计算机科学1997 Vol. 24 No. 5

并发软件开发方法学的研究

Study on parallel software development methodology

姚淑珍 金茂忠

(北京航空航天大学计算机系 北京100083)

摘 要 On the basis of researching the development process of parallel softwares and available development methods of parallel softwares, the paper puts out a development methoodlogy of parallel softwares, that combinates Petri theory and OO technique, giving effective support for modelling, analysing and generating parallel softwares.

关键词 Parallel software, Development methodology, Oriented-object, Petri net

并发软件系统中各节点既独立又统一,并发地完成系统功能,这种并发性导致系统执行结果常常具有不确定性,使系统开发人员很难设计测试用例,又因为并发软件常用于要求较高的环境,提供保证其功能正确、性能可靠的手段就显得尤为重要。这种手段不仅仅局限于编程阶段(相应机制有并行语言,并行操作系统和并行编译等),更应体现在分析和设计阶段,乃至整个开发过程,因此,在本文中我们将研究支持并发软件开发整个过程的软件开发方法学。

1. 并发软件开发方法学

并发软件的开发一般包括两个阶段。开发者首先要将整个问题分解为适当规模的可并行设计的子任务。为产生结构化的、可重用的和可移植性的软件,开发者应在考虑硬件细节之前,建立并发软件设计模型,并对模型进行验证。下一阶段就是将并发软件设计模型映射到目标环境上,这种映射可由支持运行系统自动完成,也可由程序员手工完成。但在实际应用系统开发过程中,很难做到设计模型与实现细节的分离[1]。例如,在嵌入式系统的设计方案制定过程中要考虑硬件平台,因为在这类系统中,无需考虑可移植性和可重用性,更重要的是系统的安全性和可靠性。即使需要开发可重用软件,开发者也常常要假设所用目标机的类型和特征,比如共享或分享内存,通讯网络连接方式等,这就要求并发软件的开发者在可移植性、可重用性和系统性能间取得折衷。因此,并发软件设计模型应包括交互作用的过程和数据结构两大成分。支持并发软件开发的方法学不仅要提供如何将整个问题分解为并发过程的原则,而且还要提供一种描述过程、数据结构及其交互作用的表示方法以及验证手段,这种验证手段在保证过程间协同工作的同时,还要防止发生死锁、异常终止等错误条件以及处理由于分解而引起的复杂性爆炸问题。由此可见,并发软件的建模与性能验证是整个开发过程中的两个重要环节。

目前应用的并发软件开发方法多源于顺序软件的开发方法,其中 LGDF 就是以数据流图为基础的建模工具[2]。LGDF 为并发的 Fortran 程序的设计需要,对数据流图进行了扩展。它将系统视为作用于共享的公用存储区的层次化的过程结构,过程由存储区中的数据来激活。数据流设计完成的同时,也勾画出了将设计转换为并发实现的步骤。为实现数据的完整性与依赖性,LGDF 还定义了数据流操作集合,这些数据流操作实际上是在编译时扩充到程序执行控制语句中的宏语句。这种方式可通过重新实现数据流操作来支持不同并行结构上的源代码级的可移植性。

对数据流图改进较大的是变换模式[3]。变换模式捕捉了软件设计期间系统的控制和实时两方面信息。它提供两种不同的变换表示:数据变换和控制变换。数据变换表示底层的系统功能,控制变换通过事件流完成与数据变换有关的活动,这种事件流可以激活和终止数据变换。每个控制变换都对应一个状态机描述,状态机的输入集合对应事件流输入集合,状态机的输出集合对应事件流输出集合。为模拟与控制变换对应的状态机需引入状态量集合,用状态量的流动来描述变换的行为实现。状态量集合反映了整个系统的状态,一个状态量往往代表多个控制

变换的状态。这种模型正好与我们在第3部分介绍的Petri网模型吻合。

在并发软件开发方法学研究中，人们越来越把注意力集中于面向对象技术，面向对象设计中每个模块或对象直接用来表示问题域中的一个对象或对象类，对象是现实世界实体的抽象，封装了设计所需的重要行为或数据结构信息。面向对象机制不仅方便问题的层次分解，而且可实现软件活动中的并发机制。方法学的最新研究成果反映了面向对象技术用于并发软件开发的趋势。PARSE(PARalles Software Engineering)[4]就是一个成功的例子。为了更好地进行系统分解和建模大型的复杂系统，PARSE在Petri网的基础上提出了过程图(process graph)表示，以提供高层结构和独立于语言的设计描述。它使用基于对象的设计方法将一个系统分解为并发过程，然后不断精化，进一步刻画系统的整体结构和所有动态特性。

