大规模并行计算机
—CONNECTION MACHINE

【美】Daniel Hillis 著
严挹非　刘尚德 译

南京大学出版社

大规模并行计算机
——CONNECTION MACHINE

[美] Daniel Hillis 著

严挹非　刘尚德 译

南京大学出版社

1992·南京

(苏)新登字第011号

大规模并行计算机
——CONNECTION MACHINE

[美] Daniel Hillis 著
严挺非　刘尚德 译

南京大学出版社出版
(南京大学校内)
江苏省新华书店发行　江苏省丹阳新华印刷厂印刷

开本 787×1092　1/16　印张 9.5　字数 234千
1992年9月第1版　1992年9月第1次印刷
印数 1—6000
ISBN 7-305-01534-2/TP·44
定价：6.80元

简　介

本书是当今大规模并行计算机潮流的代表人物、美国 THINKING MACHINES 公司的创办人和首席科学家 Daniel Hillis 在 80 年代中期的论著。作者和其他人所致力研制的这种机器，旨在解决常规的串行机在一个合理的时间里所不能解决的问题。

目前常规计算机的特点是顺序的串行操作，这些操作在一个单独的、功能很强的中央处理器和一个独立的大容量存储器之间进行。相反的，Connection Machine 则是一种通用的并行机，其中数千乃至数百万个处理器对同一综合问题的不同方面或不同部分同时进行处理；每个处理器都拥有它自己的一个虽小却够用的存储器，两者集成一体。

Connection Machine 计算机中所有的处理器都按所需的方式相互连接起来，其编程策略要求这些处理器的连接方式与所要处理的问题的自然结构尽可能紧密地相匹配。另外，其连接方式既能在开始处理时就确定好，也可以在处理过程中条件变化时动态地加以重配。

本书探讨的是这种计算机的结构以及如何用现有的技术来实现这一结构。它涉及硬件和软件两个方面，以及在设计过程中最终把它们结合起来。本书还提供了一种以LISP为基础的编程语言，介绍了一个实际样机，并讨论了各种有效的数据结构、分配策略，颇具哲理地探讨了 Connection Machine 物理学以及计算机科学可能发生的变革。

本书在出版后的五六年间，大规模并行计算机有了惊人的发展。Connection Machine 在这个潮流中越来越成为占主导地位的领先的机器，可以说，它已成为当今大规模并行计算机的代表了。因此，将本书视为大规模并行计算机的代表作是恰如其分的，它无疑对我国从事计算机科学及其应用研究的科技人员及高校师生是一本很有启发、很有理论和实际价值的书。

该书中译本所增加的二篇附录，可以说是对原文极重要的补充。对 CM-2 和 CM-5 两种结构的介绍，不仅大大丰富了对 Connection Machine 的实际认识，而且向读者提供了作者近期在 Connection Machine 设计思想上的重大进展和突破。

原版序言

本书由麻省理工学院出版社(MIT Press)出版，它是计算机科学、工程方面的最佳博士论文年度评选的产物。该项评选是美国计算机协会(ACM)和麻省理工学院出版社于1982年联合发起的。

我们出版这套优秀博士论文的系列丛书，旨在肯定那些在评选中获得优胜的论文同时具有出版的价值。根据ACM评选委员会的鉴定，"The Connection Machine"这篇论文具有很高的质量，在这套丛书中它应加以特别的肯定。

Daniel Hillis博士在麻省理工学院，在电子工程和计算机科学教授Gerald Sussman指导下写了题为"The Connection Machine"的论文，它被提交1985年度的评选。ACM评选委员会推荐出版这一论文，是因为Connection Machine体现了一种对经典的冯·诺埃曼(von Neumann)结构的变革，它使人工智能方面雄心勃勃的理论向实验分析、试验更靠拢了。Connection Machine是一种通用的大规模并行处理机，其处理过程和存储器结合在一起并遍布所有的存储器。这样，表达数据所进行的存储器分配将自动地分配这些数据的运算处理。处理器单元之间的通讯是从"邮包-交换通讯"编程器中提取出来的，这使Hillis博士得以发展一种LISP的低级扩展，用它为Connection Machine抽象各种困难的问题，进行编程。

美国计算机协会博士论文评奖委员会主席
John R. White

中译本序言(代)

——献给中国计算机界的朋友们

超级计算机设计师们所追求的目标，如果用三个T(Trillion，万亿)来表达，可以说是再恰当不过了：每秒一万亿次的运算速度，一万亿个字节的存储器容量和每秒一万亿个字节的数据通讯能力。这每一项目标几乎都是现有的超级计算机性能的一千倍。

