# 高性能与云计算复习提纲

## 第一讲、引言

1. 什么是高性能计算？

研究如何分解一个巨大规模的问题，并分配给多个计算机进行处理，并把这些计算结果综合起来得到最终的结果。

1. 高性能计算的等价词

并行计算、高端计算、超级计算。

1. 多核的运算速度一定会比单核的CPU快吗？

不一定、对底层体系结构不了解的话，无法充分利用硬件性能；要想发挥多核功能，设计的软件首先要能做并行计算。

1. 大规模数据处理面临的困难？

大规模PC集群可靠性很差；并行/分布式程序开发、调试困难。

1. 大数据时代的高性能计算——云计算

它除了提供大规模分布式计算外，还以组织和管理数据为核心工作之一，它获取并且维护持续变化的数据集，提供存储以及方便操作数据的编程模式。

1. 高性能计算和云计算的基础

计算：数据处理的能力。CPU主频，CPU核数。浮点计算能力Flops/s。

存储：数据存储的能力。缓存，内存，硬盘，磁带等。每秒读写的字节数Mbytes/s。

通信：数据通信的能力。内部网络，局域和广域网络。每秒传输的比特数Bits/s。

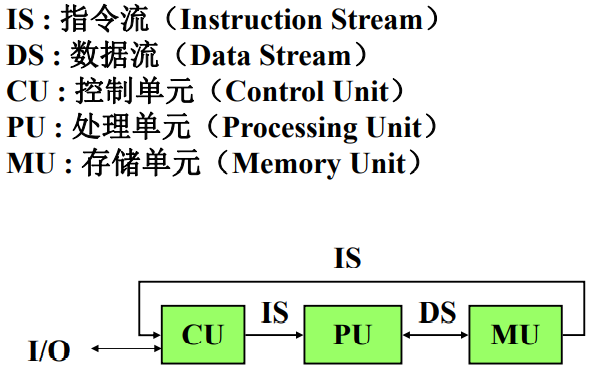
1. 高性能计算机

目前任何高性能计算和超级计算都离不开使用并行技术，所以高性能计算机肯定是并行计算机。

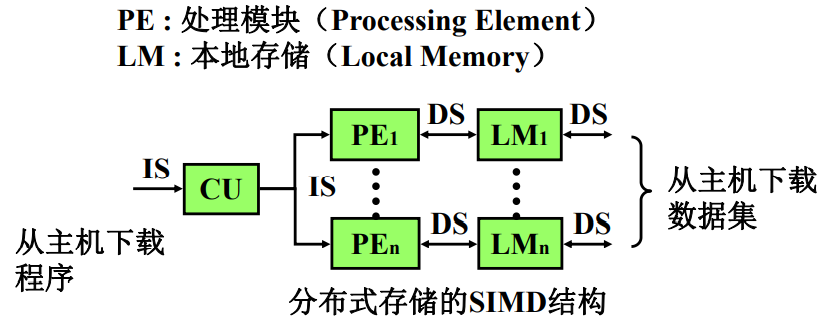
1. Flynn分类（计算机系统结构的分类方法之一）

基于指令和数据流：单指令单数据流SISD，单指令多数据流SIMD，多指令单数据流MISD，多指令多数据流MIMD。

SISD：通用的串行机：



SIMD：矢量机，专用计算机：

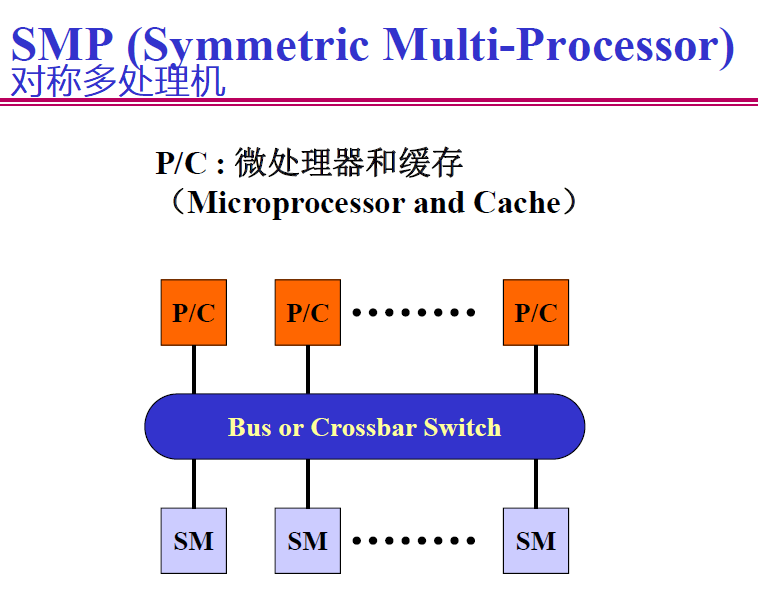
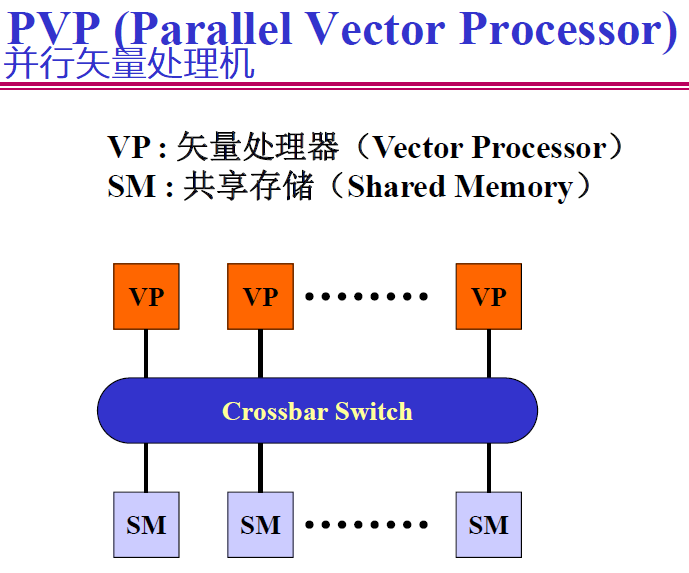
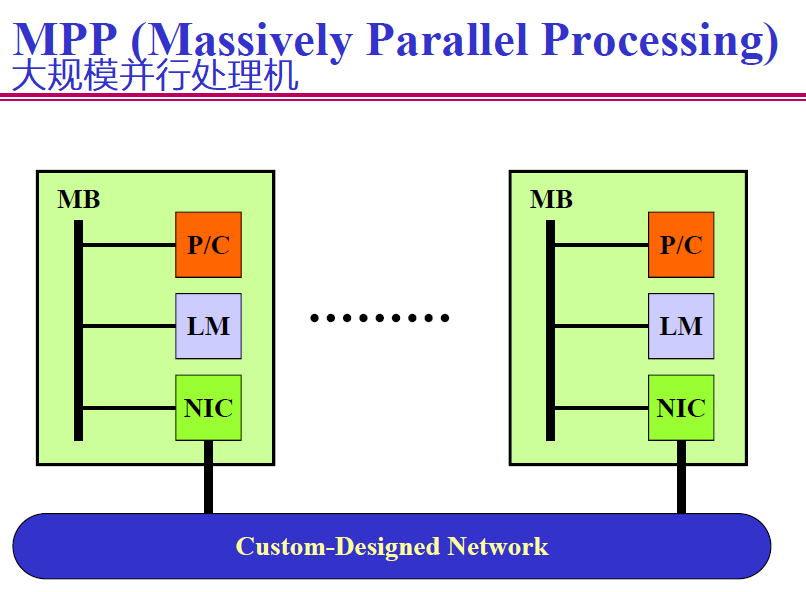
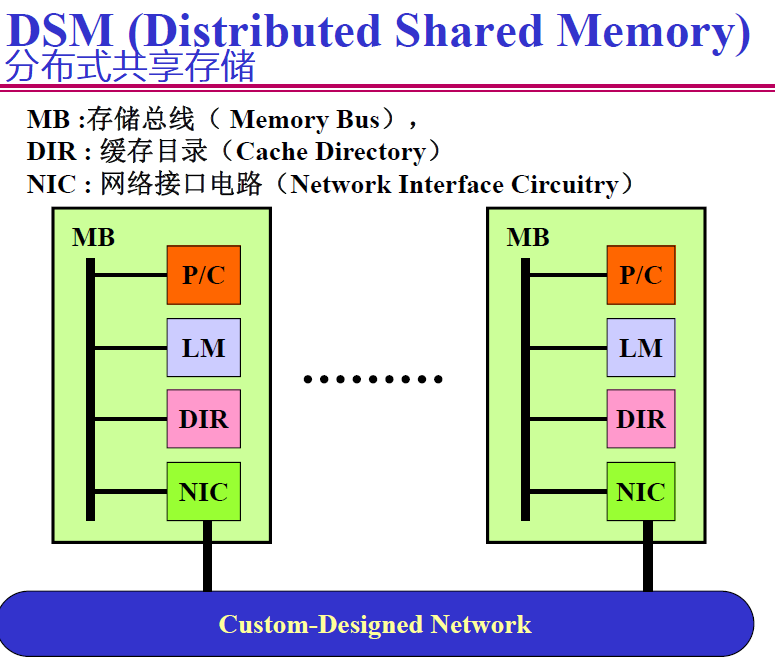
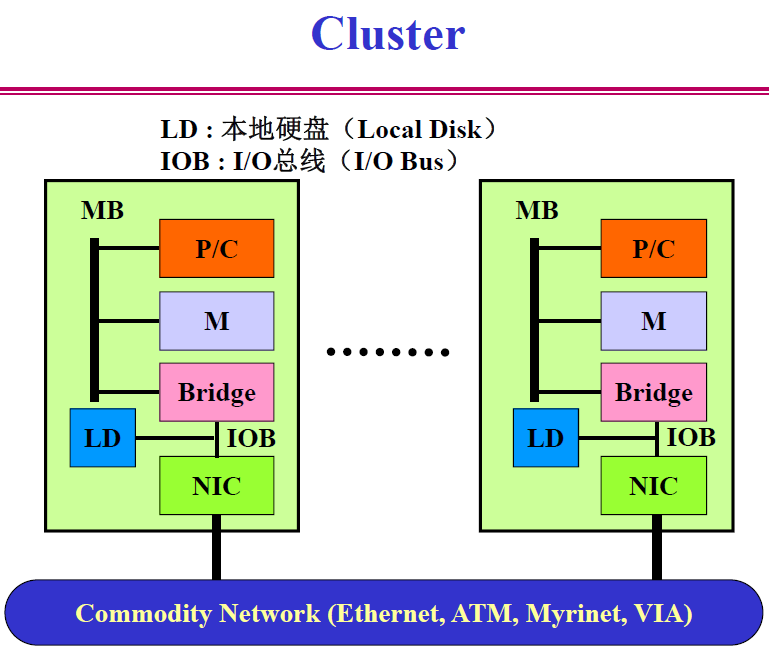


