《并行分布计算》复习提纲 （2022.12）

题型（分值）：

单选（20）、填空（10）、判断（10）、分析（25）、论述（20）、综合（15）

1. 并行计算机互联网络

（静态互联网络、动态互联网络、节点度、对剖宽度）（PC2 与 2.1）

分类：静态互连网络、动态互连网络、标准互连网络

静态互连网络：一维线性阵列、二维网孔、树、超立方、嵌入

动态互联网络：总线系统、多级互联网络、交叉开关

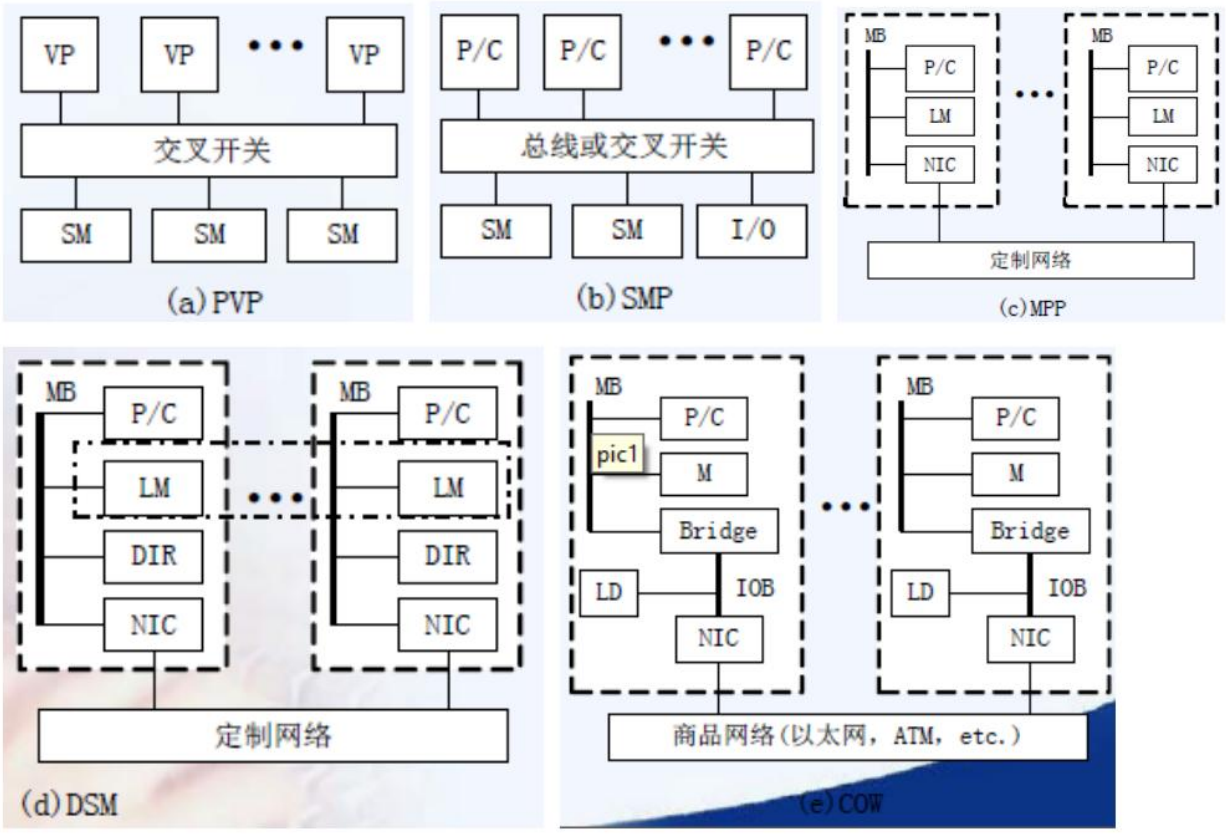
2. 并行计算机结构模型

（需要清楚 5+1 种模型，会画、清楚特点和发展趋势）（1.4.1）

单指令多数据流计算机(SIMD)：多为专用

多指令多数据流计算机(MIMD)：并行向量处理机(PVP)、对称多处理机(SMP)、大规模并行处理机(MPP)、工作站机群(COW)、分布共享存储多处理机(DSM)

结构模型：



特性结构一览表：

属性	PVP	SMP	MPP	DSM	COW
结构类型	MIMD	MIMD	MIMD	MIMD	MIMD
处理器类型	专用定制	商用	商用	商用	商用
互连网络	定制交叉开关	总线、交叉开关	定制网络	定制网络	商用网络
通信机制	共享变量	共享变量	消息传递	共享变量	消息传递
地址空间	单地址空间	单地址空间	多地址空间	单地址空间	多地址空间
系统存储器	集中共享	集中共享	分布非共享	分布共享	**分布非共享**
访存模型	UMA	UMA	NORMA	NUMA	NORMA

相互联系：

①DSM 是对 MPP 的改进和优化，实现了分布共享存储；

②SMP、MPP、DSM、COW 并行结构趋于一致；

③DSM 是 SMP 和 MPP 的自然结合；

④MPP 和 COW 的界限逐渐模糊，趋于一致，形成当代并行机的公用结构；

⑤并行计算机结构朝着高可扩放性（松散耦合）方向发展

公用结构：Shell 结构

Shell 结构包括一个专门设计和定制的电路将商品化处理器和高速缓存 C、主存 M、NIC 和磁盘 D 连接起来

优点：当处理器芯片更新换代时，只需要改变 Shell）

3.并行计算模型 （5.2）（前两种重要，后两种了解）

（一）PRAM 模型

基本概念：又称 SIMD-SM 模型，有一个集中的共享存储器和一个指令控制器，通过 SM 的 R/W 交换数据，隐式同步计算

分类、计算能力比较：见（6.PRAM 的分类）

优点：适合并行算法表示和复杂性分析，易于使用，隐藏了并行机的通讯、同步等细节

缺点：不适合 MIMD 并行机，忽略了 SM 的竞争、通讯延迟等因素

（二）异步 APRAM 模型

基本概念：又称分相 PRAM 或 MIMD-SM，每个处理器有其局部存储器、局部时钟、局部程序；无全局时钟，各处理器异步执行；处理器通过 SM 进行通讯；处理器间依赖关系，需在并行程序中显式地加入同步路障

