# 并行计算

## 第五讲 并行程序性能分析与调优

何克晶
kejinghe@gmail.com
华南理工大学
计算机科学与工程学院



華南理工大學
South China University of Technology

# 计算机的性能

- 用户
  - 通常指计算机的速度，即程序执行时间的倒数。
  - 啥是程序执行时间
    - ✓ 用户向计算机送入一个任务，到获得结果的时间。
    - ✓ 访问磁盘和访问存储器的时间，CPU运算时间，I/O动作时间以及操作系统的开销时间。
    - ✓ 并行机中 ??? CPU运算时间
- 管理员
  - 通常指吞吐率，即单位时间内能完成的工作量。

# 性能评测层次

- 机器级的性能评测
  - cpu和存储器的某些基本性能指标；
  - 并行和通信开销分析；
  - 并行机的可用性与好用性以及机器成本、价格与性价比等；
- 算法级的性能评测
  - 并行算法的加速性能。
  - 并行算法的可扩放性。
- 程序级的性能评测
  - 基准测试程序（Benchmark）：综合测试程序；数学库测试程序；应用测试程序；并行测试程序和商用测试程序。

# CPU的某些基本性能指标

- 工作负载（W）：就是计算操作的数目，通常下面三个物理量来度量。
  - 执行时间
    - 用户向计算机送入一个任务，到获得结果的时间。
    - 影响因素：算法；输入数据集及其数据结构；应用平台和操作系统；使用语言，编译/编辑器和库函数等。
  - 完成的浮点运算数 （FLOP）：
    - 在运算表达式中的赋值操作、变址计算不单独考虑，即0 FLOP；
    - 单独赋值运算、加/减/乘、比较、数据类型转换等，1 FLOP；
    - 除法和开平方，4 FLOP；
    - 正玄、指数等， 8 FLOP。
  - 执行的指令数目 ： 计算单位为百万条
    - 与机器的指令系统有关。

# CPU（2）

- 并行执行时间：

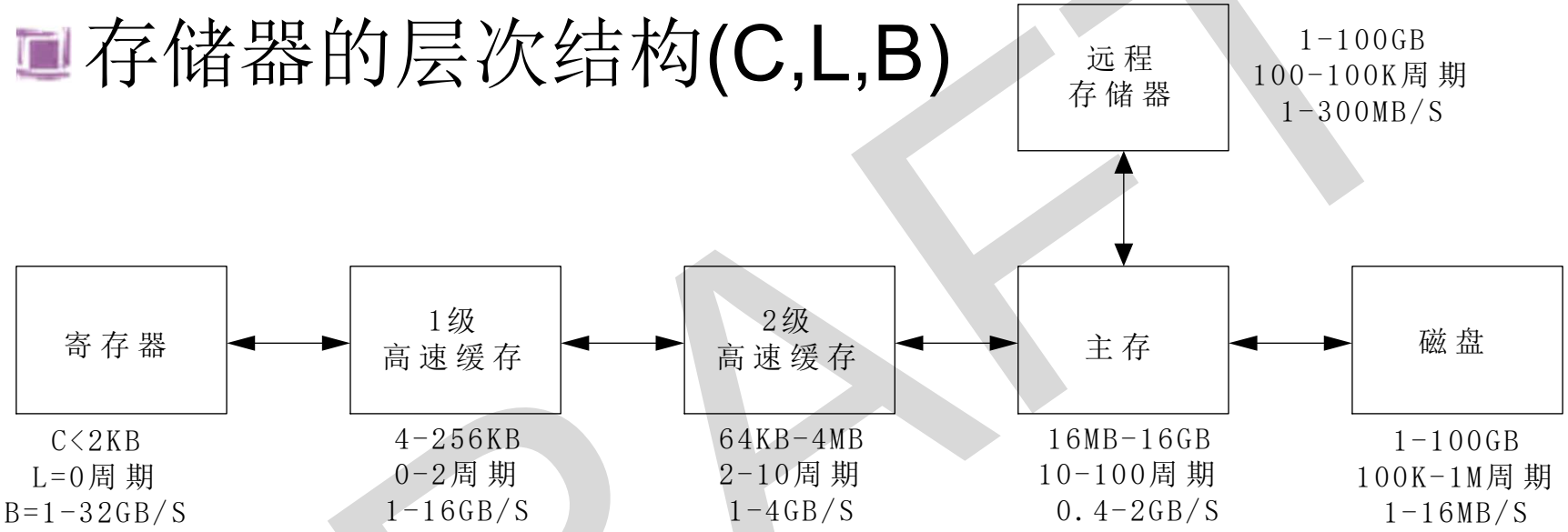
$$T_n = T_{comput} + T_{paro} + T_{comm}$$

  - $T_{comput}$ 为计算时间；
  - $T_{paro}$ 为并行开销时间，
    - 包括进程管理（如进程生成、结束和切换等）时间，组操作（如进程组的生成与消亡等）时间，进程查询（如询问进程的标志、等级、组标志和组大小等）时间；
  - $T_{comm}$ 为相互通信时间
    - 包括同步（如路障、锁、临界区、事件等）时间,通信（如点到点通信、整体通信、读/写共享变量等）时间，聚合操作（如归约、前缀运算等）

