第7章
多处理机与多计算机

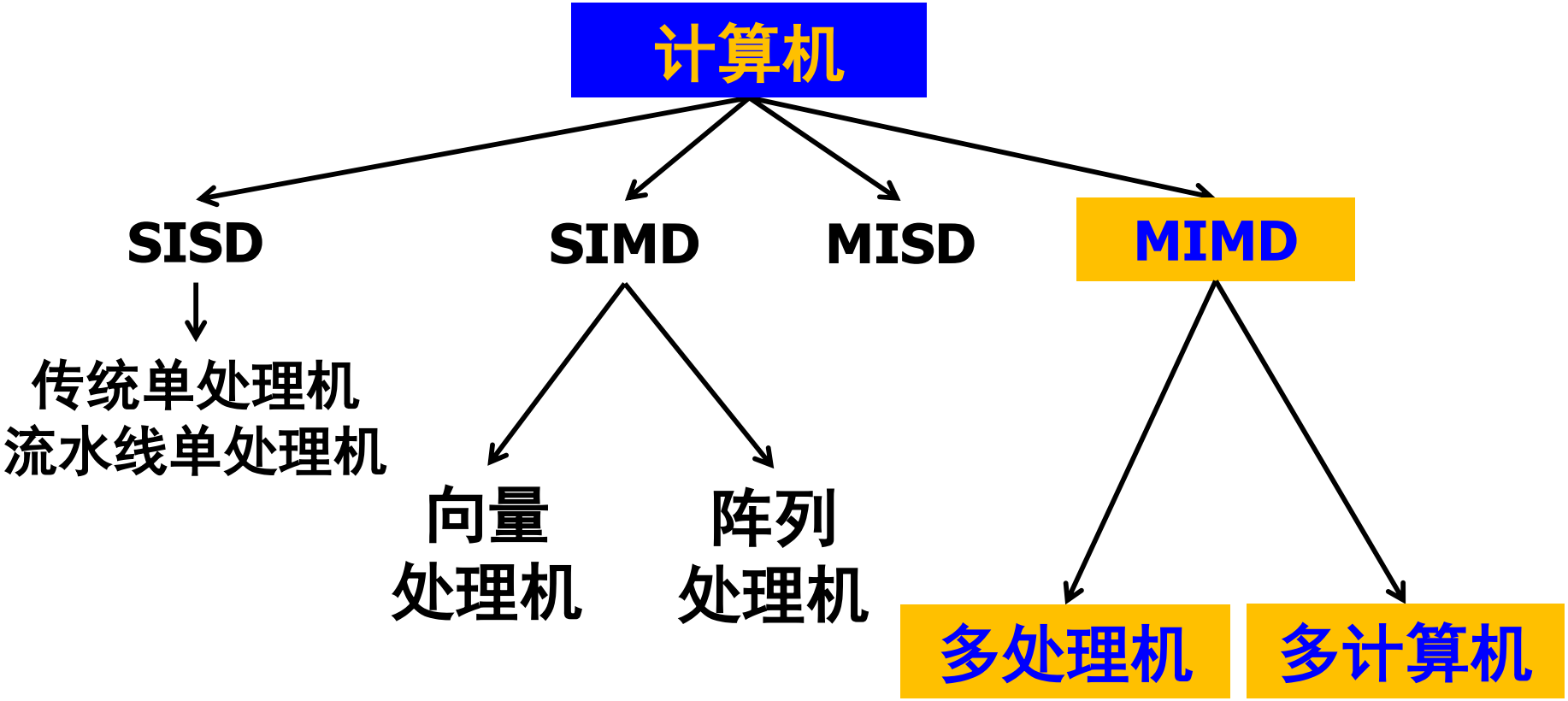
郑宏 副教授
计算机学院
北京理工大学

学习内容

- **7.1 多处理机概念**
- **7.2 多处理机结构**
- **7.3 多核处理器**
- **7.4 多处理机的多Cache一致性**
- **7.5 多处理机的机间互连形式**
- **7.6 程序并行性**
- **7.6 多处理机性能**
- **7.7 多处理机操作系统**

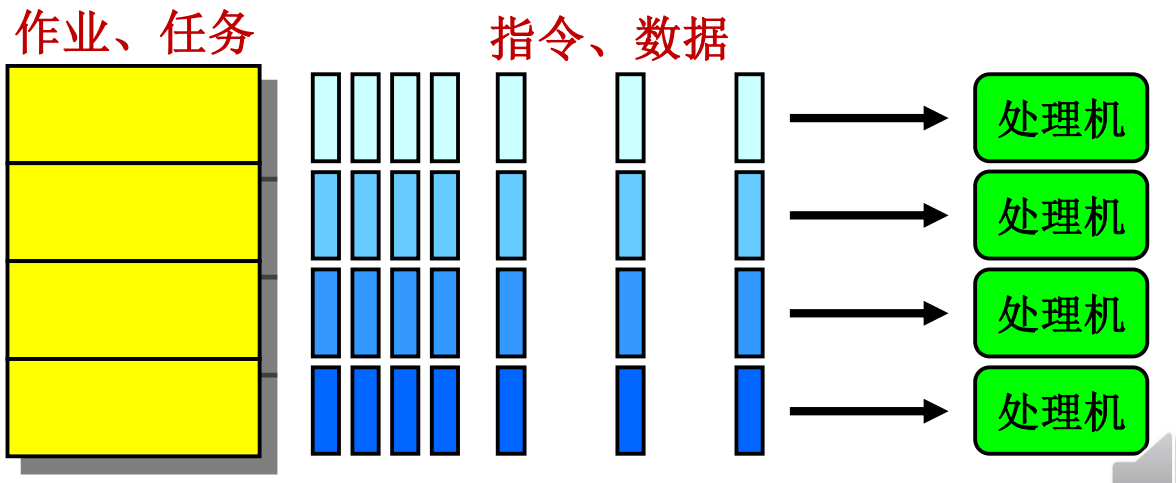
7.1 多处理机概念

- 弗林分类法：并行处理能力



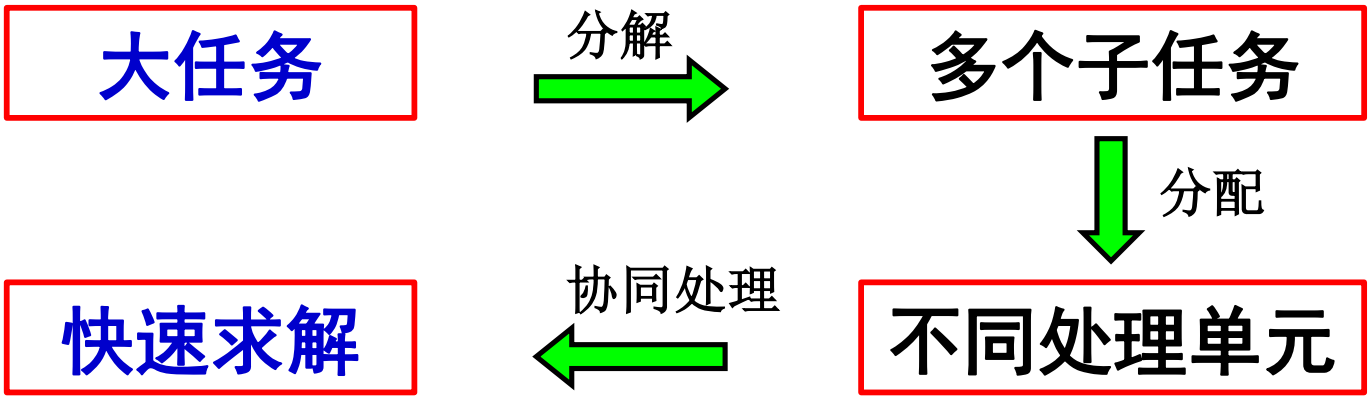
7.1.1 多处理机定义

- **多处理机：**
  - 由两台及以上处理机组成的计算机系统；
  - 各处理机有自己的控制部件、局部存储器，能执行各自的程序，可以共享公共主存和所有外设；
  - 各处理机通过某种形式互连，相互通信；
  - 实现作业、任务、指令、数据等各个级别的并行；
  - 属于MIMD。



7.1.1 多处理机定义

- **多处理：**
  - 在多个处理机上运行同一道程序或作业的不同任务。可以串行，也可以并行。
- **并行处理：并行计算，高性能计算**
  - 同时执行两个或更多个处理的一种计算方法。



7.1.1 多处理机定义

- **多处理机优点：**
  - 提高性能；
  - 提高可靠性；
  - 减少及其功耗；
  - 提高效费比 ；
  - 等。

7.1.2 多处理机分类

- 多种分类方法
- 根据实现并行技术途径，分为3类：
  - 同构型：基于资源重复
  - 异构型：基于时间重叠
  - 分布式：基于资源共享

7.1.2 多处理机分类

同构型

- 基于资源重复
- 处理机类型/功能相同
- 并行处理或执行任务

异构型

- 基于时间重叠
- 处理机类型/功能不同
- 重叠处理各任务

分布式

- 基于资源共享
- 处理机类型/功能相同或不同
- 并行/协同完成任务处理
- 由统一操作系统统一管理资源

7.1.2 多处理机分类

- 耦合度：
  - 反映多处理机之间物理连接的紧密程度和交叉作用的能力的强弱
- 根据耦合度，分为2类：
  - 紧耦合多处理机
  - 松耦合多处理机

7.1.2 多处理机分类

- 紧耦合多处理机
  - 直接耦合系统
  - 通过公共硬件（如存储器，I/O系统等）通信
  - 典型：SMP，多核处理器等
  - 一般将紧耦合多处理机称为多处理机
- 松耦合多处理机
  - 间接耦合系统
  - 通过通道、通信线路或网络、消息传递系统通信
  - 典型：机群，计算机网络等
  - 一般将松耦合多处理机称为多计算机

7.1.3 多处理机特点和主要技术问题

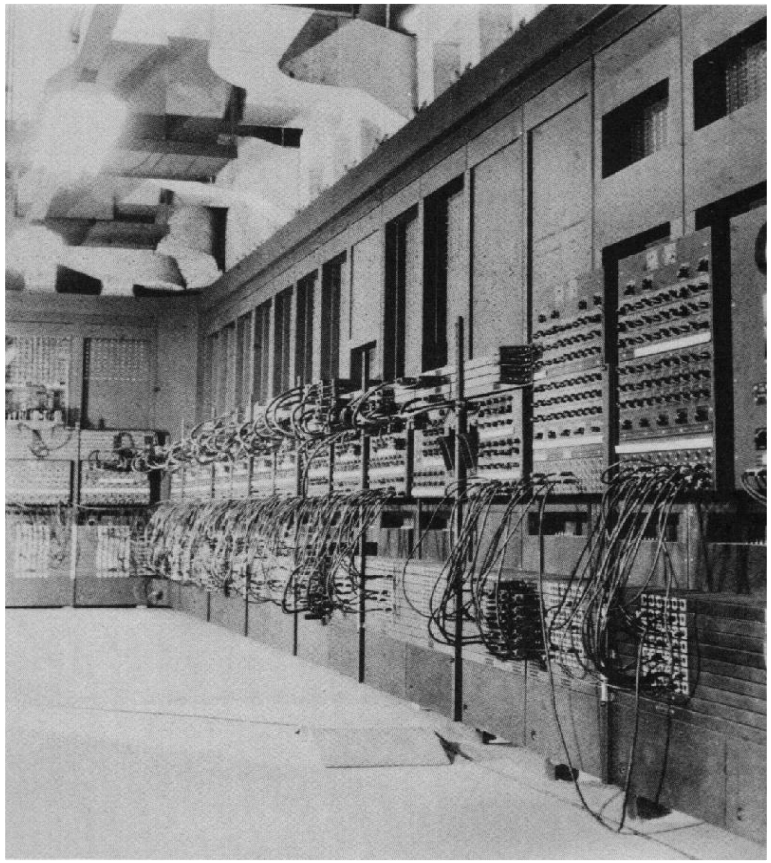
- 多处理机特点：
  - **结构灵活：**满足多种并行计算的不同需求。
  - **程序并行：**实现作业、任务之间的并行。
  - **并行任务派生：**可并行执行任务的识别、派生与分配。
  - **进程同步：**解决数据相关和控制依赖问题。
  - **资源调度与分配：**动态分配资源和调度任务，以获得更好性能和更高效率。

7.1.3 多处理机特点和主要技术问题

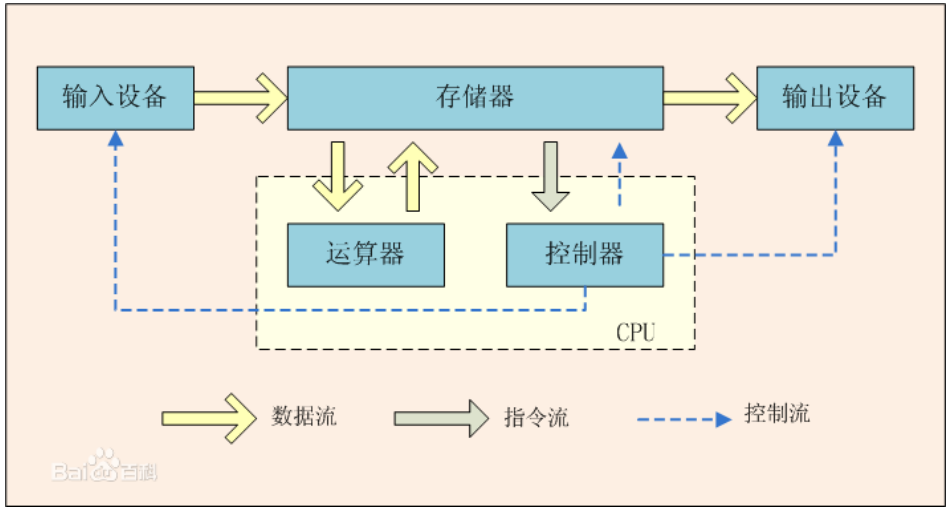
- 多处理机主要技术问题：
  - **结构灵活性与通用性：**以适应、满足多种并行计算算法不同的需求
  - **进程通信方法：**共享内存、互连网络、消息传递机制等
  - **运行模型：**数据并行 + 处理并行
    - 基于共享内存的运行模型
    - 基于消息传递机制的运行模型
  - **并行性表达：**
    - 编译程序自动发现：如 并行C或Fortran
    - 并行程序设计语言：如 HPF(High Performance Fortran)
    - 运行时库：如 LAPACK
  - **并行算法**

7.1.4 并行计算机发展

- 1946 年，世界上第一台计算机ENIAC 诞生



用途：弹道计算和氢弹研制



约翰·冯·诺伊曼
John von Neumann
1903 - 1957

匈牙利人、美国人
数学、计算机科学、
经济学

7.1.4 并行计算机发展

- 1972年，第一台并行计算机ILLIAC IV(美国伊利诺依大学)

- 60 年代末开始建造，72 年建成
- 74 年运行第一个完整程序，76年运行第一个应用程序
- 64 个处理器，SIMD，是当时性能最高的 CDC7600 机器的2-6倍
- 公认的1981年前最快，1982年退役
- 可扩展性好，但可编程性差

7.1.4 并行计算机发展

- 1976 年，超级计算元年：Cray-1 向量机

1925-1996
电子工程学学士
应用数学硕士
超级计算之父
Cray研究公司的
创始人

- 一般将Cray-1 投入运行的1976 年称为“ 超级计算元年”
- 编程方便，但可扩展性差
- 以Cray 为代表的向量机称雄超级计算机界十几载

7.1.4 并行计算机发展

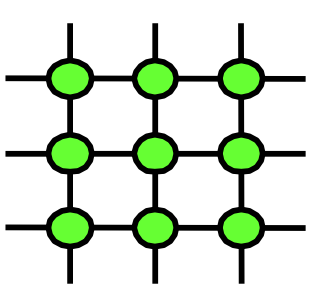
- **80 年代初期：以MIMD 并行计算机的研制为主**
  - **Denelcor HEP （1982年）：第一台商用MIMD 并行计算机**
  - **IBM 3090：80 年代普遍为银行所采用**
  - **Cray X-MP： 1982，Cray研究公司发布了第一台MIMD并行计算机。**

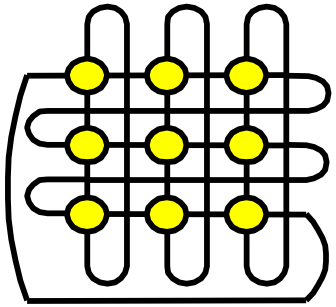
7.1.4 并行计算机发展

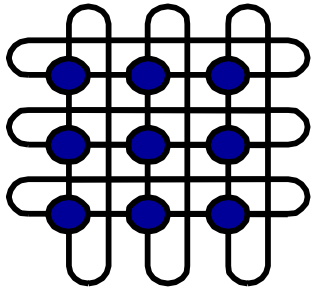
- **80 年代初期：共享存储器多处理机（Shared-Memory Multi-Processor）**
  - **SMP (Symmetrical Multi-Processing)：在一个计算机上汇集一组处理器，各处理器对称共享内存及计算机的其他资源，由单一操作系统管理，极大提高整个系统的数据处理能力**

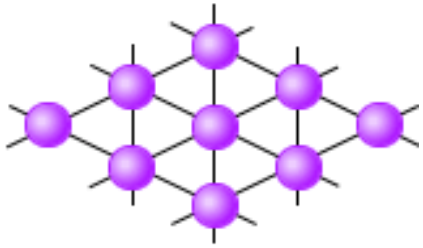
7.1.4 并行计算机发展

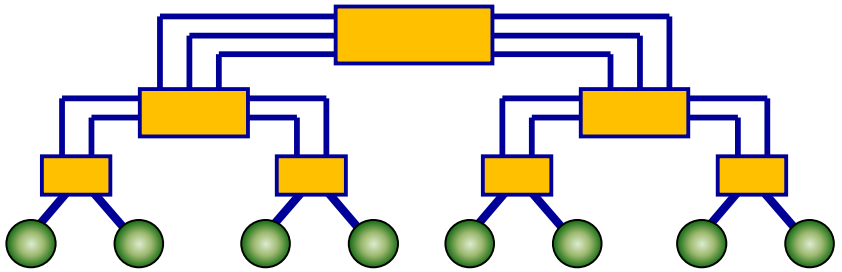
- **80 年代后期：具有强大计算能力的并行机**
  - 通过二维Mesh连接的Meiko（Sun）系统
  - 超立方体连接的MIMD 并行机：nCUBE-2(27Gflops)、Intel iPSC/80(7Gflops)
  - 共享存储向量多处理机Cray Y-MP 等

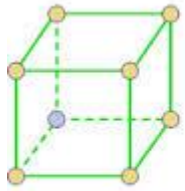




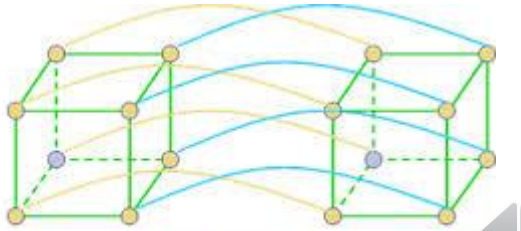








3-立方体



4-立方体

7.1.4 并行计算机发展

- **90 年代初期：MIMD 类型占据绝对主导位置**
  - **MPP (Massively Parallel Processing)：大规模并行处理结构**
    - 每个结点相对独立，有一个或多个微处理器
    - 每个结点有自己的操作系统和独立内存，避免内存访问瓶颈
    - 各个结点只能访问自己的内存模块
    - 通过高速互联网络构成
    - 扩展性较好

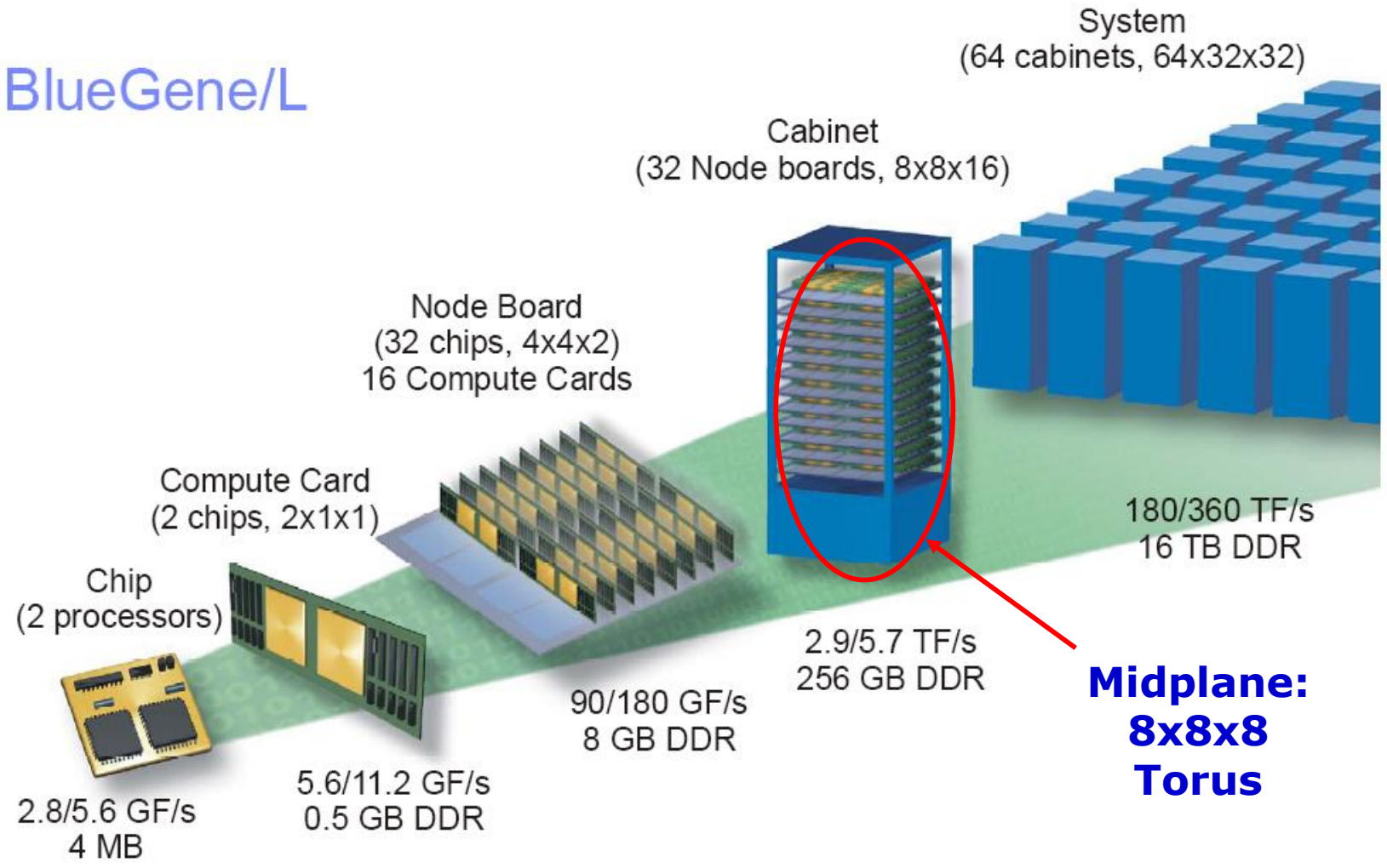
First computer to defeat a world champion!!

Garry Kasparov

Deep Blue

In February 1996, IBM's Deep Blue defeated grandmaster Garry Kasparov. It was then assigned to predict the weather in Atlanta, Georgia, during the 1996 Summer Olympic Games

Blue Gene/L: An Example



7.1.4 并行计算机发展

- **90 年代初期：MIMD 类型占据绝对主导位置**
  - **分布式存储 MPP 并行机主要有：**
    - **Intel Touchstone Delta：含512 个i860 微处理芯片，二维Mesh连接，峰值性能为32Gflops，8GB 内存。**
    - **CM–5E：含16–1K 个标量RISC SPARC 处理器（含四个向量部件，峰值性能128Mflops，32MB 内存），胖树数据网、二叉控制网及四叉诊断网连接各个处理器。**
    - **CRAY T3D：16–1K 个结点，每结点含2 个处理器（64 位RISCDECchip 21064，峰值150Mflops），局部内存64MB，最大存储规模128GB，结点间双向三维Torus 连接。**

7.1.4 并行计算机发展

- **90 年代初期：MIMD 类型占据绝对主导位置。**
  - **共享存储并行机：**
    - **CRAY Y-MP C90：由16台向量机构成的并行机 (峰值性能16Gflops)**
    - **国产YH-2 (4 台向量机，峰值性能1Gflops)**
  - **分布共享存储并行机：**
    - **Kendall Square KSR–1： 1991，它提供给用户透明的共享存储结构，每个环含32 个结点，多个环以层次结构相互连接，可扩展到1024 个结点，峰值性能为15Gflops。**

7.1.4 并行计算机发展

- **90 年代中后期：高性能微处理器+高性能网络。体系结构框架趋于统一 (DSM、SMP、NOW)**
  - **DSM (Distributed Shared Memory)：分布式共享存储**
    - 以结点为单位，每个结点有一个或多个CPU
    - 专用的高性能互联网络连接(Myrinet, Infiniband, …)
    - 分布式存储：内存模块局部在每个结点中
    - 单一的操作系统，单一的内存地址空间
    - 可扩展到上百个结点，支持消息传递、共享存储并行程序设计

7.1.4 并行计算机发展

- **90 年代中后期：高性能微处理器+高性能网络。体系结构框架趋于统一 (DSM、SMP、NOW)**
  - **DSM (Distributed Shared Memory)：分布式共享存储**
    - **NUMA 结构：在原来分布式存储并行机的基础上，增加局部内存的全局共享功能，提供共享存储并行编程环境。代表：典型代表为Cray T3E 和Cray T3E-1200，最多可扩展到2048个CPU，采用了当时最先进的微处理芯片（主频600MHz），峰值性能达到2.5Tflops。**
    - **CC–NUMA结构：CC–NUMA 结构具有比SMP 更好的可扩展性，并且能保持SMP 的共享存储特性。代表：SGI Origin 系列超级服务器**

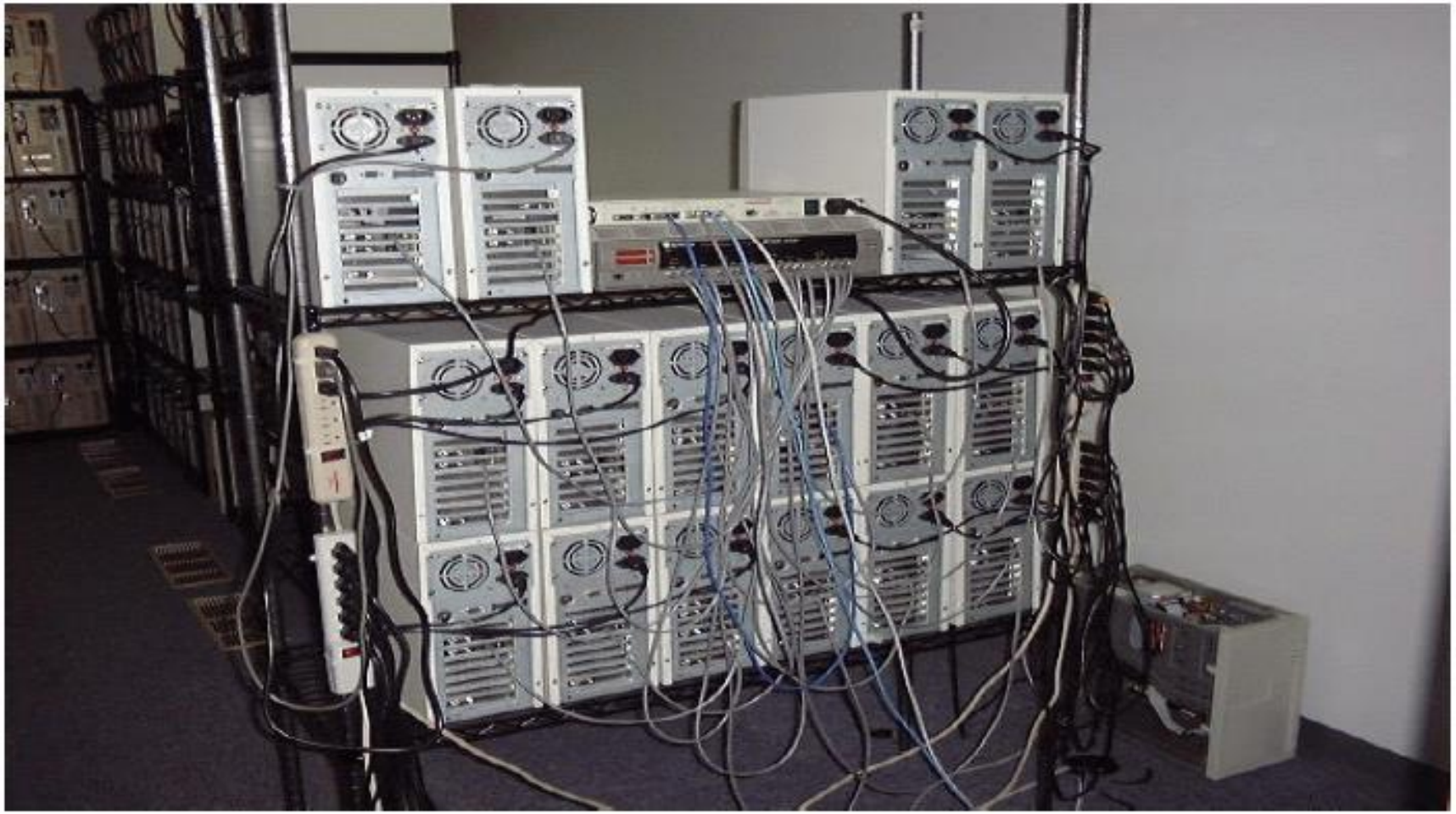
7.1.4 并行计算机发展

- **90 年代中后期：高性能微处理器+高性能网络。体系结构框架趋于统一 (DSM、SMP、NOW)**
  - 以高性能微处理芯片和互连网络通信技术为基础，共享存储对称多处理机（SMP）系统得到了迅速发展
    - SUN Ultra E10000
    - SGI Power Challenge R10000
    - HP C–240
    - DEC Alphaserver 400C