1992年，最权威的超级计算机公司——CRAY RESEARCH 将开始销售它的下一代计算机 Y-MP/16，它在通常情况下的运算速度为 10 Gflops， 峰值速度约达 16 Gflops。然而，这个速度也只是 teraflops 计算机(万亿次计算机)的百分之一。

尤其不幸的是，Y-MP/16 标志了一个 CRAY RESEARCH 时代的结束，因为Y-MP/16之后的下一代超级计算机的方案之争导致了这个曾经是超级计算机发源地的公司的分裂。Seymour Cray 这位超级计算机的鼻祖、CRAY 1 和 CRAY 2 的设计师决定 CRAY 3 采用速度更快的砷化镓芯片来提高 CRAY 机的速度。然而，制造这种芯片相当费时、成本极高。结果，他不得不在一年多前关闭了他的 Wisconsin Workshop，并成立了一家新的 CRAY COMPUTER 公司。在此之前，CRAY X-MP 和 Y-MP 系列的年轻设计师陈文卿 (Steve S. Chen) 也因公司的董事们认为他的下一个设计过于雄心勃勃而离开了公司，去另建了一个超级计算机系统公司(SSI)。与此同时，留在 CRAY RESEARCH 的设计师们也开始背离 CRAY 的传统，他们开始致力于研制配置数千个处理器的大规模并行计算机(Massively Parallel Computer)。

这种突如其来的分道扬镳，既反映了在超级计算机结构设计上的一些重大分歧，也让我们看到越来越多的人把实现 3T的目标寄托在大规模并行计算机上了。我们如果回顾一下超级计算机发展的历史过程，就不难理解 CRAY RESEARCH 这个时代的结束和大规模并行计算机兴起的必然了。

在计算机问世后的数十年间，计算机设计始终沿袭着最初的顺序操作方式，即串行方式。提高计算机的速度一直是意味着加快它的时钟，也就是减少每条指令之间的时间。到了70年代初，串行运行速度的提高显著地减慢了。因此，Cray 和他的设计组成了在这个瓶颈周围探索一条迂回道路的奇才。他们把器件更紧密地组合在一起以最大限度地减少电讯号的传递距离，采用新的电路散热技术，同时 Cray 也跟在别人后面开始采用一些有限的并行化步骤。他把一个作业加以分解，不再让 CPU 顺序去做一个作业的每一步，同时把 CPU 分成了若干相互协作的子单元，并把这些子单元装配成一条作业线，这就是我们已熟悉了的流水线方式。这一方法导致了向量处理，它可以对一个有序数组或向量中的每个元素同时进行类似的操作运算。

CRAY RESEARCH 的另一名著名设计师则致力从另一个方面来提高计算机的速度：增加处理器的数量。1982年 CRAY X-MP 诞生了。它基本上是把 2 个 CRAY 1 处理器捆

在一起，尽管两个处理器是共享一个公共存储器的，但是它们能执行不同的指令流。陈文卿虽不是第一个采用多处理器的人，但是当他把两个处理器联起来处理同一个问题时，他的X-MP成了第一台超过Seymour Cray设计的超级计算机。

这种多向量处理器和一个中央存储器联结在一起的结构成了当时占统治地位的超级计算机设计。同时，它也预示了一个时代的到来，那就是超级计算机的研制走出了少数几个专门的研究中心(即几个美国国家实验室)，进入了市场。加速这一进程的是美国国家科学基金会决定资助五个以大学为基地，致力于超级计算机应用的研究中心。数以千计的研究生和数十个公司在这些研究中心第一次尝到了超级计算机的味道。他们从那里获得了高速度，更重要的是他们利用超级计算机把数据转化成了一些十分诱人的、信息丰富的图像，从而使他们的成果"视觉化"了。科学家们用这些技术看到了过去无法看到的那些场景：风暴内部的景观、分子链的排列、飞机机翼顶部的气流状况等等。视觉化开始把计算技术变成了一种规范化的科学方法，它同时也对计算机的能力提出了更高的要求。

