并行计算硬件，性能评测

于策

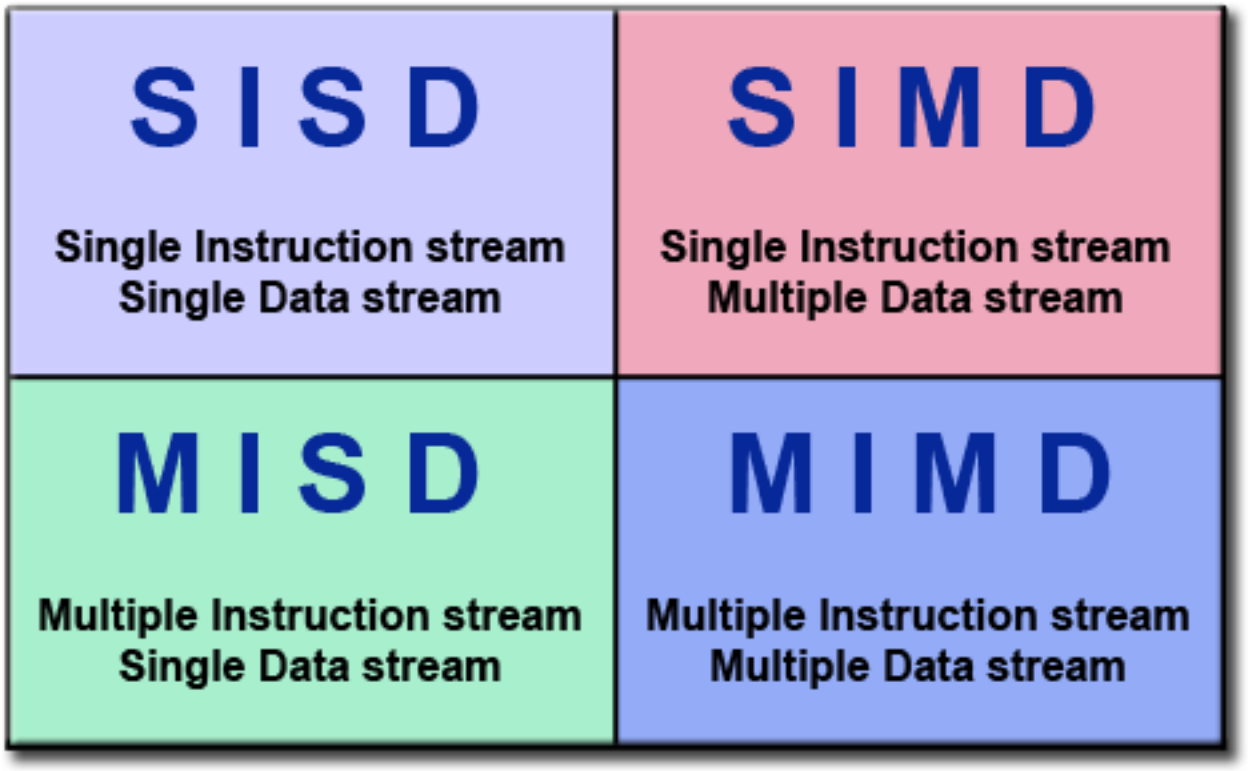
Outline

- 并行计算机系统结构
  - Flynn分类
  - 互连网络
  - 并行计算机结构模型
- 多核处理器/GPU
- 并行计算存储访问模型
- 并行计算性能评测

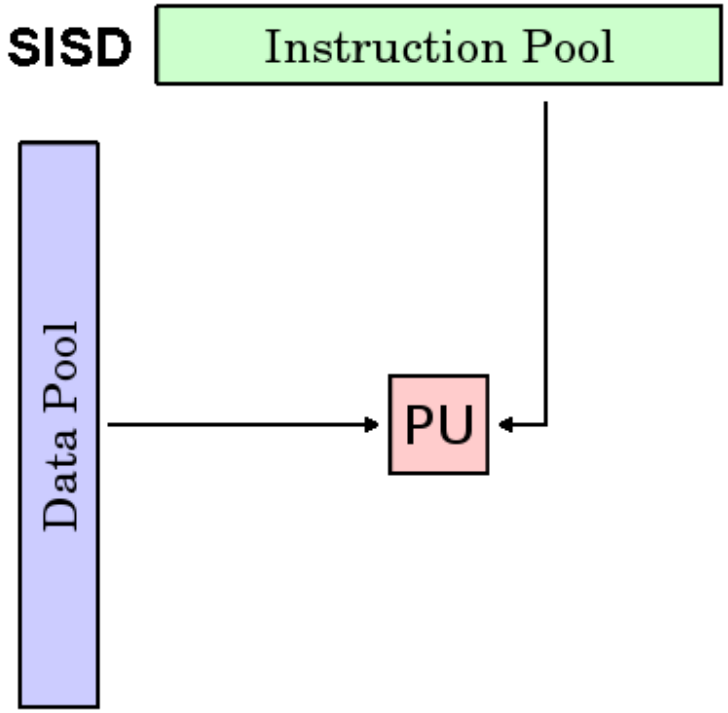
Outline

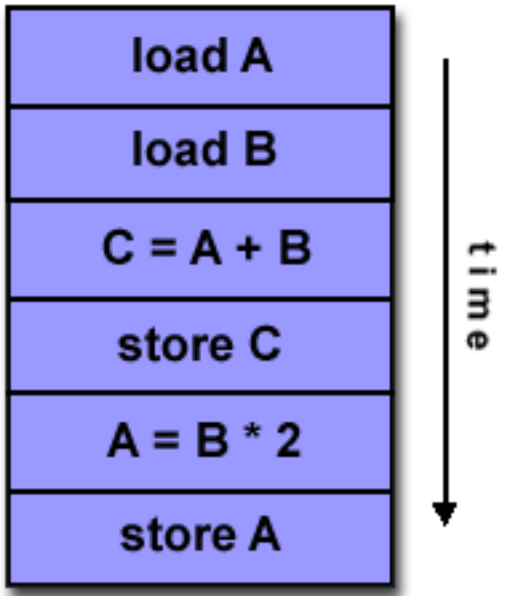
- 并行计算机系统结构
  - **Flynn分类**
  - 互连网络
  - 并行计算机结构模型
- 多核处理器/GPU
- 并行计算存储访问模型
- 并行计算性能评测

Flynn分类

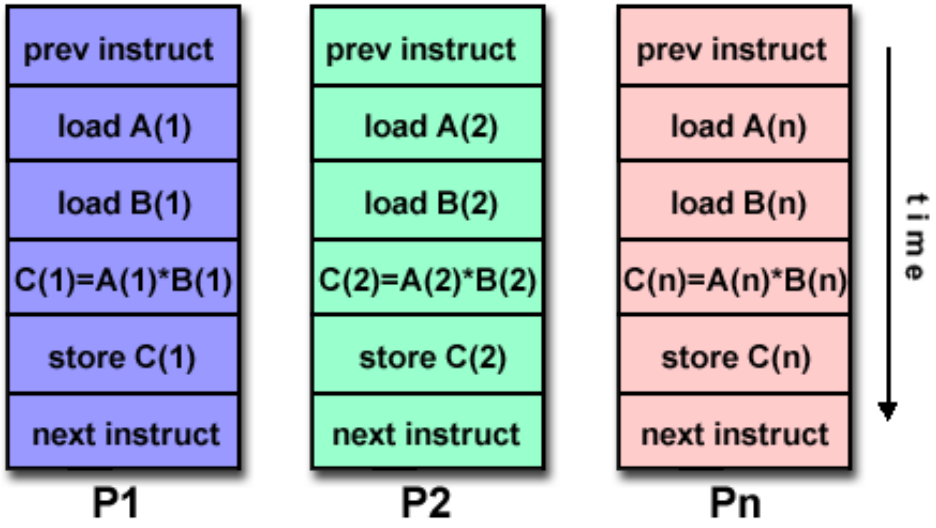

S I S D
Single Instruction stream
Single Data stream
S I M D
Single Instruction stream
Multiple Data stream
M I S D
Multiple Instruction stream
Single Data stream
M I M D
Multiple Instruction stream
Multiple Data stream

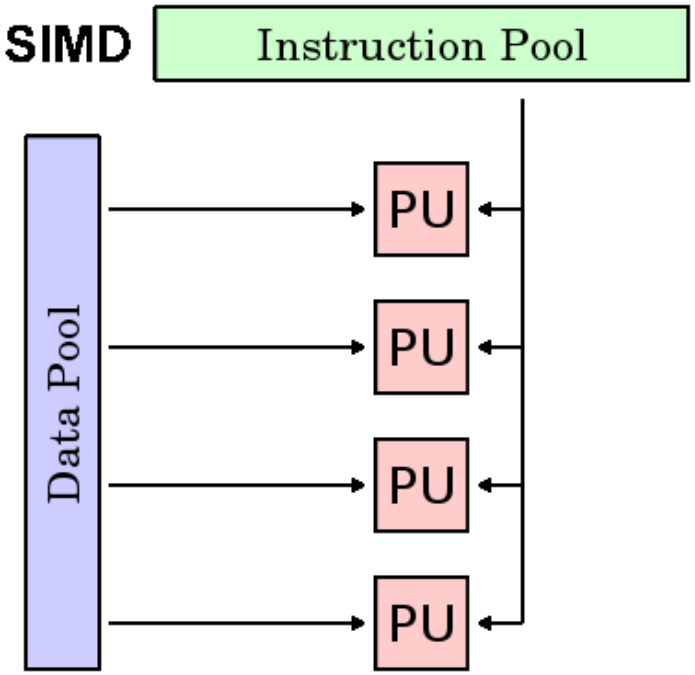
SISD

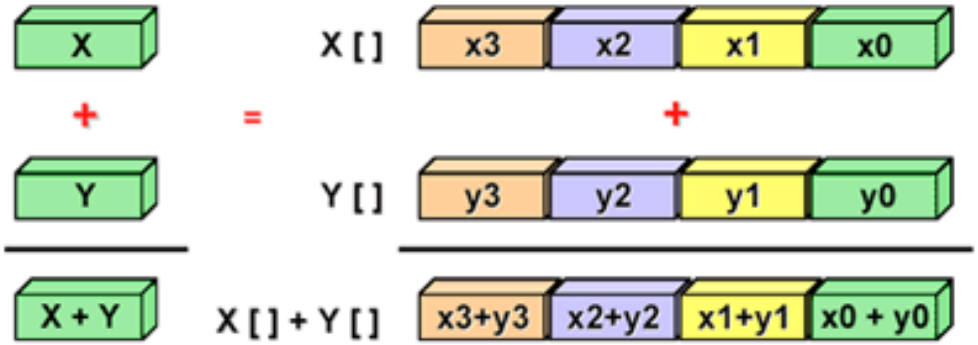

SISD
Instruction Pool
Data Pool
PU


load A
load B
C = A + B
store C
A = B * 2
store A
time

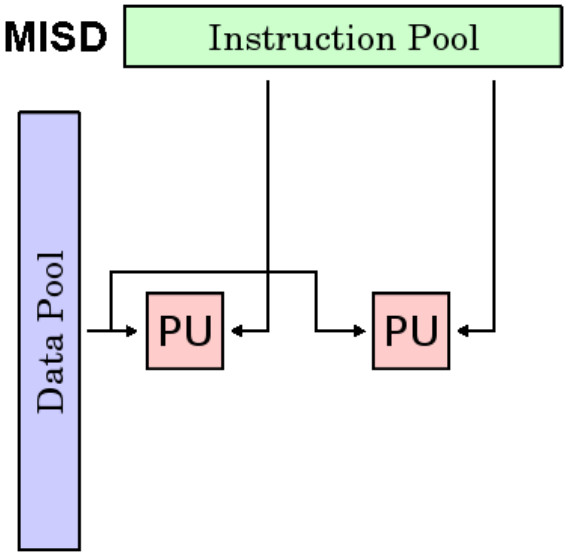
SIMD

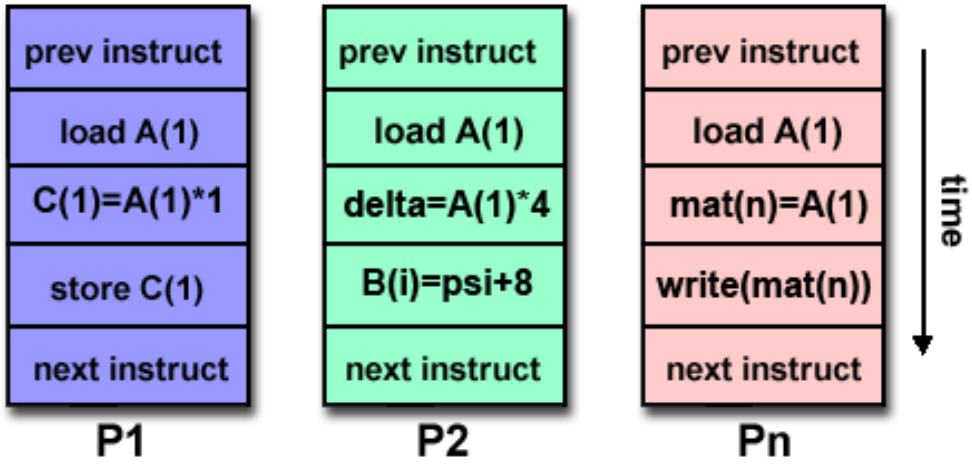

prev instruct
load A(1)
load B(1)
C(1)=A(1)*B(1)
store C(1)
next instruct
P1
prev instruct
load A(2)
load B(2)
C(2)=A(2)*B(2)
store C(2)
next instruct
P2
prev instruct
load A(n)
load B(n)
C(n)=A(n)*B(n)
store C(n)
next instruct
Pn
time


SIMD
Instruction Pool
Data Pool
PU
PU
PU
PU

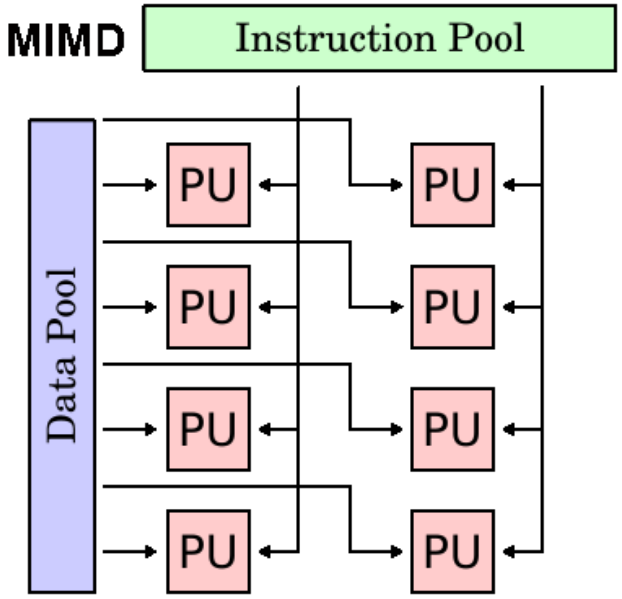

X
X [ ]
x3
x2
x1
x0
+
=
+
Y
Y [ ]
y3
y2
y1
y0
X + Y
X [ ] + Y [ ]
x3+y3
x2+y2
x1+y1
x0 + y0

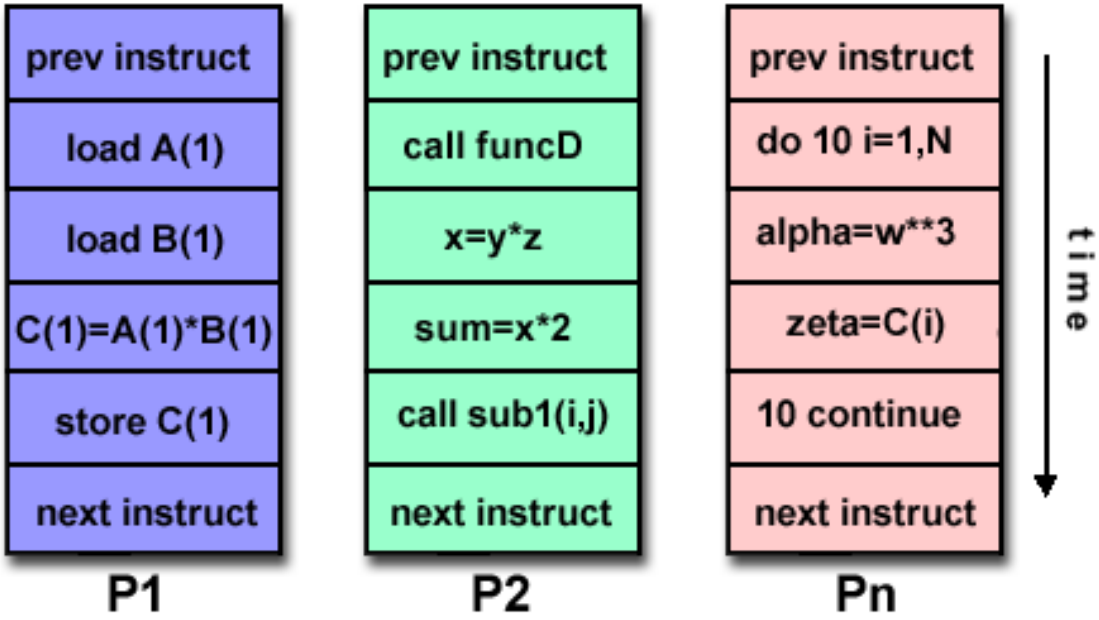
MISD


MISD
Instruction Pool
Data Pool
PU
PU


prev instruct
load A(1)
C(1)=A(1)*1
store C(1)
next instruct
P1
prev instruct
load A(1)
delta=A(1)*4
B(i)=psi+8
next instruct
P2
prev instruct
load A(1)
mat(n)=A(1)
write(mat(n))
next instruct
Pn
time

