传统数据流模型的分析及改进*

王永革

(南开大学数学研究所,天津 300071)

摘要 数据流计算机作为新一代并行机迅速发展起来,但由于数据流模型本身的局限性,使得商用数据流机器的制造还难于普及.本文通过分析运算的操作字符与非操作字符,改进传统模型的点火规则、引入变量并拓广变量的概念,定义了粗粒度数据流模型CDFM(Coarse granularity Data Flow Model).

关键词 数据流,粒度.

自从数据流模型问世以来,无论在其理论还是应用方面都得到迅速发展.但由于数据流模型的充分并行性是以结点的细粒度实现的,因而使得它具有无法避免的局限性,主要是通讯费用的巨额增长问题.近年来,国外很多人尝试通过减少结点的细粒度来降低通讯费用,但这会使模型充分利用并行性这一优点迅速降低.分析其主要原因,是因为人们对"数据准备好"、"数据依赖"这一概念的理解有一定局限性,在传统模型中,"数据准备好"是指运算对象都以赋值.胡国定在[7,8]中通过对运算的分析指出:运算对象可分为操作字符与非操作字符,因此我们觉得以如下方式理解"数据准备好"则更好些:运算对象中操作字符都已赋值;而非操作字符不一定赋值,只知道其存放位置就可认为"数据准备好".基于这一理解,我们引进了粗粒度数据流模型CDFM,对传统数据流模型作了如下改进:

1. 结点的粗粒度性;
2. 结点的非纯函数性;
3. 结点有内部状态;
4. 类似于控制流模型,引入变量的概念并拓广变量的传统理解:即变量在未赋值之前是可以访问的;
5. 点火规则的改进:可部分点火.

1 传统数据流模型

传统数据流模型具有下边两条共性[9]:

* 本文1991年6月11日收到,1991年10月31日定稿

作者王永革,27岁,1991年硕士毕业于南开大学,现在Heidelberg大学读博士,主要研究领域为应用逻辑,形式语言,并行计算模型及语义,Petri网.

本文通讯联系人:王永革,天津 300071,南开大学数学研究所

性质 1. 细粒度性:每个结点只包含一个简单的算术、逻辑或控制运算.

性质 2. 结点的函数性:结点没有内部控制状态.结点的运算仅代表一个纯粹的函数,即输出托肯(tokens)仅仅依赖于当前的输入托肯.

这两条性质使得数据流模型具有足够的简洁性与充分的并行性,目前已造出的数据流机器都具有上述二性质[3,12].但是,也正由于这两条性质,使得数据流机器的局限性很大,主要是通讯费用的增长问题.下面做一具体分析.

1.细粒度性导致了巨额通讯费用

在传统数据流模型中,为了最大地利用算法的静态并行性,一项任务(结点)仅仅是一条机器指令.但是,物极必反,在这种过分追求细节的模型中,为了保持运算的正确性,必须花大量的时间去做许多额外的工作:如数据的频繁传递和反复测试各可动结点、决定执行何种任务、以保持其同步性等等.

为了克服这种局限性,我们建议采用粗粒度数据流模型,即通过集聚更多的指令扩大结点的规模,形成宏结点.

2.等待非操作字符使能执行的运算无法执行

胡国定[7,8]的工作表明:运算对象(操作数)可分为操作字符与非操作字符.如 Post 产生式 $a_1X_1a_2X_2\rightarrow b_1X_2$ 的算域(运算对象)$\alpha=\{a_1,X_1a_2,X_2\}$可分为操作字符 $\alpha_1=\{a_1,a_2\}$和非操作字符 $\alpha_2=\{X_1,X_2\}$,α_1 在运算过程中决定了输出值 b_1 及输出中 b_1,X_2 的排列顺序.而 α_2 只是一个数据载体,在运算过程中不起决定作用.因此这个运算的执行只需知道 a_1,a_2 的值,X_1 与 X_2 的值可暂不考虑,作为变量传递过去.这表明,在粗粒度中如果仍采用传统的"数据流"概念,那将使得本来可执行的运算变得无法执行.因此,我们必须改进点火规则,以达到对"数据流"的重新认识.