与此同时，超级计算机的制造者们之间的竞争也开始白热化，特别是日本最大的三家电子公司——富士通、日立和NEC都加入了这场角逐。这些厂家本都是复杂电路设计的高手，他们甚至比CRAY更倾向于向量处理和流水线方式，因此他们造出了速度更高的处理器。眼下居于领先地位的是NEC，它生产的SX-3超级计算机中每个处理器的时钟周期为2.9毫微秒，峰值速度达5.5 Gflops——明显地比时间周期为4.1毫微秒的CRAY 2处理器快。富士通目前生产的VP-2600据称可达3.2毫微秒的时钟周期和5 Gflops的峰值速度。但是无论NEC还是富士通的设计师们都怀疑他们是否还有可能再大幅度地提高处理器的速度。NEC公司里号称"日本的Seymour Cray"的Tadeshi Watanabe就他在SX系列上所做的工作说："坦率地说，我们已处在了超级计算机设计的十字路口。"他还说："下一代SX机，也许我们还能搞到更高密度的双极型芯片，但是再下一代我就不知道了。"主持富士通公司逻辑设计工作的Keiichiro Uchide也认为，用传统的设计"要取得1个毫微秒的系统周期时间将非常困难。"然而，放弃经典的设计并非轻而易举的事，要把过去为向量处理器写的大量软件移植到大规模并行计算机上去也要付出昂贵的成本。尽管一些专用的编译程序能够帮助用户把程序修改成向量机所需的形式，但是为较大规模的并行机所用的这类编译程序还很少见。因此，日本的制造者们和Cray不约而同地寄希望于沿用老的设计，使用砷化镓芯片取代硅芯片，以取得更高的速度。Cray打算把16个砷化镓处理器联起来，但是砷化镓芯片却是因故障率高而昭著于世的，因此这种选择实在是一场代价昂贵的赌博。像日本那样有足够持久的财力和技术力量来追寻砷化镓或别的新型材料的厂家在世界上还少有。甚至日本人也只是小心翼翼地在这样做。富士通、NEC和少数几家其他的计算机公司在日本通产省制定的十年发展规划的资助下，已经制造了几种不同的样品，但研究人员说，要把这样的高速器件组合起来用到超级计算机里去还要再有十年的时间。即使如此，根据Watomabe的观察，"为了获得更高的性能，处理器的数量也必然要大大地增加。"在这场为提高速度而进行的漫长竞赛中，采用砷化镓的计算机也许会赢得20 Gflops的成就。即使在经费上为发展其他新的一代计算机找到足够的支持，这些未来的机器很可能也只是比现在的超级计算机快十倍——离teraflops这个目标依然相距甚远。

现在，我该把话题回到本书的作者Daniel Hillis和他的Connection Machine上来了。这种完全摆脱了冯·诺埃曼经典设计的大规模并行计算机在1987年问世后不仅

站住了脚，而且取得了举世瞩目的迅猛发展。事实上，在过去的一二年里，有一点几乎已经成了大多数人的一个信条，那就是：只有大规模并行计算机——这种具有数万、数十万个处理器的机器——才能取得 teraflops 的性能！只有它才能达到我在本文开头提出的那三个T的水平！

IEEE 召开的超级计算机大会是最明显的历史见证。在头两次大会上，Seymour Cray 和 CRAY RESEARCH 的董事长 John Rollwagen 始终是大会的基调发言者。但是，到了第三届超级计算机大会时，Danny Hillis，这位最大规模的并行机设计师，即本书的作者，成了大会上引人注目的明星。当 Hillis 问台下的听众有谁不同意他的主张时，举手的人廖廖无几。

大规模并行机是把大量的微处理器集中在一起来获得高速度的。它把逻辑部件、存储器和通讯机构集成一体。通常，单独的微处理器还是比 CRAY 1 里面的处理器要慢，但是当我们把一个问题分解成许多件的时候，这个处理器的集群就能以极高的速度来解决这个问题了。

大规模并行计算机的支持者们已经看到了它的许多优点。INTEL 公司的 Rattner 指出，这些微处理器芯片轻轻松松地赢得了速度。而且，由于已有大量的成品芯片，因此采用这些成品微处理器芯片相对还比较便宜。当然，要把如此大量的处理器恰当地联系起来，也产生了一系列设计上的问题。

尽管现在还没有一个标准的大规模并行计算机的设计，但是 Hillis 和他的 Connection Machine 计算机已经在事实上成了这个大规模并行化潮流的领头人和象征。Connection Machine 是一种高度创新的体系结构。Hillis 以人脑和它大量的相对较慢的神经细胞为模型，设计了这部可容纳 64000 多个处理器的计算机。在这部机器里，数据分布在所有成对的处理器/存储器胞里，同一个指令将扩散到所有的胞，由此来控制这些处理器对它们各自的数据同时进行操作。因为它们在某一时刻在做同一件事情，因此不需要任何附加的协调。这种方法是单指令流多数据(SIMD)的设计。

Hillis 的大多数竞争者都在设计多指令流多数据(MIMD)的机器。MIMD 方式非常像一出芭蕾舞剧，多处理器分别执行一个独立任务的各个部分，就像许多舞蹈演员以不同的舞姿同台演出一台芭蕾那样。MIMD 并行机比较灵活，但另一方面又比 SIMD 并行机复杂，特别是编程相对困难。