MIMD


MIMD
Instruction Pool
Data Pool
PU
PU
PU
PU
PU
PU
PU
PU

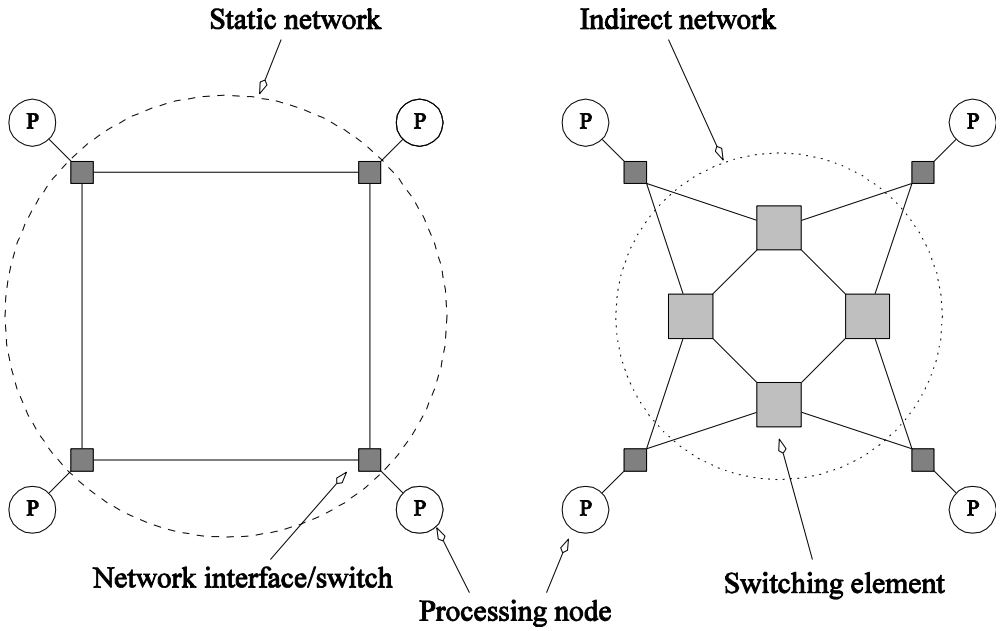

prev instruct
load A(1)
load B(1)
C(1)=A(1)*B(1)
store C(1)
next instruct
P1
prev instruct
call funcD
x=y*z
sum=x*2
call sub1(i,j)
next instruct
P2
prev instruct
do 10 i=1,N
alpha=w**3
zeta=C(i)
10 continue
next instruct
Pn
time

Outline

- 并行计算机系统结构
  - Flynn分类
  - **互连网络**
  - 并行计算机结构模型
- 多核处理器/GPU
- 并行计算存储访问模型
- 并行计算性能评测

互连网络

- 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，这种点到点的链接保持不变；
- 动态网络：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；

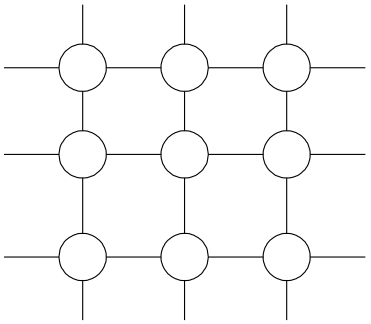


静态互连网络（1）

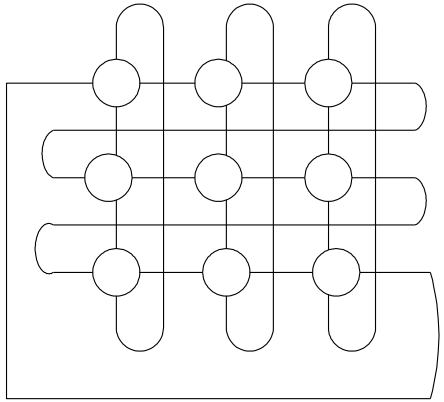
- 一维线性阵列（1-D Linear Array）：
  - 并行机中最简单、最基本的互连方式，
  - 每个节点只与其左、右近邻相连，也叫二近邻连接，
  - N个节点用N-1条边串接之，内节点度为2，直径为N-1，对剖宽度为1，
  - 当首、尾节点相连时可构成循环移位器，在拓扑结构上等同于环，环可以是单向或双向。

静态互连网络（2）

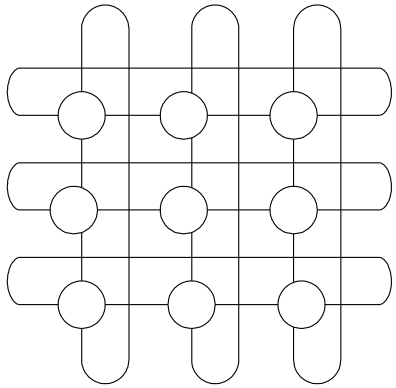
- 二维网孔（2-D Mesh）：
  - 每个节点只与其上、下、左、右的近邻相连（边界节点除外）
  - 在垂直方向上带环绕，水平方向呈蛇状，称为Illiac网孔
  - 垂直和水平方向均带环绕，则为2-D环绕



(a)2-D网孔



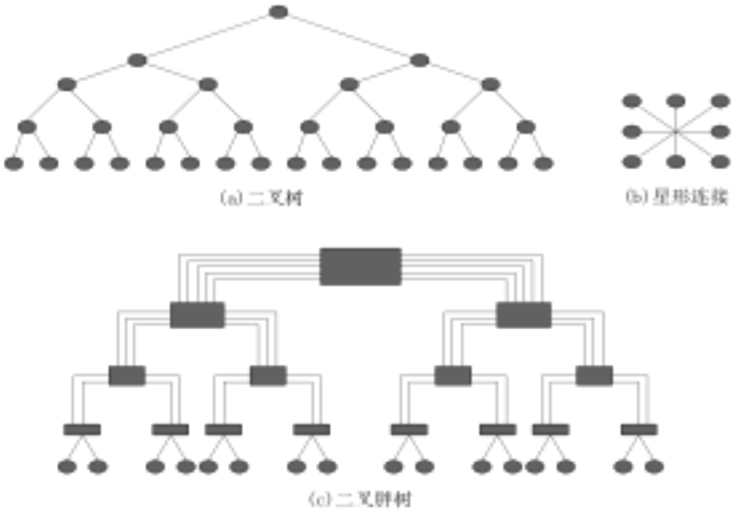
(b)Illiac网孔



(c)2-D环绕

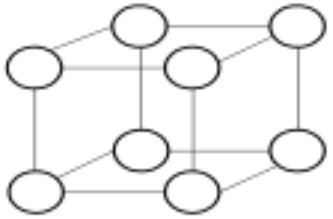
静态互连网络（3）

- 二叉树：
  - 除了根、叶节点，每个内节点只与其父节点和两个子节点相连。
  - 传统二叉树的主要问题是根易成为通信瓶颈。胖树节点间的通路自叶向根逐渐变宽。

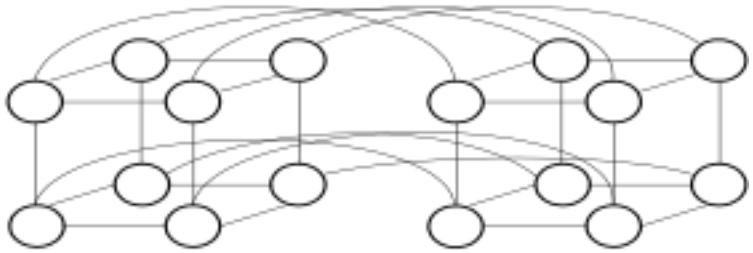


静态互连网络（4）

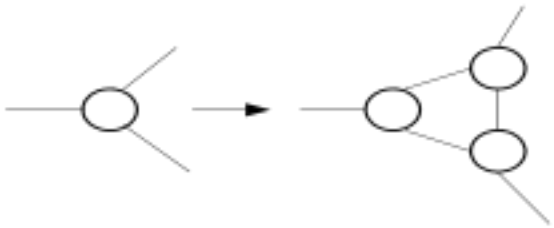
- 超立方



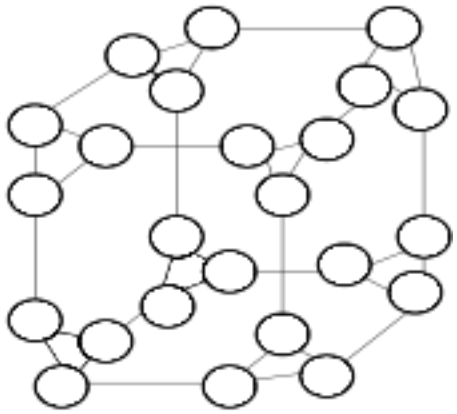
(a) 3-立方



(b) 4-立方



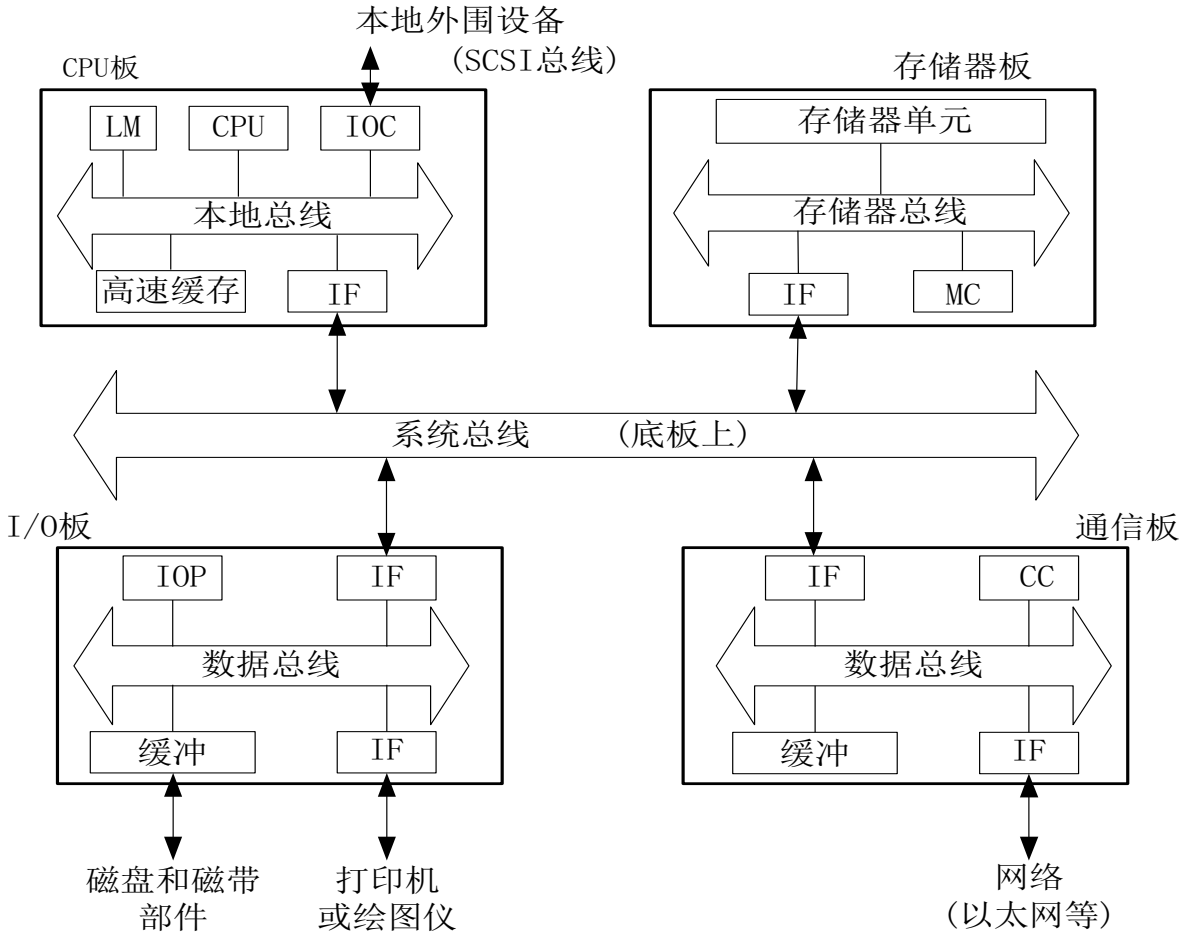
(c) 顶点代之以环



(d) 3-立方环

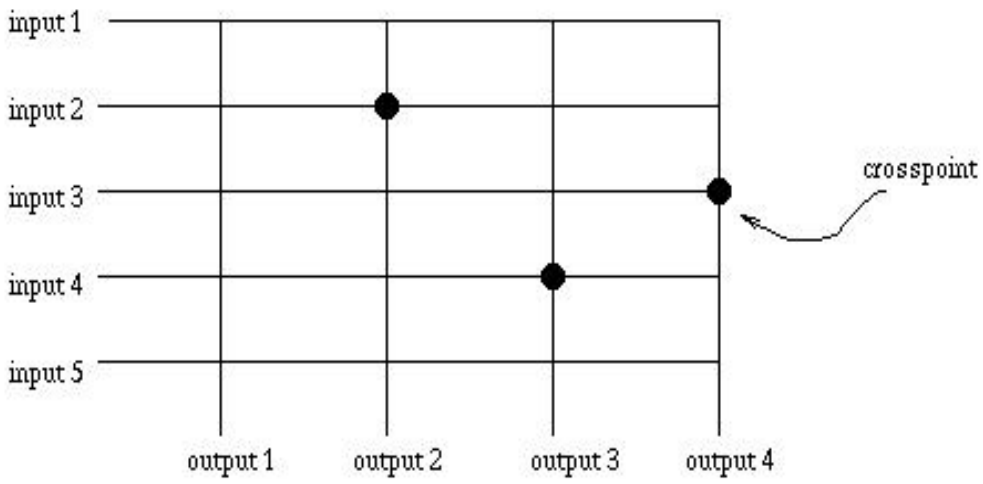
动态互连网络(1)

- 总线


本地外围设备
(SCSI总线)
CPU板
LM
CPU
IOC
本地总线
高速缓存
IF
存储器板
存储器单元
存储器总线
IF
MC
系统总线 (底板上)
I/O板
IOP
IF
数据总线
缓冲
IF
通信板
IF
CC
数据总线
缓冲
IF
磁盘和磁带
部件
打印机
或绘图仪
网络
(以太网等)

动态互连网络（2）

- 交叉开关（Crossbar）



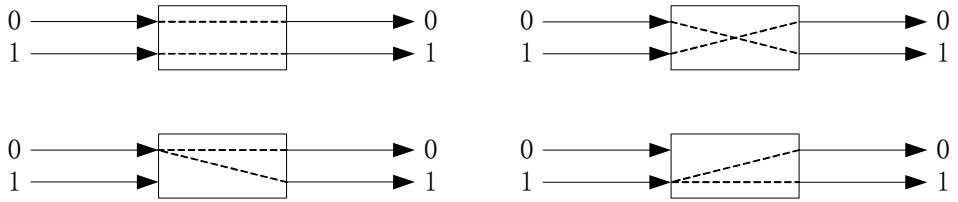
A Small Crossbar Switch

Input 2 is connected to ouput 2,
input 3 is connected to output 4,
input 4 is connected to output 3.

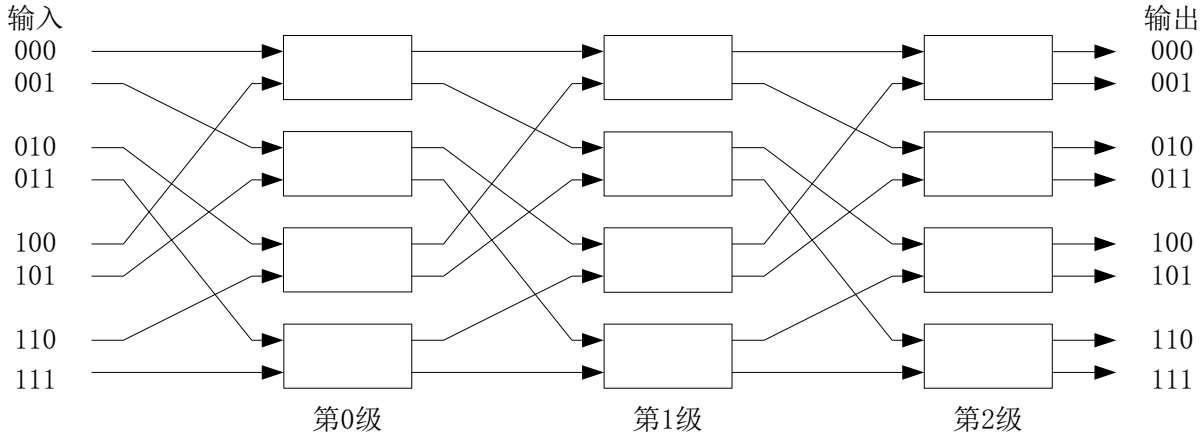
Note: Connections between free inputs and outputs are always possible.