7.1.4 并行计算机发展

- **90 年代中后期：高性能微处理器+高性能网络。体系结构框架趋于统一 (DSM、SMP、NOW)**
  - **NOW（Network of Workstations）工作站机群**
    - 每个结点都是一个完整的工作站，有独立的硬盘与UNIX系统
    - 结点间通过低成本的网络（如千兆以太网）连接
    - 每个结点安装消息传递并行程序设计软件，实现通信、负载平衡等。被大量中小型计算用户和科研院校所采用
  - **也称为COW（Cluster of Workstations）**
  - **NOW（COW）与MPP 之间的界线越来越模糊**
  - **1994年夏，Thomas Sterling和Don Becker用16台PC和以太网组装了一个计算机集群系统，命名为“Beowulf”。**

7.1.4 并行计算机发展



世界上第一台Beowulf 集群，1994年

7.1.4 并行计算机发展

- **90 年代中后期：高性能微处理器+高性能网络。体系结构框架趋于统一 (DSM、SMP、NOW)**
  - **第三代Beowulf集群Loki和Hyglac分别在LANL和加州理工学院建成，两者均采用了16个Pentium Pro处理器和100Mb/s的快速以太网，不同之处在于使用了不同的网络结构。**
  - **两台机器被联合起来用于求解NBody问题，系统总性能达到2GFLOPS，获得了1997年的戈登●贝尔性价比奖**

7.1.4 并行计算机发展

52结点的贝奥武夫机群Borg，在麦吉尔大学被用来搜寻脉冲星双星系统。

7.1.4 并行计算机发展

- **2000年之后：迅猛发展**
  - **Cluster 机群/ 集群**
    - 每个结点含多个商用处理器，结点内部共享存储
    - 采用商用机群交换机通过前端总线连接结点，结点分布存储
    - 各个结点采用Linux 操作系统、GNU编译系统和作业管理系统
  - **Constellation 星群**
    - 每个结点是一台子并行机
    - 采用商用机群交换机通过前端总线连接结点，结点分布存储
    - 各个结点运行专用的结点操作系统、编译系统和作业管理系统
  - **MPP**
    - 专用高性能网络，大多为政府直接支持

7.1.5 中国并行计算机发展

- 1958 年第一台国产计算机诞生——103型计算机

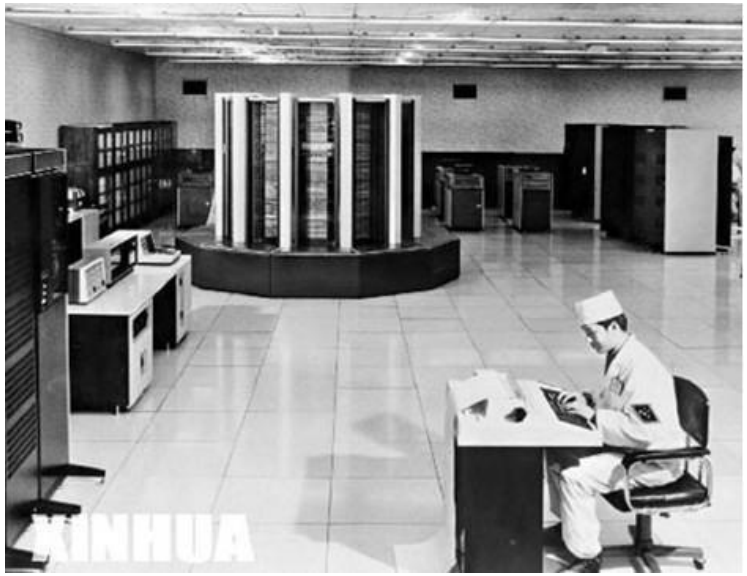
运行速度1500次/每秒

7.1.5 中国并行计算机发展

- 我国的超级计算机系列
  - 曙光：中科院计算所，曙光公司，上海超算中心
  - 深腾：联想集团
  - 银河：国防科大
  - 神威：国家并行计算机工程技术研究中心
  - 天河：国防科大

银河

- 银河一号：1983年12月，我国第一台每秒运算达1亿次以上的计算机



- 1992年，银河-II 问世，每秒运算达10亿次
- 1997年，银河-III 问世，每秒运算达130亿次

曙光

- 1993年10月，国家智能计算机研究开发中心(后成立北京市曙光计算机公司) 研制成功“曙光一号”SMP多处理机
- 2000年推出每秒3000 亿次的曙光3000 超级服务器
- 2004年6月，推出11万亿次的曙光4000A超级计算机，排名全球第十
- 2008年6月，曙光5000A发布，实际速度超160万亿次，排名世界第十

曙光5000A

神威

- 1999年9月，由国家并行计算机工程技术研究中心牵头研制成功的“神威” 计算机系统投入运行，峰值运算速度达每秒3840亿次。

神威采用国产CPU

深腾

- 2002年，联想发布深腾1800计算机，排名全球第43位，成为首家正式进入排行榜前100的中国企业
- 2003年，深腾6800计算机发布，列全世界TOP500第14位，其78.5%的整机效率列世界通用高端计算机第一名
- 2008年12月，联想发布百万亿次超级计算机深腾7000，排名全球19

实现了大规模异构体系结构、系统均衡设计方法、高效能结点机、大规模无局部盘结点及千核级应用技术突破。整机系统软件实现对大规模异构资源的统一管理和高效使用。

深腾7000

天河

- 2009年10，中国首台千万亿次超级计算机“天河一号”诞生，使中国成为继美国之后世界上第二个能够研制千万亿次超级计算机的国家。
- 2010年10月，升级后的天河一号排名世界第一
- 2013年6月，天河二号再次问鼎世界第一，功耗达24兆瓦，也是目前TOP500里功耗最大的

学习内容

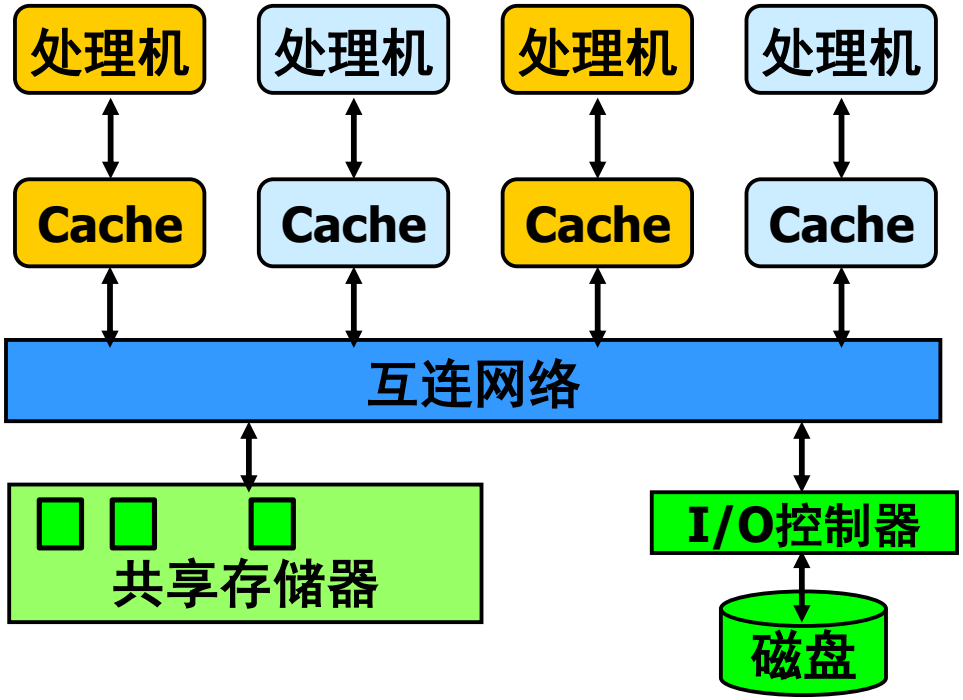
- 7.1 多处理机概念
- 7.2 多处理机结构
- 7.3 多核处理器
- 7.4 多处理机的多Cache一致性
- 7.5 多处理机的机间互连形式
- 7.6 程序并行性
- 7.6 多处理机性能
- 7.7 多处理机操作系统

7.2 多处理机结构

- 从存储器的分布和使用上看，多处理机系统分为两种结构：
  - 共享存储器结构
  - 分布式存储器结构

7.2.1 共享存储器结构

- 各处理机通过互连网络**共享存储器和I/O设备**，并通过共享存储器相互**联系**。



共享存储器结构的多处理机系统

7.2.1 共享存储器结构

- 特点：
  - 各处理机共享存储器，并通过对共享存储器读/写实现相互通信；
  - 对存储单元的任何修改对其他处理机都是可见的；
  - 延迟低，但扩展性差；

共享数据进入Cache产生了一个新的问题：Cache的一致性问题

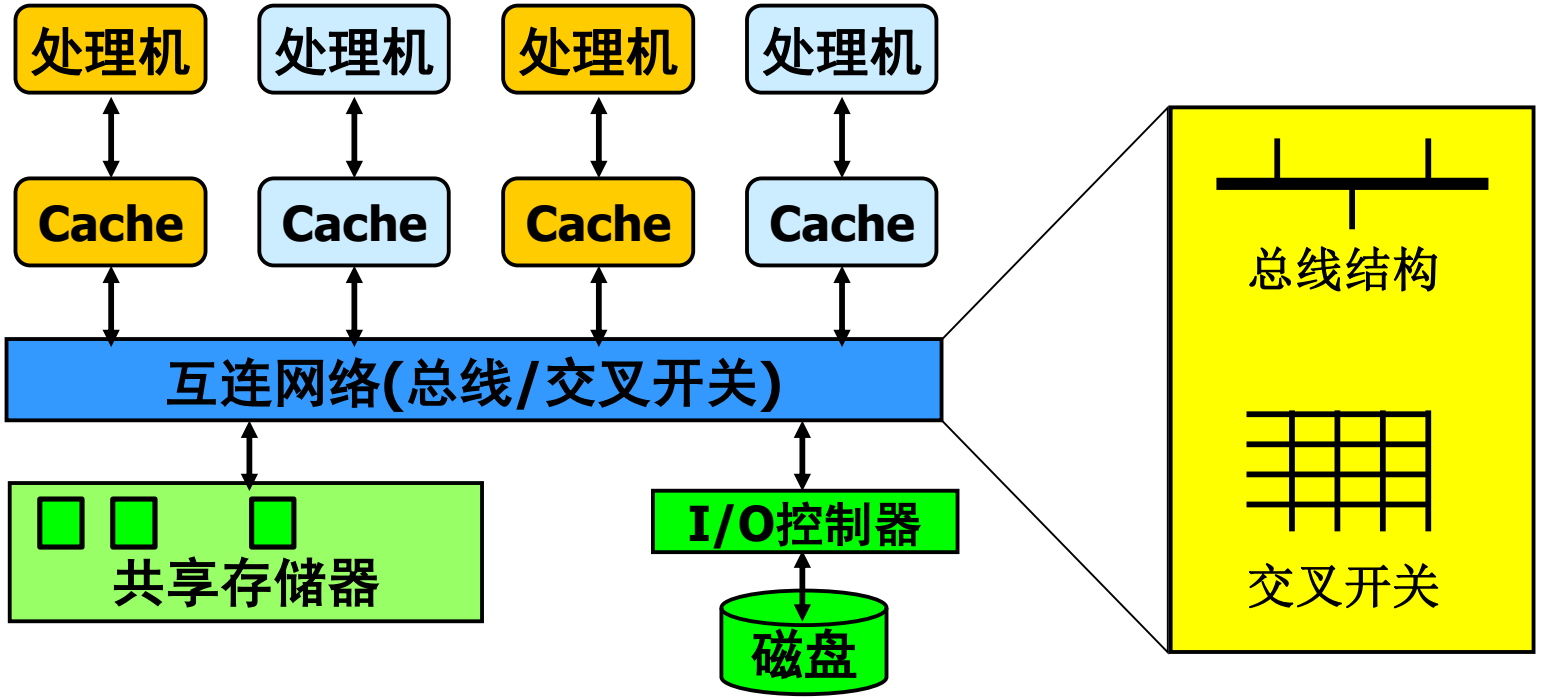
7.2.1 共享存储器结构

- 根据共享存储器实现方式不同又细分为：
  - UMA结构
  - NUMA结构
    - ccNUMA
    - NCC-NUMA
  - COMA结构

1. UMA结构

- UMA = Uniform Memory Access
- 均衡存储器访问结构
- 或 集中式共享存储器 (Centralized Shared-Memory) 结构
- 特点：
  - 各处理机对存储器的访问时间、访问功能相同
- 这种结构的处理机称为对称多处理机 (SMP = Symmetric Multiprocessors)

1. UMA结构



UMA结构的多处理机系统

- 互连网络可以是总线、交叉开关或多级交换网络。
- 大多数的对称多处理机采用总线连接。

1. UMA结构

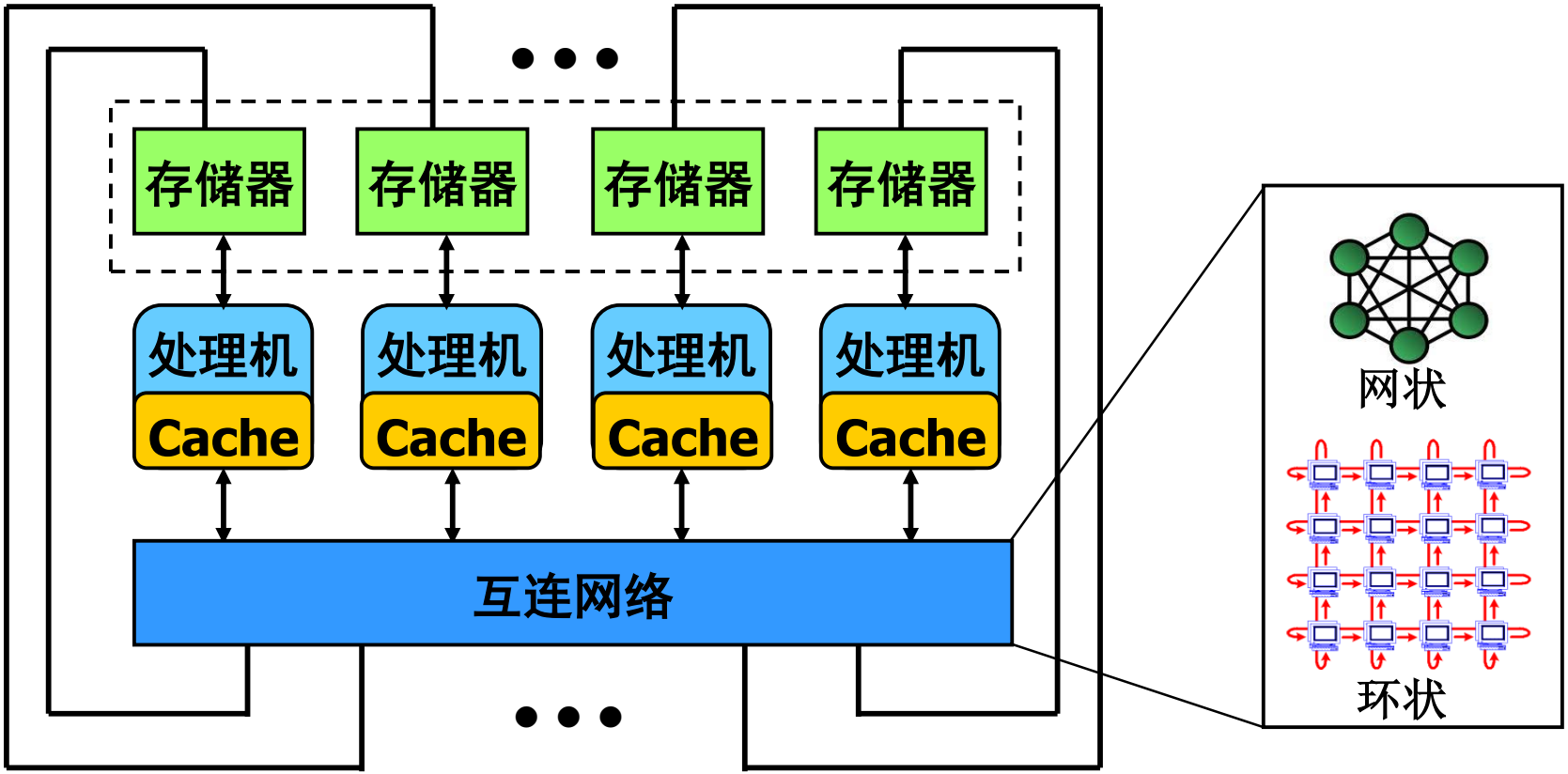
- **优点：**
  - （1）性能提高
  - （2）高可用性
  - （3）增量式增长
  - （4）可扩展性好
  - （5）透明
- **缺点：**
  - 所有处理机对共享存储器的访问都要经过互连网络，当规模较大时，访问延迟较大。因此通常增设大容量本地Cache
- 适合小规模时采用

2. NUMA结构

- NUMA = Non Uniform Memory Access
- 非均衡存储器访问结构
- 也称为分布式共享存储器 (DSM=Distributed Shared-Memory) 结构
- 特点：
  - 分布于各个处理机的存储器被统一编址，可由所有处理机共享；
  - 根据存储器位置的不同，各处理机对存储器的访问时间不相等。处理机访问本地存储器的速度较快，通过互连网络访问其他处理机上的远地存储器相对较慢。

2. NUMA结构

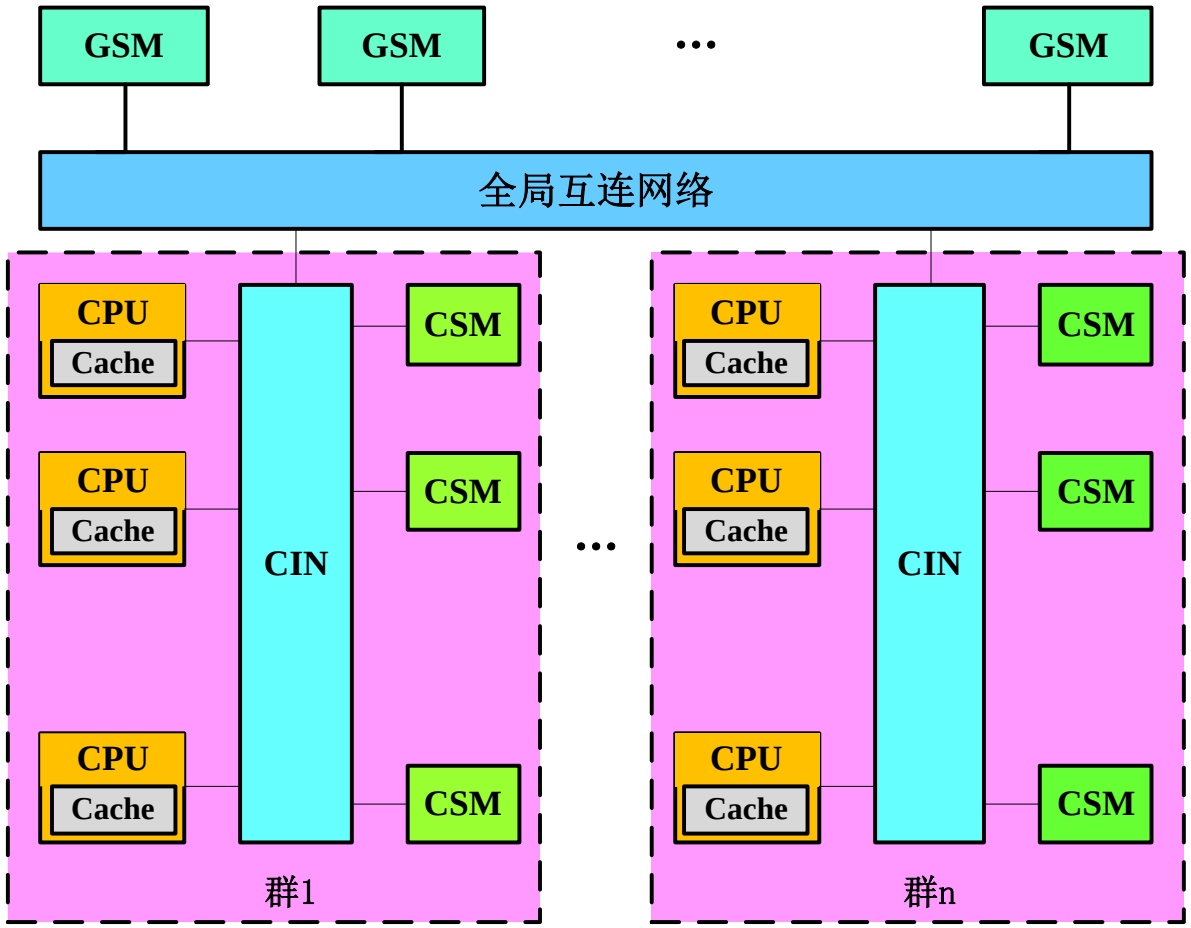
- ☑ 各处理机拥有自己的本地存储器，可以独立工作；
- ☑ 各处理机借助互连网络、通过消息传递机制相互通信；



NUMA结构的多处理机系统 – 共享本地存储器

2. NUMA结构

允许各处理机拥有自己独立结构，甚至可以是SMP。



NUMA结构的多处理机系统 – 层次式机群模型

2. NUMA结构

- 优点：
  - 比SMP扩展性好，并行程度更高，性能更好；
  - 采用与SMP相同的编程模型，为SMP编写的程序仅需少量修改即可移植运行；
  - 每个处理机都可以访问较大的存储空间，因此可以更高效地运行大程序；
  - 实现数据共享时不需要移动数据；
  - 传递包含指针的数据结构比较容易；
  - 系统构建成本较低，利用成熟技术搭建系统。

2. NUMA结构

- **缺点：**
  - 如果过多地访问远程存储器，则性能会下降；
  - 对存储器的访问不透明，需要处理分页（例如哪个页面在哪个存储器中）、进程分配等，对软件设计要求较高

3. ccNUMA

- ccNUMA = Cache-coherent Non Uniform Memory Access
- 高速缓存一致性非均匀存储访问
- 在NUMA多处理机中，逻辑上共享的存储器在物理上是分布的。如果各处理机Cache内容一致，则将这种NUMA称为ccNUMA

3. ccNUMA

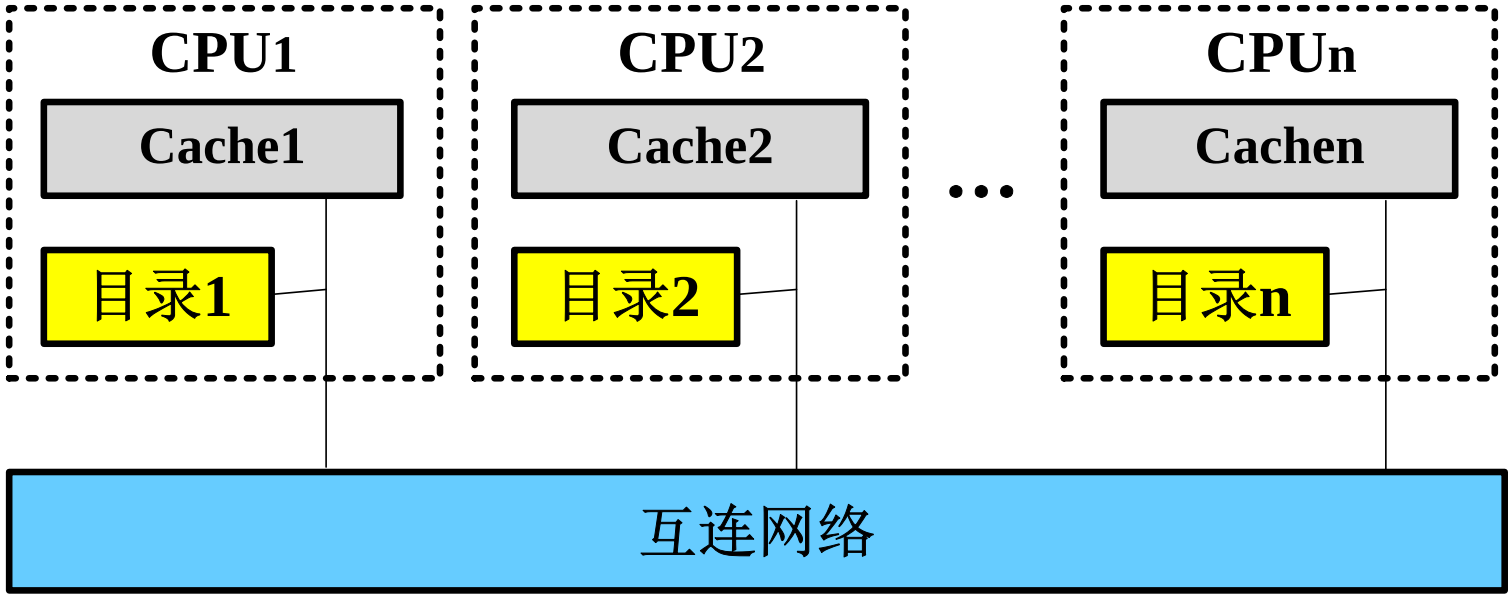
- ccNUMA注重开拓数据的局部性和增强系统的可扩展性。在实际应用中，大多数的数据访问都可在本结点内完成，网络上传输的主要是高速缓存无效性信息而不是数据。
- ccNUMA和COMA的共同特点是它们都对高速缓存一致性提供硬件支持，而在NCC-NUMA (Non-Cache Coherent Non-Uniform Memory Access) 中，则没有对高速缓存的一致性提供硬件支持。
- 绝大多数商用ccNUMA多处理机系统使用基于目录的高速缓存一致性协议。

4. COMA

- COMA = Cache-Only Memory Architecture
- 仅用高速缓存存储器结构
- NUMA的一个特例，只是将NUMA中的分布存储器换成了Cache
  - 各处理机结点上没有主存储器，没有存储层次结构，仅有Cache
  - 所有的高速缓存构成了全局地址空间，全部Cache组成了全局虚拟地址空间
  - 对远程Cache的访问通过分布式Cache目录进行

4. COMA

- 各处理机结点上没有主存储器，没有存储层次结构，仅有Cache
- 所有的高速缓存构成了全局地址空间，全部Cache组成了全局虚拟地址空间
- 对远程Cache的访问通过分布式Cache目录进行



