计算机 体系结构 多线程体系结构 (18)

计算机科学1998 Vol. 25 №. 4

70-76

多线程体系结构现状及发展

Multithreading Architecture: Current State and Prospect

肖 刚 徐 明 周兴铭

(国防科技大学并行与分布处理国家重点实验室 长沙410073)

TP303

摘 要 This paper divides the multithreading architecture into four kinds: multiple context processor, dataflow/von Nuemann hybrid architecture, multithreading MPA node and multithreading superscalar, then surveys the their state, analyses their features, and prospects their future.

关键词 Multithreading, Dataflow, Superscalar

一、引言

多线程体系结构结合了数据流结构和传统的冯氏控制流结构，既保持了指令执行的高性能，又实现了处理器的高效率，是一种通用而高效的延迟隐藏技术。早期的多线程体系结构可以追朔到 CDC 6600 和 HEP，现今的多线程处理器中的很多技术都可以在它们那里找到踪迹。CDC 6600的 I/O 单元最早采用了多线程的技术思想，它允许10个虚拟的 I/O 处理器共享一个10级的中心流水线，处理器按轮转方式调度，每个处理器每10个周期发射一条指令。HEP 是最早的商用多线程处理器，它的结点处理器轮转地调度线程运行，每周期发射一条指令，流水线长8级。为了使流水线填满，HEP 需要大量的线程，HEP 可以支持128个线程并发执行。

如今，多线程体系结构又引起了广泛的兴趣，在 MPP、数据流处理器和微处理器领域都对它展开了更深入的研究。本文介绍多线程计算模型及其体系结构，并讨论了多线程体系结构新近的发展现状及对未来的展望。

二、多线程计算模型

多线程计算模型是数据流模型和控制流模型的结合。线程是一段顺序指令序列，线程中指令的执行按控制流方式执行，而线程作为调度的基本单位，按数据流驱动方式并行执行。

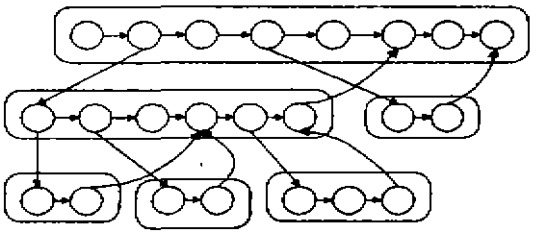

图1 多线程计算模型示例

一个多线程的计算由一组线程构成。每个线程是一个顺序任务(如指令)序列。如图1，每个块是一个线程，其中圈代表任务；水平边代表顺序关系，称为连续边。线程中的任务必须按照顺序关系从头(最左)至尾(最右)地执行。为了执行线程，给它分配一块存储器，称为活动帧，线程中的任务可以用之存储计算值。线程执行过程中可以生成其他线程。生成一个线程类似于一次过程调用，只是被调用的过程和调用过程可以并发执行。这里认为被生成线程是生成线程的子线程，而且一个线程可以生成任意多个子线程。这样，线程就由生成边连接成一个层次的激活树。每个生成边从父线程中进行生成操作的任务

实现了选择查询产生的多个集合对象之间保持原有复合关系的语义表达，避免了为保持原相应的集合对象的复合关系而使用重构这种过程性的表达方式。

主要参考文献

[1]The Object Database Standard: ODMG-93, edited by R. G. G. Cattell, Morgan Kaufmann Publishers

[2]J. V. D. Bussche and A. Heuer, Using SQL with Object-Oriented Database, Info. Sys. 18(7)1993

[3]G. M. Shaw and S. B. Zdonik, A query algebra for object-oriented database. Proc. 6th Int. Conf. Data Engineering, 1990

[4]N. Bhalla and S. Balasundaram, Operation and Queries in Object-Oriented Database Supporting Complex Objects, Information and Software Technology, 35(1)1993

[5]钟武、胡守仁. OODB 数据模型及查询代数，计算机科学，24(2)1997

[6]U. Dayal, Queries and Views in an Object-oriented Data Model. Proc. 2nd Intel. Workshop on Database Programming Languange, 1989