3.结点的函数性降低了运算速度,增加了不必要的通讯设施

传统数据流模型图中,结点是一个纯粹的函数,即输出托肯仅与当前的输入托肯值有关,而与该结点以前的计算无关,因此我们可以称传统数据流模型的纯函数性为无记忆性.作为一个简单的例子,函数 $f(n)=n!$ 的计算在传统数据流模型中可用图 1(a)表示.图中各

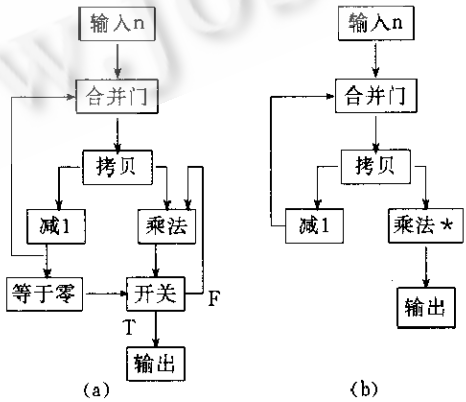

图1 阶乘的计算图

个结点都无记忆性,特别地对于"乘法"结点也是如此.这就使得必须用一个开关结点作为信

息反馈，因而大大增加了图的复杂性，同时也增加了数据流模型中最为昂贵的同步测试通讯费用. 如果允许结点有记忆性，则我们可以构造图 1(b)的模型图. 在“乘法 * ”结点中设置内部状态，记忆运算结果，并判断输入是否为零，从而决定是否输出. 与图 1(a)相比较，图 1(b)并未减少算法的静态并行性，但它有明显的优越性：减少了通讯设备，废除了多余的数据传送与反馈.

4. 点火规则的局限性

传统数据流模型分为数据驱动与需求驱动两种. 在数据驱动模型中，结点能动当且仅当相对应的托肯到达该结点的各条输入弧上. 对于语句“If f_1(弧 1)then f_2(弧 2) else f_3(弧 3)”，我们用图 2 的宏结点表示，该结点可动当且仅当 f_1，f_2 与 f_3 的变量数值分别到达弧 1、弧 2、弧 3. 实际上，如果判断 f_1 为真，则弧 3 上 f_3 的变量输入是无用的；当 f_1 为假时，f_2 的变量是无用的. 因此我们认为有必要采取部分点火的规则.

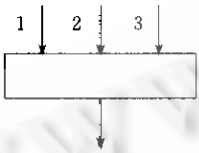

图2

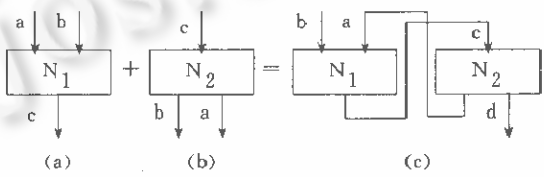

图3 数据流图的合并

2 粗粒度数据流模型 CDFM

传统数据流模型中，一个数据流图由结点部分与通道部分组成，通道连接结点. 每个通道都标以不同的名字. 结点之间以及结点与外界通讯都是通过在通道上传递数据实现的，并且通道上的数据流动是以队列的方式进行的(FIFO). 通道可分为以下三类：

1. 外部输入通道，外界向图传送数据；

2. 内部通道，结点之间传送数据；

3. 外部输出通道，结点向外界发送数据.

数据流图的动作是通过以下方式进行的：结点消耗输入通道上的数据同时在输出通道上产生新数据.

一般的大型数据流图是由若干小图复合而成的，复合方式比较简单：只要将外部输入通道、外部输出通道中同名的进行合并即可. 如图 3(a)、图 3(b)的数据流图合并后可得图 3(c)的数据流图.

为了避免传统模型中的细粒度性和纯函数性(无记忆性)，可将每个结点看成带有存储器的有穷自动机，即该结点不仅可以做简单的算术、逻辑或控制运算，而且还可以做较为复杂的运算；不仅具有纯函数性，而且具有历史记忆性. 同时为了提高运算速度，避免等待非操作字符，我们引入变量的概念，让每个结点有两条额外通道与之相连，一条用于读变量的值，另一条用于写变量，并且采用部分点火的规则，形式地有定义 2.1：(注：在以下各节，都假定用 V_1 表示数据集，用 V_2 表示变量的地址集合，V 表示 $V_1 \cup V_2$.)

定义 2.1. 结点 p 的形式规定是十元组$(I_p, R_p, W_p, O_p, S_p, s_p^0, q_{waitp}, Register_p, G_p, FS)$，其中：

I_p 是通道集，称为 p 的输入通道；

R_p 是读通道，与地址总线相连(地址总线的概念在后介绍)；

W_p 是写通道，与地址总线相连；

O_p 是通道集，称为 p 的输出通道；

S_p 是状态集；

s_p^0 是初始状态，$s_p^0 \in S_p$；

q_{waitp}是等待状态，$q_{waitp} \in S_p$；

$Register_p$ 是 p 的内部寄存器集；

G_p 是点火集，即下述三种点火规则集合的并：

(1)点火规则是$\langle s, \chi_{in}, s', \chi_{out}, \chi_w \rangle$；

(2)点火规则是$\langle s, \chi_{in}, q_{waitp}, \chi_{Register} \rangle$；

(3)点火规则是$\langle q_{waitp}, \chi_{register}, \chi_R, s', \chi_{out}, \chi_w \rangle$；

这里χ_{in}是从 I_p 到$\{\varepsilon\} \cup V$ 的映射；$\chi_{register}$是从 Register 到 V 的映射；$s, s' \in S_p \backslash \{q_{waitp}\}$；$\chi_R$ 是该结点进入等待状态时所等待的变量的地址到 V_1 的映射；χ_{out}是从 O_p 到 V 的映射；χ_w 是该结点新生变量的地址到 V_1 的映射.

