第 6 卷第 8 期
1995 年 8 月

软 件 学 报
JOURNAL OF SOFTWARE

Vol. 6, No. 8
Aug. 1995

基于 TRANSPUTER 网络的函数式语言的并行抽象机模型*

袁 伟　　孙永强

（上海交通大学计算机科学与工程系，上海 200030）

摘要　本文基于函数式语言的并行 LE 计值语义，提出了一个针对 TRANSPUTER 网络的函数式语言的并行抽象机模型——并行 LE MACHINE. 在该抽象机中由于 LE 计值分析方法的采用减少了函数式语言的惰性语义所带来的开销，并且在并行性开发中引入惰性计值的思想在保证充分利用系统中的并行处理能力的同时，基于 THREAD 的惰性进程生成法减少了一些不必要的并行性开发所带来的开销. 双向链表的栈模型亦对并行进程的实现效率有较大的提高，原型系统的测试显示系统具有较高的执行效率.

关键词　函数式语言，并行处理，抽象机模型.

函数式语言实现的困难来自函数式语言在设计时注重语言的表达能力和语义的简洁，而很少考虑当前计算机硬件技术的约束. 这与传统语言 C、PASCAL、ADA 等语言有着巨大的区别，函数式语言已远远超出了传统硬件技术的发展，在函数式语言和当今计算机硬件之间存在着很大的语义差距. 有关函数式语言实现的研究分为两个方向：一是基于当前冯·诺依曼体系结构，采用各种优化技术在传统顺序处理机上高效实现函数式语言；另一方向则研究新的体系结构来高效实现函数式语言.

本文在吸取了一些早期系统的优点（如 TURNER'S G－MACHINE）和近年来的研究工作[1-3]的基础上，提出了基于 TRANSPUTER 网络的并行实现函数式语言的抽象机模型——并行 LE－MACHINE. 由于一个抽象机模型的设计涉及两个方面：一是抽象机模型必须能较好地支持函数式语言的惰值计值语义；二是抽象机模型必须能较易映射到具体的处理机系统中，以便获得高效的实现. 本文的第 1 节介绍函数式语言的并行 LE 计值语义；第 2 节介绍在 TRANSPUTER 网络上的抽象机的实现；文章的最后给出了部分 BENCHMARK 程序的运行结果.

1　并行 LE 计值

当前函数式语言的研究重点在于如何从惰性的函数式语言中获取足够的并行性，严格

* 本文 1992－07－14 收到，1994－05－16 定稿

本研究课题得到国家教委博士基金和国家自然科学基金的资助. 作者**袁伟**，1972 年生，博士生，主要研究领域为并行处理，函数式语言. **孙永强**，1931 年生，教授，博士导师，主要研究领域为并行处理，函数式语言，计算理论.

本文通讯联系人：袁伟，上海 200030，上海交通大学计算机科学与工程系

性分析技术的作用非常明显.LE计值的严格性分析的基本思想是:仅对可能出现递归调用序列的对象的计值采用延迟调用策略,而对不会引起无限计值的对象,如原始函数、非递归定义函数等归为积极计值的范围.这样一方面减少了惰性计值的开销,同时为开发程序中的并行性开拓了更广的前提.在具体处理中,即使是采用延迟计值的对象,在必要时还可采用函数半展开方法在源代码级提高程序中的积极计值程度.在文献[1,2]中,已证明LE计值的计值能力介于惰性计值和积极计值之间.LE计值能力虽然弱于惰性计值,但是由于LE计值仍保留了对无穷数据结构(如流STREAM)和高阶函数的处理能力,所以其计值能力仍是强大,同时由于执行效率的显著提高赋予LE计值以强大的生命力.

HUDAK/GLODDBERG[4]在1985年提出了一种函数式语言自动并行化的算法——SERIAL COMBINATOR.其后的大多数系统如〈V,G〉MACHINE[5]和HDG—MAHCINE[6]均采用类似的思想.在这些方法都是采用纯静态的方法进行程序中颗粒度的划分,由于并行处理系统中影响程序并行执行效率的主要因素,如进程调度,负载均衡以及进程通讯,更主要的函数式语言程序内部的并行性都表现为一种动态的行为,显然颗粒度的动态控制势在必行,这也是以往系统实现效率低的原因之一.Eric Mohr[7]曾提出过一种动态的颗粒度控制算法,但是其算法仅适合于共享内存多处理机系统,用于类似TRANSPUTER这类分布式内存将带来巨大的通讯开销;而由于分布式内存的体系结构的多处理机系统具有很强的扩充能力,因此我们提出了一种结合上述LE计值语义的THREAD惰性进程生成法[3,8].

