计算机科学 2000Vol. 27№. 10

MAPE：一个并行系统结构性能评价模拟环境

MAPE: A MIMD Architecture Performance Evaluating Simulation Environment

王继龙[1] 方滨兴[2] 唐朔飞[1] 景晓军[2]

（清华大学计算机系 北京 100084）[1]

（哈尔滨工业大学计算机系 哈尔滨 150001）[2]

Abstract Parallel processing is a hot dot in research of computer architecture, but the high price of parallel computer limited the experiment of the research. In this paper we introduce a package called MAPE which will provide experiment environment and convenience for research on performance of MIMD architecture by simulating the running of a parallel computer. Here we mainly debate some important issues to design MAPE.

Keywords Parallel processing, MIMD, Simulation

1 引言

MIMD(Multiple Instructions Multiple Data，多指令流多数据流)计算机，这一概念来源于 Flynn[1]对计算机系统的传统分类方法，在这种结构下，系统据有多个处理器，每个处理器独立地执行各自的指令，作用于各自拥有的数据。这种结构能很好地支持高并行问题求解，同进又具有灵活、通用的结构，受到并行处理研究者的普遍重视。

国际上在 MIMD 系统的性能评价模拟系统研究方面有许多成果，如 Stanford 大学的 TANGO[2]系统、Syracuse 大学的 PAWS[3]系统、MIT 的 PISIM[4]系统、Texas 大学的 PARSA 系统[5]等都被成功地用于并行算法或系统结构的研究，上述系统研究的问题各有侧重，研究并行程度性能与研究系统结构性能通常分离，并且系统可研究的问题不具有可扩展性，同时系统程序设计语言缺乏灵活性。

本文研究的 MAPE(MIMD 系统结构性能评价模拟环境)也是 MIMD 系统结构性能评价的模拟环境。MAPE 的设计突出了集成性和可扩展性。集成性指的是可在这个环境中对系统结构性能做综合评价。

2 MAPE 系统的结构设计

MAPE 系统是由多个功能模块组成的系统。各功能模块间可协同工作以适应对系统结构综合指标的研究。每个模块也可独立运行，支持对系统结构各要素的研究。MAPE 系统的结构如图 1 所示。该结构纵向可分为三个层次：用户层、系统层和抽象机层。

用户层是用户与 MAPE 系统的接口，该层的主要功能是提供友好的系统/用户界面，便于用户完成编辑、编译、运行并行程序；描述系统结构中诸如互连网络存储体系等关键部分的特性；设置基本操作的时钟；分析模拟运行并行程序时所得到的执行踪迹数据、并对分析结果做可视化处理。用户层和系统层的接口是用户源程序、系统结构描述信息、可视化的输入数据。前两者是用户层向系统层传递的数据，后者是系统层向用户层传递的信息。

系统层据用户对并行体系结构的描述来设置模拟执行的环境；对用户书写的并行程序做串行化处理；把下层递交的模拟执行产生的数据处理成可视化要求的数据格式。系统层向抽象机层传递的信息是可串行执行的代码、模拟执行环境的设置文件、模拟执行产生的记录数据。

抽象机层模拟用户定义的系统结构，执行用户程序。同时允许挂入监测器来记录运行踪迹。

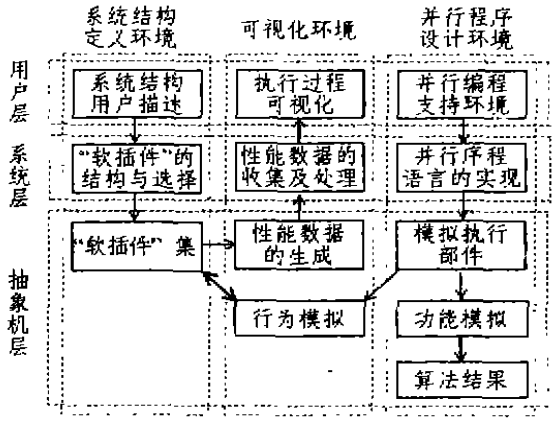

图 1 MAPE 系统结构框图

MAPE 在横向上可分为三个子系统：系统结构定义环境、并行程序设计环境和可视化环境 每个子系统都贯穿了 MAPE 的纵向三个层次，这三个子系统按如下方式协同工作：