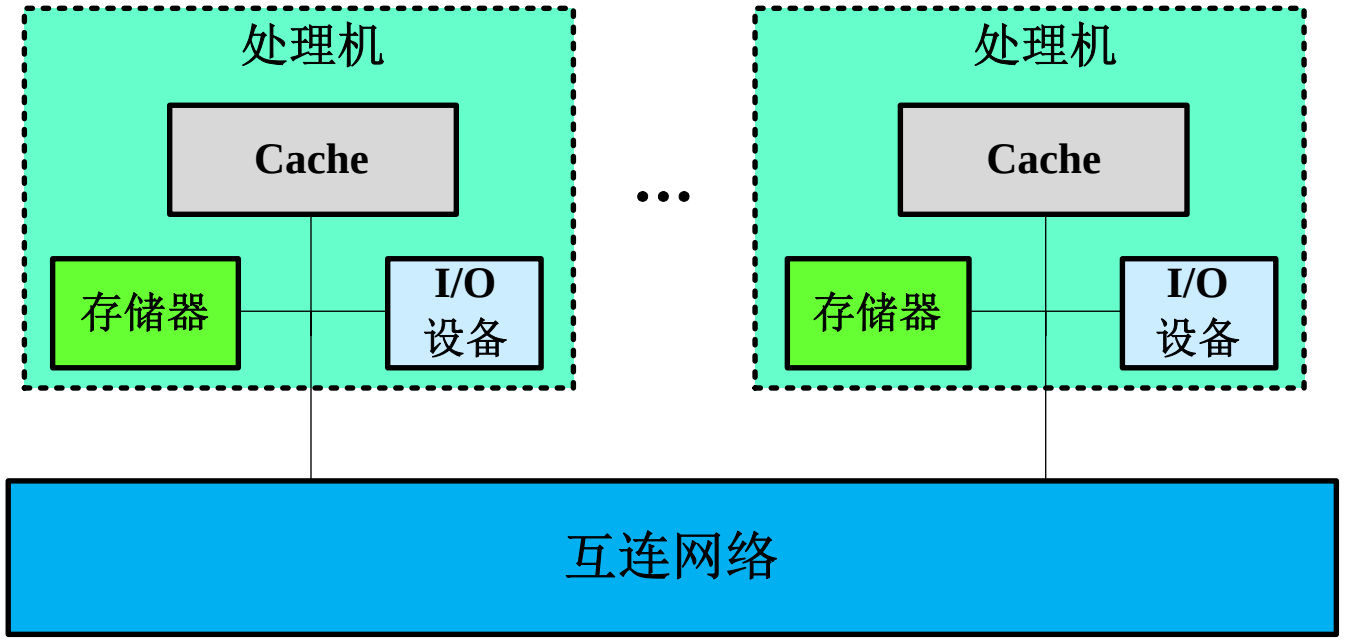
COMA多处理机结构

7.2.2 分布式存储器结构

- 也被称为非远程存储访问 (NORMA，No-Remote Memory Access) 模型
- 各处理机拥有自己的本地存储器，在本地操作系统控制下独立工作。
- 各处理机的本地存储器是私有的，不能被其他处理机访问。
- 各处理机借助互连网络、通过消息传递机制相互通信，实现数据共享。
- 大规模并行处理机（MPP）、机群（cluster）等采用了这种结构。

7.2.2 分布式存储器结构

- 各处理机借助互连网络、通过消息传递机制相互通信，实现数据共享

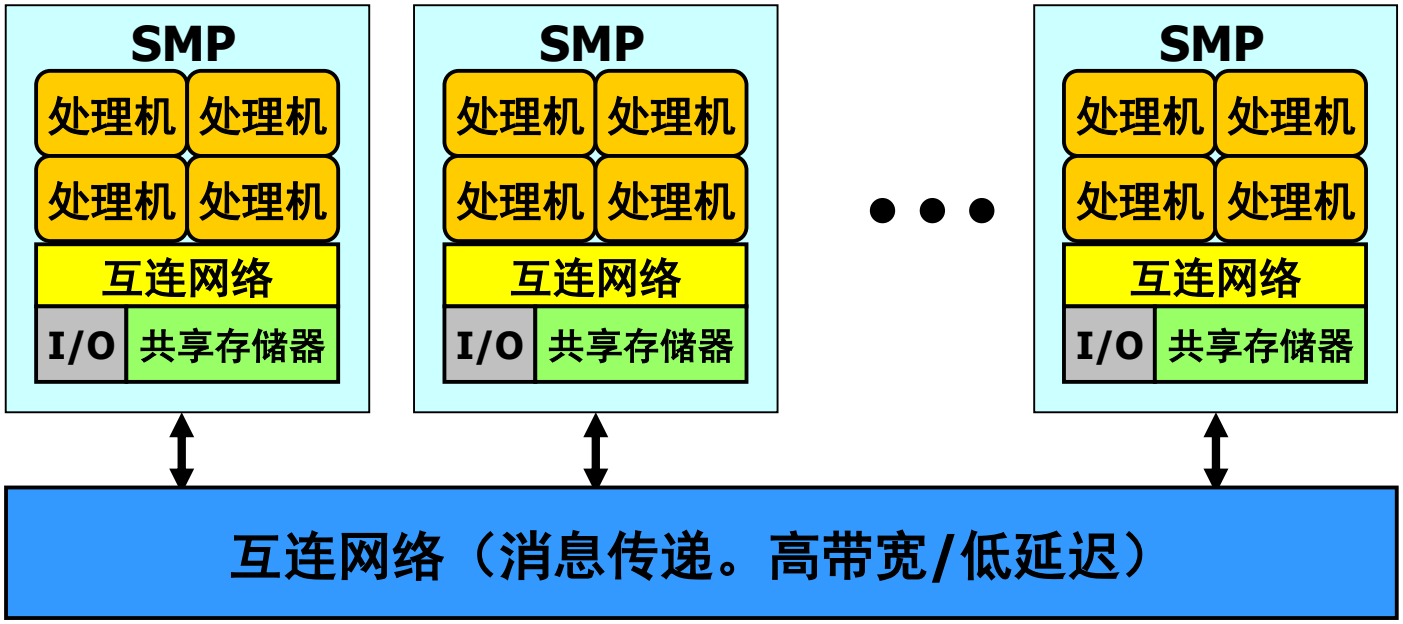


分布式存储器多处理机

7.2.2 分布式存储器结构

- **优点：**
  - 结构灵活，扩展性较好
- **缺点：**
  - 任务传输以及任务分配算法复杂，通常要设计专有算法
  - 处理机之间的访问延迟较大
  - 需要高带宽的互连

分布-共享存储器结构



混合型 分布—共享存储器结构 的多处理机系统

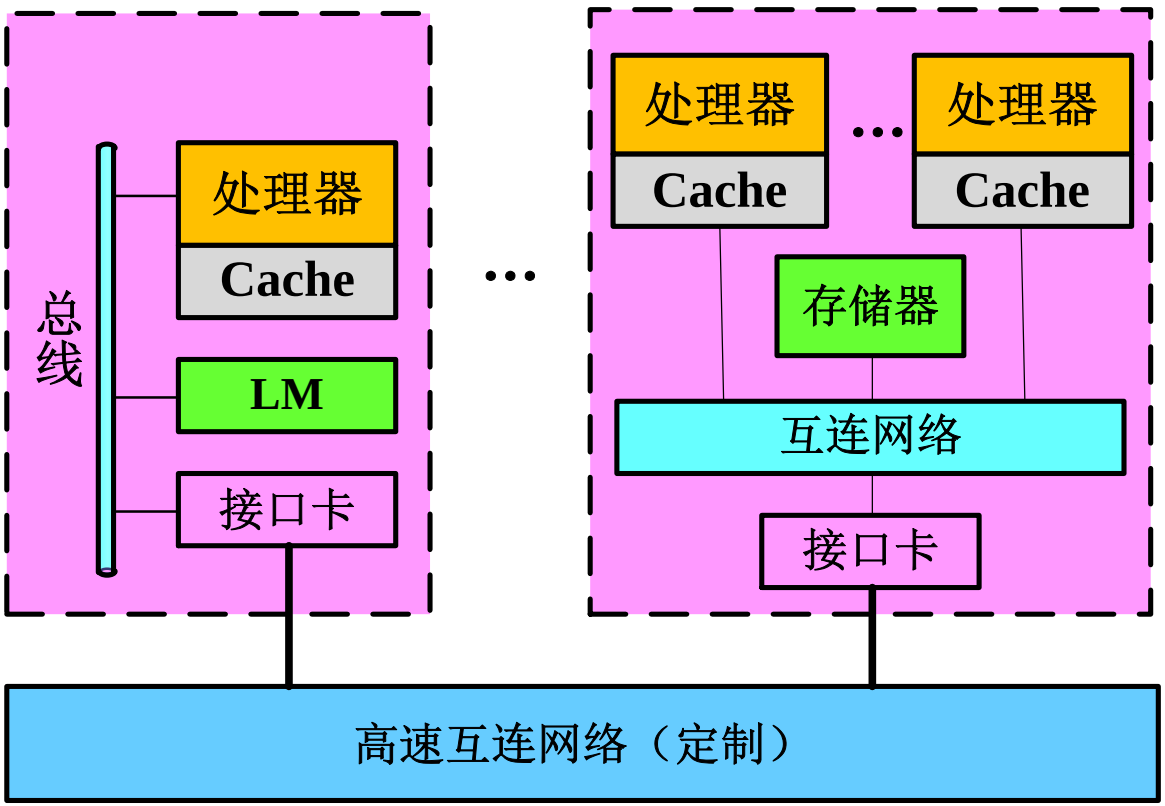
- ☑ 每个节点为共享存储器结构的SMP系统；
- ☑ 各SMP系统借助互连网络、通过消息传递机制相互通信；
- ☑ 是目前大规模并行处理（MPP）等系统普遍采用的结构。

7.2.3 大规模并行处理机

- MPP = Massively Parallel Processor
- 由几百或几千台高性能、低成本处理机组成的大规模并行计算机系统
  - 每个处理机都有自己的私有资源，如内存、网络接口等
  - 每个处理机能直接访问的只有本地存储器，但不能直接访问其它处理机的存储器
  - 处理机之间以定制的高带宽、低延迟的高速互连网络互连

7.2.3 大规模并行处理机

- MPP系统大多采用分布式存储结构。所有存储器在物理上是分布的，而且都是私有的。每个处理机能直接访问的只有本地存储器，不能直接访问其它处理机的存储器。



MPP结构

7.2.3 大规模并行处理机

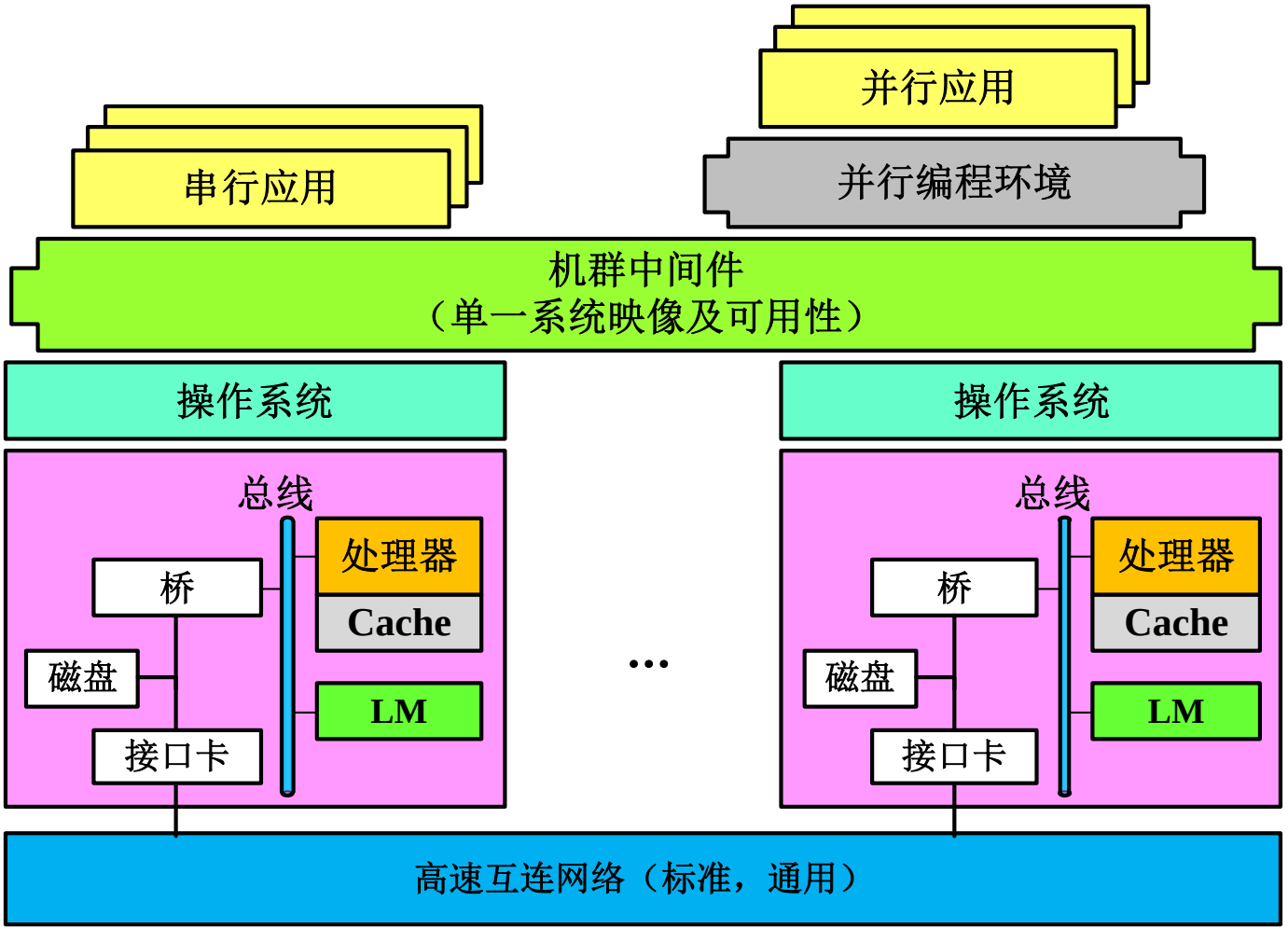
- 特点：
  - 处理结点采用商用处理机
  - 系统中有物理上的分布式存储器
  - 采用高通信带宽和低延迟的互连网络（专门设计和定制的）
  - 能扩放至成百上千乃至上万个处理机
  - 它是一种异步的MIMD机器
    - 程序系由多个进程组成，每个都有其私有地址空间，进程间采用传递消息相互作用

7.2.4 机群

- 也被称为集群
- COW = Cluster of Workstations
- NOW = Network of Workstations
- 通过一组松散耦合的计算机软件和/或硬件连接起来、高度紧密地协作完成计算工作的计算机系统
  - 结点：单独运行的商品化计算机
  - 网络：高速通用网络，如局域网或互连网络连接
  - 操作系统：并行编程环境，系统资源管理和相互协作
  - 集群中间件：屏蔽差异，在异构系统上提供统一运行环境和服务

7.2.4 机群

从结构和结点之间的通信方式来看，机群属于非均匀存储访问的MIMD型的分布式存储并行计算机

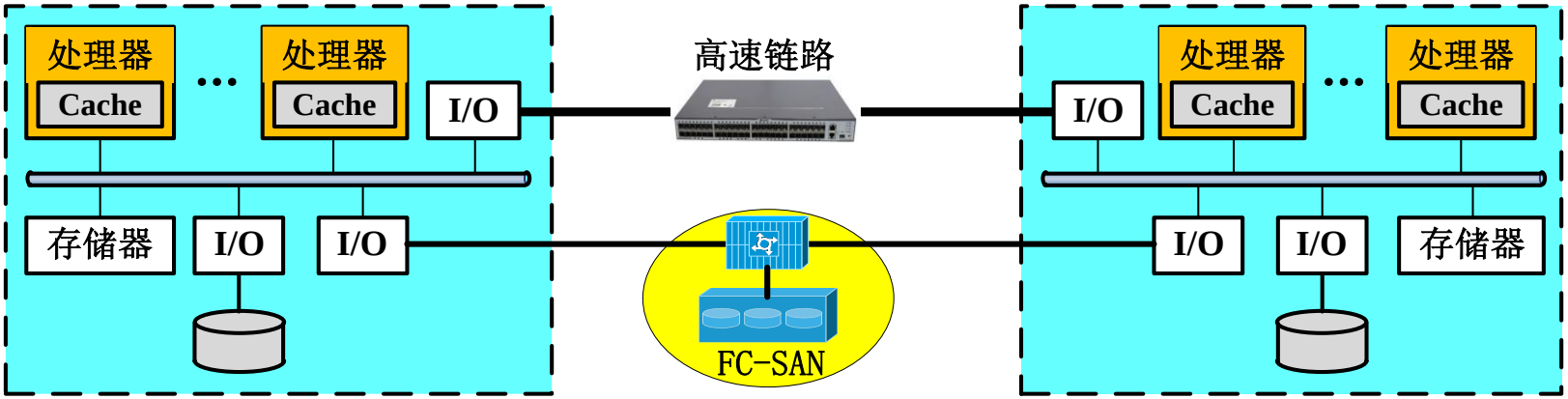


机群系统结构

结点间连接模型

两种连接模型：无共享（零共享）模型和共享磁盘模型

无共享（零共享）模型



共享磁盘模型

机群分类

高可用性机群 High-Availability Clusters

- 关键任务应用
- 提供冗余
- 避免单点故障

负载均衡机群 Load Balancing Clusters

- 将负载分发到可用结点
- 减轻处理负载
- 获得高性能和高可用性

高性能计算机群 High-Performance Clusters

- 将计算任务分配到不同计算结点
- 提高计算能力
- 高吞吐计算
- 分布式计算
- Beowulf

机群特点

- 每一个结点都是一个完整的工作站。一个结点也可以是一台PC或SMP；
- 各结点通过某种低成本的商品（标准）网络互连；
- 各结点内总是有本地磁盘；
- 结点内的网络接口松散耦合到I/O总线上；
- 每个结点中驻留有完整的操作系统

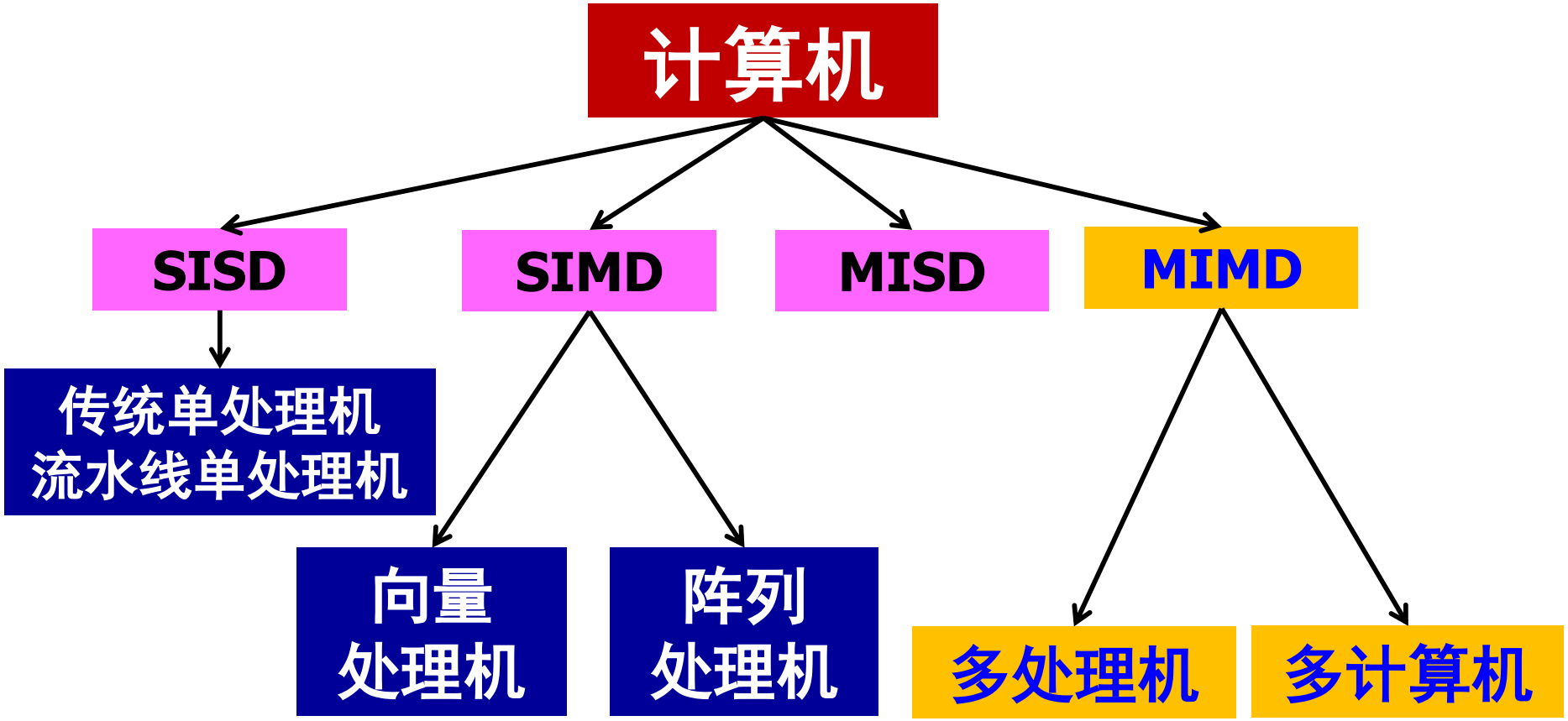
机群关键技术

- 高效的通信系统；
  - 如交换式千兆位以太网，万兆位以太网
- 并行程序设计环境；
  - 如 PVM， MPI， OpenMP，HPF，Express等
- 并行程序设计语言
  - 如 FORTRAN、C和C++
- 全局资源管理与利用
  - 操作系统，COW管理，作业管理 等

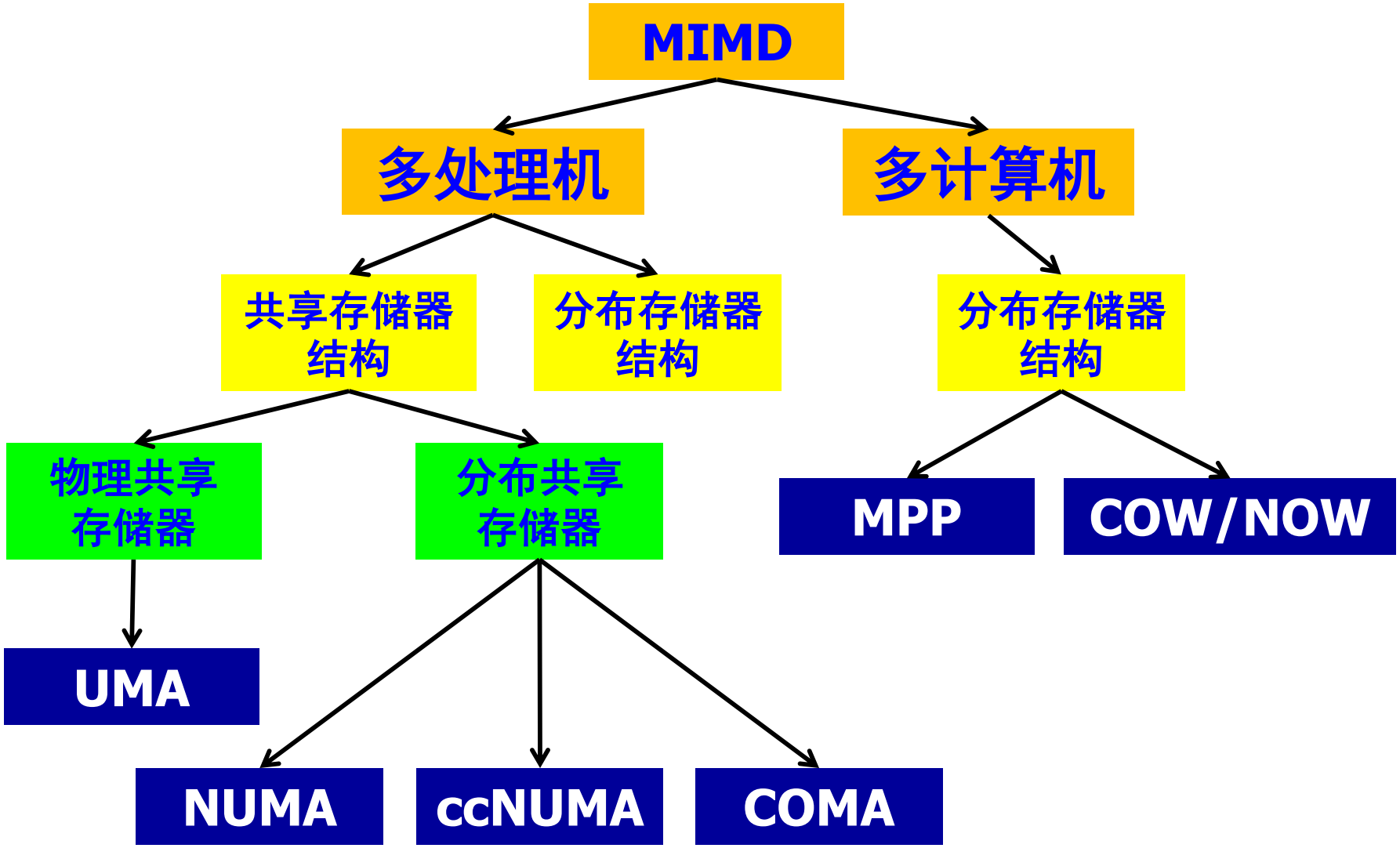
PVM、SMP、DSM、MPP和COW的比较

属性	PVP	SMP	DSM	MPP	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(以太网、ATM)
通信机制	共享变量	共享变量	共享变量	消息传递	消息传递
地址空间	单地址空间	单地址空间	单地址空间	多地址空间	多地址空间
系统存储器	集中共享	集中共享	分布共享	分布非共享	分布非共享
访存模型	UMA	UMA	NUMA	NORMA	NORMA
代表机器	Cray C-90 Cray T-90 银河1号	IBM R50 SGI Power Challenge 曙光1号	Stanford DASH Cray T 3D	Inter Paragon IBM Option White 曙光-1000/2000	Berkeley NOW Alpha Farm IBM SP2 曙光3000/4000

弗林分类法 – 并行体系结构



MIMD计算机分类


MIMD
多处理机
多计算机
共享存储器结构
分布存储器结构
分布存储器结构
物理共享存储器
分布共享存储器
MPP
COW/NOW
UMA
NUMA
ccNUMA
COMA

