第八章 线程级并行技术

.

※线程级并行(Thread Level Parallslism)与MIMD

*并行性开发策略：　ILP--Istruction Level Parallslism，DLP--Data Level Parallslism

类型&等级—

类型 \ 等级	细粒度	粗粒度
功能并行	ILP(SISD扩展)	TLP(MIMD)
数据并行	DLP(SIMD)	/
注：等级—用计算-通信比表示，即每次通信完成的工作量		

开发策略—

细粒度并行：时间重叠/资源重复方式，及资源共享方式

如流水线/超标量(ILP)技术、向量/阵列机技术，多线程技术

粗粒度并行：资源共享方式　如多处理机技术

*TLP的开发方法：　(资源共享方式)

单处理器中—多线程技术(共享功能部件[隐藏时延/提高效率])　←细粒度

多处理机中—多处理器/多计算机技术(同时执行线程)　←粗粒度

并行计算机—一组处理单元的集合，相互协作以快速完成大任务

思考：处理单元相互协作的内容、涉及硬件各有哪些？交互的实现策略有？

通信、同步；节点互连、MEM访问、节点交互；　共享MEM、消息传递

※本章主要内容

(1) 引言　　并行系统结构分类，并行处理的挑战

(2) 集中式共享存储器系统结构

结构特征，Cache一致性(实现/协议,多级Cache一致性[×])，示例

(3) 分布式共享存储器系统结构

结构特征，Cache一致性的实现，示例

(4) 同步机制　　同步事件组成，基本硬件原语，锁的实现

(5) 多计算机系统　　MPP，COW(结构/示例)

(6) 存储器连贯性模型(×)　　基本概念，顺序连贯性，宽松连贯性

(7) 并行程序的开发(×)　　并行语言，并行算法，示例

※总体要求

掌握并行系统结构相关概念，了解多处理机相关技术

第1节　引言

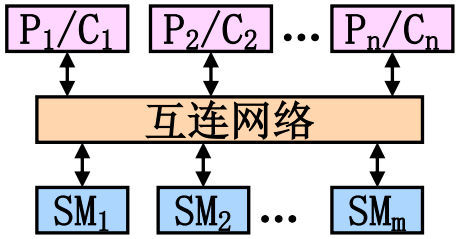
※主要内容：并行系统结构分类，并行处理的挑战

1、并行计算机的系统结构

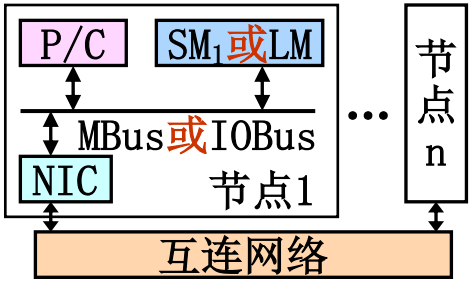
***结构模型：**　(指MEM的互连结构)

又称仓库级计算机　　又称多计算机

		MEM结构		节点耦合程度	结构名称
		位置	地址空间		
并行体系结构	数据并行(SIMD)	集中式	单地址空间	紧耦合	向量处理机，阵列处理机
		分布式			
	功能并行(MIMD)	集中式	单地址空间	紧耦合	对称共享MEM多处理器(SMP)或CSM
		分布式	单地址空间	紧耦合	分布式共享MEM多处理机(DSM)
			多地址空间	紧耦合 松耦合	大规模并行处理机(MPP) 机群(COW)



P—处理器 C—Cache(私有)
SM—共享MEM



NIC—网络接口卡 LM—局部MEM(私有)

SMP—Symmetric shared-memory Multi-Processor
CSM—Centralized Shared-Memory MP
DSM—Distributed Shared-Memory MP
MPP—Massively Parallel Processor
COW—Cluster of Workstation

思考①：集中式MEM为何无多地址空间？

思考②：处理器-处理机的差别？

思考①：多地址空间时，数据访问采用消息传递，需其他P协助访存（打包/解包等），通信效率低；　思考②：节点是否带(主)存储器

*访存模型：（指MEM的访问特征）

UMA--Uniform Memory Access NORMA--No-Remote Memory Access
NUMA--Non-Uniform Memory Access

UMA—各P可访问所有MEM，访问时延相同 ←集中式/单一空间

NUMA—各P可访问所有MEM，访问时延不同 ←分布式/单一空间

NORMA—各P仅访问本地MEM(不能访问远程MEM) ←分布式/多个空间

*通信方式：（指数据的共享方式） 引子：进程间用变量a通信，硬件如何实现？

共享存储器—通过存/取指令访问同一单元实现 ←单地址空间时

（共享地址空间）

```
P1: a=x; /*set a*/   P2: while (flag == 0);
    flag=1;                 b=a; /*use a*/
```

消息传递—通过发送/接收消息实现 ←多地址空间时

```
P1: a=x;    /*set a*/    P2: Recv(1,COMM_2,&data,&stat);
    Send(2,COMM_2,a);           b=data; /*use a*/
```

*同步机制：（指进程间的同步方式）

显式同步(SIMD为隐式同步)，串行访问临界资源

└←相应代码 多P同时执行→┴──────┴→临界区方法不适用

串行访问实现—共享MEM方式使用硬件原语(读-改-写) ←原子性操作

消息传递方式基于按序处理消息 ←常串行接收

*并行计算机结构分类：　--宏体系结构

包含内容—

节点互连：结构模型(MEM结构/节点耦合度)、互连网络

MEM访问：访存模型、Cache一致性模型、MEM连贯性模型

程序中访存的执行次序约定←┴←为提高执行效率

节点交互：通信方式、同步机制

分类结果—

参数 \ 类型		SIMD	SMP	DSM	MPP	COW
结构模型	MEM结构	集中/分布	集中式	分布式		
	节点耦合度	紧耦合	紧耦合			松耦合
	节点规模	近百个	几个	几十个	上万个	
互连网络	类型	定制网络	总线/交叉开关	定制网络		商用网络
	控制方式	集中式	分布式			
访存模型		UMA(单地址)	UMA(单地址)	NUMA(单地址)	NORMA(多地址)	
通信方式		指令(访存/IO)	共享MEM		消息传递	
同步机制		指令级同步	显式同步(MIMD为异步或松同步)			
编程模型		数据并行	共享变量		消息传递	

2、并行处理的挑战

***性能障碍：**

并行性有限—介于串行-全并行之间(~并行算法)

例1：若100个P获得的加速比为10，则程序中串行部分所占%?

解：按Amdahl定律，有10＝1/[s+(1-s)/100]，则s＝9.09%

通信开销较大—受传输协议(同步/异步)、IN时延、通信频率等影响

例2：某DSM仅通信需访问远程MEM，$T_{远程MEM}$＝200ns，通信采用同步传输协议(通信期间双方均等待)；P主频为2GHz、理想CPI为1。若程序执行时有0.2%的指令为通信指令，则实际CPI是多少?

解：Tc＝1/2G＝0.5ns，实际CPI＝1+0.2%*(200ns/0.5ns)＝1.8

***处理方法：**

针对并行性有限—采用更好的并行算法[软]

针对通信开销大—降低远程访问频率、隐藏远程访问延迟

├←缓存共享数据[硬]　└←异步方式通信、多线程[软]

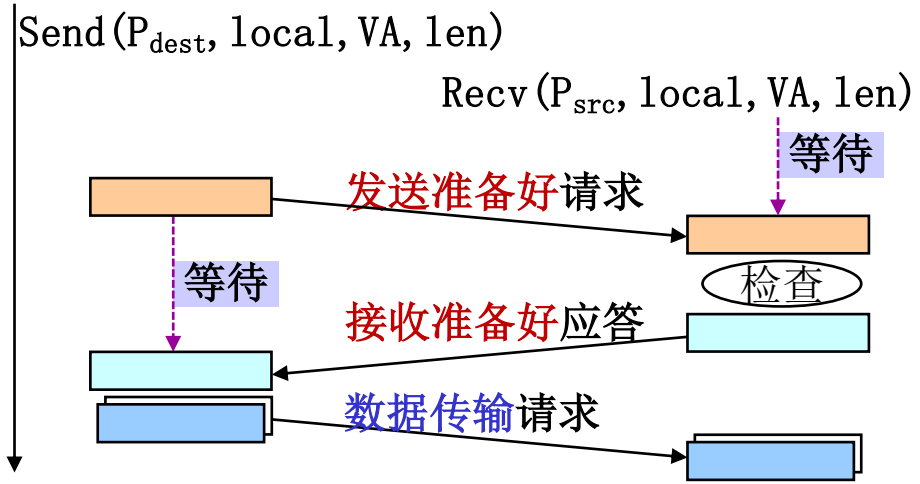
└←优化数据局部性[软]　(避免P挂起)

需保持一致性[硬/软]

