

并行算法的设计 与分析

陈国良

高等教育出版社

并行算法的设计与分析

陈 国 良

高等教育出版社

(京)112号

内 容 提 要

本书比较系统、全面地讨论了各种专用和通用并行计算模型上的算法的设计和分析方法。书中以并行计算模型为线索,强调算法、结构和模型三者之间的密切关系,着重介绍了各种最基本、常用和典型的并行算法,同时也力图反映本学科的最新成就和发展趋势。

全书共分二十章,主要包括:并行算法基础;并行算法的基本设计技术;各种计算模型上的计算机领域中诸多常用计算问题的并行算法的设计和分析方法;最后还讨论了各种并行计算模型的能力、限制及其等价性以及与并行计算有关的 NC 理论问题。

本书内容丰富,取材新颖,叙述清晰,逻辑性强。可作为高等学校计算机有关专业的高年级学生和研究生选修教材;也可供从事计算机科学理论和算法研究的科技人员阅读参考。

并行算法的设计与分析

陈 国 良

高等教育出版社出版

新华书店总店科技发行所发行

河北省香河县印刷厂印装

开本 787×1092 1/16 印张 33 字数 757 000

1994年5月第1版 1994年5月第1次印刷

印数 0 001—2 135

ISBN 7-04-004612-1/TP·135

定价 12.60 元

前 言

最近几年来, 由于元器件极限速度的限制, 使得高性能计算机的研究热点主要集中在计算并行度的开发上, 这样便促进了并行算法研究的深入和迅速发展. 在此情况下, 尽管目前并行算法的理论体系仍处于发展阶段, 但其很多基本内容, 研究方法已渐趋稳定和成熟, 从而有可能将并行算法这门学科从学术论坛上搬上大学的讲台. 作者从1984年以来已为计算机系的研究生和高年级学生讲授此课程多次, 每次讲稿都有较大的变化, 内容都不断地充实和完善. 在此基础上, 1992年又广泛地征求了国内同行专家的意见, 同时也及时参照了国外最新同类教材, 最后报请国家教育委员会高等学校计算机科学教学指导委员会审定, 确定了该课程的教学大纲和书稿的基本内容.

按照审定的内容, 本课程是属于算法研究的三个层次(并行算法理论, 并行算法的设计和分析, 并行算法的实现)的中间层次, 即它不去重点研究算法理论中的某些基本问题, 而是着重研究可有效并行求解问题类(即用多项式数目的处理器可在多对数时间内求解的 NC-类问题)的算法的设计和分析方法, 而且也略去了并行算法的具体实现环节(如并行语言、编译、执行环境与工具等). 同时本课程也是遵循计算机学科中有关算法研究的经典内容进行组织的, 它严格区别于并行数值计算与分析课程(这些课程主要是研究数学计算原理、方法、精度与稳定性等问题). 所以对于本课程所涉及的一些数值计算问题, 乃是按照上述原则, 直接根据数值分析中的数学原理, 密切结合体系结构和计算模型, 详细讨论其有效并行算法的设计和分析的方法.

全书共分二十章, 大体上可分为三大部分: 第一部分讲述并行算法的基础知识及其基本设计技术(第一、二章), 包括并行机分类及处理器互连网络、并行计算模型、并行算法的定义、分类、表达和度量以及并行算法的基本设计方法等; 第二部分讨论各种专用和通用并行计算模型上的计算机领域中诸多常用计算问题(诸如排序, 选择, 归并, 搜索, 组合运算, 模式串匹配, 表达式求值, 语言识别和语法分析, 矩阵运算, 数值求解, FFT 变换, 计算几何, 图论问题, 组合搜索等)的确定性并行算法(第三章至第十七章)和非确定性并行算法(第十八章)的设计和分析方法; 第三部分涉及到算法理论中的一些问题(第十九、二十章), 主要包括 VLSI 计算理论和并行计算模型的能力、限制及其等价性以及并行计算有关的 NC-理论问题. 书中取材内容具有相当的广度, 基本上包含了并行算法学科中主要的研究内容和主要的研究方面.

国家教育委员会已把并行算法的设计和分析列为高等学校计算机专业的专业选修课. 学生应在学过数据结构、计算机体系结构、图论和算法设计与分析等课程之后学习本课程. 全书的内容可根据不同的教学学时的规定和不同的读者对象进行不同的组合. 作为必须讲授和最低 60 学时的教学要求, 建议讲授章节及其相应的学时分配如下:

第一章(2学时): 1.1 节~1.3 节;

第二章(2学时): 2.2 节~2.4 节;

第三章(2学时): 3.1节, 3.2节;

第四章(4学时): 4.1节, 4.4节;
4.5节, 4.7节;

第五章(2学时): 5.1节, 5.3节;

第六章(2学时): 6.2节, 6.4节;

第七章(2学时): 7.1节, 7.3节;

第八章(2学时): 8.2节, 8.4节;

第九章(3学时): 9.1节, ~9.3节;

第十章(2学时): 10.1节, 10.3节;

第十一章(2学时): 11.1节, 11.2节;

第十二章(4学时): 12.2~12.4节;

第十三章(3学时): 13.1节, 13.2节, 13.4节;

第十四章(4学时): 14.1节~14.3节;

第十五章(4学时): 15.2节~15.4节;

第十六章(3学时): 16.1节, 16.4节,
16.6节;

第十七章(4学时): 17.1节~17.4节;

第十八章(4学时): 18.2节, 18.3节;
18.6节;

第十九章(4学时): 19.1节, 19.3节, 19.5节.

书中其余部分的内容是作为学生阅读和参考用的, 而带有*号的章节是任选的, 它们或是预备性的知识(要求读者课前预习), 或是深入研究性质的内容(鼓励面向研究的读者阅读). 每章之后均附有适量的密切结合课文的习题; 同时也开列了本章所引用的主要参考文献. 全书最后附录了算法复杂度一览表, 以便读者随时查阅. 同时为了供读者不断补充某些计算问题的最新研究成果, 最后还提供了一张备用表.