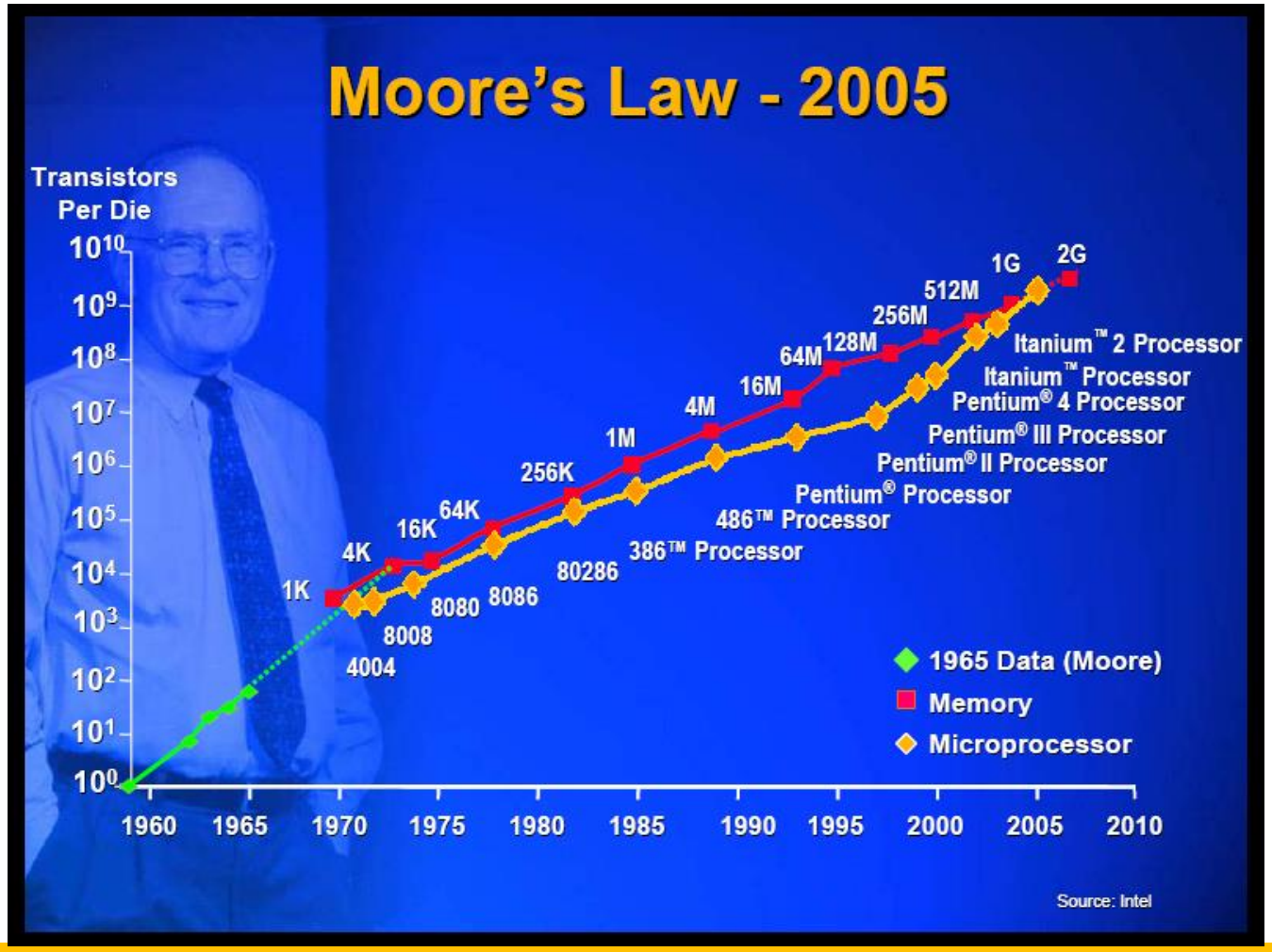
学习内容

- 7.1 多处理机概念
- 7.2 多处理机结构
- 7.3 多核处理器
- 7.4 多处理机的多Cache一致性
- 7.5 多处理机的机间互连形式
- 7.6 程序并行性
- 7.6 多处理机性能
- 7.7 多处理机操作系统

7.3 多核处理器

- Multi-core Processor
- 片上多处理机 (CMP，Chip Multi-Processors)
- 片上多处理机系统 (MPSoC，Multiprocessor System-on-Chip）
- 是多处理机的一种特殊形式：SMP on a single chip
- 1996年，美国斯坦福大学首次提出了片上多处理器(CMP)思想
- 2000年，IBM于发布了世界上第一个双核处理器POWER4
- 2005年，Intel和AMD多核处理器开始大规模应用

微处理器发展



Transistor capacity doubles every 18 months

微处理器发展

- 问题：晶体管数量 ≠ 能力
- 如何利用晶体管资源？
  - 更高的时钟频率
  - 更好的核
    - 超流水，超标量，…
    - 更好的向量处理（SIMD）
    - 问题：存储器墙 = 存储器速度不能满足CPU速度要求
  - 更大的Cache
    - 改善访存速度

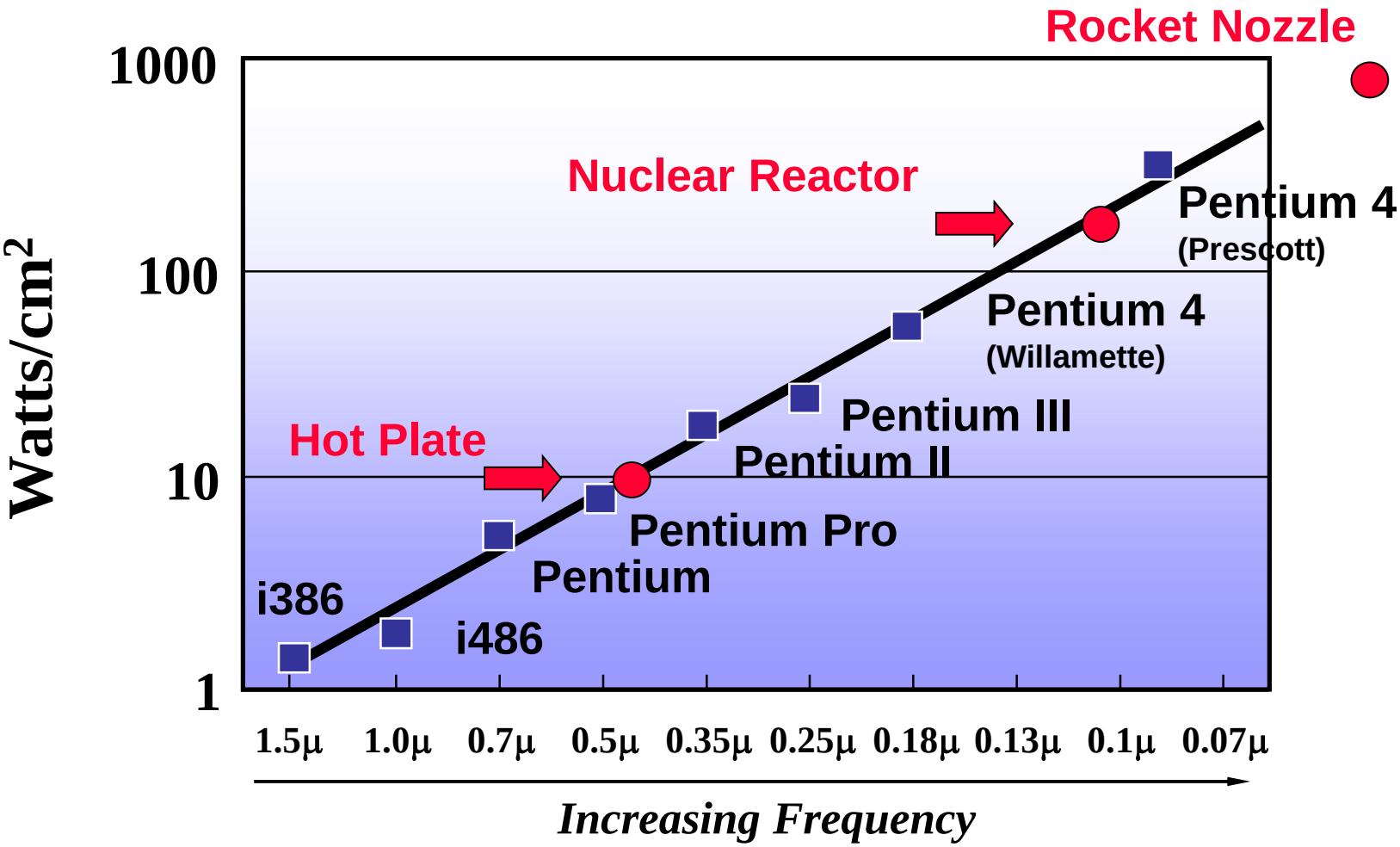
微处理器发展

- **更高的时钟频率**
  - 增加了功耗
    - 功耗与频率和电压的平方(U^2)成正比
    - 更高的频率需要更高的电压
    - 问题：漏电流导致的能量损失
  - 增加了散热和制冷需求
  - 限制了芯片面积 (光速)

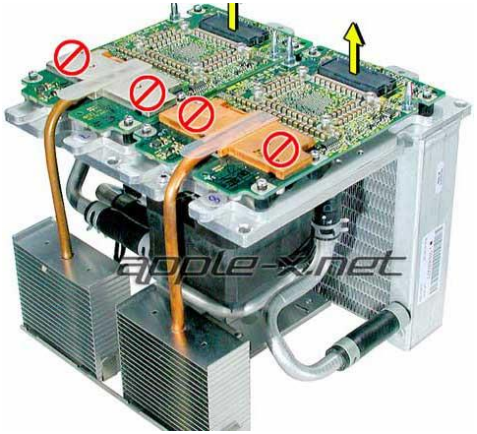
100 GHz limits chip to 3 mm (speed of light 300mm/ns)

微处理器发展

耗散功率 (Dissipated Power) ~ CV^2f



Managing the Heat Load



Liquid cooling system in Apple G5s

Heat sinks in 6XX series Pentium 4s

微处理器发展

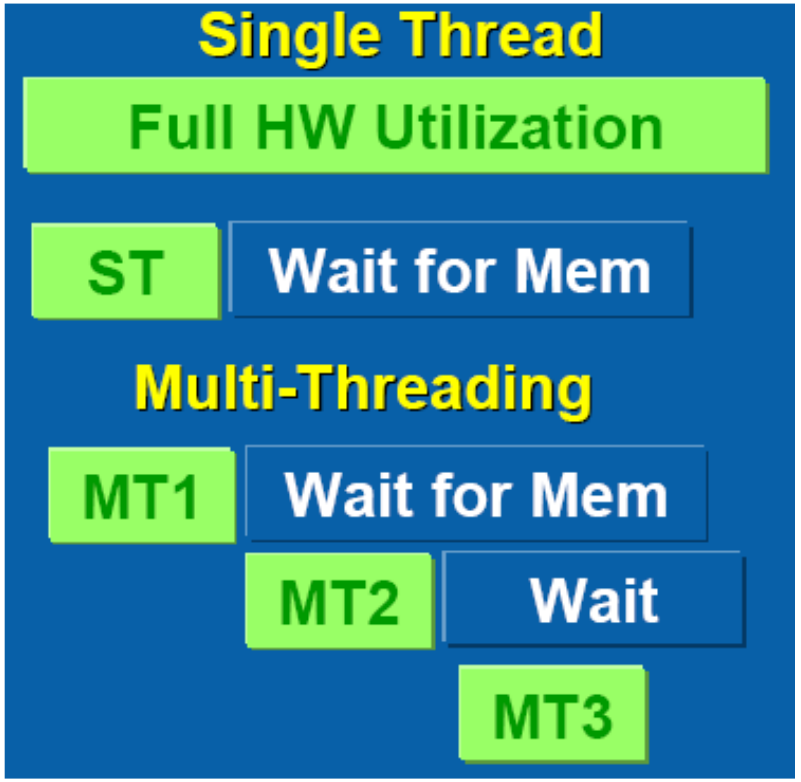
- 更多/更高的并行性
  - 增加位宽度
  - 指令级并行 (ILP)
    - 开发指令之间的并行性
    - 受限于数据、指令、控制相关
    - 通过预测可以改善
  - 线程级并行 (TLP)
    - 硬件线程 (例如 SMT：超线程)
    - 多核

进程与线程

- **进程定义：资源分配单位**
  - 一个独立的进程空间，可装入进程映像；
  - 进程关联的执行文件；
  - 进程所用系统其它资源(如设备、文件等）；
  - 一个或多个线程。 进程在创建时一般同时创建好第一个线程，其它线程按需要由用户程序请求创建。
- **线程定义：CPU分配单位（执行单位）**

TLP

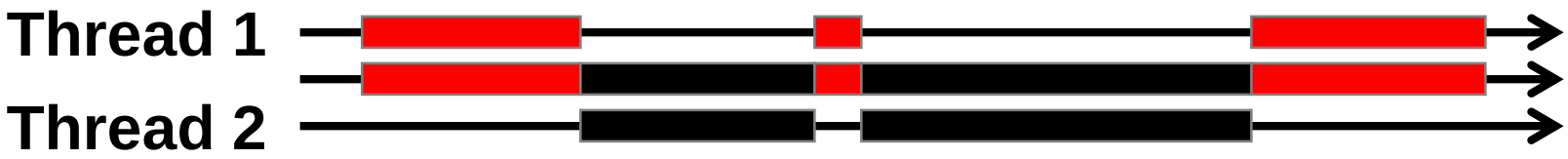
- **TLP（线程级并行）相关技术**
  - 同时多线程 (SMT = Simultaneous Multi-threading)
    - 例如: Intel 超线程
  - 芯片多处理(CMP = Chip multiprocessing) → Multi-Core Processor



Understanding SMT and CMP

Make clear Concurrency vs. Parallelism

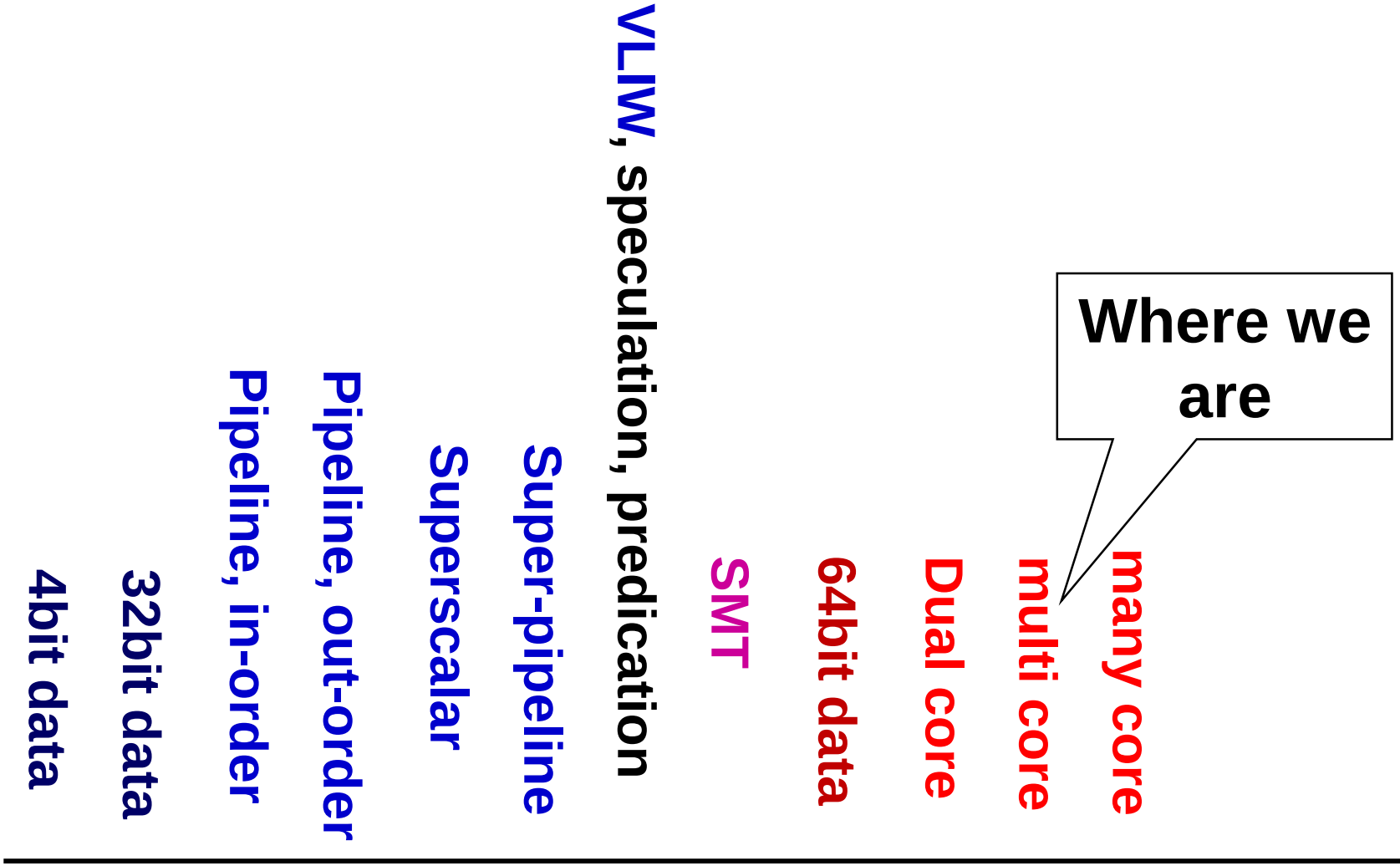
- **Concurrency:** two or more threads are in progress at the same time:



- **Parallelism:** two or more threads are executing at the same time

Multiple cores needed

A brief history of micro-architecture evolution



7.3.1 多核处理器定义与特点

- **多核处理器：**
  - 一枚处理器中集成了两个或多个独立处理单元（称为核）的处理器；
  - 每个核由一个独立处理器的所有组件所组成，可以独立运行程序指令（多指令），可以访问存储器的不同部分（多数据）。
- 只有两个核的处理器，称为双核处理器（dual-core processor）。

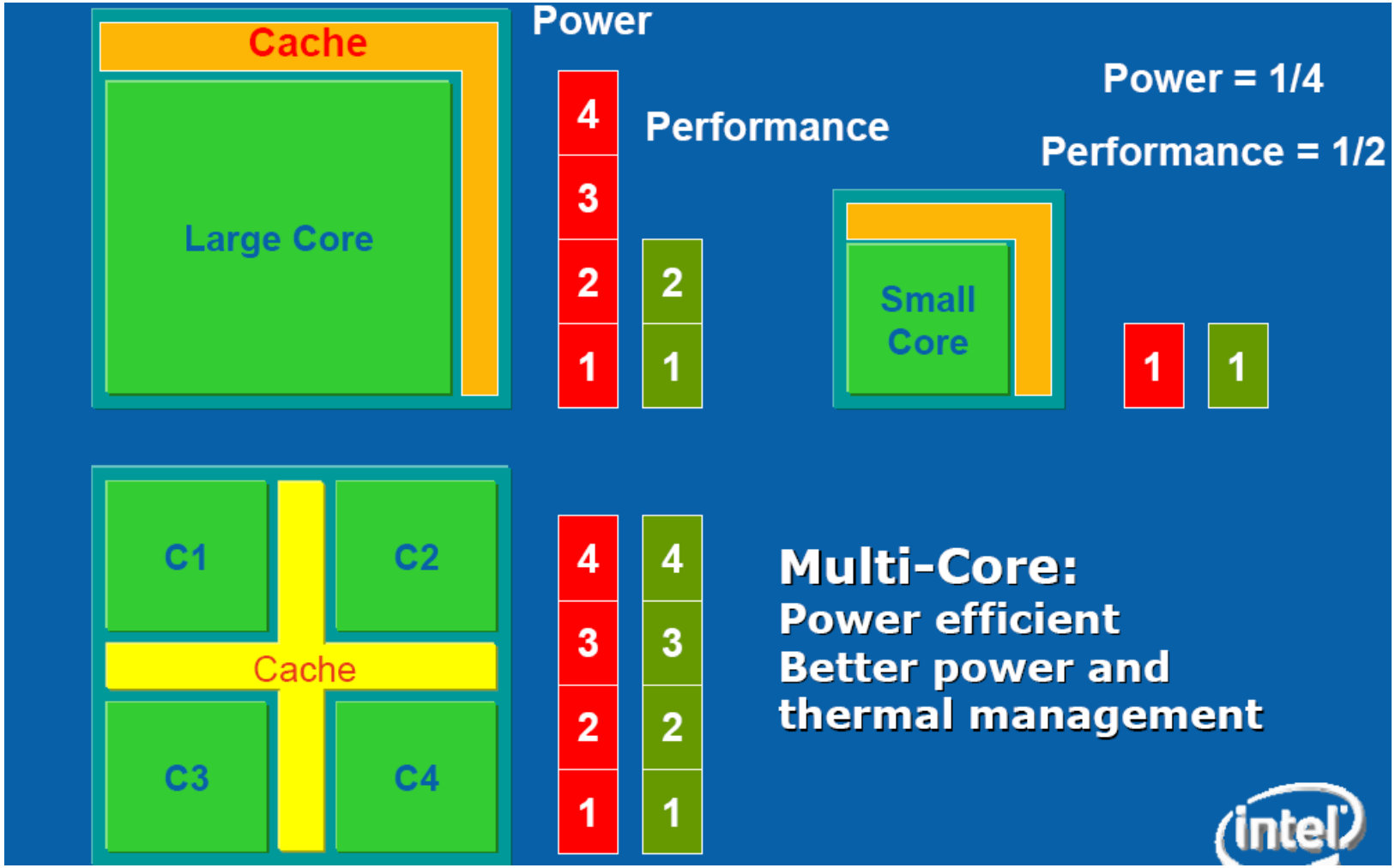
7.3.1 多核处理器定义与特点

- **众核处理器：**
  - 当一枚处理器集成的核的数量达到十几、几十、或几百、几千时，多核变为**众核**。
  - 相比多核：
    - 核的数量多
    - 核经过专门设计

7.3.1 多核处理器定义与结构

- **优点：**
  - 提高吞吐率和并行程序的速度
  - 可以实现核的紧耦合
    - 更好地实现核之间的通信（相比 SMP）
    - 共享Cache
  - 降低功耗
    - 降低时钟频率
    - 可以挂起空闲的核
- **缺点：**
  - 仅能提高并行程序的速度
  - 扩大了与存储器速度的差距

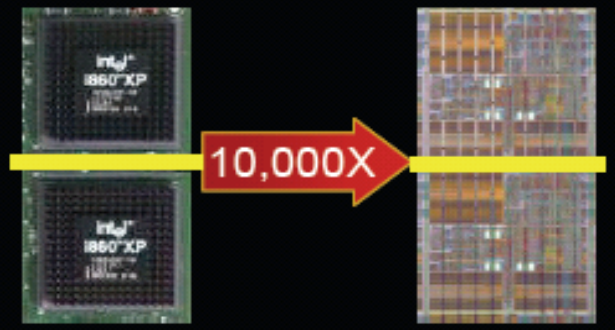
7.3.1 多核处理器定义与结构



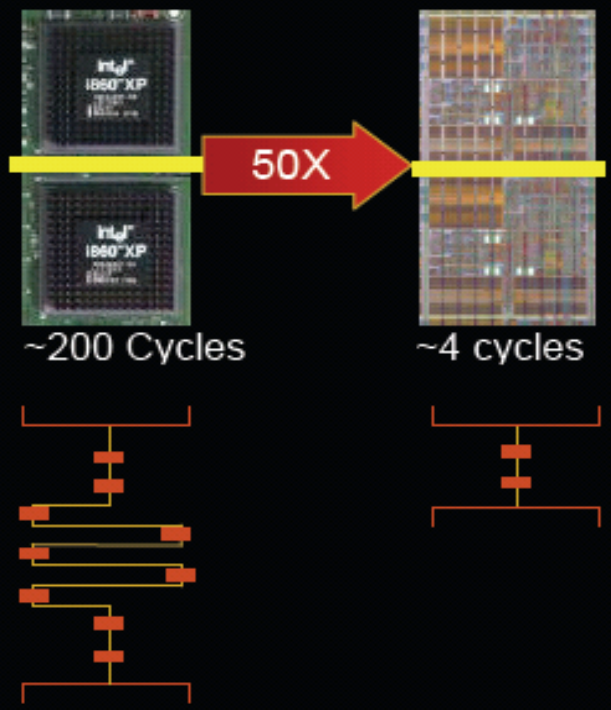
New Target for Micro-architecture – high performance/power

Changes from Multiprocessors to Integrated Multicores

- Communication Bandwidth



- Communication Latency



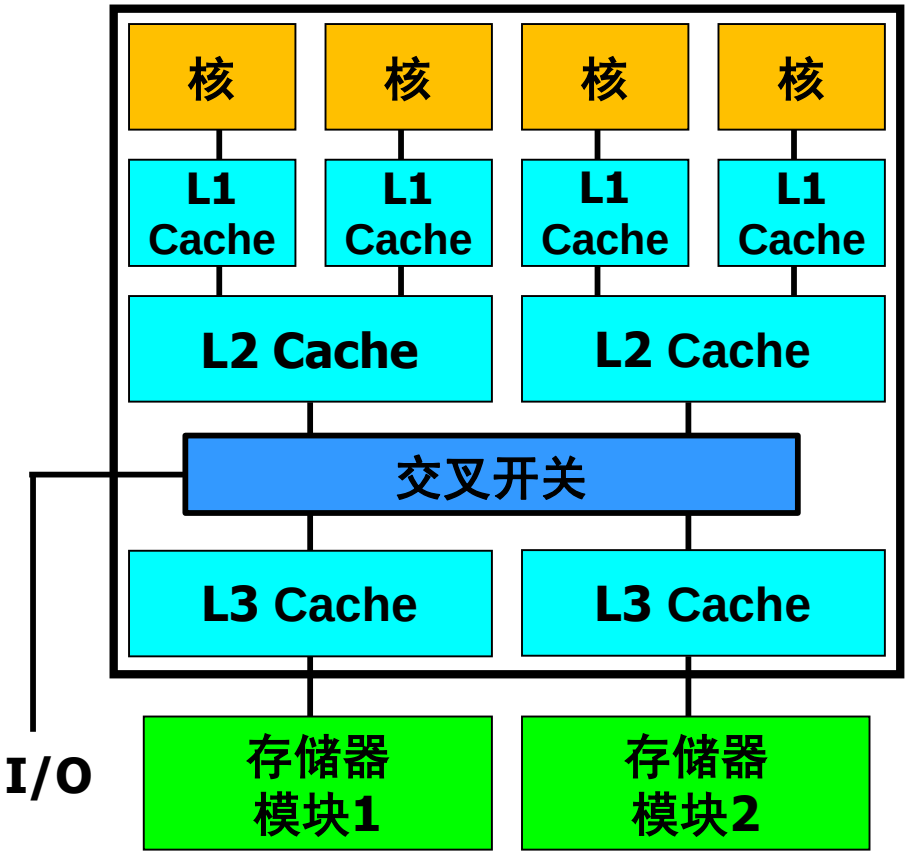
- Where is the Impact?
  - Data Locality
- Where is the Impact?
  - Granularity of Parallelism

7.3.1 多核处理器定义与结构

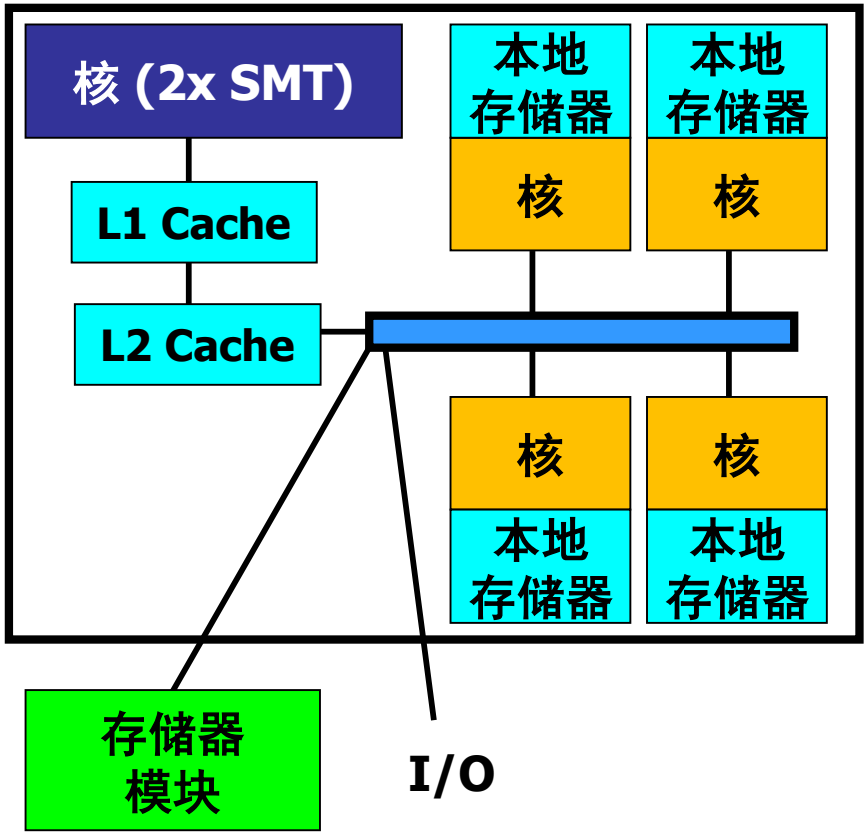
多核处理器结构设计主要考虑以下因素(1/2)：

- (1) 同构还是异构
  - 同构 CMP：集成多个相同的处理器核，同一个任务可以分配给任意一个核处理，简化了任务分配。
  - 异构 CMP：包含了不同结构的处理器核，用不同类型的处理器核处理不同的任务，是异构体系结构处理器的优势所在。
- (2) 核的数量
  - 双核，四核，八核，…

