并发程序设计基础教程

Bingfa Chengxu Sheji
Jichu Jiaocheng

主　编　赵煜辉
副主编　周　兵

北京理工大学出版社
BEIJING INSTITUTE OF TECHNOLOGY PRESS

并发程序设计基础教程

主　编　赵煜辉
副主编　周　兵
编　委　安玉艳　杨欣欣

北京理工大学出版社
BEIJING INSTITUTE OF TECHNOLOGY PRESS

内 容 简 介

本书共3大部分，分为10章，系统介绍了与并发程序设计相关的基础知识，包括硬件组成结构、软件开发工具以及设计并发程序的基本思想。本书提供了很多具体应用的例子，以便于读者学习和理解并发程序设计的方法学。

本书主要面向高年级的本科学生，同时也可以作为研究生的入门教程。

图书在版编目（CIP）数据

并发程序设计基础教程 / 赵煜辉主编. —北京：北京理工大学出版社，2008.12（2009.1 重印）

ISBN 978-7-5640-2010-1

Ⅰ. 并…　Ⅱ. 赵…　Ⅲ. 交行程序-程序设计-教材　Ⅳ. TP311.11

中国版本图书馆 CIP 数据核字（2009）第 003819 号

出版发行 / 北京理工大学出版社
社　　址 / 北京市海淀区中关村南大街 5 号
邮　　编 / 100081
电　　话 / （010）68914775（办公室）　68944990（批销中心）　68911084（读者服务部）
网　　址 / http：//www.bitpress.com.cn
经　　销 / 全国各地新华书店
印　　刷 / 北京国马印刷厂
开　　本 / 787 毫米×1092 毫米　1/16
印　　张 / 12
字　　数 / 291 千字
版　　次 / 2008 年 12 月第 1 版　　2009 年 1 月第 2 次印刷
印　　数 / 1051～4000 册　　责任校对 / 申玉琴
定　　价 / 32.00 元　　责任印制 / 边心超

前　言

本书系统介绍了并发程序设计的基础知识，并提供了很多具体应用的例子，以便于读者学习和理解并发程序设计的方法。

本书主要面向高年级的本科学生，同时也可以作为研究生的入门教程。通过学习本书，可以使他们学习到并发程序设计的基本知识，同时也可以了解一些具体的开发工具。

本书共分为 3 大部分，第一部分为基础知识，包括第 1～5 章，主要介绍与并发程序设计相关的一些基础知识。第 1 章介绍并行计算机的硬件基础知识；第 2 章介绍并行计算模型的基础知识；第 3 章介绍如何评测和调试并发程序的性能；第 4 章介绍如何在共享存储器系统上实现并发程序；第 5 章介绍如何在消息传递系统上实现并发程序。

第二部分为并发程序设计基础，包括第 6～9 章，主要介绍设计并发程序的基本思想和方法。设计并发程序和实现并发程序是有区别的。设计并发程序可以不考虑具体的软硬件环境，但实现一个并发程序，则要根据不同的软硬件环境，采用不同的编程语言和工具。第 6 章介绍划分和分治的思想，只有数据或功能的划分，才可能实现并发程序；第 7 章介绍负载平衡，即讲述如何分配任务；第 8 章介绍流水线技术，这是一种从功能划分的角度设计并发程序的方法；第 9 章介绍同步计算的概念，同步的作用是使程序的各个并发部分协同工作。

第三部分为具体算法与应用实现。主要讲述并发程序的具体算法和具体应用。考虑到学生的接受能力，这部分目前只有一章，即第 10 章，介绍并行排序算法。串行的排序算法对计算机科学与技术专业的本科生应该不陌生，这对他们学习并行排序算法很有帮助。在本书的修改过程中，作者会逐步补充一些具体的应用，比如数值计算、数字图像处理。

由于水平有限，书中难免会有不妥之处，欢迎各界专家学者和广大读者批评指正，我们的联系方式是：zhoub1631@163.com。

编　者

目　录

第一部分　基础知识

第二部分　并发程序设计基础

第三部分 具体算法与应用实现

第一部分　基础知识

第1章　并行计算机的硬件基础

自计算机问世以来，就被人们用来解决计算量非常大的任务。而这种需求总是不断增长，如图 1–1 所示就是对这种需求的一个很好的说明。虽然随着微电子技术的发展，计算机使用的处理器的运行速度越来越快，但是单靠一个处理器来解决这些计算密集型的问题，很显然是不可能的。因此，必须在一台计算机中使用多个处理器来共同完成一项计算任务。这种包含多个处理器的计算机系统称为并行计算机，为并行计算机编写程序就是并发程序设计。