首先，用户通过系统结构定义环境描述目标系统的特点，系统结构定义环境据用户定义设置模拟执行的环境，即组织抽象机。

尔后，用户通过并行程序设计环境提交一个并行程序。并行程序设计环境将并行程序代码转化为可串行执行的代码，并模拟执行。模拟执行有两方面任务：一是得出正确的执行结果；一是在抽象机上模拟并行执行可能出现的行为，并记录各类行为的发生和开销（各类操作开销可由用户设定），并生成一个数据文件提交给可视化环境。用户也可对该文件进行分析处理来计算各类性能指标。

可视化环境分析由并行程序设计环境提交的数据文件，对程序运行的过程给出可视表示。

以上三个子系统也可独立运行，并行程序设计环境可单独作为并行程序实验平台；系统结构定义环境可以用于单独研究系统结构要素，如互连网络、存储体系；而可视化环境只以一个数据文件为输入，并不依赖于其它两个子系统。

2.1 并行程序设计环境

广义地说，并行程序设计环境应包括硬件平台、操作系统、并行程序语言和编译、编辑、调试及性能分析工具等。一般来说，并行程序设计环境要满足便于使用、移植与并行效率高这一对相互矛盾的要求。

开发并行程序设计环境应从三方面考虑，即并行程序语言、并行编程模型和并行编程支持环境。

2.1.1 *并行程序语言* 其设计要考虑表示并行的能力，又要考虑与已有串行语言的兼容性。目前流行的各种并行程序设计语言往往以满足一方面的要求为主，有些侧重于可移植性（如 Express），有些强调在某一类机器上的效率，有些倾向于完全新的并行语言（如 Strand88），有些是在串行语言基础上进行扩充（如 Linda）。为了与现有语言兼容，可采取在 FORTRAN、C 等语言中扩充语句的途径。

在并行程序设计语言的实现上可采用三种策略：预处理、半自动预编译和并行化编译。所谓预处理是采用编程制导或"宏"处理（FORCE、PIE、Shedule 采用这种方法）；半自动预编译先对用户书写的并行程序做一次编译（PAT 和 MIMDizer 用这种方法）；并行化的编译可实现自动识别（如 Express 采用的 APAR）。

2.1.2 *并行编程模型* 由于并行计算机系统结构不同，软件并行的方法和技术也不一样。不同的编程模型适合不同的机器结构。分布存储编程面向分布存储的机器结构、共享存储编程适合共享存储的机器。目前，并行编程模型有共享存储编程、分布存储编程、分布共享存储编程、数据流并行编程等。在设计并行程序设计环境时，必须考虑应用问题本身的特点和并行机结构，以决定适应那一种编程模型。

2.1.3 *并行编程环境* 为了减小程序员在选择编程模型、并行程序设计、控制和调度等方面的困难，开发并行编程支持环境是必要的。开发并行程序包括设计、编写、测试、调试、性能改善等几个阶段。而作为一个并行编程支持环境至少要有：①并行语言支持或并行操作库函数支持；②一种或多种编程模型；③并行化工具（包括并行编译）；④并行任务调度。除了上面的一些功能外还应提供：并行程序的方案设计、并行化串行程序的源代码转换、全自动的并行编译与优化、源程序级的并行调试、控制流图的产生器、程序性能监视和性能预测、显示程序结构和数据流、文字与图形集成、CPU 利用率、存储器存放、通讯等方面的可视化。目前还没有一个并行程序环境具有所有这些功能。

2.1.4 *MAPE 的并行程序设计环境* MAPE 的并行程序设计环境由三部分组成。

①并行编程的支持环境。这部分主要提供一个并行程序的书写、编译、运行并行程序的工作平台；

②并行程序语言。并行程序语言应尽量与流行的串行语言兼容，或选用现存的一些比较成熟的并行程序语言，或用并行库。这样可减小实现的难度。并行编程环境应允许使用多种不同的并行程序语言，以适应不同的需求；