7.3.1 多核处理器定义与结构



带共享Cache和交叉开关的同构多核处理器结构



带Cache、本地存储器和环形总线的异构多核处理器结构

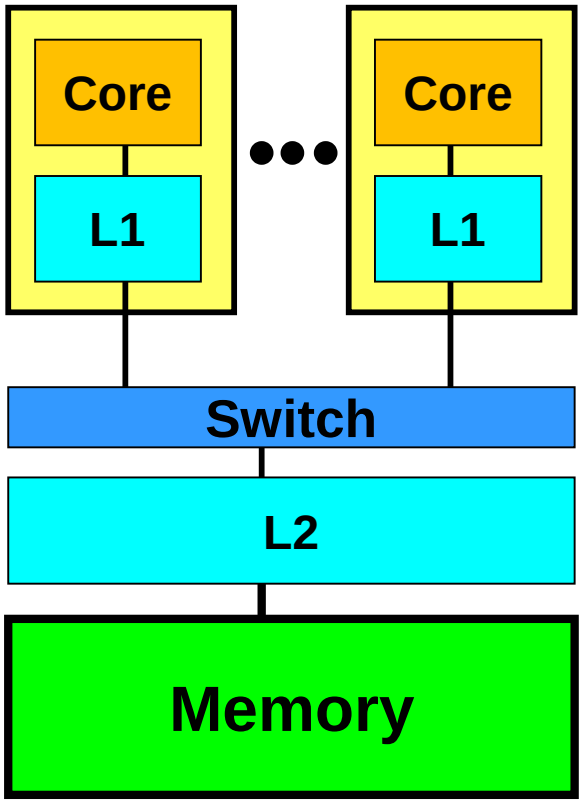
目前，商用处理器大多以同构多核处理器为主

7.3.1 多核处理器定义与结构

多核处理器结构设计主要考虑以下因素(2/2)：

- (3) Cache的设置与访问
  - 分布式存储器结构
  - 共享存储器结构，共享访问的Cache多大
  - Cache层次
- (4) 核间通信技术
  - 通过基于总线共享的Cache
  - 通过片上互连网络

7.3.1 多核处理器定义与结构

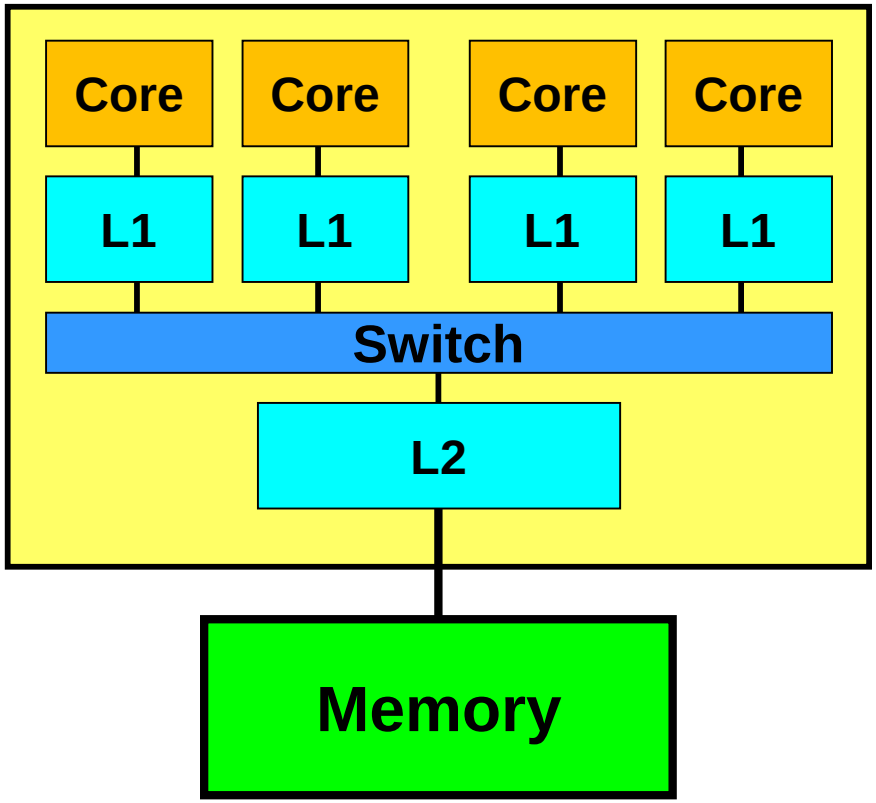


传统设计

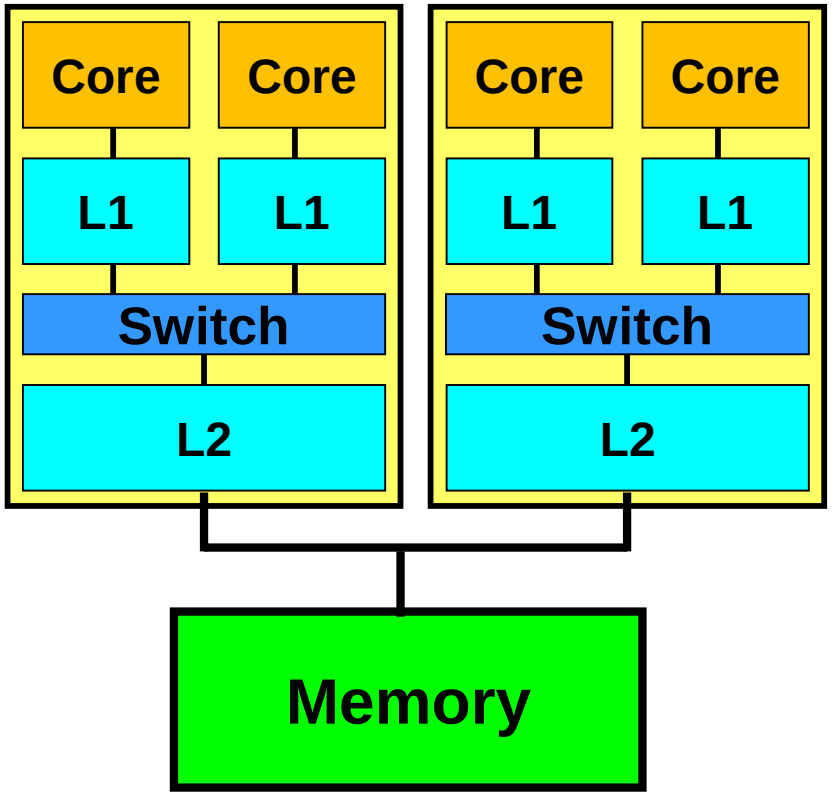
多个单核

L1 Cache私有，共享片外 L2 Cache

7.3.1 多核处理器定义与结构

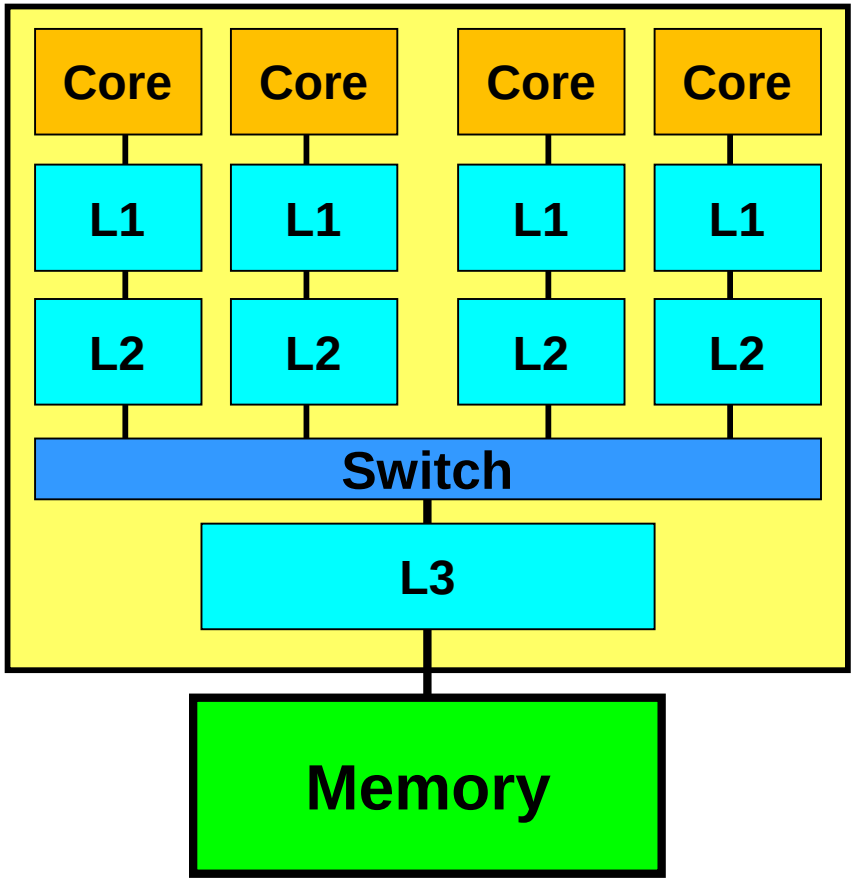


多核
L1 Cache私有
共享片内L2 Cache

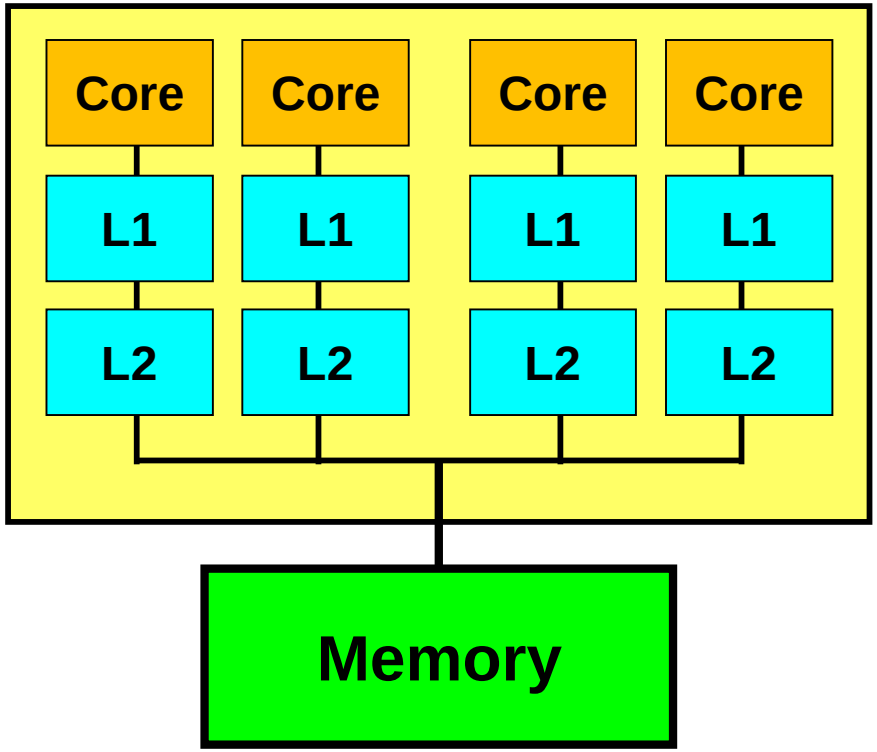


多核
L1 Cache私有
共享片内L2 Cache

7.3.1 多核处理器定义与结构



多核
L1和L2 Cache私有
共享片内L3 Cache



多核
L1和L2 Cache私有
分布式存储器

7.3.1 多核处理器定义与结构

- 共享Cache的优点：
  - 在共享Cache级不需要一致性协议
  - 通信延迟小
  - 工作集可以交叉
    - 可由某个核预取数据
    - 无需设置大的Cache
    - 提高了Cache块利用率
  - 动态共享
    - 可以在核之间动态分配Cache空间

7.3.1 多核处理器定义与结构

- **共享Cache的缺点：**
  - 核数量的增加，导致需求增加。包括：
    - 更高的带宽
    - 更大的Cache容量 → 导致更大的延迟
  - 命中延迟增加，因为要经过互连网络
  - 设计更为复杂
  - 某个核可能将其他核的数据替换出去

7.3.1 多核处理器定义与结构

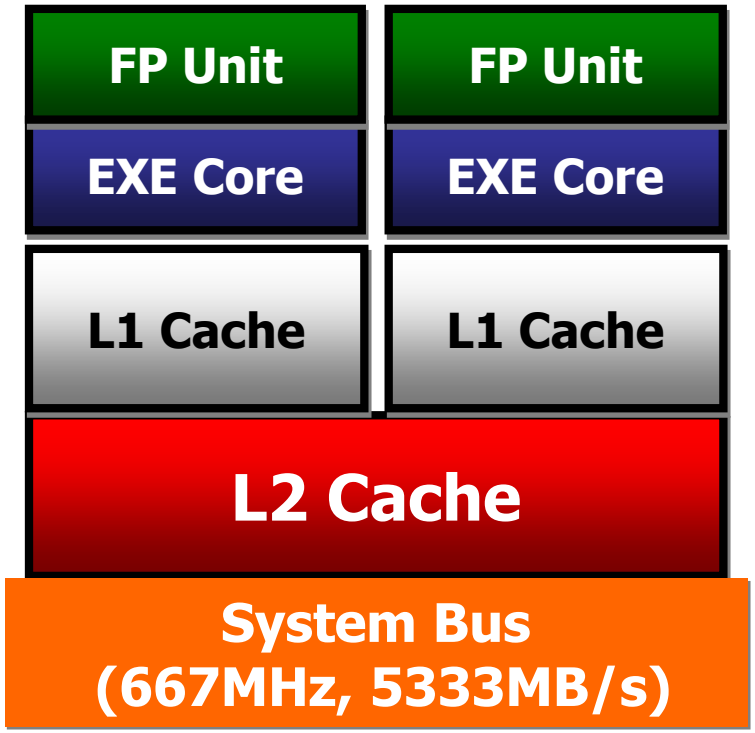
- 目前已有很多双核处理器，如：
  - Intel：Pentium D and Pentium Extreme Edition, Core Duo(2), Woodcrest, Montecito
  - IBM PowerPC
  - AMD Opteron / Athlon 64
  - Sun UltraSPARC IV
- 多核处理器 如：
  - IBM Cell (非对称)
    - 双核 PowerPC + 8个协处理单元
  - Sun Niagara
    - 8 cores，每核4个超线程
  - 通用计算图形处理器 (GPGPU) 等

7.3.1 多核处理器定义与结构

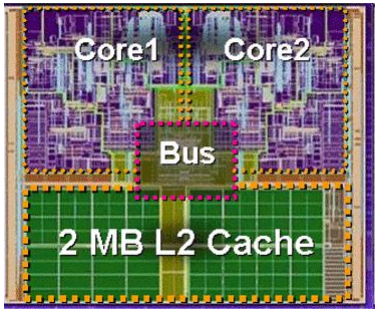
- 下列处理器分别采用了上面介绍的结构：
  - Intel Core Duo
  - Intel Core i7
  - AMD Opteron
  - ARM11 MPCore

7.3.2 Intel多核处理器

- 2006年，Intel推出了两个x86超标量双核处理器Core Due



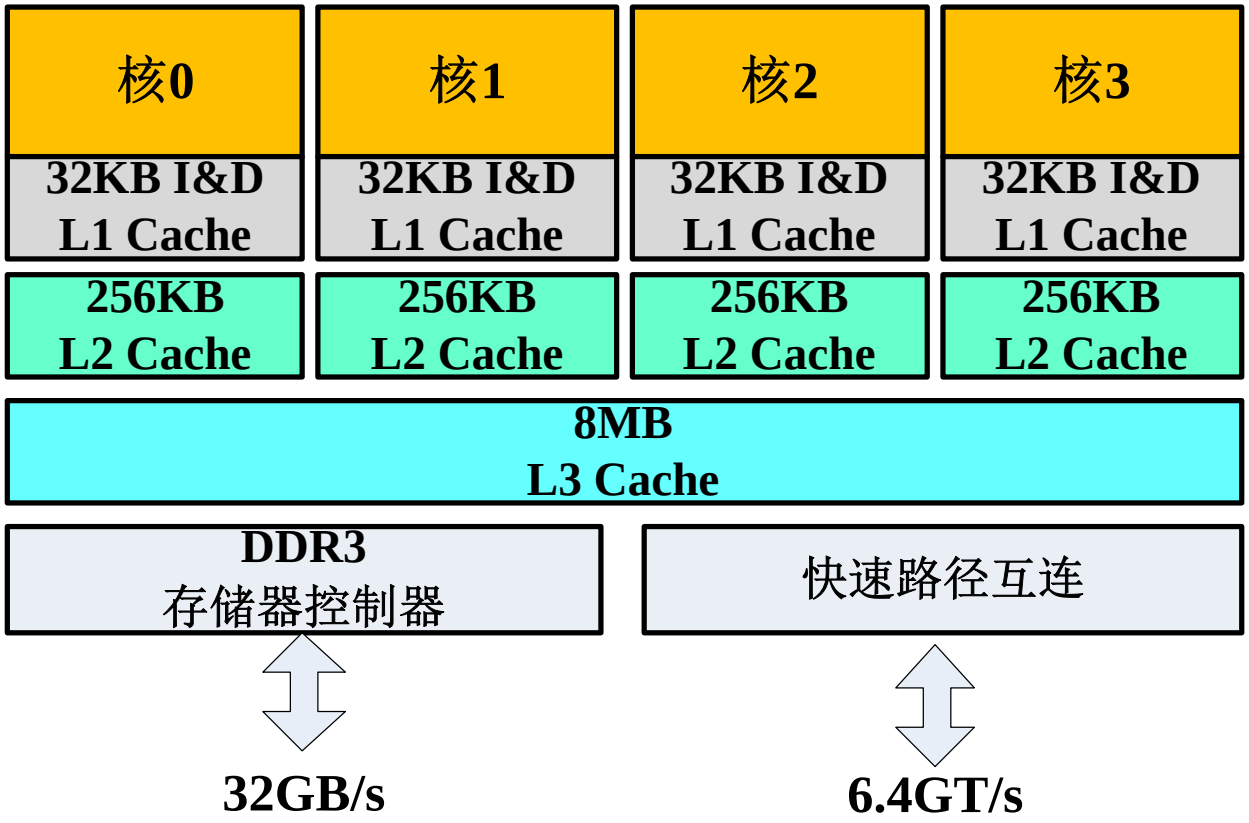
Intel Core Duo结构



- ☑ 双核
- ☑ 每核有自己的执行资源
- ☑ 每核一个私有 L1 cache
  - 32K 指令 和 32K 数据
- ☑ 双核共享 L2 cache
  - 2MB 8-路组相联；每行64字节
  - 10个时钟周期延迟；写返回更新策略

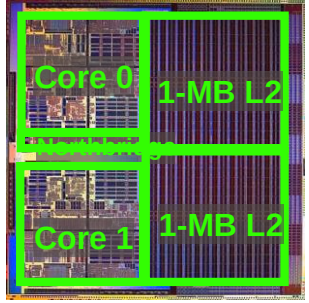
7.3.2 Intel多核处理器

- 2008年，Intel推出了4核 4个x86 SMT的Core i7

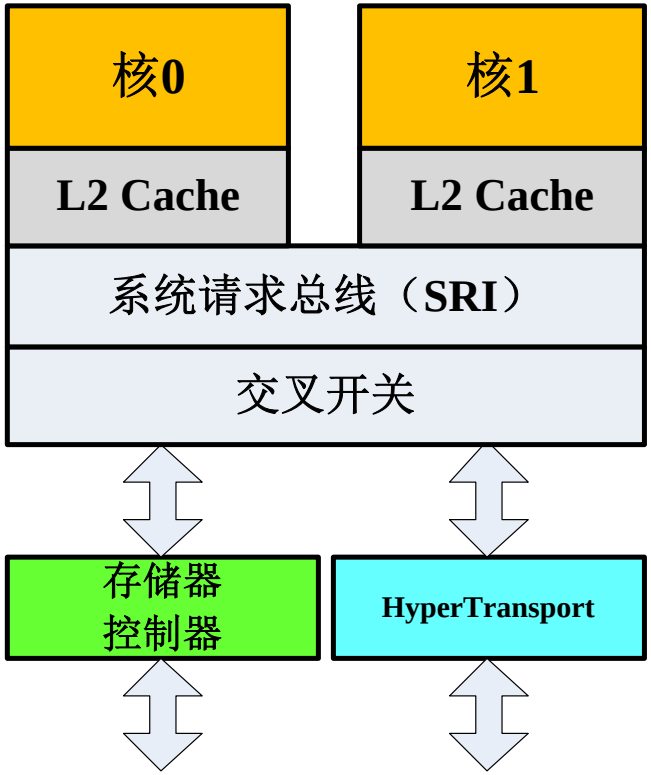


Intel Core i7结构

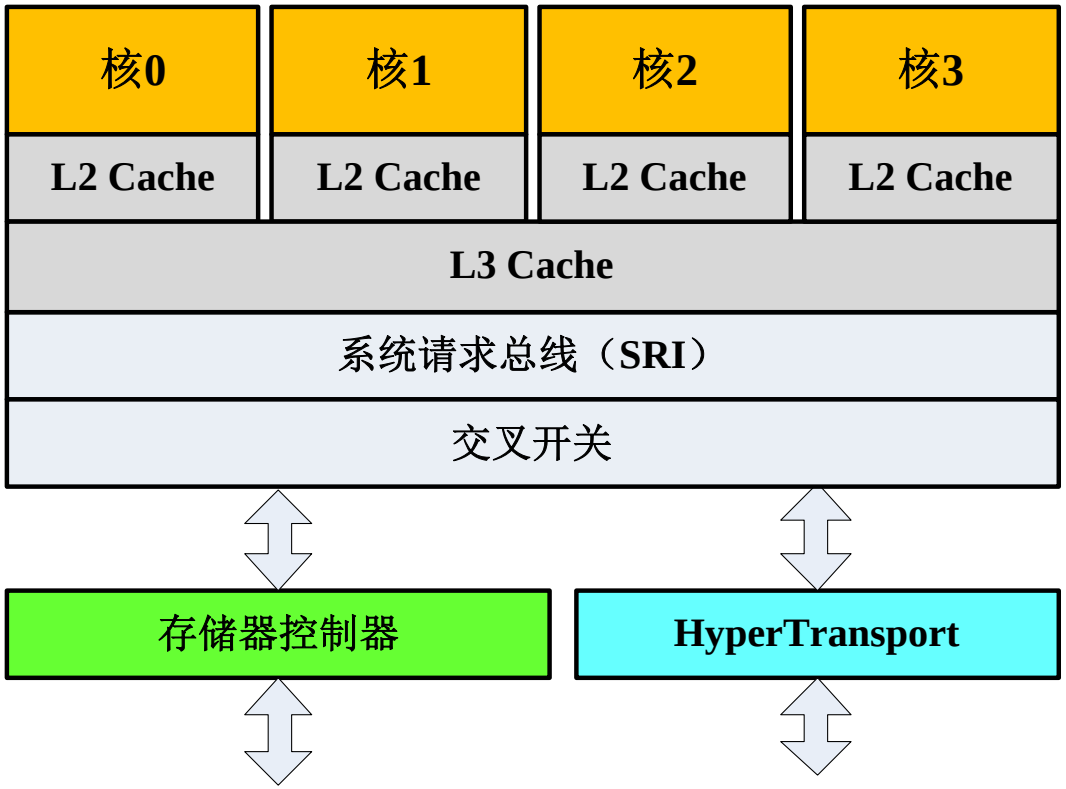
7.3.3 AMD多核处理器



- 2005年，AMD发布了第一款双核处理器 Opteron



AMD第一代
Dual-Core Opteron结构



AMD第三代Quad-core
Opteron处理器结构

7.3.3 ARM多核处理器

- **ARM 11系列： MPCore 多核架构**

ARM Cortex-A9 MPCore架构示意图

学习内容

- 7.1 多处理机概念
- 7.2 多处理机结构
- 7.3 多核处理器
- 7.4 多处理机的多Cache一致性
- 7.5 多处理机的机间互连形式
- 7.6 程序并行性
- 7.6 多处理机性能
- 7.7 多处理机操作系统

7.4 多处理机多Cache一致性

- Cache是提高系统性能的一种常用手段，多处理机也广泛应用Cache。
- 优点：
  - 减少访问时延，降低对存储器频宽要求
  - 减少同时访问存储器时的冲突 等