本书在撰写中, 曾直接或间接地引用了许多专家、学者的文献, 作者深表感谢; 但也有很多作者的优秀论文未能被引用, 作者亦深表歉意. 在本书的内容取舍和章节安排方面, 马绍汉教授、唐策善教授、康立山教授、张德富教授和张丽君教授等提出了很多宝贵意见; 书稿付梓前, 承蒙张德富教授进行了终审, 提出了不少中肯的修改意见, 作者谨此一并表示谢意.

中国科学技术大学的历届学生们, 在听取我的讲授中, 曾提出过很多可贵的建议, 不断地丰富和充实了本书的内容. 对于他们的辛勤劳动和良好的愿望, 作者尤为感谢.

《并行算法的设计与分析》一书, 在国内被推荐为高等学校试用教材, 作者感到非常高兴, 但也深深地意识到, 限于作者水平, 加之时间仓促, 书中定有不少欠妥和错误之处, 恳请诸位批评指正.

陈国良

于中国科学技术大学

1993年6月3日

目 录

前言	1
第一章 导论	1
1.1 并行计算机的体系结构及分类	1
1.1.1 并行处理系统发展谱	1
1.1.2 并行计算机的分类	2
*1.1.3 处理器互连网络	3
1.2 并行计算模型	9
1.2.1 SIMD 互连网络模型	9
1.2.2 SIMD 共享存储模型	10
1.2.3 MIMD 共享存储模型	11
1.2.4 MIMD 异步通信模型	11
1.2.5 专用并行计算模型	11
1.2.6 其他并行计算模型	12
1.3 并行算法的一般概念	13
1.3.1 并行算法的定义和术语	13
1.3.2 并行算法的复杂度	14
1.3.3 并行算法的表达	16
1.3.4 并行算法的 WT 表示	17
*1.3.5 并行算法的同步和通信	20
习题	23
参考文献	24
第二章 并行算法的基本设计技术	26
2.1 平衡树方法	26
2.1.1 求取最大值	26
2.1.2 计算前缀和	27
2.2 倍增技术	30
2.2.1 表序问题的计算	30
2.2.2 求森林的根	31
2.3 分治策略	32
2.3.1 SIMD 模型上分治算法的描述	32
2.3.2 SIMD 共享存储模型上的 FFT 算法	33
2.4 划分原理	35
2.4.1 归并原理	35
2.4.2 划分算法与归并算法	36
2.5 流水线技术	37
2.5.1 一维阵列上的流水线归并排序原理	37
2.5.2 一维阵列上的流水线归并排序算法	37
2.6 加速级联策略	39
2.6.1 常数时间求最大值算法	39
2.6.2 双对数时间算法	40

2.6.3 加速级联算法	41
*2.7 破对称技术	41
2.7.1 基本着色算法	41
2.7.2 快速 3-着色算法	42
2.7.3 最优 3-着色算法	43
习题	44
参考文献	46
第三章 比较器网络上的排序和选择算法	47
3.1 Batcher 归并和排序网络	47
3.1.1 比较操作和 $[0, 1]$ 原理	47
3.1.2 奇偶归并网络	48
3.1.3 双调归并网络	49
3.1.4 Batcher 排序网络	51
3.2 (m, n) -选择网络	53
3.2.1 分组选择网络	53
3.2.2 平衡分组选择网络	55
*3.3 AKS 排序网络	56
3.3.1 扩展图和划分网络	56
3.3.2 部分排序算法	59
3.3.3 完全排序算法	61
习题	65
参考文献	66
第四章 排序和选择的同步算法	67
4.1 Stone 双调排序算法	67
4.1.1 均匀洗牌函数及其性质	67
4.1.2 Stone 的观察及其计算模型	68
4.1.3 Stone 的并行排序算法	69
4.2 Thompson 和 Kung 双调排序算法	71
4.2.1 处理器编号方式	71
4.2.2 Thompson 和 Kung 的观察	72
4.2.3 Thompson 和 Kung 的双调排序算法	73
*4.3 Preparata 和 Vuilemin 双调排序算法	75
4.3.1 算法原理	75
4.3.2 流水线技术	76
4.3.3 算法描述	77
4.4 Akl 并行 k -选择算法	79
4.4.1 算法原理及物理描述	79

4.4.2 并行 k -选择算法	79	6.1.1 单处理机上的顺序搜索	118
4.4.3 算法分析	81	6.1.2 单处理机上有有序表的对半搜索	119
4.5 Valiant 并行归并算法	82	6.2 SIMD 共享存储模型上有序表的搜索	119
4.5.1 归并算法的基本原理	82		
4.5.2 $k = \lfloor \sqrt{pq} \rfloor$ 时 Valiant 归并	82	6.2.1 SIMD-EREW 模型上的搜索	119
4.5.3 $k = \lfloor \tau \sqrt{pq} \rfloor$ 时 Valiant 归并	83	6.2.2 SIMD-CREW 模型上的搜索	120
*4.6 Hirschberg 并行桶排序算法	84	6.3 SIMD 共享存储模型上随机序列的搜索	123
4.6.1 并行桶排序算法原理	84		
4.6.2 并行桶排序算法描述	85	6.3.1 SIMD-SM 模型上的随机序列搜索算法描述	123
4.7 Preparata 并行枚举排序算法	86	6.3.2 SIMD-SM 模型上的随机序列搜索算法分析	124
4.7.1 枚举排序及其实现方法	86	6.4 树连接的 SIMD 模型上随机序列的搜索	124
4.7.2 排序算法的设计和分析	87		
*4.8 Cole 并行归并排序算法	89	6.4.1 提问	124
4.8.1 使用覆盖和位序的归并方法	89	6.4.2 维护	125
4.8.2 Cole 最佳排序算法	91	6.5 网孔连接的 SIMD 模型上随机序列的搜索	126
4.8.3 算法的正确性证明及分析	95		
习题	98	6.5.1 提问	127
参考文献	99	6.5.2 维护	130
第五章 排序和选择的异步和分布式算法	100	*6.6 MIMD 共享存储模型上有序表的搜索	130
5.1 MIMD-CREW 模型上的异步枚举排序算法	100		
5.1.1 算法原理和描述	100	6.6.1 AVL 树及其顺序插入算法	130
5.1.2 算法举例和分析	101	6.6.2 Ellis 并行搜索和插入算法	132
5.2 MIMD-TC 模型上的异步快排序算法	102	习题	134
		参考文献	134
5.2.1 算法原理和描述	102	第七章 排列和组合	135
5.2.2 算法举例和分析	103	7.1 产生排列的顺序算法	135
5.3 分布式 k -选择算法	104	7.1.1 排列和组合	135
5.3.1 随机 k -选择算法	104	7.1.2 产生词典序的排列算法	136
5.3.2 确定 k -选择算法	106	7.1.3 产生排列的计数方法	138
5.4 分布式求中值算法	107	7.2 产生组合的顺序算法	140
5.4.1 分布式中值	107	7.2.1 产生词典序的组合算法	140
5.4.2 分布式求中值算法	108	7.2.2 产生组合的计数方法	142
*5.5 分布式定序算法	109	7.3 产生排列的并行算法	144
5.5.1 分布式计算模型	110	7.3.1 串行排列算法的并行化	149
5.5.2 分布式定序算法	110	7.3.2 自适应排列产生器	148
5.5.3 算法复杂度分析	112	7.3.3 全排列产生器	149
*5.6 分布式排序算法	113	7.4 产生组合的并行算法	150
5.6.1 模型和定义	113	7.4.1 非自适应组合产生器	151
5.6.2 静态排序算法	114	7.4.2 自适应组合产生器	154
5.6.3 算法复杂度分析	114	习题	155
习题	115	参考文献	156
参考文献	116	第八章 数据传输与选路	157
第六章 并行搜索	118	8.1 引言	157
6.1 单处理机上的搜索	118	8.2 贪心选路算法	158