MISD：处理器阵列、脉动式阵列；专用计算机。

MIMD：通用的并行计算机。

1. **并行计算机体系结构**

大部分并行计算机都是MIMD。

****另外还有：**集群（Cluster，工作站集群（COW，Cluster Of Workstation,简称Cluster）），**分布式系统。

可分为两种架构：多处理器架构/共享存储架构；多计算机架构/分布存储架构。

## 第二讲、并行计算机体系结构

1. 创建和使用并行计算机的主要原因

并行计算机是解决单处理器速度瓶颈的最好办法之一。

1. 并行计算机的两个最重要的组成部分是计算节点和节点间的通信与协作机制。
2. 并行计算体系结构的发展主要体现在计算节点性能的提高以及节点间通信技术的改进两方面。
3. 复习：基本网络部件

链路（link）：传输信息的物理介质。

交换机/路由器开关（switch/router）：用于建立变换网络。

网络接口电路（NIC）：用来连接主机和网络。

1. 通信时延：从源节点到目的节点传输一条消息所需的总时间。

在网络两端相应收发消息的软件开销。

由于通道占用导致的通道时延，即总的消息长度除以通道带宽。

沿选路路径做一系列选路决策花费在后续交换开关上的选路时延。

由于网络传输竞争导致的竞争时延。

1. 带宽

端口带宽：从任意端口到另外端口单位时间内传输消息的最大位数。

聚集带宽：从一半节点到另一半节点，单位时间内传输消息的最大位数。

链路带宽：单位时间内链路传输消息的最大位数。

对剖宽度：将网络分成两个相等部分所必须移去的最少边数。

对剖带宽：每秒钟内，在最小的对剖平面上通过所有连线的最大信息位数。等于对剖宽度与链路带宽之积。

1. 基本定义

节点度：射入或射出一个节点的边数。

网络直径，网络中任何两个节点之间的最长距离。

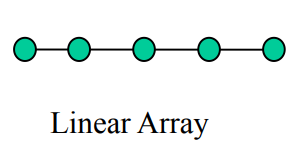
对剖宽度：对分网络各半所必须移去的最少边数。

1. 静态互连网络

处理单元间有着固定连接的一类网络，在程序执行期间，这种点到点的连接保持不变。

一维线性阵列：

并行机中最简单最基本的互连方式。每个节点只与其左右近邻相连，也叫二近邻连接。

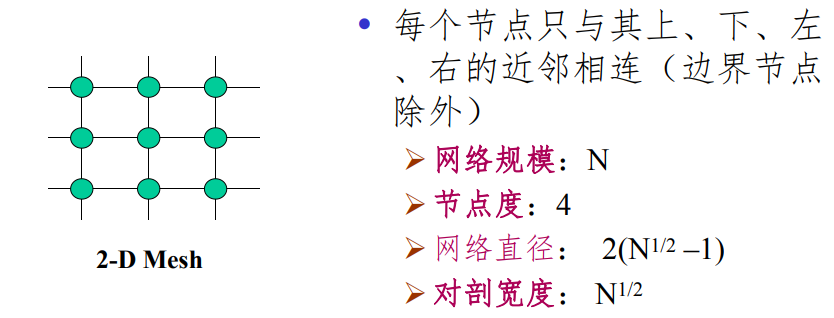
N个节点用N-1条边串接之，

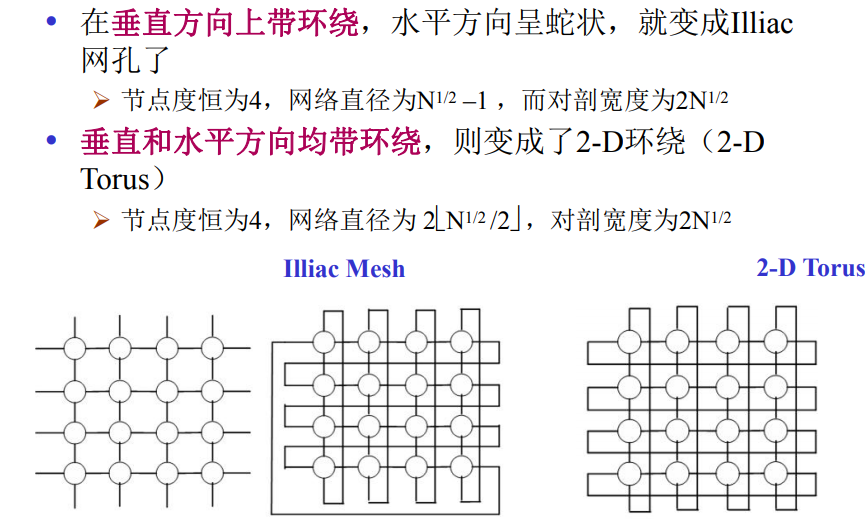
内节点度：2，直径：N-1，对剖宽度：1。

当首尾节点相连时可构成循环移位器，在拓扑结构上等同于环，环可以是单向的或双向的。

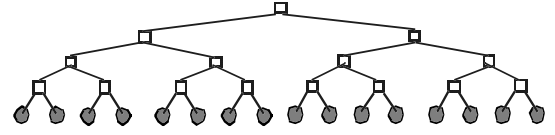
节点度恒为2，直径或为N/2下取整（双向），或为N-1（单向），对剖宽度：2。

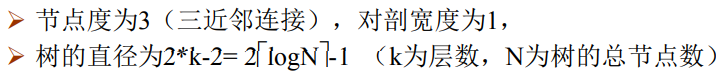
二维网孔：





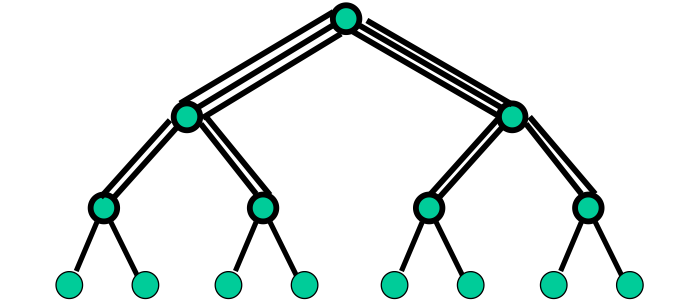
二叉树：





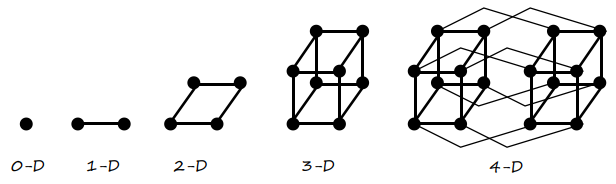
胖树：

传统二叉树的主要问题是根容易成为通信瓶颈。胖树节点间的通路自叶向根逐渐变宽。对剖宽度随着N的增大而增大。

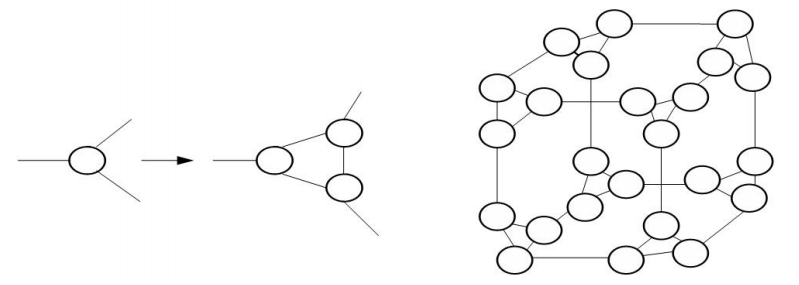


超立方：

一个n-立方由N=2的n次方个顶点组成。节点度为n，网络直径为n，对剖宽度N/2。



3-立方环：



静态互连网络特性比较：



1. 动态互连网络

用交换开关构成的，可按应用程序的要求动态地改变连接组态。

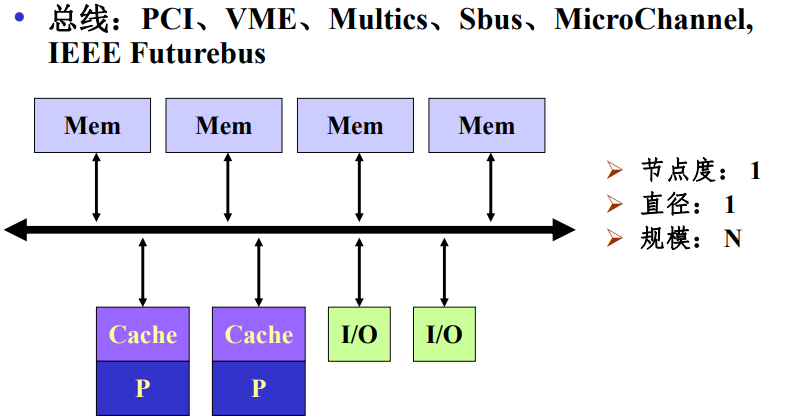
通信模式是基于程序的要求；连接是在程序执行过程中实时建立；基于总线或开关。

总线的特点：

总线的优点在于成本低，不随处理器数目的增加而增加。

总线的缺点在于扩展性不好，总线的带宽固定，随着处理器数的增加，每个处理器带宽减少。

可利用程序中的局部性原理减少对总线带宽的需求。

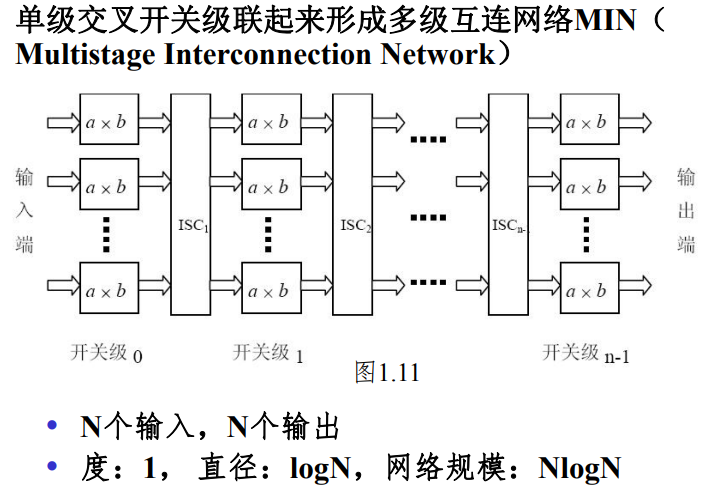