消息传输协议的类型—

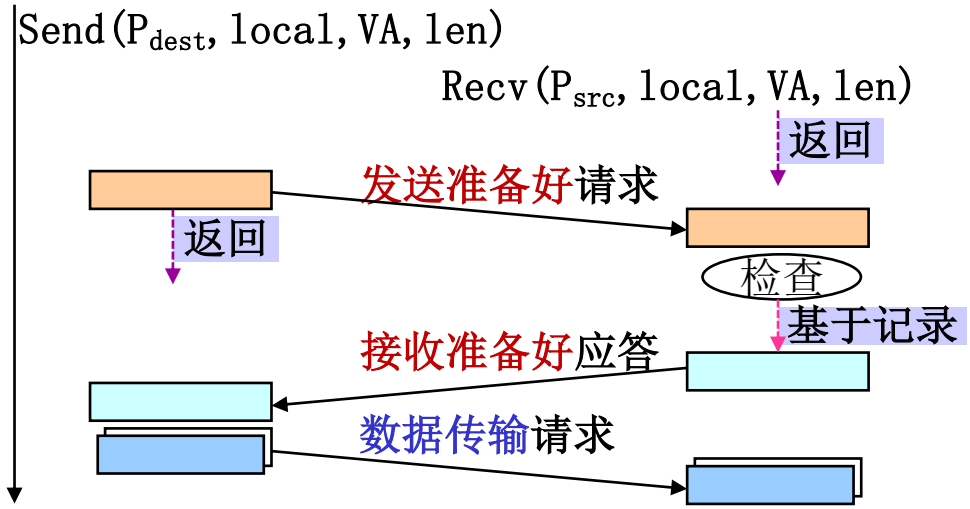
同步协议：

①启动发送
②地址变换、
本地/远程检查
③发送准备好事务
④远程检查条件
⑤应答事务
⑥数据传输事务



异步互锁协议：

①启动发送
②地址变换、
本地/远程检查
③发送准备好事务
④远程检查并记录
⑤接收准备好事务
⑥数据传输事务



异步非互锁协议：缺省异步互锁协议中的③～⑤

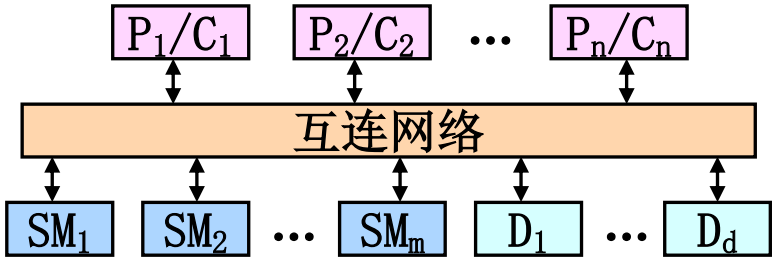
第2节　对称式共享MEM系统结构

※主要内容：结构特性，Cache一致性(协议/实现)，Cache一致性实例

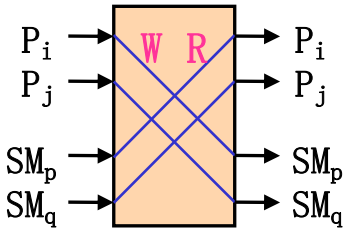
一、SMP系统结构特征　(Symmetric shared-memory Multi-Processor)

└←理论上称为CSM(Centralized Shared-Memory MP,集中式共享MEM多处理机)

*节点互连：节点为CPU或CPU核，IN为对称式网络(P-MEM间)



P—处理器
C—Cache(私有)
SM—系统MEM(共享)
D—I/O设备



*访存模型：UMA方式，时延较大($T_{读/写}=T_{MEM}+T_{IN}$)

性能优化—用Cache缓存私有数据，采用高速IN、紧耦合互连

*通信方式：共享MEM方式，时延较大($T_{读/写}=T_{MEM}+T_{IN}$)

性能优化—用Cache缓存共享数据

└→要求：各P的Cache间保持一致性

*同步机制：显式、硬件同步 (8.4节讨论)

.

二、Cache一致性

1、Cache一致性的概念

指程序中读MEM单元X的结果为最后一次写X的值　←可访问Cache

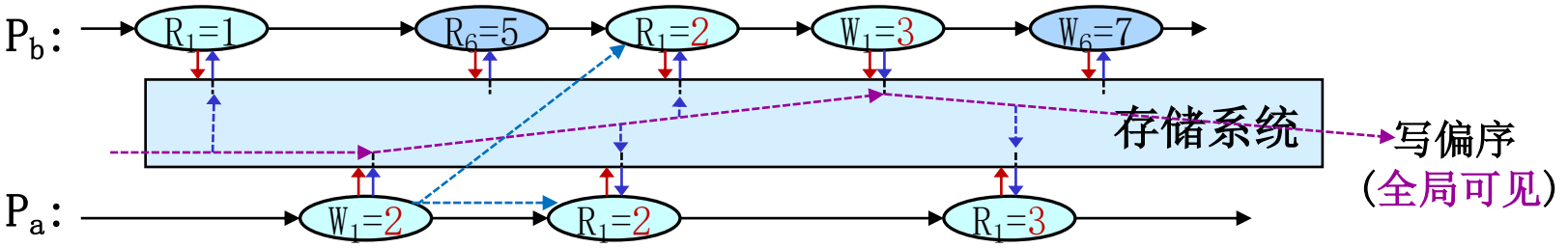
*不一致原因：某P写单元X后，其Cache、主存中为新值，其他Cache中为旧值

单元的本地状态≠全局状态→┘

*一致性的存储系统：任何读X的结果均为最新写入值，满足条件为：

(1)Pa写X后无其他P写X时，Pa及Pb读X均返回Pa写入的值

(2)2个P分别写X时，其他P看到的操作次序相同　←读时返回最新值



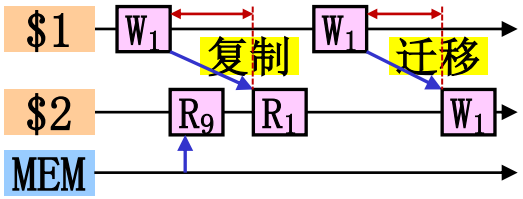
*一致性的特性：写 传 播 —某P写操作的效果对其他P可见

写串行化—各P以相同顺序看到所有写操作

思考：为何读操作没有传播&串行化要求？不影响（只使用）单元状态

2、Cache一致性的实现

*需求分析： Cache间可适时迁移&复制数据 ←尽量不通过主存(延迟&带宽)



| | └←实现写传播(面向后续R操作)

| └←实现写串行化(面向后续W操作) ←基于块

└←其他Cache的下次操作之前

*实现方案： 用一致性协议约定迁移/复制策略，各Cache协同完成

一致性协议— 各Cache跟踪共享块的状态变化，并改变自身副本的状态/数据

(∈分布式算法) 由P操作触发→┘ └←何时&如何

完成要求— 本地$的操作处理与其他$的协同具有原子性

*一致性协议的分类： (如何迁移/复制)

写作废协议— 本地$写入时通知其他$作废该副本

本$拥有唯一副本→┴→其它$之后接收迁移/复制

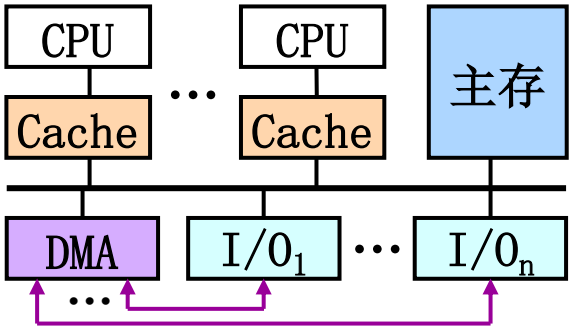
写更新协议— 本地$写入时通知其他$更新该副本

各$均拥有最新副本→┴→其它$之后访问自身

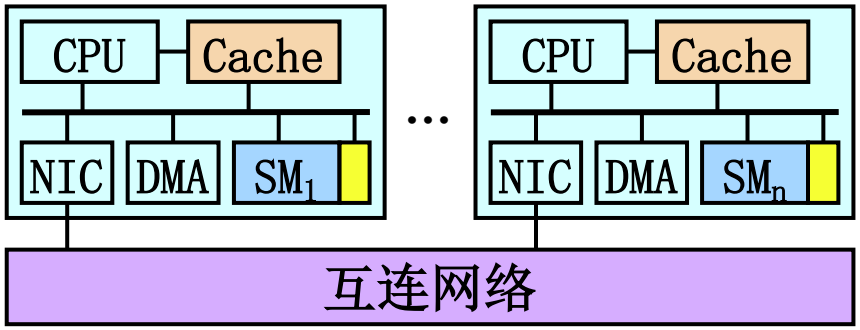
思考：哪种协议更好？ 基于通知&调块的总开销评价。

写作废适于局部性明显，写更新适于写少读多

*块状态的跟踪方法：（基于节点互连方式）



总线互连



网络互连

监听法—（适于总线互连）

块状态的跟踪：本地$广播通知，其余$监听总线

块状态的记录：在各Cache中(分布式)，内容为本节点有/无副本

（行中标记、脏位）

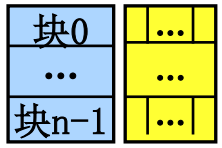
目录法—（适于网络互连）

块状态的跟踪：本地$逐个通知，相关$接收通知

（按照目录）

块状态的记录：在某节点中(集中式)，内容为有副本的节点目录

└←便于查找，如宿主节点



3、VI协议

是一种基于写作废策略的二态一致性协议　　←适于全写法

*Cache块状态：

有效态(V)—干净块(与主存同)，块同时在≥1个Cache中存在

无效态(I)—缺失块(尚未调入)，块在本地Cache中不存在

*Cache响应CPU操作的流程：　(假设采用不按写分配法)