③并行行为模拟部件。可以使并行行为模拟与并行程序的模拟执行集成于并行编程语言的实现中，但这将造成并行程序语言实现的复杂性和对具体并行程序设计语言的依赖。因此，MAPE 单独实现并行行为模拟，其输入是并行程序执行时生成的操作记录文件。

2.2 系统结构描述环境

MAPE 系统能够描述并行系统的处理器能力、存储系统结构、通讯机制。通过对以上各部分说明，可以定义 MIMD 中不同结构的目标机。在 MAPE 中，对系统结构的说明是层次化进行的，其最高层次的结构如图 2 所示。

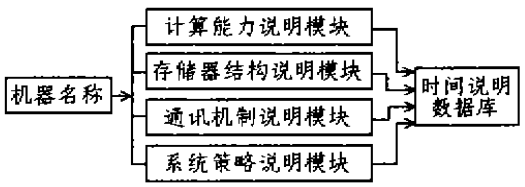

图 2 MAPE 系统结构说明模块

其中系统策略主要指处理器分配策略、存储一致性策略、路由算法、寻径方式等。对系统策略的说明与对处理器能力、存储系统结构、通讯机制的说明同时进行。下面就这二部分加以说明。

2.2.1 计算能力说明 对系统处理能力的说明，将说明所使用的语言中，系统完成主要运算操作指令的时钟，通常以机器时钟为时间单位、其层次化的说明结构如图3所示，分为串行指令和并行指令说明，串行指令设置各种运算操作的操作时间；而并行指令则是说明并行操作时间，如通常的Fork，Join及其处理器分配策略等。可根据系统的分析能力和选定的操作分类。计算能力说明中的各种操作的时间是静态值，不随系统的运行而改变，这些时间可由实际系统上运行的Benchmark测定获得。在PAPE的性能分析系统中，将根据这些说明，决定操作的时间开销。

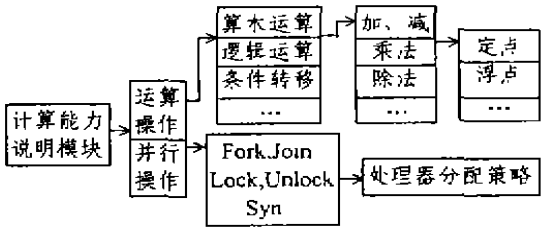

图3 计算能力说明

2.2.2 存储结构说明 该模块说明内容如图4所示。对存储结构的说明包括局部存储器结构说明和全局存储器结构说明两部分。

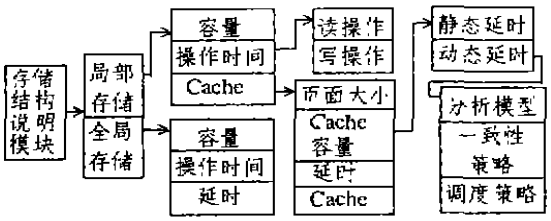

图4 存储结构说明

局部存储器说明，描述的是系统中各处理器内局部存储器的结构。包括存储器容量、操作时间及其中的Cache结构等，Cache性能是影响共享存储结构和分布共享存储结构的性能的关键因素，在此可较详细地说明。其中的延时是指由于Cache访问失败，由一致性操作及页面调度操作而引起的时间开销，这一值是动态变化的。在MAPE环境中，为了研究方便，即可将其设置成一个静态平均值，也可根据具体参数，由建立的动态分析模型分析得到。动态研究时，往往需要涉及到一致性和页面调度策略，都可在此加以说明。Cache说明中所包含的Cache说明主要用于描述多级Cache结构。全局存储器说明中的延时说明与局部存储器中的Cache延时说明相似。

对全局存储器的说明，定义了系统中各处理器共享的存储器的结构参数。

通过这些说明，可以表达MIMD中不同结构类型的并行系统的存储系统组织形式。对于分布存储结构，主要是设置局部存储器的结构。并且由于其中的Cache通常不是研究的重点，所以可设置其访问失效延时为一静态值。对于共享存储结构，则需设置全局存储和局部存储器中的Cache结构，全局存储中不涉及一致性方面的策略。对分布共享存储结构，则需要按照分布存储结构设计局部存储器，同时需要在全局存储器说明中说明作为虚拟共享存储器的容量、操作时间及延时的说明等。其中的换页调度将涉及到通过互连网络的数据传递。