交叉开关的特点：

交叉开关具有良好的带宽特性。

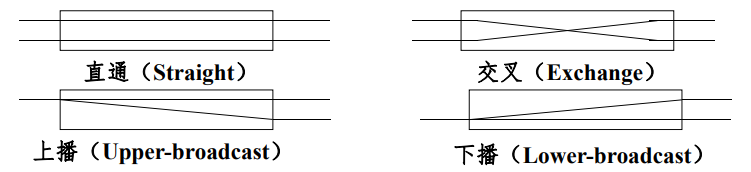
非阻塞通信：两个节点之间的通信，不会阻塞其他节点之间的通信。

代价不可扩放。



交换开关：

一个交换开关模块有n个输入和n个输出，每个输入可连接到任意输出端口，但只允许一对一或一对多的映射，不允许多对一的映射，否则将发生冲突。





n:节点规模 w：数据宽度 f：时钟频率

1. **标准互连网络**

开放、高带宽、低延迟、高可靠、扩展性好的交换网络和技术；

主要应用于集群系统。

以太网、万兆以太网、InfiniBand(IB)。

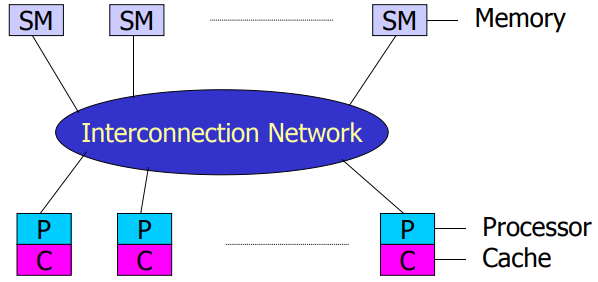
1. **访存模型**

共享存储：单一地址空间：存储模块定义了一个可在处理期间共享的单一地址空间；任何处理器可以通过互联网存取任何存储模块。

分布式存储：处理器单元独立工作，每个处理器有自己本地存储；通过消息传递来交互；处理器单元不能直接存取其他处理器单元的内存，必须通过消息传递来交换处理器之间的数据。

1. **存储器结构分类**

集中式存储器：UMA



UMA模型是均匀存储访问模型的简称。其特点是：

物理存储器被所有处理器均匀共享

所有处理器访问任何存储字取相同的时间

每台处理器可带私有缓存

外围设备也可以一定形式共享

分布式存储器：NUMA



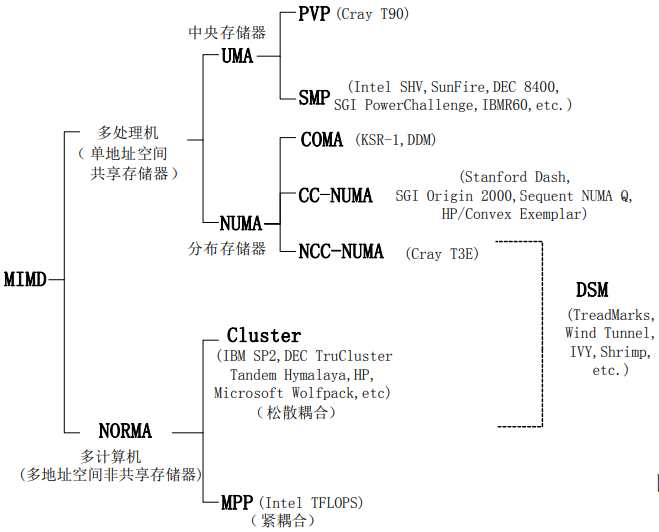
NUMA模型是非均匀存储访问模型的简称。特点：

被共享的存储器在物理上是分布在所有的处理器中，其所有本地存储器的集合就组成了全局地址空间

处理器访问的存储器的时间是不一样的；访问本地存储器LM较快，而访问外地的存储器或全局共享存储器GSM较慢

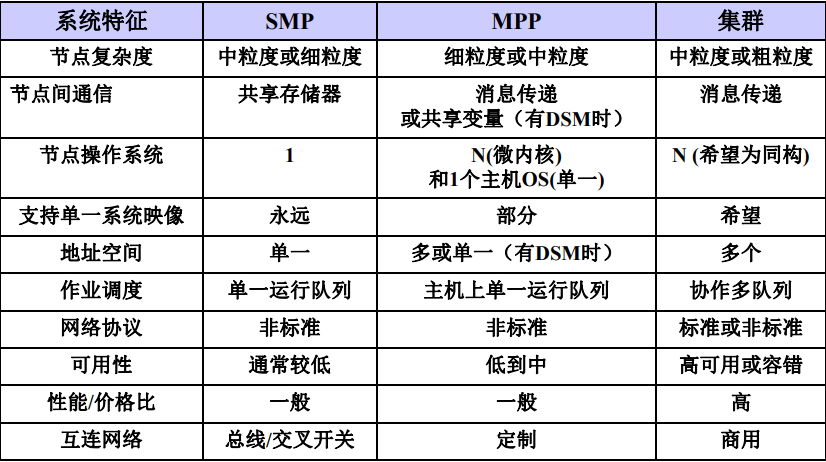
每台处理器可带私有告诉缓存，外设也可以某种形式共享

构筑并行机系统的不同存储结构



## 第三讲、并行计算机系统及性能评测

1. **SMP、MPP、集群的比较**



1. **什么是性能评测？**

性能评测：性能评价和性能分析，目的是为了提高性能。

1. **如何进行并行计算性能评测？**

机器级的性能评测：CPU和存储器的某些基本性能指标；并行和通信开销分析；并行机的可用性与好用性以及机器成本、价格与性价比。

算法级的性能评测：加速比、效率、扩展性。

程序级的性能评测：基准程序。

并行机的性能指标



1. 算法级性能评测

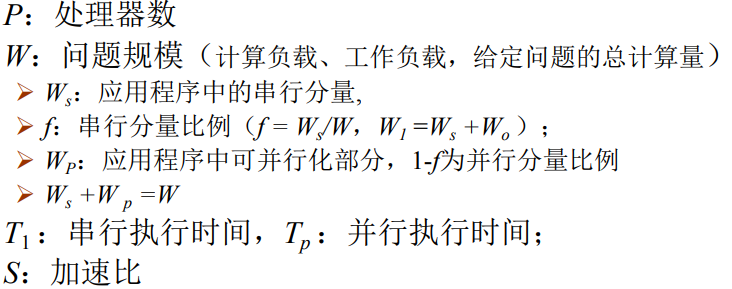
加速比：对于一个给定的应用，并行算法相对于串行算法的性能提高程度。

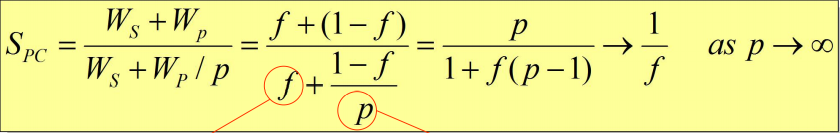
可扩展性：当系统和问题规模增大时，可维持相同性能的能力，即指应用、算法和结构能否充分利用不断增长的处理器的能力。

1. 加速比

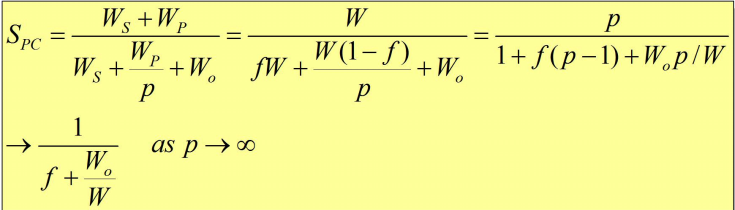
固定问题规模、固定时间、固定存储

1. **Amdahl定律，固定不变的计算负载**

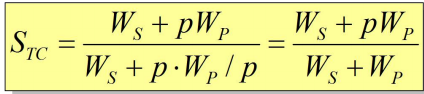




增强的Amdahl定律，考虑并行和通信的开销w0

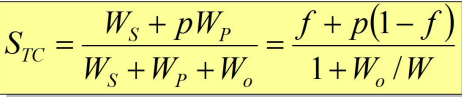


1. Gustafson定律（时间固定）





考虑并行开销：



1. Sun&Ni定律（存储固定）

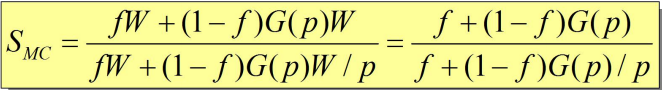
只要存储空间许可，应尽量增大问题规模以产生更好和更精确的解。

假定在单节点上使用了全部存储容量M并在相应于W的时间内求解，此时工作负载：

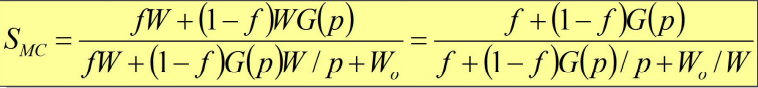


在p个节点的并行系统上，能能够求解较大规模的问题是因为存储容量可增加到pM。令因子**G(p)表示存储容量增加到p倍时并行工作负载的增加量**，所以扩大后的工作负载：