THREAD惰性进程生成法基于在函数式语言中进程的生成往往对应于函数的作用这一特点,因此延迟该部分函数作用即可达到延迟进程生成的效果.但是这种延迟计值不同于原LE计值中的延迟对象的计值,该延迟计值的驱动可能来自于计值驱动,即由父进程通过动态增加颗粒度来直接对该部分任务进行计算;另一驱动可能来自于多处理机的负载调度的需求,即生成该延迟计值部分所对应的子进程利用系统中的并行处理能力加快程序的执行效率.为与LE计值的"延迟对象"的计值相区别,我们在文献[1]LE计值的对象集中,增加了一类特殊的对象——伪延迟对象来表示这一类特殊的延迟计值,从而形成了函数式语言的并行LE计值[3,8].

并行LE计值仅是对原LE计值的并行化,即用THREAD惰性进程生成法来动态控制进程的粒度;因而并行LE计值的计值能力等价于原LE计值(详细的论述请参阅文献[8]).

2 并行抽象机——PARALLEL LE—MACHINE

2.1 数据表示和栈模型

函数式语言的实现大多数基于图归约模型,图归约模型的最大特点就是非常适合于并行处理,但是其缺点是实现中存在过多的图结点的构造、回收和REDEX的寻找开销.G—MACHINE的突出贡献在于将函数调用时图的构造过程采用编译的方法用对应于函数的代码来直接生成程序图,减少了程序图的简单复制时的解释开销从而极大地提高了程序的执行效率.在其后的SPINELESS G—MACHINE[9]和SPINELESS TAGELESS G—MACHINE中针对G—MACHINE中的图的重写开销和REDEX的寻找开销进行了进一步

的优化,但是在这些系统的归约过程中仍然存在很多图空间的管理开销. LE语义模型[1-3]中采用的是一种比惰性计值更为积极的LE计值策略,由于在LE计值中引入了"延迟对象"和"伪延迟对象"来处理函数式语言并行处理过程中的延迟计值和并行粒度大小,并且将这些处理溶于语言的基本函数的语义中,从而各种处理都统一为函数的作用形式. LE语义模型的这一特点使得我们可以在函数调用时采用类似于传统语言(如C语言)的传值的调用形式,这样既避免了以往函数式语言实现中的图的构造开销同时也蕴含地免除了显式的REDEX的寻找开销;从而与传统实现相比可显著提高系统的处理速度. 这从我们的实现原型系统的测试结构中可以得到验证(见第3节). 但是"延迟对象"和函数的部分作用的CAP的实现仍有部分图的构造开销.

A. 数据表示

在LE—MACHINE中,各类数据均采用了带标记的数据结构(这与传统实现类似),如下所示:

为了减少系统的运行开销,对于原始数据结构如整数、浮点数、布尔类型和字符等直接存放在归约系统的寄存器和栈中,仅对表数据结构等结构数据采用指针形式间接存访.

B. 栈的模型

由于程序运行中将每个函数调用的活跃记录都保存在进程的栈中,因此栈作为进程一部分状态. 顺序处理机中采用连续的地址空间来实现进程的栈,而并行处理机中由于进程需频繁地切换,所以在并行处理系统中栈也需在系统的堆中分配. 在LE—MACHINE中我们采用双向链接表的形式来实现栈,其中每个数据块对应为函数调用的活跃记录和函数计值时所需的局部临时空间并且由一个指针指向当前的活跃记录,这样进程切换时仅需涉及该指针的保存和恢复. 栈的形式如图1所示为每次函数调用的FRAM的双向链接表:

图1 STACK—FRAME机制

栈的每个数据块的大小的确定可以在编译过程中静态的决定,即根据编译时的栈的变化即可确定每个函数调用的最大的空间开销,或者采用LESTER[9]的STACKLESS编译技术亦可得到更为详尽的分析结果.