8.2.1 一维阵列上的贪心选路算法	158
8.2.2 二维阵列上贪心选路算法的分析	159
8.2.3 蝶形网络上的贪心选路算法	161
8.3 随机和确定选路算法	162
8.3.1 二维阵列上的随机选路算法	162
8.3.2 超立方网络上的随机选路算法	164
8.3.3 二维阵列上的确定选路算法	165
8.4 数据的分布和集中	167
8.4.1 数据的分布	167
8.4.2 多到一选路算法	169
8.5 线路交换模式下的选路算法	171
习题	173
参考文献	174
第九章 并行串匹配	176
9.1 引言	176
9.1.1 基本概念和研究现状	176
9.1.2 基本定义和术语	177
9.2 正文分析	178
9.2.1 SIMD-CREW 模型上的非周期串的匹配算法	178
9.2.2 SIMD-CREW 模型上的周期串的匹配算法	181
9.3 模式预处理	183
9.3.1 求 WIT 表的一般方法	183
9.3.2 完全非周期串的 WIT 表求法	184
9.3.3 SIMD-CREW 模型上任意串的 WIT 表求法	187
9.3.4 模式匹配算法的复杂度	188
*9.4 后缀树上的串匹配	188
9.4.1 数字搜索树与后缀树	189
9.4.2 子串的编码	190
9.4.3 后缀树的构造	192
9.4.4 后缀树上的并行串匹配	195
习题	196
参考文献	198
第十章 表达式求值	199
10.1 构造表达式树	199
10.1.1 全括号表达式的表达式树	199
10.1.2 表达式树上的括号操作	200
10.1.3 计算 $match(i)$ 的并行算法	201
10.2 填充游戏用于表达式求值	202
10.2.1 二叉树上的填充游戏	202
10.2.2 填充游戏用于算术表达式求值	203
10.3 最优的并行表达式求值算法	205
10.4 一般表达式求值算法	208
10.4.1 一般表达式与直线程序	208
10.4.2 仅有乘法操作符的 dag 的计算	209

10.4.3 仅有加法操作符的 dag 的计算	209
10.4.4 gbdag 图和直线程序的计算	210
10.5 正则表达式到非确定自动机的最优并行转换	213
10.5.1 基本概念和术语	213
10.5.2 SIMD-CREW 模型上的 HU 转换方法	215
习题	217
参考文献	218

第十一章 上下文无关语言的并行识别

与语法分析

11.1 一般的上下文无关语言的并行识别	219
11.1.1 基本概念和术语	219
11.1.2 残缺部分语法树及其合成规则	221
11.1.3 共享存储模型上歧义性上下文无关语言并行识别算法	223
11.2 一般上下文无关语言的并行语法分析	226
11.2.1 基本概念和算法原理	226
11.2.2 SIMD-CREW 模型上一般上下文无关语言的语法分析算法	227
11.3 括号语言的最优并行识别和语法分析	230
11.3.1 基本概念和算法原理	230
11.3.2 算法的具体实现	231
11.3.3 树的压缩技术	233
11.3.4 SIMD-CREW 模型上括号语言的语法分析算法	236
习题	236
参考文献	237

第十二章 矩阵运算

12.1 矩阵转置	238
12.1.1 单处理机上的矩阵转置算法	238
12.1.2 SIMD-MC ² 模型上的矩阵转置	238
12.1.3 SIMD-PS 模型上的矩阵转置	241
12.1.4 SIMD-EREW 模型上的矩阵转置	242
12.2 矩阵相乘	243
12.2.1 单处理机上的矩阵乘法	243
12.2.2 SIMD-MC ² 模型上的矩阵乘法	244
12.2.3 SIMD-OC 模型上的矩阵乘法	245
12.2.4 MIMD 机器上的矩阵乘法	248
12.3 矩阵和向量相乘	251
12.3.1 树连接的机器上的矩阵和向量乘法	251
12.3.2 树网结构上的矩阵和向量乘法	253
12.4 心动阵列上的矩阵运算	253
12.4.1 二维六角形阵列上的矩阵乘法	253

