIC $ DO GLOBAL
       DO 100  I=1, N
       DO 100  I=1, N
         Loop body
       100  CONTINUE
```

例 4　PROCESS 控制结构

```
CMIC $      PROCESS
            segment 1
CMIC $      ALSO  PROCESS
            segment 2
CMIC $      END  PROCESS
```

另一种控制结构是用 PROCESS、ALSO PROCESS 和 END PROCESS 指令来指明的。当不出现 ALSO PROCESS 时，PROCESS 和 END PROCESS 指明一个单进程结构，该进程仅由一个处理机执行，其它到达该进程起点的处理机要等到它完成后才能开始执行下面的程序。当出现 ALSO PROCESS 时，定义的是一种多进程结构，各进程可并行执行。

3 微任务的实现

传统的处理机的调度均由操作系统完成。这导致了较大的系统开销。为了减少开销，微任务采用了处理机自调度的方法[3]，同样，为了减少开销，微任务没有采用传统的由任

务请求分配处理机的方法，而是采用了由处理机自动获取任务去执行的方法[4]。

微任务库是微任务技术的基础，它完成处理机的申请、释放、调度、同步、互斥等功能，组织多个处理机并行执行一个微任务化程序。微任务库设计的首要目标是高效，以适应小粒度的并行任务。其次它应允许参与执行一个微任务化程序的处理机的数目动态变化，以适应批处理环境。为了实现上述目标，我们在微任务库中没有设立明显的并行实体控制机构，没有引入并行实体的状态和状态转换，而是直接用硬件信号灯来实现处理机的调度、同步、互斥，尽量用共享寄存器来存放控制数据结构。这些措施保证了微任务库的高效。另外，微任务库采用的处理机自调度方法使得参与执行一个程序的处理机数可以动态变化。

微任务并行是通过控制可用处理机对程序的数据和指令的访问来实现的。对数据访问的控制要求确定每一个变量是共享变量还是私有变量。共享变量存储于共享存储器中，能为每个处理机访问。私有变量则存储于每个处理机的栈，只能为本处理机所访问。对指令访问的控制是由用户定义的控制结构决定的。在一个微任务化子程序中，控制结构内的指令由微任务库子程序按一定的规则调度各个处理机来并行执行，不在任何控制结构内的指令则由各个处理机冗余地执行。

4 结束语

微任务技术的发展方向是自动任务化。CRAY 公司于1989年推出的自动任务化，是以微任务技术为基础的。我们在开展任务技术研究的同时开展了自动任务化的研究。随着多机系统的系统软件和应用软件研究的深入，多机系统性能一定能得到更充分的发挥。

参 考 文 献

[1] 廖湘科，陈立杰. 多处理机系统中的宏任务并行，国防科技大学学报，1989，(4)：25～32

[2] Utpal Banerjee. Dependence Analysis for Supercomputing. Kluwer Academic Publishers, 1988

[3] Tang P. and Yew P C. Processor Self—Scheduling for Multiple—Nested Parallel loops. Proc. of the Intl Conf on Parallel processing, 1986

[4] Polychronopoulos C D. Parallel Programming and Compilers. Kluwer Academic Publishers, 1988

Microtasking Parallel on Multiprocessor Systems

Shen Zhiyu Liao Xiangke

(Department of Computer Science)

Abstract

Microtasking technique and its implementation on multiprocessor systems are discussed in this paper. The loops and statement blocks of a program can be parallely executed on a multiprocessor system by microtasking technique . The distinctive features of microtasking are low overhead, high efficiency , and its applicability in fine grain parallel tasks.

Key words multiprocessor system, multitasking, microtasking, control structure, preprocessing