出发，指向子线程中的第一个任务(图中斜线)。当线程执行至最后一个任务，线程消亡。线程之间可以存在数据相关性，这是一种"生产者/消费者"关系，由线程间存在相关性的两任务之间的相关边表示(图中曲线)。任务的执行必须在所有有边指向它的任务执行结束后才能执行。如果一个线程执行到一个"消费者"任务，而相应"生产者"任务还未执行，此时线程执行阻塞。一旦"生产者"任务执行了，数据相关性得到满足，"消费者"线程就成为准备好状态，可以继续执行。

综上所述，一个多线程计算可以看作是一个由连续边、生成边和相关边连接的众多任务构成的一个有限度数的有向图。由连续边连接的任务构成线程，线程通过生成边构成激活树。

三、多线程体系结构

我们把从硬件上支持多线程计算模型的处理器体系结构统称为多线程处理器体系结构，简称多线程体系结构。构造一个多线程体系结构有两种方法。一种方法是在冯氏控制流结构上扩充对多线程并发的支持，称为多现场处理器；另一种方法是在数据流结构上扩充顺序执行的线程功能，称为数据流/冯·诺依曼混合结构，简称混合结构。现代的一些 MPA (*Massively Parallel Architecture*)中采用了支持多线程计算模型的结点来构成多处理器系统，这些结点称为多线程 MPA 结点，虽然与混合结构有许多相似之处，但由于特定环境使得它的结构和特性变化很大。多现场处理器的最新发展是将多线程和超标量技术结合，称为多线程超标量结构，它也表现出一些新特点。下面就按这四类对多线程体系结构的发展现状进行介绍。

传统上将多线程体系结构划分为粗粒度多线程、细粒度多线程和同时多线程，这种分类方法虽能部分说明各种结构在多线程执行调度特性上的区别，但无法真正把握住各类多线程结构的结构特点。

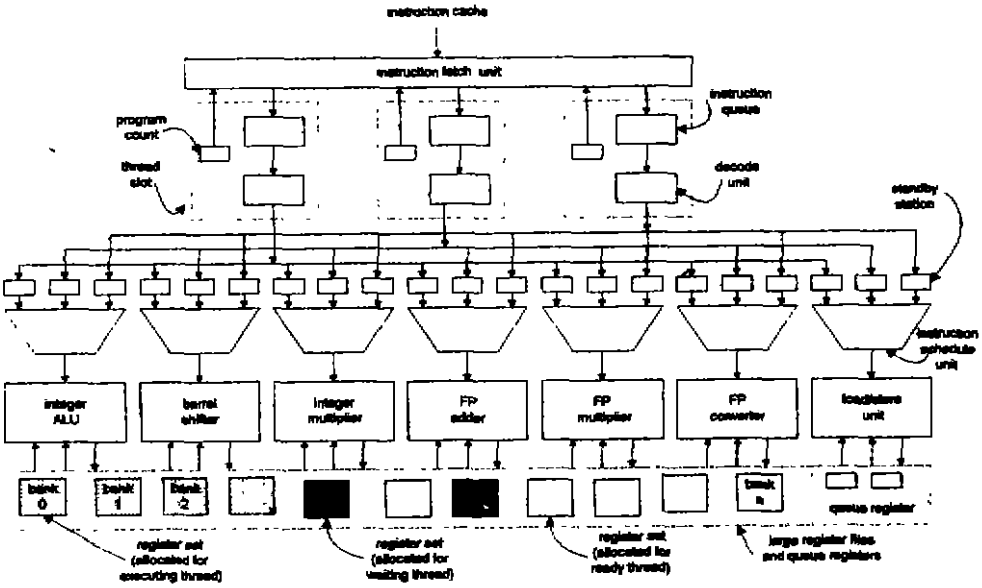

图2 三线程槽的处理器结构

3.1 多现场处理器(Multiple Context Processor)