指令类型：全局读、全局写、局部操作、同步

优缺点：易编程和分析算法的复杂度，但与现实相差较远，其上并行算法非常有限，也不适合 MIMDDM 模型

（三）BSP 模型

基本概念：“块”同步模型，是一种异步 MIMD-DM 模型，支持消息传递系统，块内异步并行，块间显式同步

（四）logP 模型

基本概念：是一种分布存储的、点到点通讯的多处理机模型，其中通讯由一组参数描述，实行隐式同步

优缺点：捕捉了 MPC 的通讯瓶颈，隐藏了并行机的网络拓扑、路由、协议，可以应用到共享存储、消息传递、数据并行的编程模型中；但难以进行算法描述、设计和分析

（注 1）BSP 模型与 logP 模型的对比：

BSP→LogP：BSP 块同步→BSP 子集同步→BSP 进程对同步＝LogP

BSP 可以常数因子模拟 LogP，LogP 可以对数因子模拟 BSP

BSP 提供了更方便的程设环境，LogP 更好地利用了机器资源

BSP 似乎更简单、方便和符合结构化编程

（注 2）四种计算模型综合比较

模型	PRAM	APRAM	BSM	LogP
体系结构	SIMD-SM	MIMD-SM	MIMD-DM	MIMD-DM
计算模式	同步计算	异步计算	异步计算	异步计算
同步方式	自动同步	路障同步	路障同步	隐式同步
通信方式	读/写共享变量	读/写共享变量	发送/接收消息	发送/接收消息

4.静态互联网络节点度、对剖宽度 （2.1）

节点度：射入或射出一个节点的边数（单向网络中，入射和出射边之和称为节点度）

网络直径：网络中任何两个节点之间的最长距离（即最大路径数）

对剖宽度：对分网络各半所必须移去的最少边数

如果从任一节点观看网络都一样，则称网络为对称的

静态网络特性一览表：

网络名称	网络规模	**节点度**	网络直径	**对剖宽度**
线性阵列	N 个节点	**2**	N-1	**1**
环形	N 个节点	**2**	⌊N/2⌋（双向）	**2**
2-D 网孔	$(\sqrt{N}\times\sqrt{N})$个节点	**4**	$2(\sqrt{N}-1)$	$\sqrt{N}$
Illiac 网孔	$(\sqrt{N}\times\sqrt{N})$个节点	**4**	$\sqrt{N}-1$	$2\sqrt{N}$
2-D 环绕	$(\sqrt{N}\times\sqrt{N})$个节点	**4**	$2\lfloor\sqrt{N}/2\rfloor$	$2\sqrt{N}$
二叉树	N 个节点	**3**	$2(\lceil\log_2 N\rceil-1)$	**1**
星形	N 个节点	**N-1**	2	**⌊N/2⌋**
超立方	$N=2^n$ 个节点	**n**	n	**N/2**
立方环	$N=k2^k$ 个节点	**3**	$2k-1+\lfloor k/2\rfloor$	**N/(2k)**