动态互联网络（3）

- 单级交叉开关级联起来形成多级互连网络



(a) 4种可能的开关连接



(b) 一种8输入的Omega网络

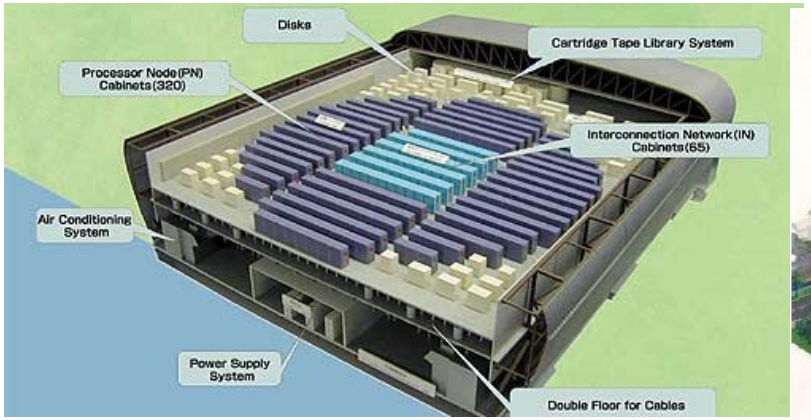
Outline

- 并行计算机系统结构
  - Flynn分类
  - 互连网络
  - **并行计算机结构模型**
- 多核处理器/GPU
- 并行计算存储访问模型
- 并行计算性能评测

PVP（Parallel Vector Processor）

- 含有为数不多、功能强大的定制向量处理器（VP），定制的高带宽纵横交叉开关及高速的数据访问。
- 通常不使用高速缓存，而是使用大量向量寄存器及指令缓存，使得该系统对程序编制的要求较高。
- 只有充分考虑了向量处理特点的程序才能在该系统上获得较好的性能。
- 银河I，NEC地球模拟器，GPU。。。

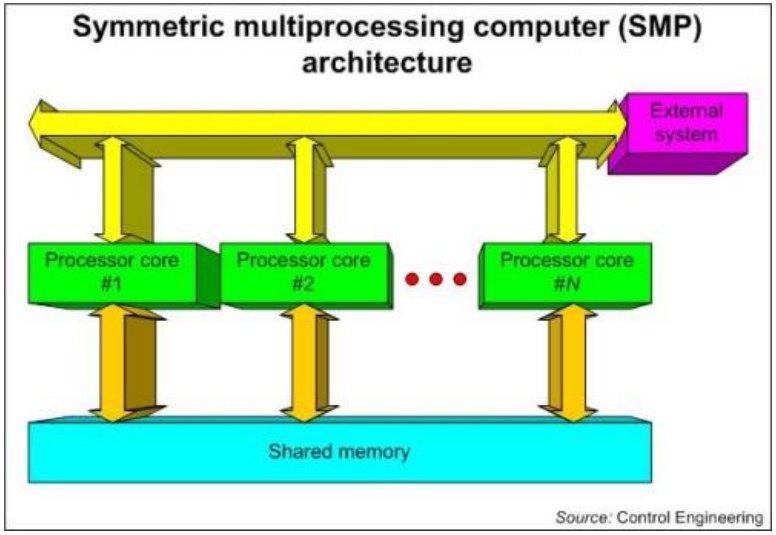
地球模拟器

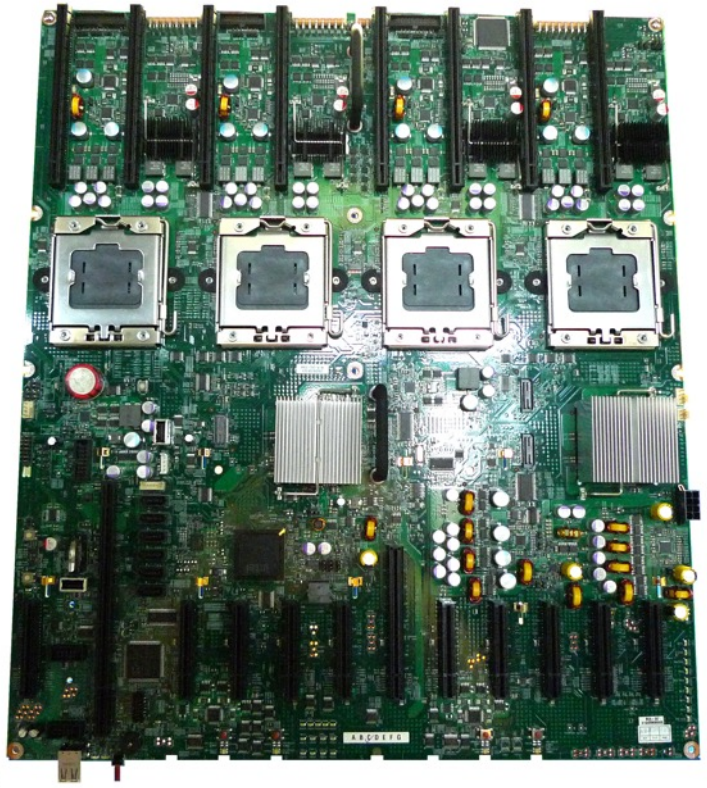

Disks
Cartridge Tape Library System
Processor Node(PN) Cabinets(320)
Interconnection Network(IN) Cabinets(65)
Air Conditioning System
Power Supply System
Double Floor for Cables


Power plant
Building for computer system
Building for operation and research

SMP（Symmetric Multiprocessor）

- 采用商品化的处理器，这些处理器通过总线或交叉开关连接到共享存储器。
- 每个处理器可等同地访问共享存储器、I/O设备和操作系统服务
- 扩展性有限





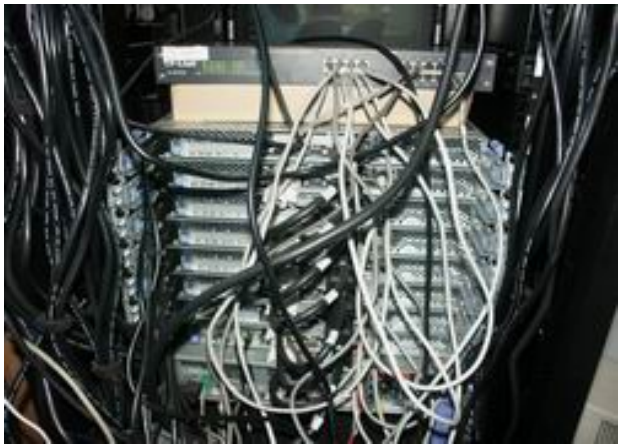
MPP（Massively Parallel Processor）

- 处理节点采用商品微处理器
- 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）
- 能扩展至成百上千乃至上万个处理器
- 异步MIMD，构成程序的多个进程有自己的地址空间，进程间通信消息传递相互作用
- Tianhe, K-Computer, Cray XT5, BlueGene。。。。

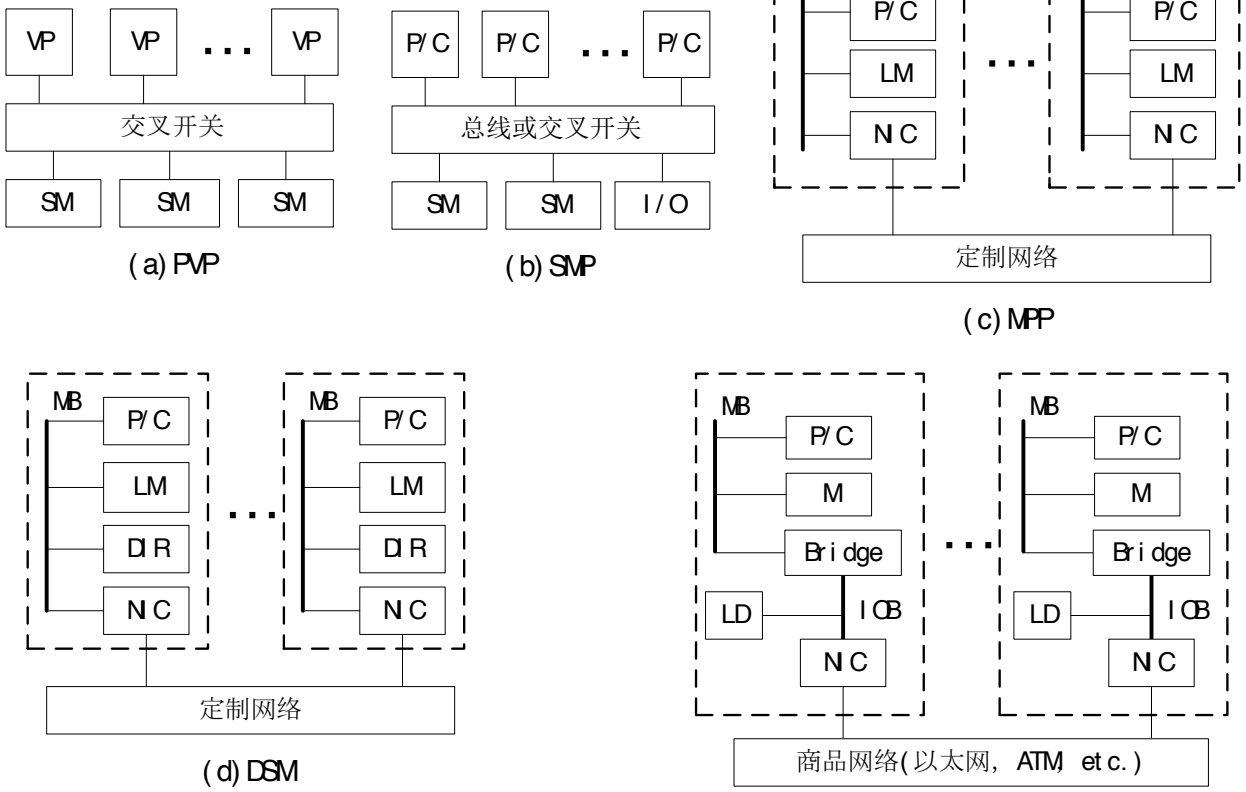
Cluster（集群）

- 分布式存储，MIMD，工作站+商用互连网络，每个节点是一个完整的计算机，有自己的磁盘和操作系统，而MPP中只有微内核
- 优点：
  - 投资风险小
  - 系统结构灵活
  - 性能/价格比高
  - 能充分利用分散的计算资源
  - 可扩展性好
- 问题
  - 通信性能
  - 并行编程环境
- IBM Cluster 1350/1600。。。

Cluster1350



并行计算机结构模型小结

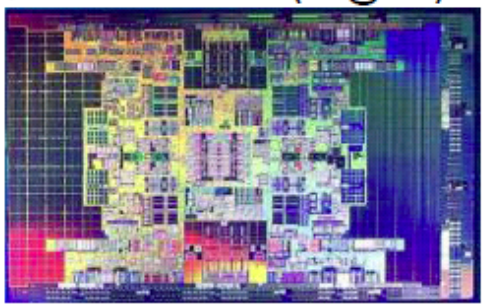


Outline

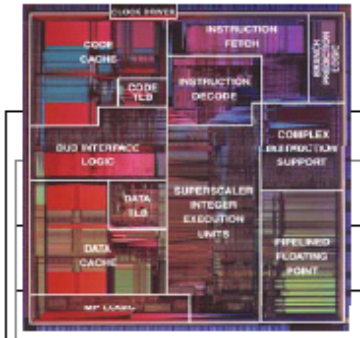
- 并行计算机系统结构
  - Flynn分类
  - 互连网络
  - 并行计算机结构模型
- **多核处理器/GPU**
- 并行计算存储访问模型
- 并行计算性能评测

摩尔定律

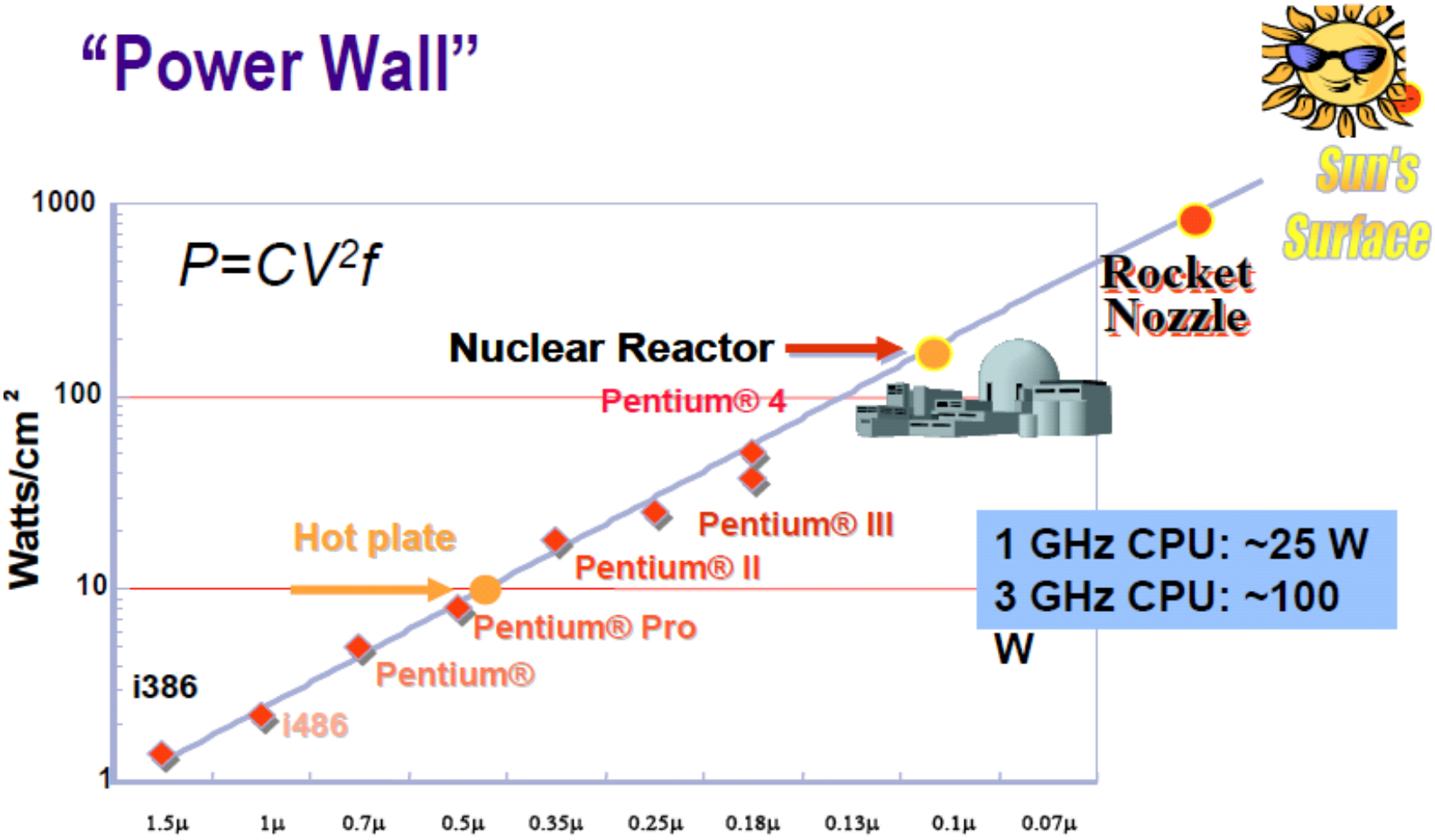
Quad-Core Itanium (2G@65n)



Pentium 4 (42M@.18μ)