这些说明参数，可用于分析存储系统对程序性能的影响。这种层次化的说明，使研究者可以根据研究及分析能力的需要，省略或增加一些参数说明。

2.2.3 通讯机制说明 对于通讯机制的说明见图5。这里的通讯主要是指通过总线或互连网络进行数据传递。在系统中，按照处理器之间，处理器和全局存储器之间以及全局存储器之间的通讯机制进行说明。

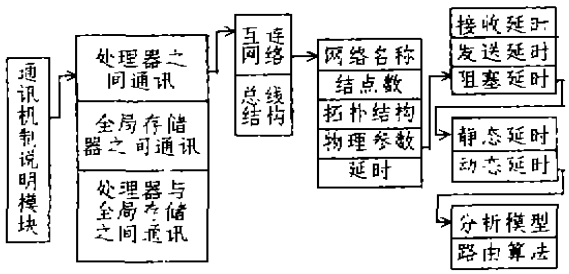

图5 通讯机制说明

通讯连接方式可选择互连网络或总线方式。对于互连网络的说明主要说明影响网络通讯效率的各主要因素的时间参数。其中的拓扑结构说明网络的连接形式。对网络连接形式的表示方式是多样的，既可用通常的互连函数形式，亦可用其它方式，如矩阵表示形式。这些表示形式，在系统中是以库的形式存在的。网络的物理参数主要包括传输率、带宽和一系列决定网络传输延时的物理参数等。通讯延时则包括数据发送延时、接收延时和阻塞延时。在这些参量中只有阻塞延时是动态变化的，在系统中可将其设置成一个静态平均值，

亦可通过动态的分析模型分析，此时往往涉及路由算法的选取、处理器与全局存储器之间和全局存储器之间的通讯说明的内容与处理器之间通讯说明的内容相似。

通过这些说明可以描述 MIMD 不同结构的通讯机制。对于共享存储结构，只需主要描述处理器与全局存储器之间的通讯。对于分布存储结构和分布共享存储结构则需描述处理器之间及共享存储器之间的通讯机制。

2.3 抽象机的组织

抽象机的组织采用与硬件机器相同的拼装技术。计算机的硬设备是由许多插件拼接而成的，而这些插件又是将许多具有独立功能的集成电路插件按插件板的设计要求组装而成的。要构造计算机系统的硬设备制造人员不必透彻了解每个集成电路插件的内部结构及实现，只要了解每个插件的使用要求、功能说明和对接头的作用等外部特征便可以按要求在设计好的插件板上安装插件，进而构造整个系统的硬设备。这样，构造计算机的主要工作是插件板的设计。

MAPE 抽象机的组织吸收了类似的思想。抽象机的各构件，如互连网络、存储系统、处理器设计成软插件，软插件具有一组外接插头，描述一组外携信息。构造抽象机的主要工作是设计软插件板，也即软插件与系统的接口。把构件库中的各类软插件组合起来插到软插件板上，一部抽象机便形成了，不同软插件的组合形成不同结构的抽象机，从而方便了对不同结构的计算机系统结构及其并行算法的研究。

MAPE 系统将提供一些软插件供研究者选用，但主要还是研究者在研究工作中根据需要自行设计各类独特软插件。软插件必须具有如下特征：

1)模块性好，独立性强，一个软插件应是可以独立存在的实体，它应该尽可能少地受外界影响。

2)正确性。必须保证软插件本身工作的正确性。

3)连接简单。要求软插件与软插件板之间的连接是简单的。

4)清晰简洁的说明。它应具有类似于硬件集成插件的功能及各项指标说明，以便于用户根据需要方便地选择。

采取这种思想，使得对系统今后的改进及扩展变得相对容易。

由于本系统面向从事体系结构研究的专业计算机人员，对于研究者需要的特殊的软插件须根据各类软插件的描述规则用 C 语言描述，要求研究者有较强的编程能力。PAPE 系统也应提供具有严格接口描述的软插件框架程序供用户填充。这样的抽象机结构便于用户的理解，而通过维护一个软插件库使得研究者之间可以共享思想。