- 例子：动态并行程序 vs 静态并行程序

# 存储器性能

## 存储器的层次结构(C,L,B)




- 容量C：各个物理存储器件能保存多少字节的数据；
- 延迟L：读取各个物理器件中一个字的时间；
- 带宽B：1秒内各层物理器件中能传送多少字节；

# 存储器性能（2）

- 估计存储器的带宽
  - RISC 加法运算 8bytes（64位） 100MHz时钟
  - B = $3*8*100*10^6$ B/s= 2.4GB/s

# 并行与通信开销

- 并行（Tparo）和通信开销（Tcomm）：相对于计算很大。

  PowerPC (每个周期 15ns 执行4flops;
  创建一个进程1.4ms 可执行372,000flops)

- 开销的测量：
  - 乒--乓方法（Ping-Pong Scheme）：节点0发送m个字节给节点1；节点1从节点0接收m个字节后，立即将消息发回节点0。总的时间除以2，即可得到点到点通信时间，也就是执行单一发送或接收操作的时间。
  - 可一般化为热土豆法（Hot-Potato），也称为救火队法（Fire-Brigade)：0——1 —— 2 —— … —— -n-1 —— 0

# 并行开销的表达式：点到点通信

- 通信开销 $t(m) = t_0 + m/r_\infty$
  - 通信启动时间 $t_0$
  - 渐近带宽$r_\infty$：传送无限长的消息时的通信速率
  - 半峰值长度$m_{1/2}$：达到一半渐近带宽所要的消息长度
  - 特定性能$\pi_0$：表示短消息带宽
  - 这四个参数的关系

$$t_0 = m_{1/2} / r_\infty = 1 / \pi_0$$

# 并行开销的表达式：整体通信

- 典型的整体通信有：
  - 播送（Broadcasting）：处理器0发送m个字节给所有的n个处理器
  - 收集（Gather）：处理0接收所有n个处理器发来在消息，所以处理器0最终接收了m n个字节；
  - 散射（Scatter）：处理器0发送了m个字节的不同消息给所有n个处理器，因此处理器0最终发送了m n个字节；
  - 全交换（Total Exchange）：每个处理器均彼此相互发送m个字节的不同消息给对方，所以总通信量为$mn^2$个字节；
  - 循环移位（Circular-shift）：处理器i发送m个字节给处理器i+1，处理器n-1发送m个字节给处理器0，所以通信量为m n个字节。

# 整体通信（2）

- 通信开销

$$T(m,n) = t_0(n) + m / r_\infty(n)$$

- 路障和通信开销比较表

表 2.2　$SP_2$ 机器的整体通信和路障同步开销表达式一览表

| 整体通信操作 | 表　达　式 |
|---|---|
| 播 送 | 52logn + (0.029logn)m |
| 收集 / 散射 | (17logn +15) + ( 0.025n-0.02)m |
| 全 交 换 | $80\log n + (0.03n^{1.29})m$ |
| 循环移位 | (6logn +60) + (0.003logn + 0.04 )m |
| 路障同步 | 94logn + 10 |

# 机器的可用性

## 可用性 vs 可靠性 vs 服务性

- 可靠性：平均无故障时间MTTF(Mean Time To Fail)，系指系统失效前平均正常运行的时间
- 服务性（可维护性）：平均修复时间MTTR(Mean Time To Repair)，系指系统失效后修理恢复正常工作的时间
- 可用性：Availability = MTTF/(MTTF+MTTR)

  Availability = 1/(1+MTTR/MTTF)

  一个系统可以为用户所使用时间的百分比，即正常运行时间的百分比。

# 机器的好用性

## 用户环境

- 并行机系统内所有支持用户与并行计算机相关的硬件/软件资源的工具的有机结合以及它们呈现给用户的表现形式；
  - 用户环境的系统设计：工具的有机结合；
  - 用户界面设计：整个用户环境的界面设计；

# 机器的好用性（2）

- 用户环境的系统设计
  - 要灵活、易于扩充和集成；
  - 要尽量使用户应用软件的开发与平台无关；
  - 不要求用户了解底层的实现细节，要向用户提供各项服务的借口而且尽量使用统一的标准；
  - 提供单一系统映像（Single System Image）：
    - 单入口（访问）点；
    - 单控制点（在单一控制台上监控系统）；
    - 单一内存映像（单地址空间）；
    - 单一作业管理系统（允许用户以独占或共享方式运行并/串行作业）；
    - 单一文件结构（用户无论在哪台机器上登录，他所看到的文件结构和一台机器上一致）；

# 机器的好用性（3）

- 用户界面
  - 实用性（Utility）；
    - 用户界面应能提供用户所需的各种服务，帮助用户完成所有任务；
  - 高效性（Efficiency）
    - 用户界面应能提供用户所需的各种服务，帮助用户完成所有任务；
  - 易学习性（Learnability）；
  - 交互性：如何使用键盘、鼠标和软件工具进行交互；
  - 美观性；
  - 实现模型，显示模型和概念模型（心理模型）

# 机器的成本、价格与性/价比

- 机器的成本与价格
- 机器的性能/价格比 Performance/Cost Ratio： 系指用单位代价（通常以百万美元表示）所获取的性能（通常以MIPS或MFLOPS表示）
- 利用率（Utilization）： 可达到的速度与峰值速度之比