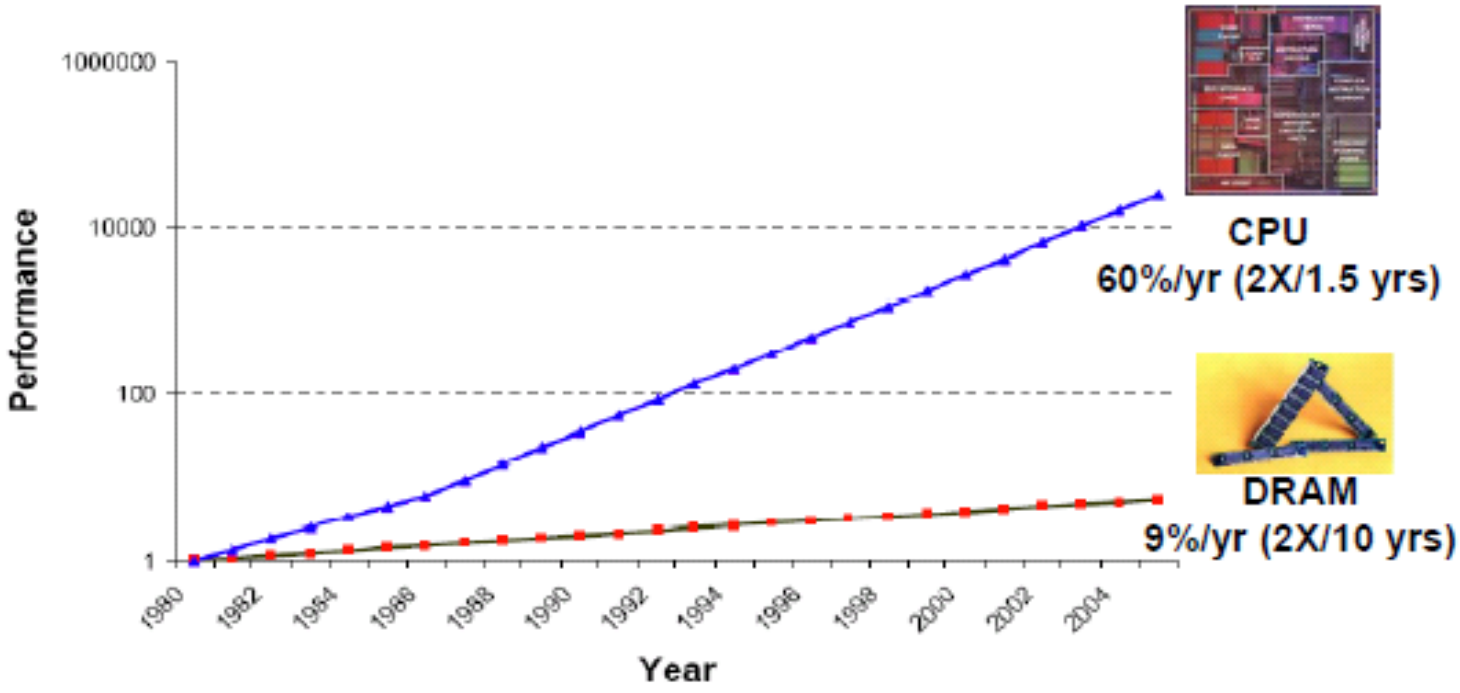


"Power Wall"

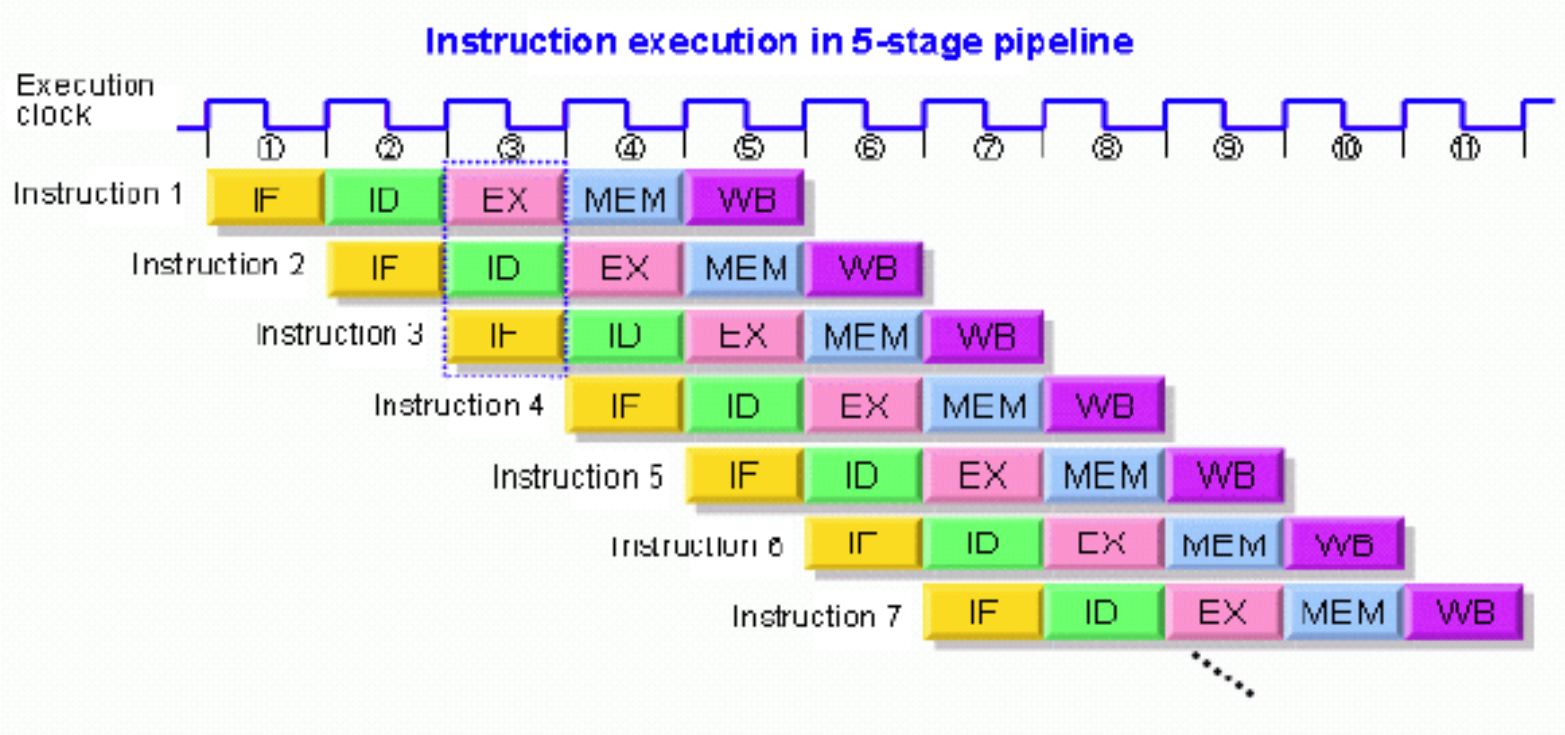

$P=CV^2f$
Watts/cm^2
1000
100
10
1
1.5µ
1µ
0.7µ
0.5µ
0.35µ
0.25µ
0.18µ
0.13µ
0.1µ
0.07µ
Nuclear Reactor
Rocket Nozzle
Sun's Surface
Pentium® 4
Pentium® III
Pentium® II
Pentium® Pro
Pentium®
i486
i386
Hot plate
1 GHz CPU: ~25 W
3 GHz CPU: ~100 W

Memory Wall

缓存占据>70%芯片面积！

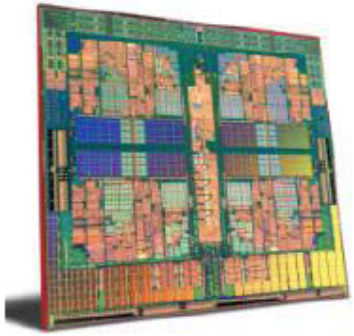


隐式指令级并行

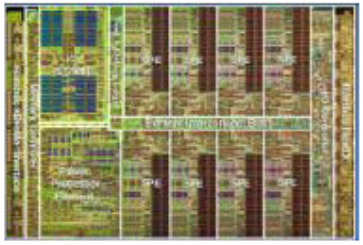

Instruction execution in 5-stage pipeline
Execution clock
① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪
Instruction 1
Instruction 2
Instruction 3
Instruction 4
Instruction 5
Instruction 6
Instruction 7
IF
ID
EX
MEM
WB

多核时代

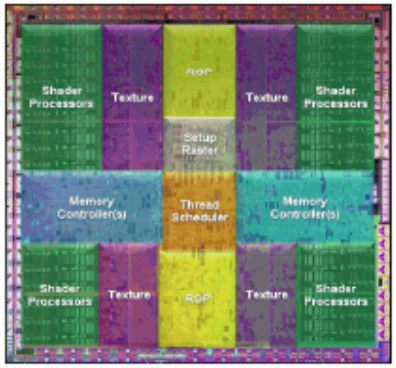
- 多核计算环境
  - 多个复杂度适中，相对低功耗的处理核心并行工作
  - CPU时钟频率基本不变
  - 计算机硬件不会更快，但会更“宽”
  - 操作系统、应用程序设计？？



Quad-core Opteron



IBM Cell Broadband Engine

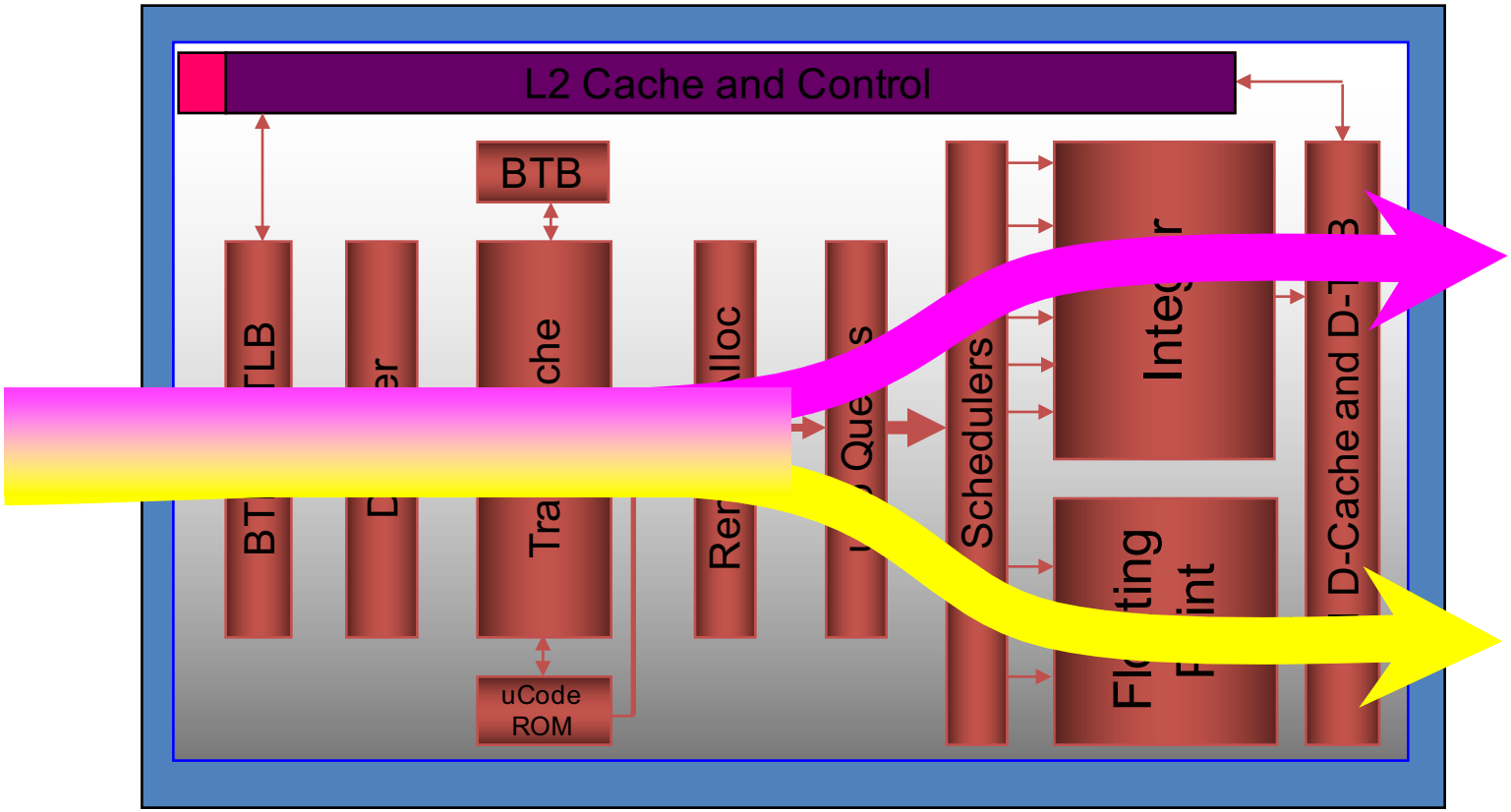


nVidia GT200

双核技术 VS. 超线程技术

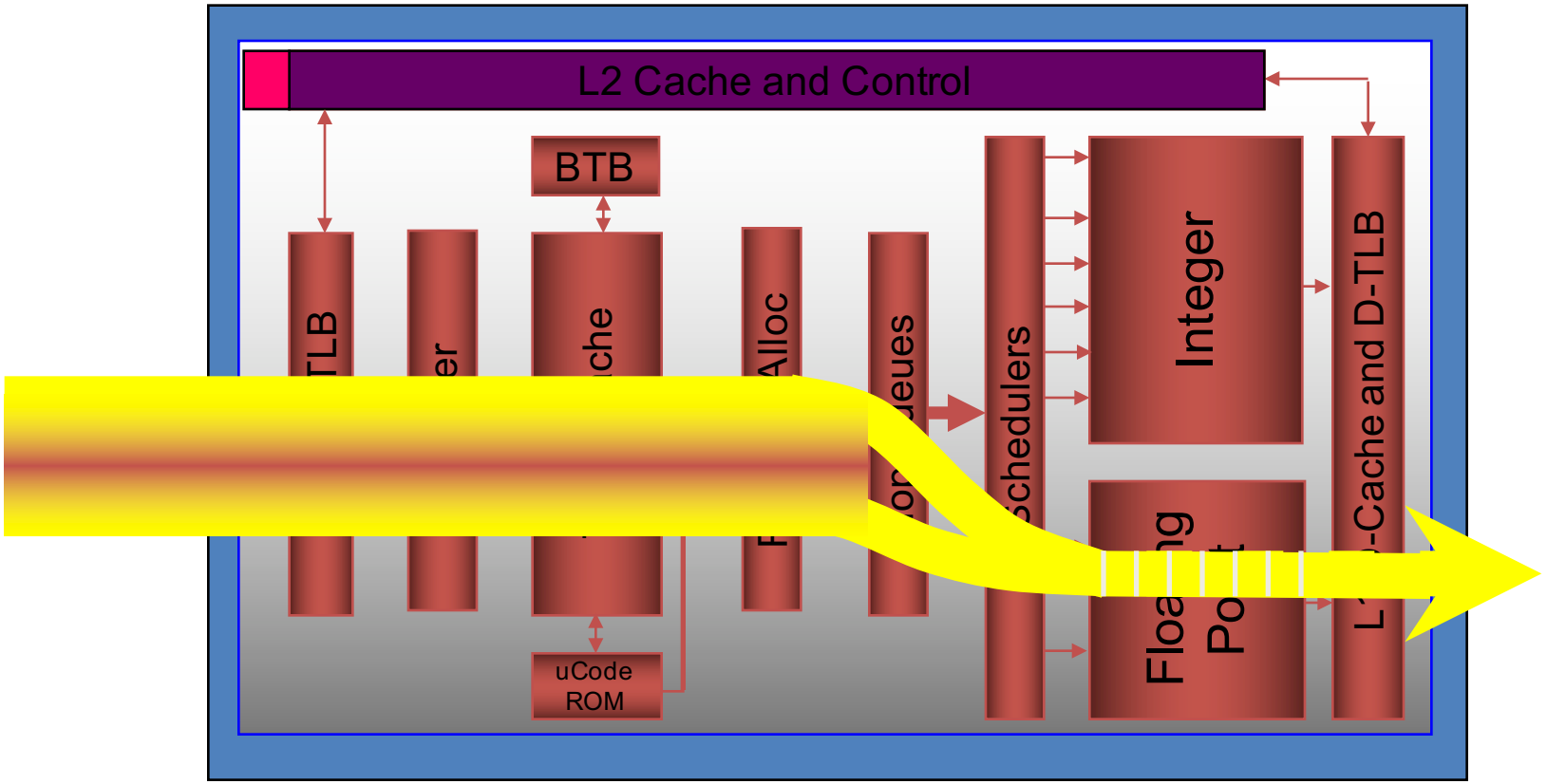
- 双核是真正意义上的双处理器
  - 不会发生资源冲突
  - 每个线程拥有自己的缓存、寄存器和运算器
- 一个3.2GHz Smithfiled在性能上并非等同于3.2GHz P4 with HT 的2倍
  - HT 使处理器的性能至少提升了1/3
  - 双核的性能相当于2块 non-HT 处理器
- 双核技术与HT技术在性能上的对比
- Ex 1: 两个floating point线程 (Smithfield client)
  - 每个线程拥有自己的FPU, 没有资源冲突
  - 尽管性能上没有提升太多，但仍然优于HT
- Ex 2: 一个integer线程与一个floating point线程
  - 性能大幅度提升
  - 没有资源冲突

Single core , With HT
(Eg. Pentium 4 Processor With HT)
Integer and Floating Point Threads