5.超线性加速 （4.2.4）

实际加速比相较于算出来的加速比更快，优于计算结果

例如：多个处理器在不同的分支方向上同时搜索，一旦有一个处理器找到了解，就结束其他处理器的搜索，从而产生超线性加速现象

6.PRAM 模型分类 （5.2.1）

分类： （C 允许、E 不允许）

（1）PRAM-CRCW 并发读并发写 （允许同时读同时写）

CPRAM-CRCW：仅允许写入相同数据 （公共的）

PPRAM-CRCW：仅允许优先级最高的处理器写入 （优先的）

APRAM-CRCW：允许任意处理器自由写入 （任意的）

（2）PRAM-CREW 并发读互斥写 （允许同时读不允许同时写）

（3）PRAM-EREW 互斥读互斥写 （不允许同时读也不允许同时写）

计算能力比较：

PRAM-CRCW 是最强的计算模型，PRAM-EREW 可 log p 倍模拟 PRAM-CREW 和 PRAM-CRCW

T EREW≥T CREW≥T CRCW

7.域分解、功能分解 （8.2）

（一）域分解（数据划分）

划分的对象是数据，可以是算法的输入数据、中间处理数据和输出数据；

将数据分解成大致相等的小数据片；

划分时考虑数据上的相应操作；

如果一个任务需要别的任务中的数据，则会产生任务间的通讯

（二）功能分解（计算划分）

划分的对象是计算，将计算划分为不同的任务，其出发点不同于域分解；

划分后，研究不同任务所需的数据，如果这些数据不相交的，则划分是成功的；

如果数据有相当的重叠，意味着要重新进行域分解和功能分解；

功能分解是一种更深层次的分解

8.带状划分、棋盘划分 （9.1）

（一）带状划分

将矩阵整行或整列地分成若干组，每组指派给一个处理器；也可将若干行或若干列指派给一个处理器

块带状划分： 每组的行和列是连续的

循环带状划分：每组的行和列是等距相间的

折中带状划分

（二）棋盘划分

将方阵划分成若干个子方阵，每个子方阵指派给一个处理器

块棋盘划分：行和列连续

循环棋盘划分：行和列等间距

折中棋盘划分

（二者相比，棋盘划分可开发出更高的并行度）

9.高速缓存一致性问题 （3.3）

简化版定义：如何保证同一单元在不同高速缓存中的备份数据一致

详细定义：当某一处理器第一次访问某一存储单元时，系统会将其内容的副本也同时传给与该处理器相连的高速缓存，以后当处理器再次访问此数据时，它就可以直接访问其高速缓存。如果另一个处理器也访问此同一个存储单元，则此数据的副本也将被传到与其相连的另一高速缓存。在此情况下，如果其中一个处理器改写了其高速缓存中的内容，但另一处高速缓存中的内容却仍是原来的值，这就造成了高速缓存中的数据不一致，即高速缓存一致性问题。

10.两种开关技术 （2.2.2）（PC2）

（一）存储转发选路（SF）

消息被分成基本的传输单位——信包，每个信包都含有寻径信息

当一个信包到达中间节点 A 时，A 把整个信包放入其通信缓冲器中，然后在选路算法的控制下选择

下一个相邻节点 B，当从 A 到 B 的通道空闲并且 B 的通信缓冲器可用时，把信包从 A 发向 B

(*)信包的传输时间：$t_{comm}(SF)=t_s+(mt_w+t_h)l=O(ml)$

缺点：每个结点必须对整个消息和信包进行缓冲，缓冲器较大；网络时延与发送信息所经历的节点数成正比

（二）切通选路（CT）

将信包进一步分成更小的片

在传递一个消息之前，就为它建立一条从源节点到目的节点的物理通道，在传递的全部过程中，线路的每一段都被占用，当消息的尾部经过网络后，整条物理链路才被废弃

(*)传输时间：$t_{comm}(CT)=ts+mt_w+lt_h=O(m+l)$

缺点：物理通道非共享；传输过程中物理通道一直被占用

（由此可见，存储转发网络的延迟与源和目的之间的距离成正比，而切通网络的延迟时间与源和目的之间的距离无关）

11.等效率函数曲线（4.3.2）

令 t_{ie} 和 t_{io} 分别是系统上第 i 个处理器的有用计算时间和额外开销时间（包括通信、同步和空闲等待时间等）

Tp 是 p 个处理器系统上并行算法的运行时间，对于任意 i 显然（假设）有 $Tp=t_{ie}+t_{io}$，且 Te+To=pTp

问题的规模 W 定义为最佳串行算法所完成的计算量，W=Te

$$S=\frac{T_e}{T_p}=\frac{T_e}{\frac{T_e+T_o}{p}}=\frac{p}{1+\frac{T_0}{T_e}}=\frac{p}{1+\frac{T_0}{W}} \qquad E=\frac{S}{p}=\frac{1}{1+\frac{T_0}{T_e}}=\frac{p}{1+\frac{T_0}{W}}$$

如果问题规模 W 保持不变，处理器数 p 增加，开销 To 增大，效率 E 下降。为了维持一定的效率（介于 0 与 1 之间），当处理数 p 增大时，需要相应地增大问题规模 W 的值。由此定义函数 $f_E(p)$为问题规模 W 随处理器数 p 变化的函数，为等效率函数

