事件触发并发数据流模型*

王瑞荣+, 汪乐宇

(浙江大学 仪器科学与工程系 数字技术及仪器研究所,浙江 杭州 310027)

An Event-Triggered Concurrent Dataflow Model

WANG Rui-Rong+, WANG Le-Yu

(Institute of Advanced Digital Technologies and Instrumentation, Department of Instrumentation Science and Engineering, Zhejiang University, Hangzhou 310027, China)

+ Corresponding author: Phn: 86-571-87952120, Fax: 86-571-87951179, E-mail: chenwang@cbeis.zju.edu.cn; wangrr@hotmail.com

http://www.zju.edu.cn

Received 2001-12-11; Accepted 2002-04-10

Wang RR, Wang LY. An event-triggered concurrent dataflow model. *Journal of Software*, 2003,14(3): 409~414.

Abstract: DHDF (dynamic homogeneous dataflow) is the kernel of most graphic programming platform. For the natural data-driven property, the dynamic homogeneous dataflow can not work properly with the event-driven operating system, which leads to two demerits: one is inefficiency in CPU using, the other is low respond speed and poor in real time performance. An ECDF (event triggered concurrent dataflow) model and its formal description are presented in this paper. Based on multi-thread and event-triggered mechanism, the real time performance and execution efficiency of dataflow-based system are improved. The experimental results of a test system prove that the event-driven concurrent dataflow can ameliorate the performance of dataflow-based system to a certain degree in most conditions comparing with the dynamic homogeneous dataflow model. ECDF model is also suited for the Reactive systems design, and high-speed burst-data flow processing especially.

Key words: dataflow; graphic programming; virtual instrument; real-time system; dataflow visual language

摘 要: DHDF(动态纯数据流)是许多图形化编程平台的核心.由于它的自然属性(数据驱动)与操作系统事件驱动模型不能很好地结合,导致了两个明显的不足:运行效率低,CPU 占用率高;对外部事件响应速度慢,系统实时性差.提出了一种 ECDF(事件触发并发数据流)模型,并给出了该模型的文法描述以及调度算法.ECDF 模型通过引入多优先级线程以及事件触发机制,在很大程度上提高了系统的实时性与运行效率.以测试系统为背景,对有关应用实例进行测试与分析,结果表明,与 DHDF 模型相比,ECDF 模型使系统的性能在不同条件下都得到了相应的提高.该模型特别适用于处理突发性高速数据流,也适用于 Reactive 系统设计.

关键词: 数据流;图形化编程;虚拟仪器;实时系统;数据流可视化语言

* Supported by the Key Science-Technology Project of the National 'Ninth Five-Year-Plan' of China under Grant No.97-772-01-06 (国家"九五"重点科技攻关项目)

第一作者简介: 王瑞荣(1973－),男,浙江义乌人,博士生,主要研究领域为虚拟仪器,可视化编程技术.

中图法分类号: TP311 文献标识码: A

数据流程序图执行模型[1,2](或简称数据流模型)不同于传统冯诺伊曼执行模型,它用图形表示程序,是存储器-程序运行模型的有力补充.近十年来,数据流模型得到了很大的发展,被广泛应用于可视化编程、处理器设计、信号处理以及可重配置计算等各个领域.

由于数据流模型结构简单、语义清晰,所以在虚拟仪器研究领域中出现的几个优秀的图形化编程平台,如LabVIEW®[3]和 HP VEE[4],都是基于数据流语言的.测控系统与仪器科学的发展,对系统的并行与实时性提出了越来越高的要求.而动态纯数据流模型的数据驱动本质导致了其对事件响应的困难,这使得基于它的虚拟仪器系统很难完成实时性要求比较高的任务.同时,由于内建的数据驱动机制需要一个动态调度器,必须连续对节点进行扫描以判断其是否能够激活,而这种连续扫描的调度机制几乎占尽了宝贵的 CPU 资源.

近年来对数据流的研究主要是对纯数据驱动的数据流模型的研究[5~7],而纯数据流模型不能很好地与事件驱动的操作系统协调,因此必须对单线程动态纯数据流的驱动机制与运行机制加以改进和扩展才能提高系统的响应速度和运行效率.通过对测控系统进行建模和对操作系统的运行机制进行分析,提出了一种多线程事件触发数据流模型.本文不仅给出了该模型的文法描述,也给出了运行调度算法.实验结果表明:① 该模型明显降低了 CPU 占用率,提高了程序的运行效率;② 解决了纯数据流模型中的阻塞调用问题,可以在线程中对硬件操作进行阻塞调用;③ 提高了对外部事件的响应速度,增强了系统实时性;④ 通过引入带有 FIFO 结构的边结构,解决了并发节点之间的数据传输与缓冲等问题.