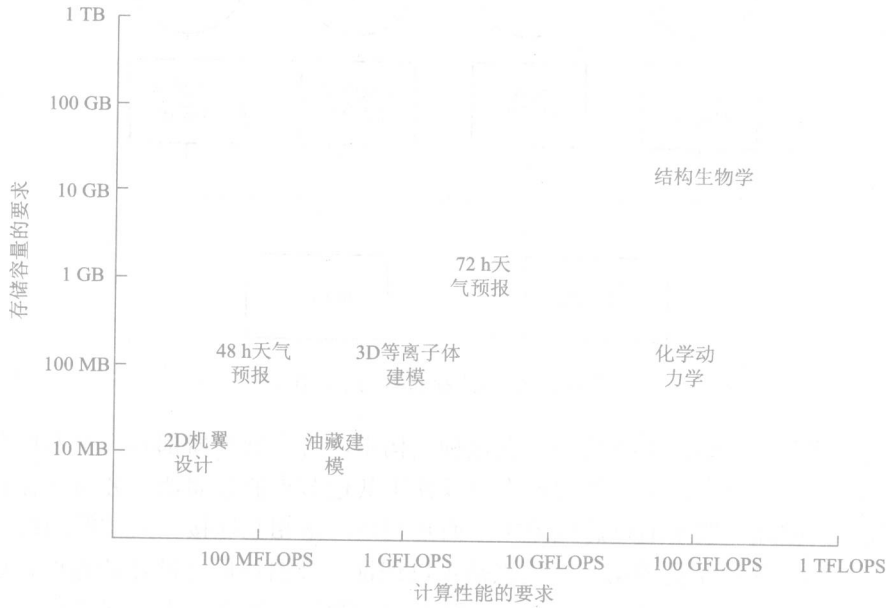

图 1–1　对计算量的需求

本书以讲述如何进行并发程序设计为主，但为了便于读者理解并发程序设计的思想，有必要先了解一下并行计算机的硬件基础。

1.1　并行计算机的组成

1.1.1　基本硬件组成和分类

并行计算机包含多个处理器模块（现代处理器往往自带高速缓存），这些处理器还需要一

定量的内存才能正常工作。此外，为了让这些处理器能够协同工作，还需要网络连接来传递数据。因此，一般情况下，一个并行计算机系统的最基本的组成包括 3 部分：多个处理器模块；一个或多个存储器模块；网络连接部件。

并行计算机系统经过多年发展，现在已经有很多系统。为了充分了解这些系统的特点，先来看看它们是如何分类的。

1. 根据存储器与处理器的连接方式分类

根据存储器与处理器的连接方式可以分为集中式存储器结构和分布式存储器结构。

集中式存储器结构如图 1–2 所示。在集中式存储器结构系统中，所有的处理器共同使用单一的一个存储器模块，而且一般通过总线连接，这种方式适合小型的多处理器系统（Multiprocessor），如 2 个或 4 个处理器，一般最多不超过 8 个。如果使用多总线（Multiple Buses），那么可以扩展到 16 个左右的处理器。但是如果处理器进一步增加，达到成百上千个，那么由于物理位置的差异以及访问冲突的增加，会导致一些处理器无法有效地访问内存。另外由于使用总线，一般无法扩展处理器个数。

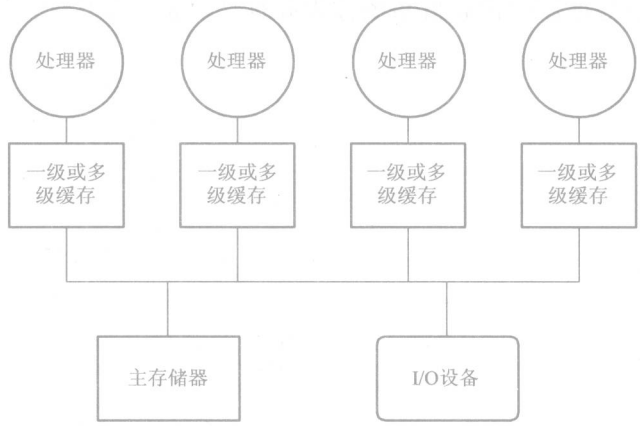

图 1–2 集中式存储器的基本结构

分布式存储器结构如图 1–3 所示。在这种结构中，每个处理器都和一个本地存储器模块相连，两者组成一个节点。每个处理器也可以使用其他节点的存储器。分布式存储器的节点一般都比较多，因此一般不通过总线连接，而是利用网络相互连接（具体的网络连接结构见 1.1.2 和 1.1.3 节）。这种结构的第一个优点是可以保证处理器访问存储器的速度，这是因为每个处理器都有一个本地存储器，可以存放计算需要的数据，不会出现访问冲突；第二个优点是可扩展性比较好，可以在网络中增加计算节点。这种结构的缺点是数据通信比较困难，需要专门的通信方法，才能实现处理器对其他节点的存储器的访问。

2. 根据数据的通信方式分类

根据数据的通信方式分类，一般可以划分为共享地址空间（Shared Address Space）系统和消息传递（Message Passing）系统。

在共享地址空间系统中，数据通信依靠处理器对相同地址的存储器单元的访问实现。通信过程及其控制一般都由硬件自动完成。在共享地址空间系统中，所有的存储器都可以被处理器共享使用，即使存储器在物理上是分布式的。为了实现共享，存储器的地址是统一编址

