普通高等学校教材

TEXTBOOK FOR HIGHER EDUCATION

分布式系统设计原理与应用

FENBUSHI XITONG SHEJI YUANLI YU YINGYONG

黄光球 陆秋琴 李 艳 编

西北工业大学出版社

高等学校教材

分布式系统设计原理与应用

黄光球　陆秋琴　李　艳　编

西北工业大学出版社

【内容简介】 本书较全面地介绍了分布式系统设计中的一些重要问题，包括基本概念、问题和一些可能的解决方案，主要内容有 DCDL 语言、分布式环境下的并行计算、Petri 网行为分析模型、互斥算法和选举算法、死锁处理、自适应路由、分布式系统可靠性设计、静态负载调度、数据管理等内容。

本书可作为研究生的分布式系统设计课程的教材，也可作为高年级本科生和研究生的高级操作系统课程的教材。

图书在版编目(CIP)数据

分布式系统设计原理与应用/黄光球，陆秋琴，李艳编. —西安：西北工业大学出版社，2008.8
ISBN 978-7-5612-2444-1

Ⅰ. 分… Ⅱ. ①黄…②陆…③李… Ⅲ. 分布式计算机系统 Ⅳ. TP338.8

中国版本图书馆 CIP 数据核字(2008)第 123848 号

出版发行：西北工业大学出版社
通信地址：西安市友谊西路 127 号 邮编：710072
电　　话：(029)88493844　88491757
网　　址：www.nwpup.com
印 刷 者：陕西兴平报社印刷厂
开　　本：787 mm×1 092 mm　1/16
印　　张：16.875
字　　数：409 千字
版　　次：2008 年 8 月第 1 版　　2008 年 8 月第 1 次印刷
定　　价：28.00 元

前　言

近年来，分布式处理系统获得了突飞猛进的发展，并呈现出前所未有的广阔应用前景，为此我们有必要学习和系统了解当前国内外分布式系统设计的最新理论与技术。分布式系统对用户看起来像是普通系统，然而它是运行在一系列自治处理单元上的系统，每个处理单元有各自的存储器空间并且消息的传播延迟不能忽略不计。这些处理单元间有紧密的合作。系统必须支持任意数量的进程，合乎处理单元的动态扩展。在现实条件下，分布式系统多种多样并涉及不同的系统体系结构，分布式系统设计原理则是在抽象层次上描述如何设计一个高效的分布式应用系统，因此建立在抽象层次之上的分布式系统设计原理所涉及的理论与算法是分布式系统设计的关键所在。

分布式系统设计原理提供了尽可能多的计算机处理能力和数据的透明访问，同时可实现高性能与高可靠性的目标。对一些用户来说，一个分布式系统是为了解决单个问题而紧密结合在一起的多处理机的集合。对另一些用户来说，一个分布式系统可能意味着一个由地理上分散的各自独立的处理机组成的计算机网络，这些处理机连接在一起以实现对不同资源的共享。

本书涵盖了设计一个分布式系统所涉及的所有内容，较全面地介绍了分布式系统设计中所用到的形式化方法、并行计算、行为分析、互斥和选举、死锁处理、特殊网络路由、系统可靠性、负载分配等方面的概念与算法，并详细地给出了分布式系统设计实例。

由于分布式系统设计的原理抽象难懂，本书的特点是对于复杂难懂的原理或定义，通过给出大量的例子进行解析来降低难度，使读者能充分理解原理或定义的含义；对于复杂的算法，通过给出详细的分析过程来增加透明度，使读者能充分掌握算法的设计技巧。第 1 章到第 12 章讲述原理，第 13 章介绍了一个分布式操作系统的实例。通过学习，使读者对分布式系统设计原理有一个完整的认识和了解，并能够了解与跟踪当前的最新技术。

本书得到了西安建筑科技大学新办专业重点教材项目的资助。

编　者

2008 年 6 月

目　录

第1章 绪 论

1.1 分布式系统的定义

1.1.1 计算机组织结构与分类

迄今为止，两个基本的计算机组织结构如下：

(1)物理共享式存储器结构。该结构有一个为所有 CPU 所共享的单一全局存储器地址空间，如图 1.1 所示。这样的系统称做紧耦合系统。在一个物理共享式存储器系统中，CPU 间的通信通过读和写共享存储器的操作进行。