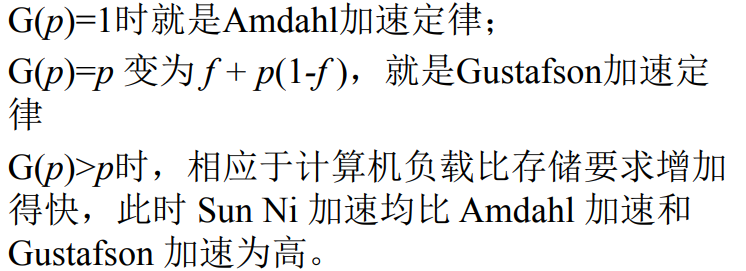




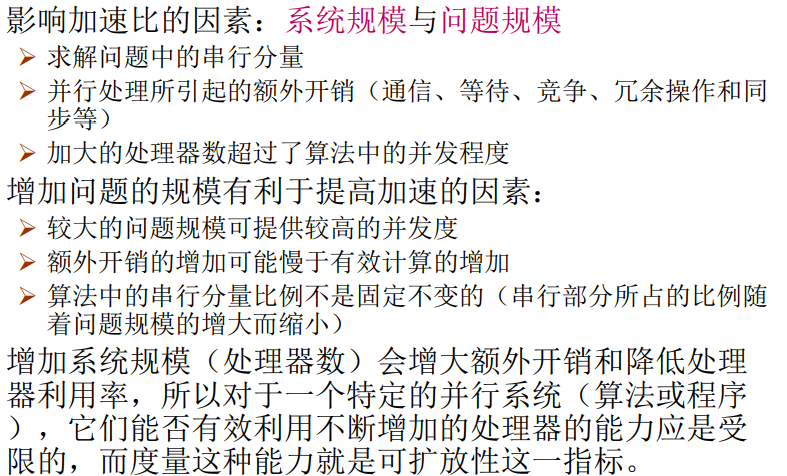
考虑并行开销：



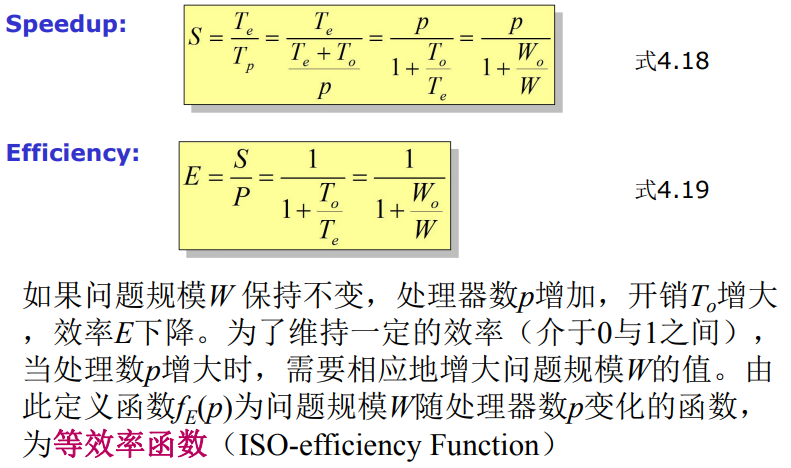
Sun&Ni定律特性：

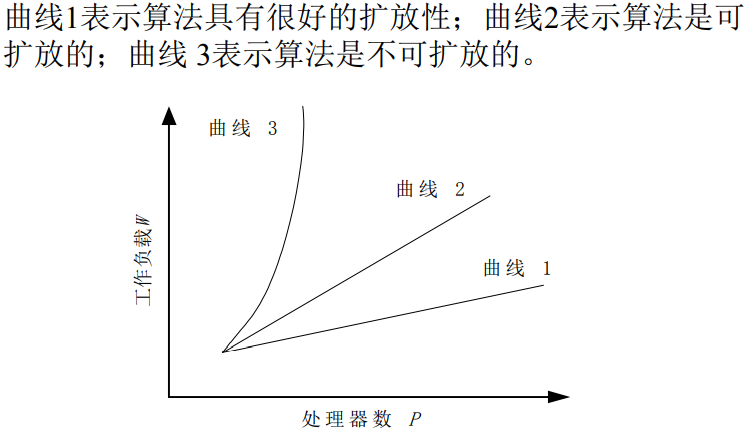


1. 可扩展性评测标准



等效率函数：



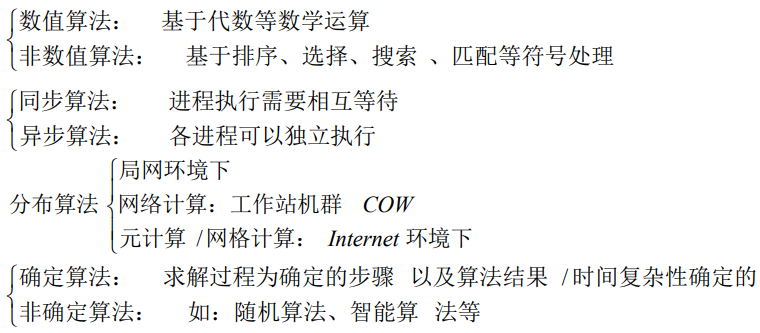


## 第四讲、并行算法设计-上

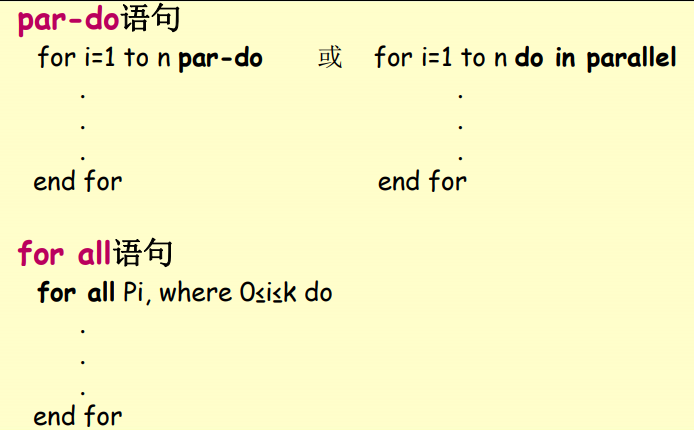
1. 并行算法

一组可同时执行且可互相协作的诸进程的集合。

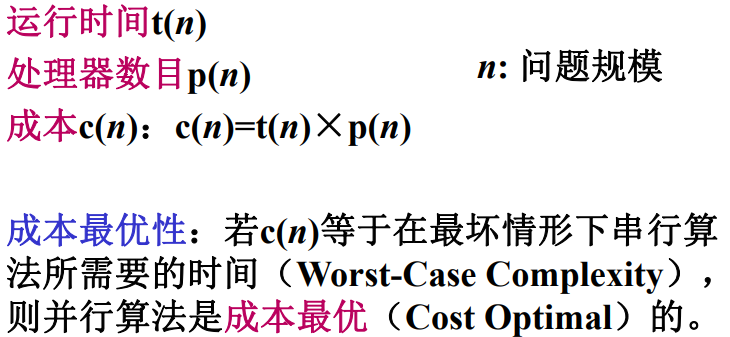
1. 分类

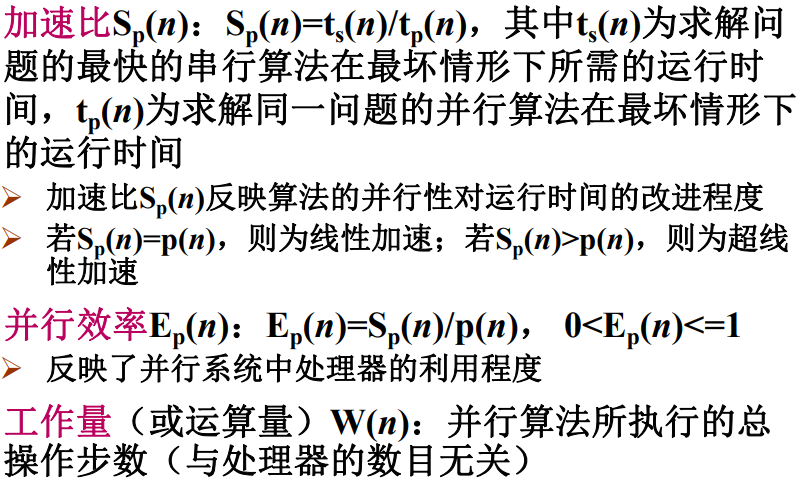


1. 并行算法的表示



1. 并行算法的复杂性度量

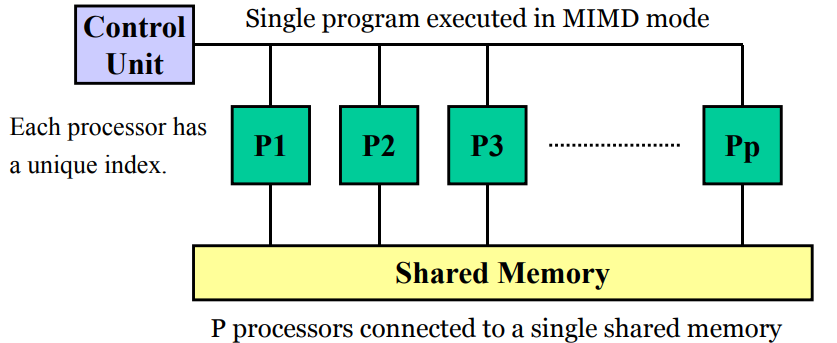




1. **PRAM模型**

**Parallel Random Access Machine 并行随机存取机器**

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，通过共享存储的R/W交换数据，隐式同步计算；在一个时钟周期内，每个处理器执行一条指令，可完成3个操作：从存储区取出操作数、完成一个算术、逻辑运算、将结果存回存储器。



PRAM模型特点：

全局共享存储，单一地址空间

同步、通信和并行化的开销为零

**优点：**

Ø简单：PRAM模型特别适合于并行算法的表达、分 析和比较，使用简单

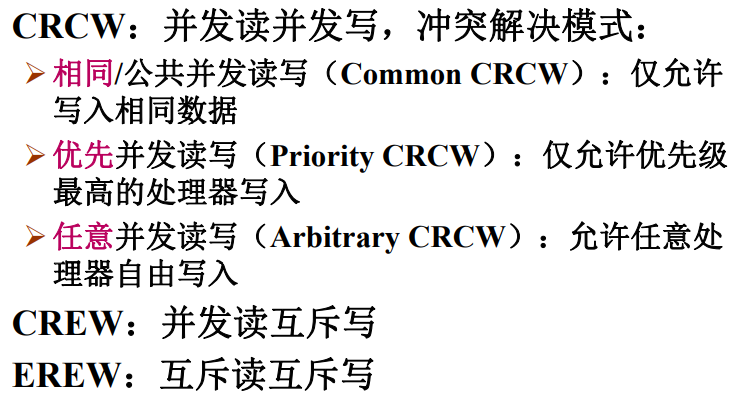
Ø易于扩展：根据需要，可以在PRAM模型中加入一 些诸如同步和通信等需要考虑的内容 •

**缺点：**

Ø 模型中使用了一个全局共享存储器，不适合于分布存储结构 的并行机

Ø PRAM模型是同步的，不能反映现实中很多系统的异步性

Ø PRAM模型假设了每个处理器可在单位时间访问共享存储器 的任一单元，因此要求处理机间通信无延迟、无限带宽和无 开销，这是不现实的存储数据的存取模式



**PRAM-CRCW是最强的计算模型。**

1. BSP模型

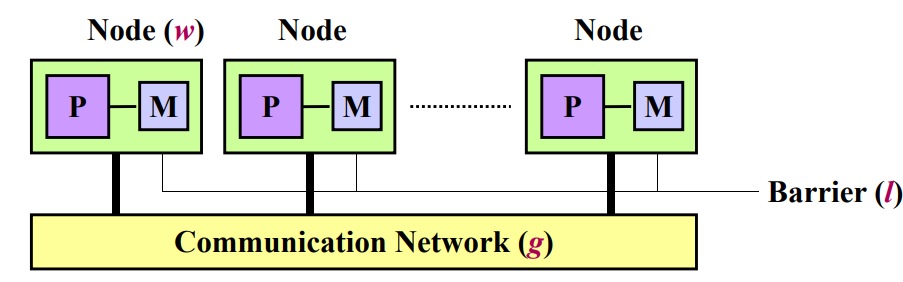
块同步模型，是一种异步MIMD-DM模型，支持消息传递系统，块内异步并行，块间显示同步。