# 算法级性能评测

- 加速比性能定律
  - 并行系统的加速比是指对于一个给定的应用，并行算法（或并行程序）的执行速度相对于串行算法（或串行程序）的执行速度加快了多少倍。
  - Amdahl 定律
  - Gustafson定律
  - Sun Ni定律
- 可扩放性评测标准
  - 等效率度量标准
  - 等速度度量标准
  - 平均延迟度量标准

# Amdahl vs Gustafson vs Sun Ni

- Amdahl：适用于固定计算负载
  - 实时性要求高的应用中，时间是关键，而计算负载固定不变。当计算负载一定时，增加处理器可提高计算速度。
- Gustafson：适用于可扩放问题
  - 精度要求高的应用中，精度是关键，而计算时间固定不变。当计算时间一定时，加大计算量，即增多处理器，才能提高精度。
- Sun Ni：受限于存储器
  - 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。

# 考虑参数

- 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（$f = W_s/W$，$W_s=W_1$）；
  - $W_P$：应用程序中可并行化部分，1-f为并行分量比例；
  - $W_s + W_p = W$；
- $T_s=T_1$：串行执行时间，$T_p$：并行执行时间；
- S：加速比，E：效率；

# Amdahl vs Gustafson

- Amdahl：适用于固定计算负载

$$S=\frac{Ws+Wp}{Ws+W_P/p} \Rightarrow S=\frac{f+(1-f)}{f+\frac{1-f}{p}}=\frac{p}{1+f(p-1)} \Rightarrow S=1/f$$

$W_s+W_p$可相应地表示为f+（1-f）

$p\to\infty$

- Gustafson：适用于可扩放问题

$$S'=\frac{W_S+pWp}{W_S+p\cdot Wp/p}=\frac{W_S+pWp}{W_S+W_P} \Rightarrow S'=f+p(1-f)=p+f(1-p)=p-f(p-1)$$

$W_s+W_p$可相应地表示为f+（1-f）

$p\to\infty$ S’和几乎成线性关系，其斜率为1-f

# Amdahl vs Gustafson（2）

- Amdahl：适用于固定计算负载

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

W $_o$为额外开销

S= 1 / (f+$W_0$/W)