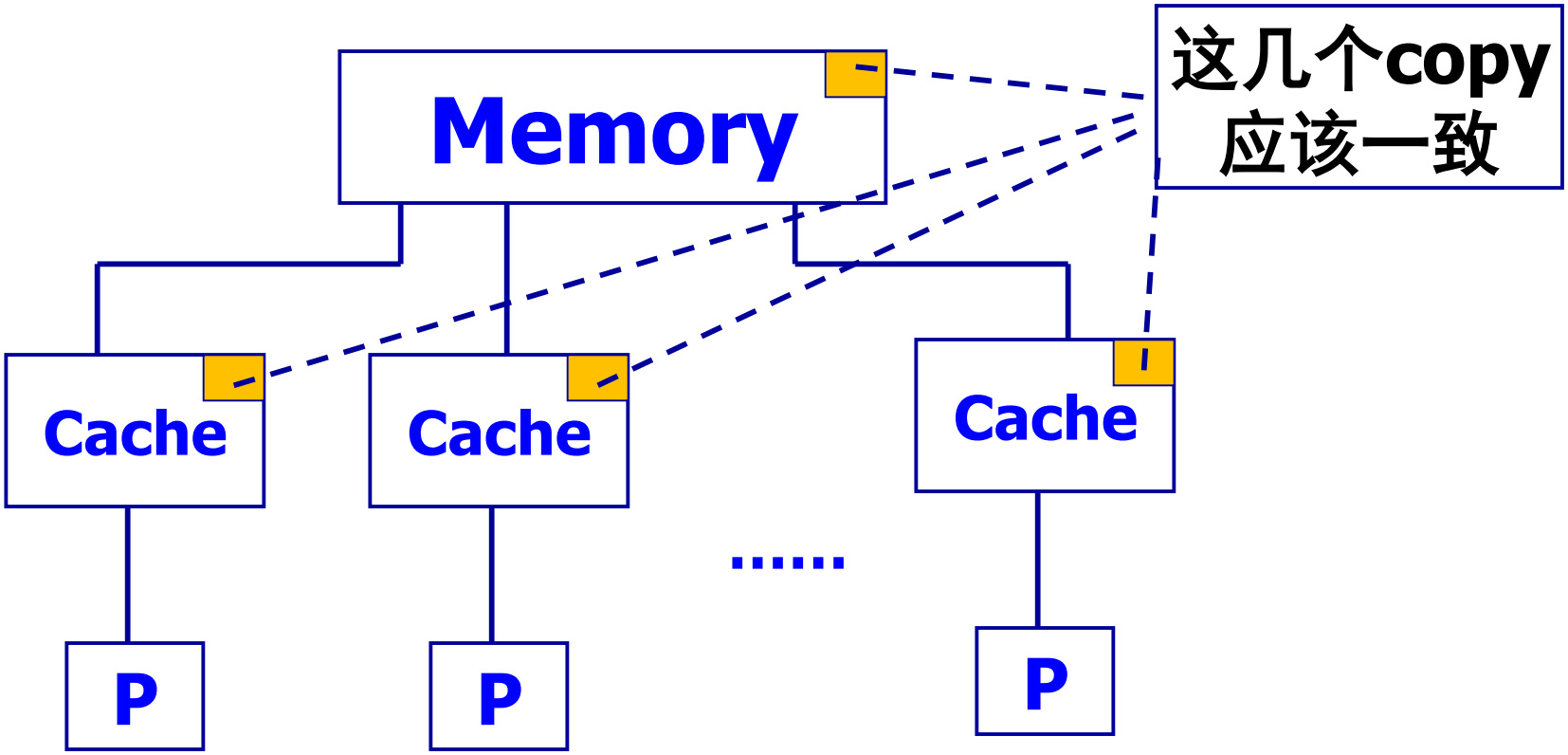
7.4 多处理机多Cache一致性

- 当把共享数据装入Cache中时，会在多个Cache中形成副本
- 新问题：

共享数据进入Cache时产生一个新问题：

Cache一致性 (Cache Coherence) 问题

7.4.2 多Cache一致性问题的产生

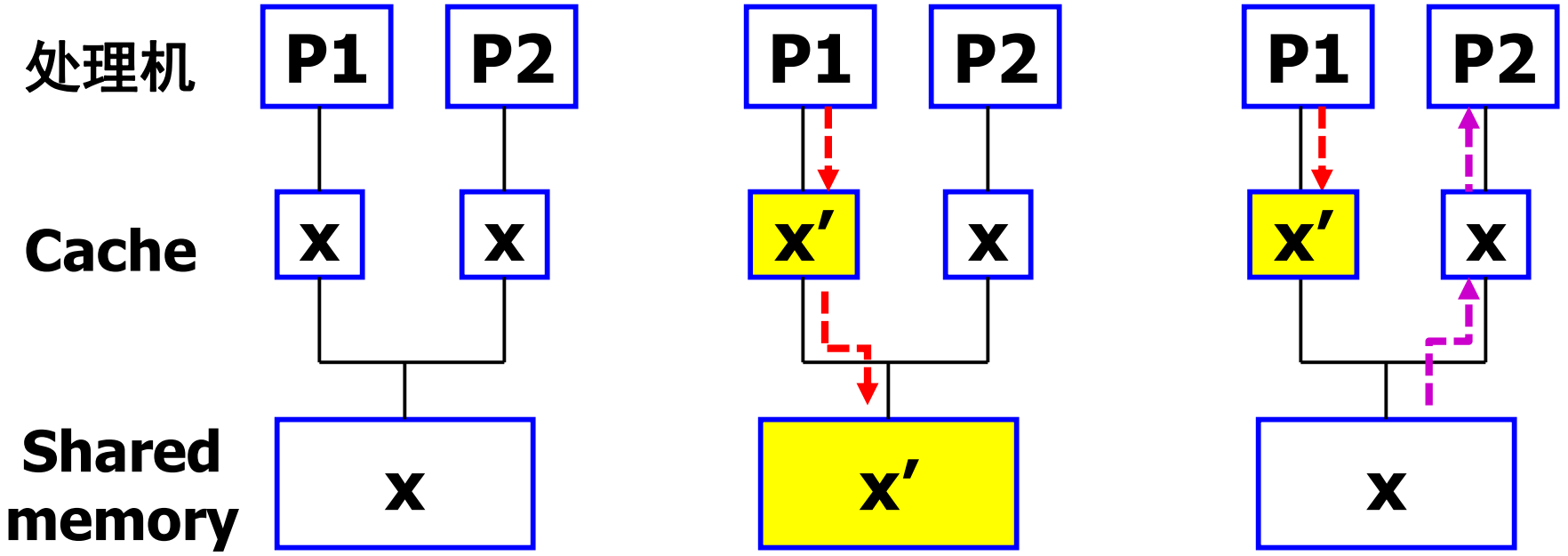


在多处理器系统中，多个Cache中对应的共享数据的copy应该一致。

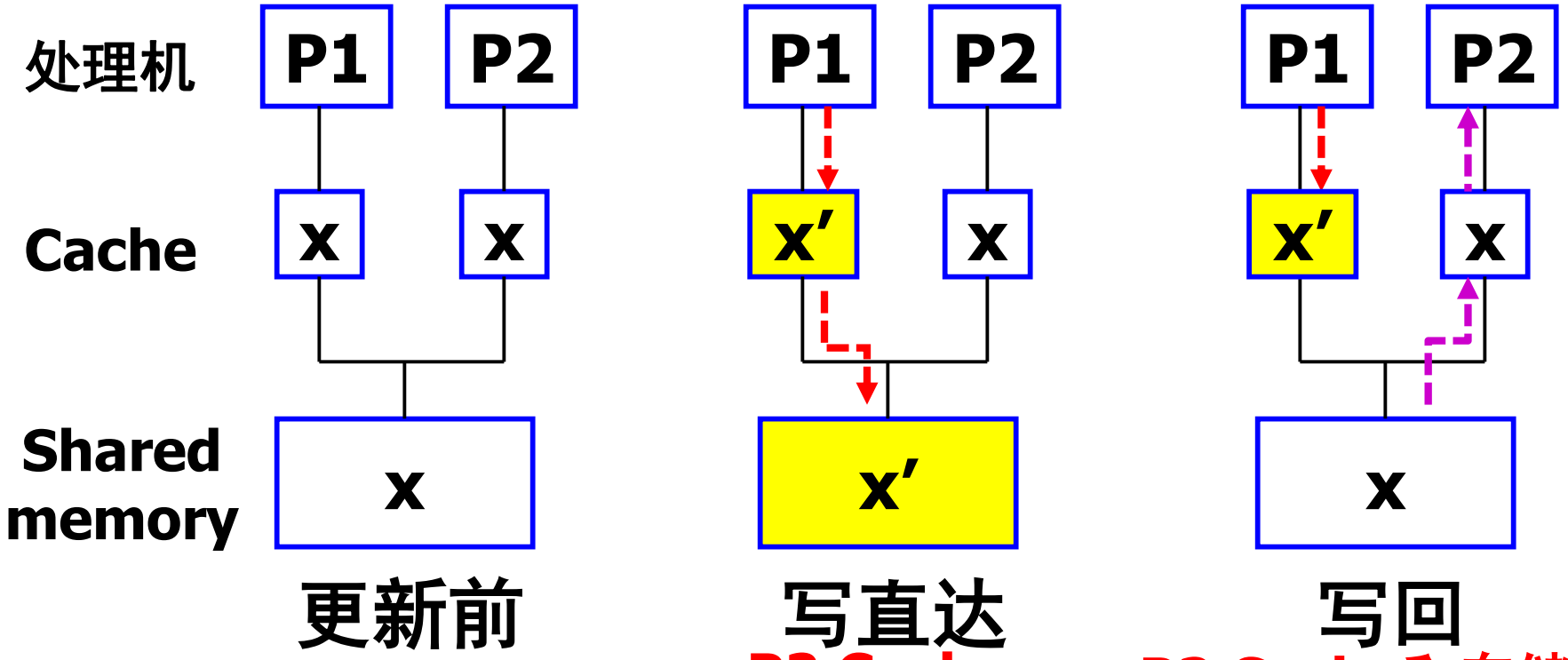
7.4.2 多Cache一致性问题的产生

- 不一致的三种原因：
  - 共享可写数据
  - 进程迁移
  - I/O传输

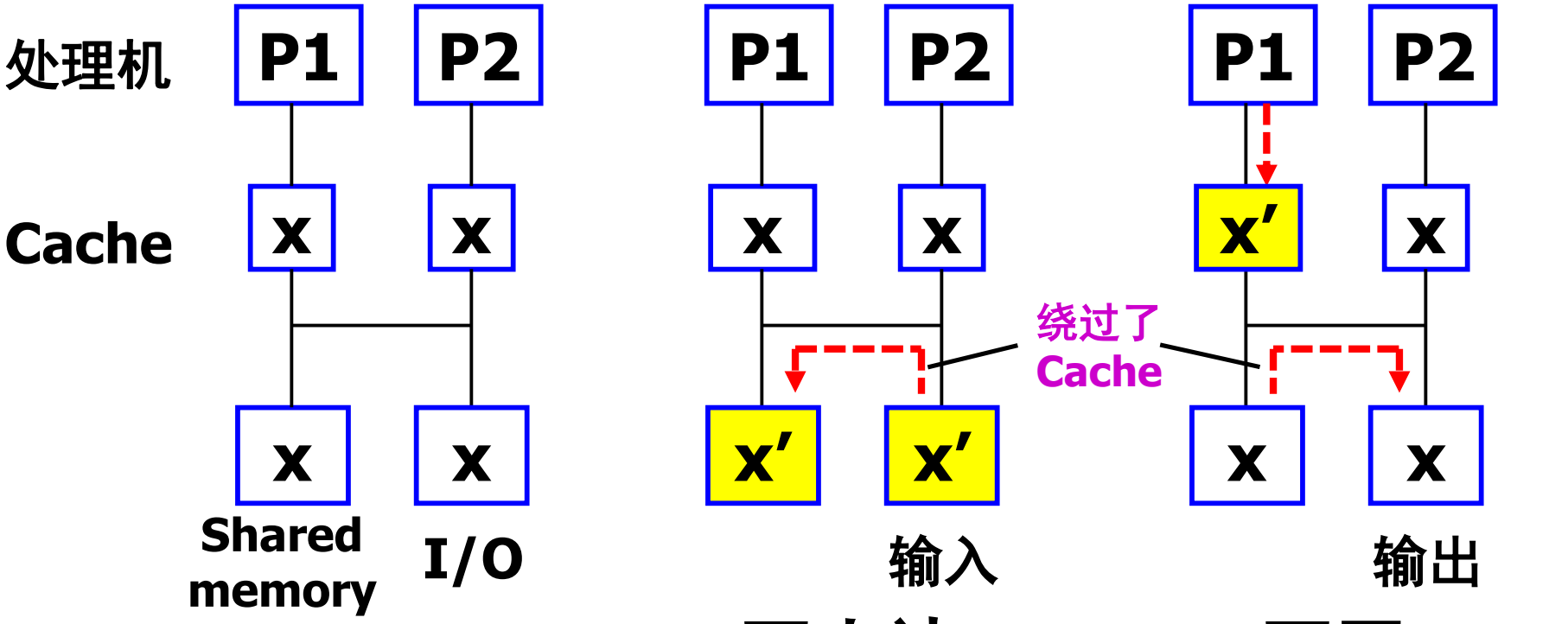
共享可写数据引起的不一致性



进程迁移引起的不一致性



I/O传输引起的不一致性



7.4.3 多Cache一致性问题解决方法

- 在上述三种情况中，不论采用写直达法还是写回法，在多处理机中都可能引起Cache不一致问题。
- 解决方法：2类
  - 基于硬件的方法
    - 思想：跟踪共享数据的状态
    - Cache一致性协议：监听协议和基于目录的协议
  - 基于软件的方法
    - 由编译和操作系统检测并解决

基于硬件的方法：两种Cache一致性协议

1. 监听协议(Snooping Protocol) — 拥有数据副本的Cache各自记录数据块的状态，无集中的状态。

2. 基于目录的协议(Directory based Protocol) — 将每个数据块的共享状态记录保存在某个地点 — 目录。

监听协议 (Snoopy Protocol)

- Goodman 1983
- 适用于：具有广播能力的总线结构多机系统。但只适用于小规模的多处理机系统
- 分布式算法：分散到所有Cache控制器。各Cache控制器自行产生状态变化及相应操作
- 两种策略：
  - 写无效（Write-Invalidate）（或 写作废）
  - 写更新（Write-Update）（或 播写法）

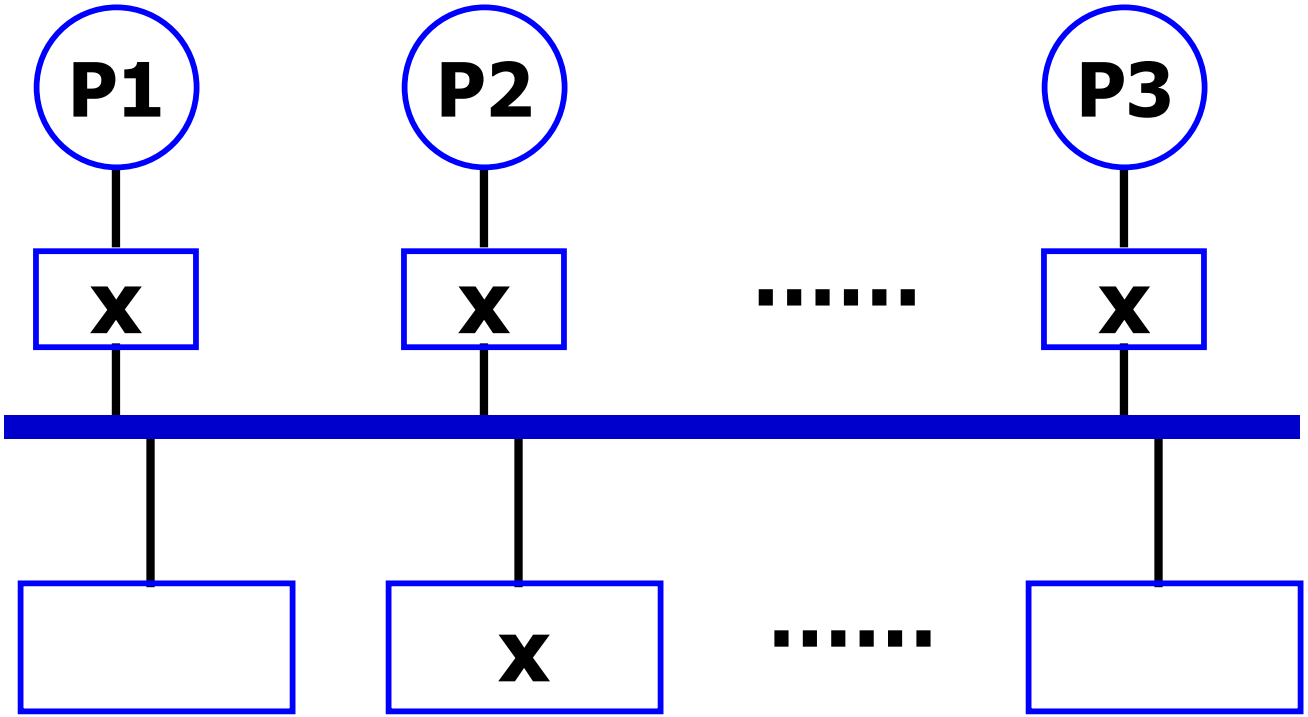
监听协议 (Snoopy Protocol)

- 写无效（Write-Invalidate）（或 写作废）
  - 任何一个处理器写或修改它的私有Cache中的共享数据时，它都使所有其它Cache中的副本失效。
  - 对Write-through：它也更新存储器中的副本（最终是：只有一个Cache中的副本和存储器中的副本是有效的）。
  - 对Write-back：它使存储器中的副本也失效（最终是：只有一个Cache中的副本是有效的）。

监听协议 (Snoopy Protocol)

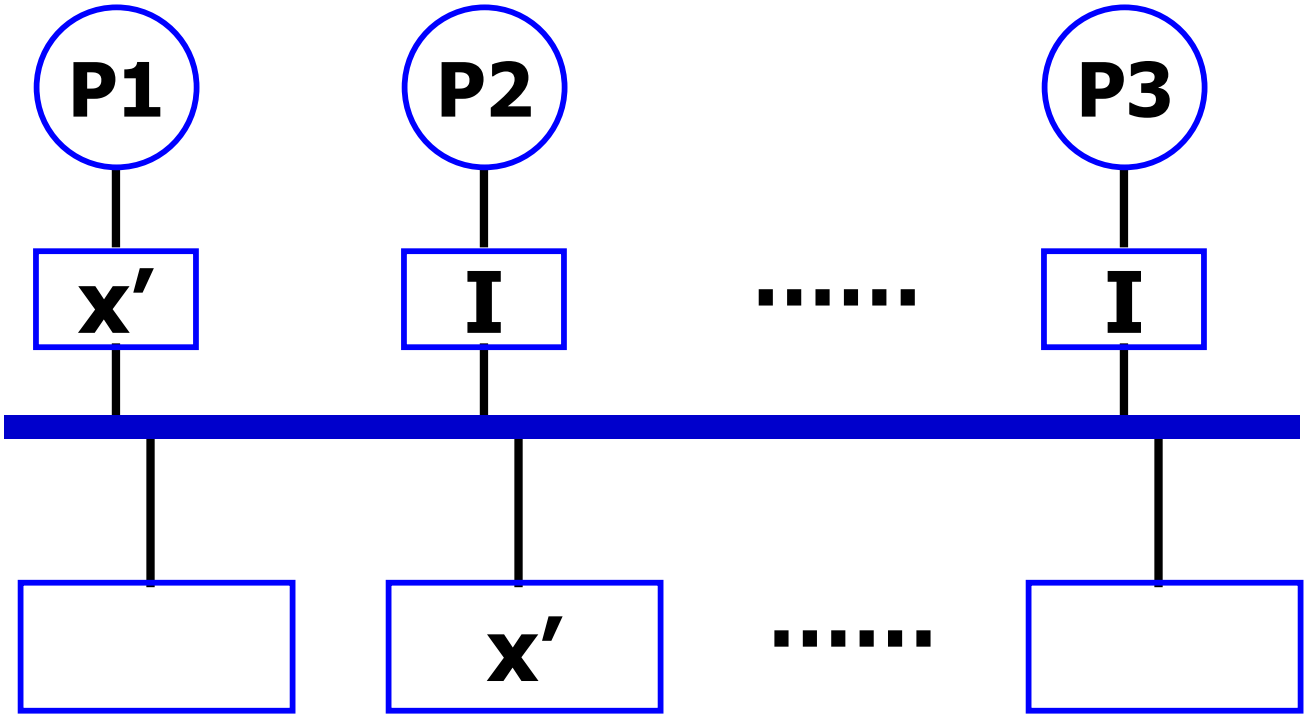
- 写更新（Write-Update）（或 播写法）
  - 任何一个处理器写或修改它的私有Cache中的共享数据时，它都立即更新所有其它Cache中的副本。
  - 对Write-through：它也更新主存储器中的副本。
  - 对Write-back：对存储器中副本的更新延迟到这个Cache被置换的时刻。

监听协议 (Snoopy Protocol)



更新前
数据块x在共享存储器和3台处理机的Cache中的副本一致

写无效 + 写直达



处理机

Cache

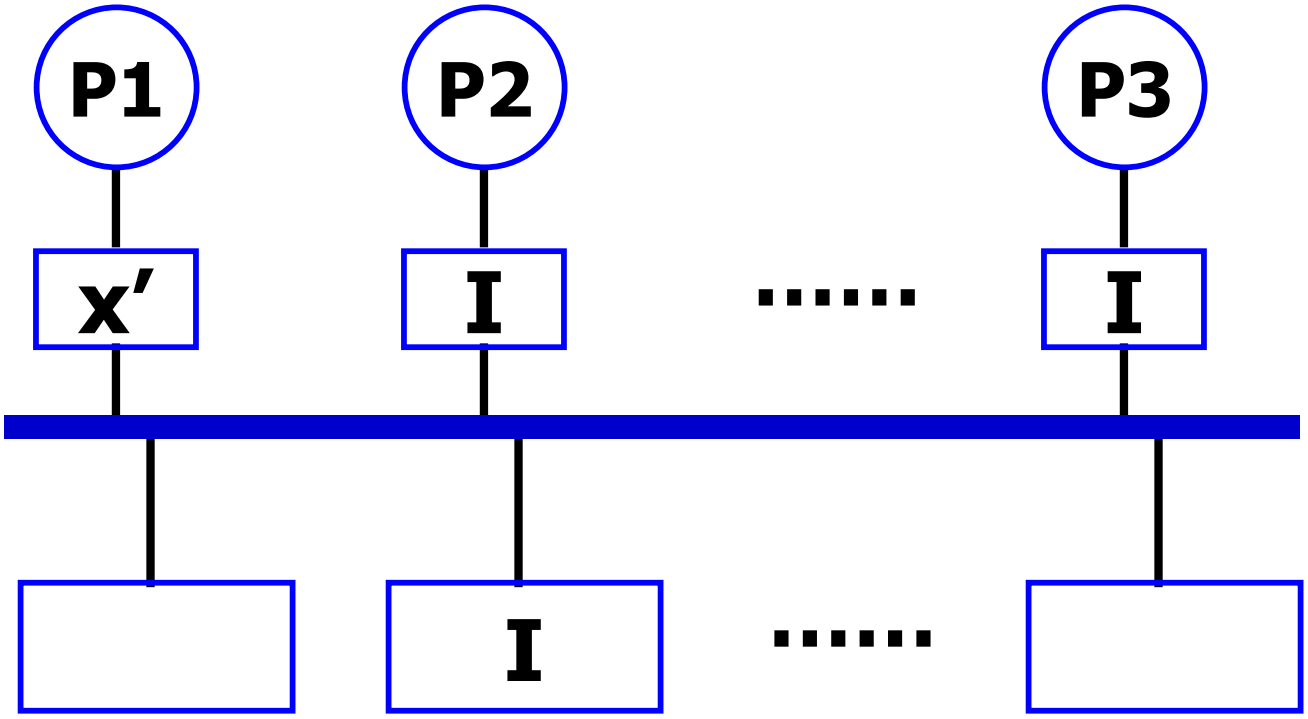
总线

共享存储器

I表示无效

只有一个Cache中的副本和存储器中的副本是有效的

写无效 + 写回



处理机

Cache

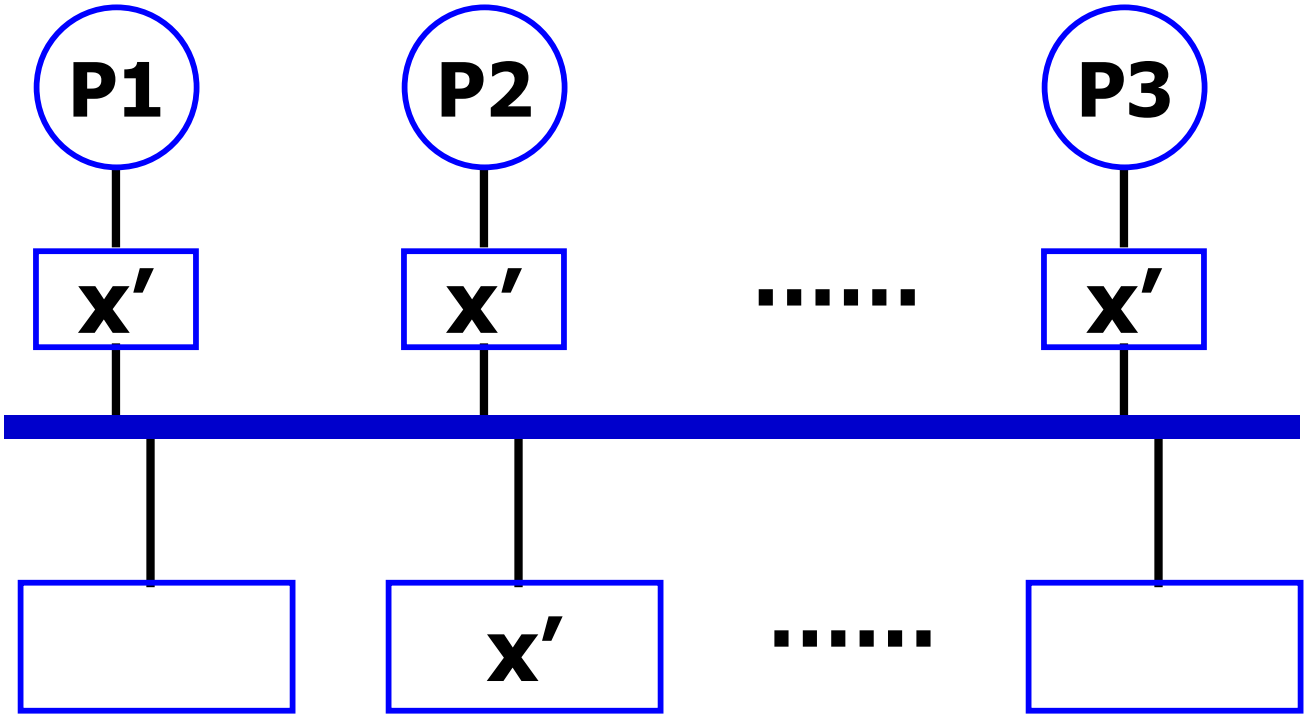
总线

共享存储器

I表示无效

只有一个Cache中的副本是有效的

写更新 + 写直达



更新所有Cache以及主存储器中的副本

写无效

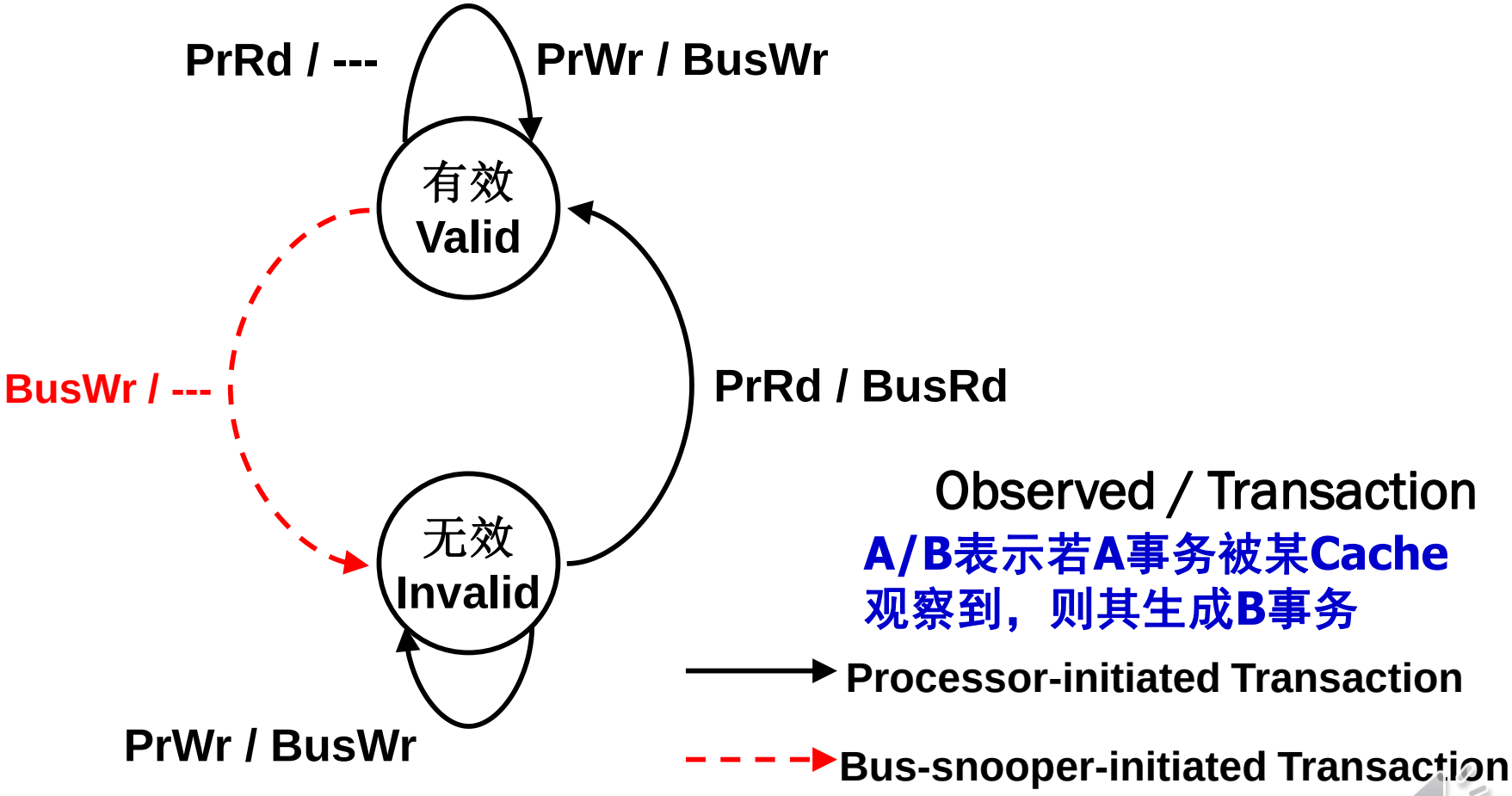
- **主要开销在两个方面：**
  - （1）作废各Cache副本的开销；
  - （2）由作废引起缺失造成的开销，即处理机需要访问已经作废的数据时将引起Cache的缺失。
- **后果：**
  - 如果一个处理机经常对某个块连续写，且各处理处理机间对共享块的竞争较小，这时写无效策略维护一致性的开销是很小的。
  - 如发生严重竞争，将产生较多的作废，引起更多的作废缺失，结果是共享数据在各Cache间倒来倒去，产生颠簸现象。当缓存块比较大时，这种颠簸现象更为严重。

监听协议：写无效 + 写直达

- **Cache块状态：**
  - **有效(V)：干净块(数据与主存中相同)，该块可同时在多个Cache中存在**
  - **无效(I)：此块不在Cache中【或已作废】**
- **协议基本思想：**
  - **满足Cache工作流程需求，写操作时写主存，通过总线事务通知其它Cache作废该块**