1 动态纯数据流模型

1.1 单线程动态纯数据流模型

在单线程动态纯数据流模型中,一个数据流程图描述的程序是由能够传输数据的边连接而成的节点的集合.该数据流程图可以表示为一个二元组: $P=(N,E)$.其中 N 是一个节点集合,每个节点包含一个输入端子的集合 T_I,一个输出端子集合 T_O; E 是有序积 $(C\cdot T_o)\times(C\cdot T_I)$ 的一个子集,其中元素称为边.每个节点代表着一定的计算,每条边代表着一条数据流通路径.这些基本的计算节点和传输数据的边可以构成复杂的计算.

每条边可以表示为 $e=(u.o_t,v.i_t)$,表示该边由源节点 u 的输出端口 o_t 连到目标节点 v 的输入端口 i_t.

每个节点可以表示为一个六元组: $N=(E_M,E,T_I,T_o,a,S)$,其中

E_M ——节点的可激活函数,返回一个布尔变量,代表该节点是否能够激活.该函数检查节点的所有输入端口,若所有输入端口的数据已经更新,则返回真值.

E——节点的计算函数,当节点被激活后执行该函数,该计算是构成复杂计算的基本单元.

T_I——输入端子集合,可以为空集.

T_o——输出端子集合,可以为空集.

α——节点的属性集,依节点的用途而定.

S——节点的状态,表示节点激活情况.

1.2 调度算法

单线程动态纯数据流模型的运行调度算法比较简单,这也是它广泛应用于可视化编程的原因.其中节点运算可以由一个函数(或称模块)来完成.其调度运行模型可以表述如下:

① 对节点列表 N_L 中的每个节点 n 作可激活性检查.若激活函数 E_M 返回的值为真,则运行该节点的计算函数 E.在计算完成之后,置节点的输出端口集 T_o 中的所有端口的状态为数据更新状态,将输入端口集 T_I 中的所有端口的状态为过期状态.

② 对边列表 E_L 中的所有边 e 进行数据更新检查.若输入端的数据更新,则将数据从源节点输出端口传递到目标节点的输入端口,并更新目标节点的输入端口的状态为数据更新状态,同时将源节点的输出端口的状态

置为过期状态.

③ 若没有人工干预则返回第①步,否则退出.

2 事件触发并发数据流模型

2.1 事件触发并发数据流模型

事件触发并发数据流模型,是在研究测控系统的特性之后,在动态纯数据流的基础上提出来的.一般来讲,用于测控领域的虚拟仪器系统可以分为 3 个层次:数据输入层,其主要功能是从物理硬件获取数据或自身产生数据,是数据处理层的数据来源;数据处理层,完成各种数据处理,如滤波、控制算法计算等等;数据输出层,包括从网络输出数据、把数据输出到存储器或输出到显示器等等.从实时性的角度来考虑,数据输入层的优先级应该高于数据处理层的优先级,而数据处理层的优先级又应该高于数据输出层的优先级(在不考虑实时控制的情况下).

在事件触发并发数据流模型中,不同优先级的任务可以用不同优先级的线程节点来表示.不同线程节点之间用带有先入先出队列(FIFO)的边连接,不至于在运行时丢失数据.数据流的调度器由事先定义的消息激活.调度器执行完毕又可以触发数据传输事件,通知线程输出节点可以输出数据.模型如图 1 所示.