PrRd(读)—命中时完成操作，缺失时先调块

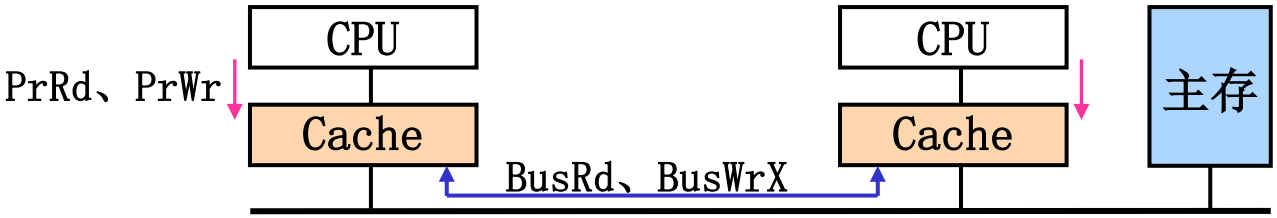
从主存→ ——→产生总线事务

PrWr(写)—命中时写$及主存，缺失时写主存，并通知其他$作废该块

——→产生总线事务

*Cache可见的操作/事务：　① PrRd、PrWr

② BusRd(调块)、BusWrX(写字&通知[互斥写])



*Cache可见操作/事务的处理：

操作—根据块状态&操作类型，产生事务、改变块状态、完成操作

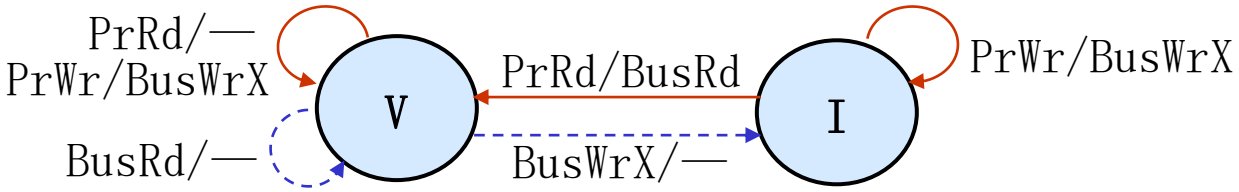
(R缺失时)BusRd←┴→BusWrX(W时)　　└←I→V(R缺失时)

事务—根据块状态&事务类型，响应事务、改变块状态

I	BusRd/BusWrX：	不响应	不变	←不命中
V	BusRd：	无需响应	不变	←主存响应
V	BusWrX：	无需响应	I态(被作废)	←反正作废

数据的复制/迁移—无(R时主存提供数据、W时反正作废)

*协议的状态转换图：



说明：①A/B—Cache观察到A操作或事务时，产生B事务

②实线—处理操作时的状态变化，虚线—处理监听时的状态变化

思考：若调块时产生替换，被替换块状态、调入块状态、Cache行信息有何变化？

被替换块i：V→I　Cache行：有效位=1、Tag=i、缓存块＝i、LRU=3

所调入块j：I→V　　有效位=1、Tag=j、缓存块＝j、LRU=0

变成I态（直接被覆盖[全写法]）

4、MESI协议

是一种基于写作废策略的四态一致性协议　　←适于写回法

└←是三态协议MSI的优化

*Cache块的状态：

修改态(M)—脏　块(异于主存)，块同时在一个Cache中存在

独占态(E)—干净块(与主存同)，块同时在一个Cache中存在

共享态(S)—干净块(与主存同)，块同时在多个Cache中存在

无效态(I)—缺失块(尚未调入)，块在本地Cache中不存在

	某个块在各Cache中的状态枚举(可保持一致性)										
P0 Cache	I	S	S	I	S	E	I	I	M	I	I
P1 Cache	I	S	I	S	S	I	E	I	I	M	I
P2 Cache	I	I	S	S	S	I	I	E	I	I	M

.

*Cache响应CPU操作的流程：

$状态 / CPU操作	命中	缺失	
		有空闲位置	无空闲位置
PrRd(读)	完成读操作	调入目标块；转左列	选择被替换块i；块i为M态时写回主存；转左列
PrWr(写)	通知其他Cache作废目标块、完成写操作		

*Cache可见的操作/事务：（操作来自CPU/自身、事务来自总线）

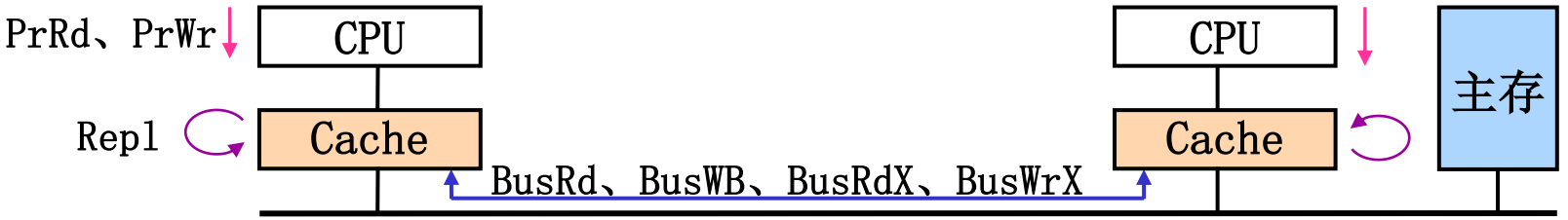
①PrRd、PrWr ←可能产生调入、通知作废等事务

②Repl(替换操作伪定义) ←可能[M态时]产生写回事务

③BusRd(调入)、BusWB(写回)、BusRdX(通知&调块)、BusWrX(通知)

1个事务效率更高←┘ └←写0字节即可

教材中表示为：RdMiss、写回、WtMiss、Invalidate



*Cache可见操作/事务的处理：

操作—根据块状态&操作类型，产生事务、改变块状态、完成操作

BusRdX或BusWrX(W时)←⊥→BusRd(R缺失时)、BusWB(替换时)