2.4 MAPE 的可视化环境

并行程序执行的可视化是研究并行程序和系统结构性能的一个重要趋势，通过控制流和数据流模式的图形动画可使程序员直观地看到并行程序运行过程，使用户形象地看到并行程序的瓶颈，为并行程序调试和提高系统性能提供有效手段。

2.4.1 可视化的基本概念 可视化这一计算技术包含图像理解与图像综合，它是由复杂的多元时变数据集产生图像的工具，它研究人与计算机在感知、应用、交流视觉信息方面的方式方法。可视化涉及下列几个相关领域：计算机图形学、图像处理、计算机视觉、人机界面、信息处理、CAD。按用户和数据间交互方式的不同，可视化策略可分为如下三种：

·事后处理。亦即可视化过程在数据计算完成之后才开始。

·跟踪。跟踪方式下随着计算的推进可以做一些可视化工作，它要求显示中间结果，并允许监控程序在达到目的或出现异常时终止计算过程，因而具有一定交互性。

·驾驭。是充分交互的，计算与数据的可视化同时进行，并且可以根据当前计算状态对计算过程进行实时交互控制。

可视化的一般过程如下：1)计算数据的采集、组织、变换、压缩；2)几何图元的提取和可视模型的构造；3)图形的绘制与显示。

2.4.2 MAPE 的可视化环境 MAPE 可视化环境采用事后处理的策略，在并行程序模拟执行结束后，通过分析模拟执行中生成的记录文件再现执行的过程。在可视化工作中必须解决两个基本问题。首先，必须抽象出要可视化的系统的模型。其次，在第一步完成之后，找出描绘这个模型的好方案。

可视化的难点在于，我们很难抽象出可视化对象的物理模型，例如，没有人能说出一个进程看起来象什么。而且，即使我们能抽象出一个模型，这个模型也很难给我们一个直观感受。一个物理特性通常只对应一类实物，而一类计算机领域的设计则可以对应许多不同的实现，所以，一幅图像可能符合某一个人的想法，但却与其它人的想法完全不一致。

计算机科学家们在用物理实物可视化计算机问题方面做出了一些努力，例如，基于消息传递机制的多机系统的互联网络结构可抽象为一个无向图。许多系统设计工具也企图从物理实物中获得抽象模型，以实现简洁明了和良好的可移植性。但由于物理实物的特性限制是严格的，所以这类抽象模型的适用价值是有限的。

得到了模型之后，我们还需设计出表示它的图像。由于程序运行记录是具有复杂相互关系的多元实变数据，因此寻找一些图形，并能够利用它们在合适的抽象模型上描述问题的基本行为也是不容易的。

值得一提的是，如果我们能在第一阶段找出一个具有广泛适用性的直观模型，那么我们就有可能共享其它研究领域的技术和成果。

可视化的目的是为了发现尚不了解或明白难于理解的东西，而不是仅仅画出了已非常了解的东西，可视化工具必须能在无需用户参与的情况下提供有意义的显示，从而达到启发用户的效果。如果一切显示的安排都要在用户控制下进行，那么就等于要求用户对可视化的对象非常了解，也就失去了可视化的指导意义，充其量只是一个教学软件。所以可视化中的信息组织非常关键。此外，可视化要首先强调使用价值，视觉效果则次之，因此可视化环境的开发指导原则是：

1)可视化模型适合所求解的问题，例如分析一个基于控制流观点的程序，要显示的可能是进程阻塞时间、上下文切换频率等性能指标，而对于一个数据流程序，基于进程的模型将变得毫无意义。

2)具有良好的可扩展性。可视化应尽量避免固有的限制。也许对某一现有的系统来说能够同时描述16个进程已经足够了，但随着软硬件的升级需求可能会迅速改变。

3)色彩运用得当。色彩运用要掌握三个原则：①增加信息密度；②强调关键点；③增强可识别性。虽然色彩是有吸引力的，但只有在能够增加信息量时使用色彩才是合适的。应尽量避免使画面看上去象一个调色板。