在HDG—MACHINE和(V,G)—MACHINE均采用单向链接表的形式来实现归约过程中的栈,这类实现在运行中的函数进入和函数返回时涉及的栈数据块的管理均需堆的空间管理,并且涉及单向链接表的插入和删除,与传统顺序机的栈的实现相比这一实现方式极大地降低了系统的执行效率. 在LE—MACHINE中,基于这样一种观察:函数返回时需释放一个FRAME的数据空间而在下一次函数调用时又需申请一个新的FRAME数据块,显然原来释放的FRAME数据块可直接用于下一次函数调用. 基于上述观察LE—MACHIEN中采用一种新的管理策略:即在系统中始终维持一个空闲栈的FRAME数据块的双向链表,并且这一链表与当前运行态进程栈的双向链表相联;这样在函数调用和返回时仅需移动栈的指针FP. 采用该方式避免了以往系统中函数调用所涉及的堆的管理和链接表

的维护开销,仅在系统中的空闲FRAME数据块耗尽时才涉及堆的管理来分配一组新的FRAM数据块.唯一增加的开销在于就绪态进程进入运行态时进程栈队列和空闲栈队列的联结和运行态进程进入挂起状态时该进程栈队列和空闲栈队列的链接和分离.运行状态过程如图所示:

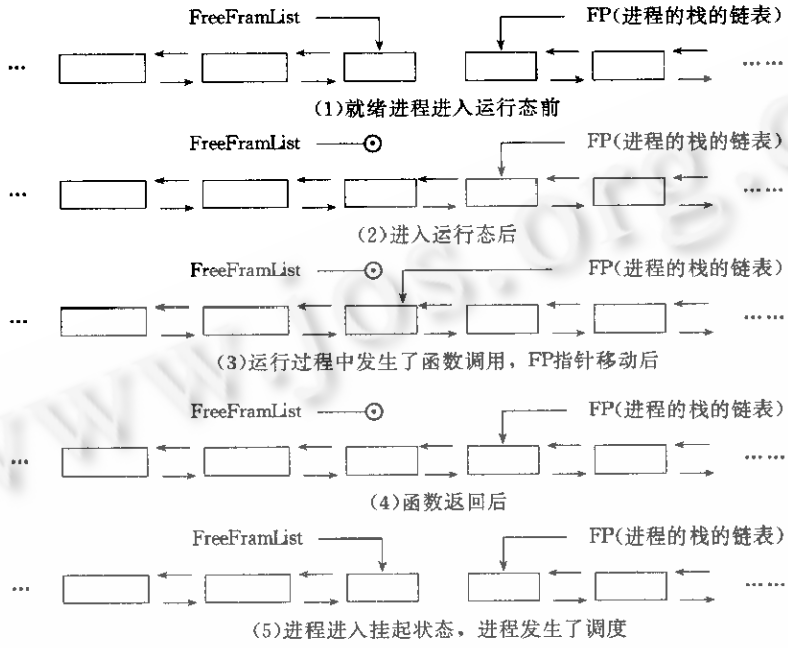


图2 程序的运行过程

2.2 并行LE—MACHINE

由于分布式体系结构的多处理机系统具有较强的扩充性,所以我们选择分布式的体系结构作为我们实现的目标系统(TRANSPUTER系统正是一种分布式内存的多处理机系统).并行LE—MACHINE采用多线程(THREAD)并行体系结构(类似OS2操作系统),并行LE计值语义中的"伪延迟对象"采用THREAD来表示.

A.抽象机模型

定义1. LE—MACHINE由一组处理单元所构成的集合和处理机间的通讯网络构成.

LE—MACHINE={PE集合,PE_NETWORK,PE_STATS}

其中PE集合:处理机的集合.

PE_NETWORK:处理机间的通讯网络

PE_STATS:∪PE_STAT(i);PE_STAT(i)为处理机i的状态,i∈PE_SET.

定义2. 处理机PE_STAT(i)由一组进程和一些分布于处理机i中的THREAD所构成的计算任务组成:(THREAD对应于LE语义中伪延迟对象)

PE_STAT(i)=〈{TASK.V}∪TASK_SET,THREADS,{V:FRAME F V_S},PC〉

其中THREADS为处理机PE中的LE语义伪延迟对象队列[3,8];

TASK.V 为当前运行态进程,V为该进程栈指针FP所指向的FRAME数据块;

F 为当前FRAME数据块的代码指针;

V_S　　为该 FRAME 的计值空间;

PC　　为当前活跃进程的代码指针.