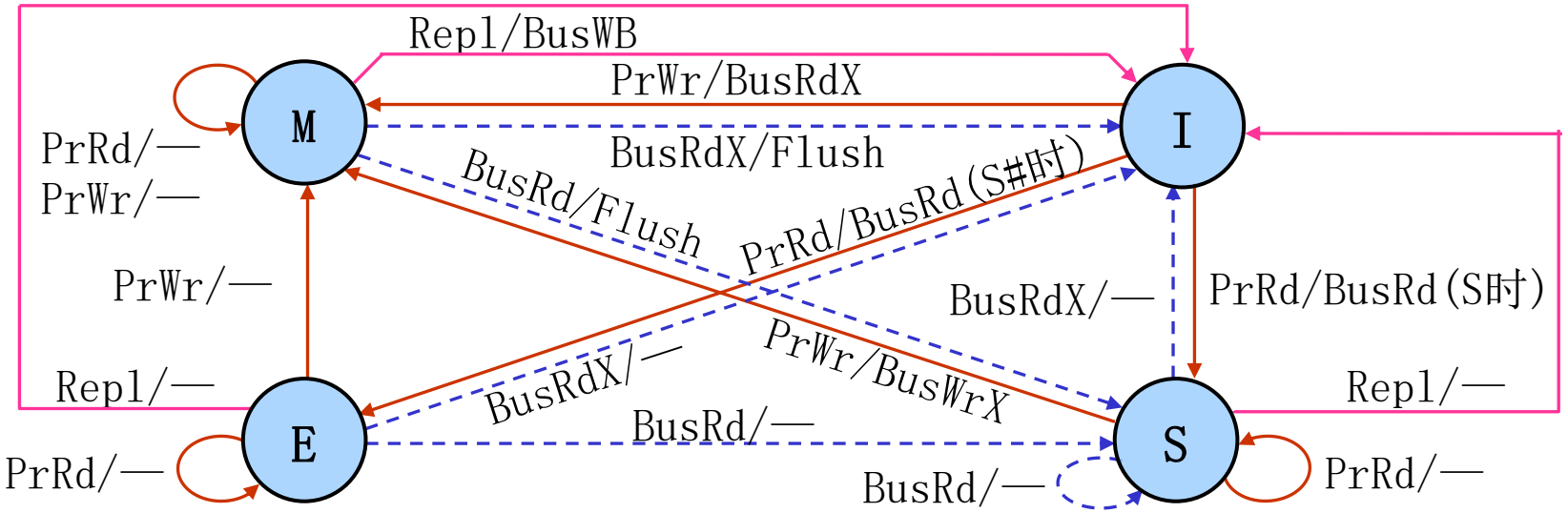
事务—根据块状态&事务类型，响应事务、改变块状态

S/E/M　　BusRd：　无/无/复制数据　S/S/S态　　←M态数据新

S/E/M　　BusRdX：无/无/迁移数据　I态(被作废)　←M态数据新

迁移/复制的需求—M态$提供数据，主存在复制时更新数据

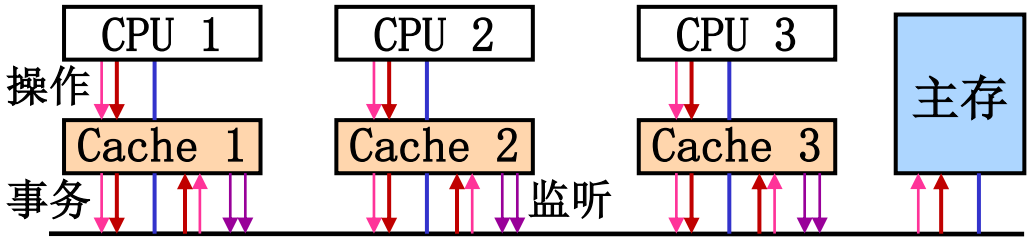
*协议的状态转换图：



注：①实线/虚线—处理操作/监听时的状态变化　②S—块共享信号(监听命中时有效)

③A/B—观察到操作或事务A时，产生事务B　④Flush(冲刷)—响应事务时提供数据

例：写出不同CPU对块u操作时，各Cache的状态转换



操作(PrRd/PrWr)，事务(BusRd/BusRdX/BusWB)，监听状态(命中H、M态命中HM)

Pi操作	Cache1 块u状态	Cache2 块u状态	Cache3 块u状态	产生的总线事务		监听的 块u状态	
	I	I	I	类型	数据提供	H	HM
P1读u块	E	I	I	BusRd	主存	0	0
P3读u块	S	I	S	BusRd	主存	1 ($1)	0
P3写u块	I	I	M	BusWrX	主存	1 ($1)	0
P1读u块	S	I	S	BusRd	Cache3	1 ($3)	1 ($3)
P2写u块	I	M	I	BusRdX	主存	1 ($1$3)	0

思考：从存储系统一致性角度看，Cache写策略与Cache一致性有何不同？

前者为本地$-主存间的一致性，
后者为本地$-其他$-主存间的一致性

※疑问1：SMP的系统结构，只涉及Cache一致性吗？

解答：SMP结构∈宏体系结构，

宏体系结构＝{节点互连、MEM访问、节点交互}

(1)节点互连—P-MEM互连为集中式结构，IN可总线/网络，

P可带Cache　　(组成原理/互连网络已讲过)

(2)MEM访问—MEM空间为单一地址空间，访存模型为UMA，

Cache需实现一致性　　(常规访存+Cache一致性协议)

(3)节点交互—通信方式 为共享MEM　　(用存/取指令实现[不用讲])

同步机制 为显式同步　　(用读-改-写指令实现[稍后讲])

示例：通过a通信、通过flag同步

P1	P2
a=x /*set a*/	while (flag==0);
flag=1;	b=a; /*use a*/

※疑问2：SMP上如何执行并行程序？

解答：①OS提供软硬件管理的API

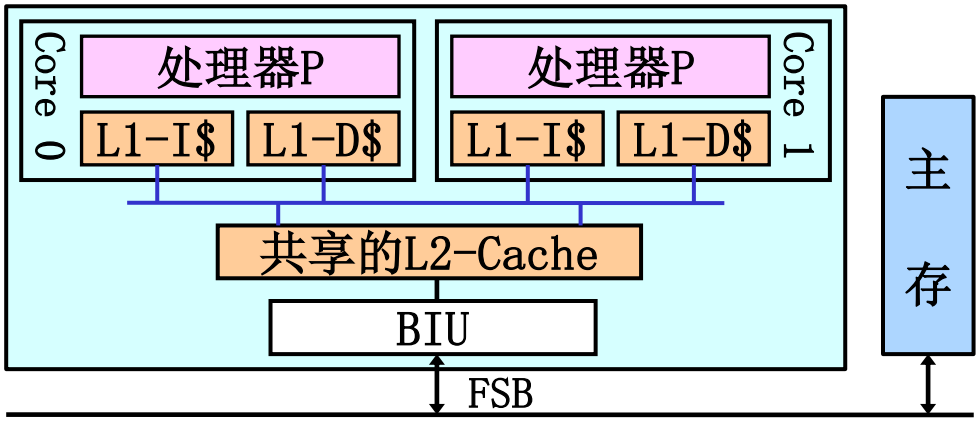
└←线程的映射/调度有变化(带P#)

②OS在某个核上运行

三、SMP举例　　(以Intel Core 2 CPU为例)

*节点间互连：2个IA内核，集中式共享MEM，IN拓扑结构为总线

(L2$及主存)



特点—可扩展性较差(集中式MEM)，访存性能较差(总线拓扑)

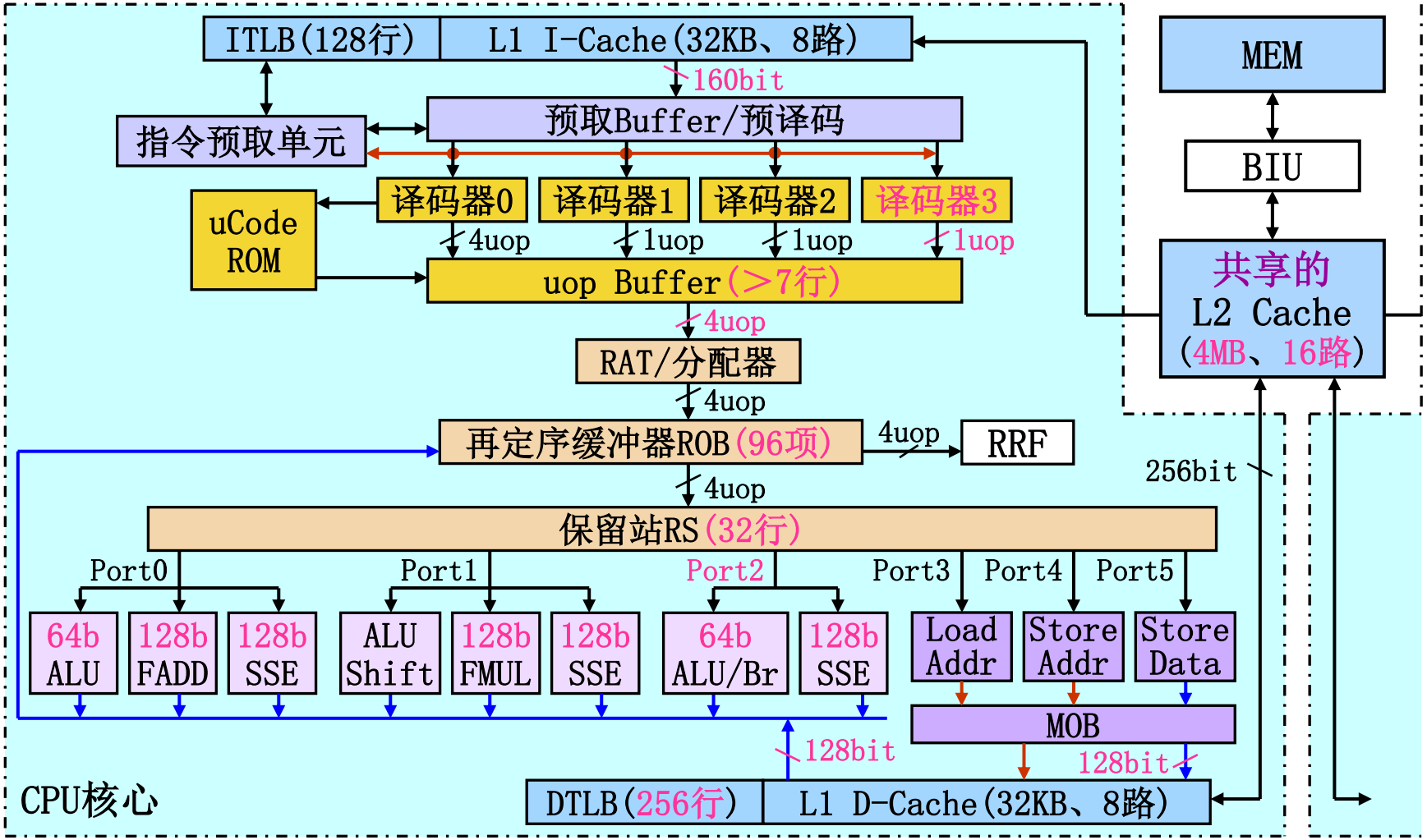
*存储器访问：UMA方式，支持Cache一致性(MESI协议)

*节点间交互：通信方式—共享MEM

同步机制—显式、硬件同步(互斥/事件/栅障)

*内核结构：（微架构Yonah→Core→Penryn）

4路超标量流水，动态执行，SSE4指令



回顾：处理访存冲突、动态执行、超标量、Cache写操作流水化的技术、融合

高级智能Cache— 各Core共享L2 Cache ←总线(如何提高带宽?)