p→∞

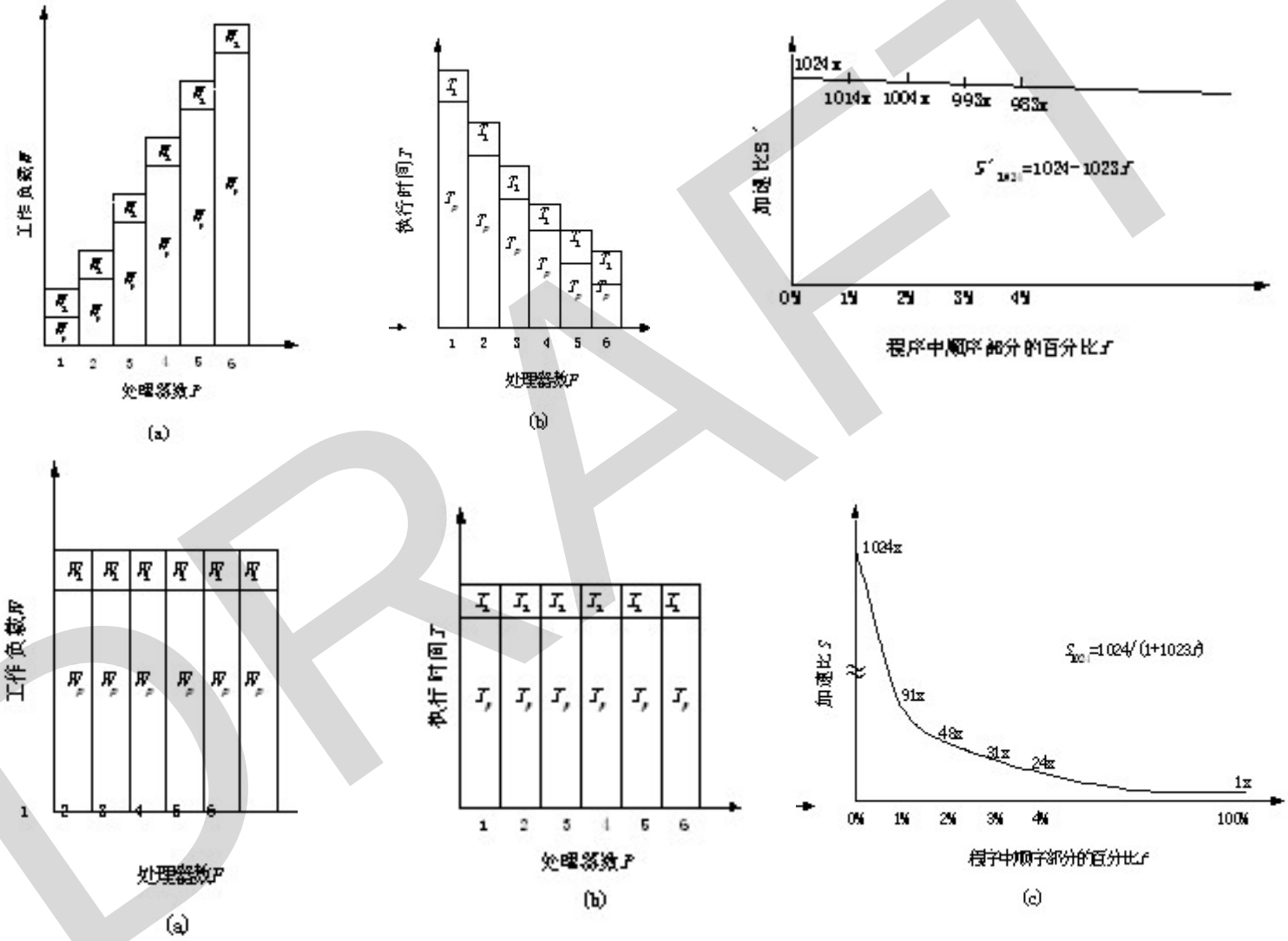
串行分量越大和并行额外开销越大，则加速越小。

- Gustafson：适用于可扩放问题

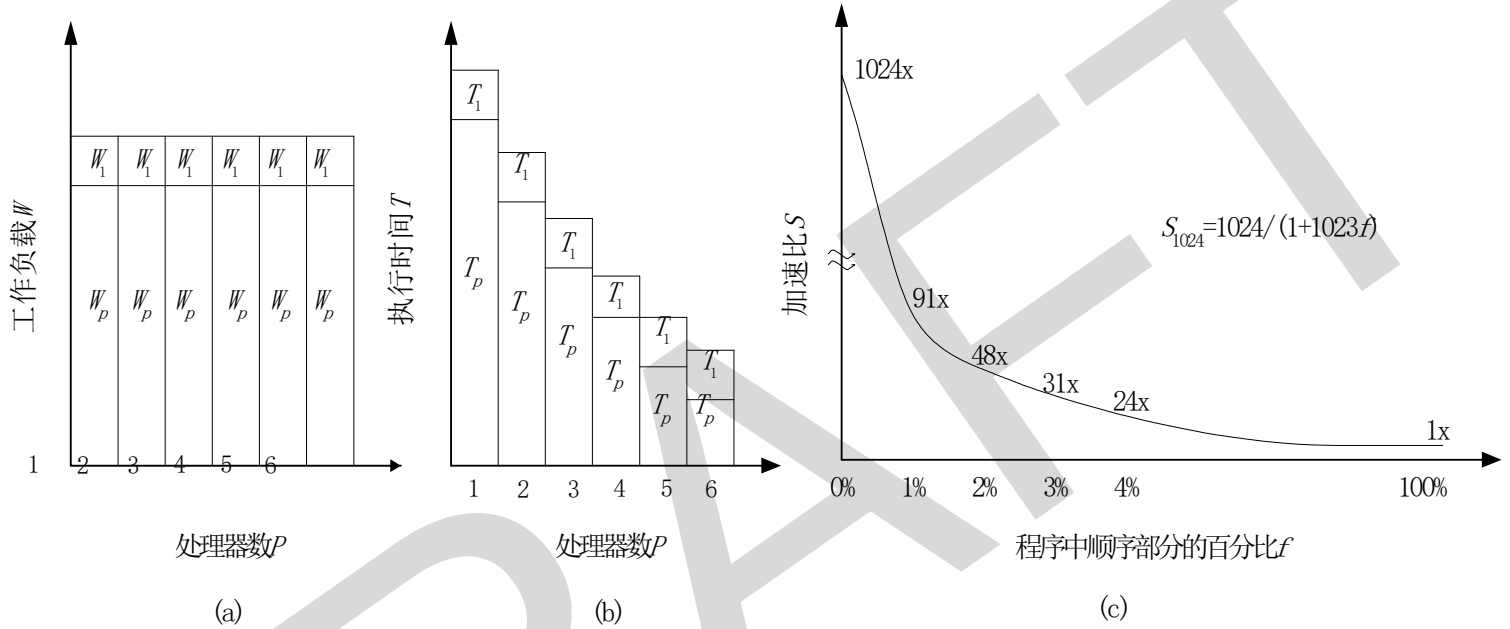
$$S'=\frac{W_S+pW_P}{W_S+W_P+W_O}=\frac{f+p(1-f)}{1+W_O/W}$$