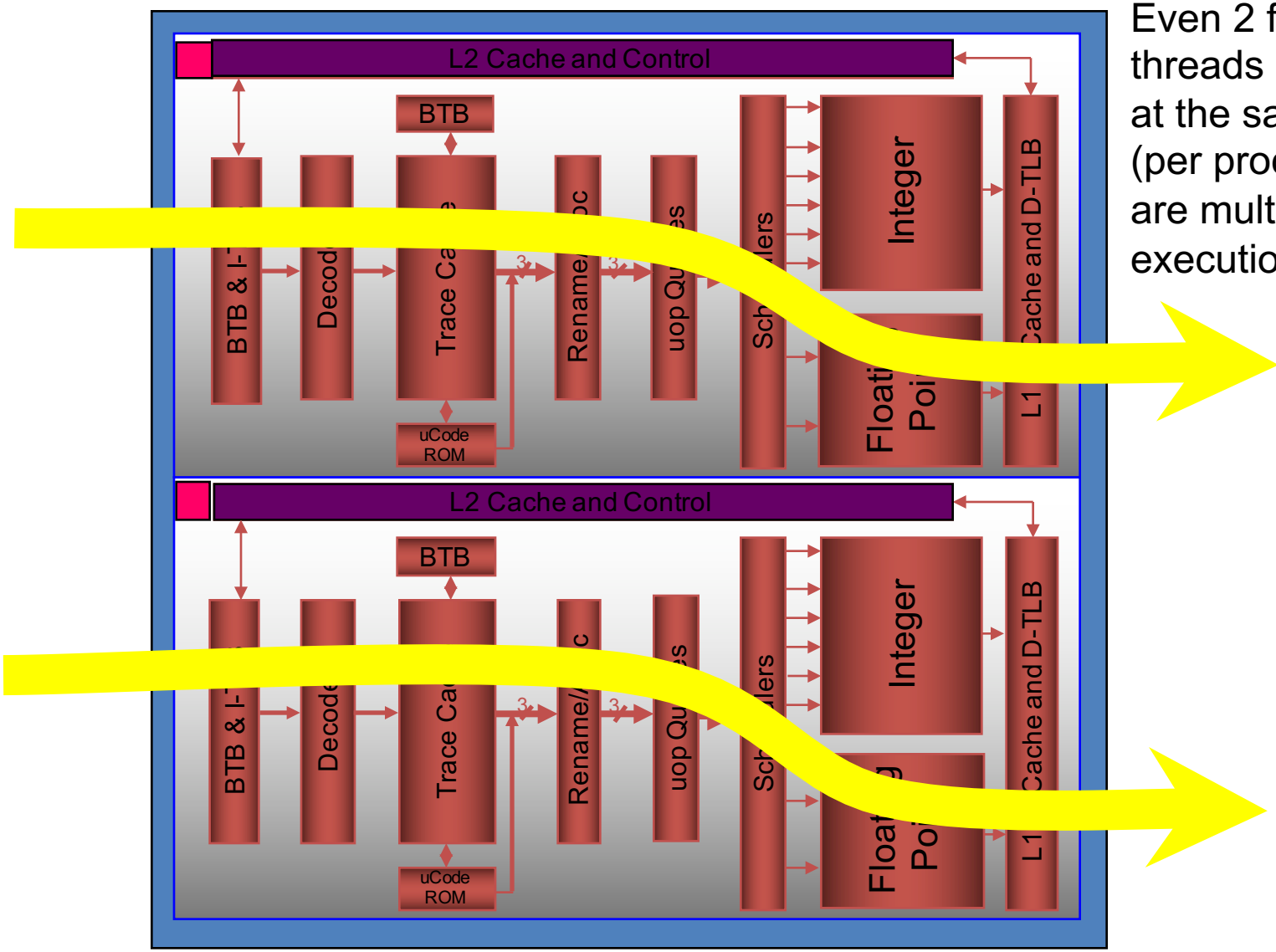
2 threads can be executed at the same time (per processor) if they’ re not competing for the same execution resource

Single core , With HT
(Eg. Pentium 4 Processor with HT)
Two Floating Point Threads



2 threads CANNOT be executed at the same time (per processor) if they’ re competing for the same execution resource (eg. 2 floating point threads in a P4P architecture)

Dual core , Without HT
(Eg. Pentium D Processor)
Two Floating Point Threads



Even 2 floating point threads can be executed at the same time now (per processor) as there are multiple floating point execution units

Dual core , With HT

(Eg. Dual Core Pentium Processor Extreme Edition)

Supports HT

Multiple Integer and Floating Point Threads

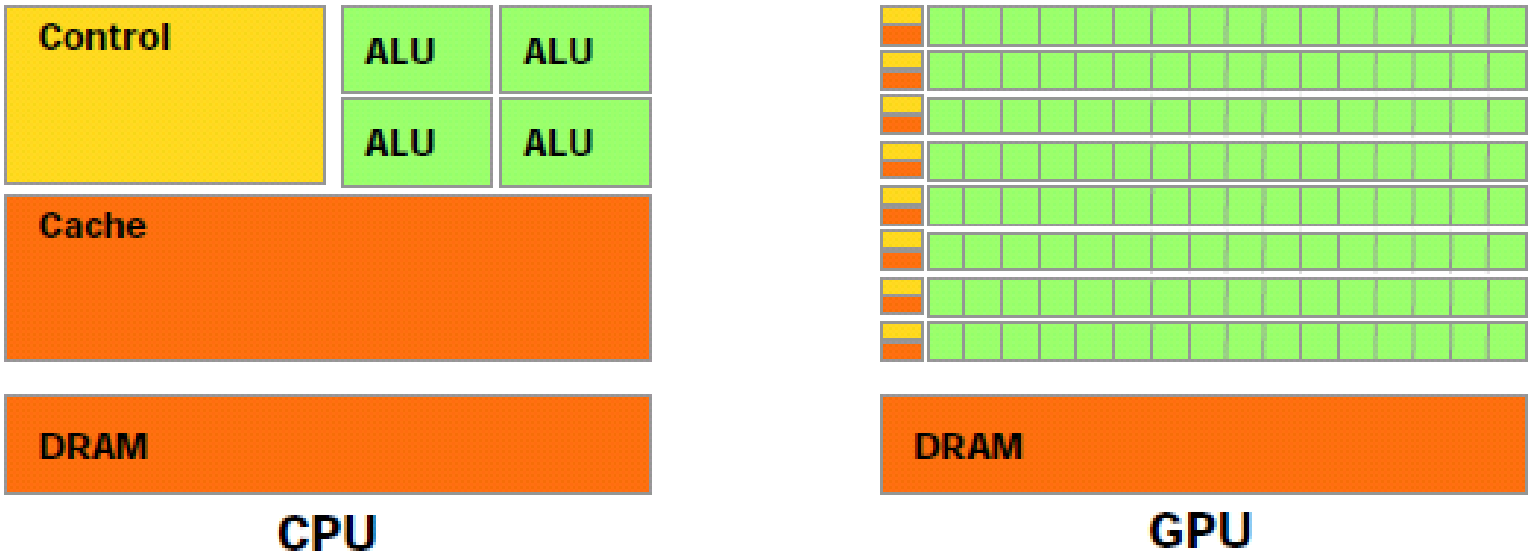
With dual core & HT together, maximum # of threads that can be executed at a time is 4 per processor

GPU: Graphic Processing Unit

- GPU在显卡中的地位相当于计算机中的CPU
  - GPU使显卡减少了对CPU的依赖，并进行部分原本CPU的工作
  - 当前GPU已经不再局限于3D图形处理，GPU通用计算技术发展已经引起业界的关注，事实也证明在浮点运算、并行计算等部分计算方面，GPU可以提供数十倍乃至于上百倍于CPU的性能

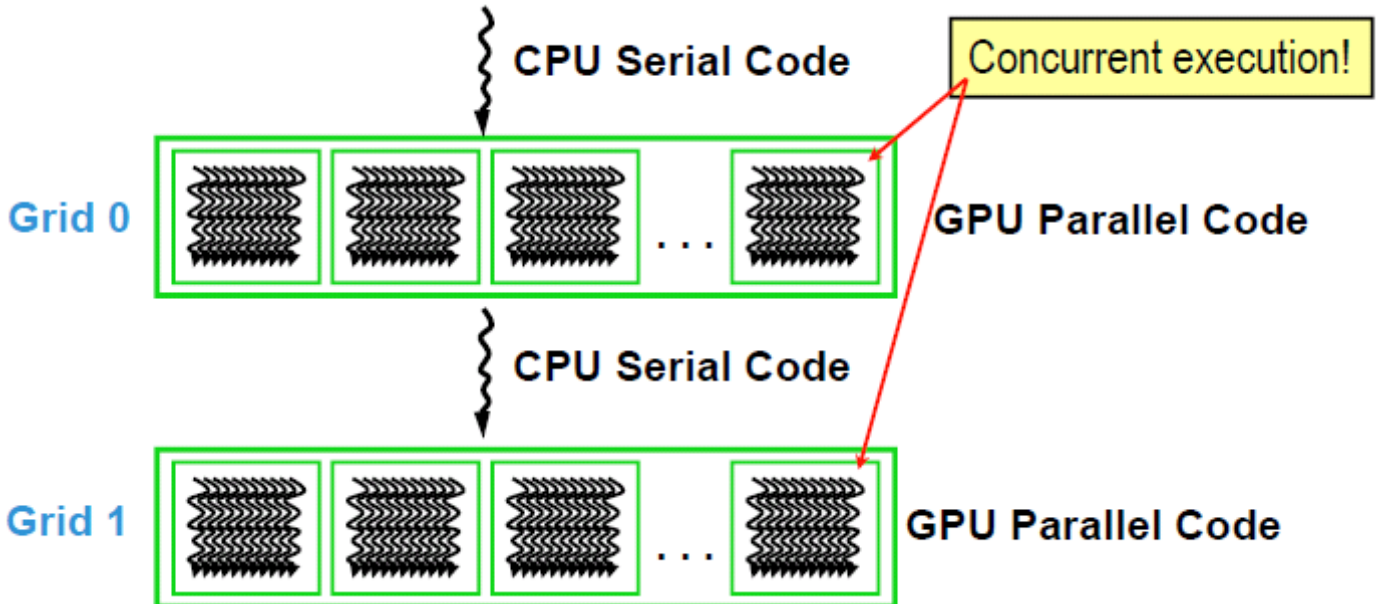
GPU设计哲学

- GPU将更多元件用于数据处理，而非控制和存储

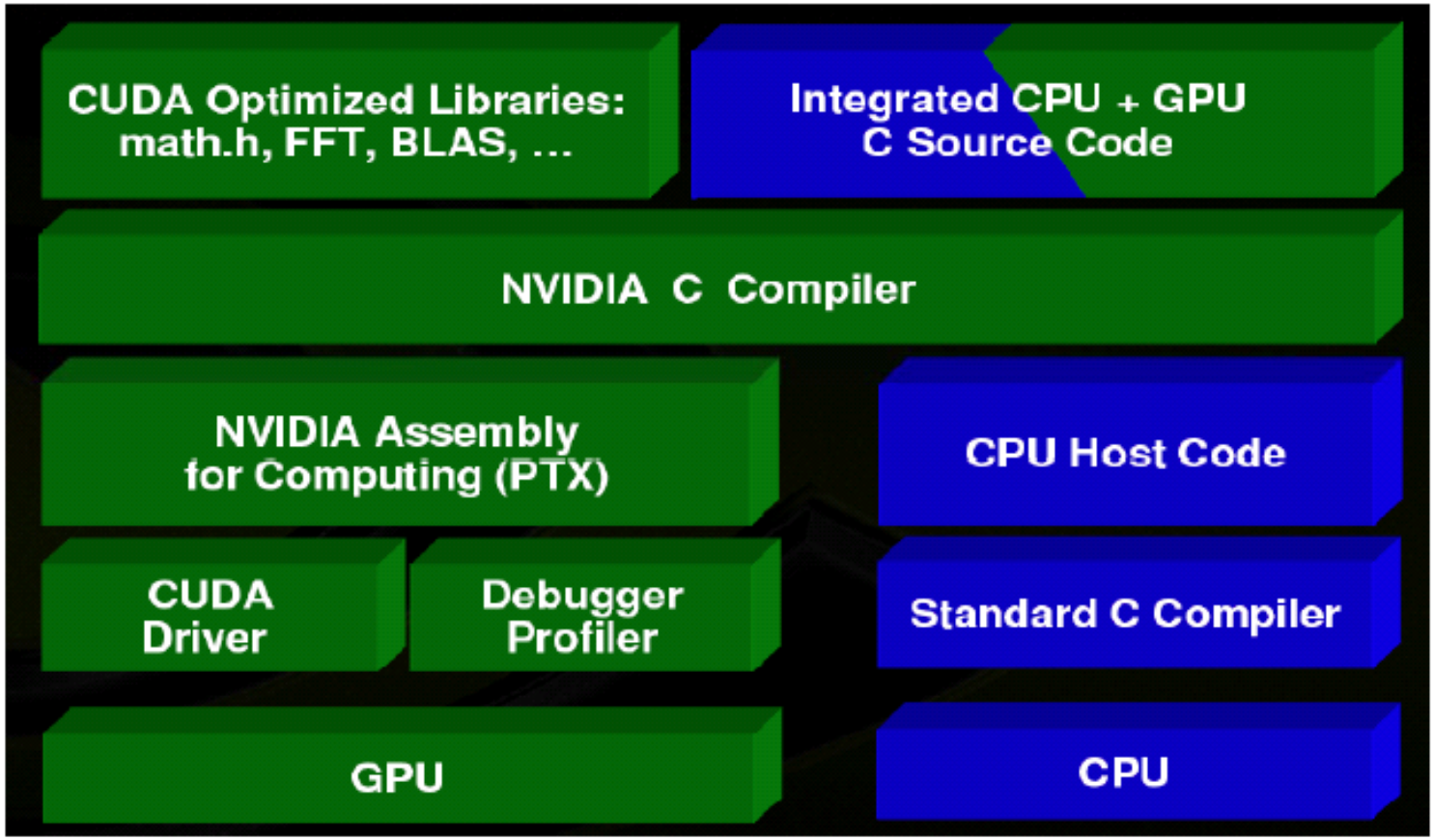


GPU编程

- GPU编程模型
  - GPU可以视为超大规模并行协处理器
  - SPMD（单程序多数据）模式，数据并行



Nvidia CUDA

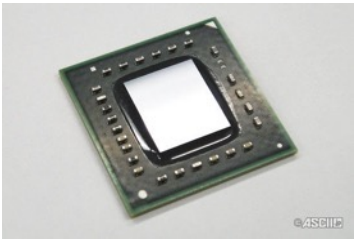

CUDA Optimized Libraries:
math.h, FFT, BLAS, ...
Integrated CPU + GPU
C Source Code
NVIDIA C Compiler
NVIDIA Assembly
for Computing (PTX)
CPU Host Code
CUDA
Driver
Debugger
Profiler
Standard C Compiler
GPU
CPU

APU: Accelerated Processing Units

- APU（加速处理器）集成了高性能串行和并行处理内核，在视觉运算、安全性、每瓦性能以及设备外形方面取得了重大的突破。

AMD E系列加速处理器型号和特性对比									
型号	CPU时钟频率	CPU核心芯片	TDP	二级缓存总容量	Radeon™流处理器数量[3]	GPU时钟频率	DIRECTX®版本	UVD	DDR3速度
AMD E系列加速处理器									
E-350	1.6 GHz	2个核心	18瓦	1MB	80	492MHz	11	UVD 3	DDR3-1066/DDR3L-1066