(面向多P) └→$T_{通信}=T_{L2\$}$或$T_{MEM}$

智能内存访问— Cache设置预取器(3*2个+1*2个), ←降低F

(面向单P) 采用内存消歧技术 ←减少访存串行化

└←改进MEM连贯性模型，如W-R的R可提前 (稍后)

宽位动态执行— 4路超标量，宏指令融合＋微指令融合

(面向单P) (Core为3路) (增加指令功能&OPD个数，CO层)

└→Core i7改为AVX，ISA层)

高级数字媒体增强— 3个128位SSE ←提高并行度/吞吐率

(面向单P) (Core为2个64位)

智能功率能力— 能独立控制各核的电压及时钟频率

(面向单P) | (绿色能耗) (睿频)

└←基于当前任务队列 (CO-OS协同)

.

第3节　分布式共享MEM系统结构

※主要内容：结构特性，Cache一致性，SMP/DSM示例

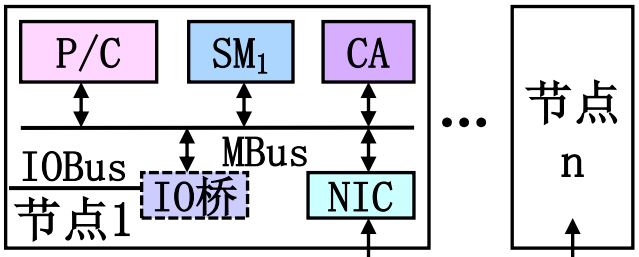
一、DSM系统结构特征

*节点互连：节点为CPU+MEM，

IN可为非对称式网络(P-P间)

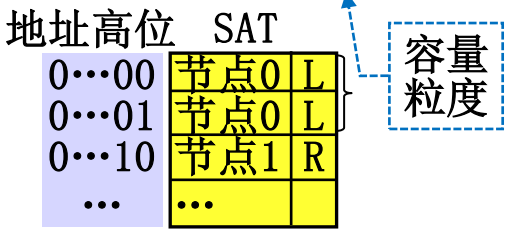
*访存模型：NUMA方式　←NIC应紧耦合互连

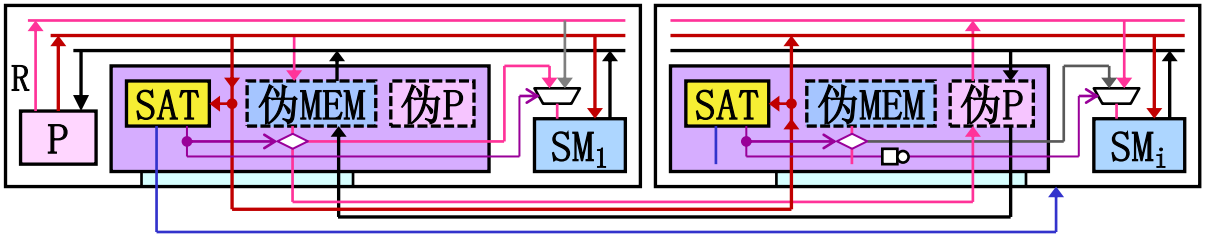
访问实现—　(CA负责)



CA维护MEM全局地址表SAT，有伪P和伪MEM功能

地址: 高位 | 低位L　(LM→SM，$S_{LMi}=k*2^L$)　(处理和组织远程访问)





*通信方式：共享MEM方式，共享数据可以缓存在Cache中　←无需再讲

*同步机制：显式、硬件同步　←稍后讨论

引子：DSM除Cache一致性、同步机制外，其余应该都掌握了

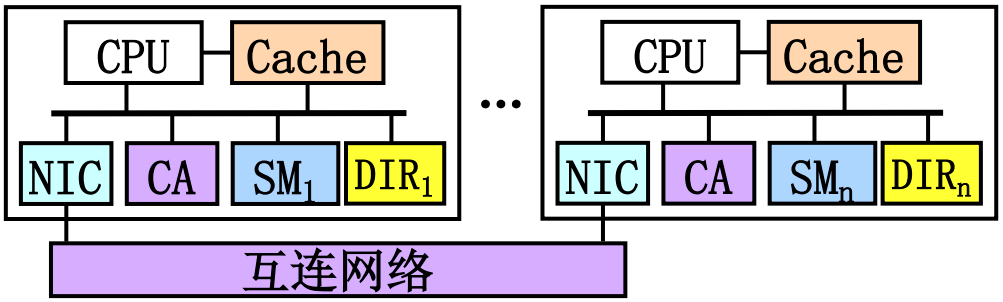
二、Cache一致性

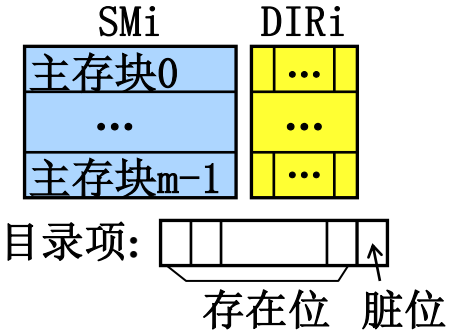
1、Cache一致性的实现

*实现方案：用一致性协议约定迁移/复制策略，各Cache协同完成

*一致性协议的类型：写作废协议，少见写更新协议(开销大)