12.4.2	二维六角形阵列上方阵的 LU 分解	255
*12.4.3	六角形阵列上的方阵求逆	258
12.4.4	一维阵列上求解三角形线性系	259
习题		261
参考文献		264
第十三章 数值计算		265
13.1	n 阶线性代数方程组的求解	265
13.1.1	SIMD-CREW 模型上的 Gauss-Jordan 算法	266
13.1.2	MIMD-CREW 模型上的 Gauss-Seidel 算法	268
*13.1.3	紧耦合多处理机系统中 LU 算法的效率分析	270
13.2	非线性方程的求根	272
13.2.1	SIMD-CREW 模型上的求根算法	272
13.2.2	MIMD-CREW 模型上的牛顿求根法	274
*13.2.3	Fibonacci 分点法异步求根算法	275
*13.3	偏微分方程的求解	278
13.3.1	偏微分方程的差分数值求解法	278
13.3.2	SIMD-MC ² 模型上的 PDE 求解方法	279
13.4	方阵的特征值与特征向量 Jacobi 求法	282
13.4.1	对称方阵对角化方法	282
13.4.2	SIMD-CC 模型上的求特征值算法	284
习题		286
参考文献		287
第十四章 FFT 和卷积与滤波		289
14.1	快速富立叶变换	289
14.1.1	离散富立叶变换	289
14.1.2	顺序的 FFT 算法	290
*14.1.3	FFT 应用于多项式乘积	291
14.2	DFT 直接并行计算法	292
14.2.1	SIMD 模型上系数矩阵的计算	292
14.2.2	SIMD-MT 模型上的 DFT 算法	292
14.3	并行 FFT 算法	295
14.3.1	SIMD-MC ² 模型上的 FFT 算法	295
14.3.2	SIMD-BF 模型上的 FFT 算法	298
*14.3.3	SIMD-PS 模型上的 FFT 计算	299
14.3.4	SIMD-CC 模型上的 FFT 计算	301
*14.3.5	一维心动阵列上的 DFT 计算	305
14.4	心动阵列上的卷积与滤波计算	306
14.4.1	一维卷积在线性阵列上的实现	306
14.4.2	无限冲激滤波在线性阵列上的实现	308
14.4.3	中值滤波在线性阵列上的实现	309
习题		311

参考文献	312
------	-----

第十五章 图论算法 313

15.1	图的并行搜索	313
15.1.1	算法原理	313
15.1.2	p -深度优先搜索	314
15.1.3	p -宽深优先搜索	314
15.1.4	p -宽度优先搜索	314
15.2	图的传递闭包	316
15.2.1	传递闭包问题	316
15.2.2	SIMD-CC 模型上的传递闭包算法	316
*15.2.3	二维心动阵列上的传递闭包算法	317
15.3	图的连通分量	320
15.3.1	SIMD-CC 模型上的连通分量算法	320
15.3.2	SIMD-SM 模型上的连通分量算法	322
15.3.3	SIMD-TC 模型上的连通分量算法	324
15.3.4	SIMD-MT 模型上的连通分量算法	325
15.4	图的最短路径	327
15.4.1	所有顶点对间的最短路径算法	328
*15.4.2	MIMD-SM 模型上单源最短路径算法	330
15.5	图的最小生成树	333
15.5.1	SIMD-EREW 模型上最小生成树算法	333
*15.5.2	MIMD-SM 模型上最小生成树算法	336
15.5.3	树机模型上最小生成树算法	339
*15.6	图的着色	342
15.6.1	二分图的边着色算法	342
15.6.2	外平面图最优顶点着色算法	343
15.6.3	外平面图最优边着色算法	349
15.6.4	Halin 图最优边着色算法	351
习题		355
参考文献		358

第十六章 图象分析和计算几何 360

16.1	分量标定	360
16.1.1	二维阵列上分量标定平易算法	360
16.1.2	二维阵列上 Levaldi 分量标定算法	362
16.1.3	二维阵列上递归的分量标定算法	364
16.2	Hough 变换	365
16.2.1	二维图象的 Hough 变换	365
16.2.2	二维阵列上象素 Hough 变换的计算	366
16.3	近邻问题	367
16.4	包含问题	368
16.4.1	判断点在多边形中	369
16.4.2	判断点在平面细图中	371
16.5	相交问题	373
16.6	构造问题	374

16.6.1 求凸壳问题的下界	374	18.6.3 快速随机并行排序算法	490
16.6.2 顺序求凸壳算法	375	*18.7 最大匹配和完备匹配	432
16.6.3 SIMD-MT 模型上求凸壳算法	376	18.7.1 图的代数性质	433
16.6.4 SIMD-EREW 模型上求凸壳算法	378	18.7.2 测试完备匹配存在的随机算法	435
习题	383	习题	438
参考文献	384	参考文献	439
第十七章 组合搜索	385	第十九章 VLSI 计算理论	441
17.1 基于分治法的与树搜索	385	19.1 VLSI 电路模型和计算模型	441
17.1.1 搜索树	385	19.1.1 VLSI 电路模型	441
17.1.2 SIMD 模型上的与树搜索	386	19.1.2 VLSI 计算模型	443
17.2 基于分枝限界法的或树搜索	386	*19.2 VLSI 面-时下界理论	444
17.2.1 8-谜问题	387	19.2.1 几种基本的下界论点	444
17.2.2 串行分枝限界算法	388	19.2.2 信息流和穿越序列	446
17.2.3 用串行分枝限界法求 TSP	389	19.3 典型计算图的结构布局法	448
17.2.4 并行 TSP 算法	393	19.3.1 树的布局	448
17.3 串行的 α - β 搜索算法	394	19.3.2 网孔和树网的布局	449
17.3.1 博弈树与最大最小原理	394	19.3.3 洗牌交换网的布局	449
17.3.2 串行的 α - β 算法	395	19.3.4 立方环的布局	452
17.4 树机上的并行搜索算法	396	19.3.5 蝶形网的布局	453
17.4.1 树分裂算法	397	19.4 典型计算图的布局下界	453
17.4.2 主变量分裂算法	398	19.4.1 树的布局下界	453
*17.5 MIMD 模型上 α - β 搜索算法	400	19.4.2 树网的布局下界	455
17.5.1 基本设计原理	400	19.4.3 洗牌交换网的布局下界	458
17.5.2 算法的形式描述	401	19.4.4 蝶形网的布局下界	458
17.5.3 算法讨论和分析	405	19.5 分治布局法	459
习题	401	19.5.1 分离集	459
参考文献	403	19.5.2 强分离集	461
第十八章 随机算法	409	19.5.3 通道生成	462
18.1 引言	409	*19.5.4 分治布局法	463
*18.1.1 概率论的基本知识	409	*19.6 VLSI 布局理论	466
18.1.2 随机算法的模型及其度量	412	19.6.1 平面图的分离定理	466
18.2 部分独立集	413	19.6.2 图的交叉点数	467
18.2.1 有向环图	414	19.6.3 布局下界定理	468
18.2.2 平面图	415	习题	468
18.3 三角形平面细图中点的位置	416	参考文献	469
18.3.1 细图层次	417	*第二十章 模型与下界	471
18.3.2 细图层次的构造算法	417	20.1 不同 PRAM 模型的相互模拟	471
*18.4 模式匹配	419	20.1.1 在 PRAM-EREW 上模拟 PPRAM- CRCW	471
18.4.1 指纹函数	420	20.1.2 在 CPRAM-CRCW 上模拟 PPRAM- CRCW	472
18.4.2 串匹配	423	20.1.3 在 APRAM-CRCW 上模拟 PPRAM- CRCW	474
18.4.3 二维数组的匹配	424	20.2 PRAM-CREW 的下界	475
18.5 多项式恒等的验证	425	20.2.1 理想的 PRAM 模型	475
18.5.1 基本技术	425	20.2.2 形式描述	475
18.5.2 矩阵乘积的验证	426	20.2.3 特定问题的下界	478
18.6 排序	427		
18.6.1 随机采样及随机快排序	427		
18.6.2 并行随机快排序算法	428		