APU



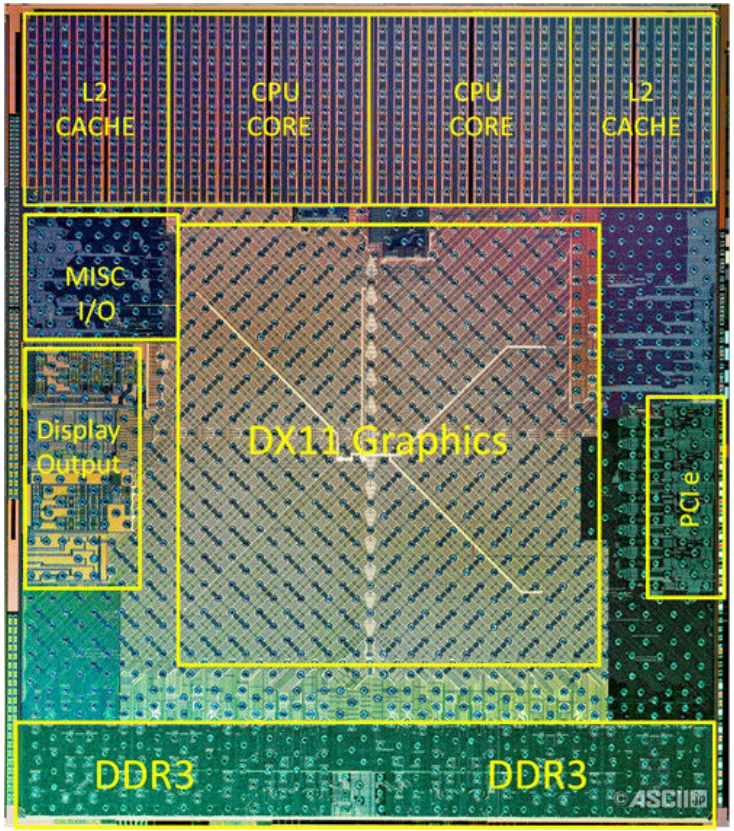
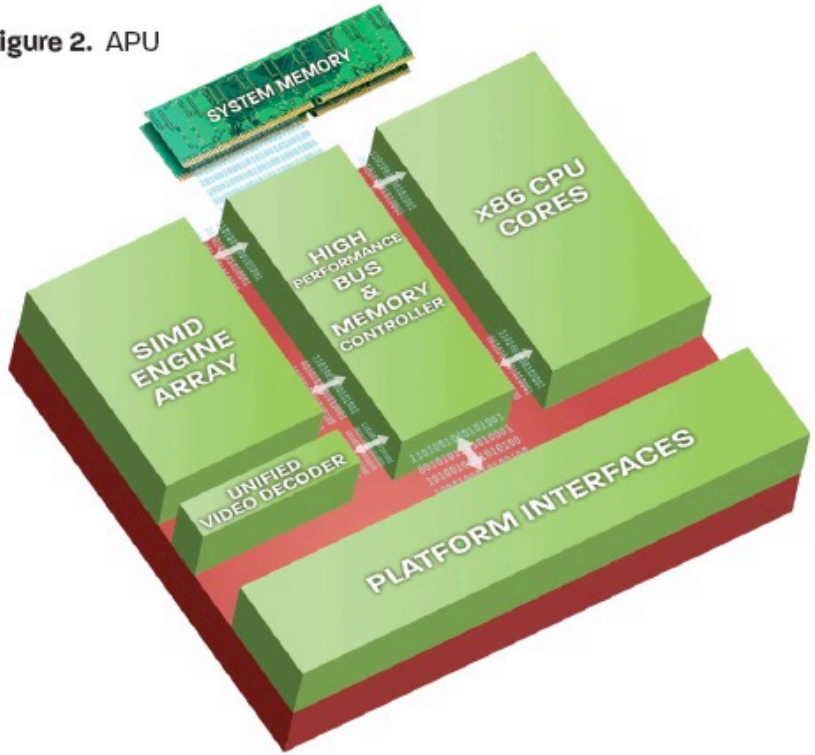
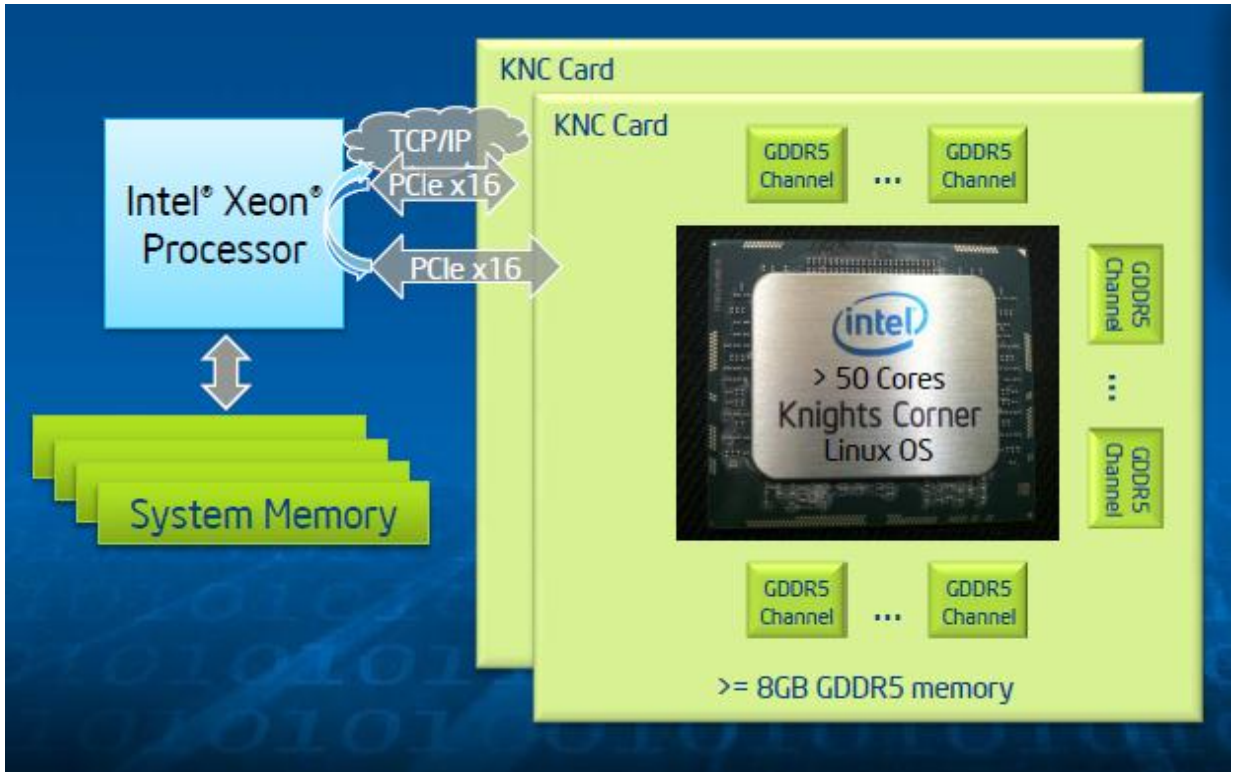


Figure 2. APU



MIC: Intel Phi

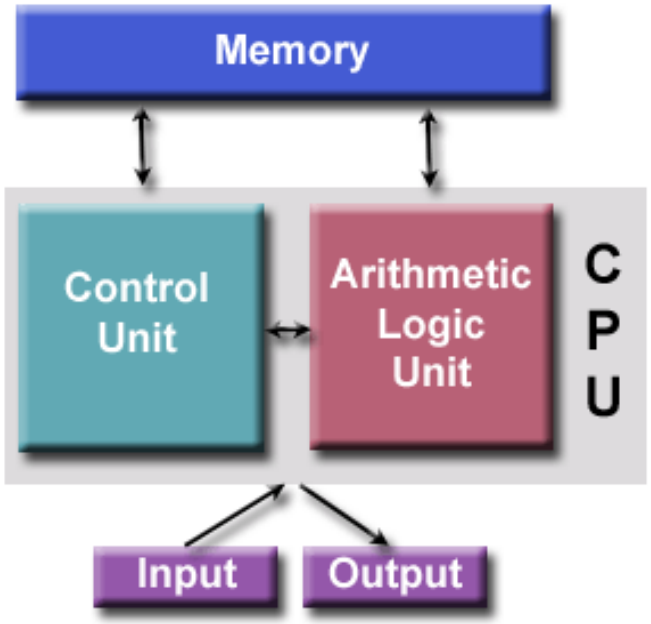
- 核数：57
- 缓存：28.5MB
- 主频：1.1GHz
- 内存：8GB
- 通道：12



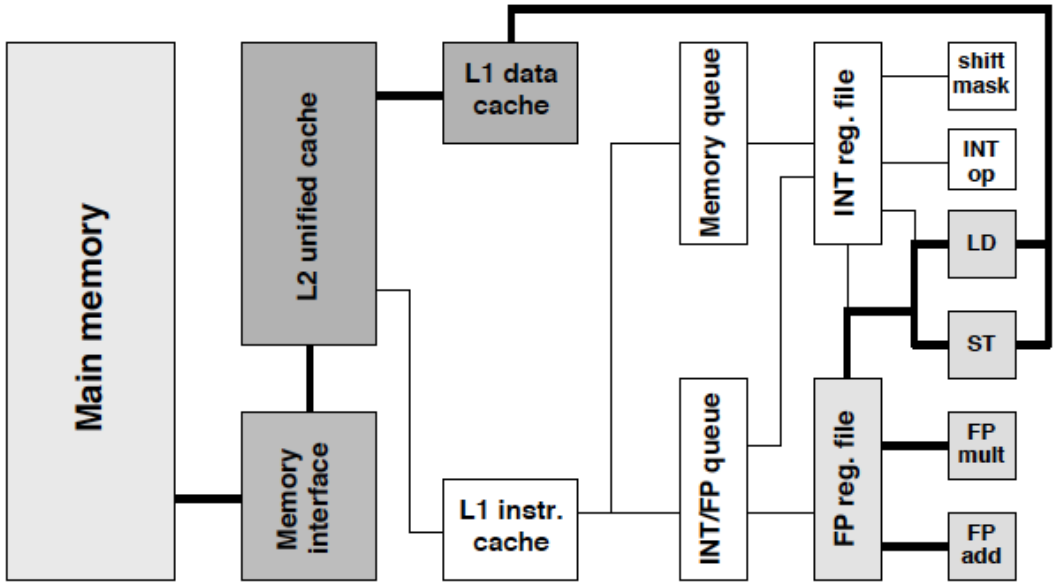
Outline

- 并行计算机系统结构
  - Flynn分类
  - 互连网络
  - 并行计算机结构模型
- 多核处理器/GPU
- **并行计算存储访问模型**
- 并行计算性能评测

处理器基本结构

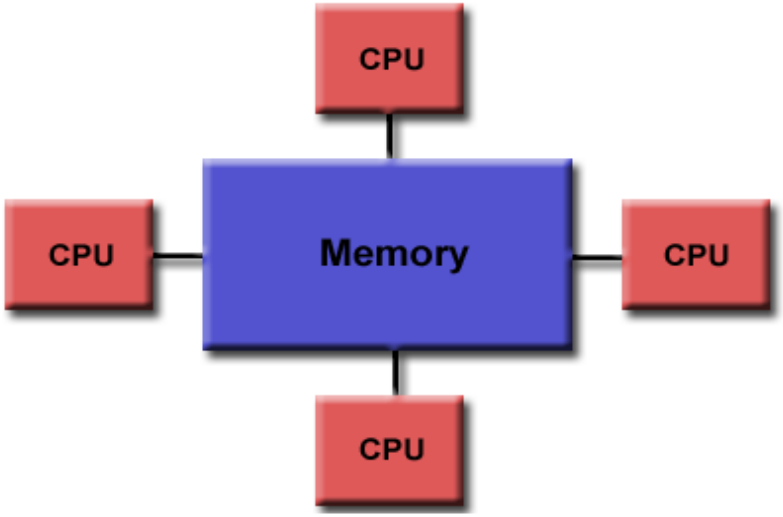


von Neumann模型

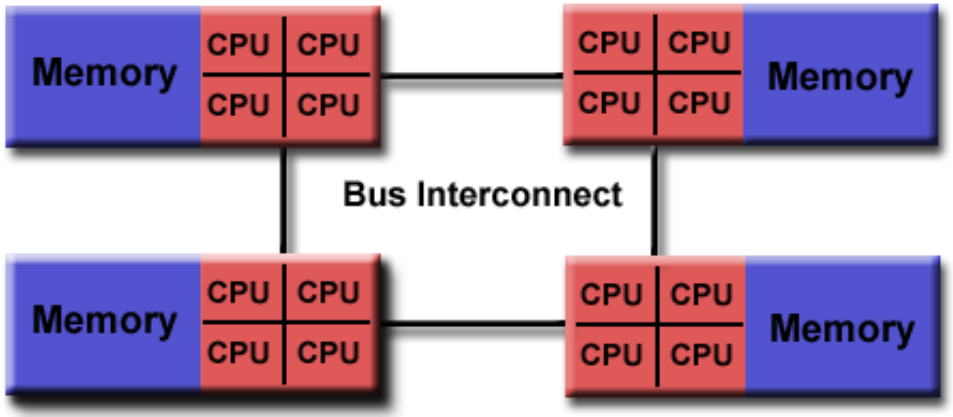


CPU逻辑结构

共享存储访问

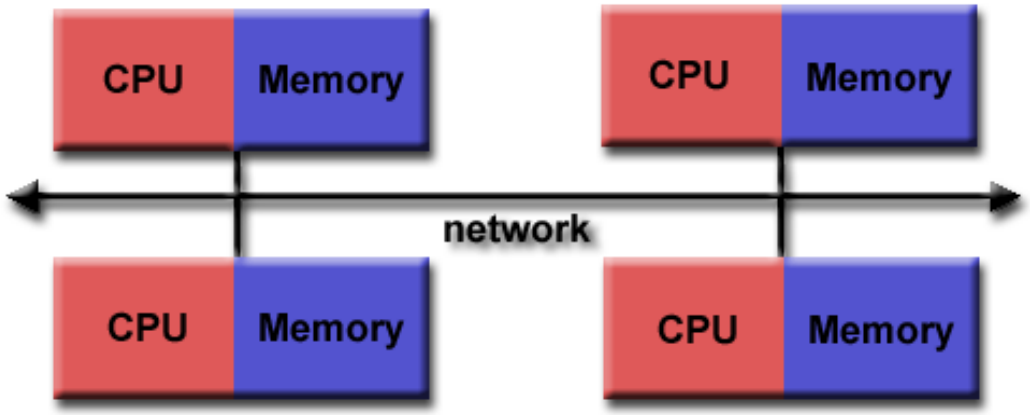


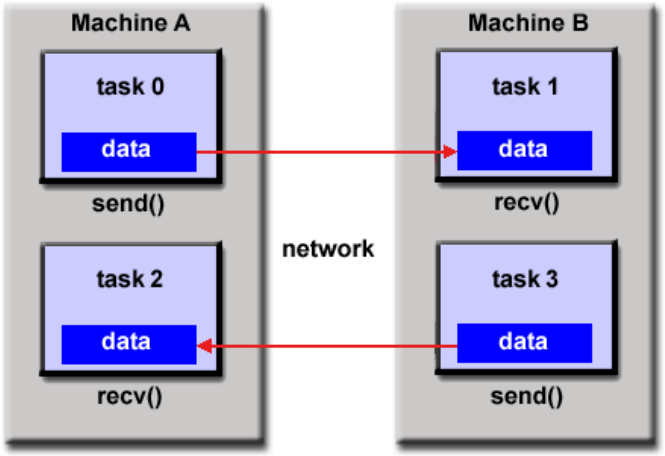
- UMA
- 均匀存储访问
- 每个处理器可以等同地访问统一的存储空间



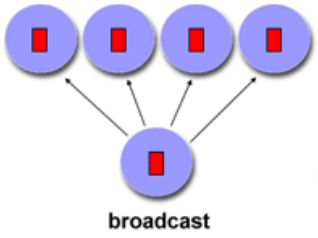
- NUMA
- 非均匀存储访问
- 存储全局统一编址，每个处

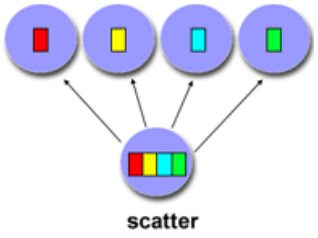
分布式存储访问

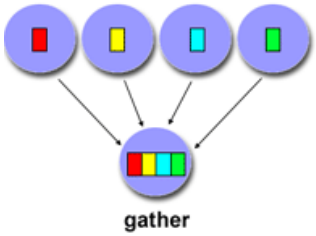


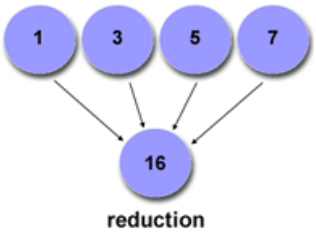


- 每个物理节点有自己私有的存储空间，各物理节点通过高速网络连接；
- 运行在不同物理节点的任务，不能直接访问其他物理节点的存储空间，相互数据交换通过消息传递实现。