3个组成部分：

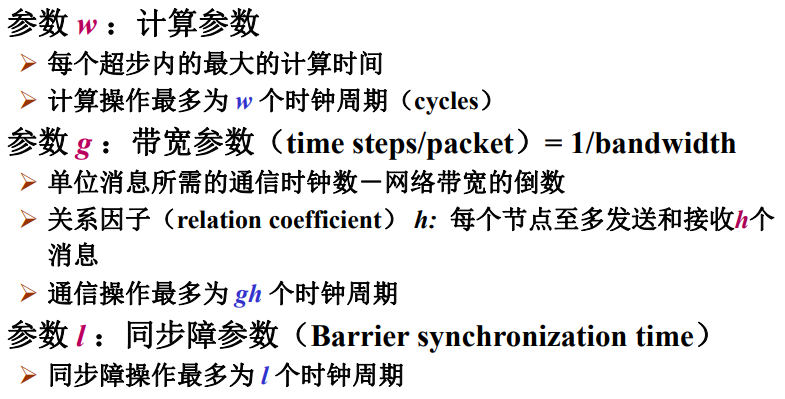
节点：处理器、本地存储

通信网络：点到点、消息传递

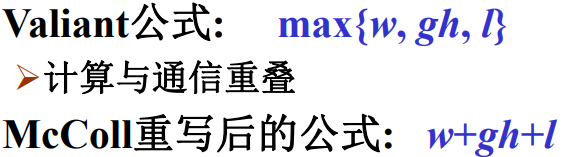
同步障：同步机制



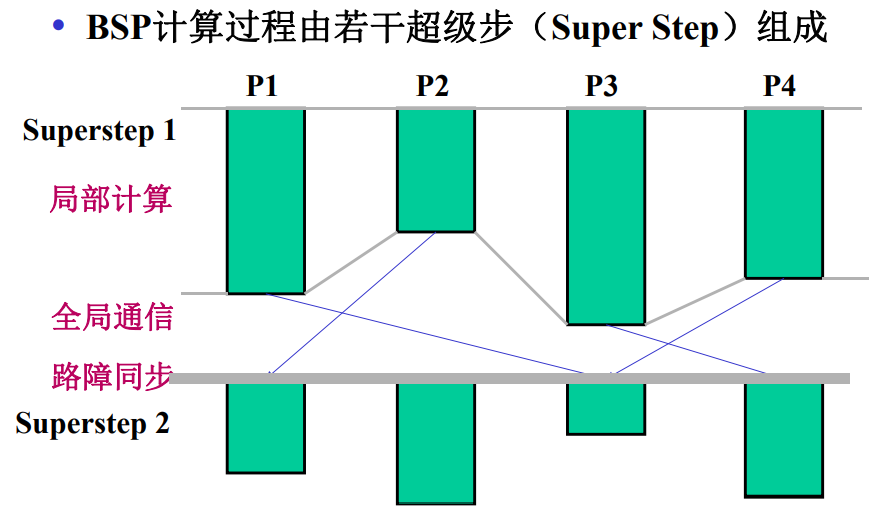
模型参数：



时间复杂度：



超步



**优点：**

Ø BSP模型试图为软件和硬件之间架起一座类似于冯诺伊曼机的桥 梁，因此，BSP模型也常叫做桥模型

Ø 将处理器和路由器分开，强调了计算任务和通信任务的分开，而 路由器仅仅完成点到点的消息传递，不提供组合、复制和广播等 功能，这样做既掩盖具体的互连网络拓扑，又简化了通信协议

**缺点：**

Ø 需要显式同步，编程效率不高

Ø BSP模型中的全局障碍同步假定是用特殊的硬件支持的，这在很 多并行机中可能没有相应的硬件

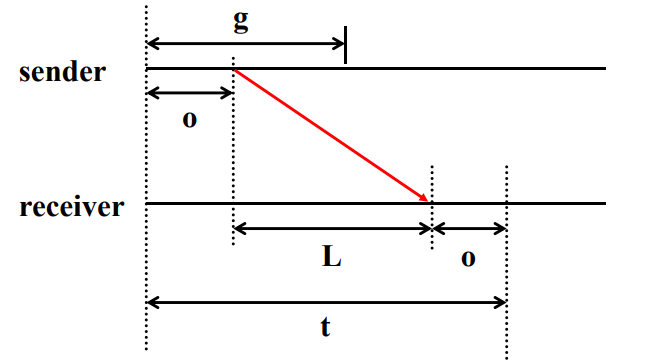
1. LogP模型

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。

模型参数：



LogP图例：



**优点：**

异步（ Asynchronous ）模型，没有同步障

捕捉了并行计算机的通讯瓶颈

通信由一组参数描述，但并不涉及具体的网络 结构，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

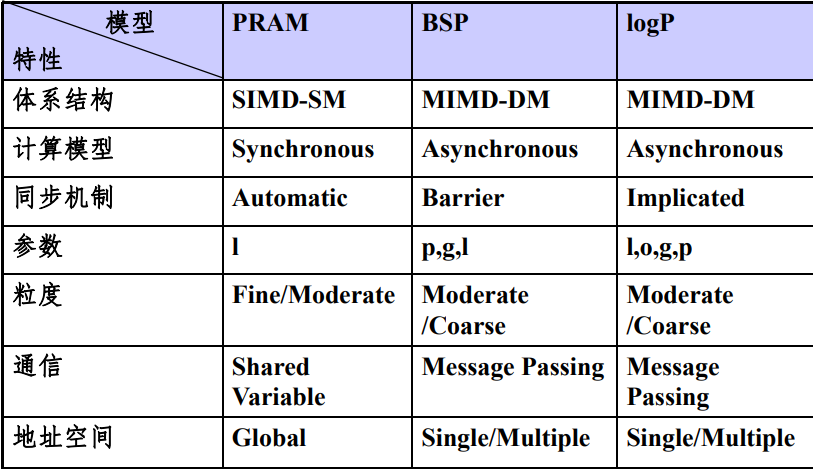
**缺点：**

• 难以进行算法描述、设计和分析

• 对网络中的通信模式描述的不够深入。如重发 消息可能占满带宽、中间路由器缓存饱和等未 加描述

• LogP模型主要适用于消息传递算法设计，对于 共享存储模式，则简单地认为远地读操作相当 于两次消息传递，未考虑流水线预取技术、 Cache引起的数据不一致性以及Cache命中率 对计算的影响

1. 三种模型的比较



1. 并行算法的基本设计策略

串行算法的直接并行化

发掘和利用现有串行算法中的并行性，直接将串行算法改造为并行算法。

从问题描述开始设计并行算法

从问题本身描述除法，不考虑相应的串行算法，设计一个全新的并行算法。

借用已有算法求解新问题

找出求解问题和某个已解决问题之间的联系；改造或利用已知算法应用到求解问题上。

## 第五讲、并行算法设计-下

1. 并行算法的基本设计技术

平衡树方法、倍增方法、分治策略、划分原理、流水线技术。

1. 平衡树方法：

树叶节点为输入，中间节点为处理节点，由叶向根或由根向叶逐层并行处理。

（求前缀和问题）

1. 倍增技术：

又称指针跳跃技术，特别适合于处理链表或有向树之类的数据结构；

当递归调用时，所要处理数据之间的距离逐步加倍，经过k步后即可完成距离为2的k次方的所有数据的计算。

（求元素表序问题）

1. 分治策略：

将原问题划分成若干个相同的子问题分而治之，若子问题仍然较大，则可以反复递归应用分治策略处理这些子问题，直至子问题易求解。

（并行快排序问题）

1. 划分原理：

将原问题划分成p个独立的规模几乎相等的子问题；

P台处理器并行地求解各子问题。

均匀划分、方根划分、对数划分、功能划分

1. 流水线技术：

将算法流程划分成p个前后衔接的任务片段，每个任务片段的输出作为下一个任务片段的输入。

（DFT离散傅里叶变换）

1. PCAM设计方法学

设计并行算法的四个阶段：划分、通信、组合、映射。

划分：分解成小的任务，开拓并发性。

通信：确定诸任务间的数据交换，检测划分的合理性。

组合；依据任务的局部性，组合成更大的任务。

映射：将每个任务分配到处理器上，提高算法的性能。

1. 划分方法描述：

分为两类划分：

先进行数据分解称为域分解，再进行计算功能的分解称为功能划分。

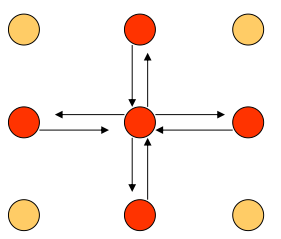
1. 通信方法描述：

通信是PCAM设计过程的重要阶段；划分产生的诸任务，一般不能完全独立执行，需要在任务间进行数据交流；从而产生了通信；功能分解确定零诸任务之间的数据流；诸任务是并发执行的，通信则限制了这种并发性。

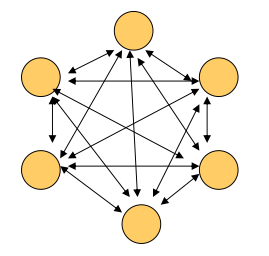
四种通信模式：

局部/全局通信、结构化/非结构化通信、静态/动态通信、同步/异步通信

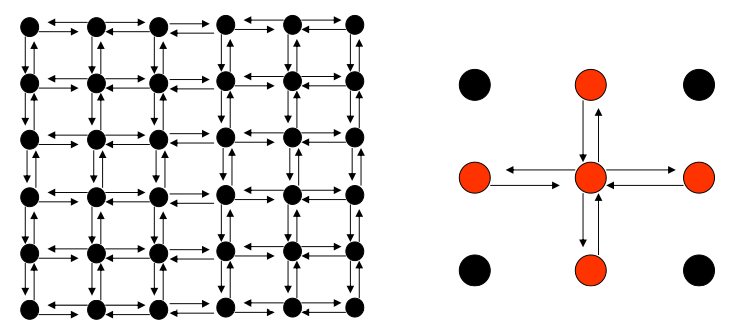
局部通信：通信限制在一个领域内：



全局通信：许多任务参与，非局部的



结构化通信：每个任务的通信模式是相同点



非结构化通信：没有一个统一的通信模式

动态通信：通信模式不规整

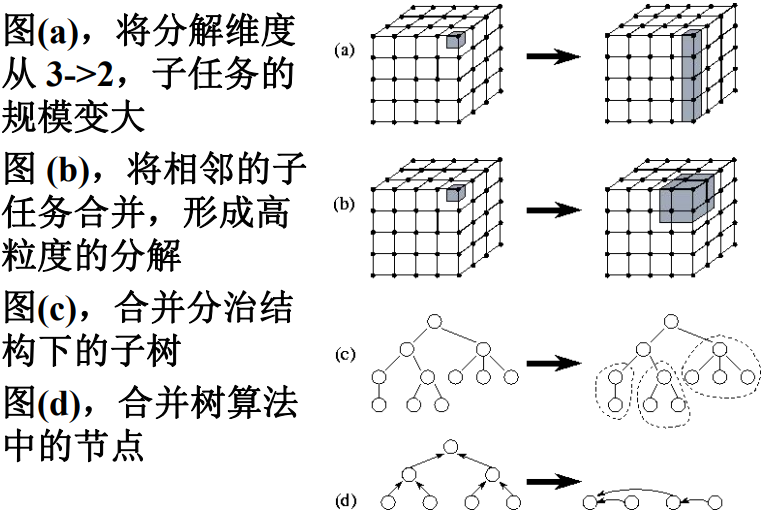
同步通信：通信双方同时产生通信操作

异步通信：非同步通信操作，一般由接收者请求

1. 组合方法描述

组合是由抽象到具体的过程，是使得组合的任务能在一类并行机上有效地执行。合并小尺寸任务，减少任务数，如果任务数恰好等于处理器数，则也完成了映射过程。通过增加任务的粒度和重复计算，可以减少通信成本。保持映射和扩展的灵活性，降低软件工程成本。