无论是变换模式还是基于对象的，大多数并发软件设计方法都不同程度地引入了Petri网思想。下一部分我们将讨论基于Petri网的并发软件开发方法学。

2. Petri网方法

并发软件与顺序软件相比，其性能验证显得更为重要，不仅因为并发软件行为的不确定而且因为并发软件常用于恶劣环境。形式化描述为性能验证提供了很大方便，所以在并发软件开发方法学研究中形式化模型起着很大的作用。描述系统并发性的形式化模型主要有轨迹模型、过程代数、基于状态的模型和Petri网模型。性能验证需求和并发软件开发特点决定了形式化模型不仅应提供基于系统行为的逻辑表示而且应是可操作的。与其它形式化模型相比，Petri网更能体现这种可操作性。Petri网的并发性使它易于描述系统的并发、竞争和同步等特征。Petri网的基本成份是位置和转移，前者可用于表示系统状态，后者可用来模拟工具的行为，位置和转移间用有向弧相连，转移的输入位置用于表示转移启动的条件，转移的输出位置用于表示转移启动的结果，各位置所代表的系统状态用标记来表示。体现Petri网动态行为的启动规则是，若位置代表的条件满足，转移就可启动，启动后，转移的输入位置中的标记数等于原标记数减去位置到转移的输入弧数，转移的输出位置中的标记数等于原标记数加上转移到位置的输出弧数。这样，每个转移的启动(对应工具的激活)都会通过位置中标记数的增减反应出系统状态的变化，这种变化又会以输入位置的形式作用于相应转移(从而激活其它工具)，以上行为就构成了基本Petri网的可操作模型。

自六十年代初提出基本Petri网至今，为扩大Petri网的建模分析能力，已形成多种高级Petri网，例如，着色网、谓词网、时间网、随机网和分层网等。这些高级网不仅可较好地解决基本网所无法解决的复杂性和结构分解问题，而且所建立的体现系统行为的可操作模型不仅可完成到计算算法或数据流模型的映射，而且可借助仿真技术评价系统的动态特性，从而分析系统的预期性能。即使在编码阶段，高级Petri网也可以对Ada程序等进行验证[5]。

为更好地对复杂系统建模，目前的研究趋向于将Petri网与面向对象思想结合，其中PROTOB[6]就是一个典型的实例。PROTOB基于Petri网提出了开发离散事件动态系统的面向对象方法学。在PROTOB中，每个对象都对应一个高级Petri网，对象间的相互通讯是采用在有对应关系的端口间发送和接收消息来完成的。通过生成和连接PROTOB对象实例来获得整个系统模型。为便于分布式系统的集中管理和拓宽对象语义的表达能力，还应在PROTOB中增加优先级的机制。另外，对象间的接口关系也不需严格对应，一旦对象创建，即成为一个服务器，只需向对象的入口发送一条消息，该对象即可完成相应的服务，并发回应答消息。图1给出了基于以上思想的一个简单缓冲存储区的Petri网模型。

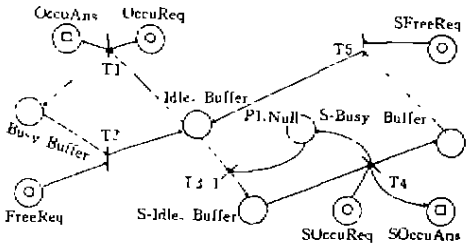

图1 简单缓冲存储区的Petri网模型

图1中，缓冲存储区接收外部对象的缓冲站申请，假如有空闲缓冲站，即分配一个，向外界发出应答。当收到外部对象的缓冲站释放请求时则释放指定的缓冲站。该缓冲存储区在任何时刻都必须保留一个空闲缓冲站以响应外界的特殊请求。转移T1接收外界的缓冲站占用请求，从位置Idle中取出一个标记放入位置Busy，同时发送消息通知外界对象已将一个缓冲站分配给它使用。转移T2接收外界的缓冲站释放请求，根据消息中所带的缓冲站号从位置Busy中