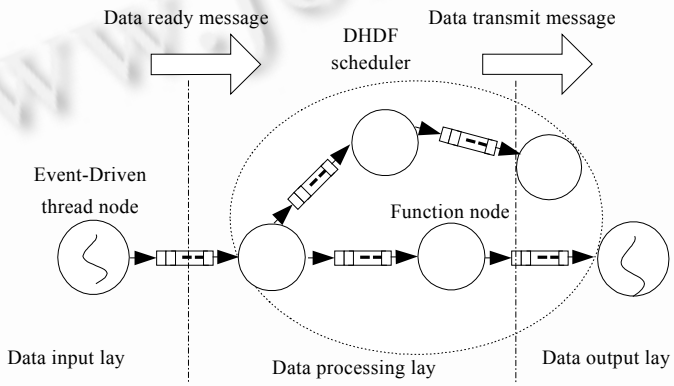

Fig.1 Event-Trigged current dataflow model

图 1 事件触发并发数据流模型

2.2 文法描述

一个事件触发并发数据流程序可以表示为一个四元组: $P=(F,T,A,E)$,其中 F 是函数节点集,每个节点包含一个输出端口集 Q_o,一个输入端口集 Q_I;T 是一个线程节点集,也包含一个输入端口集 Q_I 和一个输出端口集 Q_o;A 是有序积 $((F\cdot Q_o)\vee(T\cdot Q_o))\times((F\cdot Q_I)\vee(T\cdot Q_I))$ 的一个子集,其元素称为边;E 是一个事件集,含有线程之间通信的事件(或消息).

A 中的边 $a=(u.qo,v.qi,q)$ 表示由源节点 u 的输出端口 qo 到目标节点 v 的输入端口 qi 的连接.其中 q 是内建的先入先出队列结构.源节点可以将数据通过边内 FIFO 结构缓冲后传输给目标节点.这样的结构,再加上对边内数据读写的同步,就可以顺利解决线程间数据传输与同步的问题.

函数节点 F 可以用一个七元组来表示, $F=(B,E_M,p,Q_I,Q_o,a,s)$,其中

B——是否线程节点的标志,这里为 FALSE,表示为非线程节点;

E_M——为函数节点的可激活函数,返回一个布尔量;

p——节点的执行函数,这里用 p 来替代纯数据流模型中的执行函数,以避免与事件集 E 的命名冲突;

Q_I——输入端口集合;

Q_o——输出端口集合;

α——节点的属性集,依节点的用途而定;

S——节点的状态集.

线程节点可以用一个九元组来表示,$T=(B,P_L,E_L,I,E_s,Q_I,Q_o,a,S)$,其中

B——是否线程节点的标志,这里为 TRUE,表示为线程节点;

P_L——线程的优先级;

E_L——监听事件集,指定线程节点响应哪些事件;

I——监听函数,对不同的事件作响应的处理;

E_S——触发事件集,预先定义的用于触发调度器运行的消息集;

Q_I——输入端口集合;

Q_o——输出端口集合;

α——节点的属性集,依节点的用途而定;

S——节点的状态集.

2.3 事件触发并发数据流运行调度算法

在引入带有 FIFO 结构的边之后,目标节点和源节点通过读写 FIFO 直接实现数据传输.因此在程序编辑完成之后,不需要对边进行操作.

程序完成之后是一个节点列表 L_N.算法开始:

① 遍历节点列表 L_N,对每个节点的 B 进行判断,若 B 为 TRUE,则把线程节点放入一个线程节点列表 L_T,并且在节点列表中删除该节点.

② 遍历线程节点列表 L_T,执行监听函数 I,启动线程并对事件进行监听.

③ 若相应的事件发生,则通过监听函数读取数据,若数据满足一定数量,则发出数据就绪消息,触发纯数据流调度器.

④ 若数据流调度器被触发,则往下执行,否则等待消息.

⑤ 遍历节点列表 L_N,若满足激活条件则将该节点放入激活节点列表 L_A.

⑥ 查询 L_A,若 $L_A=\varnothing$,调度器关闭,发送数据发送消息.

⑦ 执行 L_A 中的所有节点的执行函数 p,回到算法④.

3 试验结果分析

试验环境:PIII866PC 机,操作系统 WINDOWS2000,程序用 VC6.0 编写.该 PC 机带有多媒体计时器,因此计时精度可以达到 1us.WINDOWS2000 是一个基于 NT 的多任务抢占式调度操作系统,其中的线程可以分为 6 个层次,32 个优先级.