组合例子：



表面-容积效应

通信量与任务子集的表面成正比，计算量与任务子集的体积成正比。增加重复计算有可能减少通信量。

重复计算减少通信量，但增加了计算量，应保持恰当的平衡。

1. 映射方法描述

每个任务要映射到具体的处理器，第六个为到运行机器上。任务数大于处理器数时，存在负载平衡和任务调度问题。

映射的目标：减少算法的执行时间。

Ø并发的任务 ◊ 不同的处理器 Ø任务之间存在高通信的 ◊ 同一处理器

映射的两种策略：

使得任务可以被不同的处理器并发的执行，增强并发性；

将通信频繁的任务放到一个处理器上，增强局部性。

1. 负载平衡算法

静态的：事先确定；概率的：随机确定；动态的：执行期间动态负载；

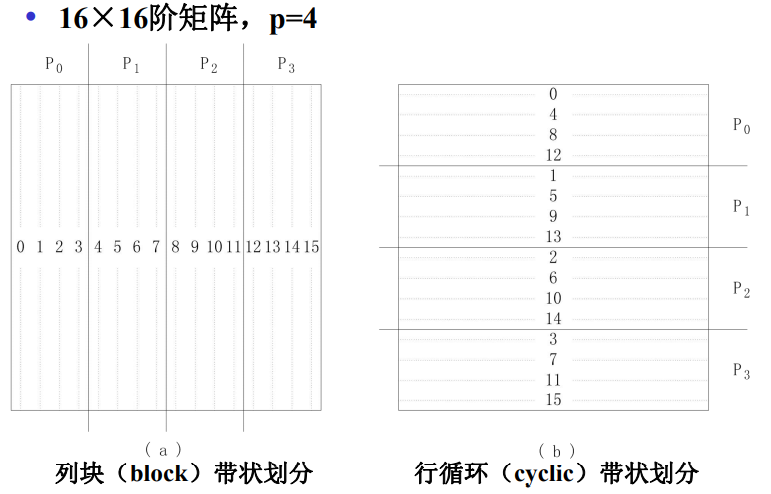
基于域分解的：递归对剖、局部算法、概率方法、循环映射。

## 第六讲、并行程序设计基础

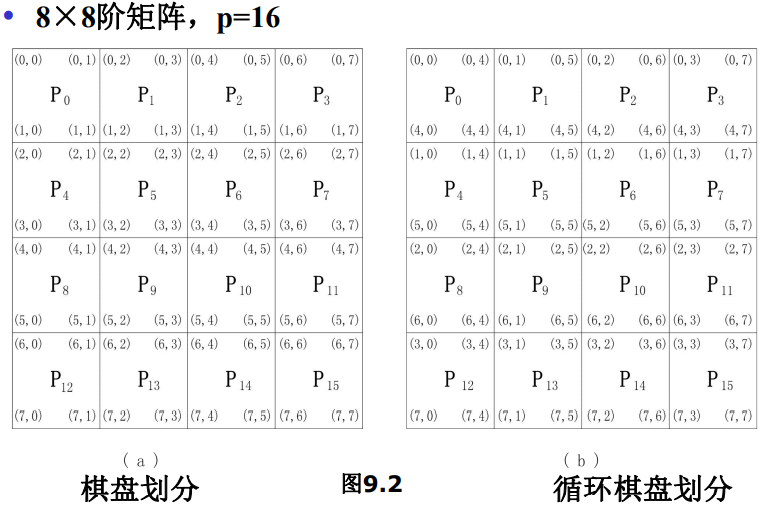
1. 并行算法的设计例子：矩阵乘法

划分方法：带状划分、棋盘划分

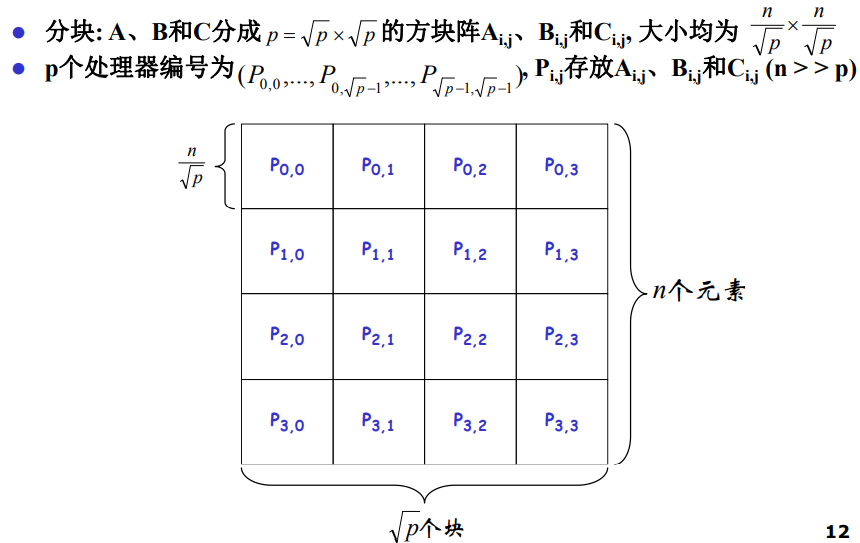
带状划分：



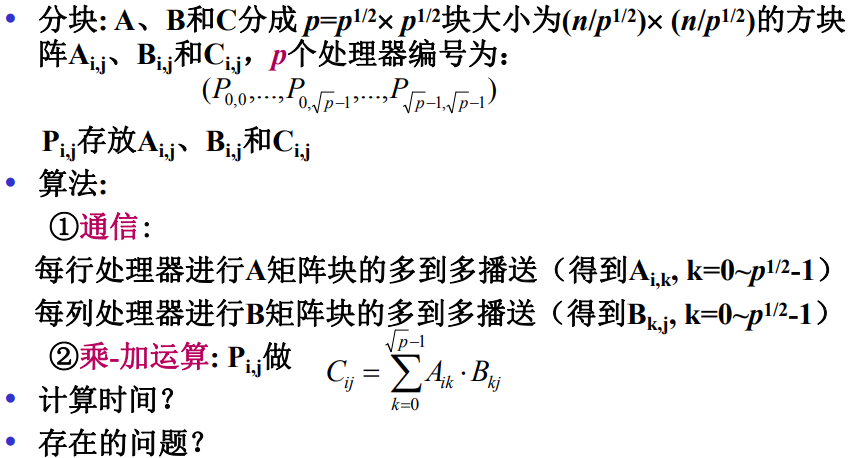
棋盘划分：



1. 矩阵分块

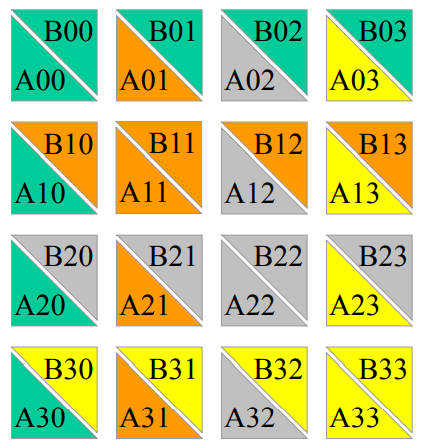


1. 简单并行分块算法

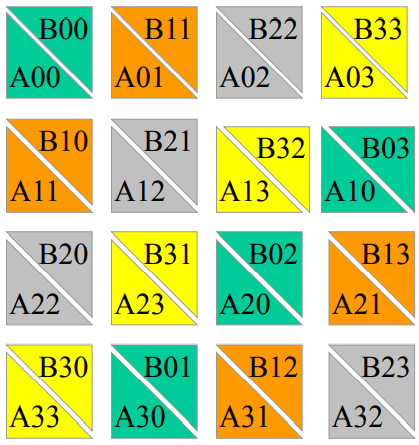


1. Cannon乘法

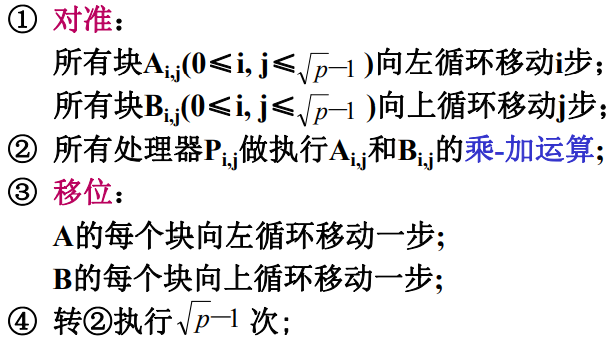
矩阵分块的颜色表示：每个三角代表一个矩阵块，只有相同颜色的三角可以相乘。



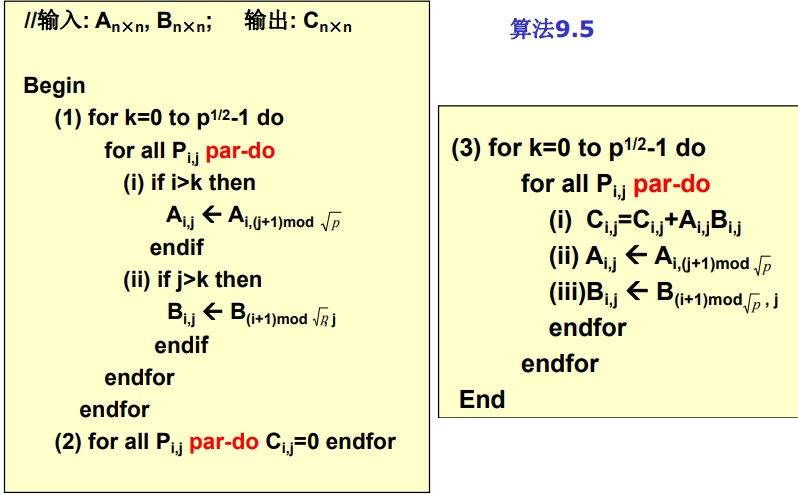