*块状态的跟踪方法：目录法 ←DIR_1~DIR_n构成SM的目录





目录项组成— 有本主存块副本的节点，常采用位向量方式

思考：怎么知道脏位的节点号？ 存在位=1的节点仅1个

目录的存放— 在宿主节点中，或指定节点中

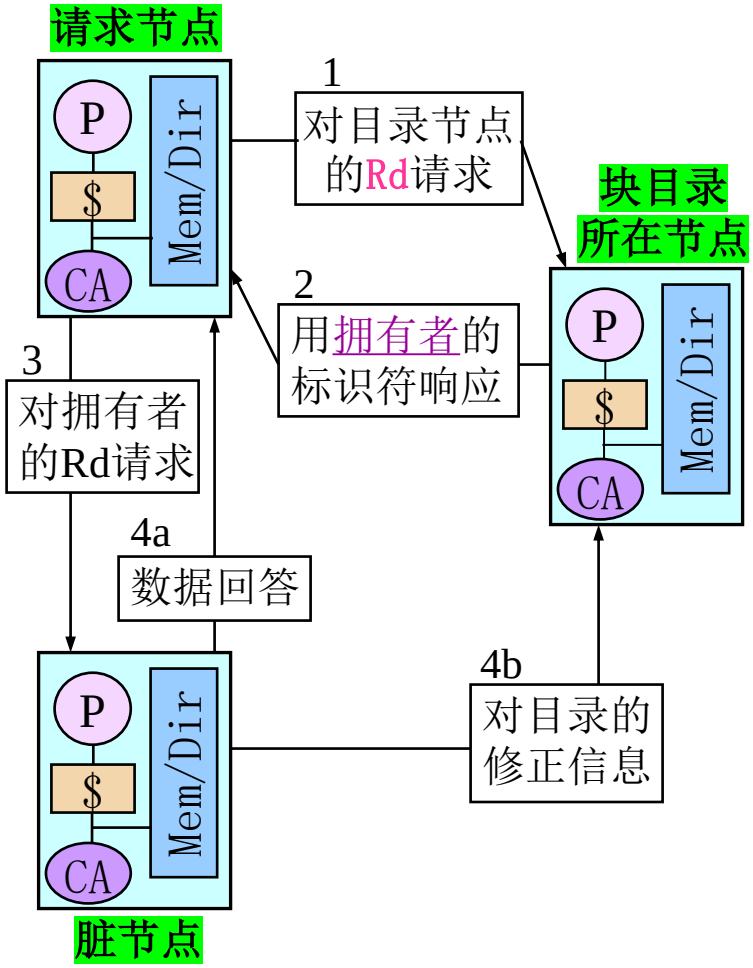
└←便于查找 └←利于管理、易拥塞

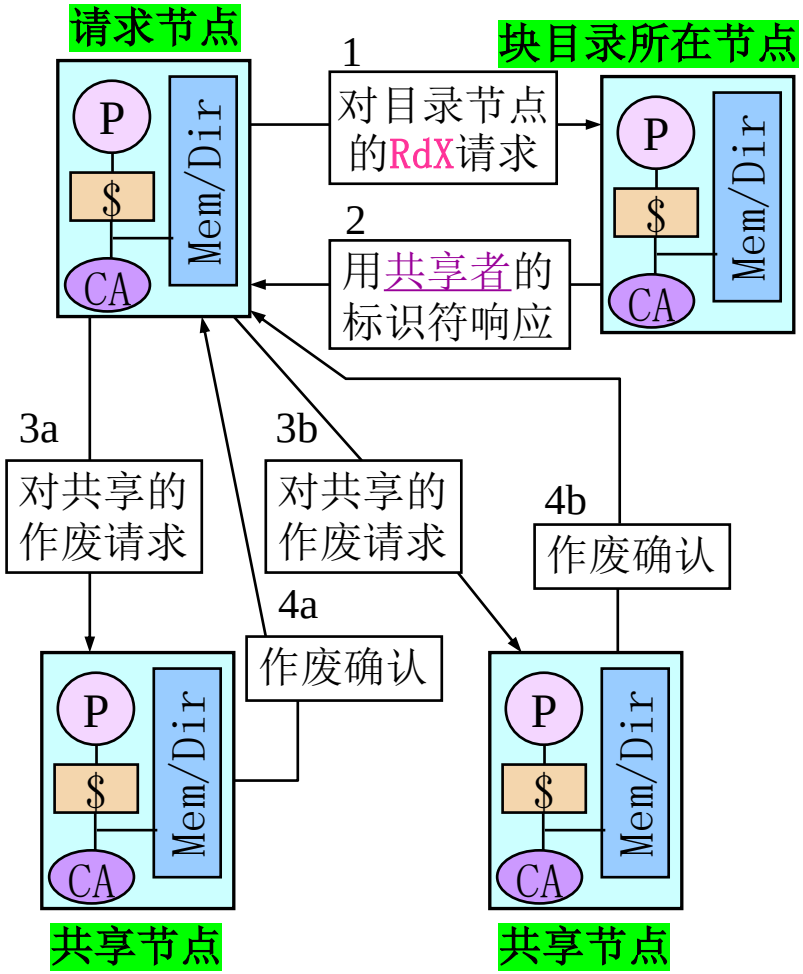
状态的跟踪— 本地Cache逐个通知相关节点(根据目录)，

相关节点接收通知并响应

*Cache相应CUP操作的过程：

获取目录、通知其他Cache、处理操作

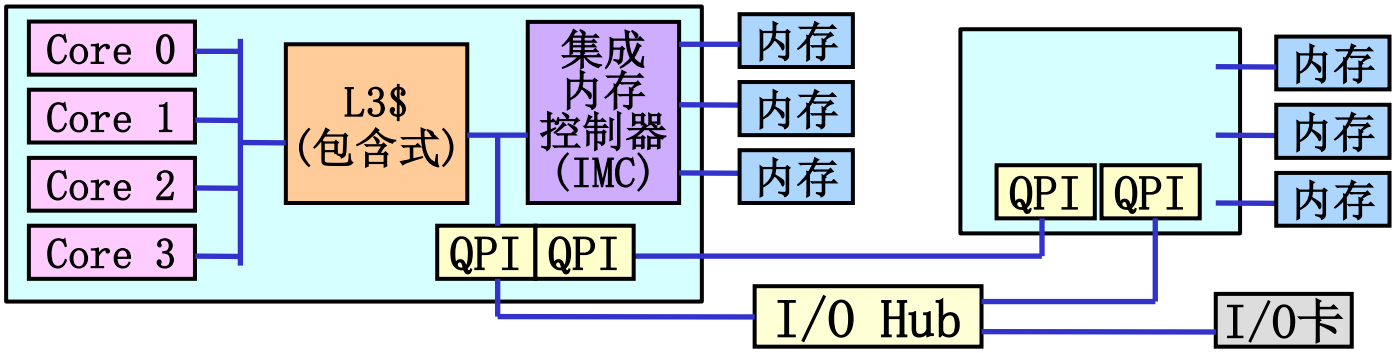




思考：拥有者=目录节点时的处理？

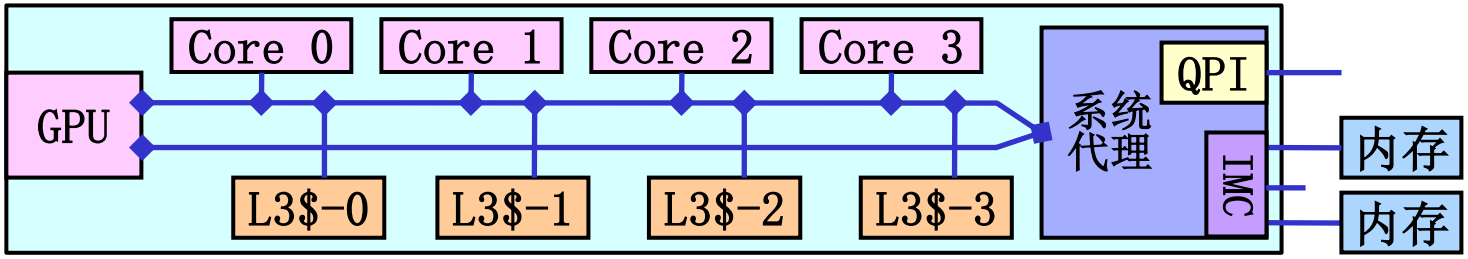
三、DSM举例　　(以Intel Core i7 CPU为例)

*节点间互连：4个IA内核＋GPU(2代起)；分布式共享MEM(2代起)；
节点内IN拓扑结构为总线(1代)、环形总线(2代起)，
节点间IN为QPI网络(拓扑结构可选)



注：QPI—Quick Path Interconnect(快速通道互连)，一种高速串行总线

环形总线—静态网络(开关为3×3)，函数f(i)=i±1，分布式控制

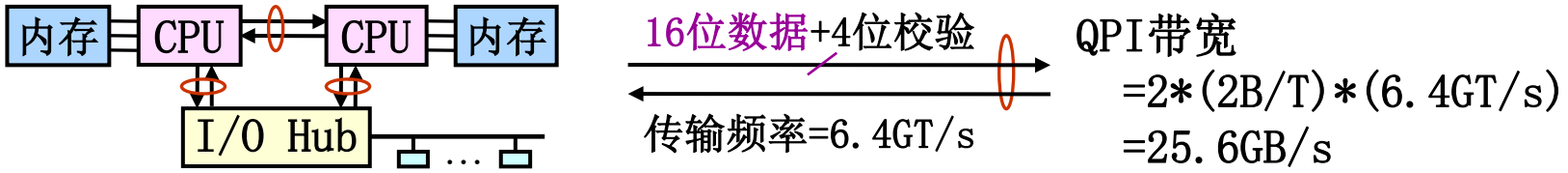


注：GPU—图形处理器，系统代理—管理非核心部件(相当于北桥)

QPI总线—同步串行总线，包传输、点对点链接(线路交换)

└←频率高(无线间干扰)、距离长(差分采样)　←组成原理讲过

配置：全双工、单向20条，数据包为80位($T_{传输}$=4CLK)



特点：带宽高、效率高(多条总线)、通信方便(不经过芯片组)

*存储器访问：NUMA方式(2代起)，支持Cache一致性(MESI协议)

集成MEM控制器(IMC)—CPU直接连接MEM(3通道、DDR3)

特点：延迟低(多为本地访问)、带宽高(不受限于总线)