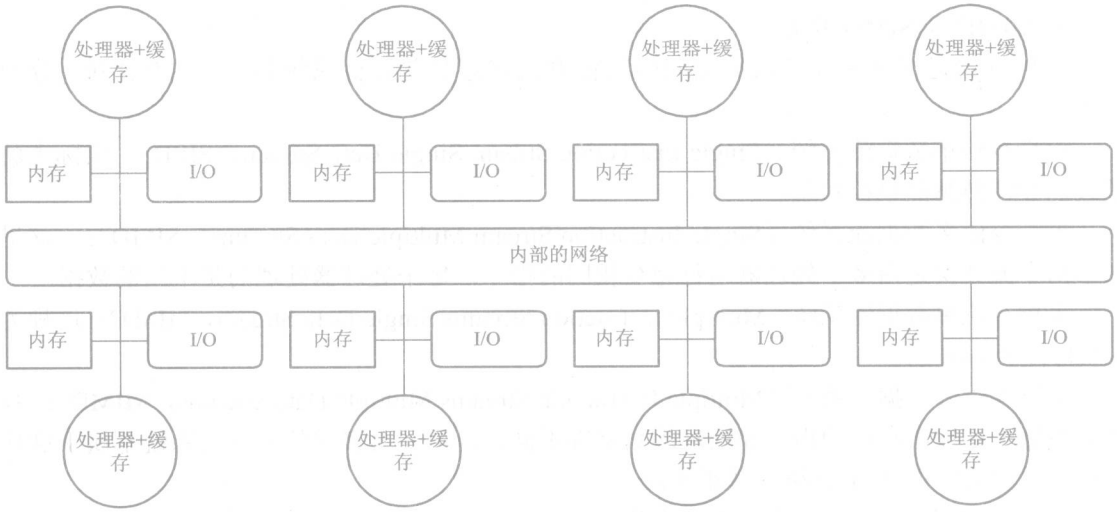

图 1–3　分布式存储器的基本结构

的，因此又称为单地址空间（Single Address Space）系统。由于地址空间是统一编址的，所以单个的处理器节点不能单独工作，因此可扩展性比较差；而且一个节点的故障往往导致整个系统的崩溃，因此可靠性也比较差。这种系统又称为共享存储器多处理器（Shared Memory Multiprocessors）系统。

共享地址空间系统可以根据存储器与处理器的连接方式进一步分类。如果存储器的结构是集中式的，所有处理器可以以相同的速度访问内存，那么就称为均匀存储访问系统（Uniform Memory Access，UMA）或对称共享存储器多处理器系统（Symmetric Shared-memory Multiprocessors，SMP）。如果存储器的结构是分布式的，那么处理器访问存储器的速度就取决于存储器的位置，访问本地存储器的速度最快，也就是说，处理器访问内存的速度是不一样的，所以叫做非均匀存储系统（Nonuniform Memory Access，NUMA）或分布式共享存储器系统（Distributed Shared-Memory，DSM）。

在消息传递系统中，所有的节点就是单独的一个计算机系统，可以是单 CPU 的个人计算机或工作站，甚至还可以是多处理器的 SMP。每个节点的存储器独自编址，也就是说，同一个地址与多个存储器单元对应。因此，数据通信就无法依靠访问存储器单元来实现，必须通过节点之间相互发送包含有数据的消息实现。在消息传递系统中，单个节点的故障不会造成整个系统的崩溃，而且可以采用容错技术，来减小单节点故障的影响，因此可靠性比较高。而且单个节点也比较容易加入到系统中，可扩展性也比较强。消息传递系统往往又称为消息传递多计算机系统。

消息传递系统也可以分为两大类：大规模并行处理机系统（MPP）和集群系统（Cluster）。两类系统的界限并不明确，如果要严格区分，可以认为集群系统是构建在局域网环境下的并行系统，比如以太网，采用星型连接，优点是可扩展性非常高，但网络传输速度比较慢；而 MPP 则利用结构更复杂的专用网络技术，正如 1.1.2 和 1.1.3 节将要讲述的网络技术。MPP 的网络传输速度较高，但可扩展性稍差，因为它必须满足一定的网络拓扑结构要求。

3. MIMD 和 SIMD 分类

这种方式是根据指令和数据之间的工作方式来划分并行机系统的。一共有 4 种，分别如下。

单指令流单数据流系统（Single Instruction Stream Single Data Stream，SISD）：实际上就是人们非常熟悉的单机系统。

单指令流多数据流系统（Single Instruction Stream Multiple Data Streams，SIMD）：这里的单指令流其实是指多个处理器运行完全相同的指令，每个处理器处理的是不同的数据。

多指令流单数据流系统（Multiple Instruction Streams Single Data Stream，MISD）：这种类型实际是不存在的。

多指令流多数据流系统（Multiple Instruction Streams Multiple Data Streams，MIMD）：系统中的每个处理器执行的指令和处理的数据都不同。目前，大多数的并行机系统都属于这种类型，比如前面提到的 MPP 和集群系统。

MIMD 类型的并行系统有下面两个优点。

（1）MIMD 系统在使用上有很大的灵活性。它既可以作为一个单用户系统，为用户提供高性能的服务；也可以作为多用户系统，同时执行多个不同的应用程序，为多个用户提供服务。