多现场处理器直接在传统的冯氏单处理器结构中引入对多个线程并发(并行)执行的支持(如多个硬件现场，高速的硬件现场切换机制等等)[3]。

多现场处理器的设计目标有两个：一个是提高流水线的利用率。由于指令间的数据相关和控制相关造成的流水线互锁以及存储延迟会引起流水线的阻塞，造成大量资源浪费和性能下降，而多个线程共享流水线后，就可以减少流水线阻塞，从而提高处理器资源利用率和(多个线程的)整体性能；另一个是简化流水线的设计。由于多个线程按某种方式(如轮转)共享一条流水线，而各个线程之间相互独立，无隐式相关性，这就使得流水线不必考虑(或简化)由于指令之间的相关性引起的互锁，从而极大地简化了流水线控制和指令发射机制的复杂度。多现场处理器基本上都是多个硬件现场构成的逻辑处理器共享大的功能单元阵列的结构，其中好的线程调度算法和高速的现场管理机制是获得高性能的关键技术。

Hirata[4]中提出的结构是一个典型的多现场处理器结构(图2)。处理器包含几个指令队列和译码单元,称为线程槽。每个线程槽再加上一个程序计数器,构成一个逻辑处理器。取指单元和功能单元被所有逻辑处理器共享。

取指单元交叉地为各个逻辑处理器取出指令并放入它的指令队列。译码单元从指令队列中取出一条指令译码,分支预测在译码单元执行。译码后的指令采用记分板技术进行数据相关性检查,无相关性的指令发射出去,否则被锁住。

发射出的指令由指令调度单元动态地调度,分配到功能单元执行。若指令与其他逻辑处理器发射出的指令无资源冲突,此指令就立即执行。否则,冲突的指令由调度单元仲裁,被选中的指令执行,而未被选中的放入暂存站(standby station),并保持到被选中为止。暂存站实现了指令乱序执行(仍为顺序发射)。

大的通用寄存器文件被分割成多个体,每个体对应一个线程的私有寄存器文件。体数大于逻辑处理器数,这样可以支持更多的并发线程。由于暂存站可以存储操作数,所以每个体只需两个读端口和一个写端口。逻辑处理器之间可以通过队列寄存器实现寄存器级通讯。

多现场处理器结构的研究工作还有很多,如Farrens 和 Prasadh 对多现场处理器结构性能的研究,DEMUS 对调度算法的研究,等等。

3.2 混合结构

混合结构是纯数据流结构的发展。每个线程是数据流图的一个子图或一个顺序执行序列。一旦线程的第一条指令开始执行,其后的指令就不中断地执行下去。线程代替了指令作为数据流模型中的调度单位。通过在数据流中引入指令顺序执行机制,克服了数据流模型通讯开销太大和无法实现关键段等灵活控制的缺点,同时又保持了数据流模型可以开发极大的并行性和隐藏延迟的作用。

P-RISC[5]中线程描述符(continuation)由〈FP,IP〉对构成,它们在处理机中循环,又称令牌(token)。P-RISC 的结构又称循环流水线结构(图3),这是混合结构处理器的基本组织结构形式。令牌存储在令牌队列中,一个令牌出队并进入执行流水线,取出 IP 所指的指令并相对于 FP 所指的数据帧处执行,就启动了一个线程的执行。FP 所指的数据帧可以看作是线程运行时的私有寄存器。流水线是传统的取指令、取操作数、执行和结果存储四段结构。P-RISC 中每个线程按控制流顺序地执行,线程现场切换则是通过令牌入/出队列完成。

P-RISC 的指令格式和执行类似于 RISC 指令,只是在执行末尾将产生一个新的线程描述符。一般算术逻辑指令产生描述符〈FP,IP+1〉,跳转指令 Jump x 产生〈FP,x〉。P-RISC 中采用 fork 指令 fork IPt 生成线程,它将产生〈FP,IP+1〉和〈FP,IPt〉两个描述符。Join 指令 Join x 利用 FP+x 位置的存储单元实现两个线程的同步。它使此存储单元翻转,若为0则不产生描述符,若为1,则产生〈FP,IP+1〉。Join 指令的执行将使一个线程结束。load/store 指令都是分段操作,它将产生用于存储操作的消息,并结束该线程,生成另一个线程响应返回的消息,即存储操作结果。P-RISC 通过 I-structure 实现线程的同步。

P-RISC 有两种多线程执行方法。一种方法是:只要线程不因 load 和 join 指令结束,就继续执行下去即产生〈FP,IP+1〉。当线程结束时,则通过从令牌队列中取出一个新的令牌实现现场切换;另一种方法是:每周期都从队列中取一个新令牌,并把新生成的令牌放入队列中。