NUMA实现：OS/硬件管理全局地址表，QPI网络实现远程访问

包含式L3-Cache—各内核L2$是L3$的子集

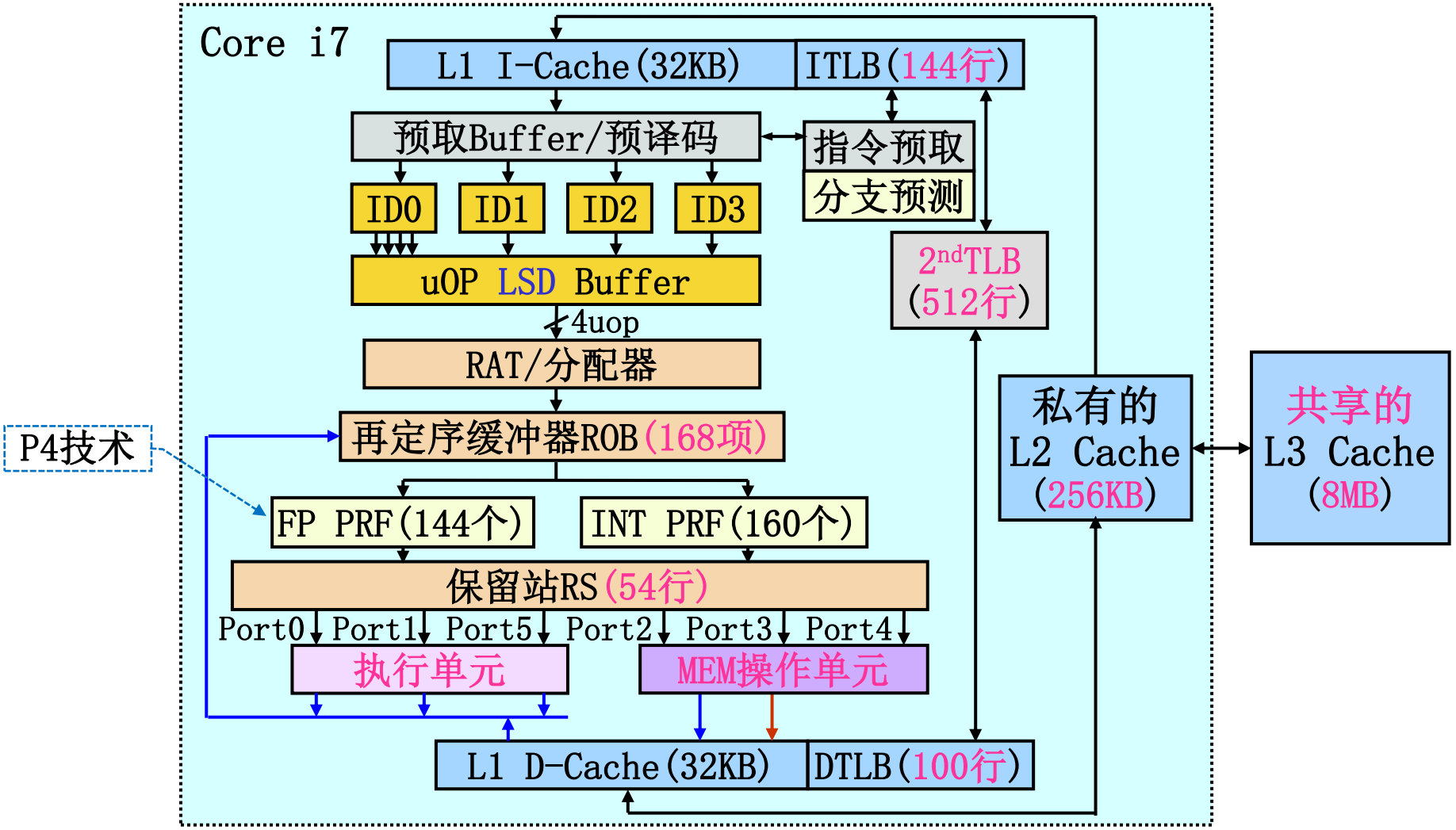
共享状态表示：4位包含位(对应各核)　←减少L3$→L2$传播

替换算法：基于传统(LFU/MRU等)、内容(Size/LRU-MIN)、代价

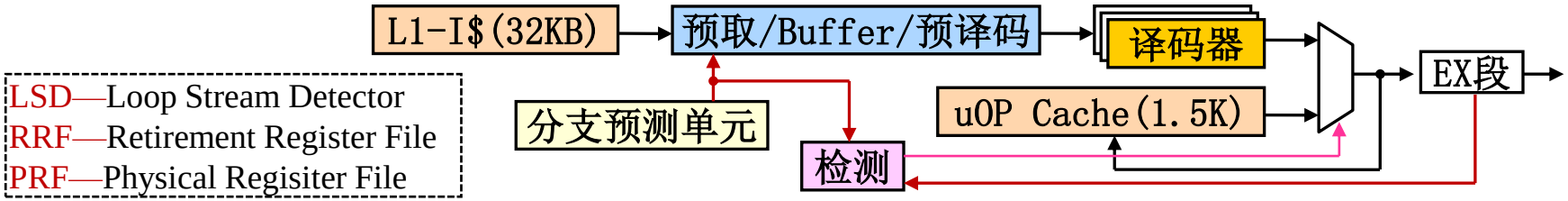
*节点间交互：　同Core 2

*内核结构： （微架构Nehalem→Sandy Bridge→Haswell→Skylake）

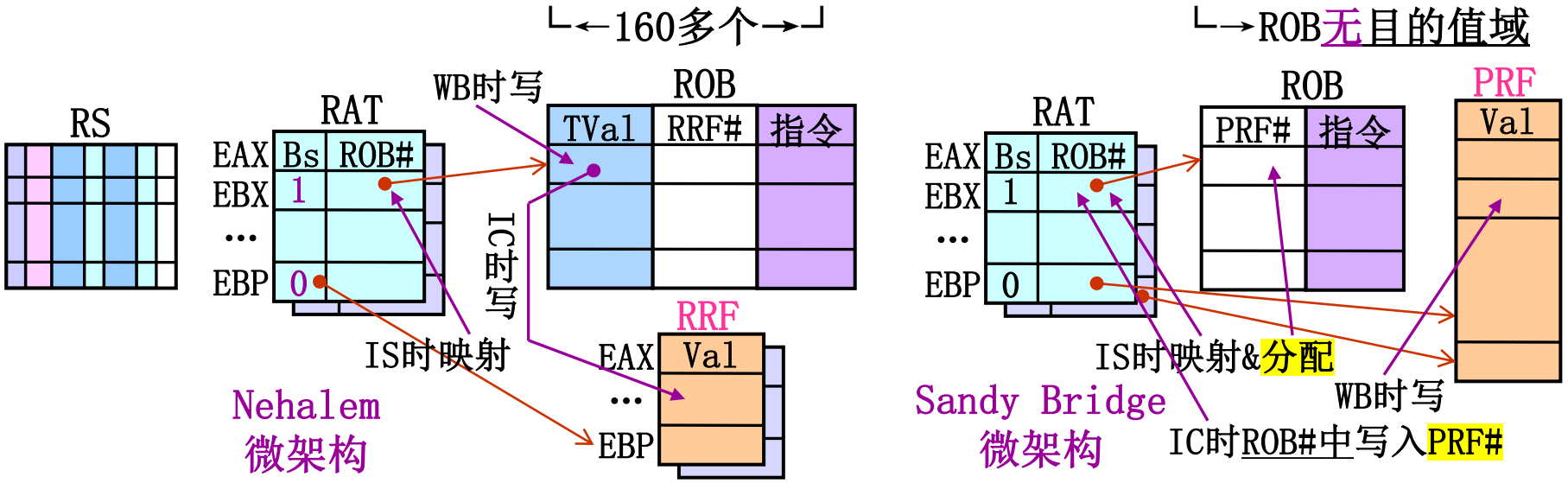
4路超标量流水，动态执行，AVX指令(≥2代)，SMT



LSD Buffer—避免取指&译码，降低时延(H≈80%)及功耗



PRF—代替RRF(个数更多)，降低WAR&WAW，确认段无数据移动



AVX(Advanced Vector Extensions)—256位向量、VEX编码、3/4个OPD、数据重排、不对齐访存

第4节　同步机制

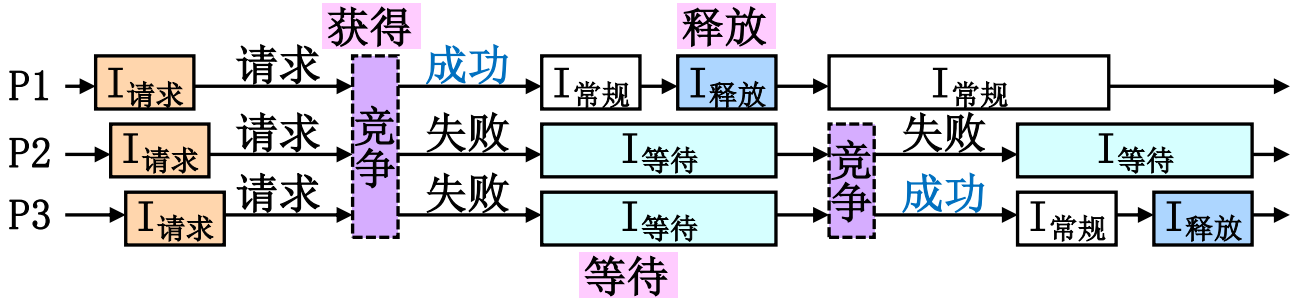
※主要内容：同步事件组成，基本硬件原语，锁的实现方法

1、同步事件的组成

*事件类型：互斥(排他访问)，事件(数据同步[点点])、栅障(控制同步[全局])

资源状态表示—MEM单元中信息，如互斥0/1(闲/忙)、共享0/x(无/有)

*事件组成：同步权的获得方法、等待算法、释放方法



思考：单P中竞争如何实现的？适于多P吗？

P/V操作，代码执行有原子性(轮流执行)
不适用，　指令功能有原子性(并行执行)

*实现策略：获得(有竞争)用硬件实现，其余(无竞争)用软件实现

	API	获得(硬)	等待(软)	释放(软)
互斥(竞争式)	Lock()、Unlock()	读-改-写	等待/阻塞	写
事件(主从式)	Signal()、Wait()	从:读-改-写		主:写
栅障(等待式)	Barrier()	读-改-写		/