等效率曲线图：

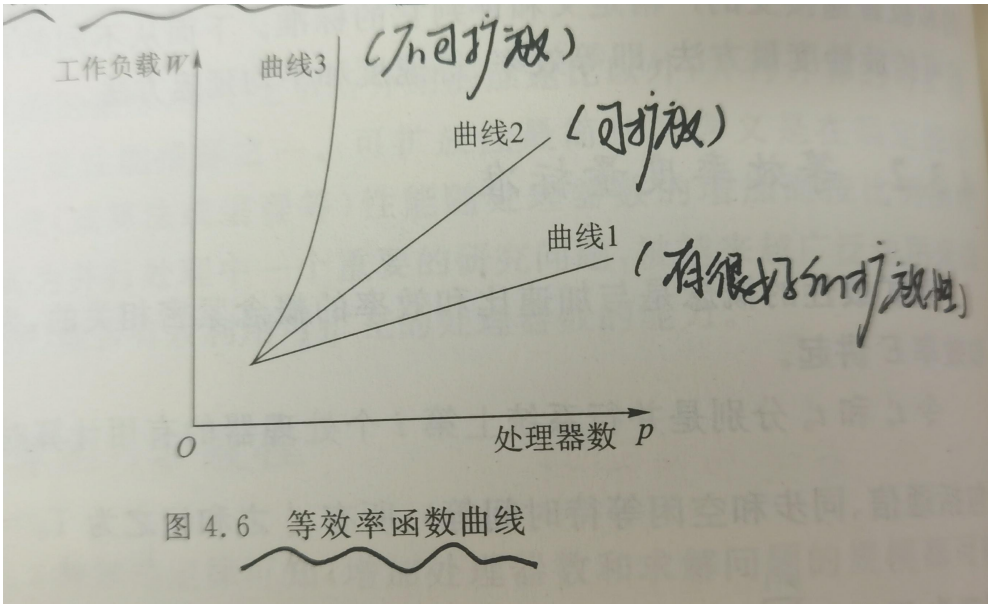


图 4.6 等效率函数曲线

曲线 1 表示算法具有很好的扩放性；曲线 2 表示算法是可扩放的；曲线 3 表示算法是不可扩放的。

优点：简单可定量计算的、少量的参数计算等效率函数

缺点：如果 To 难以计算出（在共享存储并行机中）

12.三种加速比性能定律 （4.2）

定义参数：

p：并行系统中处理器数

W：问题规模　　W_s：应用程序中的串行分量　　W_p：应用程序中可并行化部分

f：串行分量比例　　1-f：并行分量比例

$T_s=T_1$：串行执行时间　　T_p：并行执行时间

S：加速（比）　　E：效率

对应关系：

$W=W_s+W_p$　　$W_s=W_1$　　$f=W_s/W$　　$f+(1-f)=1$

定律一览表：

定律	Amdahl 定律	Gustafson 定律	Sun 和 Ni 定律
出发点	①固定不变的计算负载； ②固定的计算负载分布在多个处理器上的，增加处理器加快执行速度，从而达到了加速的目的	①对于很多大型计算，精度要求很高，即在此类应用中精度是个关键因素，而计算时间是固定不变的，此时为了提高精度，必须加大计算量，相应地亦必须增多处理器数才能维持时间不变； ②增多处理器必须相应增大问题规模才有实际意义	充分利用存储空间等计算资源，尽量增大问题规模以产生更好/更精确的解
适用条件	实时应用问题，任务量不变	任务量增加	从存储角度出发（存储量增加）
备注	当问题的计算负载或规模固定时，必须通过增加处理器数目来降低计算时间； 加速比受到算法中串行工作量的限制，实际上也受并行实现时的额外开销的限制	随着处理器数目的增加，串行执行部分 f 不再是并行算法的瓶颈	是 Amdahl 定律和 Gustafson 定律的推广； G(p)=1 时就是 Amdahl 加速定律； G(p)=p 变为 f+p(1-f)，就是 Gustafson 加速定律； G(p)>p 时，相应于计算机负载比存储要求增加得快，此时 Sun 和 Ni 加速均比 Amdahl 加速和 Gustafson 加速高

公式推导：

①Amdahl 定律

$$T_s + fW + (1-f)W \qquad T_p = fW + \frac{(1-f)W}{p}$$

$$S_p = \frac{W}{fW + \frac{(1-f)W}{p}} = \frac{p}{pf + 1 - f} = \frac{1}{\frac{(p-1)f+1}{p}} \xrightarrow[p\to\infty]{} \frac{1}{f}$$

②Gustafson 定律

$$S' = \frac{W_S + pW_P}{W_S + p \cdot W_P/p} = \frac{W_S + pW_P}{W_S + W_P}$$

$$S' = f + p(1-f) = p + f(1-p) = p - f(p-1)$$

③Sun 和 Ni 定律

设单机上的存储器容量为 M，其工作负载 W=fW+(1-f)W

当并行系统有 p 个结点时，存储容量扩大了 pM，用 G(p)表示系统的存储容量增加 p 倍时工作负载的增加量。则存储容量扩大后的工作负载为 W=fW+(1-f)G(p)W，所以存储受限的加速为：

$$S'' = \frac{fW + (1-f)G(p)W}{fW + (1-f)G(p)W/p} = \frac{f + (1-f)G(p)}{f + (1-f)G(p)/p}$$