值得注意的是，Hillis 正在结束这场SIMD和MIMD的纷争。他在最新推出的 CM-5 结构中把这两种方式结合起来了，使 Connection Machine 成了随时可在 SIMD 和 MIMD 两种方式之间加以变换的高度灵活的并行机。Hillis相信，最终会有更多的人转而成为他这种方法的信奉者，Connection Machine 将赢得这场争夺 teraflops 竞赛的胜利。他指着 CM-3 说："我对我们将拥有第一台真正的 teraflops 计算机而感到满意。"这一设计意味着 Thinking Machines 具备了制造一系列解决不同规模问题的计算机的能力。

这里我还想特别指出，这种新型的并行机对中国有着特别的意义。这一崭新的技术对于中国的国情也许是再合适不过了：

首先，在美国、日本和其他发达国家已为过去的计算机软件付出了巨额的投资，这是

应用这一新的设计时的瓶颈，它也许会迫使那里的用户在一段时间内继续使用他们原来的机器，即使他们能得到大规模并行计算机。然而，对中国来说这种转变的痛苦要少得多。

第二，十多年来，微处理器的速度每年不断提高，而价格成本却逐年降低，存储器亦是如此。它们发展进步之快令人瞠目。大规模并行计算机一旦设计成功之后，它的性能将随着这些微处理器与存储器自然地年年提高，价格也自然地年年降低。

第三，这种新型计算机的设计，本质上是在寻找一种最有效的方式来驱动一群微处理器，它实在是一项投入大量脑力活动的工程，而中国具有如此丰富的脑力资源——充分发挥这一优势，何惧在这一领域不能迎头赶上。与此相反，向量机那样的传统超级计算机的设计却以越来越高级的中央处理器为中心，这就需要非常先进的制造能力、需要强大的基础工业作后盾——它恰恰不是中国的长处。

我的好朋友严挹非看到了这本书以及该书所论述的大规模并行计算机将为中国提供一个极好的机会，去赶上和超过西方并在这项激动人心的新技术的应用中取得自己应有的地位。他和刘尚德先生为这本书的诞生投注了极大的热情和努力。我作为最初建议他将此书译出的人在看到这项工作业已告成之际感到无比欣慰。我由衷地希望，这本书的出版将使中国的计算机设计者们能有所受益，我并且深信他们将会设计出更好的计算机来。

美国大陆石油公司研究部
研究员，地球物理学家
汪贤驯(Shein S. Wang)博士
一九九一年十一月，Oklahoma.

目　录

第一章 概 论

1.1 我们想制造一种思维机器

总有一天，也许就在不久的将来，我们会创造出一种具有某些人类智能的机器——思维计算机。设计这种机器必然面临的众多问题之一是要求以远远高于传统计算机能达到的速度快速地处理数量庞大的信息。本书将介绍一种新型的计算机，称为 Connection Machine(以下简称 CM 机)。这种机器通过许多简单而相同的处理器/存储器胞(例如可以有一百万个)之间的相互配合来完成其操作，由于它的操作是并行的，因此其速度远远高于传统的常规计算机。