下边对各类点火规则给予直观解释：

(1)$\langle s, \chi_{in}, s', \chi_{out}, \chi_w \rangle$：当结点 p 处于状态 s，每个输入通道 in_i 上的第一个值(记住各通道是 FIFO 队列)是 $\chi_{in}(in_i)$时，结点可被点火；点火过程是这样的：消耗各输入通道 in_i 上的输入值 $\chi_{in}(in_i)$，在每个输出通道 out_j 上产生 $\chi_{out}(out_j)$，并放到 out_j 的队列中去；将产生的全局变量的值以 χ_w 的形式通过写通道写入存储器中；最后结点 p 进入状态 s'.

(2)$\langle s, \chi_{in}, q_{waitp}, \chi_{Register} \rangle$：当结点 p 处于状态 s，每个输入通道 in_i 上的第一个值是 $\chi_{in}(in_i)$时，结点可被点火. 但很可能是由于本节点的操作字符有一些是以变量的形式(地址)传送过来的，尚未赋值，所以这次点火只是进入等待状态 q_{waitp}，并将各输入通道 in_i 上的值 $\chi_{in}(in_i)$及原状态 s 以 $\chi_{register}$的形式存入内部寄存器 Register 中. 进入等待状态的结点不能再被输入通道上的值点火.

(3)$\langle q_{waitp}, \chi_{register}, \chi_R, s', \chi_{out}, \chi_w \rangle$：进入等待状态的结点 p，通过读通道去测试它所等待的各变量是否已赋值. 如已赋值，则该结点可被激活，激活后，通过通道 R 读入变量的值，利用内部寄存器 Register 中保存的以前的输入值及原状态 s，在输出通道 out_j 的队列尾部产生值 $\chi_{out}(out_j)$，并将产生的全局变量的值以 χ_w 的形式通过写通道写入存储器中，最后结点进入状态 s'.

注：我们这里对变量的概念进行了推广，即变量在未赋值之前是可以被访问的.

FS 是对点火规则的合理性要求集，这主要依赖于具体的模型. 比如，在上述模型中可要求第 1、3 两种经过中间等待状态而使结点发生的点火与第 1 种不经过等待状态的直接点火是相容的，也就是说，如果某些操作字符在输入通道上是以变量的形式传送过来使结点进入等待状态，等待这些变量被赋值，赋值之后激活结点而产生的输出值应和第一次操作字符在输入通道上以数值的形式传送过来而使结点以第一种方式点火后所产生的输出值相同(在

操作字符相同的情况下).

在本节的最后，我们给出数据流图的定义：

定义 2.2. 数据流网络 N 是结点连同通道的集合 P. 粗粒度数据流模型 $CDFM_N$ 是数据流网络 N 的一个规定，即 $CDFM_N$ 是一个十一元组：$(I_N, O_N, R_N, W_N, Register_N, \sum_N, \sigma_N^0, G_N, FS_N, Bus_N, Memo_N)$，其中

$I_N=(\bigcup_{p\in P}I_p)\backslash(\bigcup_{p\in P}O_p)$是 N 的输入通道；

$O_N=(\bigcup_{p\in P}O_p)\backslash(\bigcup_{p\in P}I_p)$是 N 的输出通道；

$R_N=\bigcup_{p\in P}R_p$；

$W_N=\bigcup_{p\in P}W_p$；

$Register_N=\bigcup_{p\in P}Register_p$；

$FS_N=\bigcup_{p\in P}FS_p$

$Memo_N$ 是存储器；

Bus_N 是地址总线，连接 R_N，W_N 中的读写通道与存储器 $Memo_N$；

$\sum_N$ 是 N 的构形集，即 $P\cup C\cup Register_N\cup Memo_N$ 上的映射 σ 的集合. 直观上讲，对于 $\sum_N$ 中的任一构形 σ，任给结点 $p\in P$，$\sigma(p)$为 N 的当前状态；任给通道 $c\in C$，$\sigma(c)$为通道 c 上的当前数据队列；任给寄存器 $Register_p\in Register_N$，$\sigma(Register_p)$为该寄存器中的当前值；任给 $v\in Memo_N$，$\sigma(v)$为存储器单元 v 中的当前值；$\sigma_N^0\in\sum_N$，将每个结点 p 映射为 s_p^0，将 C，Register，Memo 映射为空值 ε；G_N 是变迁集，其可能性有以下三种：