（2）MIMD 系统可以获得较高的性价比。因为构建一个 MIMD 系统可以使用通用的商用部件，比如集群系统完全可以用普通的局域网来构建。而 SIMD 系统一般要使用专用的 CPU，因而价格昂贵。

1.1.2 静态网络连接结构

对于大型的并行机系统，要使用多个处理器/存储器模块，这些模块之间通过一定的网络结构相互联系。网络结构一般可以分为两种：静态网络连接结构和动态网络连接结构。本节介绍静态网络连接结构，1.1.3 节介绍动态网络连接结构。

静态互联网络是指网络连接方式是固定的，不能改动。常用的静态网络连接结构有以下几种：完全连接；线状/环状连接；网格/花环连接；超立方体连接；树状连接。此外，还可以采用嵌入（Embedding）技术，将一种网络连接结构映射为另一种逻辑网络连接结构。

1. 完全连接

完全连接结构是指两两节点之间都有一条物理链路。因此，需要 $n(n-1)/2$ 条链路。这种结构的优点是消息延迟最小，因为任何两个节点之间的通信，只需要经过一条链路，而不必通过中间节点传送。但当节点数比较多时，这种方法代价很大，不经济，而且也很难布线。所以实际的并行机都采用连接个数有限的连接方式。

2. 线状/环状连接

如图 1–4 所示，最简单的连接方式是将节点依次顺序连接，从 0 到 N。这种结构无法提供容错性能，任何一条链路失效，都将导致数据无法从失效点的一端传到另一端。如果把线状连接的两端也连接起来，就形成了环状连接。由于从一个节点到另一个节点有两条路径，所以有一定的容错能力。一条链路失效，数据仍可以传送。

图 1–4 线状/环状连接

3. 网格/花环连接

图 1–5 是一个二维网格/花环结构示意图。一个节点同相邻的 4 个节点相连。除了最简单的二维网格/花环，实际上还有三维或更高维的网格/花环。在三维网格中，一个节点同相邻的 6 个节点相连，即每一维方向上各有两个邻节点。同理，d 维有 $2d$ 个相邻的节点。对许多科学和技术问题，因为数据是按照二维或三维数组组织的，所以网格/花环结构在解决这些问题时，非常合适。

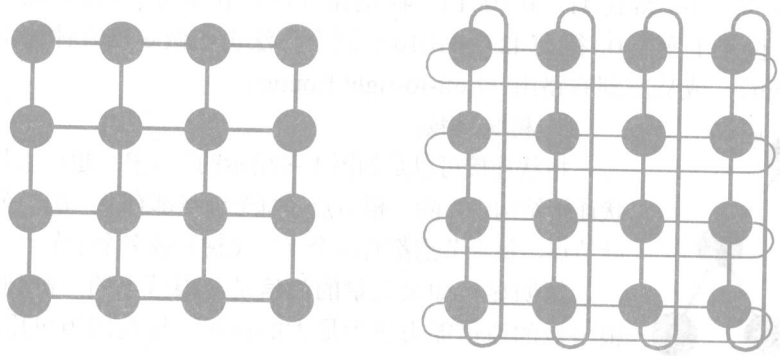

图 1–5　二维网格/花环结构

4. 超立方体连接

一个 d 维二元超立方体结构是指在每一维上每个节点同另一个节点相连。如图 1–6 所示为三维超立方体结构。在 d 维超立方体中，每个节点有一个 d 位的二进制地址。每一位对应一维，取值 0 或 1。如三维超立方体地址有 3 位，000 和 001，010，100 相连。相连接的节点只有一位取值不同。同样的规则对高维的超立方体同样适用。

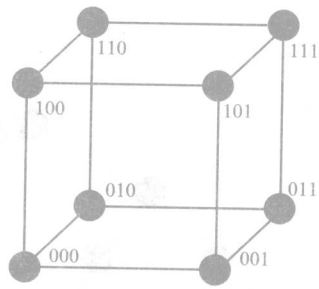

图 1–6　三维超立方体结构

一个 d 维超立方体可以由两个 d–1 维超立方体组成。图 1–7 显示了一个 4 维超立方体由两个三维超立方体组成。两个三维超立方体中间有 8 条链路，这 8 条链路就是第 4 维链路。

