

# 第五讲 性能评估



# 学习目标

- 预测并行程序的性能。
- 了解限制并行程序性能的障碍。



# 性能度量

- 执行时间
- 加速比
- 效率
- 可扩展性
- 其他：可移植性、编程能力等。



# 目录

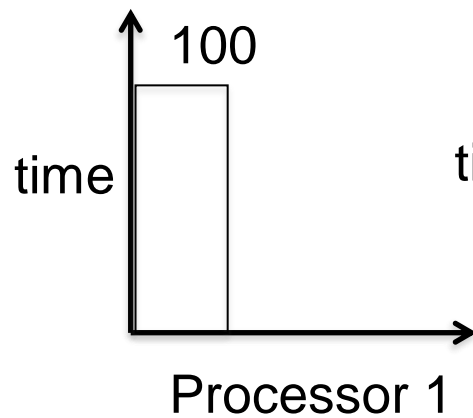
- 加速比和效率
- Amdahl定律
- 可扩展性

# 加速比

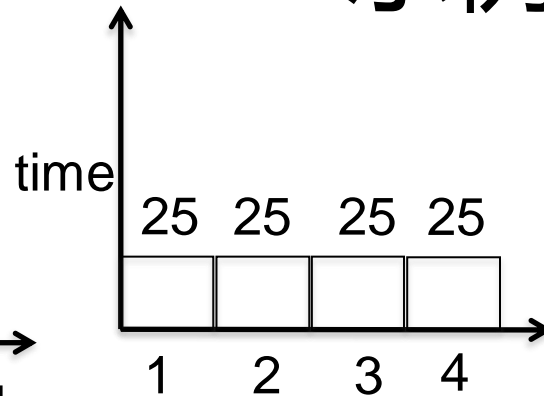
$$Sp = \frac{T_s}{T_p}$$

- $T_s$  = 串行算法的运行时间
- $T_p$  = 使用 $p$ 个处理器的并行算法的运行时间

# 示例



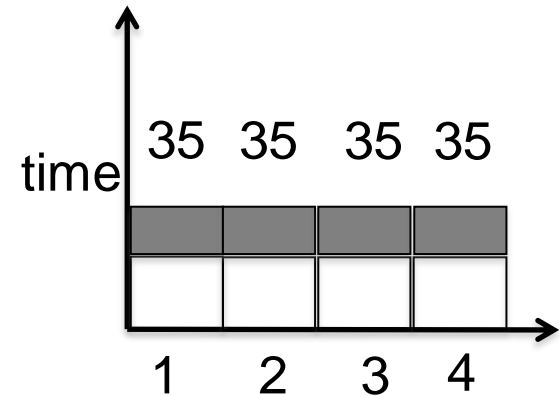
(a)



(b)

$$S_p = 100/25 = 4.0$$

完美并行化



(c)

$$S_p = 100/35 = 2.85$$

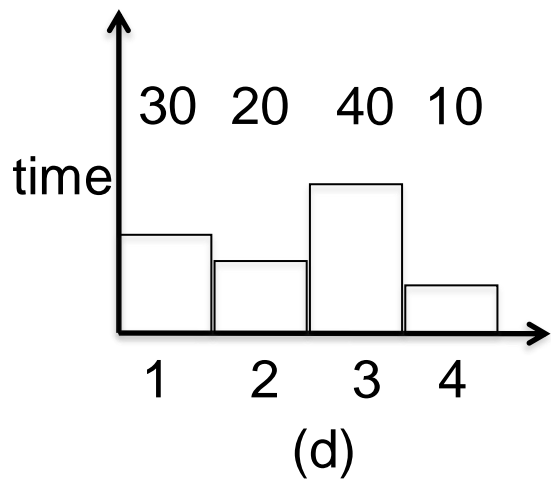
完美的负载平衡

但代价为10

(串行部分&并行开销)