现有的机器速度太慢

尽管我们尚不了解人类智能的结构，但目前的计算机结构缺少智能却是显然的。现在我们来考虑这样一个特殊问题：要求用一个句子描述图 1.1 所示的图。我们几乎可以毫无

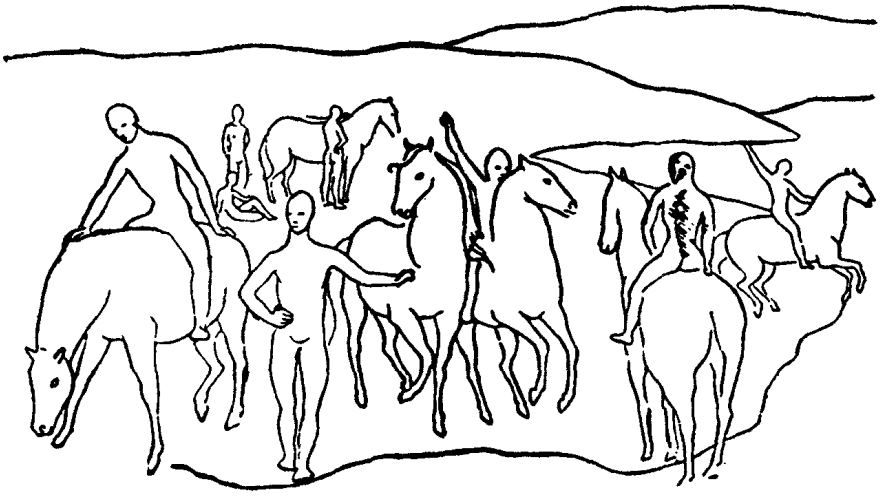

图 1.1 饮水地，巴勃罗·毕加索，1905

困难地说："它是一群人和马。"描述这样一幅图对于我们人来说是轻而易举的，但对一台现代数字计算机，它却是一件几乎不可能完成的事。给定这样的图像，计算机首先要处理图画上几十万个像素的视频信息以便找出线条，连接区域和阴影的结构，然后计算机用这些线条和区域来构造这些物体形状和它们的三维空间位置的某种三维模型。接着，它与已知的图形库比较对照这些物体，从而识别出脸、手和重迭的山丘等等。即使这样，也还不足以了解这幅画的含义。理解这幅画需要大量有关现实世界的常识性知识。例如，为了识别作为山丘的简单起伏线，就必须想到山丘的形象；识别马的尾巴，就必须预先想到它位于马的尾部。

即使计算机在其存储器内存储了这样的信息，若没有首先考虑并剔除许多可能相关的

信息，诸如人常坐在椅子上，马背上可以放马鞍以及毕加索有时用多种透视来表现景色等等，计算机有可能还是找不到这样的信息。结果是：这些事实对这个特定的图像的解释是不相干的，但计算机却没有办法在考察这些事实之前，就预先除排其相关性。计算机一旦识别出图中的物体，就可以形成一个句子，给出一个简明的描述。但这涉及到哪些细节是有用的，并要选择一个恰当的观点。例如，把这幅画描述成"两座山丘，部分被生物遮挡"恐怕是不会令人满意的，尽管这样的描述可能是准确的。

我们了解了上述工作的步骤，因此可以着手编写程序，产生这幅图画的一句简单的描述。然而，这个过程是冗长的，最后生成的程序运行的时间也极长。人脑几乎毫不费力完成的事对现有的最快的计算机也可能需要几天才能完成。这些电子巨人在做几行数的加法时胜过人脑，就如同我们人脑在符号思维处理方面胜过它们一样。

计算机和人脑的比较

计算机的不足究竟在哪里？部分原因是因为我们尚未完全了解思维的算法，另一部分原因是速度。有人可能会猜测计算机速度慢的原因是电子元件的速度比人脑的生物元件速度慢得多。然而，情况并非如此。晶体管能够以毫微秒级的时间开关，而神经的开关时间要毫秒级，电子元件的开关速度要比神经快一百万倍。似乎更可能的原因是，人脑的神经细胞数量比计算机拥有的晶体管个数多。但这仍然未能解释计算机和人脑处理图像信息速度的巨大差异。在我们所能了解的限度内，人脑有10^{10}个神经细胞，每个神经细胞每秒开关不超过一千次。因此，人脑每秒大约有10^{13}次开关动作。而现代数字计算机可以拥有多达10^{9}个晶体管，每个晶体管通常每秒能开关约10^{9}次。因此，现代计算机每秒总的开关动作将高达10^{18}次，或者说，是人脑的十万倍。从而，计算机纯粹计算的能力，必然远大于人脑。然而，我们知道实际情况正好与此相反。那么，以上的计算错在那里呢？

1.2 传统的计算机结构反映了一种过时的假设

计算机慢的原因之一，是它们的硬件使用效率极低。当今大型计算机每秒实际动作次数不到1.1节中计算的动作次数的四分之一。低效率的原因有技术的，但更主要的是历史的原因。当我们采用了各种适合于当今情况的假设时，就从不同的技术角度改进了当代计算机结构的基本形态。这里介绍的计算机—— Connection Machine 计算机是非常切合当今计算机技术的机器结构，并且我们希望它也非常适合思维计算机的要求。

现代大型计算机约有1平方米的芯片面积。这样大小的面积，容纳近 1000 兆个晶体管，组成计算机的处理器和存储器。这里有趣的一点是，处理器和存储器两者都由同样材料制成的。但是，未必总要这样。冯·诺埃曼和他的同事们进行第一台计算机设计时，他们的处理器就由速度相对快，但价格昂贵的开关元件——真空管组成的。相反，存储器都是由速度较慢而价格便宜的元件——延迟线和存储管组成的。结果形成一种使昂贵的电子管部件尽可能忙碌工作的二部结构。我们称这种存储器为一方，处理器为另一方的二部结构为冯·诺埃曼结构。它是当今绝大多数计算机构成的方式。这种基本设计方法如此成功，以至当今大多计算机设计仍在沿用它，尽管存储器/处理器分开的技术理由现在已不再被认为是合理的了。