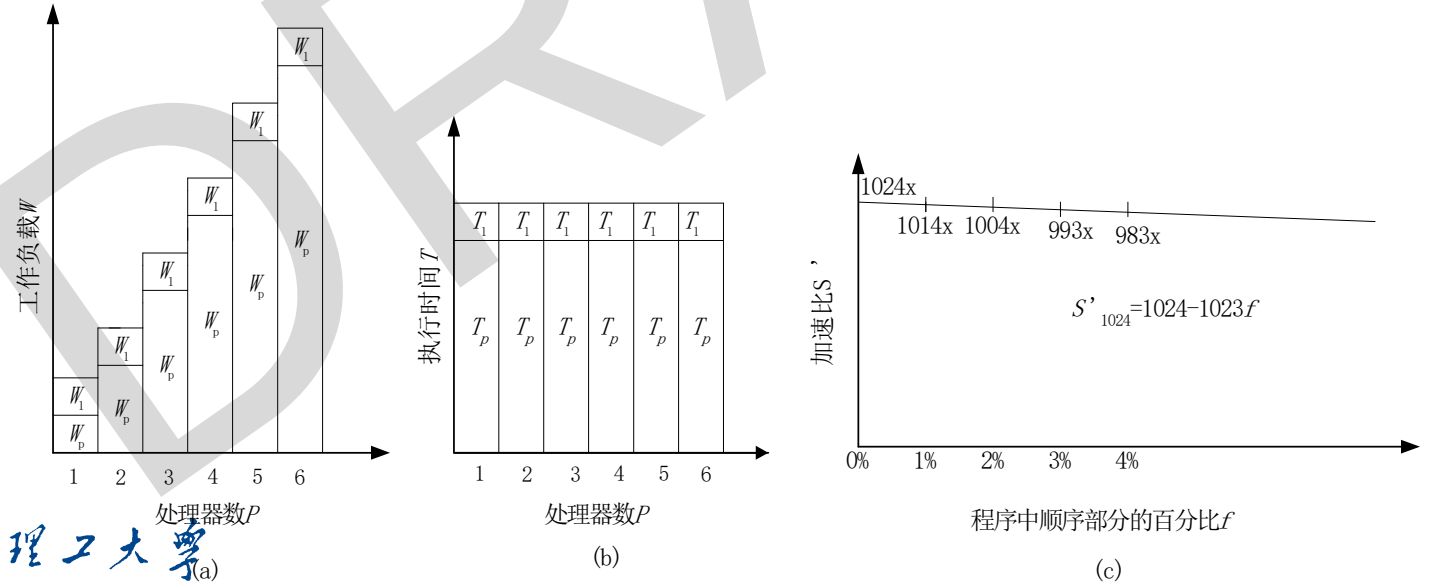
W $_o$为p的函数，可能随p增大、减小或不变。

# Amdahl's law vs Gustafson's



# Amdahl's law vs Gustafson's



# Sun 和 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（此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

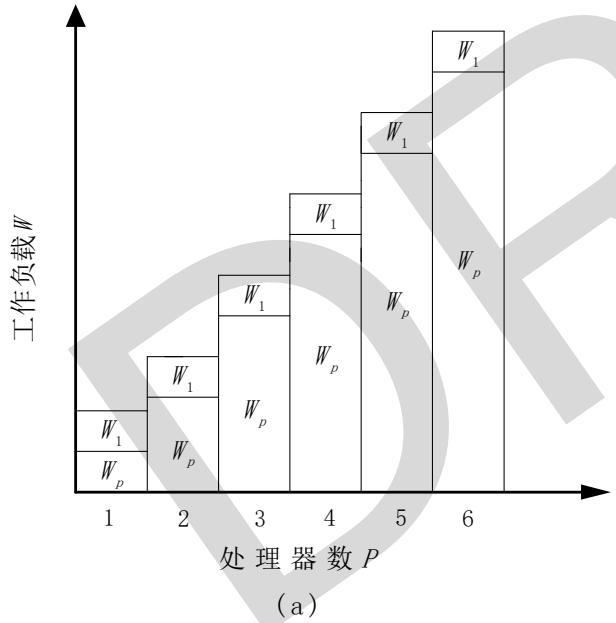
$$S'' = \frac{fW + (1-f)G(p)W}{fW + (1-f)G(p)W/p} = \frac{f + (1-f)G(p)}{f + (1-f)G(p)/p}$$

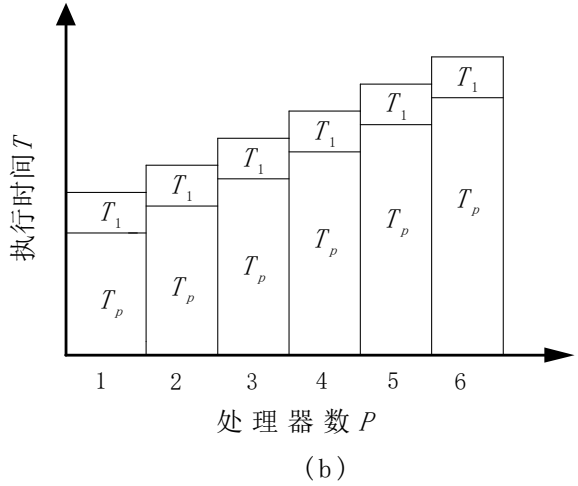
- 并行开销$W_o$:

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

# Sun 和 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。



# 加速比讨论

- 参考的加速经验公式： p/log p≤S≤p
  - p的加速比（也叫线性加速比）：很少通信开销的矩阵相加、内积运算等
  - p/log p的加速 比：分治类的应用问题
- 通信密集类的应用问题 ： S = 1 / C(p)
- 超线性加速 ？？？ 串行-〉并行 算法得到优化
- 绝对加速：最佳并行算法与串行算法 √
- 相对加速：同一算法在单机和并行机的运行时间

# Why可扩放性评测标准

- 加速比：进行并行计算而加速整个计算的过程的能力。
- 影响加速比的因素：处理器数与问题规模
  - 求解问题中的串行分量；
  - 并行处理所引起的额外开销（通信、等待、竞争、冗余操作和同步等）；
  - 加大的处理器数超过了算法中的并发程度。
- 增加问题的规模有利于提高加速的因素：
  - 较大的问题规模可提供较高的并发度；
  - 额外开销的增加可能慢于有效计算的增加；
  - 算法中的串行分量比例不是固定不变的（串行部分所占的比例随着问题规模的增大而缩小）。
- 增加处理器数会增大额外开销和降低处理器利用率
  - 一个特定的并行系统（算法或程序），能否有效利用不断增加的处理器的能力应是受限的，而度量这能力就是可扩放性这一指标。