13.并行计算对云计算、大数据、人工智能的支撑作用

核心支撑：算力（并行计算）+算法（并行计算）+数据（并行存储）

算力=并行体系结构（硬件）+并行程序

算法=并行分布式算法（思维）+并行编程

数据=分布式存储技术（数据组织）+存储系统

14.PCAM 设计方法学 （8.1）

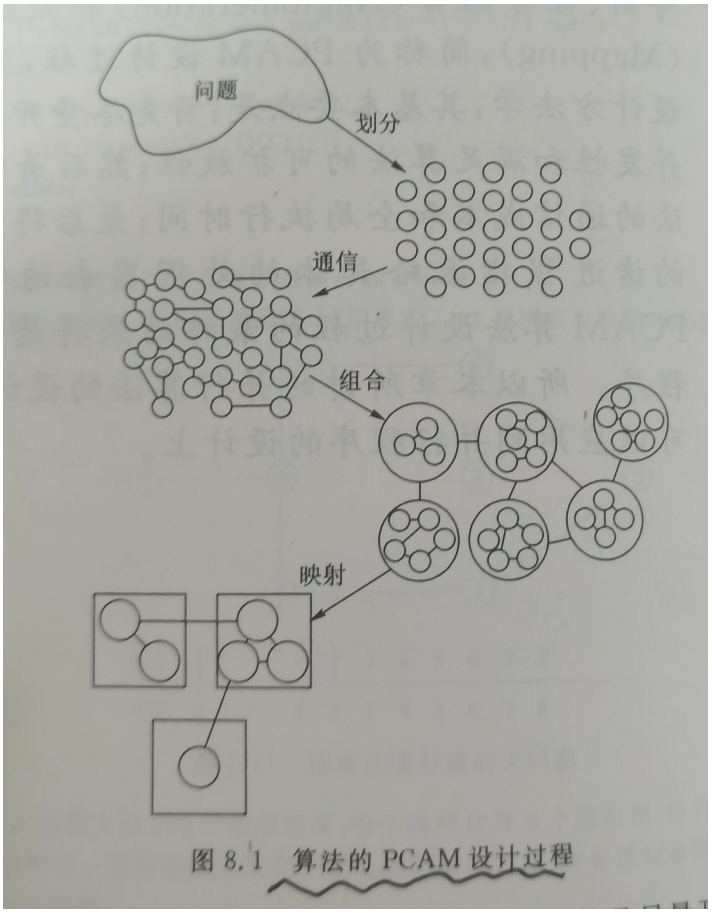


图 8.1 算法的 PCAM 设计过程

（1）**划分**：将整个计算分解成一些小的任务，其目的是尽量开拓并行执行的机会

（2）**通信**：确定诸任务执行中所需交换的数据和协调诸任务的执行，由此可见测上述划分的合理性

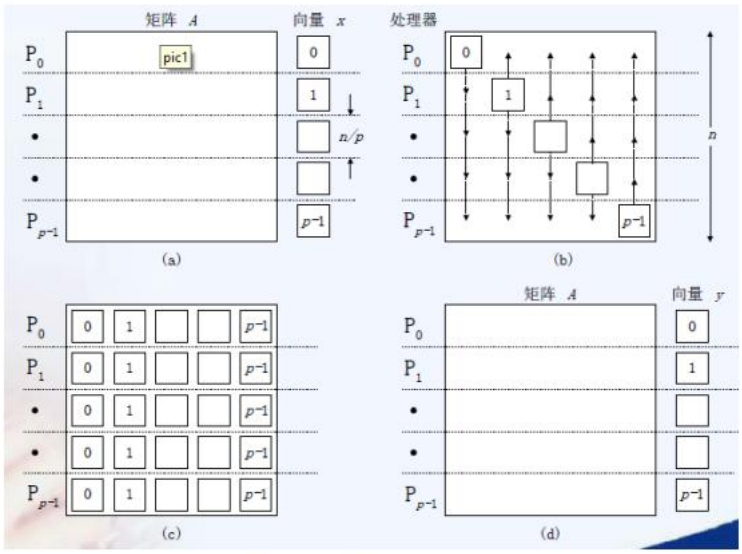
（3）**组合**：按性能要求和实现的代价来考察前两阶段的结果，必要时将一些小的任务组合成更大的任务以提高性能或减少通信开销

（4）**映射**：将每个任务分配到一个处理器上，其目的是最小化全局执行时间和通信成本以及最大化处理器的利用率

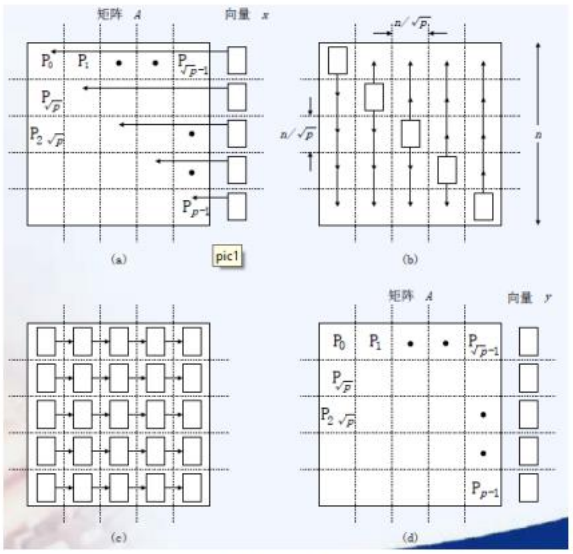
15. 矩阵和向量相乘的并行化（9.3）

带状划分：每个处理器分到矩阵一行/列的数据，计算开始前，每个处理器把自己存储的向量中的元素发给其他所有处理器，这样每个处理器就可以独立地完成一行的运算，最后合并计算结果

棋盘划分：每个处理器分到一个小矩阵的数据，计算开始前，处于对角线上的处理器把的向量中的元素发送给其他处理器，之后，每个处理器可以独自完成运算最后合并计算结果（向右发送）



带状划分



棋盘划分

脉动算法：局部串行，全局并行

16. 快速排序算法的并行化（6.1.2）

(*)对于长度为 n 的序列，在最坏情况下的划分的两个子序别为 n-1 及 1 的长度、相应的运行时间为 $t(n)=t(n-1)+\Theta(n)$，其解为 $t(n)=\Theta(n^2)$。理想的情况

是所划分的两个子序列等长，相应的运行时间为 $t(n)=2t(n/2)+\Theta(n)$，其解为 $t(n)=\Theta(nlogn)$。

(*)一种自然的并行化方法是并行地调用快排序对两个所划分的子序列进行快速排序。这种方法并不改变串行算法本身的属性，很容易改成并行形式。但是，如果用 n 个处理器排序 n 个数，若用一个处理器将$(A_1\cdots\cdots A_n)$划分成$(A_1\cdots A_s),(A_{s+1}\cdots\cdots A_n)$是不会得到成本最优算法的。

只有将划分步并行化，才有可能得到成本最优算法。

PRAM-CRCW 上快排序算法：

①构造一棵二叉排序树，其中主元是根；

②小于等于主元的元素处于左子树，大于主元的元素处于右子树；

③其左、右子树分别也为二叉排序树。

具体步骤：

①每个处理器分得一个元素，每个处理器将自己的元素编号并行地写入容量为 1 的全局变量，只有一 个能写入成功，将其作为主元；

②每个处理器将它的元素与主元进行大小比较，如果相等则退出，所有小于等于主元的元素并行地将自己的编号写入主元的左子节点，大于主元的元素同理，最终左子节点、右子节点均只有一个元素成功写入；

③未退出的处理器并行读取相应的处理元；

④重复步骤②直至所有处理机退出；

⑤中序遍历所得二叉排序树，得到最终结果。

构造二叉排序树的时间复杂度 O(log n)

17.几种 DSM 系统实现方案 （分布共享存储计算机系统）（3.3）

DSM 结构特性：

共享存储系统采用分布共享，减少集中共享的冲突；

采用高速缓存来缓和由共享引起的冲突和分布存储引起的长延迟；

保持了共享编程的方便性和软件的可移植性

DSM 系统分类：

①硬件实现的共享存储：CC-NUMA、NCC-NUMA、COMA

②软件实现的共享存储：共享虚拟存储(SVM)