存储器/处理器分开导致低效率

在一台大型的冯·诺埃曼式计算机中，它一千兆左右的晶体管中几乎没有一个是任何时刻都在进行有效工作的。因为在这种机器中，几乎所有的晶体管是在它的存储器部分，而在任一给定的时间里，这些存储器只有很小一部分地址被存取。这种二部结构让专做处理工作的芯片始终保持在特别忙的状态，但是这只占所有芯片的百分之二或百分之三，其余的百分之九十七处于闲置状态，而经过处理、封装好的芯片，每平方米要上百万美元，这是一种昂贵的资源浪费。我们要是测算计算机的其他成本，譬如一千米导线，其结果必将是一样的。多数硬件是在存储器里，因此，多数硬件在大部分时间里无事可做。

当我们制造大型计算机的时候，这个问题甚至变得更糟。机器中存储器的容量是成比例地直线增加的，而处理器规模的增加却毫不显眼。结果导致我们制造更大的、芯片更多的计算机时，或者说，我们要在每单位面积芯片上塞入更多的晶体管时，这些计算机的存储器与处理能力的比就更大，从而效率就更低。这个低效率与我们使处理器速度有多快没有什么关系，因为计算机的工作时间有很大一部分为处理器和存储器之间的数据传送所占用了。这称为冯·诺埃曼瓶颈。我们制造的计算机越大，瓶颈效应就变得越严重。

1.3 用并行机解决问题

显而易见的回答是，摆脱冯·诺埃曼结构，而制造一种更为均匀的，存储器和处理器工作相结合的计算机。如今，制造具有几十万，甚至数百万个微型处理单元的计算机已非难事。这种计算机的纯计算能力比最快的传统计算机高好几个数量级。现在的问题是如何使这个纯计算能力跟实际应用所关心的问题配合起来，又如何安排硬件操作。如何把实际应用问题分解为几十万个能够并行执行的部分？如何协调上百万个处理单元的活动，去完成单一的任务？作为对这些问题的回答，我们设计了 Connection Machine 计算机。

图像处理：每个像素一个处理器

例如，在图像处理中，我们知道能够有效地使用处理单元的二维连接网格进行二维滤波运算。在这个应用实例中，把图像的每一个像素信息存储在各自的处理单元内是最自然的。1000×1000 个像素点的图像需用一百万个处理器。在这种情况下，每一步计算可以在

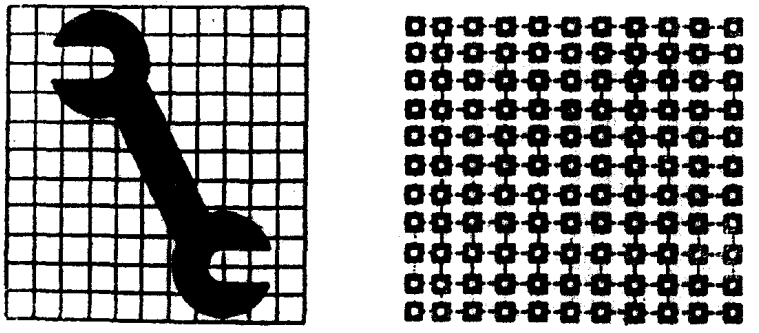

图 1.2 在机器视觉应用中，一个处理器/存储器胞对应处理图像上的一个点，由于该计算是二维的，因此把处理器连成二维网格。

像素处理器内就地运行，或者通过直接跟这些处理器二维连接网格中相邻的处理器通讯来运行（见图 1.2）。这样一类计算的一个典型的处理步要对每个点计算其紧邻点的平均值。二维网格能够同时对图像中所有的点计算上述平均值。例如，计算每点的四个紧邻点的平均值，就需要四个同时进行的处理步。在此期间，每个处理单元送一个值到上下左右四个紧邻的处理单元。在这些处理步中，每个处理单元还接收来自相反方向的值，并把它加到本身累积的平均值中。在这四步运算通常所需的时间里进行了 400 万次算术运算。

超大规模集成电路(VLSI)模似：每个晶体管一个处理器

图像处理的例子之所以行得通，是因为这个问题的结构和处理单元的通讯结构相一致。应用问题是二维的，硬件结构也是二维的。在别的应用中，问题的自然结构并不都那么有规律，而且在细节上还取决于正在进行计算的数据。在人工智能领域之外，一个此类应用的例子是有几十万个晶体管的集成电路的模拟问题。这个问题在检验超大规模集成电路(VLSI)中经常出现。显然，这项运算可以同时地进行，因为这些晶体管是同时工作的。十万个晶体管可以用十万个处理器来模拟。为此，处理器之间必须有像晶体管连线模式一样的连线（见图 1.3）。每个处理器模拟一个单独的晶体管，它直接跟那些模拟相连的晶体管的处理器通讯。当晶体管控制极上的电压变化时，模拟这个晶体管的处理器计算晶体管的响应，并把这个变化通知那些模拟相连的晶体管的处理器。如果许多晶体管控制极电压一起变化，那么，正如实际电路中那样要同时计算它们的响应。处理器的自然连线模式取决于所要模拟电路的实际连线模式。