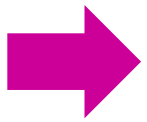
监听协议：写无效 + 写直达

Invalidation-based Protocol on Write-Through cache



监听协议：写无效 + 写回

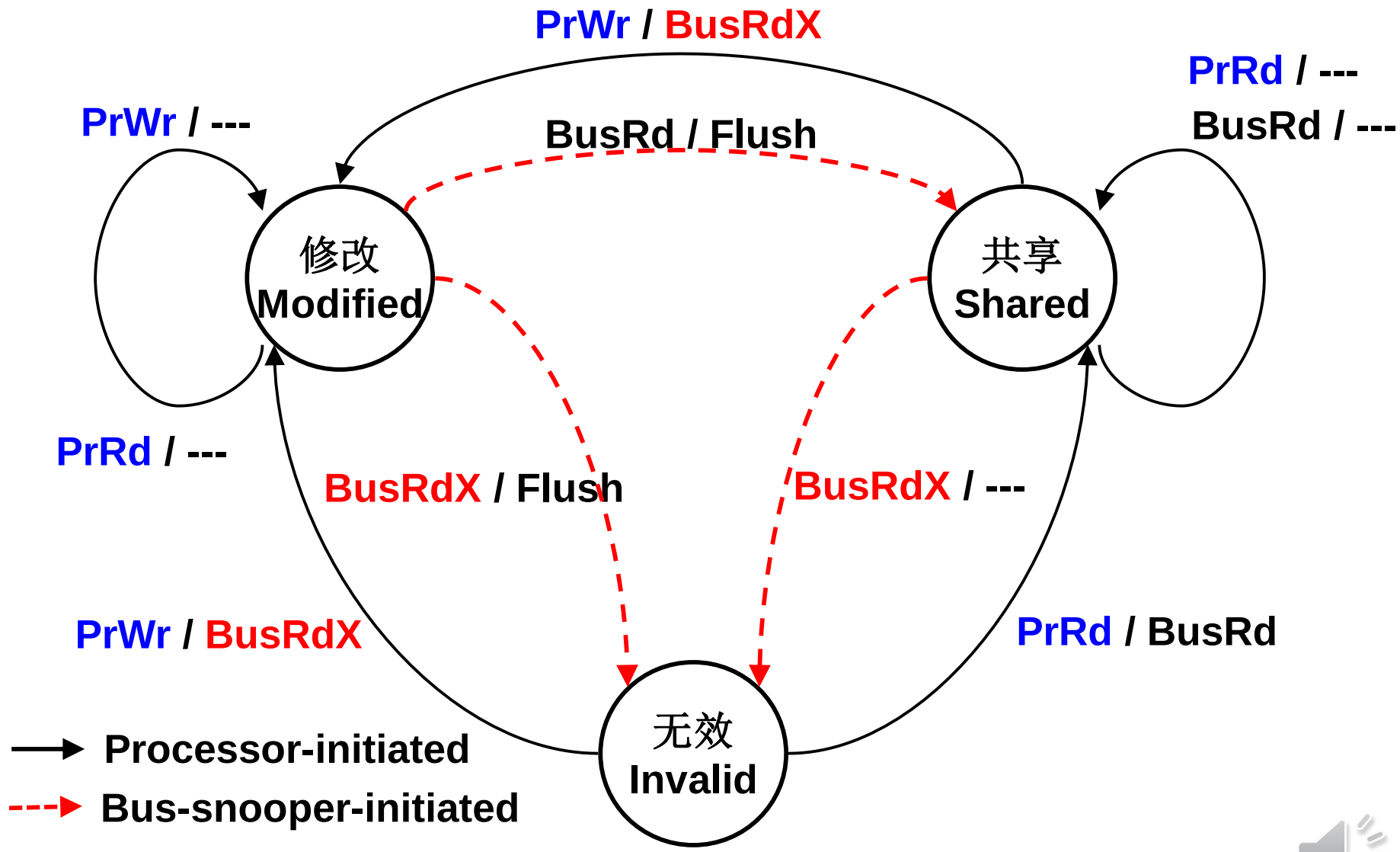
- Cache块状态
  - 修改(Modified) (或增加 独占(Exclusive))
    - 数据被修改了
    - 仅该Cache拥有正确的副本
  - 共享(Shared)
    - 存储器一致
    - 一个或多个Cache都拥有正确的副本
  - 无效(Invalid)



MSI 协议
MESI 协议

写回协议：
在更新存储器之前，可以多次写Cache行

监听协议：写无效 + 写回 (MSI)



监听协议的开销分析

- （1）采用写无效协议
  - 无效后，当其它处理机再读该副本时，出现Read miss，要有开销。
- （2）采用写更新协议
  - 需要更新所有Cache和memory中的副本，所以开销大，有些处理机对更新后的数据可能不会使用。

基于目录的协议

- 基于目录协议的基本思想：
  - 当处理机台数增加时，一般不用总线结构，而采用多级互连网络。但多级互连网络实现广播功能代价很大。
  - 为什么需要广播功能？把一致性命令，如Write、Read等命令发送给所有的Cache。
  - 能不能只发送给存放该副本的Cache?

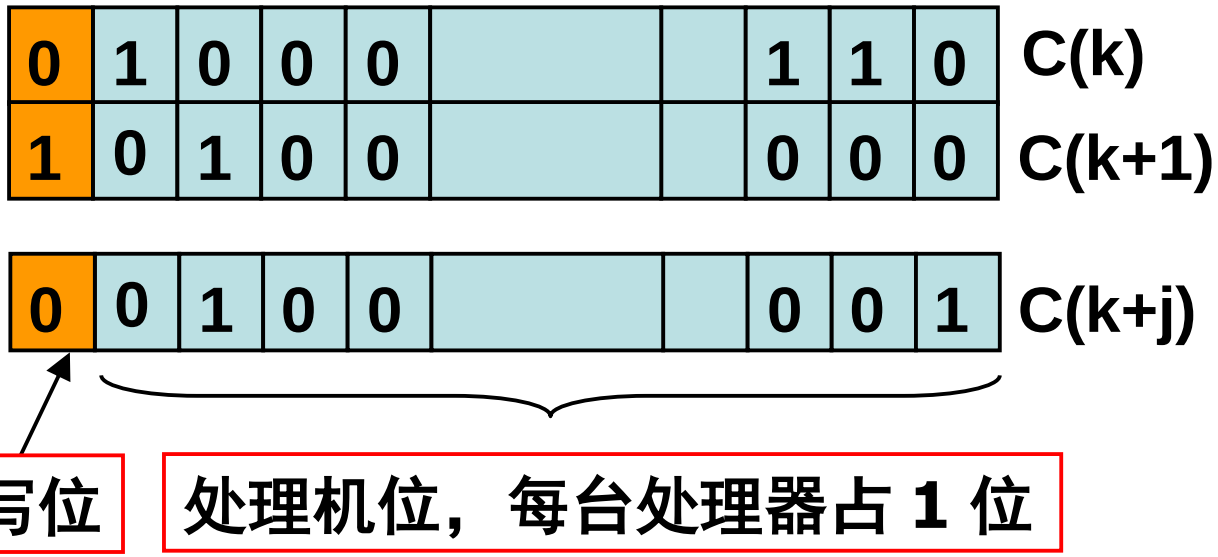
    ——基于目录的协议的基本思想

基于目录的协议

- 在每个结点增加目录存储器，用于存放目录。
- 目录里放什么？
  - 有关Cache副本驻留在哪里的信息：所有共享数据块的所有Cache副本的地址表。
  - 目录必须跟踪记录每个共享数据块的状态。

基于目录的协议

- 目录结构



处理机位表示相应处理机Cache中是否存在共享数据块的副本（存在或不存在）。

如果重写位为“1”，而且有且只有一个处理机位为“1”，则表示该处理机可以对该数据块进行修改。

基于目录的协议

目录的存放方式：2种（1/2)

- **1. 集中式目录方式（中心目录）**
  - 1976年 Tang提出。
  - 用一个中心目录存放所有Cache目录的副本，它能提供为保证一致性所需要的所有信息。
  - 缺点：容量非常大，必须采用联想方法来检查，扩展性差，冲突多，检索时间长。

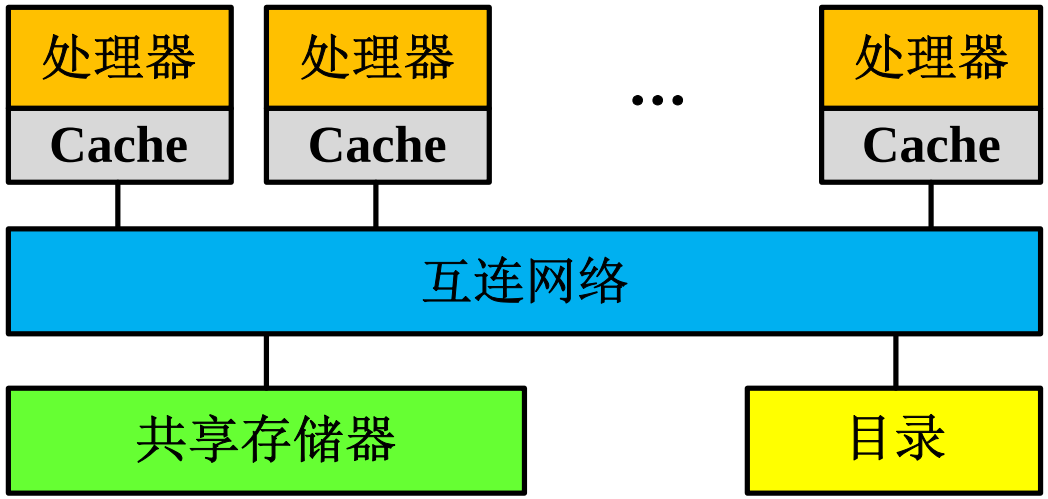
基于目录的协议

目录的存放方式：2种（1/2）

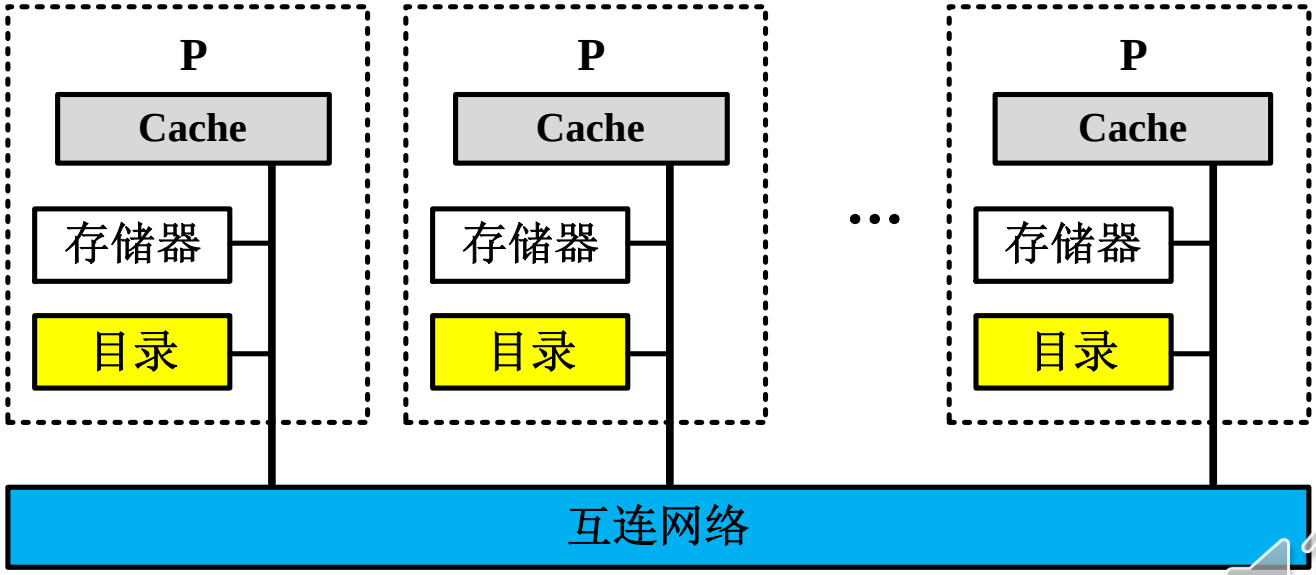
- **2. 分布式目录方式**
  - 1978年 Censier和Feautrier 提出。
  - 每个存储模块维护各自的目录，目录中记录着每个存储块的当前信息。当前信息指明哪些Cache有该存储块的副本。

基于目录的协议

集中式目录方式



分布式目录方式

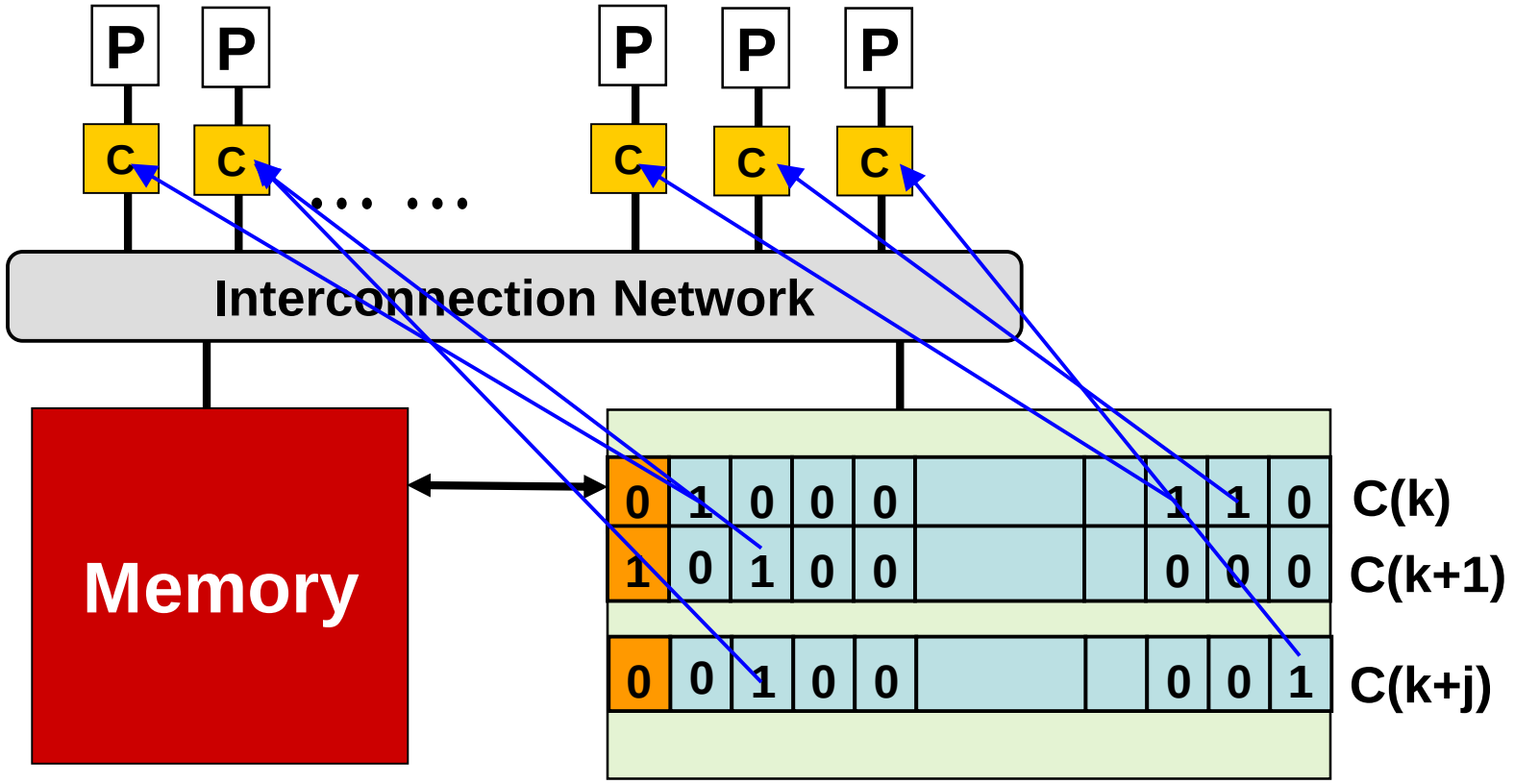


基于目录的协议

目录类型：3种

- **1. 全映射（full map）目录：**存放与全局存储器中每个块有关的数据。系统中的每个Cache可以同时存储任何数据块的副本，即每个目录项包含N个指针（N是处理机数目）。
- **2. 有限（limited）目录：**每个目录项有固定数目的指针（小于N）。
- **3. 链式（chained）目录：**将目录分布到各个Cache（其余同全映射目录）。

全映射目录

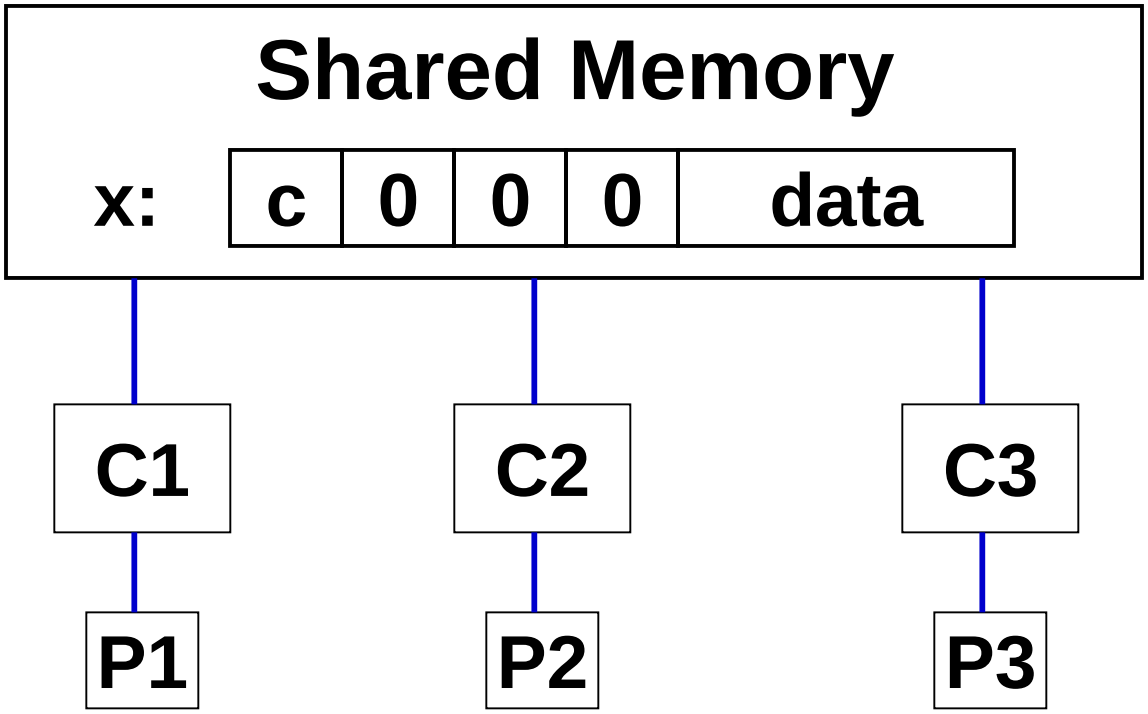


一个重写位，说明是否允许向Cache数据块写入数据

每台处理机一个处理机位，指示该处理机Cache中是否存在该数据块

我们来看三台处理机（3个Cache）使用全映射目录的例子。

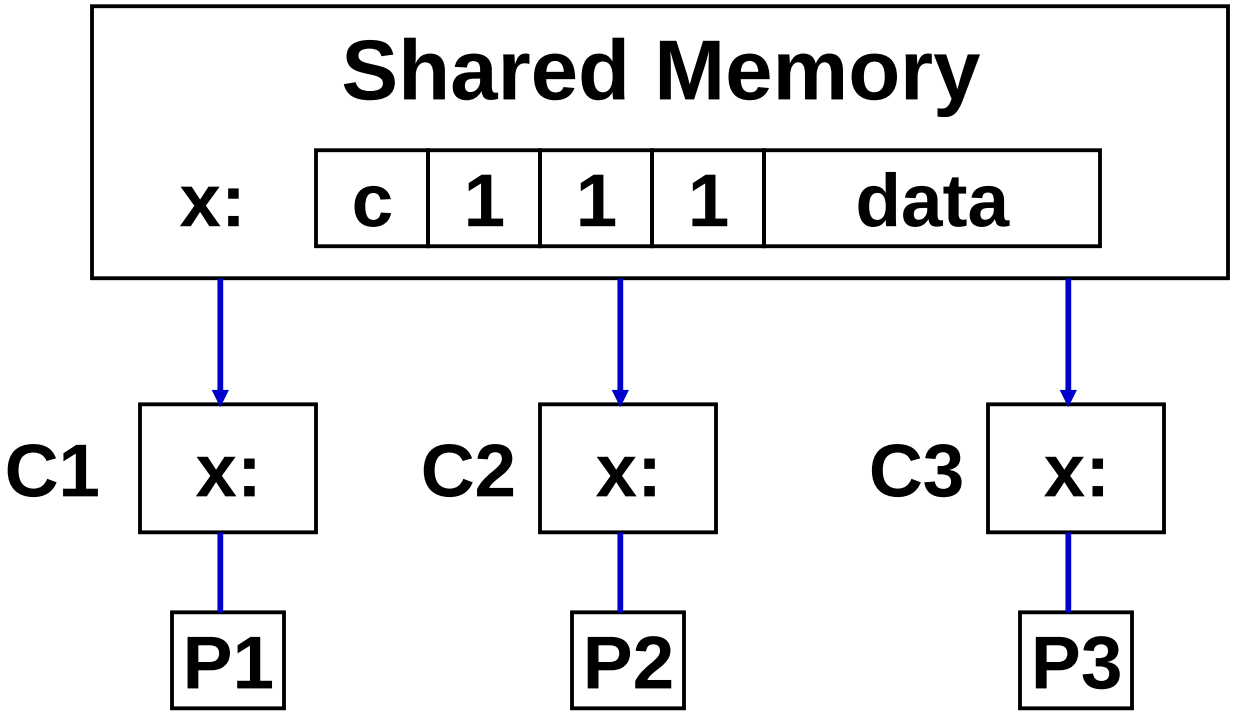
（1）C1，C2，C3都没有单元X的副本。



（2）C1，C2，C3同时请求X单元的副本。

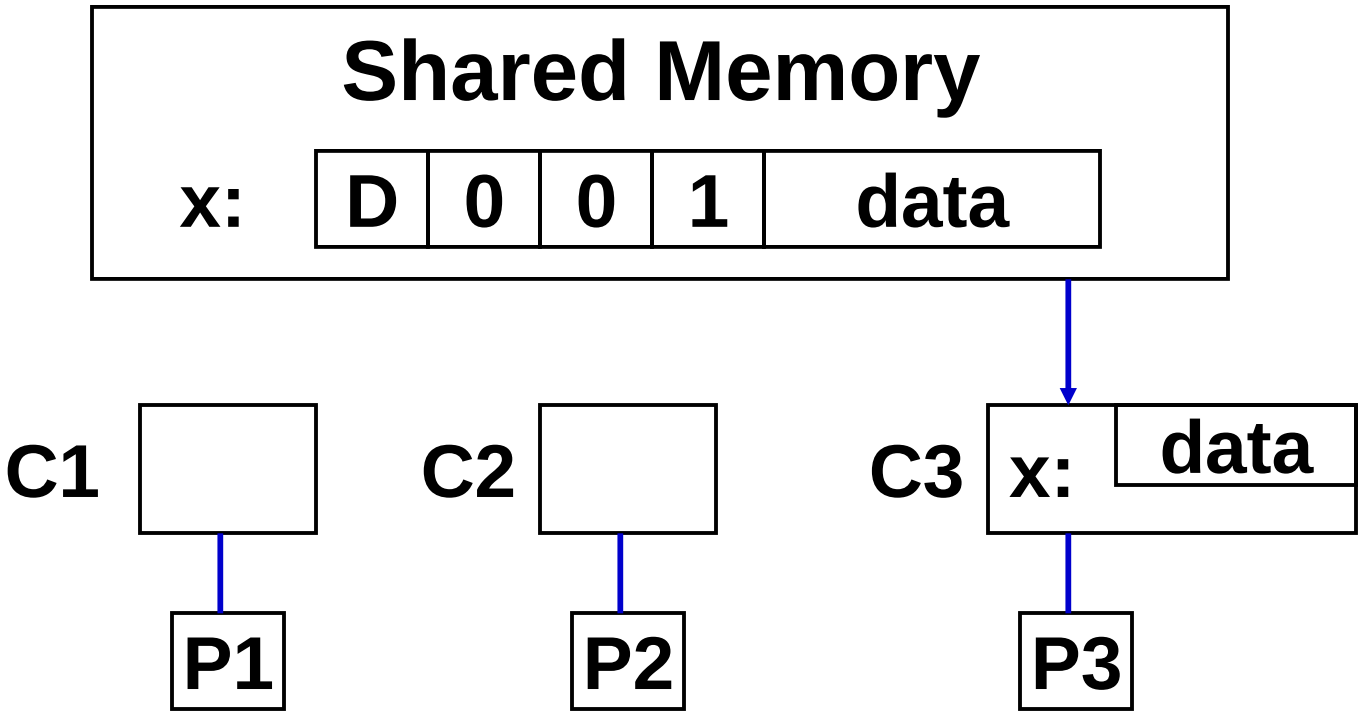
这时目录项中的3个指针（处理机位）被置1，表示这些Cache中已有数据副本。

目录项的重写位被置为未写 (c) 状态，表示无一处理机允许写入该数据块。



（3）C3请求对该块的写允许权。

成功后，重写位被置成允许写(D)状态，且有一个指针指向C3的数据块。



（3）C3获得写允许权的过程

P3向C3发出写请求时 (1/2)：

- （1）C3检测出包含单元X的Cache块是有效的，但Cache中的块重写位状态表示不允许处理机对该块进行写操作。
- （2）C3向包含单元X的存储器模块发出写请求，并暂停P3工作。
- （3）该存储器模块发出一个无效请求给C1和C2（根据目录项的内容发2个或多个无效请求）

（3）C3获得写允许权的过程

P3向C3发出写请求时 (2/2)：

- （4）C1和C2收到无效请求后，把Cache块置为无效，并发送一个回答信号给请求的存储器模块。
- （5）存储器模块收到回答信号后，将重写位置1，清除指向C1、C2的指针，发出允许信号给C3。
- （6）C3收到写允许信号后，修改Cache的状态并激活处理机P3。

学习内容

- 7.1 多处理机概念
- 7.2 多处理机结构
- 7.3 多核处理器
- 7.4 多处理机的多Cache一致性
- 7.5 多处理机的机间互连形式
- 7.6 程序并行性
- 7.6 多处理机性能
- 7.7 多处理机操作系统

7.5 多处理机的机间互连形式

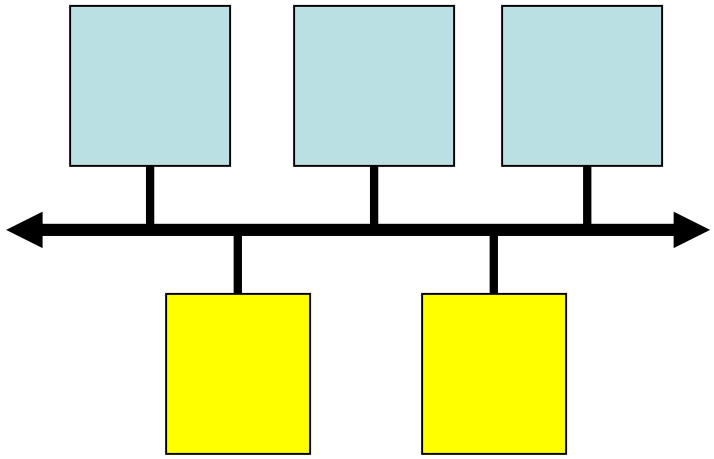
- 互连网络是并行计算机的核心。
- 多处理机的机间互连要求：
  - 高速率
  - 低成本
  - 能实现各种复杂、无规则的互连而不发生冲突
  - 具有良好的可扩展性

7.5 多处理机的机间互连形式

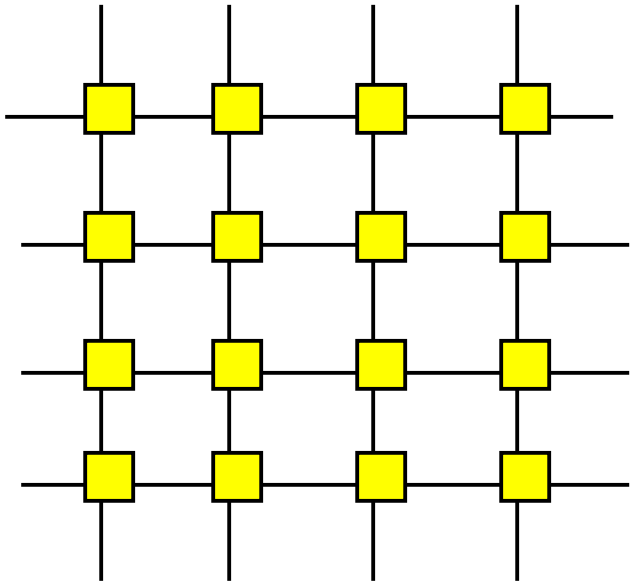
- 主流选择：
  - 总线形 (Bus)
  - 环形 (Ring)
  - 交叉开关 (Crossbar)
  - 带缓存的交叉开关 (Buffered crossbar)
  - 多级交叉开关
  - 多端口存储器
  - 网状 (Mesh)
  - 应用特定的 (Application-specific)