(2)物理分布式存储器结构。该结构没有共享存储器，每个 CPU 均有自己的本地存储器，如图 1.2 所示，CPU 和本地存储器的组合称做处理单元(Processing Element，PE)或简称处理机。这样的系统称做松耦合系统。在一个物理分布式存储器系统中，处理机间的通信通过互连网络上的消息传递来进行，发送处理机用于发送命令，接收处理机用于接收命令。

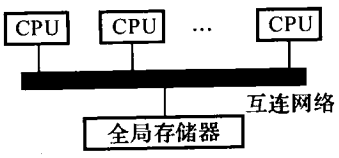

图 1.1 物理共享式存储器结构

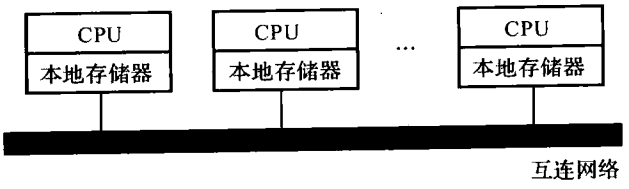

图 1.2 物理分布式存储器结构

在图 1.1 和图 1.2 中，互连网络均用于连接整个系统的 PE 并支持数据和指令的传送，然而，通信模型的选择无须和物理系统联系在一起。因此，可以区分为以硬件表现的物理共享和以编程模型表现的逻辑共享。图 1.3 表示了 4 种可能的共享组合。

	逻辑共享式	逻辑分布式
物理共享式	共享存储器	模拟消息传递
物理分布式	分布式共享存储器	消息传递

图 1.3 物理和逻辑共享式/分布式存储器的比较

(1)共享存储器。该结构在单处理机并行程序设计和多处理机共享存储器程序设计中采用，进程间必须同步以保证对共享数据访问的一致性。

(2)消息传递。该结构在分布式存储器系统中采用，其中通信是通过消息的传递进行的。

(3)模拟消息传递。该结构在一个物理共享存储器体系结构上实现逻辑分布式编程模型。

消息通过共享存储器的缓冲区传递。

(4)分布式共享存储器。该结构也叫做共享虚拟存储器或分布式存储器系统。这个模型通过使分布式存储器系统对程序员来说就像共享存储器系统一样,使得在分布式存储器环境下编程尽可能如在共享存储器系统那样容易。

计算机依据两种模型构建,即冯·诺依曼模型和非冯·诺依曼模型。

(1)冯·诺依曼模型。该模型的特点是:①使用单一的处理部件来完成计算、存储以及通信工作;②存储单元是定长的线性组织;③存储空间的单元是直接寻址的;④使用低级机器语言,指令通过操作码来完成简单的操作;⑤对计算进行集中的顺序控制。

(2)非冯·诺依曼模型。数据流模型是基于贪心求值(Greedy Evaluation)策略的,即一个函数(程序)只要它的输入数据一就绪就马上执行。有些模型是基于懒惰求值(Lazy Evaluation)策略的,即只有需要函数(程序)的结果时才执行求值。

目前大部分计算机(包括串行和并行计算机)是建立在冯·诺依曼模型上的,用 Flynn 分类法对冯·诺依曼计算机进行分类,其实质是基于指令流和数据流的多重性。该分类法如下:

(1)单指令单数据(Single Instruction Single Data,SISD)系统。图 1.4 为 SISD 模型。

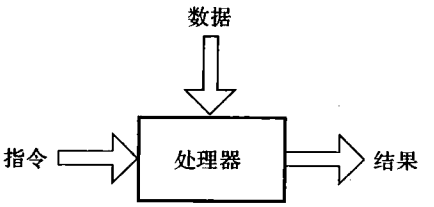

图 1.4 SISD 模型

(2)单指令多数据(Single Instruction Multiple Data,SIMD)系统。该系统的一条指令同时对不同的数据集进行并行操作。这里的数据集个数是同时进行操作的处理器的数量。图 1.5 为 SIMD 模型,图中描述了对多个数据集中的每个数据元素都加 1 这样一个简单操作。