Monsoon[6]中,每个线程是一个独立的指令流,且每个线程拥有一个独立的寄存器集(三个寄存器)。线程的状态由计算描述符(CD)描述(图4)。其中寄存器 C 定义了线程的执行现场。它含两个指针:指令指针(IP),指向将被执行的下一条指令;帧指针(FP)指向线程的数据存储器。C 中还含一个 P 域,用于线程的 Join 同步操作。其余的 V、T1、T2、T3为线程的私有寄存器,且 V 用于纯数据流指令。Monsoon 也采用了类似的循环流水线结构,每个处理器只支

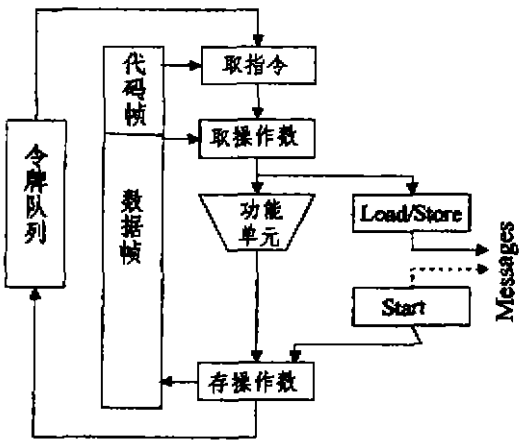

图3 P-RISC 结构

持八个线程同时执行，但可以支持32000个激活的线程。准备好的线程的CD中的C、V寄存器保持在硬件管理的线程队列中，若一个线程可以在处理器上运行，其CD就会零负载地迅速从队列中弹出，相应的线程就开始执行。激活执行的线程将连续执行，直至结束，所以可以使用临时寄存器，而且临时寄存器也不用作为线程现场的一部分在线程切换时存入线程队列中。

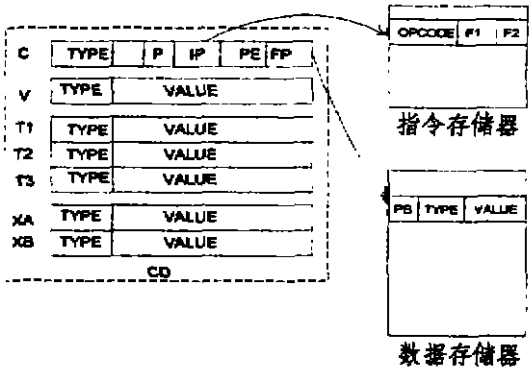

图4 Monsoon线程描述符

Monsoon中引入了fork指令用于生成线程，而采用隐含的Join操作实现线程的同步。存储操作是分段操作，从而可以充分利用多线程机制隐藏延迟。每个线程的执行可以充分利用私有寄存器提高执行效率。线程的同步通过支持I-structure的同步存储器实现。

其他的混合结构还有很多，它们的区别主要在线程的定义和性质上不同。例如，Iannucci提出了一个类似的混合结构思想，他主要讨论如何生成线程（其中称为SQ--scheduling quanta），以及线程的性质和高效支持的可能性。Bic提出一个多线程模型，此模型把数据流图被分解成SCS(Sequential Code Sequents)，类似于Iannucci的SQ。但SCS可能很大，SCS执行也可能被挂起。Hum和Gao的SAM模型对线程进行了一些限制，使其具有原子性，不可中断，称为super-actor。Super-actor有两个性质：线程不可中断，即一旦线程执行，就运行直至结束；线程间无数据共享，线程间的通讯只有通过线程结束时的结果值进行。所以，只有在其所有输入值都就绪的情况下线程才会进入准备好状态，准备好状态的线程分配一个现场就进入执行状态开始执行，直至执行结束，再发信号给其他线程。

"混合"结构还可以用于其他领域，如MASA是一个采用"混合"结构进行并行符号处理（执行并行Lisp）的处理器。MASA同前面几个结构的最大不同是它采用future变量作为同步通讯的基本结构。