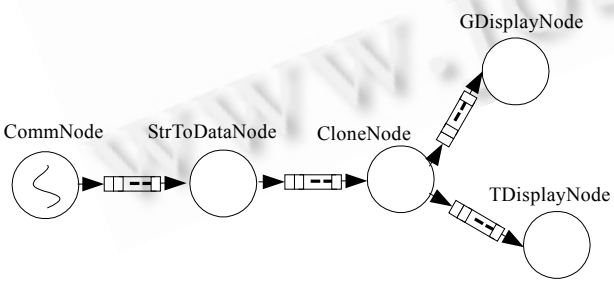

Fig.2 An example of dataflow program

图 2 数据流程序实例

试验例程:串口节点 CommNode 从外部读入字符数据;数据转换节点 StrToDataNode 对来自串口的数字符数据进行一定的变换,并将结果发送给克隆节点;克隆节点 CloneNode 对数据进行克隆,分两路输出,一路送给图形显示节点 GDisplayNode,一路送给文本显示节点 TDisplayNode;图形显示节点将输入的数据用二维图形方式动态显示;文本显示节点将数据用文本方式显示.如图 2 所示.由另外一台计算机通过串口给例程中的串口节点发送数据,发送的数据为随机数,每个随机数用 5 个字符表示.

在事件触发并发数据流模型中:给每条边的FIFO结构分配100个单位;串口读取节点为线程节点,线程优先级为 Normal.在纯数据流模型中,串口节点是一个异步读取串口数据的函数节点.其余的节点是相同的.这是一

个典型的测试系统模型程序.虽然简单,但足以体现事件触发并发数据流模型的优良特性.

3.1 CPU占用率

首先测试出各节点的平均运行时间,见表 1.对于 ECDF,还有一个参数,那就是平均消息传递时间,为316.7μs.

Table 1 The average execute-time of nodes

表 1 各节点的平均运行时间

Node	CommNode	StrToDataNode	CloneNode	GDisplayNode	TDisplayNode
Executing-Time (μs)	252.2	110.0	44.9	7215.6	346.1

图 3 是两种模型的 CPU 占用率的比较曲线.从图中可以看出:外部数据输入速率越低,ECDF 相对于 DPDF 的运行效率越高.DPDF 模型造成 CPU 占用率 100%的原因正是它的无限循环扫描机制.而 ECDF 模型在引入事件驱动机制以后,其运行依赖于外部数据的到达.在没有外部数据输入的情况下,ECDF 模型可以完全让出 CPU 资源.

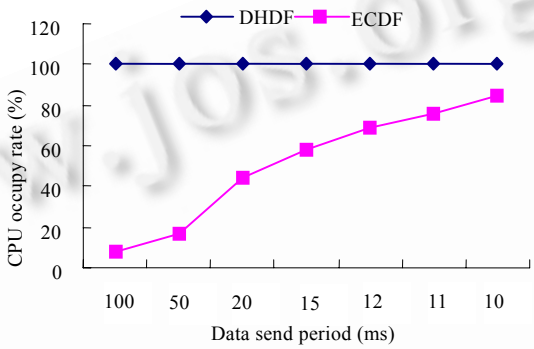

Fig.3 CPU occupy rate comparison of two dataflow models

图 3 两种数据流模型 CPU 占用率的比较

有一点值得注意:理论上,在 10ms(大于所有节点处理时间总和 8 285.5μs)处不应该出现 FIFO 溢出,但是试验结果说明,由于操作系统内部一些系统线程的调度运行以及消息的传递与处理,在外部数据发送间隔小于等于 10ms 的时候出现了 FIFO 溢出.这是由于在一定时间内系统调度其他任务时造成的.在 ECDF 模型中增大 FIFO 的深度就可以解决这个问题.

FIFO 深度是在程序运行之前确定的,用户可以根据需要来确定 FIFO 的大小.一般情况下,数据输入层的线程节点与数据处理层的非线程节点之间需要考虑用较深的 FIFO,这样可以防止外部数据输入速度高于内部数据处理速度时的数据丢失.FIFO 是用链表来实现的.理论上,它的上限可以达到占用 PC 机系统总虚拟内存的容量.

3.2 实时性

判断一个系统的实时性不仅要看任务计算结果的逻辑正确性,而且要看任务的完成时限.我们从两个方面测试系统的实时性,一是在系统任务繁重的情况下对外部数据输入的响应时间;二是在外部数据暂时高于系统数据处理时间的情况下,系统对高速数据的接受能力.由于 WINDOWS2000 不是一个实时系统,不能精确测量系统的实时性,但还是可以作定性分析的.

在第 1 种测试情况下,通过调整数据输入节点的线程优先级,能够防止在系统任务繁重的情况下因线程调度而丢失数据.这样既允许了外部程序的运行,又使数据采集任务得以顺利完成.相对而言,纯数据流响应时间较慢,最恶劣的情况就是在数据采集节点刚执行完时,数据马上到达,程序最快也要经历其余 4 个节点的运行时间总和 7 716.6μs,才能再次采集数据.这还是在不考虑中间线程被其他线程中断的情况下.