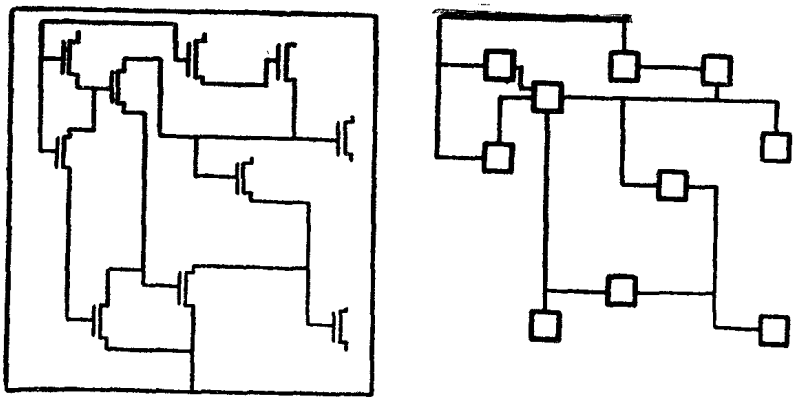

图 1.3　在 VLSI 模拟应用中，用单个的处理器 / 存储器胞模拟每个晶体管。这些处理器按电路模式连接起来。

语义网络：每个概念一个处理器

据我们所知，人脑在模拟晶体管方面并不是特别灵的，但它在解决需要处理非结构化数据的问题方面却似乎在行一些。这类处理工作也能够用以模拟数据结构的模式连结的处理器来做。例如，许多智能模拟程序以语义网络形式来表示数据。语义网络是一个标记图形，图中每个顶点表示一个概念，每条棱表示概念之间的关系。例如，“苹果”和“红色”用两个节点和连接这两个节点的“色彩”(COLOR) 连线加以表达（见图 1.4）。人们可能想从这样一个网络提取的许多知识并非是用这些连线直接显式表达的，而必须通过对各种连线进行搜索推导出来。例如，若我们知道“我的一个苹果”是一个“苹果”，我们就能够从“我