3.3 多线程MPA结点

MPA是高性能计算机体系结构的发展方向，它通过把程序分配到通过高速互连网络连接的大量处理器结点上并行地执行来达到极高的性能。但是多年的研究发现传统的处理器不适合作MPA的处理器结点。这主要是因为它们不能很好地解决MPA中的两个关键问题：一个是远程取操作延迟。问题的实质在于如何处理通讯系统中的长延迟，或者说，如何避免远程取操作时处理器结点的空闲。另一个问题是同步取引发的空闲。同步取的延迟是不确定的，而且传统的同步方法开销也太大，所以人们提出了采用多线程方式的处理器作为MPA的结点。

多线程MPA结点在很多方面具有与混合结构相同的特性（如线程特性），区别主要是在实现组织结构上，即多线程MPA结点基本上都采用了同步协处理器（处理消息）和计算处理器（处理运算）的双处理器耦合结构，这是因为MPA中通讯更加频繁和重要，需要专门优化。

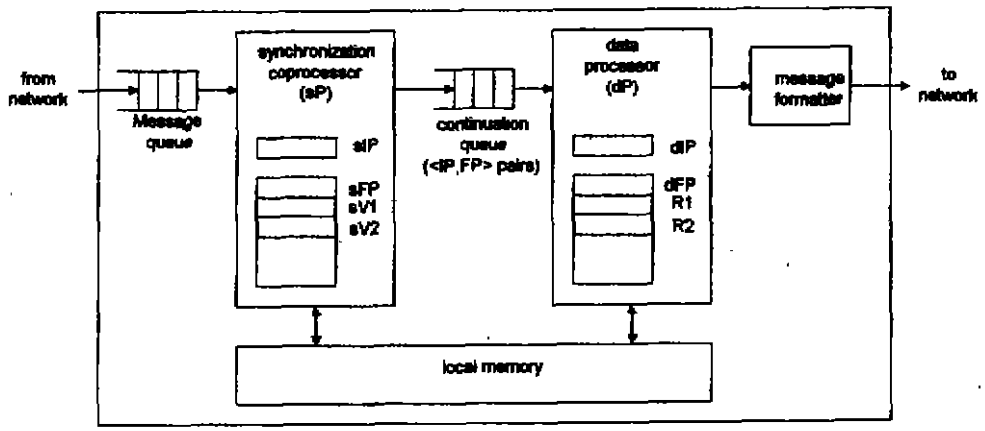

图5 *T结点结构图

*T是一个代表[7](图5),它的前身是MIT J-Machine(结点为MDP)。其中,同步协处理器(sP)和数据处理器(dP)是两个独立的处理器,都执行相应的线程(在混合结构中,实质上把这两个处理器合为一个处理器)。dP是一个常规的RISC处理器的扩充,它可以高效地执行顺序的线程。线程现场由(IP、FP)对构成,存储在线程队列(continuation queue)中。dP每次从队列中取出一个线程执行,每个线程执行不可中断。dP扩充了四条指令:start、next、rload和rstore用于启动、停止线程和远程存取。它们的执行一般是通过消息生成器(message formatter)发出相应的消息到网络中,由其他节点接收处理。sP专门用于处理网络中的消息,类似于数据流处理器,消息激活处理器执行,对消息进行相应的处理,并通过线程队列激活dP中的计算线程运行。

EARTH-MANNA系统是一个由通用微处理器(Intel i860)构成的多线程MPA系统,它的前身是MTA-Tera。EARTH系统具有同*T类似的结构。它最主要令人感兴趣是它的每个节点由两个Intel i860构成,一个作执行单元(EU),一个作同步单元(SU)。它说明了用通用微处理器也可以实现多线程模型,但需要一些技巧。

其它的多线程MPA结点都具有类似的双处理器耦合结构,如DAVRID、APRIL(Alewife的节点处理器)和TAM、S-TAM等。

3.4 多线程超标量

多线程超标量是在超标量基础上扩充了多线程机制,但由于每个线程仍是按超标量方式运行,所以多线程超标量不能简化流水线设计。它的主要目的是提高超标量处理器中功能单元的利用率,开发出更大的指令级并行性(ILP),以及利用多线程的延迟隐藏特性,隐藏存储延迟、长浮点操作延迟等各种引起流水线阻塞的延迟,提高多个线程的整体性能。