Bus network

- **Advantages:**
  - **Well-understood.**
  - **Easy to program.**
  - **Many standards.**
- **Disadvantages:**
  - **Contention.**
  - **Significant capacitive load.**



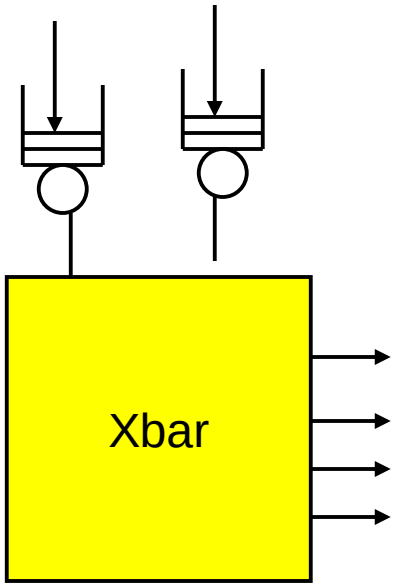
Crossbar



- **Advantages:**
  - **No contention.**
  - **Simple design.**
- **Disadvantages:**
  - **Not feasible for large numbers of ports.**

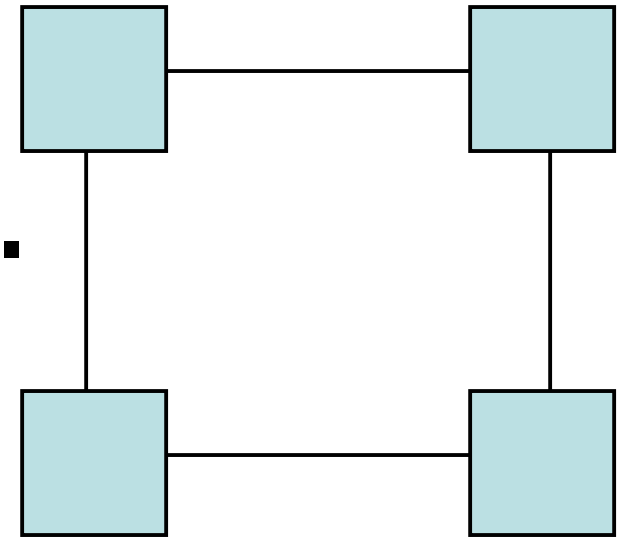
Buffered crossbar

- **Advantages:**
  - **Smaller than crossbar.**
  - **Can achieve high utilization.**
- **Disadvantages:**
  - **Requires scheduling.**



Mesh

- **Advantages:**
  - **Well-understood.**
  - **Regular architecture.**
- **Disadvantages:**
  - **Poor utilization.**



学习内容

- 7.1 多处理机概念
- 7.2 多处理机结构
- 7.3 多核处理器
- 7.4 多处理机的多Cache一致性
- 7.5 多处理机的机间互连形式
- 7.6 程序并行性
- 7.6 多处理机性能
- 7.7 多处理机操作系统

7.6 程序并行性

- 并行处理面临着两个重要的挑战：
  - 程序中有限的并行性
  - 相对较高的通信开销

$$\text{系统加速比} = \frac{1}{(1-\text{可加速部分比例}) + \frac{\text{可加速部分比例}}{\text{理论加速比}}}$$

并行处理面临的挑战

1. 程序中有限的并行性

- 使机器要达到好的加速比十分困难

例：假设有100个处理器，希望获得80倍的加速比。问原始程序中串行部分的百分比应达到多少？

a. 10%
b. 5%
c. 1%
d. <1%

Amdahl's Law Answers

$$\text{Speedup}_{\text{overall}} = \frac{1}{\left(1 - \text{Fraction}_{\text{enhanced}}\right) + \frac{\text{Fraction}_{\text{parallel}}}{\text{Speedup}_{\text{parallel}}}}$$

$$80 = \frac{1}{\left(1 - \text{Fraction}_{\text{parallel}}\right) + \frac{\text{Fraction}_{\text{parallel}}}{100}}$$

$$80 \times \left(\left(1 - \text{Fraction}_{\text{parallel}}\right) + \frac{\text{Fraction}_{\text{parallel}}}{100}\right) = 1$$

$$79 = 80 \times \text{Fraction}_{\text{parallel}} - 0.8 \times \text{Fraction}_{\text{parallel}}$$

$$\text{Fraction}_{\text{parallel}} = 79 / 79.2 = 99.75\%$$

并行处理面临的挑战

2. 相对较高的通信开销

- 在现有的机器中，处理器之间的数据通信大约需要50～10000个时钟周期。

例： 一台32个处理器的计算机，对远程存储器访问时间为2000ns。除了通信以外，假设计算中的访问均命中局部存储器。当发出一个远程请求时，本处理器挂起。处理器时钟时间为10ns，如果指令基本的CPI为1.0(设所有访存均命中Cache)，求在没有远程访问的状态下与有0.5%的指令需要远程访问的状态下，前者比后者快多少?

并行处理面临的挑战

解：有0.5%远程访问的机器的实际CPI为

CPI＝基本CPI＋远程访问率×远程访问开销

＝1.0 ＋ 0.5%×远程访问开销

远程访问开销＝远程访问时间/时钟时间

＝2000ns/10ns＝200个时钟

∴ CPI＝1.0＋0.5%×200＝2.0

它为只有局部访问的机器的2.0 / 1.0＝2倍，因此在没有远程访问的状态下的机器速度是有0.5%远程访问的机器速度的2倍。

并行处理面临的挑战

解决问题的方法：

- 并行性不足： 采用并行性更好的算法
- 远程访问延迟的降低：靠体系结构支持和编程技术
- 例如：减少远程访问频率的措施：
  - 缓存共享数据 (HW)
  - 合理分布数据，尽可能多地访问本地数据 (SW)
- 目前：使用硬件方法，通过使用Cache减少延迟

并行性开发途径：语言、编译、算法、OS等方面

7.6.1 并行算法

- 算法：
  - 求解问题的方法和步骤。
- 并行算法
  - 用多台处理机联合求解问题的方法和步骤。
- 并行算法与串行算法最大的不同：
  - 并行算法不仅要考虑问题本身，而且还要考虑所使用的并行模型，网络连接等。

并行算法的发展

- 基于向量运算的并行算法设计阶段
- 基于多向量处理机的并行算法设计阶段
- SIMD类并行机上的算法设计阶段
- MIMD类并行机上的并行算法设计阶段
- 现代并行算法设计——以MIMD为主，要求可扩展性、可移植性

并行算法分类

- 根据运算的基本对象的不同分为：
  - 数值并行算法（数值计算）
  - 非数值并行算法（符号计算）
- 根据进程之间的依赖关系分为：
  - 同步并行算法（步调一致）
  - 异步并行算法（步调、进展互不相同）
  - 纯并行算法（各部分之间没有关系）。

并行算法分类

- 根据并行计算任务的大小分为：
  - 粗粒度并行
  - 细粒度并行
  - 中粒度并行

并行的粒度越小，就有可能开发更多的并行性，提高并行度，但是通信次数和通信量就增加很多。

- 并行算法分为：
  - 多机并行
  - 多线程并行

并行算法设计

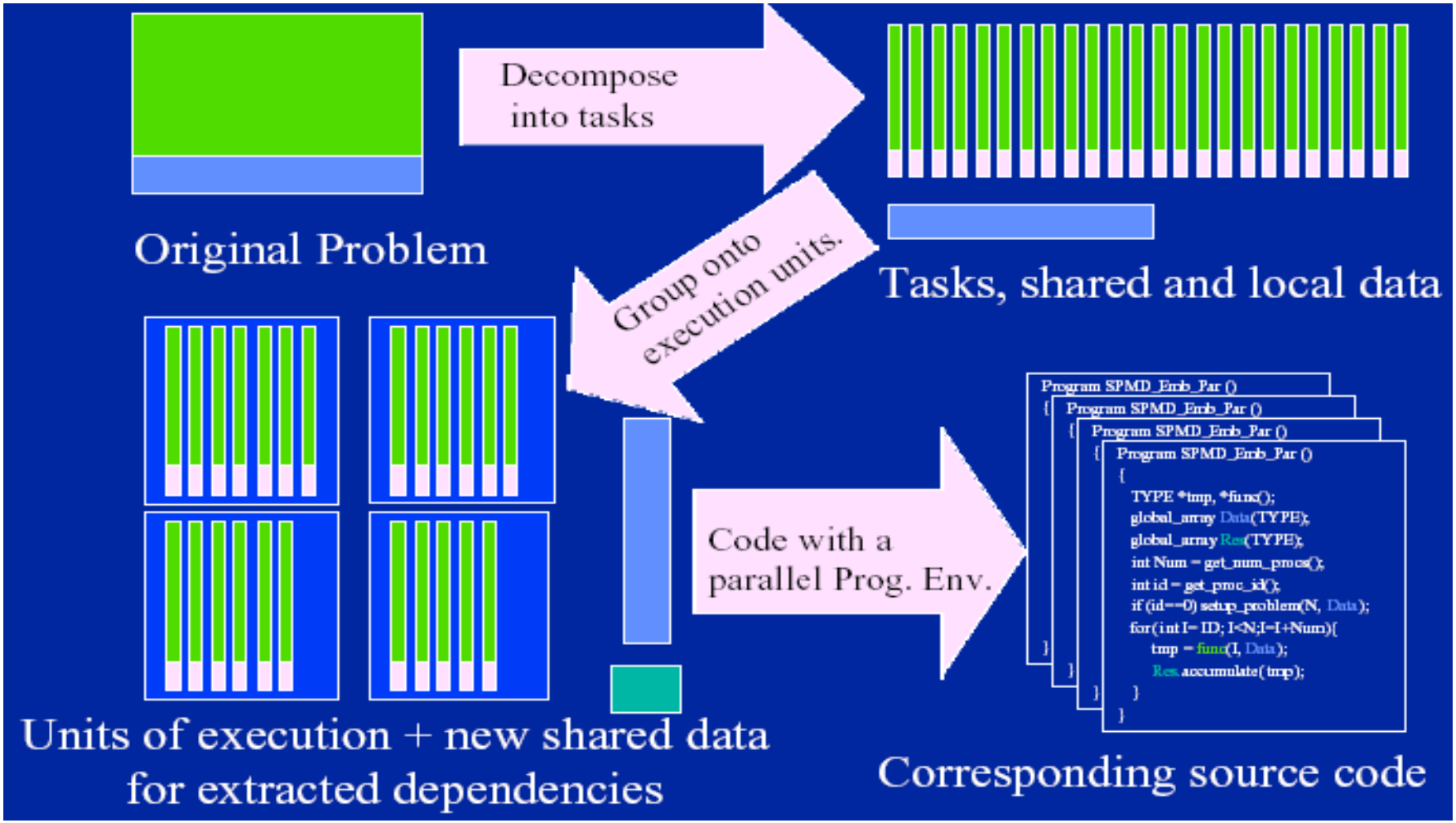
- 并行算法是提高计算机并行性能的关键
- 并行算法设计：
  - 以MIMD 为主
  - 可扩展、可移植
  - 中/大粒度任务级并行
  - 每个进程发挥单机性能 （数据结构、程序设计、通信方式）

并行算法设计

- **并行算法依赖一个简单事实：**独立的计算可同时执行。
- 所谓独立计算是指其每个结果元只出现一次的计算。
- **如何实现并行计算？**

如何实现并行计算?

分而治之!



并行计算基本设计技术(1/3)

- 划分法(Partitioning)
  - 首先，将原问题分成p个独立的近乎大小相等的子问题；其次，用p台处理器并行求解诸子问题。
  - 划分的难点在于要留心分解子问题，使得子问题的解很容易被组合成原问题的解。
  - 例如(m, n)-selection网络。
- 分治法(Divide-and-Conquer)
  - 将原问题规模从大到小逐渐分解成一些特性相同的子问题；直到子问题很容易求解为止。
  - 分治很自然地导致递归过程，其注意力集中在子问题地合并上。
  - 例如FFT的计算。

并行计算基本设计技术 (2/3)

- **平衡树法(Balanced Tree)**
  - 将输入元素作为叶节点构筑一棵平衡二叉树；然后自叶向根往返遍历。
  - 此法的优点是在树中能快速存取所需的信息。
  - 例如数据播送、求最大/最小值以及求和/前缀计算等。
- **倍增法(Doubling)/指针跳跃法(Pointer Jumping)**
  - 使用递归计算，将需要处理的数据间的距离逐步加倍，经k步后就可以完成距离为2^k的所有数据的计算。
  - 此法特别适合于处理以链表或有根树之类为数据结构的问题。
  - 例如表序问题的计算和求森林根等。

并行计算基本设计技术 (3/3)

- **流水线法(Piplining)**
  - 将原任务t分成一系列子任务t_1，t_2，…，t_m，使得一旦t_i完成，后继的子任务就立即开始，并以同样的速率计算之。
  - 例如systolic计算。
- **破对称法(Symmetry Breaking)**
  - 打破某些问题的对称性，使原问题可并行计算。
  - 例如有向环图的顶点着色。

并行算法设计

- 建立并行算法的一种**普遍原则**：
  - 反复将每一计算分裂成具有同等复杂性的两个独立部份，称为**递推倍增法**。
  - 将各部分之间的关联用结点组成的树来描述。**提高并行性就是用交换律、结合律、分配律对树进行变换。**
  - 增大树中每一层的结点数（增大各处理机可并行运行的过程数），降低树的高度（降低多处理机运算的级数）。

并行算法的一般设计方法

- 串行算法的直接并行化
- 从问题描述开始设计并行算法
- 借用已有算法解新问题

并行算法评价

- 性能的参数：
  - P：可以并行处理的处理机数；
  - T_P：P台处理机运算的级数，也就是树高；
  - S_P：加速比，单处理机顺序运算的级数T_1与P台处理机并行运算的级数T_P之比；
  - E_P：效率（P台处理机的设备利用率），$E_P=S_P/P$。

在多处理机上，好的并行算法应当是以提高系统的速度性能为前提，再在此基础上尽可能地提高系统的效率。

7.6.2 程序段间的相关性分析

- 程序段中各类数据的相关是限制程序并行的重要因素。
- 多处理机上的相关 (1/2)：
  - 数据相关：
    - “先写后读”，可以顺序串行，但不能并行
  - 数据反相关
    - “先读后写”，可以顺序串行，不能交换串行，在特殊情况下可以并行。

7.6.2 程序段间的相关性分析

- 多处理机上的相关 (2/2)：
  - 数据输出相关
    - “写–写”，可以顺序串行，不能交换串行，在特殊情况下可以并行。
  - 控制相关
    - 条件语句
  - 相互交换
    - 同时具有“先写后读”和“先读后写”相关，以交换数据为目的

7.6.2 程序段间的相关性分析

相关 执行	数据相关	数据反相关	数据输出相关	相互交换	无相关
顺序串行	√	√	√	×	√
交换串行	有条件	×	×	×	√
并行	×	有条件	有条件	√（完全同步）	√

7.6.3 并行程序设计语言

- 为了加强程序并行性的识别能力，有必要在程序语言中增加能明确表示并发进程的成分，这就是并行程序设计语言。
- 设计并行程序设计语言方法：
  - (1) 重新设计新语言
  - (2) 扩充现有语言功能
  - (3) 不改变现有语言，提供函数库或并行化编译系统

(1) 重新设计新的并行语言

- 可以完全摆脱串行语言的束缚,从语言成分上直接支持并行，这样就可以使并行程序的书写更方便、更自然,相应的并行程序也更容易在并行机上实现。
- **缺点：** 没有统一计算机模型。
- 虽有并行语言，但没有一个被普遍接纳

(2) 扩充现有的串行语言

- 在现有的程序设计语言的基础上扩展出能表示并行进程的语句。
- 若用原来的串行编译器来编译，标注的并行扩充部分将不起作用，仍将该程序作为一般的串行程序处理。
- 若使用扩充后的并行编译器来编译，则该并行编译器就会根据标注的要求，将原来串行执行的部分转化为并行执行。

(3) 提供并行函数库和并行化编译系统

- 为已有的串行语言提供并行运行库。只需要在原来的串行程序中加入对并行库的调用，就可以实现并行程序设计。
  - 如现在流行的MPI（消息传递接口）并行程序设计就属于这种方式。
- 针对以上的方式实现并行语言，一般采用下述集中编译器方法完成并行语言的编译处理：
  - 新语言编译器
  - 预编译处理
  - 并行函数与类库
  - 并行化编译系统

并行程序设计语言关键技术

(1)进程管理

①进程创建与消亡：显式、隐式

显式如FORK-JOIN，隐式如cobegin-coend、parfor等

②进程激活：创建时激活、收到消息时激活

一般采用创建时激活

③进程粒度：一般提供中粒度(2K~10K指令)

进程描述手段(因进程创建代价较大)

并行程序设计语言关键技术

(2)进程通信与同步

通信方式：同步、异步互锁、异步非互锁

通信 — 进程间数据传递，一般采用异步方式(提高性能)

同步 — 进程间协同，一般采用同步方式(提高可靠性)

在多处理机系统中，处理机的数目多少是不会影响程序的编写的，所编写的并行程序可以在机数不同的多处理机系统上通用。

并行程序设计模型

- 3种程序设计模型 (1/3):
  - （1）共享内存模型
    - 多个线程或进程同时运行
    - 它们共享同一内存资源，每个线程或进程都可以访问该内存的任何地方
    - 例如 openMP 就是采用共享内存模型

并行程序设计模型

- 3种程序设计模型 (2/3)：
  - （2）分布式内存模型
    - 多个独立处理结点同时工作，每个处理结点都有一个本地的私有内存空间
    - 进程可以直接访问其私有内存空间。 若一个进程需要访问另一个处理结点处的私有空间，则此进程需要发送信息给那个进程来进行访问
    - MPI 就是采用分布式内存模型

并行程序设计模型

- 3种程序设计模型 (3/3):
  - （3）分布式共享内存模型
    - 整个内存空间被分为共有空间和私有空间
    - 每个线程可以访问所有的共有空间，并且每个线程都有自己独立的私有空间
    - Unified Parallel C 就是采用分割全局地址空间模型

学习内容

- 7.1 多处理机概念
- 7.2 多处理机结构
- 7.3 多核处理器
- 7.4 多处理机的多Cache一致性
- 7.5 多处理机的机间互连形式
- 7.6 程序并行性
- 7.6 多处理机性能
- 7.7 多处理机操作系统

7.7 多处理机性能

- 使用多处理机的主要目的是为了用多处理机并发执行多个任务来提高解题速度。
- 引起峰值性能下降的原因：
  - 因处理机间通信而产生的延迟
  - 一台处理机与其它处理机同步所需的开销
  - 当没有足够多任务时，一台或多台处理机处于空闲状态
  - 由于一台或多台处理机执行无用的工作
  - 系统控制和操作调度所需开销

7.7 多处理机性能

- 任务粒度的大小会显著影响多处理机的性能和效率。
- 并行性在很大程度上依赖于R/C比值
  - R — 程序用于有效计算的执行时间
  - C — 处理机间通信等辅助开销时间
- 通常：
  - R/C比值小，细粒度并行，并行性低
  - R/C比值大，粗粒度并行，并行性高

为获得最佳性能，应对并行性和额外开销大小进行权衡，也要与应用问题的粒度取得适配。

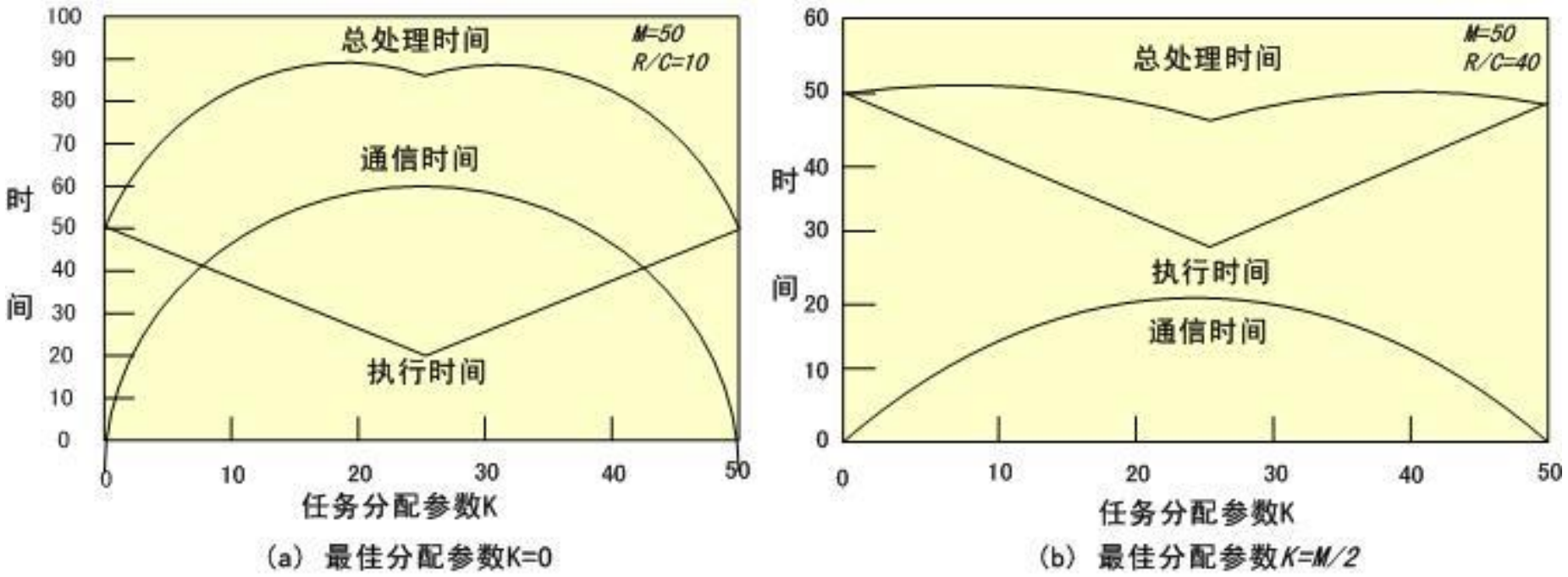
7.7.1 基本性能

- 假设有一个包含 M 个任务的应用程序，希望在一个由 N 台处理机组成的系统上以最快的速度执行这个程序。
- 假设：
  - （1）每个任务的执行时间为R个单位；
  - （2）当两个任务不在同一台处理机上时，其通信所需的额外开销为C个单位时间。当两个任务在同一台处理机上时，通信所需的额外开销为0。

总处理时间＝R•max(M－K，K) ＋ C•(M－K)•K

总处理时间＝执行时间 + 通信和其它额外开销的时间

7.7.1 基本性能



两种不同R/C比值的并行执行时间

对于该模型的结论是：当 R/C ＜ M/2 时，把所有任务分配给同一台处理机能使总处理时间最小；当 R/C ＞ M/2 时，把任务平均地分配给两台处理机能使总处理时间最小。也就是说使任务分配参数 K＝0 或 K＝M/2。

当M为奇数时，应使K尽可能接近 M/2。

7.7.2 N台处理机系统的基本性能

- 我们将第 Ki 个任务分配给第 i 台处理机。基本性能等式可推广为：

$$\text{总处理时间} = R \cdot Max(Ki) + \frac{C}{2}\sum_{i} K(M - Ki)$$

$$= R \cdot Max(Ki) + \frac{C}{2}(M^2 - \sum_{i} Ki^2)$$

为使总处理时间最小：将所有的任务都集中在一台处理机上，或者将任务平均分配给所有处理机。

所谓平均是指如果M是N的倍数，则每台处理机分得M / N个任务，否则除一台处理机外其它处理机分得M / N个任务，那一台处理机分得剩余的任务。

7.7.2 N台处理机系统的基本性能

$$\text{加速比} = \frac{R \cdot M}{\left(\frac{RM}{N} + \frac{CM^2}{2} + \frac{CM}{2N}\right)} = \frac{R}{\frac{R}{N} + \frac{CM\left(1 - 1/N\right)}{2}}$$

$$= \frac{\frac{RN}{C}}{\frac{R}{C} + \frac{M(N-1)}{2}}$$

如果分母中的第一项远远大于第二项，即M，N较小，R / C较大，加速比与N成正比例。如果处理机台数N很大， 则分母主要由第二项决定，加速比与R / CM成正比例，而不依赖于处理机的台数了。

所以处理机的台数不应超过由成本与R / C比值函数所决定的极大值。

学习内容

- 7.1 多处理机概念
- 7.2 多处理机结构
- 7.3 多核处理器
- 7.4 多处理机的多Cache一致性
- 7.5 多处理机的机间互连形式
- 7.6 程序并行性
- 7.6 多处理机性能
- 7.7 多处理机操作系统

7.8 多处理机操作系统

- 由多台计算机协同工作来完成所要求任务的操作系统
  - 负责处理机分配、进程调度、同步和通信、存储系统管理、文件系统与I/O设备管理、故障管理与恢复等
- 按照结构来划分，目前有三种类型：
  - 主从式(Master-slave)
  - 独立监督式(Separate Supervisor)
  - 浮动监督式(Floating Supervisor)

7.8 多处理机操作系统

- 主从式(Master-slave)
  - 由一台主处理机进行系统的集中控制，负责记录、控制其它从处理机的状态，并分配任务给从处理机。
  - 优点：硬件和软件结构相对简单
  - 缺点：对主处理机可靠性要求很高。当不可恢复错误发生时，系统容易崩溃，此时必须重新启动主处理机。系统灵活性差，在控制使用系统资源方面效率也不高。

7.8 多处理机操作系统

- 独立监督式(Separate supervisor) (1/2)
  - 操作系统将控制功能分散给多台处理机，共同完成对整个系统的控制工作。每个处理机均有各自的管理程序(操作系统的内核)。
  - 优点：
    - 每个处理机都有其专用的管理程序，故访问公用表格的冲突较少，阻塞情况自然也就较少，系统的效率较高
    - 每个处理相对独立，因此一台处理机出现故障不会引起整个系统崩溃

7.8 多处理机操作系统

- 独立监督式(Separate supervisor) (2/2)
  - 缺点：
    - 减少了对控制专用处理机的需求，但是实现更复杂
    - 每个管理程序都有一套自用表格，但仍有一些共享表格，从而发生表格访问冲突问题，导致进程调度复杂性和开销的加大
    - 修复故障造成的损害或重新执行故障机未完成的工作非常困难
    - 各处理机负荷的平衡比较困难。

7.8 多处理机操作系统

- **浮动监督式(Floating supervisor)**
  - 系统中每次只有一台处理机作为执行全面管理功能的“主处理机”，“主处理机”可以根据需要浮动，即从一台切换到另一台处理机。这样，即使执行管理功能的主处理机故障，系统也能照样运行下去；
  - 优点：
    - 系统可靠性更强，没有单主处理崩溃瓶颈
    - 更好的平衡处理机负载
  - 缺点：系统实现较复杂

多处理机操作系统的难度

- 处理机的分配和进程调度
- 进程间的同步
- 进程间的通信
- 存储系统的管理
- 文件系统的管理
- 系统重组 等

本章重点

- 多处理机特点及主要技术问题
- 多处理机结构
  - 紧耦合和松耦合
  - UMA、NUMA、ccNUMA和COMA结构及特点
  - MPP、机群(COW/NOW)结构及特点
  - 多核处理器结构及特点
- 多处理机Cache一致性问题及协议
  - 监听协议：写无效与写更新 + 写直达与写回
  - 基于目录的协议

本章重点

- 程序并行性
  - 并行算法及研究思路
  - 多处理机上的相关
  - 并行程序设计模型
- 多处理机的性能
  - 任务粒度与系统性能的关系
  - 多处理机性能的基本模型
- 多处理机的操作系统
  - 3类不同的操作系统的特点及适用场合

第7章 作业7

- 7-2
- 7-6
- 7-7