FOR $i=1$ to n DO IN PARALELL

$x_i \leftarrow x_i+1$

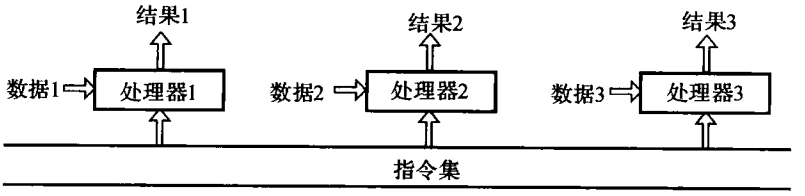

图 1.5 SIMD 模型

该类计算机可以直接进行并行运算。例如,对于具有 n 个处理器 $P_1, P_2, \cdots, P_n$ 的 SIMD 体系结构计算机来说,若对每个处理器都给予共同的"对数据加 1"的指令,则每个处理器 P_i 会得到 x_i,并对它加 1。可以看出处理器的数据不同而指令相同。在 SIMD 模型中,有两种类型的体系结构:共享存储器模型和直接连接网络。

(3)多指令单数据(Multiple Instruction Single Data,MISD)系统。该系统是对单个数据集执行多个不同操作的理论模型。目前为止,还没有设计出符合这种模型的计算机。图 1.6

展示了该模型的结构。

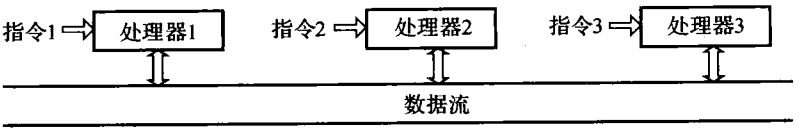

图 1.6 MISD 模型

(4)多指令多数据(Multiple Instruction Multiple Data,MIMD)系统。该系统是指多处理器系统,它有多个处理器,可以独立运行并生成全局系统的结果。每个处理器都可以执行单独的指令对单独的数据集进行操作。

1.1.2 分布式系统的特点及定义

分布式系统是一个相对较新的领域,到目前为止还没有一致的定义。与顺序计算相比,并行的、并发的和分布式的计算包括多个 PE 间的集体协同动作。这些术语在范围上相互覆盖,有时也交换使用,其定义和不同含义如下:

(1)并行。表示某些动作可以按同一次序同时执行。单指令流多数据流(SIMD)计算机就是一个使用多个数据处理单元在许多数据项上同时进行相同或相似操作的例子。

(2)并发。表示某些动作可以以任意次序执行。例如,在多指令流多数据流(MIMD)并行计算机上进行部分独立的操作。

(3)分布式。若一个系统的部件在不同地方,部件之间要么不存在或仅存在有限的合作,要么存在紧密的合作,则该系统是分散式系统。当一个分散式系统不存在或仅存在有限的合作时,它就被称做网络系统;否则它就被称为分布式系统,其表示在不同地方的部件之间存在紧密的合作。如果一个系统具有多 PE、硬件互连、PE 故障无关、共享状态等特征,它就是一个分布式系统。以下是分布式系统要求的属性:

1)进程数目任意,每个进程也被称做一个逻辑资源。

2)PE 数目任意,每个 PE 也被称做一个物理资源。

3)基于消息传递的通信,这提供了合作式消息传递方式。

4)进程间合作,或者说多个进程用于协同解决同一个任务而不是几个独立的任务。

5)通信延迟,两个 PE 间的通信延迟不可忽略。

6)故障独立,没有任何单个逻辑或物理的资源故障会导致整个系统的瘫痪。

7)故障化解,即系统必须提供在资源故障的情况下重新配置系统拓扑和资源分配的手段。

基于以上定义我们可以看出:计算机网络(局域网、城域网和广域网)不是分布式系统,这是因为不同站点的进程没有协同工作;一个物理共享存储器多处理机不是分布式系统,因为它没有故障独立性。分布式共享存储器是一个逻辑共享存储器系统,但是它具有资源故障独立性并支持故障化解的特点,因此该系统可以被当做特殊的分布式系统。

在一个分布式系统中,一系列为了解决同一个问题而合作的进程运行在不同的 PE 上,用户可能知道也可能不知道这些进程的位置;在工作站模型(C/S 模型)中,通过区分本地进程和远程进程,用户通常知道进程的位置,系统通过支持进程迁移可在不同的 PE 之间共享 CPU 周期;在处理机池模型中,用户不知道进程的位置。处理机池中的 PE 没有直接隶属于它们的