Tullsen的SMT(同时多线程)[8]是多线程超标量结构中最有代表性的研究(图6)。SMT中每个周期可以从多个激活的线程中取出多条指令同时发射到多个功能部件上并行执行。由于每个线程可以独占使用全部资源,所以单线程性能并未受到太大影响,而其他线程可以同时发射指令使得单个线程剩余的未利用发射带宽不至被浪费,多个线程之间并行执行又可以互相屏蔽互锁和长延迟操作引起的阻塞延迟。按照SMT中的术语,即可以同时消除竖直浪费和水平浪费,达到充分利用资源,提高效率和性能的作用。

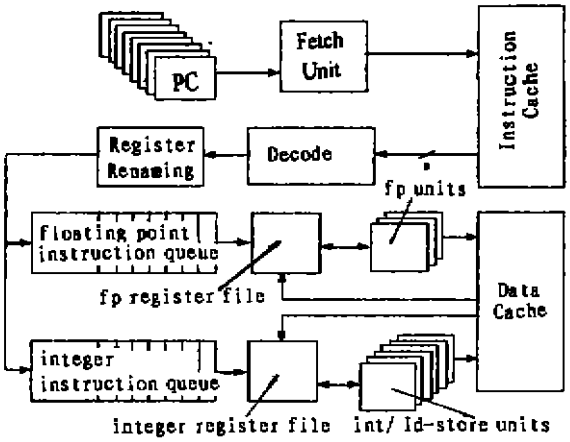

图6 SMT组织结构图

SMT是在超标量处理器上直接扩充得到的。主要扩充的硬件机制包括:多个程序计数器,硬件现场;每个线程一个独立的返回栈,用于存储过程返回地址;每个线程一套指令回退,指令队列清洗,陷入机制;每个分支目标缓冲区项扩充一个线程id域;TLB项的tag中含线程id或处理器id;更大的寄存器文件,以支持所有线程的逻辑寄存器和换名寄存器,流水线增加两级以适应增大寄存器后寄存器读写时间的延长。

Tullsen的研究表明,SMT对超标量的扩充并不很多,但得到的性能提高却十分显著。SMT给出的结论中有两点值得注意:一、对瓶颈的分析表明,因为SMT可以利用线程间并行性弥补线程内并行性的不足,所以它受分支预测、前瞻性执行、存储带宽等的影响不大,而这些对传统超标量性能影响很大。但是取指机制和带宽,寄存器文件组织和访问延迟成为SMT中的瓶颈,会限制SMT的实际性能,并可能限制其规模的扩充;二、SMT的结论主要是对单个应用形成的线程进行性能分析得到的,Jack[9]对单个应用形成的多线程进行了性能分析,指出利用多线程加速单个应用是可能的,这需要依赖高性能的多线程并行编译器的支持。但由于SMT中线程间的同步和通讯都是显式的,也就是说线程间所有的数据相关都转化成显式的同步指令操作,所以这种并行性是一种线程级并行性,介于细粒度的ILP和粗粒度的进程级并行性之间,虽然可以转化成线程间的ILP,但潜力不大,尤其是对于通用程序。只有扩充了高效的同步机制,允许线程间保持隐式的数据相关,才有可能开发出巨大的线程间ILP。

Gunther[10]也提出了一个类似的系统concurro。concurro中采用了更多的数据流技术。每个现场包含一组私有寄存器和取指/译码部件,线程之间可以通过全局寄存器也可以通过专门的寄存器通道同步

和通讯。引入了 I-structure 同步机制，并扩充了 M-structure 同步机制，并提供了专门的 fork/terminate 指令管理线程。这些机制使 concurro 中线程管理更加灵活，线程间同步通讯更加高效，从而可能开发出更大的线程间并行性。

Gulati 在一个超标量处理器 SDSP 上扩充了多线程功能，但每周期只允许一个线程发射指令，多线程交叉执行，共享功能单元。Serrano 和 Yamamoto 中也提出了一个类似的系统，称为多流超标量。