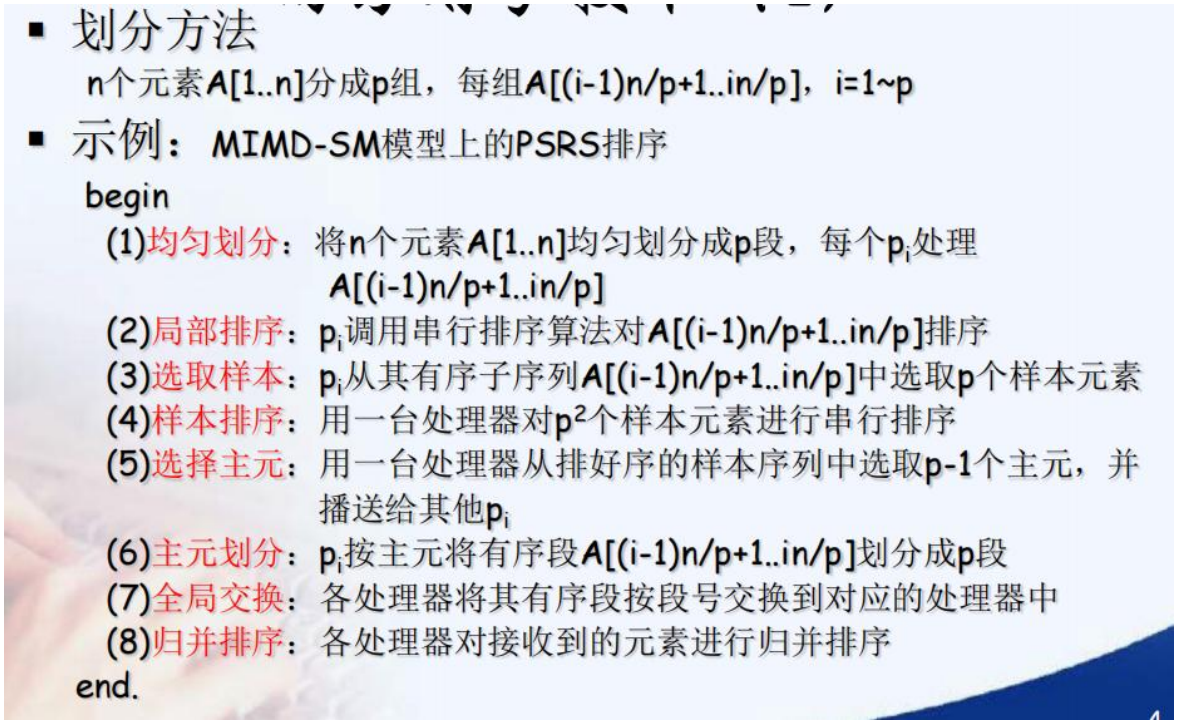
DSM 系统特性一览表：

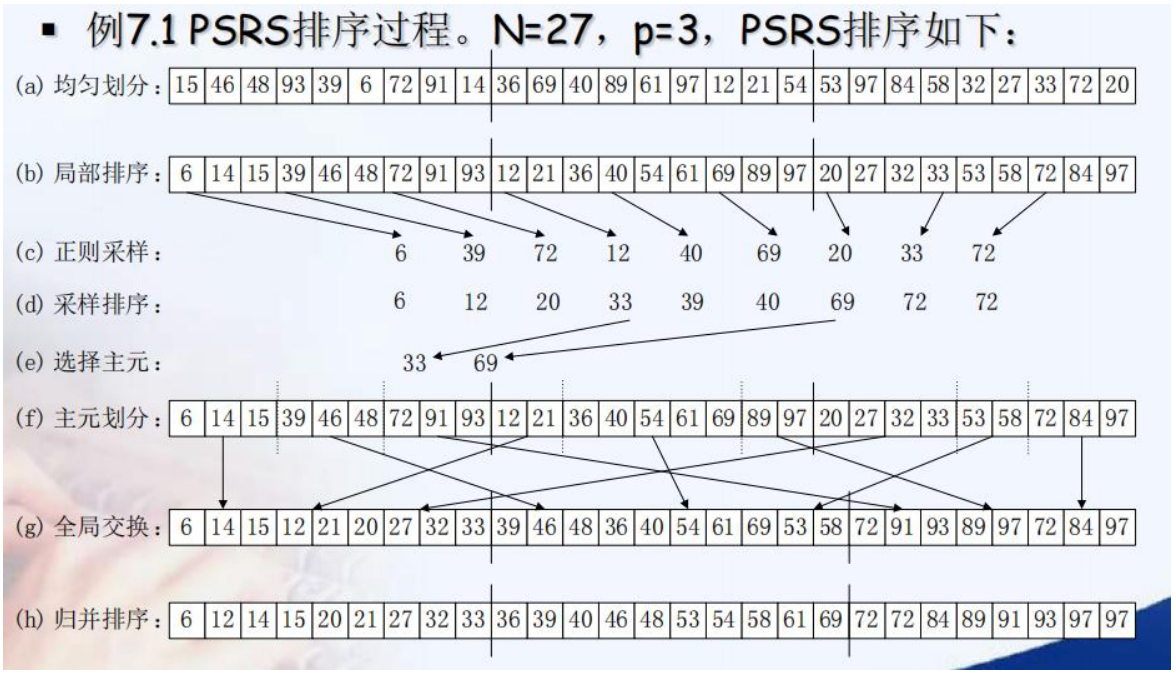
DSM 系统	CC-NUMA	NCC-NUMA	COMA	SVM
全称	高速缓存一致性非均匀存储结构系统	非高速缓存一致性非均匀存储结构系统	全高速缓存存储结构系统	共享虚拟存储系统
特点	共享存储器分布于各节点之中，节点之间通过可扩放性好的互连网络相连，每个处理器都能缓存共享单元	虽然每个处理器都有高速缓存，但硬件不负责维护高速缓存一致性	共享存储器的地址是活动的，存储单元与物理地址分离，数据可以根据访存模式动态地在各节点的存储器间移动和复制，每个节点的存储器相当于一个大容量高速缓存	用软件的方法把分布于各节点的多个独立编址的存储器组织成一个统一编址的共享存储空间
维持高速缓存一致性的方法	通常采用基于目录的方法来维持	由编译器或程序员来维护	由在每个节点存储器维护	/
优点	/	/	在本地共享存储器命中的概率较高	在消息传递的系统上实现共享存储的界面
缺点	/	/	当存储器的访问不在本节点命中时，由于存储器的地址是活动的，需要一种机制来查找被访问单元的当前位置，因此延迟很大	难以获得满意的性能