选出表示该缓冲站的标记放入位置 Idle。转移 T3从位置 Idle 中取出一个标记放入位置 S_Idle，代表留出一个缓冲站作特殊使用。位置 Idle 既是 T1的输入位置又是 T3的输入位置，T1和 T3的转移名后的数字代表优先级（其余转移的缺省优先级为0），T3的优先级高于 T1，说明在发生冲突时 T3发生。转移 T4的动作与 T1类似，T4发生后产生一个 Null 类型的标记放入 P1，这时只要 Idle 中有标记，T3就会发生，保留一个空闲缓冲站供特殊使用。

3. 基于面向对象 Petri 网的并发软件开发方法学

利用面向对象 Petri 网对并发软件开发时，开发周期分四个阶段：

(1)建模　利用面向对象 Petri 网建立系统的可执行模型，其中每一个对象都对应一个高级 Petri 网的图形表示。对象（代表软件成分的类）可以建立在层次结构中其它对象之上；它们相互通讯的方式是通过端口发送和接收消息。整个系统模型的获得是通过图形方式产生 Petri 网对象并且相互连接 Petri 网对象完成的。

(2)性能分析　面向对象 Petri 不仅可对具有并发、同步和竞争等特性的并发软件建模，而且可利用性能分析理论对模型进行分析与验证。对这种模型可应用三类方法进行性能分析。第一类是分层可达性图，它在网系统性能不变的情况下，对系统建立分层模型，从而实现问题的逐步求解。第二种是高级 Petri 网的可覆盖性树，它包含了网系统的所有可达状态或可覆盖状态枚举，为系统性能分析提供依据。第三种是关系代数框架理论，它将网系统及其相关性质形式化为关系代数中的关系，然后通过关系代数方程求解来对网系统进行分析与调整。

(3)仿真　性能分析是在系统的逻辑设计模型上进行的，为体现 Petri 网的可操作性，在该阶段还要通过考虑实现细节，不断细化改进，从而获得由 Petri 网对象实例相互连接而成的体系结构。利用仿真技术建立一个最终系统的可操作模型，该模型具有系统的所有功能，它运行在目标环境上，但并没有与设备相连，这些设备仍然由 Petri 网对象来代表。若选定面向对象语言作为仿真目标语言，则在指定 Petri 网对象对处理机的分配以及指定目标分布系统的任务后，仿真程序可由面向对象 Petri 网描述自动生成。

(4)应用生成　运用仿真程序可以完成实现细节的调整以及分配的调整。在应用生成阶段要完成与目标设备的连接，即通过适当的接口将 Petri 网对象与设备连接起来。接口的生成取决于数据定义以及翻译命令的定义，还取决于所处理事件的定义。最后，从系统的面向对象 Petri 模型自动产生系统的完整实现。

结论　尽管目前已有多种并发软件开发方法，但它们要么支持某一种特定系统，要么支持并发软件开发的某一阶段。为解决这一问题，本文从并发软件的开发过程出发，探讨了将 Petri 理论与面向对象技术结合，形成并发软件开发方法学的途径，这种方法学将支持并发软件系统的说明、设计和验证阶段。Petri 网应用上的最大争议是它在描述问题分解的难度和在描述系统规模上的限制。本文基于面向对象和 Petri 网理论所提出的并发软件方法学从面向对象、分层理论、可覆盖性和关系代数等方面努力克服这一问题，从而将对并发软件的开发和质量保证提供有效的支持。

参 考 文 献

[1]C. Pancake et al., Do Parallel Languages respond to the needs of scientific programmers, IEEE Computer, 23 (12)1990

[2]R. G. Babb, Parallel processing with large-grain data flow techniques, IEEE Computer, 17(7)1984

[3]Innes Jelly and Ian Gorton, Software engineering for parallel systems, Infor. & Software Technology, 36(7) 1994

[4]Ian Gorton et al., Object-Based Modelling of Parallel Programs, IEEE Parallel & Distributed Technology, summer 1995

[5]Mdendon Jr, W. et al., Analysis of an Ada system using coloured Petri nets and occurrence graphs, in Proc. Int. Conf. on Application and Theory of Petri Nets, June 1992

[6]M. Baldassari and G. Bruno, PROTOB: An Object Oriented Methodology for Developing Discrete Event Dynamic Systems, Computer Languages, 16(1)