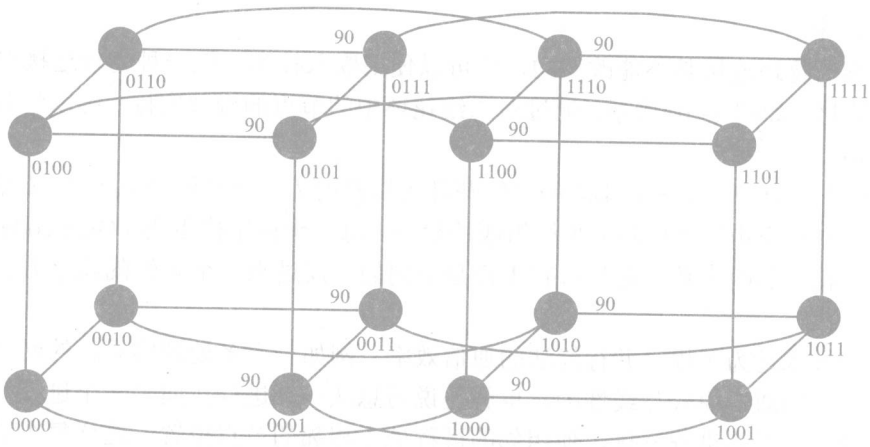

图 1–7　一个 4 维超立方体由两个三维超立方体组成

超立方体有一个可以得到最小距离的不会死锁的路由算法。在这个算法中，通过对地址的异或（Exclusive-OR）操作来发现路径。比如，在一个 6 维超立方体中，要寻找从 13 号节点到 42 号节点的路径，先对 13 号节点和 42 号节点的二进制地址进行异或操作，结果是 100111，有 4 个 1，所以要经过 3 个中间节点。从结果的最高位开始，如果为 1，那么当前节点的地址的对应位就要改变。因此，源节点 13(001101)的最高位从 0 变为 1，得到 45(101101)。45 的第 4 位从 1 变为 0，得到 41（101001）。41 的第 5 位从 0 变为 1，得到 43（101011）。最后，41 的最后一位从 1 变为 0，得到 42(101010)。这个算法也叫做 e-立方体路由算法(e-Cube Routing Algorithm)，或从左到右路由（Left-to-right Routing）。

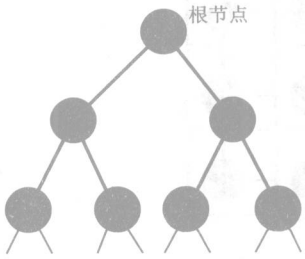

图 1–8　二叉树

5. 树状连接

树状连接可以是如图 1–8 所示的二叉树，也可以是多叉树。在树状连接结构中，除了根节点，每个节点都有唯一的一个父节点。除了叶节点，每个节点都有两个（二叉树）或多个子节点（多叉树）。

很明显，如果大量的通信是从根节点的一侧到另一侧，那么根节点的通信能力成为最大的瓶颈，这是因为可用的链路数比较少。因此 Leiserson 在 1985 年提出了胖树（Fat Tree）结构，就是大量增加根节点的链路。如图 1–9 所示，根节点的链路数随着高度呈指数增长。

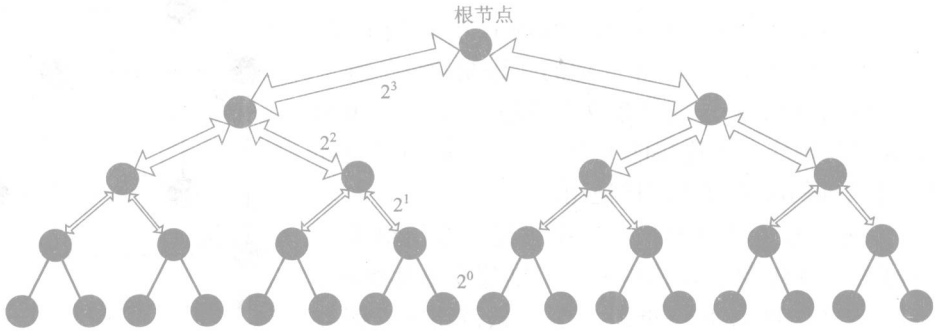

图 1–9　胖树

6. 嵌入技术

静态网络的物理连接是不能改变的，但可以使用嵌入技术，把一种物理连接映射为另一种逻辑上的连接。如图 1–10 所示，通过对二维花环中的节点的合理安排，可以使节点映射为一种环状结构。

另外，如图 1–11 所示，一个 4×4 的二维花环可以映射为一个 4 维超立方体(对照图 1–7)。其中每个节点有一个由 x 坐标和 y 坐标组成的地址。每个坐标用格雷码（Gray code）来标识。在格雷码中，相连的两个数字有且只有 1 位是不同的。反过来一个 4 维的超立方体也可以映射为 4×4 花环。

使用嵌入技术是为了提高并行算法的通信效率。例如一个多处理器系统的网络结构是网格结构，但算法的通信模式是线性的，也就是说消息从一个进程传向另一个进程，再传向下一个。那么通过合理的进程安排，使相邻的进程处于相邻的节点位置，这样每个消息只需经过一个链路就可到达目的地。这个合理的安排结果就形成如图 1–10 所示的环状嵌入结构。