每个进程 TASK 均有一个由伪延迟对象所构成的任务队列,表示该进程计值过程中所生成的伪延迟对象,该伪延迟对象亦被处理机 PE 中的负载调度进程用于多处理机系统的负载均衡.

定义 3. 处理机 PEi 和 PEj 间的负载迁移:

迁移前的状态:

PE_STAT(PEi)=⟨{TASK. V}∪TASK_SET,{X:THREAD Fi V_S2}∪THREADS,{V:FRAME F V_S},PC⟩

PE_STAT(PEj)=⟨{},{},NULL,PCj=LoadSharingProc⟩

迁移后的状态:

PE_STAT(PEi)=⟨{TASK. V}∪TASK_SET,{X:THREAD THREAD_WAIT}∪THREADS,{V:FRAME F V_S},PC⟩

PE_STAT(PEj)=⟨{TASK2. U},{},{U:FRAME Fi V_S2},PCj=Fi⟩,

其中,进程 TASK2 的结果目的地址为⟨PEi,TASK,X⟩.

THREAD 惰性进程生成法的基本实现算法:

1. 对于程序中可被并行计值的子表达式,在第一次计值时生成对应伪延迟对象的 THREAD 结构[THREAD F X1…XN],其中 F 为该子表达式计算的程序入口地址,X1…XN 为存放在 FRAME 结构中的计值环境. 该伪延迟对象作为子表达式结果返回,并作为中间结果压入栈中;

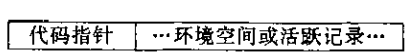
2. 将该 THREAD 链接在本进程的 THREAD 队列的尾部;

3. 当负载调度时,从各进程的 THREAD 队列首部开始寻找一个有负载的 THREAD 扩散到其它处理机中,生成相应的子进程进行计算. 同时本处理机中的 THREAD 的 F 标记为 THREAD_WAIT(进程挂起的系统原语的程序入口);

4. 子进程计值结束后,将结果送入原 THREAD,并修改 THREAD 的 F 为 THREAD_RETURN(将栈顶元素作为函数返回结果返回的系统原语的程序入口地址),并唤醒处于睡眠状态的父进程;

5. 当计值过程中引用 THREAD 的值时,则将该 THREAD 链入进程的栈链表的栈顶中,然后移动 FP 指针指向该 FRAM 块,原代码指针压栈保存后程序跳入 THREAD 中 F 的代码段,进行相应的计算.

LE－MACHINE 所新引入的"延迟对象"和"伪延迟对象"由于这两类对象中需保存运行时的环境,所以也采用了与栈的 FRAME 类似的数据结构:



对于 THREAD 和"延迟对象",代码指针为该被延迟计算部分的程序入口,其它部分是计值所需的环境;对于栈则代码指针用于存放函数的返回地址. 采用这一统一的表示形式和并行 LE 语义是分不开的,并行 LE 语义要求在一个基本函数对其参数计值时若对象为 THREAD 或需要对参数积极计值而对象为"延迟对象"时,均须进入对 THREAD 或"延迟

对象”的计值.采用以上统一的FRAM的数据格式,可在这类计值时,将该对象插入当前的栈链表,然后类似函数调用一样对THREAD或“延迟对象”进行计值,如图3所示.

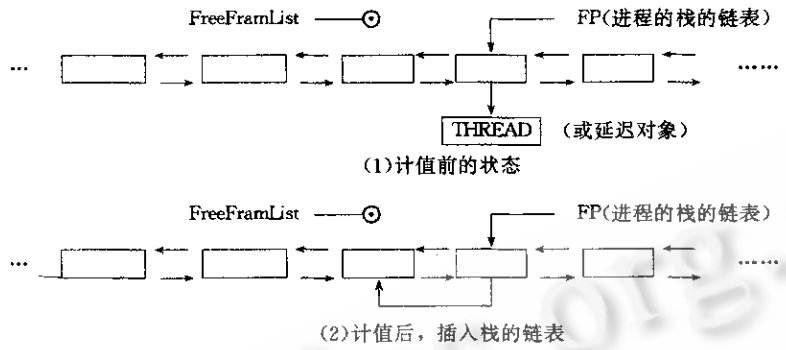


图3 延迟对象的计值过程

3 性能评价

我们在由4台TRANSPUTER所构成的分布式内存体系结构的多处理机系统上,已初步实现了基于LE计值语义的LE—MACHINE.该硬件系统为T800—20所构成,每一台处理机有可用内存1M,处理机间通过20M BPS的通讯端口相联.由于系统配置所限,目前拓扑结构为环(RING).