# What可扩放性评测标准

- 并行计算的可扩放性（Scalability）也是主要性能指标
  - 可扩放性最简朴的含意是在确定的应用背景下，计算机系统（或算法或程序等）性能随处理器数的增加而按比例提高的能力
- 可扩放性是算法和结构的组合，研究可扩放性时，总是将并行算法和体系结构一并考虑
  - 算法的可扩放性: 该算法针对某一特定机器的可扩放性；
  - 体系结构的可扩放性: 该体系结构的机器的某一并行算法的可扩放性
- 并行算法的可扩放性:调整什么和按什么比例调整
  - 并行计算要调整的是处理数p和问题规模W，
  - 两者可按不同比例进行调整，此比例关系（可能是线性的，多项式的或指数的等）就反映了可扩放的程度。

# 可扩放性评测标准（2)

- 可扩放性研究的主要目的：
  - 确定解决某类问题用何种并行算法与何种并行体系结构的组合，可以有效地利用大量的处理器；
  - 对于运行于某种体系结构的并行机上的某种算法当移植到大规模处理机上后运行的性能；
  - 对固定的问题规模，确定在某类并行机上最优的处理器数与可获得的最大的加速比；
  - 用于指导改进并行算法和并行机体系结构，以使并行算法尽可能地充分利用可扩充的大量处理器。
- 前无一个公认的、标准的和被普遍接受的严格定义和评判它的标准

# 等效率度量标准

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

$$T_e = \sum_{i=0}^{p-1} t_e^i = Ts \qquad T_o = \sum_{i=0}^{p-1} t_o^i$$

- $T_p$ 是p个处理器系统上并行算法的运行时间，对于任意i显然有$T_p = t_{ie} + t_{io}$，且 $T_e + T_o = p * T_p$

- 问题的规模W定义为最佳串行算法所完成的计算量，即 $W=T_e$

# 等效率度量标准（2）

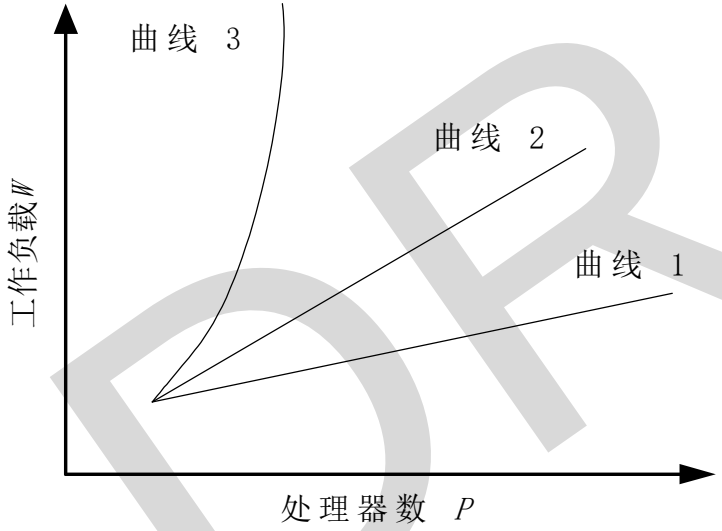
- 并行算法的加速比 vs 效率

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

- 如果问题规模W 保持不变，处理器数p增加，开销$T_o$增大，效率E下降。
- 为了维持一定的效率（介于0与1之间），当处理数p增大时，需要相应地增大问题规模W的值。
- 由此定义函数$f_E$（p）为问题规模W随处理器数p变化的函数，为等效率函数（ISO-efficiency Function）（Kumar1987）

# 等效率度量标准（3）

- 随着系统规模的增加(处理器数目的增加)，测量增加多少运算量会保持效率不变；
- 增加越少表明可扩放性越好；




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

# 等效率度量例子

例子：用p个处理器算N个数之和

- $T_p = N/p + 2\log p$, $T_s = N$
  - $p = 4, N = 64, E = T_s/pT_p = 64/4(16+4) = 64/80 = 0.8$
  - $p = 8, N = 192, E = 192/8(24+6) = 192/240 = 0.8$
- 求等效率函数 (E=0.8) ?
- E=0.8的等效率函数是$8p \log p$
- 如果 $N<8p \log p$
  - $p = 8, N = 144, E = 144/8(18+6) = 144/192 = 0.75$
- 如果 $N>8p \log p$
  - $p = 8, N = 384, E = 352/8(44+6) = 352/400 = 0.88$

# 等效率度量标准（4）

$$E = \frac{S}{P} = \frac{1}{1 + \frac{T_o}{T_e}} = \frac{1}{1 + \frac{T_o}{W}}$$

- 计算T0是等效率函数的唯一关键参数；
- T0通常包括了通信、同步、等待等非有效计算时间；
- 在共享存储的并行计算机中，T0主要是非局部访问的读/写时间，进程调度时间，存储竞争时间，以及高速缓存一致性操作时间等，这些都是难以准确计算的；
- 解析计算：等效率度量
- 试验测试为主：等速度测量和平均延迟