20.3	PRAM-EREW 的下界	478
20.3.1	工具和方法	478
20.3.2	主要下界	479
20.4	PRAM-CRCW 的下界	480
20.4.1	PRAM-CRCW 与无界扇入电路	481
20.4.2	无界扇入电路的下界	487
20.5	P-完全导论	490
20.5.1	问题的可并行化	490
20.5.2	NC 类和 RNC 类	492
20.5.3	P-完全问题范例	496

20.5.4	小结	502
	习题	503
	参考文献	504
附录 A	复杂度表示及其符号	506
A.1	大-O 及其运算	506
A.2	大- Ω 和大- Θ	506
A.3	小-o 和小- ω	507
附录 B	算法复杂度一览表	508

第一章 导 论

本章首先介绍一下并行计算机体系结构中的若干问题及其分类; 然后讨论各种常用的并行计算模型; 最后介绍并行算法的一般概念. 其中处理器互连网络是讨论并行计算模型所需的最基本的硬件知识, 而并行计算模型是以后各章节所讨论的并行算法的共同基础. 在介绍并行算法一般概念时也示例性地给出了几个并行算法, 以期读者对并行算法的设计和分析有个初步的认识.

1.1 并行计算机的体系结构及分类

1.1.1 并行处理系统发展谱^{[1][2]}

计算机发展的总趋势是速度越来越快, 容量越来越大, 体系结构越来越完善, 软件越来越丰富, 从而处理能力越来越强. 计算机的发展历史表明, 为了达到更高的处理性能, 除了提高元器件的速度外, 系统结构也必须不断改进, 特别是前者的速度达到极限时, 后者将变成了问题的焦点. 计算机系统结构的改进, 主要围绕着增加同一时间间隔内的操作数量, 即所谓并行处理技术; 为并行处理所设计的计算机统称之为并行计算机; 在并行计算机上求解问题称之为并行计算; 在并行计算机上实现求解问题的算法可称之为并行算法.

并行处理是一种有效的强调开发计算过程中的并发事件 (concurrent events) 的信息处理方式. 并发性 (concurrency) 包含并行性 (parallelism)、同时性 (simultaneity) 和流水线 (pipelining). 并行事件可在同一时间间隔内发生在多个资源中; 同时事件可在同一时刻发生; 流水线事件可在部分重叠时间内出现. 开发问题中的并行性包括开发计算并行性, 搜索并行性和逻辑并行性. 其中计算并行性与算法的设计与分析有密切关系.

并行处理的发展主要是为满足某些含大量计算的应用领域的要求, 例如气象预报, 海洋学的研究, 空气动力学计算, 天体物理的计算模拟, 卫星图象处理, 核反应堆实验, 生物系统模拟, 社会经济模型构造, 国防尖端技术, 人工智能的研究等等. 为了达到上述高性能的要求, 于是就从单处理机开始沿着时间重叠, 资源重复和资源共享这三大计算机结构学派生出不同的多处理机系统. 图 1.1 示出了从单处理机 (uniprocessor) 和多计算机 (multicomputer) 两个极端向现代并行处理系统发展的过程. 在单处理机内部增设流水线处理部件, 多功能单元, 关联合存储器等只能开发指令级的并行性, 而为了进一步提高并行度, 必然导致向开发程序和任务级的并行性的多计算机系统发展. 把多台计算机连接起来, 相互配合, 各尽其能, 沿着功能专门化, 多机群和网络化这三种基本技术措施向异构型多处理系统, 同构型多处理系统和分布处理系统发展.