软件的编译采用首先根据LE计值语义生成LE—MACHINE的抽象机指令,然后通过宏定义用C语言来实现各抽象机指令,C语言代码使用INMOS公司的ANSI C编译器生成可执行的TRANSPUTER二进制代码.

采用了并行LE计值语义进行归约,特别是THREAD惰性进程生成法的应用在充分开发系统的并行处理能力的同时极大地减少进程生成的数目,同时由于计值的积极性也充分开发了程序中的并行性;采用本文中LE—MACHINE的栈的优化也加速了程序的运行速度.我们采用NFIB,QUEENS,TAK等BENCHMARK程序在我们的原型系统中进行了测试,测试结果与其它系统对比如下:

表1 基准程序的测试结果

单位:秒

	LE—MACHINE (4 T800—20)	HDG—MACHINE[6] (4 T800—25)	LISP—M1
拓扑结构	环	全互连	LISP处理机
nfib20	0.129	0.373	FIB:20 0.16
nfib25	1.051	/	FIB:25 19.99
nfib30	9.870	/	/
tak 18 12 6	0.287	1.433	0.47
tak 24 16 8	9.966	/	18.77
queens 8	0.721	/	/
queens 6	0.076	0.076	/

注: NFIB 求解NFIB(N)=1+NFIB(N−1)+NFIB(N−2),IF N>2,否则为1.
QUEENS N求解N皇后的所有解法
TAK 函数是LISP中的一个基本测试函数.

4 结束语

从上节的测试数据可以看出,基于 LE 语义的 LE—MACHINE 具有较好的执行效率.但是系统中仍然存在很多需要改进之处:如在目标代码的生成中充分利用 TRANSPUTER 中 W 基地址寻址方式,使得 W 寄存器与本文抽象机中 FRMAE 指针 FP 相对应,显然与目前 C 语言的方式相比将极大地提高抽象机在 TRANSPUTER 上执行速度;其他类似问题还有文献[2]中的臆测并行性的开发.函数式语言的并行实现这一研究领域非常复杂,还有许多工作有待我们去研究.

参考文献

1 杨澜,孙永强.用于开发 FP 语言并行性的静态分析技术.计算机学报,1990,**13**(**10**):721—727.

2 孙永强,袁伟.函数式语言 FP 中并行的开发和控制.计算机学报,1992,**15**(**5**):321—331.

3 Sun Yongqiang, Yang Lan *et al*. A simulated parallel machine for implementation FP language. NEW GENERATION COMPUTER SYSTEM, 1989. 303—309.

4 Hudak Paul. Serail combinator: optimal grains of parallelism. In: ACM Symposium on Lisp and Functional Programming, 1988. 383—399.

5 Augurtsson Lennart. Parallel graph reduction with the ⟨V,G⟩—machine. Proc of IFIP Conf. on Functional Programming Languages and Computer Architecture, London, 1989. 202—213.

6 Kindon Hugh *et al*. The HDG—machine: a highly distributed graph—reducer for a transputer network. The Computer Journal, 1992,**34**(**4**):98—102.

7 Mohr Erich, Kranz D A *et al*. Lazy task creation: a technique for increasing the granularity of parallel programs. IEEE Trans. on Parallel and Distributed System, 1991,**2**(**3**):264—280.

8 袁伟,孙永强.基于松耦合 MIMD 计算机系统的函数式语言并行实现技术,上海交通大学学报,1993,**27**(**4**):54—63.

9 Johnsson T. Spineless tageless G—machine. ACM Conference on Functional Programming Language, 1989. 184—201.

PARALLEL ABSTRACT MACHINE MODEL OF FUNCTION LANGUAGE FOR TRANSPUTER NETWORK

Yuan Wei Sun Yongqiang

(*Department of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai* 200030)

Abstract This paper puts forward a parallel abstract machine—parallel LE machine for transputer networks, which is based on the parallel LE semantic model. To increase the granularity of parallel process during the running time, the thread lazy process creation method is adapted in the LE machine, which decreases the time of process creation and explored the parallelism of the parallel process system at the same time. With the aid of bidirection linked stack frame structure, the prototype system of this method shows a good performance.

Key words Function language, parallel process, abstract machine model.