18. 划分设计技术 （7.1）（PC7）

（一）均匀划分技术（最基本的划分方法）

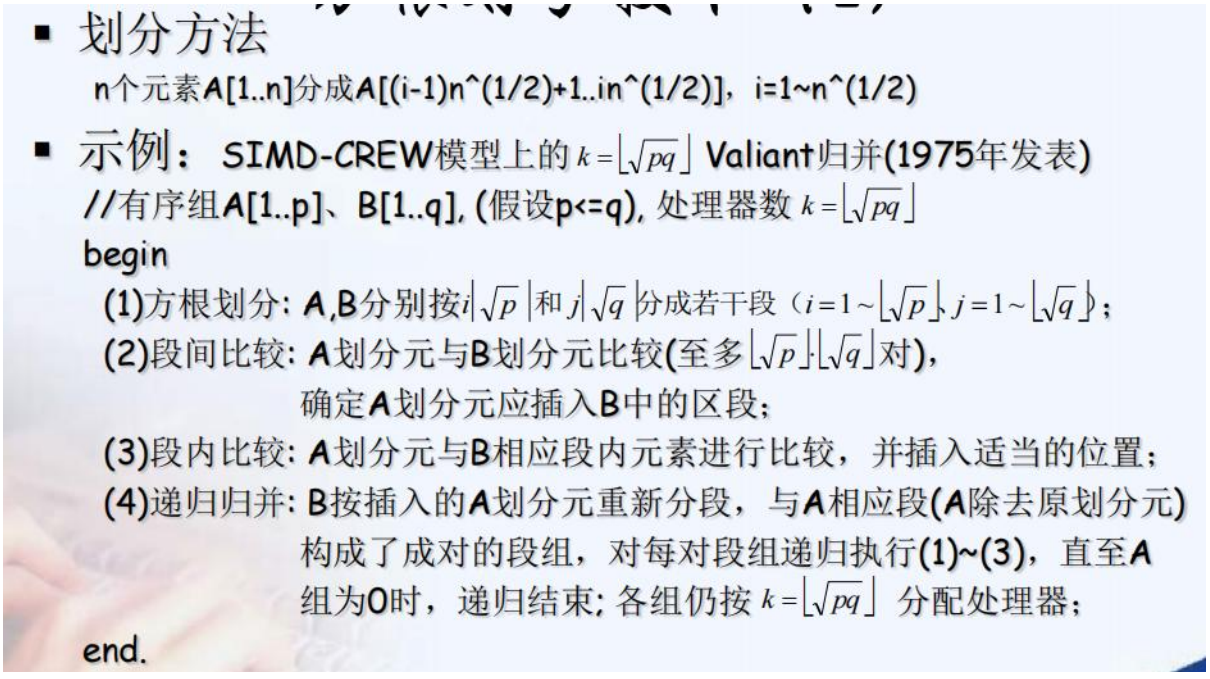
将 n 个元素分割成 p 段，每段含有 n/p 个元素且分配给一台处理器

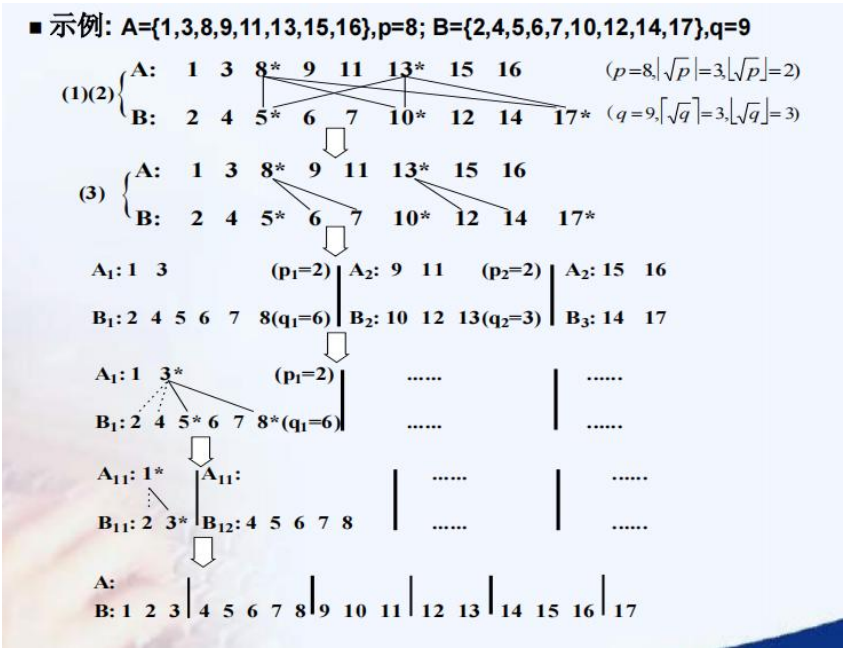




（二）方根划分技术

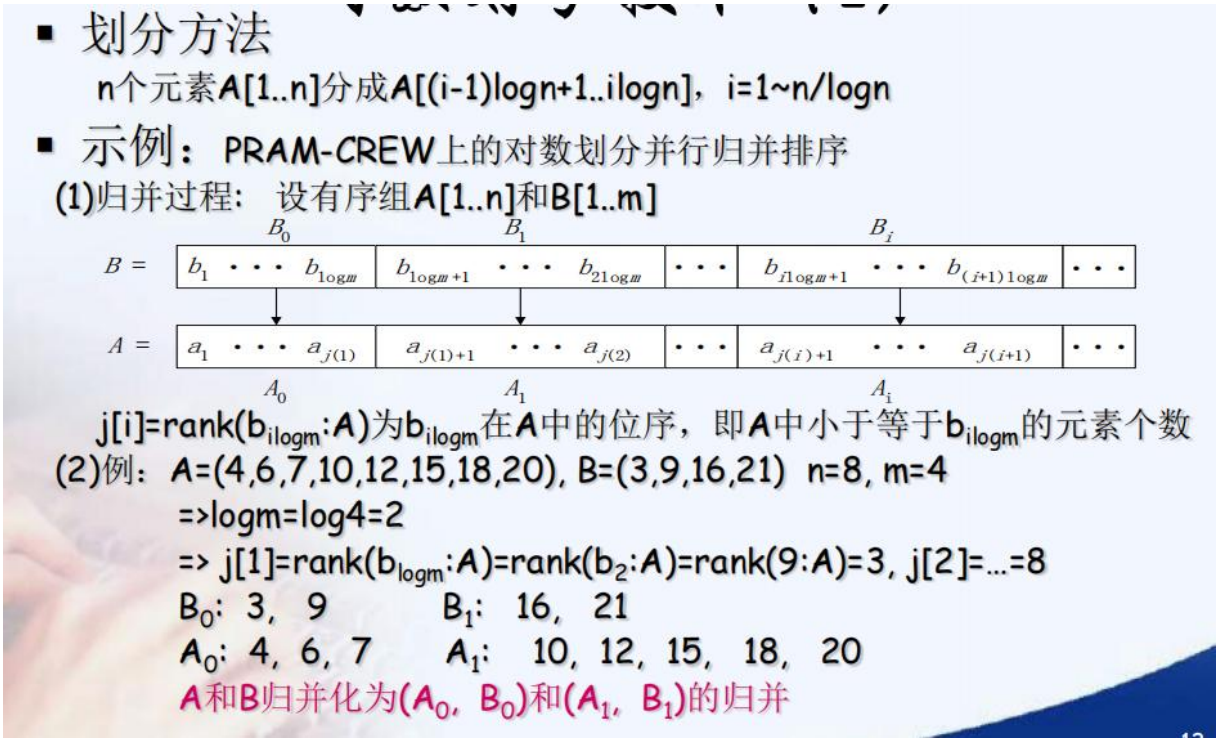
取每第 i √n（i=1，2，…）个元素作为划分元素而将序列划分成若干段，然后分段处理之。





（三）对数划分技术

取每第 ilog n（i=1，2，…）个元素作为划分元素，而将序列划分成若干段，然后分段处理之。



（四）功能划分技术

将长为 n 的序列化分成等长的一些组，每组中的元素数应大于或等于 m（最后一组除外），然后各组可并行处理