块的重排，块Aij向左循环移动i步，块Bij向上循环移动j步



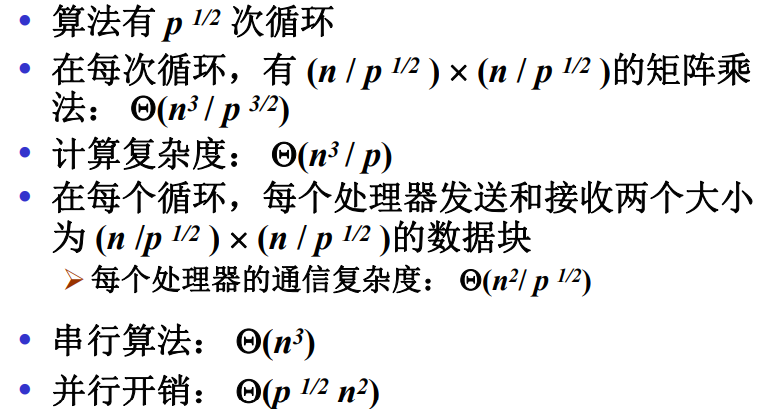
Cannon算法描述：



伪代码：



复杂度分析：

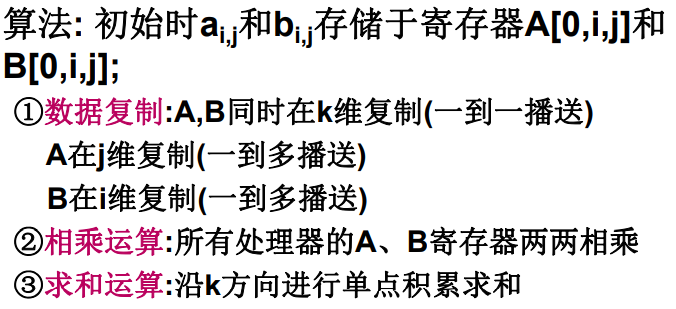


1. DNS

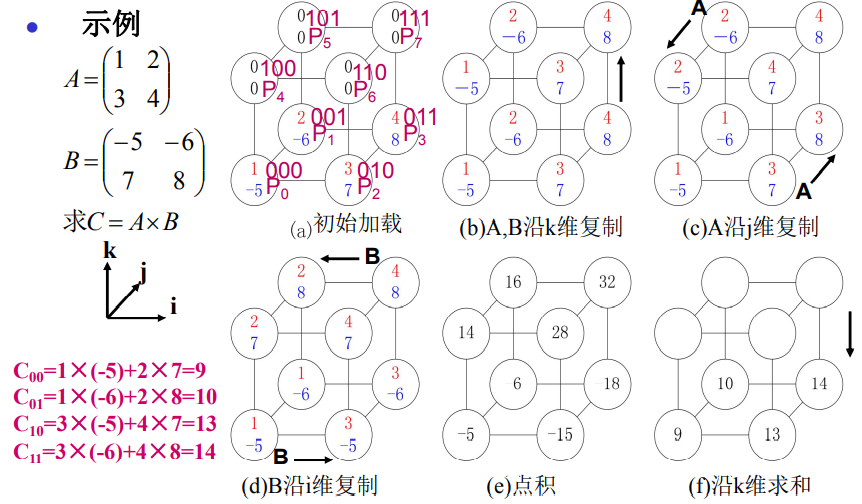
利用更多的处理器达到更高的加速比。

基本思想：通过一到一和一到多的播送方法，使得处理器（k,i,j）拥有Aik和Bkj，进行本地相乘，再沿k方向进行单点积累求和，结果存储在处理器（0,i,j）中。

算法描述：



矩阵乘法示例：



1. 并行编程风范

相并行、分治并行、流水线并行、主-从并行、工作池并行。

1. 并行程序设计的基本问题

进程的同构性：

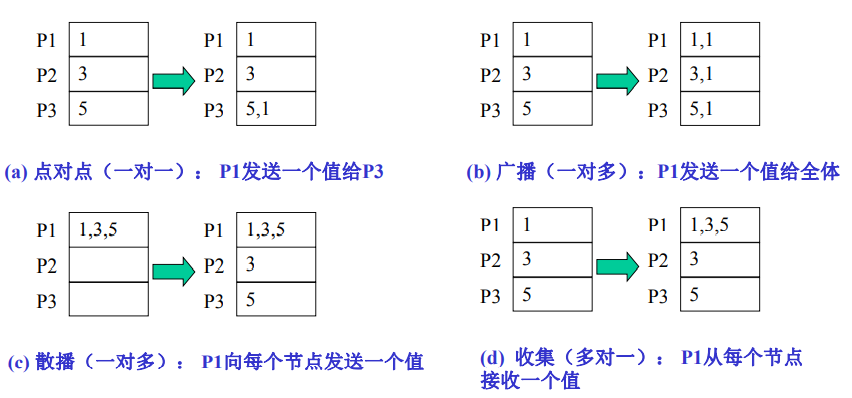
进程是并行程序的基本计算单位。SIMD：所有进程再统一时间执行相同的指令；MIMD：各个进程在同一时间可以执行不同的指令。

静态和动态并行性：

静态并行性表示程序的结构以及进程的个数在运行之前就可确定，就认为该程序具有静态并行性；否则就认为该程序具有动态并行性，即意味着进程要在运行时创建和终止。

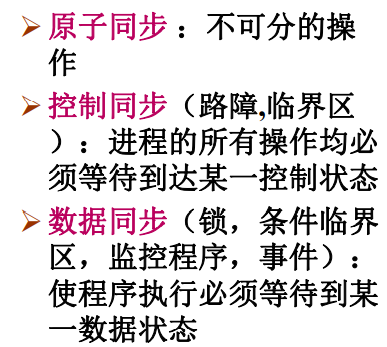
进程交互：

通信：两个或多个进程间进行传送数的操作，方式有共享变量、父进程传给子进程、消息传递。





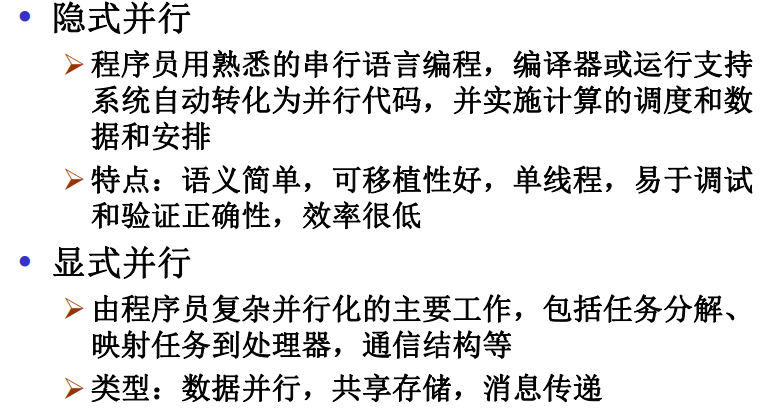
同步：导致进程间相互等待或继续执行的操作。



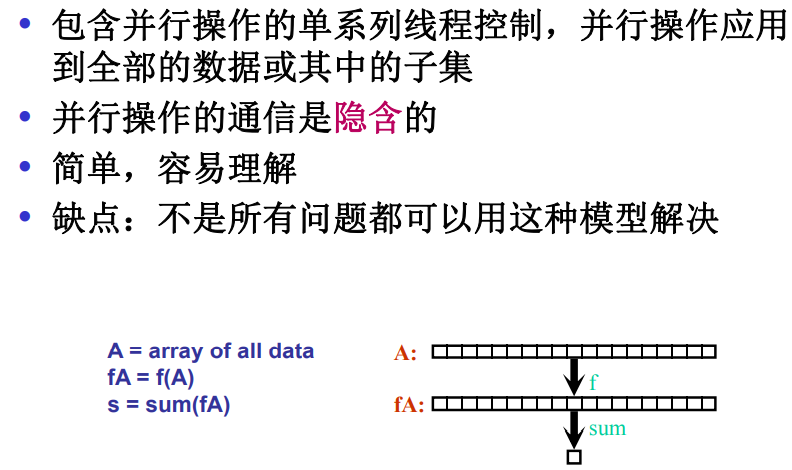
聚集：用一串超步将各分进程计算所得的部分结果合并为一个完整的结果，每个超步包含一个短的计算和一个简单的通信或/和同步。