2、基本硬件原语　　—面向多P的同步

指具有读-改-写功能的原子操作，是多P同步的基本操作

*原语类型：（示例为获得代码[硬件原语带底纹]，M[lock]=0表示空闲）

原子交换—Ri←1; M[lock]←→Ri; //Ri=0时获得，未获得时写入值相同

测试&置定—Ri←M[lock], if(M[lock]=0) M[lock]←1; //Ri=0时获得

LL/SC—（Load Linked/Store Conditional，链接读/条件写）

```
Rj←1;
Ri←M[lock]; //LL指令触发写检测，Ri=0时才应SC(可能成功)
if(M[lock]未写过) {M[lock]←Rj, Rj←1} else Rj←0;
                    //SC指令检测写状态，Rj=1时获得
```

取并加1—Ri←M[lock1]++; //lock1为请求号，

Rj←M[lock2]; //lock2为服务号，释放时++

//Rj=Ri时获得(类似银行排队)，适于DSM

*应用选择—1~2种即可，基于节点互连结构、等待算法选择

(SMP/DSM)　(等待/阻塞)

3、锁的实现方法

***简单的软件锁：** —适于SMP

获取方法— 各P执行硬件原语(如Test&Set)，成功者获得同步权

等待算法—

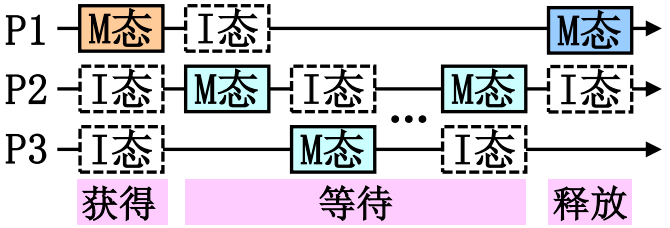
释放方法—

```
        mov   $1, #1     //R1←1
Lock:   t&s   $1, var    //R1←→var，R1=0时成功(≤1个)
        bnz   $1, Lock   //原var=1时获得失败，等待
        ret              //原var=0时获得成功
Unlock: st    var, #0    //var←0(常规写)，释放控制权
        ret
```

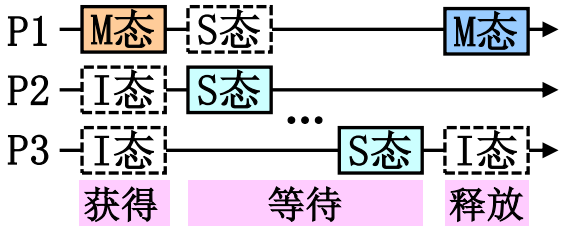
性能分析— 总线流量大(每条t&s产生)，

可扩展性差(流量几何↑)，

公平性差(不确定)



性能优化— 锁空闲时才尝试获得(减少MEM的写操作)




```
Lock:   test  var, #0    //比较var=0?
        bnz   Lock       //var=1时锁不空闲，等待
        t&s   $1, var    //R1←var、var=0时var←1
        bnz   $1, Lock   //原var=1时获得失败，等待
        ret              //原var=0时获取成功
```

*高级的软件锁： —适于DSM

目标—锁被释放时，只有1个P去尝试获得(如票锁[加号锁])

```
锁结构：  struct node {
            int T;        //票号，   申请时修改(+1)
            int S;        //服务号，释放时修改(+1)
          } lock;
```

获取方法—各P执行硬件原语(如Fetch&increment[取并加1])申请，票号＝服务号时获得

```
myTicket = fetch&inc(lock.T);   //请求(取T)
```

等待算法—

```
while (myTicket != lock.S);     //等待(等S)
```

释放方法—

```
lock.S++;
```

性能分析—总线流量小，可扩展性好，公平性好

(等待时不产生) (流量线性↑) (FIFO)

*基本设计思路：获得原语 基于应用特性&机器结构选择，

等待算法 考虑总线流量&可扩展性 ←策略常为忙等待

第5节　多计算机系统

※主要内容：MPP，COW，实例

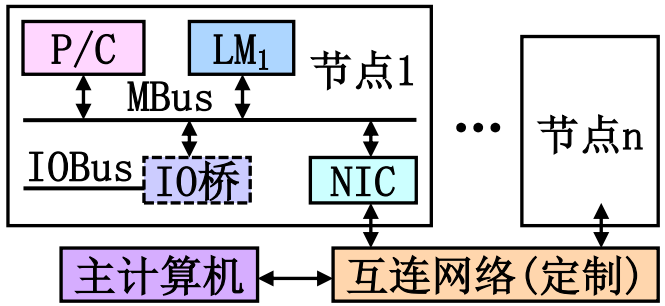
多计算机系统—指MEM非共享(均为分布式)，节点互连可为紧耦合/松耦合

一、大规模并行处理机 --MPP(Massively Parallel Processor)

*节点互连：节点为CPU+MEM，

└←MEM分布非共享

节点紧耦合互连(IN为定制网络)



*访存模型：NORMA(No-Remote Memory Access)

应用需求—MEM中需驻留部分OS　←减少远程访问，常无I/O设备

*通信方式：消息传递方式　←常采用异步传输协议(隐藏时延)

*同步机制：显式、软件同步(各节点发送消息串行访问临界资源)

串行访问实现—资源所在节点按序处理消息　←常串行接收

*可用性管理：系统软件/OS需提供单一系统映像(SSI)　←主计算机中

SSI包含内容：单一的登录、文件系统、作业管理系统、编程模型

└←相对于单P系统

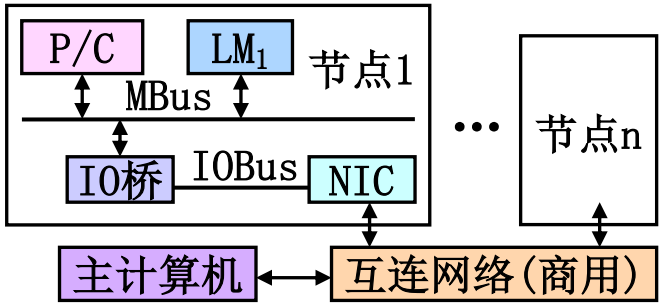
二、机群 --COW(Cluster Of Workstation)

1、系统结构

*节点互连：节点为CPU+MEM+I/O，

└←MEM分布非共享

节点松耦合互连(IN为商用网络)



*访存模型： NORMA

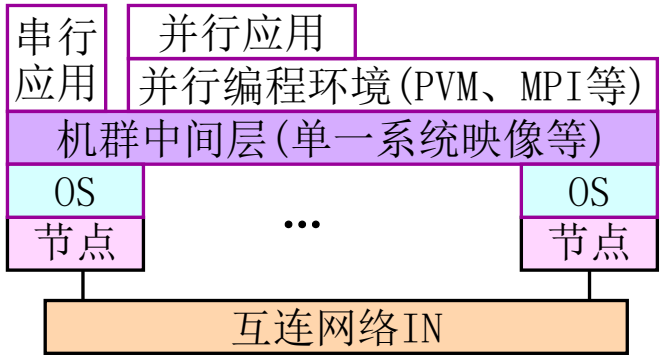
应用需求—磁盘中需驻留全部OS ←减少远程访问

*通信方式： 消息传递方式 ←同MPP

*同步机制： 显式、软件同步 ←同MPP

*可用性管理： 集群管理系统提供SSI

(系统软件)



第八章小结&思考

(1)并行计算机的定义？宏体系结构包含的内容、常见分类？

(2)SMP的结构特征？如何优化访存性能？

(3)存储系统一致性的特征？Cache一致性的实现方法&要求？Cache一致性协议的思想、状态跟踪方法、副本改变方法？MESI协议适用的Cache写策略？

(4)相对于SMP，DSM的结构特征、Cache一致性实现有何不同？

(5)同步事件的类型、组成？基本硬件原语的功能？基于Test&Set的锁实现？

(6)MPP的结构特征与DSM有何不同？与COW有何不同？同步实现的保证？

(1)一组相互协作的处理单元集合，节点互连＋MEM访问＋节点交互，SMP/DSM/MPP/COW

(2)集中式MEM/对称式IN、UMA访存、共享MEM通信、显式硬件同步；用Cache缓存数据

(3)写传播+写串行化，各Cache协同实现一致性协议，操作处理-Cache协同具有原子性；各Cache跟踪块状态变化、改变状态/数据，监听法/目录法，写作废+写更新，写回

(4)分布式MEM/非对称式IN、NUMA访存，跟踪采用目录法、通知逐个进行

(5)互斥/事件/栅障，获得-等待-释放，读-改-写，

获得&等待：①L1: Test&Set R1,lock ②bnz L1 ③ret(获得) 释放：ST lock,#0

(6)NORMA访存、消息传递通信、显式软件同步，节点间松耦合互连，处理消息串行化