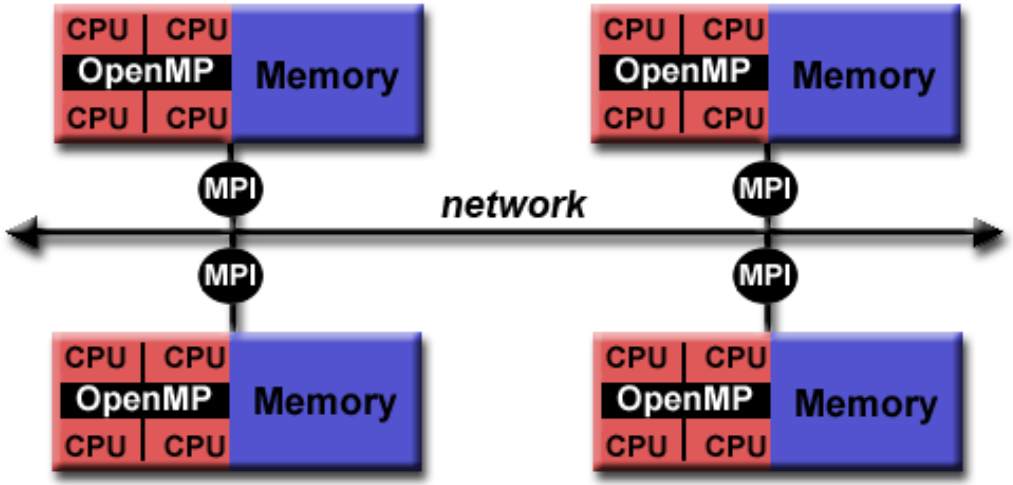






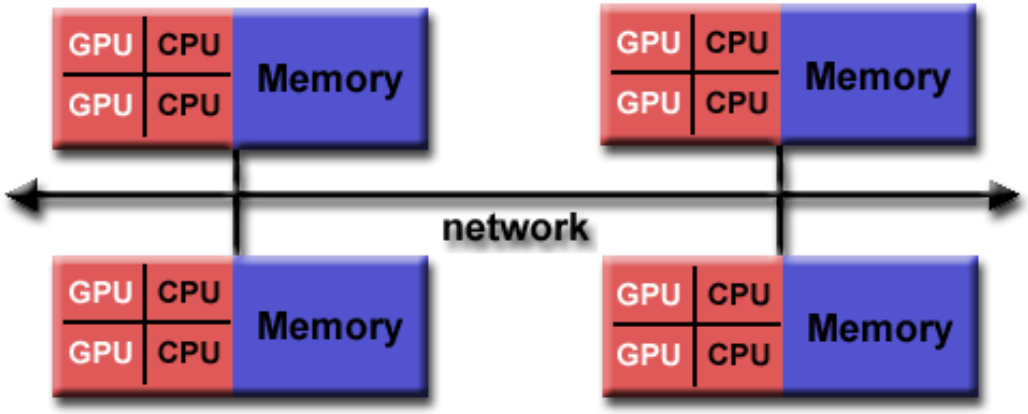


多层混合并行计算



- 物理节点内部任务可实现为OpenMP或多线程代码
- 运行在不同物理节点的任务通过消息传递交换数据

- 物理节点内部分计算分配到GPU
- 物理节点间的任务通过消息传递交换数据



主要编程模型（接口）

- OpenMP
  - (http://openmp.org)：共享内存并行计算环境.
- MPI（Message Passing Interface）
  - (http://www.mpi-forum.org/)：分布式内存并行计算环境
- GPU
  - CUDA: (Compute Unified Device Architecture)：用于Nvidia GPU. (http://www.nvidia.com/object/cuda_home_new.html)
  - OpenCL: (http://www.khronos.org/opencl/) open source API for developing parallel processing applications on a variety of CPUs, as well as AMD/ATI and NVIDIA GPUs

Outline

- 并行计算机系统结构
  - Flynn分类
  - 互连网络
  - 并行计算机结构模型
- 多核处理器/GPU
- 并行计算存储访问模型
- **并行计算性能评测**

并行计算性能评测：CPU性能指标

- 工作负载
  - 执行时间 （Elapsed Time）
  - 浮点运算数（各种操作折算为浮点运算数的经验规则），Flop
  - 指令数目，MIPS
- 并行执行时间

  T_{comput} 为计算时间，T_{paro} 为并行开销时间，T_{comm}为相互通信时间

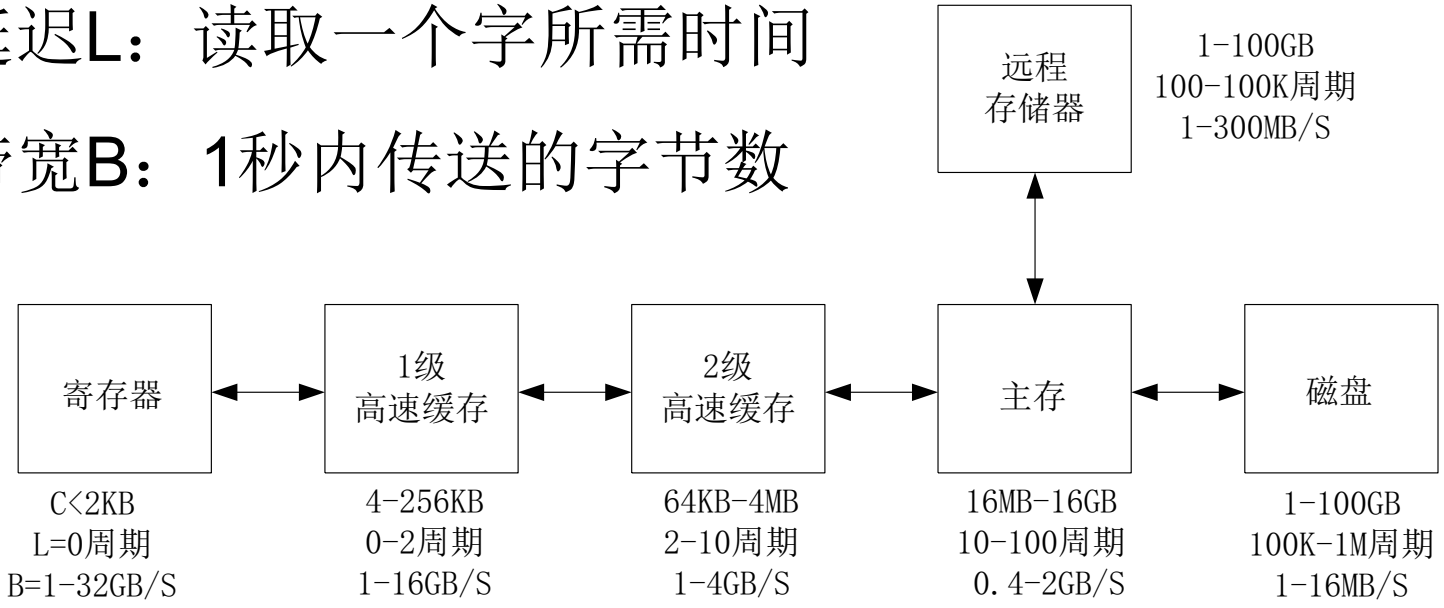
$$T_n = T_{comput} + T_{paro} + T_{comm}$$

并行计算性能评测：存储器层次

容量C：能保存数据的字节数

延迟L：读取一个字所需时间

带宽B：1秒内传送的字节数



并行计算性能评测：性能指标

- 计算/通信比

$$\frac{Computation\ Time}{Communication\ Time} = \frac{t_{comp}}{t_{comm}}$$

- 加速比

$$S(n) = \frac{Execution\ time\ (one\ processor\ system)}{Execution\ time\ (multiprocessor\ system)} = \frac{t_s}{t_p}$$

并行计算性能评测：性能指标

- 开销
  - 部分处理器的空闲
  - 并行版本中所需的顺序计算中不会出现的额外计算
  - 发送消息所需的通信时间
- 效率

$$E = \frac{\text{Execution time using one processor}}{\text{Execution time using a multiprocessor} \times \text{number of processors}}$$

$$= \frac{t_s}{t_p \times n}$$

并行计算性能评测：性能指标

- 代价
  - 代价＝（执行时间）×（所使用的处理器总数）

The *processor-time* product or *cost* (or *work*) of a computation defined as

$$\text{Cost} = (\text{execution time}) \times (\text{total number of processors used})$$

The cost of a sequential computation is simply its execution time, t_s. The cost of a parallel computation is $t_p \times n$. The parallel execution time, t_p, is given by $t_s/S(n)$. Hence, the cost of a parallel computation is given by

$$\text{Cost} = \frac{t_s n}{S(n)} = \frac{t_s}{E}$$

参数定义

- P：处理器数；
- W：问题规模（计算负载、工作负载，给定问题的总计算量）；
  - W_s：应用程序中的串行分量，f是串行分量比例（$f = W_s/W$）；
  - W_P：应用程序中可并行化部分，1-f为并行分量比例；
  - $W_s + W_p = W$；
- T_s：串行执行时间，T_p：并行执行时间；
- S：加速比，E：效率。

加速比性能定律

- Amdahl 定律
- Gustafson 定律
- Sun and Ni 定律

加速比性能定律

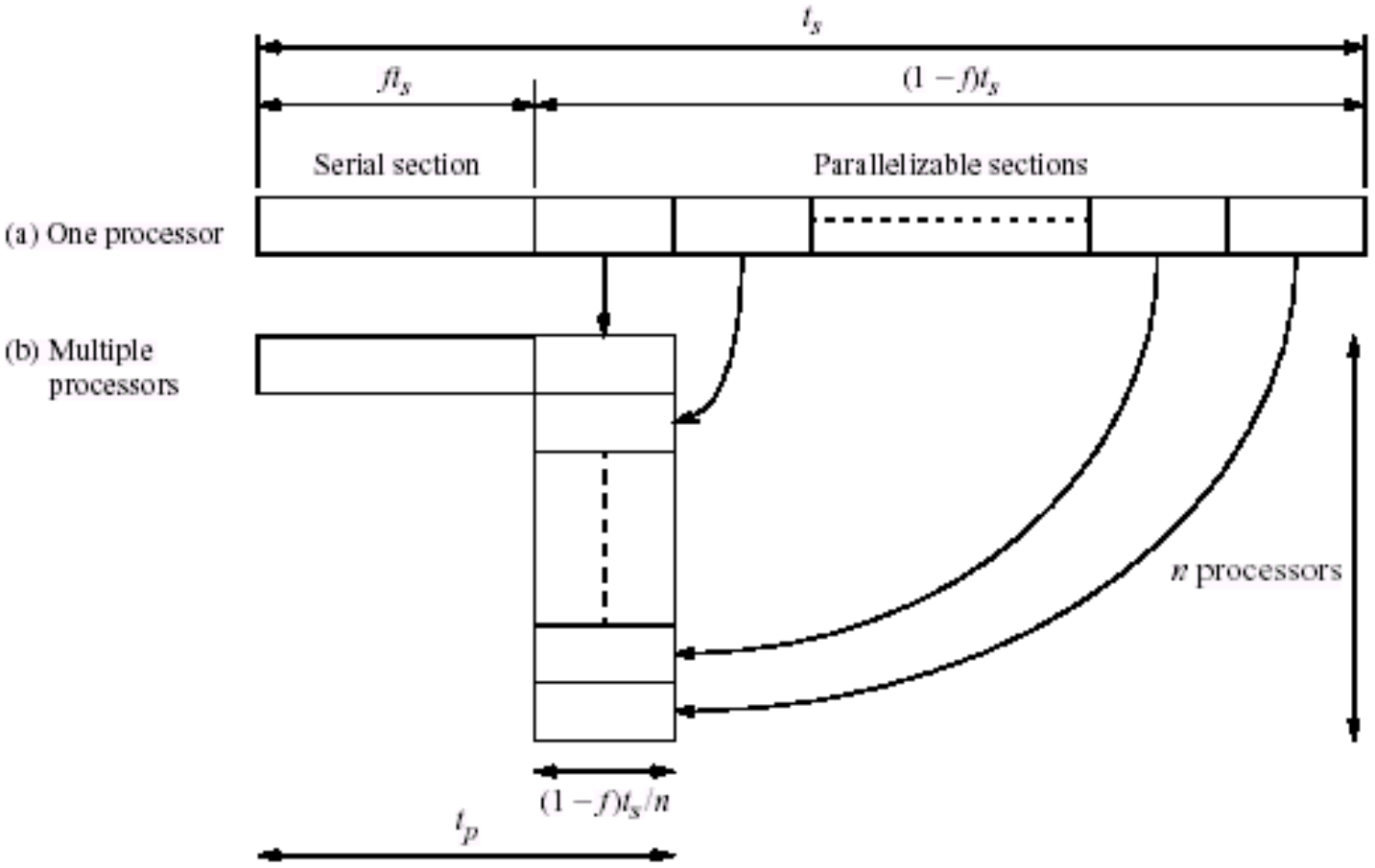
- **Amdahl 定律**
- Gustafson 定律
- Sun and Ni 定律