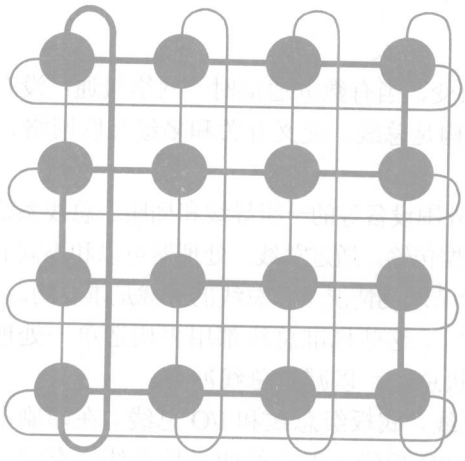

图 1–10　环状连接嵌入到二维花环

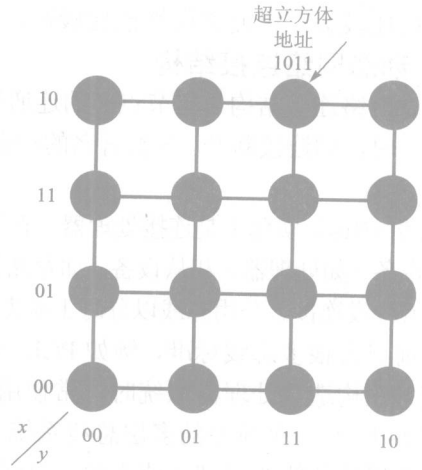

图 1–11　一个 4×4 的二维花环映射为一个 4 维超立方体

7. 网络性能的衡量

对于各种静态网络结构，可以用下面几个术语来衡量其传输特性。

带宽（Bandwidth）：是指在单位时间内可以传输的位数，单位是位/秒（bit/s）。

网络延迟（Network Latency）：是指消息经过网络传输，从源点到目的地所需时间。

通信延迟（Communication Latency）：是指发送消息所需的总时间，一般指从开始准备发送消息，到把消息发送出去的这段时间。这包括软件开销和接口延迟，但并不考虑消息何时到达目的地。

消息延迟（Message Latency）或启动时间（Startup Time）：是指发送 0 长度的消息所需的时间。它包括查找路由、消息打包解包以及通过物理传输介质所需的时间。它反映的是系统的通信性能，是最基本的通信代价。

以上 3 种延迟的定义，有时可以不加区别地使用。

直径（Diameter）：是网络内距离最远的节点间的最小链路数。

对分宽度（Bisection Width）：是指当网络被分成两个相等的部分时，所必须割断的链路数。

两个节点间的链路数是造成消息延迟的重要因素，因此直径可以用来判断最糟糕的情况下的消息延迟。直径是单一消息所经过的最大的距离，它可以用来发现并行算法的通信上限，即最大的通信代价。比如一个排序的例子，如果要按顺序把数字保存到节点上，就需要在节点间移动数字。最差的情况是，数字要在距离最远的两个节点间移动，那么移动一个数字需要的最大的通信次数就等于直径。

对分宽度可以用来发现消息延迟的下限，即最小的通信代价。例如有时需要将 n 个数据从网络的一半移动到另一半，如果对分宽度是 w，那么同时最多有 w 条链路可以使用，因此至少要 n/w 次通信才能完成操作。

根据以上定义，可以得到下面一些结果。

N 个节点的线状网络的直径是 N–1，对分宽度是 1。环状结构的直径是 $N/2$，对分宽度是 2。一个 $n \times n$ 网格的直径是 2（n–1），对分宽度是 n。N 个节点的超立方体的直径是 $\log_2 N$，对分宽度是 $N/2$。可见，超立方体的直径等于维数，当节点数很大时，直径并不是很大，但

对分宽度比较大，因而通信性能比较高。

1.1.3 动态网络连接结构

动态网络连接结构是指节点间的连通性不断发生改变，当有数据通信时，网络接通；没有数据通信时，则连接断开。3 类著名的动态网络连接结构是总线、交叉开关和多级互联网络。

1. 总线

总线（Bus）实际上是连接处理器、存储模块和 I/O 外围设备等的一组导线和插座。总线系统用于主设备（如处理器）和从设备（如存储器）之间的数据传输。通过总线，处理器可以和不同的存储器或外设连接。公用总线以分时工作为基础，在多个请求的情况下，总线的仲裁是重要的。

目前已有很多总线标准，例如 PCI，VGP，PCI-E 等。这些标准总线都用来构造单一处理器系统。在构造多处理器系统时，常使用多总线和层状总线，以减少总线冲突。

图 1–12 是一个典型的多层总线系统，包括板级总线、底板级总线和 I/O 总线。在印刷电路板上实现的总线叫局部（或本地）总线。习惯上，CPU 板级上的总线叫本地总线，存储器板级上的总线叫存储器总线，I/O 板级和通信板级上的总线叫数据总线。系统总线是在底板上实现的，它为所有插入板之间的通信提供了通路。在各插入板中均设有专用逻辑接口（IF）和专用控制器，包括 I/O 控制器（IOC）、存储器控制器（MC）和通信控制器（CC）。I/O 设备是通过 I/O 总线（如 SCSI）与计算机系统连接的。