1. 并行程序设计模型

隐式和显示并行：



数据并行：



共享存储：

消息传递：

## 第七讲、共享存储编程

1. 共享存储编程

所有线程可以存取相同的、全局共享的存储空间；

数据可以是共享的或私有的；

共享数据可以被所有线程所存取；

私有数据只能被拥有者线程所存取；

数据传输对程序员是透明的；

需要显示说明同步，也有许多是隐式的。

1. 线程与进程的区别

多个进程的内部数据和状态都是完全独立的，而多线程是共享一块内存空间和一组系统资源，有可能互相影响；

线程本身的数据通常只有寄存器数据，以及一个程序执行时使用的堆栈，所以线程的切换比进程切换的负担要小。

1. 什么是是OpenMP？

OpenMP应用编程接口API是在共享存储体系结构上的一个编程模型。

1. OpenMP体系结构

三个API分量：编译制导、运行库例程、环境变量。

1. 为什么需要OpenMP

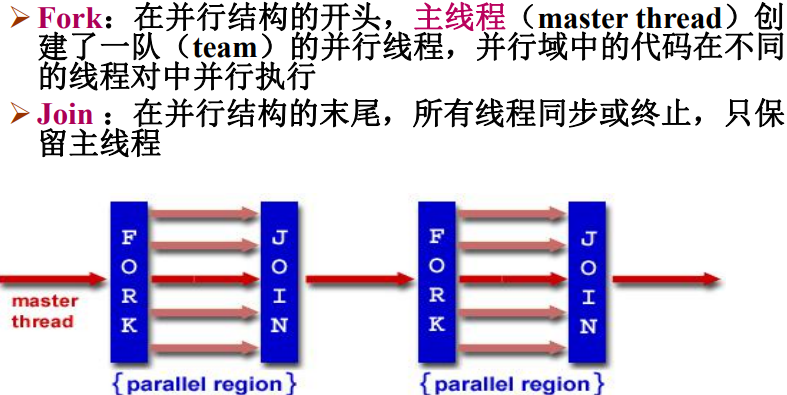
分布存储编程如MPI对程序员要求高，大量的科学应用程序需要很好地被继承和移植；

通过使用OpenMP编程接口，可以较好的利用多核处理器的并发功能，提高程序的执行效率。

1. OpenMP并行编程模型

基于线程的并行编程模型。

OpenMP使用Fork-Join并行执行模型：



1. OpenMP控制结构



1. OpenMP数据域



## 第八讲、消息传递编程

1. 消息传递

分布式存储类型的并行机，一台处理器无法直接访问另一台处理器的地址空间。

消息传递：显式地通过发送和接受消息来实现处理器之间的数据交换。子程序间通信，复杂又灵活。

消息传递时MPP和Cluster的主要编程方式。

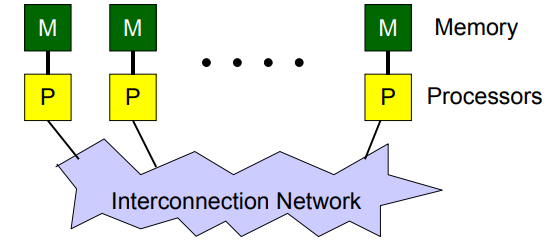
1. 消息传递编程模型

在一个消息传递系统中，每个处理器运行一个子程序

类似通常的串行程序；

所有变量都是私有的；

通过特殊的函数调用来通信。



1. 消息

消息时指子程序之间传递的数据包。

存取：每个子系统需要连接到消息传递系统。

地址：消息要有地址才能发送。

接收方：接受进程能够处理消息。

一个消息传递系统类似于：邮局、电话、传真、电子邮件等。

通信类型：

点到点、群集。

1. 点到点通信

最简单的消息传递模式：一个进程发送消息给另一个。

几种方式：

同步发送：提供消息完成的消息，类似电话、传真

异步发送：仅知道消息已经发走，类似明信片

阻塞操作：仅当传输操作完成才从调用中返回。

非阻塞操作：直接返回-以后再测试（或等待）是否完成。

1. 群集通信

一次有多个进程参与，可以建立在点到点通信的基础上。例如同步、广播、规约操作。

1. 什么是MPI？

MPI是一个消息传递接口标准；

MPI提供一个可移植、高效、灵活的消息传递接口库；

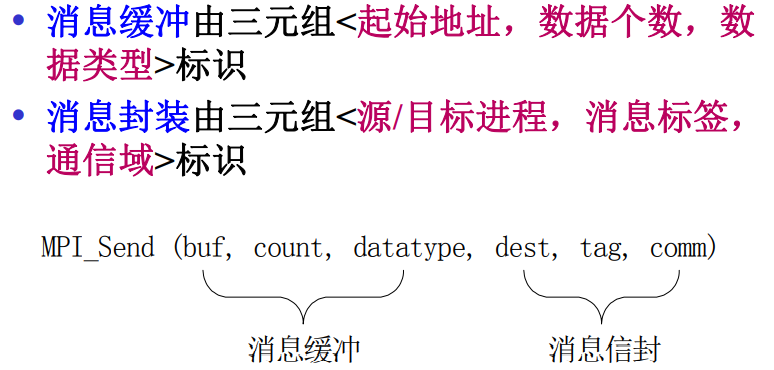
MPI以语言独立的形式存在，可运行在不同的操作系统和硬件平台上。

1. MPI消息

一个消息好比一封信；

消息的内容即信的内容，在MPI中称为消息缓冲；

消息的接收者即信的地址，在MPI中成为消息封装。



1. 消息标签

添加标签使得服务进程可以对两个不同的用户进程分别处理，提高灵活性。

1. 数据类型

MPI的消息类型分为两种：预定义类型和派生数据类型。

预定义数据类型：MPI通过提供预定义数据类型来解决异构计算中的互操作性问题，建立于具体语言的对应关系。

派生数据类型：MPI引入派生数据类型来定义由数据类型不同且地址空间不连续的数据项组成的消息。

1. 通信域

通信域包括进程组和通信上下文等内容，用来描述通信进程间的通信关系。

通信域分为组内通信域和组间通信域，分别用来实现MIPI的组内通信和组间通信。

进程组：进程的有限、有序集。



通信上下文：安全地区别不同的通信以免相互干扰。



1. 消息状态

存放接受消息的状态信息，包括消息的源进程标识、消息标签、包含的数据项个数等。

1. 点对点通信

通信模式：

同步通信模式：只有相应过程已经启动，发送过程才正确返回。

缓冲通信模式：缓冲通信模式的发送不管接受操作是否已经启动都可以执行。

标准通信模式：是否对发送的数据进行缓冲由MPI的实现来决定，而不是由用户程序来控制。

就绪通信模式：发送操作只有在接受进程相应的接受操作已经开始才进行发送。

1. 群集通信

是一个进程组中的所有金册灰姑娘都参加的全局操作。

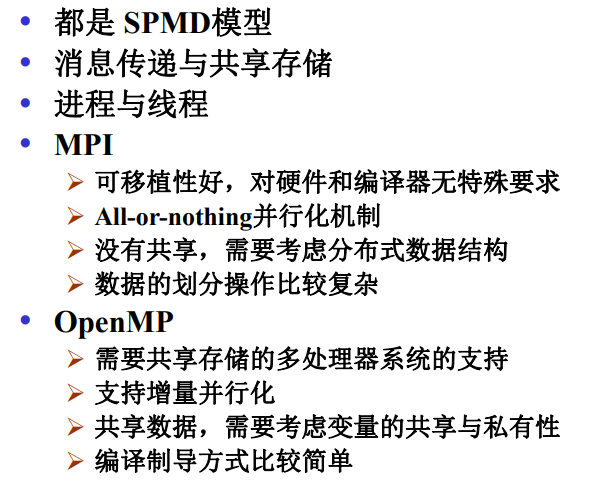
一般实现三个功能：

通信：主要完成组内数据的传输。

聚集：在通信的基础上对给定的数据完成一定的操作。

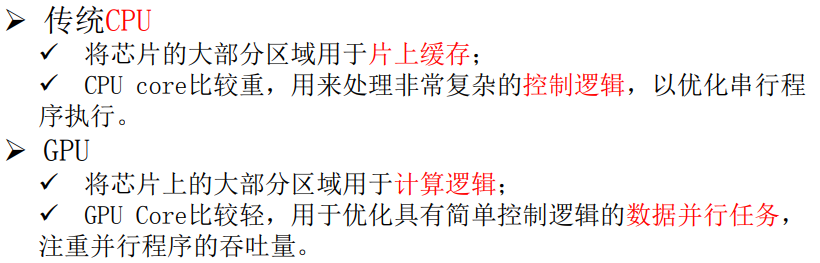
同步：实现组类所有进程在执行进度上去的一致。

1. MPI和OpenMP



## 第九讲、CUDA编程

1. CPU和GPU的比较



1. CUDA

Compute Unified Device Architecture，2006年英伟达发布，将 GPU计算由传统图形处理领域扩展到通用计算领域，支持c，c++， Fortran和Python等语言，提供2组API（一个底层，一个更高层），是实 现更高层第三方API和库的基础。CUDA只针对英伟达硬件。

## 第十讲、大规模数据处理及Map/Reduce编程

1. MapReduce：大规模数据处理

MapReduce是一个编程模型，提供：

自动的并行化和分布；

容错；

I/O调度/状态和监控；

运行时系统处理输入数据的分割、执行调度、处理及其故障，以及管理所需的机器内部通信等细节。

这使得没有任何并行和分布式系统经验的程序员能够容易地利用一个大的分布式系统的资源。

1. MapReduce编程模型设计理念

MapReduce将复杂的、运行于大规模集群上的并行计算过程高度的抽象到了两个行数：Map和Reduce。

MapReduce采用“分而治之”策略，一个存储在分布式文件系统中的大规模数据集，会被切分成许多独立的分片，这些分片可以被国歌Map任务并行处理。

MapReduce设计理念是“计算向数据靠拢”，而不是“数据向计算靠拢”，因为数据的迁移需要大量的网络传输开销。

1. MapReduce工作流程



1. Map

数据源的记录（如：文件行、数据库的 行等）以key\*value 对（如：文件名，行 ）的形式送给map函数

map()产生一个或多个中间结果值 （ intermediate values ），每个中间结果值 都带有一个输出key

1. Reduce

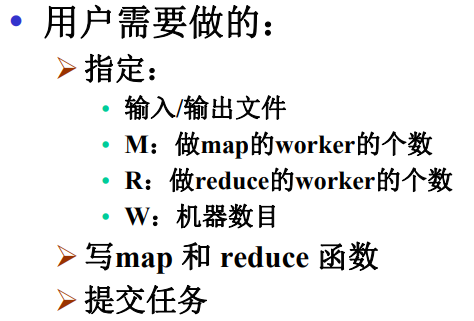
• 在map过程结束后，所有属于同一输出 key的中间值并合并成一个列表

• reduce() 将具有相同输出key的中间值合 并为一个或多个最终值（final values ）

Ø一般一个最终结果值对应一个key

只有在map阶段完全结束后才能启动reduce阶段。

1. 如何用MapReduce？



## 第十一讲、云计算平台

1. 云计算的5个特征

Ø 随需应变自助服务

Ø 随时随地用任何网络设备访问

Ø 多人共享资源池

Ø 快速重新部署灵活度

Ø 可被监控与量测的服务

1. 云计算的3中服务模式

Ø软件即服务（SaaS）

Ø平台即服务（PaaS）

Ø基础架构即服务（IaaS）

1. 什么是虚拟化？

在虚拟化技术中，可同时运行多个操作系统，而且每一个操作系统中都有多个程序在运行，每一个操作系统都运行在一个虚拟的CPU或者是虚拟主机上。

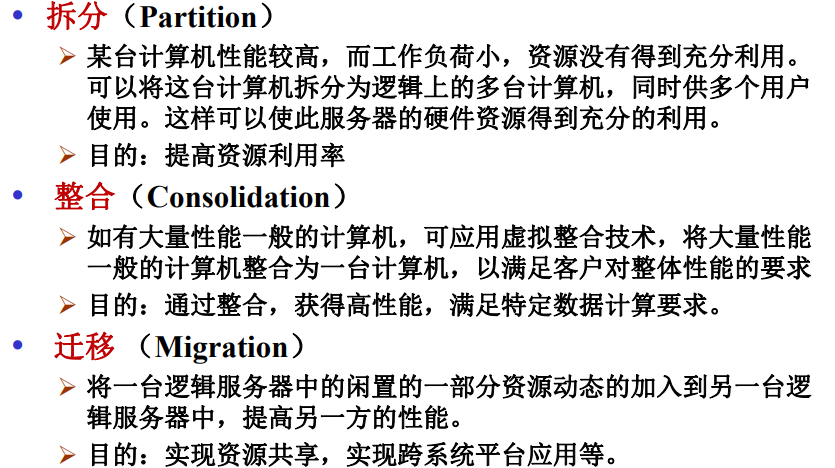
1. 服务器虚拟化

• 将服务器物理资源抽象成逻辑资源，让一台服 务器变成几台甚至上百台相互隔离的虚拟服务 器，不再受限于物理上的界限

• 让CPU、内存、磁盘、I/O等硬件变成可以动 态管理的“资源池” ，

• 可提高资源的利用率，简化系统管理，实现服 务器整合，让IT对业务的变化更具适应力

1. 服务器虚拟化的类型



1. 云存储和传统存储的区别