# 等速度度量标准

- 速度：
  - 常以每秒多少次浮点运算(Flops)来表明。浮点运算数目也可以看作为工作负责W。
- 在并行系统中，提高速度可以用增加处理器的方法。
- 如果速度能以处理器的增加而线性增加（即平均速度不变）,则说明有很好的扩放性。

# 等速度度量标准（2）

- p 表示处理器个数，W表示要求解问题的工作量或称问题规模（在此可指浮点操作个数），T为并行执行时间，定义并行计算的速度V=W(工作量)/ T(并行时间)

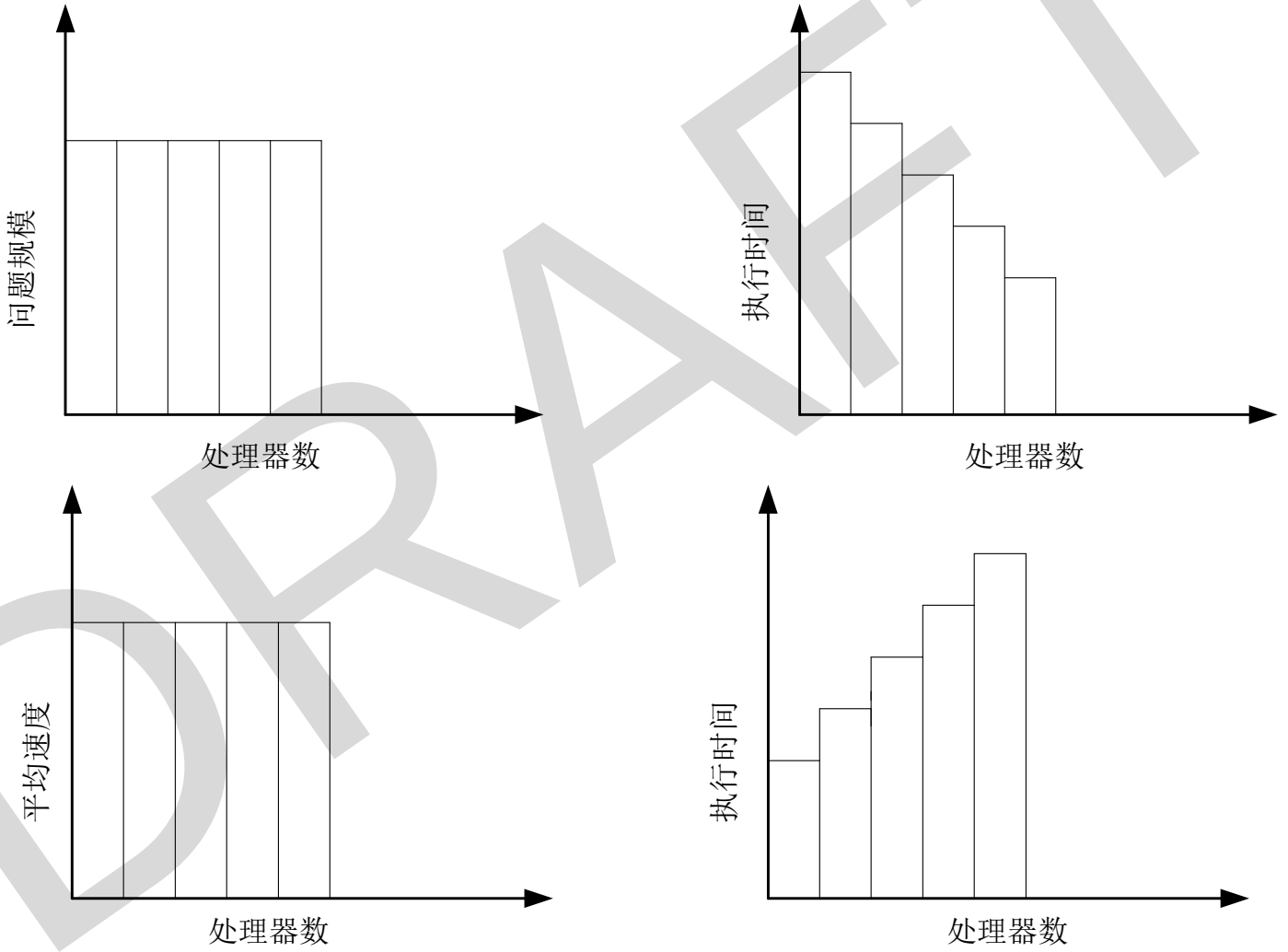
- p个处理器的并行系统的平均速度定义为并行速度V除以处理器个数 p：

$$\overline{V} = \frac{V}{p} = \frac{W}{pT}$$

- W是使用p个处理器时算法的工作量，令W’表示当处理数从p增大到p’时，为了保持整个系统的平均速度不变所需执行的工作量，则可得到处理器数从 p到p’时平均速度可扩放度量标准公式