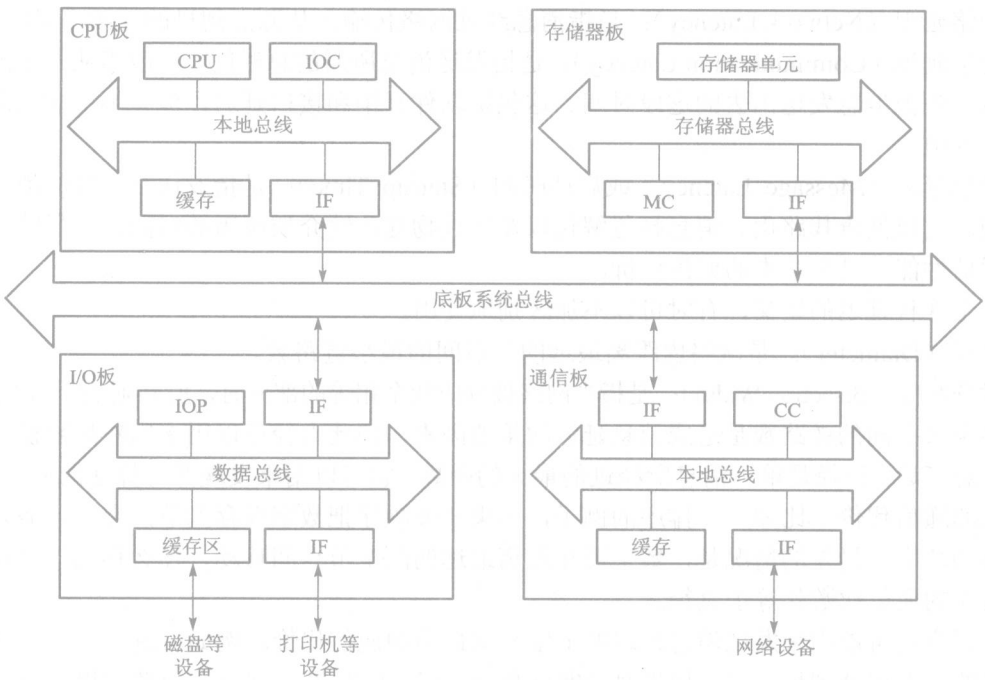

图 1–12 一个典型的多层总线系统

2. 交叉开关

交叉开关 （Crossbar Switcher）是一种高带宽网络，像电话交换机一样，交叉点开关能在源/目的对之间建立动态连接。每个交叉点开关为一个源/目的对提供一条专用连接通路，其开关状态可根据程序的要求动态地置为“开”或“关”。图 1–13 是一个 4×4 的交叉开关，其

中每个输入口有一个接收器和输入缓冲器，以处理接收到的请求；每个输出口有一个发送器，以传递输出数据信号至一条通信链上；“开关控制”决定开关的“开/关”状态。

在并行处理中，交叉开关的使用通常有两种方式：一种是用于处理器之间的通信，此时每一行和每一列只能接通一个交叉点开关，所以一个 $n\times n$ 的交叉开关网络一次最多可接通 n 个对；另一种是用于处理器和存储模块之间的通信，此时每个存储模块一次只能有一个处理器请求访问，所以每一列中只能接通一个交叉点开关，但为了支持并行（或交叉）存储访问，每一行可同时接通多个交叉点开关。一般而言，交叉开关的成本为 n^2，其中 n 为端口数，这样就限制了它在大型系统中的应用。

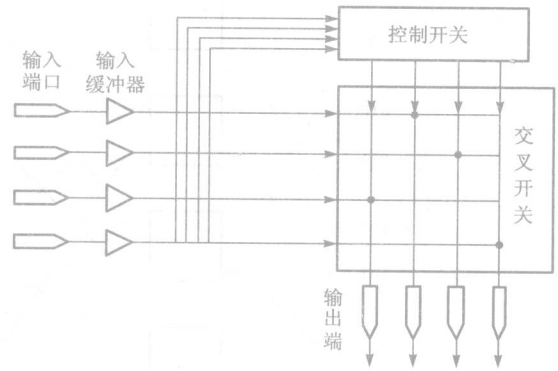

图 1–13　一个 4×4 的交叉开关

3. 多级互联网络

为了构筑大型开关网络，通常可将单级交叉开关级联起来形成一个多级互联网络（Multistage Interconnection Network，MIN），它已被用在 MIMD 和 SIMD 并行机中。一种通用多级互联网络如图 1–14 所示，其中每一级都用了多个 $a\times b$ 开关单元，相邻各级之间有着固定的级联拓扑。为了在输入和输出之间建立所需的连接，可以动态设置开关状态。多级互联网络的主要优点是采用模块结构，可扩展性好，但延迟随网络尺寸呈现对数增长。