终端，分布式共享存储器模型就是一种特殊的处理机池模型。

1.2 互连网络与特殊网络

计算机网络是分布式系统的一部分，互连网络是并行/分布式系统中处理机连接在一起的一整套通信链路、总线或交换机。互连网络的分类如图 1.7 所示。

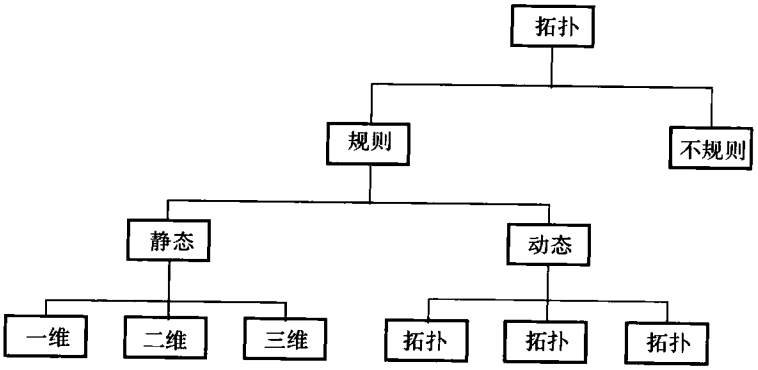

图 1.7 互连网络的分类

一个互连网络的拓扑可以是动态的或静态的。静态网络是由固定的点到点直接连接组成的网络；动态网络利用交换机实现动态配置来满足用户的通信要求。静态网络的拓扑可以根据网络布局所要求的维数进一步划分，如 1 维、2 维和 n 维等。总的来说，一个静态互连网络可以用图 $G=(V,E)$ 表示，V 和 E 分别表示顶点集和边集。每个顶点代表一个处理机（或节点），每条边代表一条通信链路。以下是几种典型的静态网络：

(1)全连接型网络：每个处理机和系统中的所有其他处理机直接相连，如图 1.8 所示。对于一个具有 n 个处理机的全连接型网络共有 $n(n-1)/2$ 条通信链路。

(2)直线型网络和环形网络：直线型网络中，处理机排列在一条直线上，相邻节点相连，内部处理机有两个连接而边界处理机只有一个，如图 1.9 所示。如果两个边界节点相连，就形成了环型网络，如图 1.10 所示。一个节点数为 n 的环型网络，其网络直径为 $n/2$（网络中两个节点间的最大距离称为网络的直径），每个节点的度数为 2。

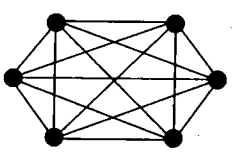

图 1.8 全连接型网络

图 1.9 直线型网络

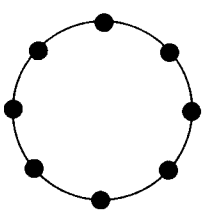

图 1.10 环形网络

(3)网格。一个 k 元 n 维网格，有 k^n 个节点，其内部节点度数为 $2n$，网络直径为 $n(k-1)$，每个节点可用地址 $(u_n,u_{n-1},\cdots,u_1)$ 描述，其中 $1\leqslant u_i\leqslant k$，$1\leqslant i\leqslant k$。如果两个节点 $(u_n,u_{n-1},\cdots,u_1)$ 和 $(v_n,v_{n-1},\cdots,v_1)$ 的地址中有且仅有一个元（维）不同，比如说第 i 维，并且 $|u_i-v_i|=1$，则它们相互连接。对于网格来说，沿着每一维的节点构成一条直线。图 1.11 是一个 3

元2维网格，2维和3维网格是最流行的，许多并行计算机都是基于它们构造的。

(4)圆环。在一网格中，如果沿着每一维的节点连成环形，则该网络形成一个 n 维的圆环。一个 k 元 n 维圆环，有 k^n 个节点，每个节点度数为 $2n$，网络直径为 $n(k-1)/2$。圆环是带回绕连接的网格。图1.12是一个3元2维圆环。