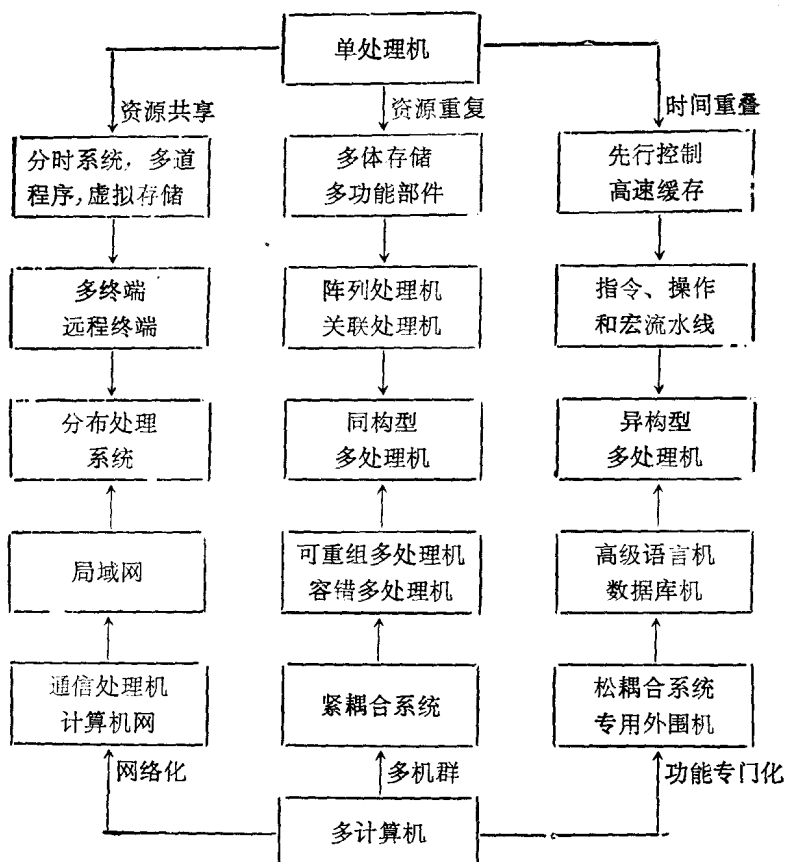


图 1.1 并行处理系统发展谱

1.1.2 并行计算机的分类

1. Flynn 分类法^[3]

1966 年美国的 M. J. Flynn 按照指令流和数据流将计算机系统分为四类。

- ① 单指令流单数据流 SISD (Single Instruction stream Single Data stream) 计算机。这就是传统的顺序处理机, 有时称单处理机。
- ② 单指令流多数据流 SIMD (Single Instruction stream Multiple Data stream) 计算机。如阵列处理机, 关联处理机和流水线处理机均属此类计算机。
- ③ 多指令流单数据流 MISD (Multiple Instruction stream Single Data stream) 计算机。实际上以何种机器为代表尚存疑议。
- ④ 多指令流多数据流 MIMD (Multiple Instruction stream Multiple Data stream) 计算机。如多处理机和多计算机均属此类计算机。

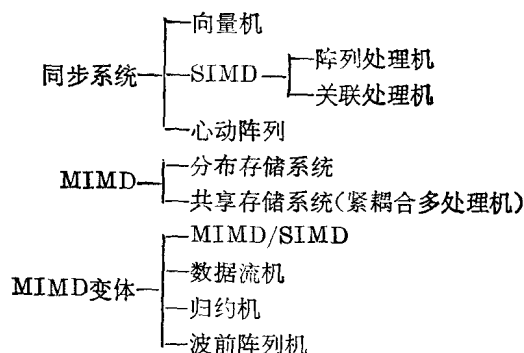
2. Händler 分类法^[4]

1977 年 Händler 根据计算机系统中流水线和并行度出现的级别, 将一台计算机表示为三对整数。令 POU 代表处理器控制单元, ALU 代表算术逻辑单元, BLC 代表位一级电路。于是一台计算机可表示如下:

$$T(O) = \langle K \times K', D \times D', W \times W' \rangle \quad (1.1)$$

其中, K = PCU 数目, K' = 能够执行流水线的 POU 数目, D = 每个 PCU 所控制的 ALU 数目, D' = 能够执行流水线的 ALU 数目, W = ALU 或处理单元 PE 的位数, W' = 所有 ALU 或单一 PE 中流水线段数。

3. 按体系结构分类^[5]



① **同步结构**: 同步并行计算机通过全局时钟、中央控制单元或向量单元控制器实现并发操作。其中, **流水线向量机**(pipelined vector processors)直接支持向量和矩阵等高速运算; **阵列处理机**(array processors)是近代大规模并行机(massively parallel computers)的前驱的主要机型, 直接支持大型科学计算; **关联处理机**(associative processors)是一种特殊的阵列机, 直接支持非数值计算方面的诸如按内容存取、施行逻辑比较等操作; **心动阵列**(systolic array)是一种专用结构的计算机, 采用同步流水线操作, 能直接支持信号处理等计算。

② **MIMD 结构**: 这种结构由各按局部数据独自执行指令的处理器所组成, 直接支持可大量自治求解的问题计算。其中, **分布存储系统**(distributed memory system)的各处理器节点的进程之间, 通过互连网络(如环、网孔、树、超立方等)用传递消息(passing message)的方式进行同步; **共享存储系统**(shared memory system)的处理器之间通过公共总线、交叉开关或多级互连网络, 使用一个均可访问的全局共享变量进行通信与同步, 这样的系统也叫做**紧耦合多处理机**(tightly coupled multiprocessor)。

③ **MIMD 变体结构**: 其中, **MIMD/SIMD** 是一种混合结构, 能直接支持图象处理 专家系统等方面的应用; **数据流结构**(data flow architecture)是一种数据驱动原理的新型计算机, 能有效地开拓大规模并行度; **归约结构**(reduction architecture)是一种需求驱动原理的新型计算机, 直接支持应用(函数)式程序设计语言; **波前阵列结构**(wavefront array architecture)是同步的心动结构与异步的数据流结构的结合, 在处理稀疏矩阵运算方面特别有效。

1.1.3 处理器互连网络

在并行机中如何将各个处理器或处理器与存储器互连起来是个关键问题。本节引入几种典型的在并行计算模型中经常使用的互连网络^[6]。

1. 一维线性连接

一维连接又称为线性阵列(linear array), 简记之为 LA. 其连接方式是并行机中最简单、最基本的互连方式. 其中每个处理器只与其左右近邻相连, 所以也叫作二近邻连接. 这种连接方式是心动结构的最基本形式. 当首尾处理器相连时可构成循环移位连接, 在拓扑结构上等同于环.