3.5 小结——比较

最后，我们将四大类多线程体系结构的特点总结成下表：

表1 多线程体系结构比较

	多现场处理器	多线程超标量	混合结构	多线程 MPA 结点
来源	冯氏控制流结构上扩充多线程并发(并行)	超标量处理器上扩充多线程并行(并发)支持	数据流结构上扩充顺序线程，数据流技术成分更多	适合 MPA 结构的处理器结点
目标	提高利用率，简化流水线设计，可能支持线程并行	提高功能单元利用率和 ILP，隐藏延迟	提高执行效率，隐藏延迟，多机多线程可以并行	解决 MPA 中的通讯延迟问题
结构	多个硬件现场(逻辑处理器)共享物理功能单元	超标量中扩充多个硬件现场，支持多指令流并发	循环流水线结构	同步协处理器和计算处理器双处理器耦合结构
调度	难点在于现场管理		难点在于多机调度	
现场切换	现场较大，切换基本上无存储移动		现场小，切换表现为现场的移动	
线程管理	一般由操作系统管理		一般提供专门指令	
同步、通讯	一般由操作系统提供，硬件可能支持		一般提供低级的指令和同步机制	
线程特点	线程较大，一般为一个过程调用或一个循环体，线程间显式地数据相关		线程较小，常是两个同步点间的一段顺序指令，线程间存在隐式的数据相关	

四、展望

多线程体系结构的优点是明显的。线程的顺序执行模式提高了执行效率，克服了数据流中效率低和无法实现关键段等问题；线程的数据流调度模式使其保持了数据流隐藏延迟、开发并行性的特点，正好补充了控制流结构的不足。

多线程体系结构也暴露出一些问题。一个是可编程性，在混合结构和多线程 MPA 中，编程需要使用专门的(数据流)并行程序设计语言，以及并行软件开发环境等的支持。并行软件开发困难的问题在这里丝毫未得到改善，这使得它们只能是专用的大型机，不可能成为通用结构。而在多现场处理器和多线程超标量中，则需要多线程编程技术和编译支持，这些方面都是现今研究薄弱的地方。可编程性差会降低它的使用性能。另一个是资源开销、设计复杂度问题。多线程结构比原来的结构更加复杂了，支持的硬件现场规模、现场切换效率、调度算法等对性能影响很大。过大的规模会使设计复杂，会降低实际可实现的性能，但规模小又不足以体现出多线程体系结构的优势。例如，并行的线程数不足会使隐藏 MPA 中远程访问的能力下降，而硬件现场不足也会使多现场处理器仍无法充分提高资源利用率。但总的来说，这些问题都是可以解决的，需要的只是研究和时间的投入。多线程体系结构的未来发展将主要在以下几个方面展开：

第一，多线程技术作为一种通用的高效的延迟隐藏技术将更广泛地同各种处理器结构结合发展出新的处理器结构。例如，有人对多线程向量处理器和多线程 VLIW 处理器进行了一些探讨。多线程可以和几乎所有的结构结合，提高它们的执行效率和资源利用率，而且结合后往往表现出一些意想不到的效果；

第二，多线程体系结构要想获得真正的成功必须克服前面提到的两个缺点。首要的是解决可编程性问题，需要对多线程程序设计语言、软件环境，尤其是多线程编译技术进行深入的研究。这是当前研究的薄弱环节。多线程的设计复杂度增加问题则促使要寻求新的硬件结构技术，支持高性能的多线程并行(并发)。

第三，传统的对多线程的研究基本上都集中在利用多线程的延迟隐藏特性上，但多线程超标量中已经表现出了可以利用多线程结构开发更大的并行性，提高处理器的性能。这方面的研究将使得多线程技术得到一次更大的发展，我们将在这一领域开展深入的研究。

参考文献

[1] Gregory T. Byrd and Mark A. Holliday, Multithreaded Processor Architectures, IEEE Spectrum, Aug. 1995

[2] Ben Lee and A. R. Hurson, Dataflow Architecture and Multithreading, IEEE Computer, Aug. 1994

[3] Kai Hwang, Advanced Computer Architecture: Parallelism, Scalability, Programmability, McGraw Hill, 1993

[4] Hiroaki Hirate etc., An Elementary Processor Architecture with Simultaneous Instruction Issuing from Multiple Threads, Proc. 19th Ann. Int'l Symp. Computer Architecture, 1992