$$\Psi(p,p') = \frac{W/p}{W'/p'} = \frac{p'W}{pW'}$$

# 等速度度量标准（cont'd）



# 平均延迟度量标准

- $T_i$为$P_i$的执行时间，包括包括延迟$L_i$，Pi的总延迟时间为“L i+启动时间+停止时间”。定义系统平均延迟时间为

$$\overline{L}(W,p)=\sum_{i=1}^{p}\left(T_{para}-T_i+L_i\right)\Big/p$$

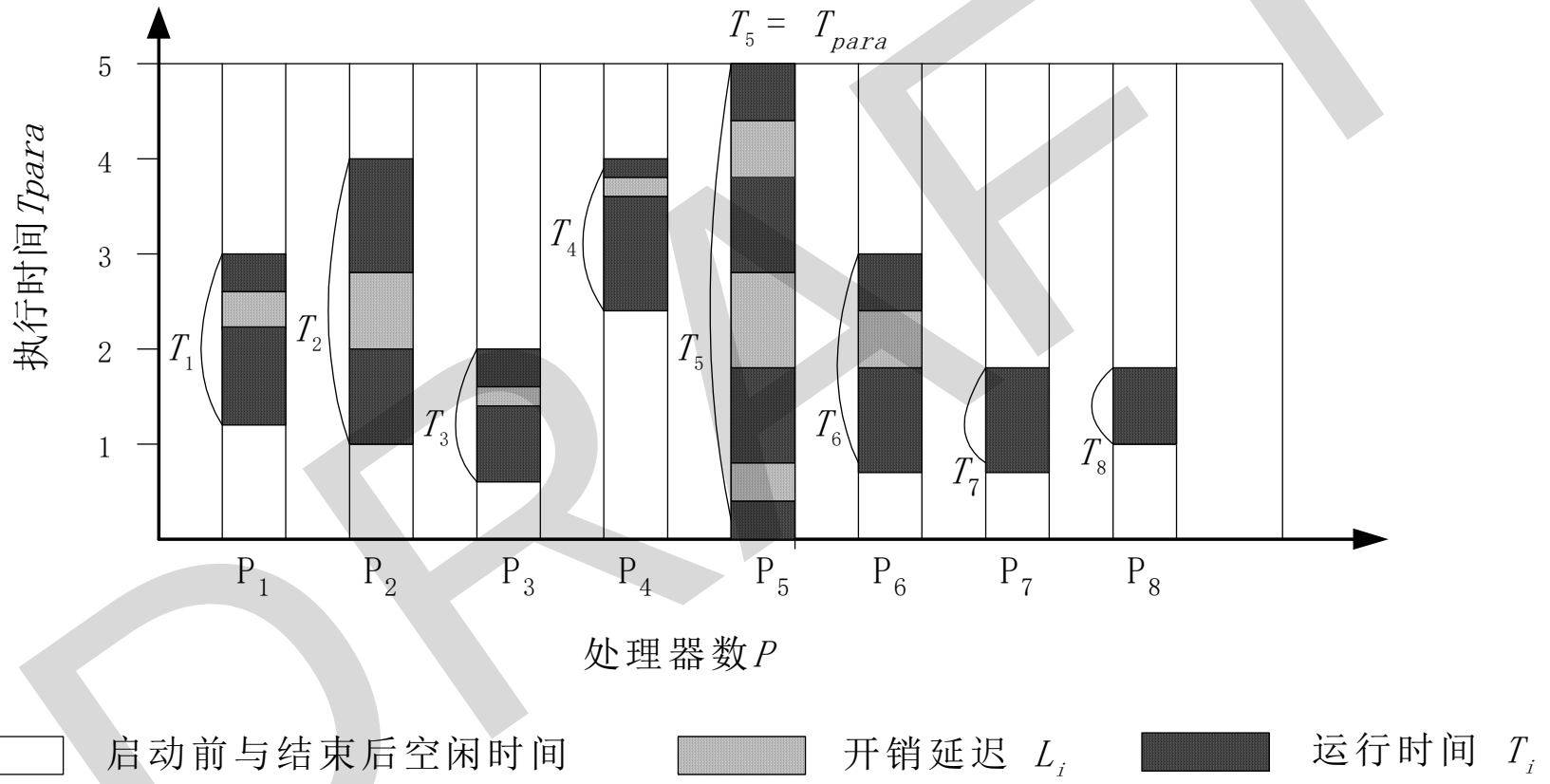
- $pT_{para}=T_o+T_s$ $\qquad T_o=p\overline{L}(W,p)$

$$\overline{L}(W,p)=T_{para}-T_{seq}/p$$

- $\overline{L}(W,p)$ 在p个处理器上求解工作量为W问题的平均延迟
- $\overline{L}(W',p')$在p’个处理器上求解工作量为W’问题的平均延迟当处理器数由p变到p’，而推持并行执行效率不变，则定义平均延迟可扩放性度量标准为

$$\Phi(E,p,p')=\frac{\overline{L}(W,p)}{\overline{L}(W',p')}$$

# 平均延迟度量标准（Cont'd)



# 三种标准比较

| | 优点 | 缺点 |
|---|---|---|
| 等效率度量标准 | 简单可定量计算的、少量的参数计算等效率函数； | 如果$T_o$无法计算出（在共享存储并行机中）； |
| 等速度度量标准 | 直观地使用易测量的机器性能速度指标来度量； | 某些非浮点运算可能造成性能的变化； |
| 平均延迟度量标准 | 平均延迟能在更低层次上衡量机器的性能； | 需要特定的软硬件才能获得平均延迟； |