约定处理器的数目 $n=2^m$ (m 为正整数) 令其地址的二进制表示式为 $P=p_{m-1}p_{m-2}\cdots p_0$ (下同), 则线性阵列的连接函数 LO 可定义如下:

$$\left. \begin{aligned} LO_{-1}(P) &= P-1, 1 \leq P \leq n-1 \\ LO_{+1}(P) &= P+1, 0 \leq P \leq n-2 \end{aligned} \right\} \quad (1.2)$$

其中, LO_{-1} 和 LO_{+1} 分别表示左和右连接.

2. 网孔连接

在网孔连接(mesh-connected)中, 各处理器置于 q -维网格的格点上. 当 $q=2$ 时, 就是四近邻连接, 简记之为 MO^2 , 其中每个处理器只与其上下左右近邻相连, 连接函数 MO^2 可定义如下:

$$\left. \begin{aligned} MO^2_{-1}(P) &= P-1 \\ MO^2_{+1}(P) &= P+1 \\ MO^2_{-\sqrt{n}}(P) &= P-\sqrt{n} \\ MO^2_{+\sqrt{n}}(P) &= P+\sqrt{n} \end{aligned} \right\} \bmod n, 0 \leq P \leq n-1 \quad (1.3)$$

上述连接函数也可用置换轮换表示之:

$$\left. \begin{aligned} MO^2_{-1} &= (n-1, \dots, 1, 0) \\ MO^2_{+1} &= (0, 1, \dots, n-1) \\ MO^2_{-\sqrt{n}} &= \prod_{j=0}^{\sqrt{n}-1} (j+n-\sqrt{n}, \dots, j+2\sqrt{n}, j+\sqrt{n}, j) \\ MO^2_{+\sqrt{n}} &= \prod_{j=0}^{\sqrt{n}-1} (j, j+\sqrt{n}, j+2\sqrt{n}, \dots, j+n-\sqrt{n}) \end{aligned} \right\} \quad (1.4)$$

例 1.1 对于如图 1.2 所示的 $n=16$ 的二维网孔, $MO^2_{-1}=(15, 14, \dots, 1, 0)$, $MO^2_{+1}=(0, 1, \dots, 14, 15)$, $MO^2_{-\sqrt{n}}=(15, 11, 7, 3)(14, 10, 6, 2)(13, 9, 5, 1)(12, 8, 4, 0)$ 和 $MO^2_{+\sqrt{n}}=(0, 4, 8, 12)(1, 5, 9, 13)(2, 6, 10, 14)(3, 7, 11, 15)$. $\square$

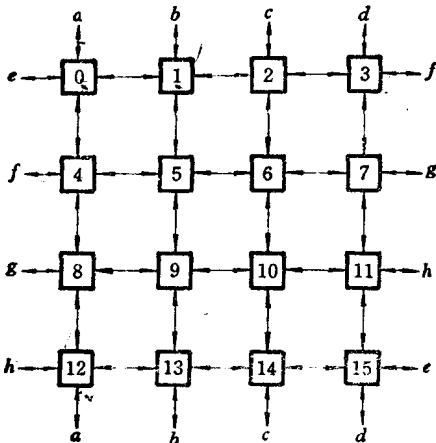


图 1.2 $n=16$ 的二维网孔

在 MO^2 模型上已设计出许多有效的并行算法, 但 MO^2 的通信功能较差, 在最坏情况下, 任意两处理器之间的信息交换至少需要 $\sqrt{n}-1$ 步^[1].

3. 树形连接

二叉树是大家非常熟悉的数据结构. 树连接(tree-connected), 简记为 TC, 除了根节点和叶节点外, 每个内节点都与其父节点和两个子节点相连, 因此二叉树连接可视为三近邻连接. 假定二叉树有 d 级(也称为 d 层, 编号自根至叶为 0 到 $d-1$), 则共

有 $n=2^d-1$ 个节点. 图 1.3 示出了 $d=4$ 的二叉树. 树的典型用法是, 根和叶节点具有 I/O 功能, 且叶节点执行并行运算, 而内节点仅负责叶节点间的通信, 最长通信路径为树高. 显然, 根易成通信瓶颈. 为此可使用 **X-树**, 使得同级的兄弟节点彼此相连^[7].

就通信功能而言, 树优于网孔. 两者的结合就产生了树网连接.

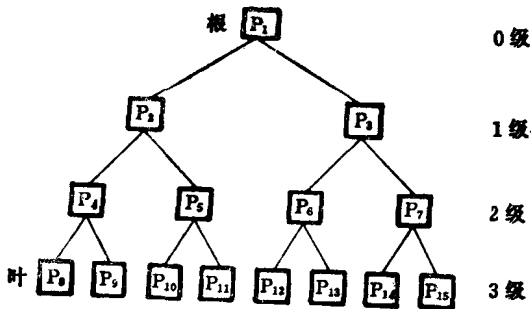


图 1.3 四级二叉树

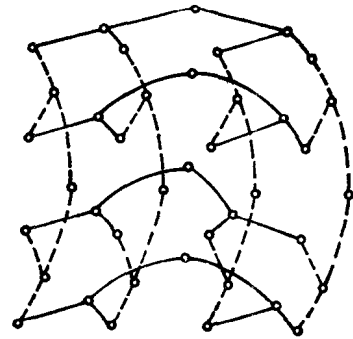


图 1.4 4x4 树网

4. 树网连接

树网 (mesh of tree), 简记之为 MT. 如果去掉原网孔中的水平和垂直连接, 而各行各列代之以二叉树就可构成此种结构. 如此处理, 如图 1.4 所示, 原网孔格点上的处理器就变成了二叉树的叶节点处理器, 它们通过相应的行树和列树相互通信. 数据的出入经过树根. 这种结构, 既具有良好的通信功能, 又能继承网孔上很多有效算法, 同时也非常适合于 VLSI 集成化, 所以在其上已经发展了很多并行算法.