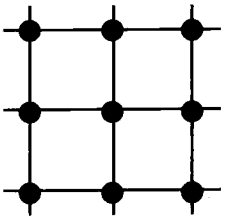

图1.11 3元二维网格

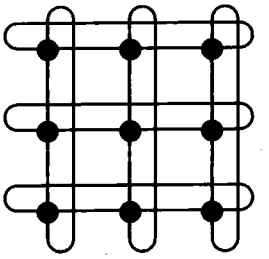

图1.12 3元二维圆环

(5)弦环网络。在环形网络中增加额外连接，使得图变成度数为3,4,…的正则图(每个节点的度数都相同的图称为正则图，k-正则图就是每个节点的度数均为 k)，就得弦环网络。一个度数为 d 的弦环网络，节点编号为 $0,1,2,\cdots,n-1$，若节点 i 与节点 j 相连，则有 $j=(i+d) \bmod n, 3\leqslant d\leqslant n-1$。

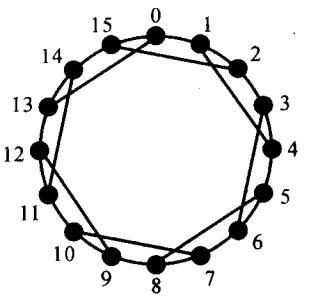

图1.13 3度弦环网络

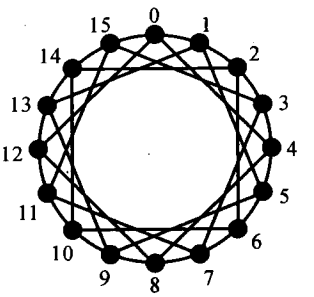

图1.14 4度弦环网络

(6)Barrel Shifter 网络。该网络是在弦环网络中增加以下额外的连接得到的：节点 i 都和与其距离为2的幂次方的节点 j 相连接：$j=i+2^k, k=0,1,2,\cdots$。如图1.15所示。

(7)树型和星型。任意两个顶点间只存在唯一通路的图就是树。树中的节点可以按层次排列，在二叉树(见图1.16(a))中，内部节点度数为3，有多个子节点和一个父节点，树根度数为2，叶节点度数为1。星型(见图1.16(b))是两层的树，有一个节点的度数最高，该节点为中心节点。

(8)超立方型。一个 n 维的超立方体(又称 n-立方体)包含 $N=2^n$ 个节点，节点度数为 n，直径为 n。每个节点有一个 n 位的二进制地址。两个节点的二进制标识当且仅当有1位不同时，这两个节点相邻。一般来说，任一节点 $(b_1,b_2,\cdots,b_n)$ 通过边与下述 n 个节点连接(其中的 $\tilde{b}_i$ 表示对该位取反，$i=1,2,\cdots,n$)：

$$\tilde{b}_1 b_2 b_3 \cdots b_n, b_1 \tilde{b}_2 b_3 \cdots b_n, \cdots, b_1 b_2 b_3 \cdots \tilde{b}_n$$

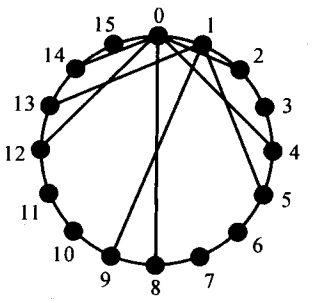

图 1.15　Barrel Shifter 网络

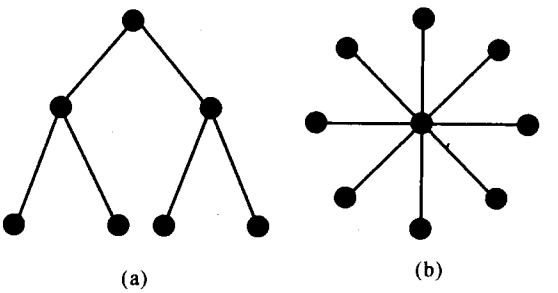

图 1.16　二叉树型和星型
(a)二叉树型；(b)星型

例如，在 3 -立方体里，000 与 100，010 与 001 连接；001 与 101，001 与 000 连接。从 n -立方体的定义来看，我们可以观察到如下的属性：

1)属性 1。n -立方体就是有 2^n 个节点的 n 正则图。

2)属性 2。两个节点 $a=(a_1,a_2,\cdots,a_n)$ 和 $b=(b_1,b_2,\cdots,b_n)$ 之间的距离是 a 和 b 之间不同的二进制位的个数，即海明距离。

例如，在 5 -立方体里，(11100)和(11011)之间的距离是 3。这是因为它们有 3 个二进制位不同。(11110)与(11010)相邻，(11010)与(11011)相邻。