- 划分方法

  n个元素A[1..n]分成等长的p组，每组满足某种特性。

- 示例：(m, n)选择问题(求出n个元素中前m个最小者)
  - 功能划分：要求每组元素个数必须大于m；
  - 算法：p194算法7.4

    输入：$A=(a_1,\ldots,a_n)$；输出：前m个最小者；

    Begin

    (1) 功能划分：将A划分成g=n/m组，每组含m个元素；

    (2) 局部排序：使用Batcher排序网络将各组并行进行排序；

    (3) 两两比较：将所排序的各组两两进行比较，从而形成MIN序列；

    (4) 排序-比较：对各个MIN序列，重复执行第(2)和第(3)步，直至选出m个最小者。

    End

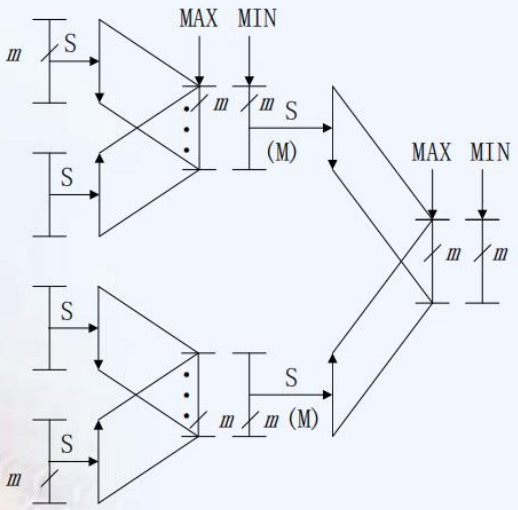


图6.3 (m-n)-选择过程