计算机科学1998 Vol. 25 №. 4

单向非对称链路路由的研究与进展*)

Researchs and Advances in Routing with Unidirectional and Asymmetrical links

潘建平 顾冠群
(东南大学计算机系 南京 210096)

摘 要 新型单向或非对称链路为传统分组网络的互连和接入提供了更大的灵活性,但也直接影响了传统路由协议原有某些算法假设和协议机制,几乎现有的路由协议都无法直接运行在单向链路之上。本文描述单向路由短期研究的一些进展与成果以及我们在解决远期方案中邻接发现和路由生成所做的工作。

关键词 单向链路,非对称链路,路由协议

传统分组网络的路由协议总是假设传输媒体具有双向和对称的特征,无论是基于距离向量还是链路状态的路由协议,都认为从A到B的路径蕴含着从B到A的反向路径。但是随着Internet的不断发展,新型网络应用和传输服务的出现,原有假设的合理性和可行性就明显出现问题。首先,许多网络应用本身就具有非对称性。如文件下载和Web浏览等主流应用,从客户端到服务器仅存在少量的命令或反馈,而大量的数据则是从服务器下载到客户端。随着网络应用的进一步发展,这种非对称性将可能更加显著。需要指出的是,上述是指数据流的非对称性,因为单向的数据流也会伴随反向的协议控制,也就是从协议研究的角度而言,数据流和控制流仍然必须构成一个闭合回路。其次,许多新型的传输服务本身就是单向或非对称的。卫星通信的广播投递形式能够容易地构造新型应用所需的多点投递服务,并且卫星的覆盖面广、带宽大,能灵活地适应移动用户的高吞吐通信需求。但卫星通信的发射设备要远比接收设备庞大、复杂和昂贵,对小型和个体移动用户而言,仅携带接收设备更为合适。因此对于许多非对称应用可利用卫星信道提供大量的数据传输,使用其它通信设施(如地面带宽较小的公众电话网)传输少量的控制信息。这能够极大地缓解现有地面通信设施过度拥挤的局面。此外根据网络应用的发展趋势和调制编码的技术,许多新出现的用户环路传输技术,如Cabel Modem和xDSL,大多提供非对称的传输能力,一般下行带宽要远大于上行带宽。还有一些特殊的传输介质或是媒体访问控制的原因或受现有布线系统的影响,只能提供单向的传输能力。

尽管非对称的极限就是单向,但两者从协议研究的角度来讲是有区别的。单向是从图论的观点描述结点之间的有向边,传统的双向对称链路可分解成两个对称反向的单向链路;是否对称是依据链路本身的带宽分配而言的,两个容量等同反向伴随的链路可合并成一个双向对称链路。对于路由协议可以使用反向距离向量无穷大表示链路的单向,链路状态则关心是否容量对称。基于单向链路的路由协议更加困难,因为缺乏显式的反向通道。

1 对传统协议的影响

单向和非对称链路的出现,给传统网络层和运

*)国家自然科学基金,国家863高科技计划和国家教委博士点基金资助项目。潘建平 博士,研究方向为高速、高性能网络及协议。顾冠群 教授,博士导师,研究领域涉及计算机网络,分布式处理和CIMS等。

[5]Rishiyur S. Nikhil, Can dataflow subsume von Nueman Computing?, Proc. 16th Ann. Int'l Symp. Computer Architecture, 1989

[6]Gregory M. Papadopoulos and Kenneth R. Traub, Multithreading: A Revisionist View of Dataflow Architecture, ISCA, 1991. pp. 342-351

[7]Rishiyur S. Nikhil et al., * T: A Multithreaded Massively Parallel Architecture, Same to[4]

[8]Dean Michael Tullsen, Simultaneous Multithreading, Ph. D Dissertation, Univ. of Washington, 1996

[9]Jack L. Lo, etc., Coverting Thread-Level Parallelism to Instruction-Level Parallelism via Simultaneous Multithreading

[10]Bernarl Karl Gunther, Superscalar Peformance in a Multithreaded Microprocessor, Ph. D Dissertation, Univ. of California, 1993