3)属性 3。设 $e(k)$ 为 k -立方体的边数，有如下的递归方程：

$$e(k)=2e(k-1)+2^{k-1}$$

递归代入 $e(k)$，就有 $e(k)=k2^{k-1}$。如果 $n=2^k$ 是 k -立方体的节点个数，就有：

$$e(k)=k2^{k-1}=(n\text{lb}n)/2$$

这就是 k -立方体的边的个数。

实际上，节点度数随着维数的增加线性增大，使得超立方很难是一个可伸缩的结构。由于难以伸缩和扩展到更高维数的超立方，超立方结构逐渐被其他结构所取代了。但是，在超立方上设计的算法介于网络相关的算法和网络无关的算法之间。超立方算法的研究对于弥补使用 PRAM(并行随机访问机)的网络无关设计和使用网络模型的网络相关设计之间的差距仍然十分重要。图 1.17 表示了一个 3 维超立方体。

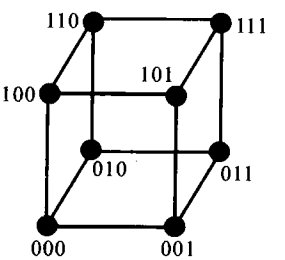

图 1.17　3 维超立方体

选择一个互连网络时，通常使用以下度量参数：节点度数、直径和对分宽度(即把一个网络分成相等的两部分所割到的边的最小数目)。

(9)多边星型。假定 k 是正整数，考察 k 个符号的置换。对 k 个符号，有 k 个置换。相应于 k 个置换可以定义 k 个节点。当且仅当两节点相应的排列最左位和任意一个其他位置不同时，这两个节点相邻。由此得到的图被称做 k -星型图。以 $k=3$ 为例。在这种情况下，有 6 个置换：

$P_0=(1,2,3)$，　$P_1=(1,3,2)$，　$P_2=(2,1,3)$，　$P_3=(2,3,1)$，　$P_4=(3,1,2)$，　$P_5=(3,2,1)$

可简写为：$P_0(1)=1,P_0(2)=2$ 和 $P_0(3)=3;P_1(1)=1,P_1(2)=3$ 和 $P_1(3)=2;\cdots$。

在图 1.18 中给出相应的 3 -星型图。星型图也可以用递归的方式定义为：为了构建 k -星型图，考虑有 k 份 $(k-1)$-星型图。在任意一个 $(k-1)$-星型图中，每个顶点都用 $k-1$ 个符号