5. 金字塔连接

金字塔 (pyramid) 结构也是树与网孔的结合, 它在图象处理中颇为有用. 一个规模为 p 的金字塔是一棵高度为 $\log_4 p$ 的完全四叉树, 其中, $p=k^2$ 是二维网孔形式的金字塔之底. 金字塔的层数自底向上从 0 开始递增编号, 顶部只有一个节点, 第 k 层的每个节点与九个其他节点相连, 它们分别是 $k-1$ 层的一个父节点, $k+1$ 层的四个子节点 (若存在的话) 和同层四个近邻节点 (若存在的话). 因此一个高为 $\log_4 p$ 的金字塔共有 $(4p-1)/3$ 个处理器节点. 图 1.5 是 $p=16$ 的金字塔.

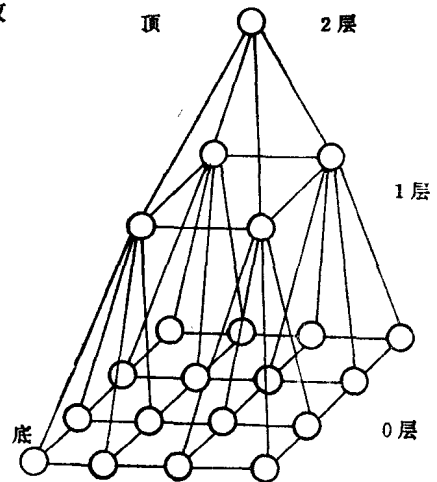


图 1.5 $p=16$ 的金字塔

6. 超立方连接

一个 $n=2^q$ 个节点的网络可以组织成一个 q -维超立方 (hypercube) 连接. 当 $q=3$ 时就是立方连接 (cube-connected), 简记之为 CC. 一个 $q=4$ 的超立方连接示于图 1.6.

在超立方连接中, 如果顶点采用合适的标记方法, 使得每维中处理器编号的二进制表示仅一位不同, 则其连接函数 CC 可定义如下:

$$CC_i(p_{m-1} \cdots p_{i+1} p_i p_{i-1} \cdots p_0) = p_{m-1} \cdots p_{i+1} \bar{p}_i p_{i-1} \cdots p_0 \quad (1.5)$$

其中 $0 \leq i < m$, $\bar{p}_i = 1 - p_i$.

OO_i 也可用置换轮换表示之:

$$OO_i = \prod_{j=0}^{n-1} (j, OO_i(j)), \quad 0 \leq i < m \quad (1.6)$$

(j 的第 i 位 = 0)

例 1.2 对于 $n=8$ 的立方连接, $OO_0 = (0, 1)(2, 3)(4, 5)(6, 7)$, $OO_1 = (0, 2)(1, 3)(4, 6)(5, 7)$ 和 $OO_2 = (0, 4)(1, 5)(2, 6)(3, 7)$. $\square$

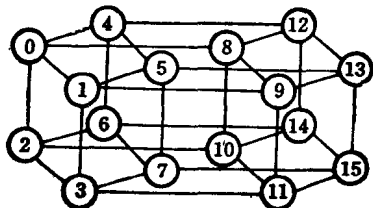


图 1.6 4-维超立方

立方连接是一种非常通用的互连结构。但由于 q -维超立方中每个节点将与 q 个近邻相连，当问题的规模增大时，将导致工程技术实现的困难，从而在某种意义上限制了超立方网络的应用。

7. 立方环连接

立方环(cube-connected-cycles), 简记之为 CCC, 是由 Preparata 等 1981 年提出的一种通用互连结构^[8]。它结合了环网与立方连接的优点及性质, 在立方网络的每个顶点代之以一个环, 故得名为带环的立方网络, 简称为立方环。在此网络中, 每个顶点的度 ≤ 3 , 从而与问题的规模无关。这一点对 VLSI 的实现极为重要。

CCC 的构造如下: 令 $n = 2^q$, 其中 $q = r + 2^r$, r 取满足 $r + 2^r \geq q$ 的最小整数。在环内: 编号为 j 的顶点与编号为 $(j+1) \bmod 2^r$ 的顶点相连, 其中, $0 \leq j < n$; 在环间: 编号为 $(l2^r + j)$ 的顶点与编号为 $(l2^r + j + (1 - 2l_j)2^j)$ 的顶点相连, 其中, $0 \leq l < 2^{q-r}$, $0 \leq j < 2^r$, 且 l_j 是 l 的二进制表示的第 j 位(最右为第 0 位)的值, $(1 - 2l_j)2^j$ 表示将 l 的二进制第 j 位取反。可见每个环是由 2^r 根连线连接而成的。每个环的下标为 l (共有 2^{q-r} 个环)。若将每个环上的顶点集视为一个超顶点, 则 CCC 就蜕化为一个 $q-r$ 维的立方网络了。图 1.7 示出了一个 $n=32$ ($q=5$, $r=2$) 的 CCC 网络。它共有 2^{5-2} 个环, 每个环内共有 $2^2=4$ 个顶点。

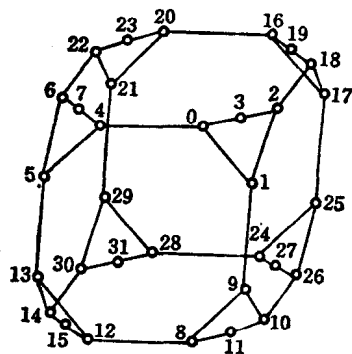


图 1.7 $n=32$ 的 CCC 结构

若每个顶点上的处理器的编号 i 的二进制表示有 q 位, 它可分解为一对整数 (l, p) , 其中, l 占 $q-r$ 位, p 占 r 位, 且满足 $l2^r + p = i$, $0 \leq i < n$ 。于是每个处理器有三个与其他处理器相连的连接函数 F (forward)、 B (backward) 和 L (lateral), 定义如下:

$$\left. \begin{array}{l} \text{环内 } F(l, p) \text{ 连向 } B(l, (p+1) \bmod 2^r) \\ \quad B(l, p) \text{ 连向 } F(l, (p-1) \bmod 2^r) \\ \text{环间: } L(l, p) \text{ 连向 } F(l + s2^r, p) \end{array} \right\} \quad (1.7)$$

其中, $s = 1 - 2l_r$.

Galil 和 Paul 研究表明, CCC 能够有效地模拟许多网络, 所以是一种非常通用的互连网络^[9].