Amdahl 定律

- 出发点：
  - 固定不变的计算负载；
  - 固定的计算负载分布在多个处理器上的，
  - 增加处理器加快执行速度，从而达到加速的目的。

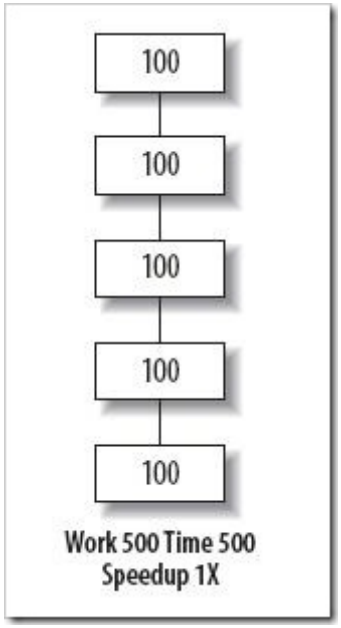
1922-

Amdahl 定律：应用情形



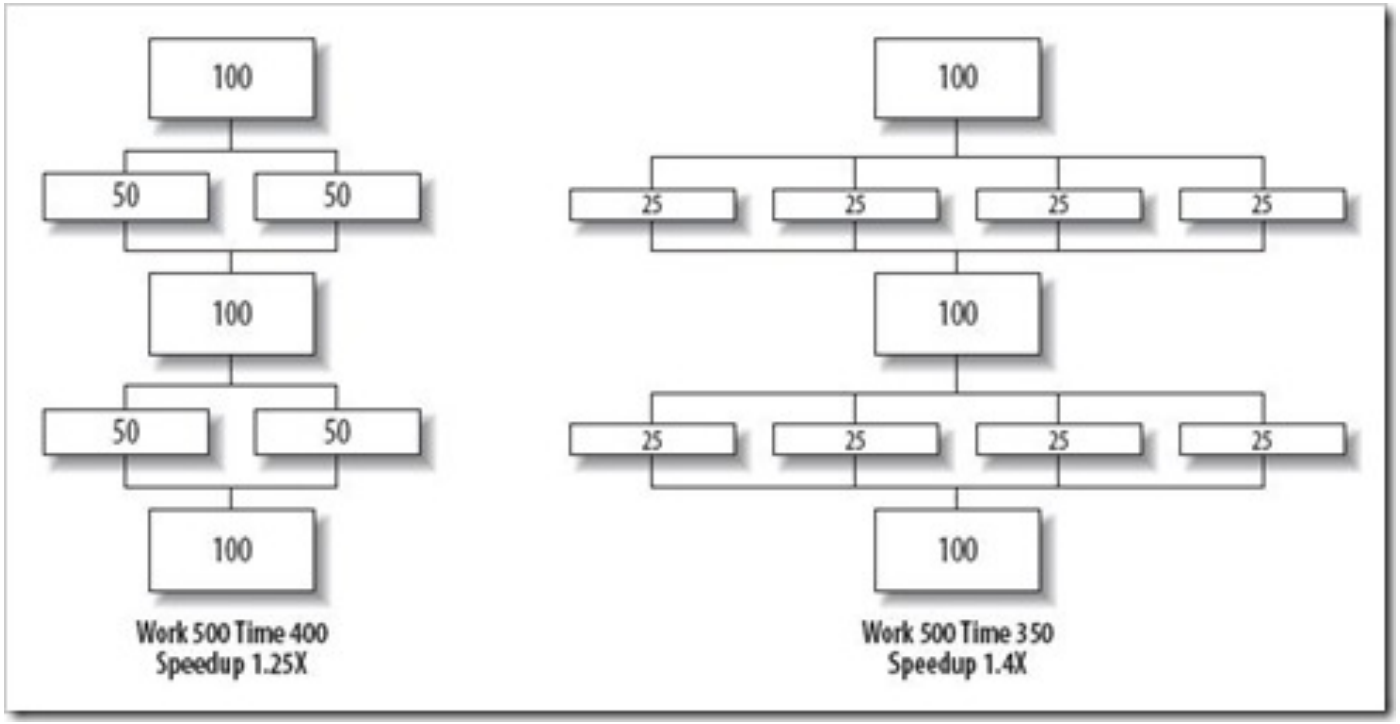
Amdahl 定律：示例（1）

- 程序由 5 个相同的部分组成，且每个部分运行时间均花费 100 秒



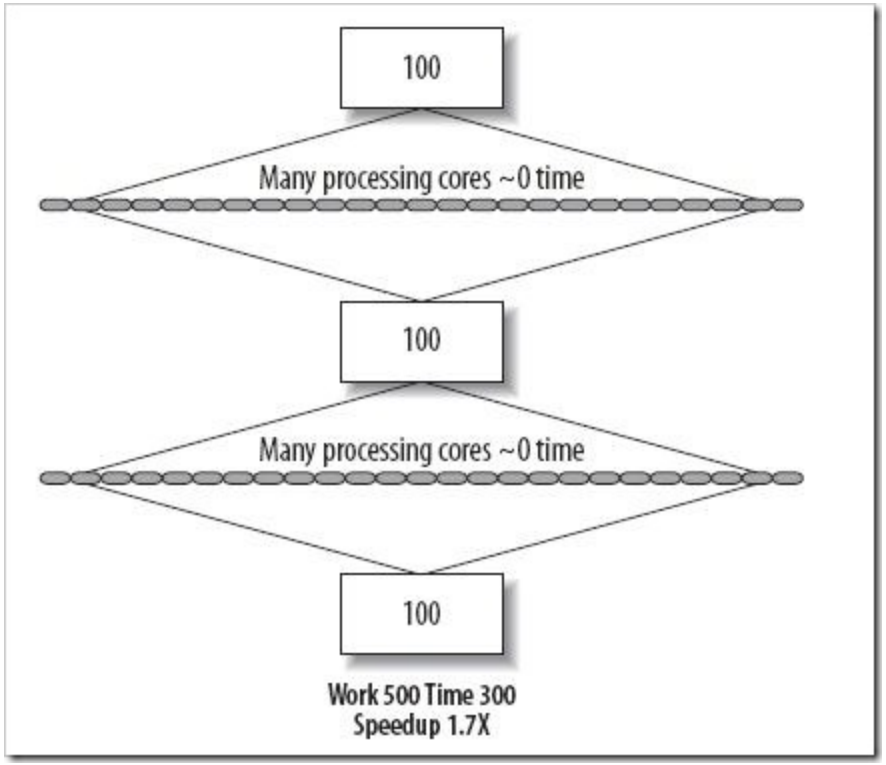
Amdahl 定律：示例（2）

- 并行化其中两个部分



Amdahl 定律：示例（3）

- 增加处理器数



Amdahl 定律：公式

- 固定负载的加速公式：

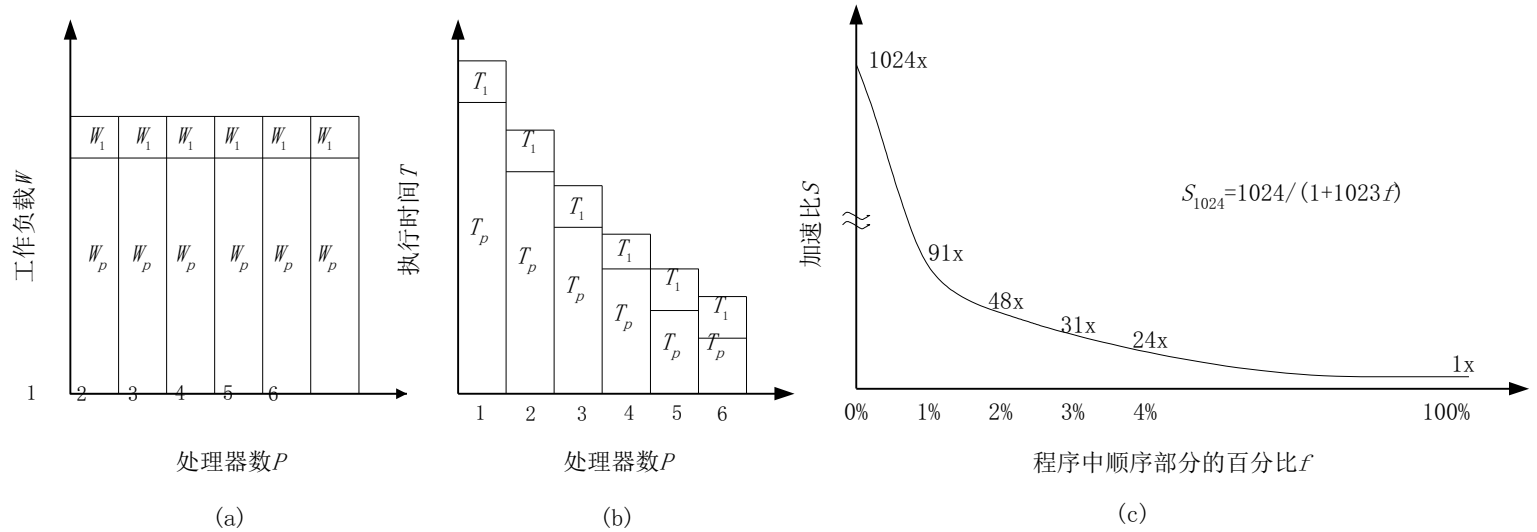
$$S = \frac{Ws + Wp}{Ws + W_P / p}$$

- $W_s + W_p$可相应地表示为f+（1-f）

$$S = \frac{f + (1 \quad f)}{f + \frac{1 \quad f}{p}} = \frac{p}{1 + f(p \quad 1)}$$

- p→∞时，上式极限为：　　S= 1 / f

Amdahl 定律：曲线图



Amdahl 定律：考虑额外开销

- W_O为额外开销

$$S = \frac{W_S + W_P}{W_S + \frac{W_P}{p} + W_O} = \frac{W}{fW + \frac{W(1 \quad f)}{p} + W_O} = \frac{p}{1 + f(p \quad 1) + W_O p / W}$$

加速比性能定律

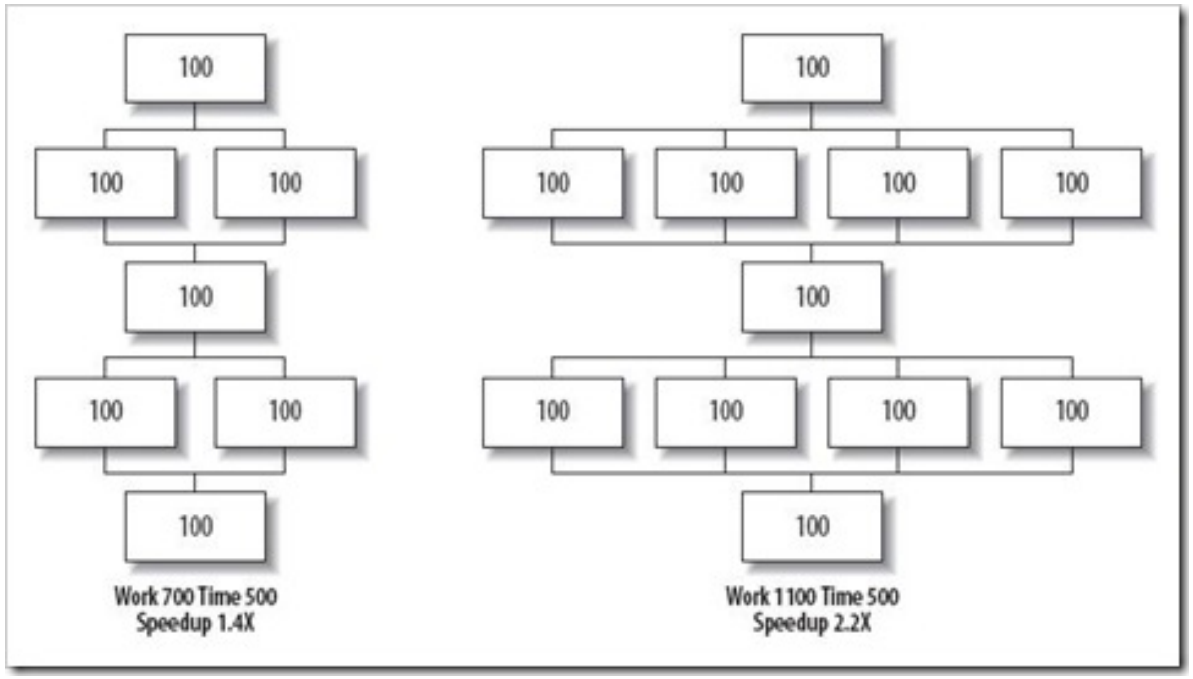
- Amdahl 定律
- **Gustafson 定律**
- Sun and Ni 定律

Gustafson 定律

- 出发点：
  - 对于很多大型计算，精度要求很高，即在此类应用中精度是个关键因素，而计算时间是固定不变的。此时为了提高精度，必须加大计算量，相应地亦必须增多处理器数才能维持时间不变；
  - 除非学术研究，在实际应用中没有必要固定工作负载而计算程序运行在不同数目的处理器上，增多处理器必须相应地增大问题规模才有实际意义。

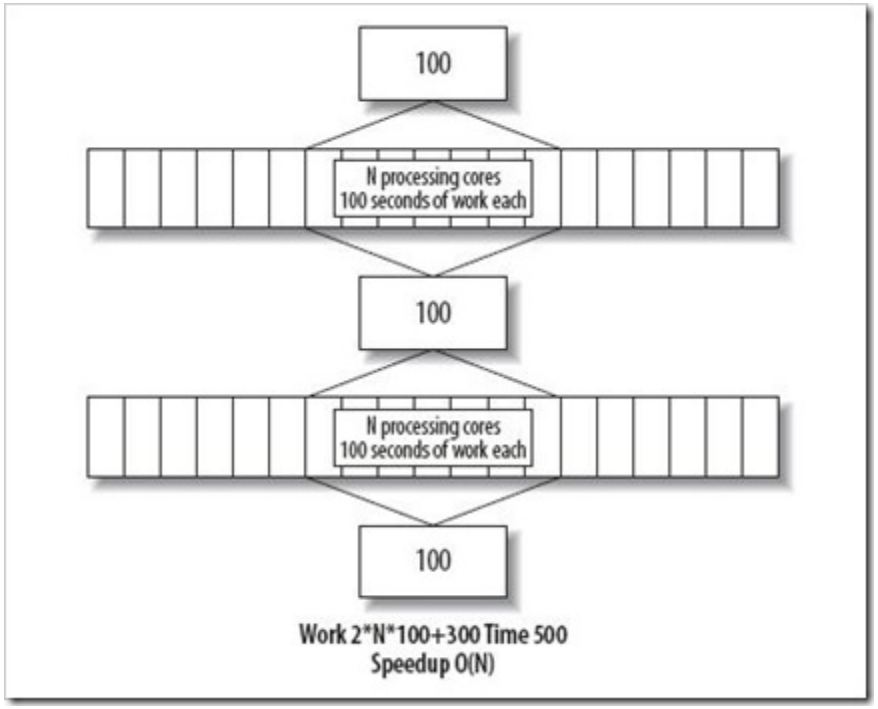
1955-

并行的同时增加计算量


100
100
100
100
100
100
100
Work 700 Time 500
Speedup 1.4X
100
100
100
100
100
100
100
100
100
Work 1100 Time 500
Speedup 2.2X

增加处理器数与计算量

- 串行部分仍然花费相同的时间量，随着其在整体中所占比例的下降，变得越来越不重要。



Gustafson 定律：公式

- Gustafson加速定律：

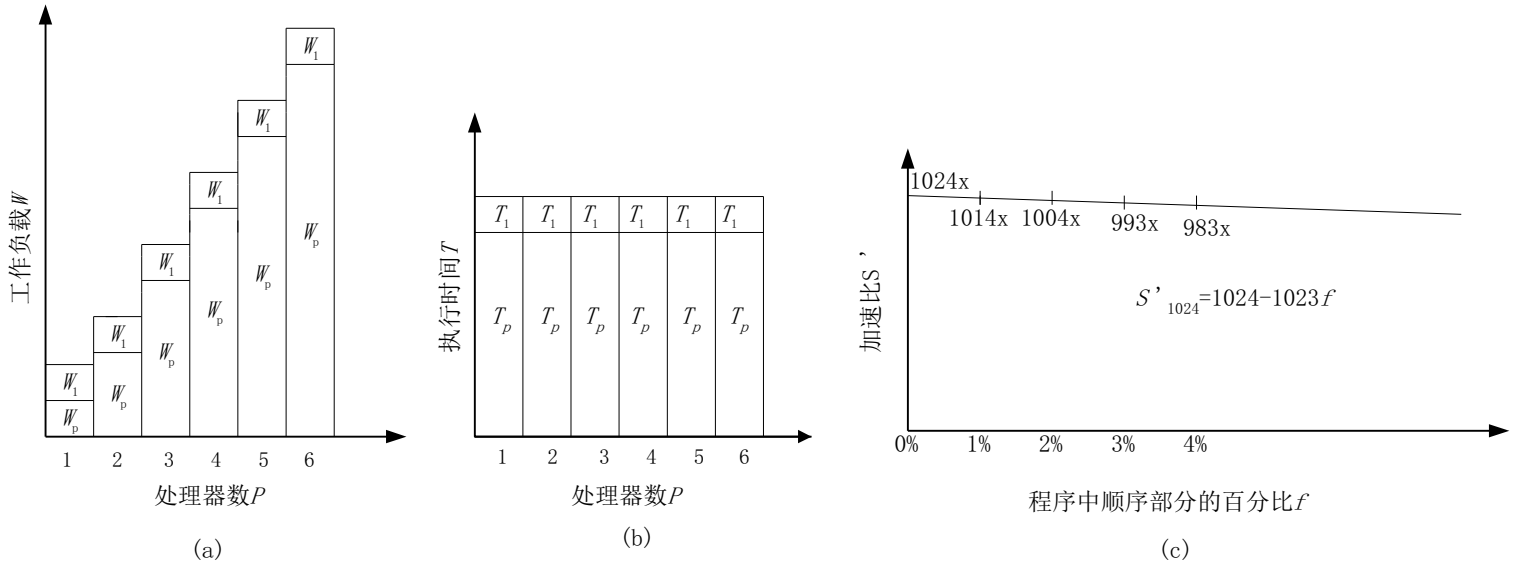
$$S' = \frac{W_S + pWp}{W_S + p\ Wp/p} = \frac{W_S + pWp}{W_S + W_P}$$

$$S' = f + p(1\text{-}f) = p + f(1\text{-}p) = p\text{-}f(p\text{-}1)$$

- 并行开销W_o：

$$S' = \frac{W_S + pW_P}{W_S + W_P + W_O} = \frac{f + p(1 \quad f)}{1 + W_O / W}$$

Gustafson 定律：曲线图



加速比性能定律

- Amdahl 定律
- Gustafson 定律
- **Sun and Ni 定律**

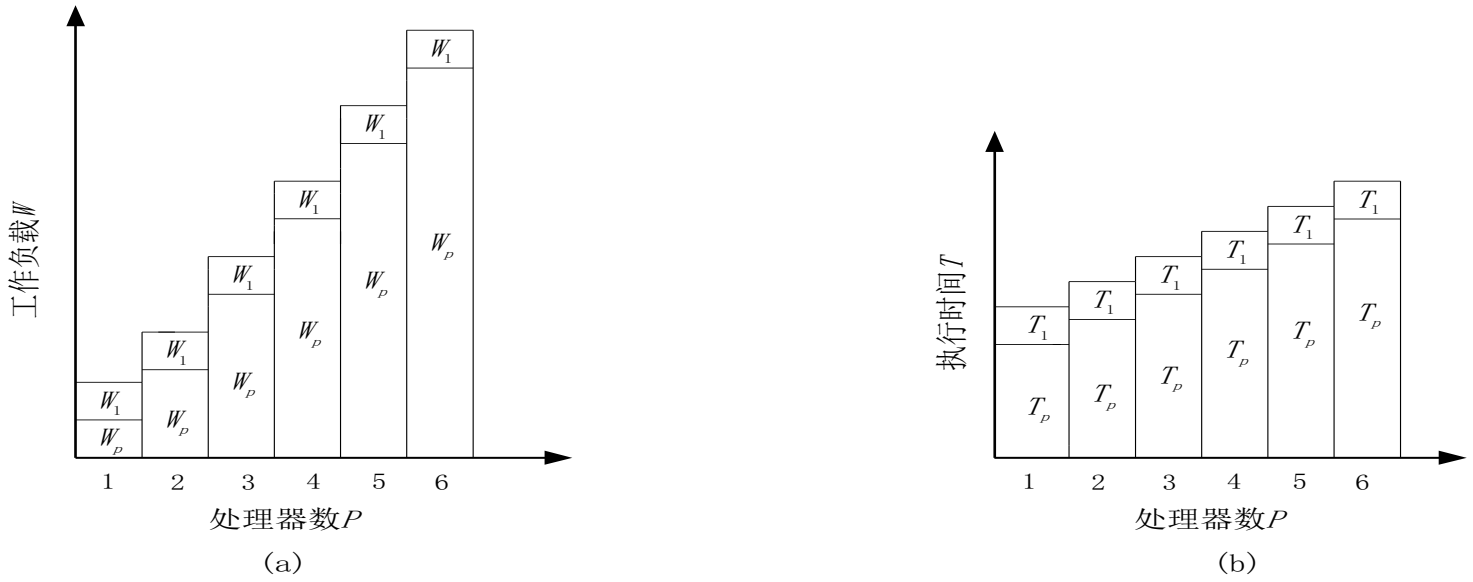
Sun-Ni定律

- 基本思想：
  - 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时可能使执行时间略有增加）。
  - 假定在单节点上使用了全部存储容量M并在相应于W的时间内求解之，此时工作负载W= fW + （1-f）W。
  - 在p 个节点的并行系统上，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）反应存储容量增加到p倍时并行工作负载的增加量，所以扩大后的工作负载W = fW + （1-f）G（p）W。
- 存储受限的加速公式 ：
- 并行开销W=0：

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

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

Sun-Ni定律(cont'd)



- G（p）=1时就是Amdahl加速定律；
- G（p）=p 变为 f + p（1-f），就是Gustafson加速定律
- G（p）>p时，相应于计算机负载比存储要求增加得快，此时 Sun和 N i 加速均比 Amdahl 加速和 Gustafson 加速为高。