的置换标识。替换方法如下：

1)复制 k 份($k-1$)-星型图。

2)在 k 份($k-1$)-星型图每个顶点坐标的最右边加入符号 i，$i=1,2,\cdots,k$。

3)最右边新加入的符号 i 位不变，用 $1,2,\cdots,i-1,i+1,\cdots,k$ 将其他位置上的符号 i 替换成缺失的那个符号。

4)生成如下的新边：当且仅当两节点相应的排列最左位和任意一个其他位置不同，而其他位置相同时，这两个节点相邻。

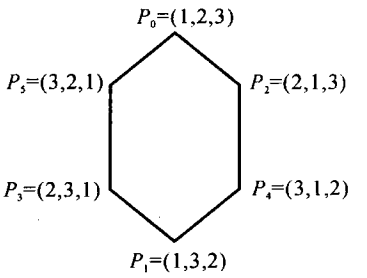

图 1.18 3-星型图

k-立方体和 k-星型图的拓扑的基本参数如表 1.1 所示。

表 1.1 k-立方体和 k-星型图的拓扑的基本参数

性质	k-立方体	k-星型图
顶点数	2^k	$k!$
度	k	$k-1$
直径	k	$3(k-1)/2$

例 1.1 生成 2-星型图，1-星型图的 1 个节点为(1)。

对节点(1)最右边补 1，得(1,1)，用 2 替换前面的 1，得(2,1)。

对节点(1)最右边补 2，得(1,2)，用 1 替换前面的 1，得(1,2)。

生成如下的新边：因(2,1)和(1,2)的最左位和任意一个其他位置不同，而其他位置相同，则(2,1)和(1,2)相连。

例 1.2 生成 3-星型图，2-星型图的 2 个节点为(1,2)，(2,1)。

对节点(1,2)，(2,1)最右边补 1，得(1,2,1)，(2,1,1)，用 2,3 置换前面的 1，得(3,2,1)，(2,3,1)。

对节点(1,2)，(2,1)最右边补 2，得(1,2,2)，(2,1,2)，用 1,3 置换前面的 2，得(1,3,2)，(3,1,2)。

对节点(1,2)，(2,1)最右边补 3，得(1,2,3)，(2,1,3)，用 1,2 置换前面的 3，得(1,2,3)，(2,1,3)。

生成如下的新边：

(1,2,3)和(3,2,1)相连，(1,2,3)和(2,1,3)相连；
(3,2,1)和(2,3,1)相连，(3,2,1)和(1,2,3)相连；
(2,3,1)和(1,3,2)相连，(2,3,1)和(3,2,1)相连；
(1,3,2)和(3,1,2)相连，(1,3,2)和(2,3,1)相连；
(3,1,2)和(2,1,3)相连，(3,1,2)和(1,3,2)相连；
(2,1,3)和(1,2,3)相连。(2,1,3)和(3,1,2)相连。

例 1.3 生成 4-星型图，3-星型图的 6 个节点为(1,2,3)，(2,1,3)，(3,1,2)，(1,3,2)，(2,3,1)，(3,2,1)。

最右边补 1，用 2,3,4 对上述 6 个节点进行第 1 次置换，得

(4,2,3,1)，(2,4,3,1)，(3,4,2,1)，(4,3,2,1)，(2,3,4,1)，(3,2,4,1)

最右边补 2,用 1,3,4 对上述 6 个节点进行第 2 次置换,得

(1,4,3,2),(4,1,3,2),(3,1,4,2),(1,3,4,2),(4,3,1,2),(3,4,1,2)

最右边补 3,用 1,2,4 对上述 6 个节点进行第 3 次置换,得

(1,2,4,3),(2,1,4,3),(4,1,2,3),(1,4,2,3),(2,4,1,3),(4,2,1,3)

最右边补 4,用 1,2,3 对上述 6 个节点进行第 4 次置换,得

(1,2,3,4),(2,1,3,4),(3,1,2,4),(1,3,2,4),(2,3,1,4),(3,2,1,4)

生成如下的新边:

(1,2,3,4)和(2,1,3,4)相连,(1,2,3,4)和(3,2,1,4)相连,(1,2,3,4)和(4,2,3,1)相连;
(2,1,3,4)和(3,1,2,4)相连,(2,1,3,4)和(1,2,3,4)相连,(2,1,3,4)和(4,1,3,2)相连;
(3,1,2,4)和(1,3,2,4)相连,(3,1,2,4)和(2,1,3,4)相连,(3,1,2,4)和(4,1,2,3)相连;
(1,3,2,4)和(4,3,2,1)相连,(1,3,2,4)和(3,1,2,4)相连,(1,3,2,4)和(2,3,1,4)相连;
(2,3,1,4)和(3,2,1,4)相连,(2,3,1,4)和(1,3,2,4)相连,(2,3,1,4)和(4,3,1,2)相连;
(3,2,1,4)和(1,2,3,4)相连,(3,2,1,4)和(2,3,1,4)相连,(3,2,1,4)和(4,2,1,3)相连。

1.3 设计模型

一个分布式系统是一个对用户看起来像普通系统,然而运行在一系列自治处理单元(PE)上的系统,每个 PE 有各自的物理存储器空间并且信息传输延迟不能忽略不计。在这些 PE 间有紧密的合作。系统必须支持任意数量的进程和 PE 的动态扩展。图 1.19(a)为基于物理观点(b)为逻辑观点的系统结构。

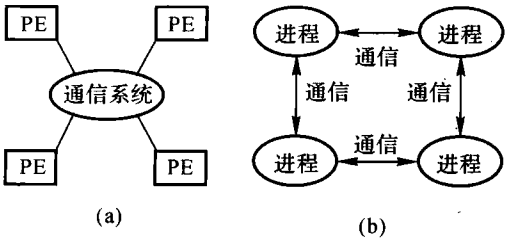

图 1.19 设计模型的系统结构

(a); (b)