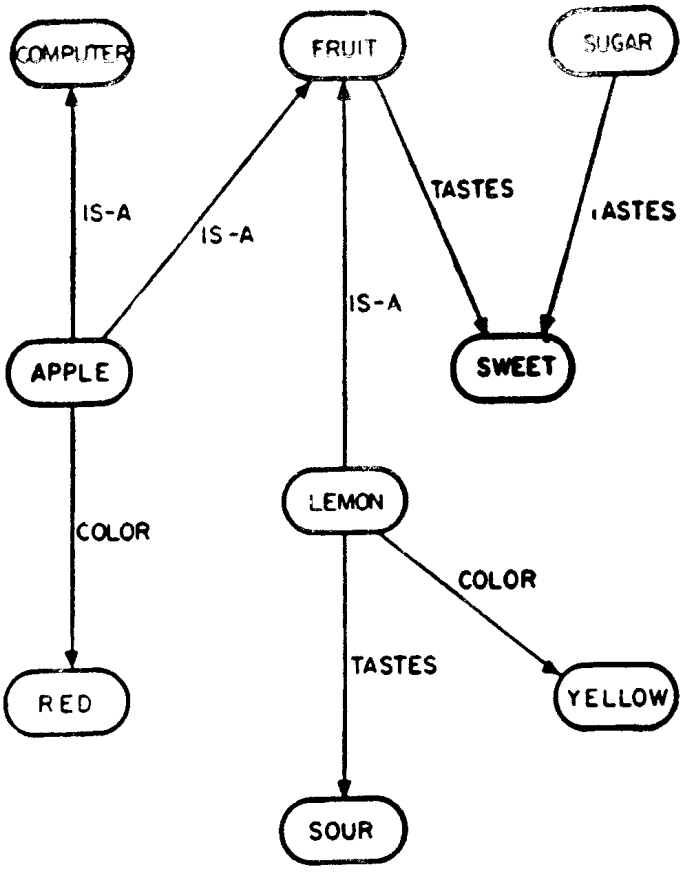

图 1.4　在语义网络中，用一个处理器 / 存储器胞能表示一个概念，单元之间的连线表示概念之间的关系。

的——苹果"和"苹果"之间的"是一个"(is a)连线以及"苹果"和"红色"之间的"色彩"(color of)连线的组合推导出"我的苹果是红色的"。

在实际应用的数据库里，有几十万个概念和上百万条关系连线，推导规则远比简单的"是一个"(is a) 之类的演译复杂。例如，有一些处理异常、矛盾、和不确定性的规则。该规则系统需要表示和处理有关部分和全体的关系，空间和时间的关系以及因果关系的信息。这类计算会变得非常复杂。由这类数据库解答一个简单的常识问题如"我抛出我的苹果，它将是否下落？"会占用一台串行计算机好多个小时。然而，人几乎一瞬间就能答出这个问题。因此我们可以有充分的理由相信，这项演译可以并行地进行。

这类从语义网络检索常识性知识的应用是 CM 计算机设计的主题之一。现有一些专门设计的基于语义网络的知识表达语言，如 NEIL(Fahlman, 1979)，它使得检索所必须的推演以并行方式加以运算。在这种系统中，每一个概念或判定都能用它自己的独立的处理单元来表示。为了进行推演，有关联的概念之间必须进行通讯，因此，必须把跟概念相对应的处理器连接起来。在这种情况下，硬件的拓扑结构取决于网络中储存的信息结构。例如，若苹果和红色有关联，那么，在代表苹果的处理器和代表红色的处理器之间必须有连接，以便能够把有关苹果的推演跟有关红色的推演联系起来。提供一个连接模式跟网络中储存的数据模式相一致的处理器群，就能够迅速地，且并行地执行检索操作。

有许许多多这类例子。只要硬件以与问题的特有结构相一致的方式连接起来，每个问

题在计算时就能够达到最大限度的并行。在那些设置了大量的处理存储胞，并且这些胞能够针对具体问题的固有结构重新配置连接模式的机器上，必能迅速地求解每一个相应的问题。

1.4 从一个算法看并行机在结构上的要求

我们将详细地研究一种特殊的并行算法，并用该算法来说明并行机在结构上的要求。我们以在一个大的图上的两个顶点之间找出一条最短路径作例子。这种算法很适用，因为除了简单有效，它性质上跟许多人工智能中的活化扩散 (spreading activation) 计算是相似的。

求解的问题是：

给定一个具有顶点 V 和连线 $E \subset V \times V$ 的图，以及任意一对顶点 $a, b \in V$，找出连接顶点 $a_1, V_1, V_2, \cdots, b$ 的最短序列的长度 k，使所有的连线 $(a_1, V_1), (V_1, V_2), \cdots, (V_{k-1}, b) \in E$ 都在图中。

举一个具体的例子。假定一个图有 10^4 个顶点，并且平均每个顶点任意连接 10^2 条连线。在这样一种图中，通过不多于三条连线的路径，就可以把几乎任何任意挑选出来的顶点对连起来了。

从顶点 A 到顶点 B 的最短路径的算法，首先用各顶点与顶点 A 的距离标注每个顶点。做法是：先标注顶点 A 为 0；接着用 1 标注所有连到 A 的顶点，再用 2 标出那些未作标注且跟标注为 1 的顶点相连的顶点。以此类推(见图 1.5)，一直到把顶点 B 标出为止。顶点 B 的标号值就等于最短连接路径长度。任何一条从 B 出发的具有单调递减标号的路径将以这个步数引到顶点 A。这种算法的通常的优化算法是同时从 A 和 B 传播标号，直到它们会合为止。但是，为了明了起见，我们还是用它最简单的形式。

理论上，我们能够像这样来叙述这个算法：

算法 I："找出从 A 到 B 的最短路径的长度"

1. 标出所有顶点为 $+\infty$；
2. 用 0 标记顶点 A；
3. 除顶点 A 之外，各顶点用与之相邻的顶点中最小的那个标号加 1 来加以标注。重复这个步骤，直到顶点 B 的标号标出为止；
4. 结束。B 的标号值就是最短路径长度。

我们用这个路径长度算法的例子来引出 CM 计算机的结构。

这类算法在传统的计算机上运行是很慢的。假设上述每一步要费一个单位时间。算法 I 所用的时间将与连接路径的长度成正比。对于上述的 10^4 个顶点的随机图，第三步将要重复二到三次。这样，确定一条路径的长度大约要 6 步。不幸的是，上面给出的那些算法步和在冯·诺埃曼机器上执行的各种操作步并不相当。把这个算法直接译成 Lisp 所得的程序效率是很低的。这个程序的运行时间跟顶点乘路径长度乘各顶点级数的平均值成正比。例如，上面提到的图将需要几百万个内部循环操作。用 VAX-11/750 计算机在一个试验图上