4)具有良好的可交互性。试图用一幅画面反映出所有信息通常是不可能的，一个计算机应用问题通常是复杂的，有多方面的因素要考虑，可视化应能够帮助用户决定下一步需要哪些信息，最起码要使理解一个可视化的复杂程度低于理解问题本身的复杂度。从下面三方面着手可使可视化得到优化：①从不同角度反映问题；②使观察问题的层次能够改变，例如可以从进程级下降到基本部件级或上升到处理器级；③对于特定的角度和层次，能有选择地反映各类性能数据。

5)能提供有价值的图表显示。可视化是确认可视化对象的手段，把可视化对象的有关数据以图表方式显示出来是最简单明了的方法。这种方法把可视化对象及其性能数据同时展示给用户。为此，可以提供一个交互机制，允许用户用鼠标请求图表显示。

6)具有缺省显示，且能提供足够的有用信息。为了产生一个显示，不应要求操作者具备复杂的知识或必须从一大堆参数集里进行选择。因为可视化的性质是指导性的，如果让用户来选择显示内容，那么受用户本身对问题理解程度的限制，这种选择可能是杂乱无章的，所以一个可视化工具应提供一组缺省显示，使得它们在大多数情况下能为研究者提供足够多的信息。

7)避免罗列底层细节，在设计可视化工具时，经常有显示底层细节的倾向。但是，用户使用可视化工具时就好比看一块手表，如果揭开盖从背面看，看到的是许多精心选择的齿轮、金属杆、宝石和发条。看着这些机械的运动也许会觉得有趣，但却无法看着这些齿轮而知道时间。对于手表，用户只希望有一个简单的时间显示，而对如何为这些齿轮布局不感兴趣。同理，可视化工具如果过多渲染底层细节，虽然可能有好的视觉效果，但必然会给用户增加许多负担去理解它们。

8)控制力求简单。尽可能减少可视化工具中的选择按钮。在一个可视化工具中可能设置了许多选择，但要保证仅有少数相关选择是可见的。减少选择将使控制更加简单，因此产生有用的结果也更加容易。如果一定要提供某些用户一种机制来选择许多控制，那么也要把这些控制隐藏起来，只在需要时显示。

4 结论

本文对MAPE系统结构的设计具有以下几个特色：

①力争使对MAPE系统结构的设计体现出模块化思想，即将一个庞大的系统划分成尽量小的独立的功能模块，从而使整个系统具有可编程性。特别地，在抽象机的组织上提出了"软插件"思想；在系统各模块之间的接口定义中尽量地采用数据接口，从而增强了系统的可扩展性。

②采用系统结构定义与程序设计相分离的结构，使程序设计语言的选择变得相当灵活。国际上做性能评价模似系统的一个流行方式是，提供一个带系统结构定义语句的语言，系统结构描述和算法描述都在用户程序中进行。这种方式方便了系统的开发，但对系统的可扩展性造成了限制：完全限制了程序设计语言的选择；只能评价系统结构说明语句能力之内的几种结构，这两点对系统开放性的影响是致命的。

③MAPE以"软插件"方式组织抽象机结构，"软插件"与系统只存在数据接口，"软插件"的具体实现对系统透明，因此增强了对系统结构的描述能力。

参考文献

1 Flynn M J. Very High Speed Computing Systems. Proc. of IEEE, 1966, 54: 1901～1909

2 Davis H, et al. *Multiprocessor Simulation and Tracing Using TANGO*. ICPP'91, 1991. Ⅱ-99～107

3 Pease D, et al. PAWS: A Performance Evaluation Tool for Parallel Computing System. IEEE Computer, 1991 (Jan.): 18～29

4 Wills D S, Dally W J. Pi. A Parallel Architecture Interface. Int'l Parallel Processing Symposium'92. 1992. 345～352

5 Shirazi B, et al. PARSA: A Parallel Program Scheduling and Assessment Environment. ICPP'93, 1993. Ⅱ-68～72