1.3.1 同步网络模型

同步网络系统由一组位于有向网络图节点位置的 PE 组成,自现在开始,我们约定 PE 为进程。为了形式化地定义同步网络系统,从有向图 G 开始,$G=(V,E)$。用 n 表示有向图 G 的节点个数,即 $n=|V|$。对于 G 中的任意节点 i,out_nbrs_i 表示 i 的出向邻接点,即在 G 中存在 i 到这些节点的有向边;in_nbrs_i 表示 i 的入向邻接点,即在 G 中存在这些节点到 i 的有向边;distance(i,j)表示在 G 中从 i 到 j 的最短有向路径的长度;若不存在这样的路径,则 distance$(i,j)=\infty$;diam 为图 G 的直径,即所有(i,j)中 distance(i,j)的最大值。假设 M 为一个固定消息的集合,null 表示空消息。对于每个 $i\in V$,存在一个进程,它在形式上由下列几个部分组成:

(1)$states_i$:一组状态的集合,不一定是有限集;

(2)$start_i$:$states_i$ 的非空子集,称为开始状态集或初始状态集;

(3)msg_i:一个消息生成函数,从 $states_i \times out\_nbrs_i \rightarrow M \cup \{null\}$;

(4)$trans_i$:一个状态转移函数,从状态集和 $M \cup \{null\}$ 中元素组成的向量(以 in_nbrs_i 为下标)映射到 $states_i$。

对于每个进程都有一个状态集合,其中包括开始状态集。状态集不一定是有限集。对于每个状态和出向邻节点,消息生成函数定义了:从给定状态开始,进程 i 向指定的邻节点发出消息。状态转移函数则定义了:针对每个状态和所有来自入向邻节点的消息的集合,进程 i 所要迁移的状态。对应于 G 中的每一条边(i,j),存在一条通道(也称为链路),它是在任意时刻持有 M 中最多一条消息的一个场所。

整个系统的运行以所有进程处于任意开始状态和所有通道都为空开始。之后,所有的进程在每个时间步重复执行下面的步骤:

(1)对当前状态应用消息生成函数,生成向所有出向邻节点发送的消息;把这些消息放在相应的通道上。

(2)对当前状态和入向消息应用状态转移函数,获得新的状态;删除入向通道上的所有消息。

上述两步的组合成为一轮(Round)。进程停止是指不会有任何消息生成,同时唯一的状态转移是自循环。

为了推断同步网络系统的行为,我们需要对系统的运行有一个形式化的定义:一个系统的状态赋值(State Assignment)是对系统中的每个进程赋值一个状态。消息赋值(Message Assignment)是对每个通道赋值一条消息(也可以是空消息)。系统的运行定义为一个无限序列:

$$C_0, M_1, N_1, C_1, M_2, N_2, C_2, \cdots$$

其中,C_r 是状态赋值,M_r 和 N_r 为消息。C_r 代表 r 轮后系统的状态,而 M_r 和 N_r 则代表第 r 轮所发送和接收的消息,因为通道可能丢失消息,所以 $M_r \neq N_r$。我们通常把 C_r 当成时刻 r 的状态赋值(也就是执行 r 轮后系统的状态)。

$$\underbrace{\cdots, C_0}_{\text{第0轮}}, \underbrace{M_1, N_1, C_1}_{\text{第1轮}}, \underbrace{M_2, N_2, C_2}_{\text{第2轮}}, \cdots, \underbrace{M_r, N_r, C_r}_{\text{第}r\text{轮}}, \cdots$$

设 α 和 α' 是一个系统的两次运行,若在 α 和 α' 两次运行中,进程 i 有相同的状态序列、相同的输入消息序列和相同的输出消息序列,则称 α 和 α' 对于进程 i 来说是不可区分的,记作 $\alpha \overset{i}{\sim} \alpha'$。若在 α 和 α' 两次运行中,进程 i 直到第 r 轮运行完后都有相同的状态序列、相同的输入消息序列和相同的输出消息序列,则称 α 和 α' 对于进程 i 来说是 r 轮不可区分的。

1.3.2 异步系统模型

在异步模型中,系统组件以任意速度运行。与同步模型一样,异步模型不难描述。异步模型与同步模型之间的细微差别主要在于它涉及活性条件。例如,异步模型要求每一组件不断地得到机会来运行。然而,由于事件发生顺序的不确定性,异步模型比同步模型更难编程、对时间的要求更低。因此,为异步模型所设计的算法是通用的、可移植的,能够保证在具有任意时序行为的网络中正确运行。下面描述两个重要的异步模型。