第 2 种测试情况,为了使测量的结果比较客观,减少相对误差,人为地提高了处理节点的处理时间.通过调整后 5 个节点的处理时间总和达到了 63.275ms.发送端的数据周期为 20ms.结果 DHDF 模型的数据丢失率一直保

持在 62.3%.而 ECDF 模型的测试结果如图 4 所示.其中的 FIFO 是指 CommNode 节点到 StrToDataNode 节点之间的连接.FIFO 的溢出时间明显地随着 FIFO 深度的增加而向后推移,两者之间基本呈线性关系.结果表明:ECDF 模型在处理高速突发性数据流方面比 DHDF 模型的实时性能要好.

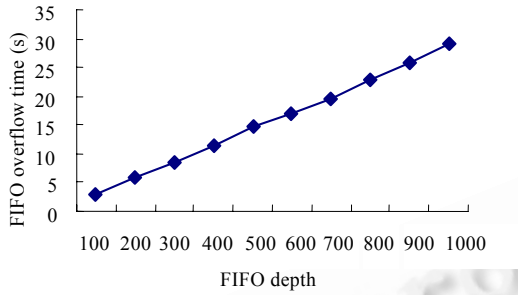

Fig.4 The relation between FIFO depth and overflow time

图 4 FIFO 深度与溢出时间的关系图

4 结 论

事件触发并发数据流显著提高了系统的运行效率与对突发性高速外部数据进行处理的实时性.该模型提供的并发多线程机制解决了数据流系统中的阻塞调用问题,使得系统可以快速响应外部事件.通过引入带有 FIFO 队列数据结构的边的连接,解决了线程节点之间的数据传输问题,为开发高效率、高性能的基于数据流的图形化平台编程平台提供了理论依据与核心支持.该模型已经应用到最新版本的面向虚拟仪器的图形化编程平台中,实践证明,该模型是有效和可行的.

试验表明,该模型也存在一定的缺点,即在引入事件触发与消息传递的同时也引入了一部分处理时间,平均为 316.7μs.在没有其他系统任务的情况下,处理周期性外部数据输入,处理速度要低于纯数据流.该模型特别适合于处理间歇、高速的外部数据输入,也适用于 Reactive 系统[8].

随着多处理器并行机的不断普及,多线程并发数据流的性能还会得到显著的提高.

致谢 浙江大学仪器科学与工程系数字技术及仪器研究所耿晨歌副教授对本文的完成提出了很多有益的建议,在此表示感谢.

References:

[1] Whiting PG, Pascoer SV. A history of data-flow languages. IEEE Annals of the History of Computing, 1994,16(4):38~59.

[2] Davis AL, Keller RM. Data flow program graphs. Computer, 1982,15(2):26~39.

[3] Andrade HA, Kovner S. Software synthesis from dataflow models for G and LabVIEW™. In: IEEE, ed. Proceedings of the IEEE Conference Record of the 32nd Asilomar Conference on Signals, Systems and Computers, Vol 2. Pacific Grove, CA: IEEE, 1998. 1705~1709.

[4] Klinger M. Reusable test executive and test programs methodology and implementation comparison between HP VEE and LabView. In: IEEE, ed. Proceedings of the AUTOTESTCON'99——IEEE Systems Readiness Technology Conference. San Antonio, TX: IEEE, 1999.

[5] Lee EA, Messerschmitt DG. Static scheduling of synchronous data flow programs for digital signal processing. IEEE Transactions on Computers, 1987,C-36(1):24~34.

[6] Bilsen G, Engels M, Lauwereins R. Cyclo-Static dataflow. IEEE Transactions on Signal Processing, 1996,44(2):397~407.

[7] Wauters P, Engels M, Lauwereings R. Cyclo-Dynamic dataflow. In: IEEE, ed. Proceedings of IEEE the 4th Euromicro Workshop on Parallel and Distributed Processing. Braga: IEEE, 1996. 319~326.

[8] Muthiayen D, Alagar VS, Khendek F. An approach to a synthesis of formal and visual description techniques for the development of real-time reactive systems. In: IEEE, ed. Proceedings of the IEEE Conference on Real-Time Computing Systems and Applications. Cheju Island: IEEE, 2000. 491~497.