(1)内部变迁 $\tau:\sigma\to\sigma'$，如存在结点 p 的点火规则：

$\langle 1\rangle\langle s,\chi_{in},s',\chi_{out},\chi_w\rangle$；或

$\langle 2\rangle\langle s,\chi_{in},q_{waitp},\chi_{Register}\rangle$；或

$\langle 3\rangle\langle q_{waitp},\chi_{Register},\chi_R,s',\chi_{out},\chi_w\rangle$

使得 $\tau:\sigma\to\sigma'$ 是 p 的一次点火，形式地说(注：这里只对情形$\langle 1\rangle\langle s,\chi_{in},s',\chi_{out},\chi_w\rangle$加以说明，其余二情况类似)，下列四条件成立：

$A:\sigma(p)=s,\sigma'(p)=s'$；

$\sigma(p')=\sigma'(p')$, If $p\neq p'$；

$B:\sigma(in_i)=\sigma'(in_i)\cdot x_i$, If $\chi_{in}=[(in_1,x_1),\cdots,(in_n,x_n)]$ and $in_i\in I_p$；

$\sigma'(out_i)=y_i\cdot\sigma(out_i)$, If $\chi_{out}=[(out_1,y_1),\cdots,(out_m,y_m)]$, and $out_i\in O_p$；

$\sigma(c)=\sigma'(c)$, If not $c\in I_p\cup O_p$；

$C:\sigma(Register_p)=\sigma'(Register_p)$ If $p\in P$；

$D:\sigma'(v_i)=z_i$, If $\forall\ (v_i,z_i)\in\chi_w$；这里 $v_i,v_j\in Memo_N$.

$\sigma'(v_j)=\sigma(v_j)$, If $z(v_j,z)\in\chi_w$.

(2)输入变迁 $\tau:\sigma\to\sigma'$，若存在 $in_i\in I_N$，$x\in V_1$ 使得

$\sigma'(in_i)=x\cdot\sigma(in_i)$, and $\sigma'(in_j)=\sigma(in_j)$ for $i\neq j$.

(3)输出变迁 $\tau:\sigma\to\sigma'$，若存在 $out_j\in O_N$，$y\in V_1$，使得

$\sigma(out_i)=\sigma'(out_i)\cdot y$ and $\sigma(out_j)=\sigma'(out_j)$ for $i\neq j$.

关于模型 CDFM 与传统模型的比较及 CDFM 的实际应用，我们将另文处理.

致谢 本文是在胡国定先生的鼓励、徐书润教授和胡久稔研究员的亲自指导下完成的，作者在此表示衷心的感谢，同时感谢审稿人提出的修改意见.

参考文献

1 王永革. 序佩特里网(Petri)计算能力分析. 软件学报，1993；**4(3)**：35—41.

2 Agha G, Actors. A model of concurrent computation in distributed systems. MIT Press, Cambridge, Mass., 1986.

3 Avind, Nikhil R S. Executing a program on the MIT tagged—token dataflow architecture. IEEE Trans. on Comput., 1990；**39(3)**：300—318.

4 Brock J D, Ackerman W B. Scenarios: a model of non—determinate computation. In: Diaz, Ramos eds. Formalization of Programing Concepts, LNCS 107, 252—259, Springer Verlag, 1981.

5 Broy M. Nondeterministic data flow programs: how to avoid the merge anomaly. Science of Computer Programing, 1988(**10**)：65—85.

6 Gluck—Hiltrop E, Johnk M, Schurfeld U. The stollmann data flow machine. In: E. Odijk M. Rem, J. C. Syre eds. PARLE'89, LNCS 365, 1989；216—234.

7 Hu Guoding. On the mathematical model of computing machine. J. of Computer Science and Technology, 1988；**3(4)**.

8 Hu Guoding. Parallel computation simulating sequential computation, 1990. (to print)

9 IEEE Computer, 1982；**15(2)**.

10 Jonsson B, Kok J N. Comparing two fully abstract dataflow models. In: E. Odijk M. Rem, J. C. Syre eds. PARLE'89, LNCS 366, 1989；216—234.

11 Sharp S A. Data flow computing. ELLIS HORWOOD LIMITED, 1985.

12 Sargeant J, Kirkham C C. Stored data structures on the manchester dataflow machine. Proc. 13th Annu. Sympo. Comput. Architecture, 1986；235—242.

AN ANALYSIS OF TRADITIONAL DATA FLOW MODEL AND ITS MODIFICATION

Wang Yongge

(*Nankai Institute of Mathematics, Nankai University, Tianjin* 300071)

Abstract As a new generation of parallel computer, data flow computer has developed very quickly, but because of its own deficiency, it is very difficult to build dataflow computers for commercial use. In this paper, by analysing the operating words and non—operating words of computation, modifying the firing rule of traditional dataflow model, introducing variables in the model, they will define a Coarse granularity Data Flow Model CDFM.

Key words Dataflow, granularity.