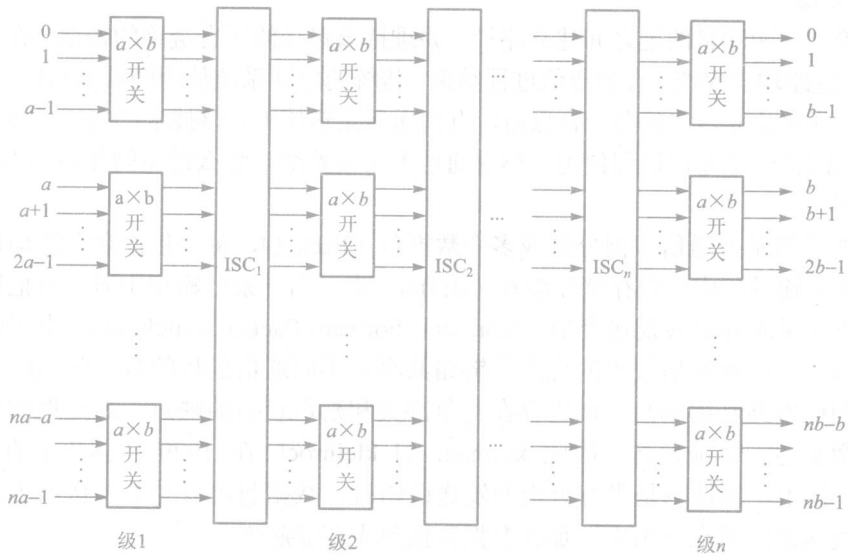

图 1–14　一种通用多级互联网络

图 1–15 显示了 Omega 网络的结构。它有 8 个输入和输出。其中每个小方格代表一个 2x2 的交叉开关。输入端和输出端用地址给定。目的地址的最高位控制第 1 级中的交叉开关。0 表示选择上输出，1 表示选择下输出。只要按照目的地址控制交叉开关，任何一个输入端都可以与任何一个输出端连通。

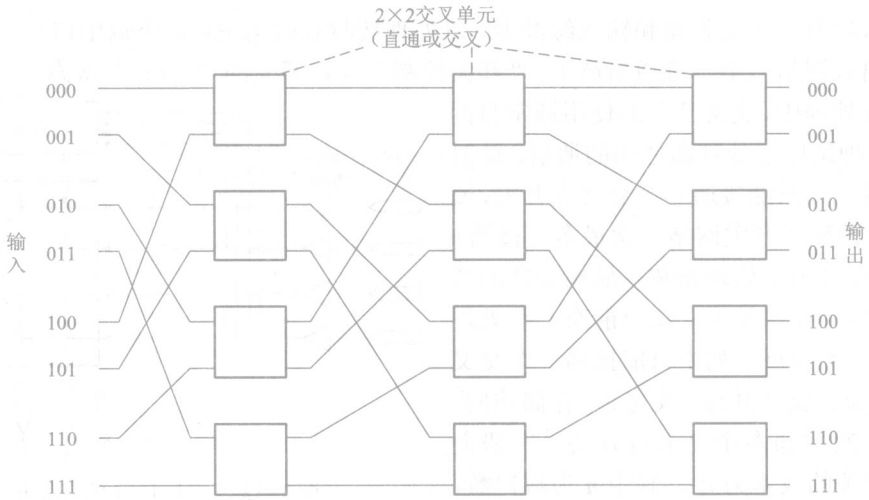

图 1–15 Omega 网络的结构

1.1.4 网络通信方法

前两节讲的是节点之间是如何连接的，本节介绍的是节点间是如何通信的。两个节点之间如果有一条物理链路，那么实现通信是很简单的。但大多数情况下，通信需要经过若干中间节点，所以首先是要确定通信过程如何在节点间中继。一般情况下有两种方式：电路交换（Circuit Switching）和包交换（Packet Switching）。

1. 电路交换

电路交换需在源和目的地之间建立路径，并维持该路径的所有链路的畅通。在通信的整个过程中，所有链路均被保留，直至通信过程结束。传统的电话系统使用的就是电路交换的方式。一旦一条电话线路建立，它就会一直保留，直到通话完毕挂机。该技术的缺点是路径中的链路不能共享，只能被一个通信过程使用。如果通信不是很繁忙，那么链路的使用率不高。

2. 包交换

包交换的思想是把通信数据分割成多个数据包（Packet），每个包都含有源地址和目的地址。数据包在传输过程中，先存储在缓存（Buffer）中，当一条链路空闲时，再把数据包发送出去。这种方式又叫存储转发包交换（Store-and-Forward Packet Switching）。电子邮件系统就是使用这种方式。这种通信技术的优点是链路共享，不同通信过程的数据包可以依次从一条链路发送，因而链路利用率高。但也存在竞争冲突和延迟较高的缺点。高延迟的原因之一是需要先存储数据包，才能发送。因此 kermani 和 kleinrock 在 1979 年提出了直通（Virtual Cut-through）技术。该技术是当有可用的发送链路时，数据包可以马上发送出去而不必先保存。但如果发送链路正在使用中，那就要把数据包先存储起来。

还有一种叫虫孔路由（Wormhole Routing）的技术被 Dally 和 Seitz 在 1987 年提出，用来改进存储转发路由技术。在存储转发路由技术中，所有的数据包作为一条消息，被整体保存在节点中，并作为被整体发送出去。但在虫孔路由中，数据会被分割成更小的单元，叫做片（Flit）。一个片只有一两个字节。进行通信时，先把只包含地址信息的消息头传送出去，用来建立路由。一旦消息头成功传送，表明有可用的链路。那么随后的片就会经过这条链路移向下一个节点，而且在所有片通过这条链路之前，该链路会一直保留，就像电路交换一样独占