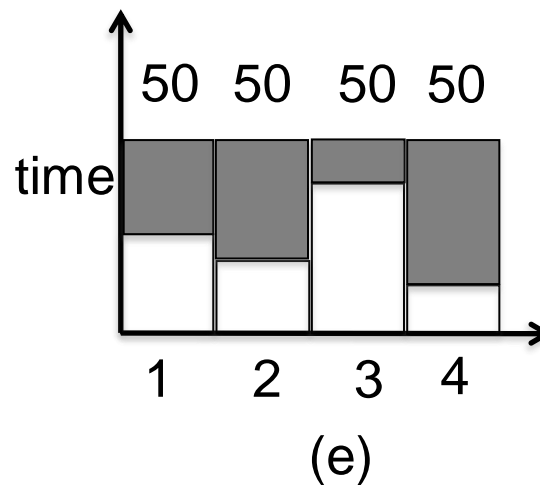
# 示例



$$Sp=100/40=2.5$$

无并行代价

但存在负载不均衡



$$Sp=100/50=2$$

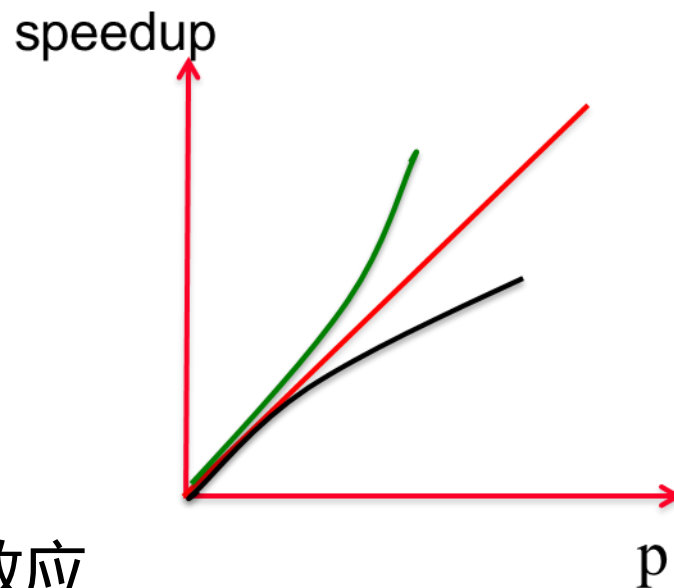
负载不均衡

并行代价



# 好的加速比是怎样的

- 线性加速比
- 亚线性加速比
- 超线性加速比
  - 硬件特性 : 内存层次、缓存效应
  - 算法 : 深度优先树遍历





# 超线性加速比

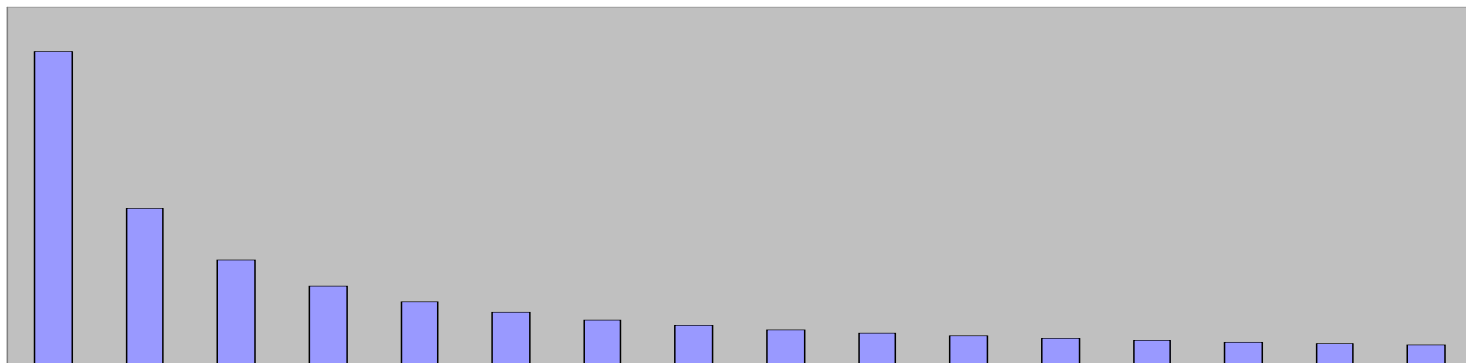


# 加速比

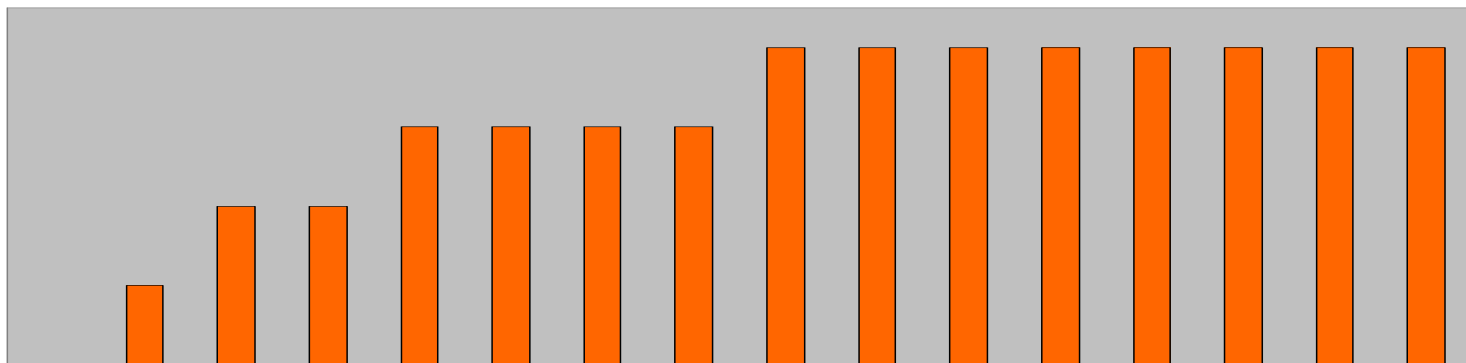
$$y(n, p) \propto \frac{S(n) + f(n)}{S(n) + f(n)/p + k(n, p)}$$



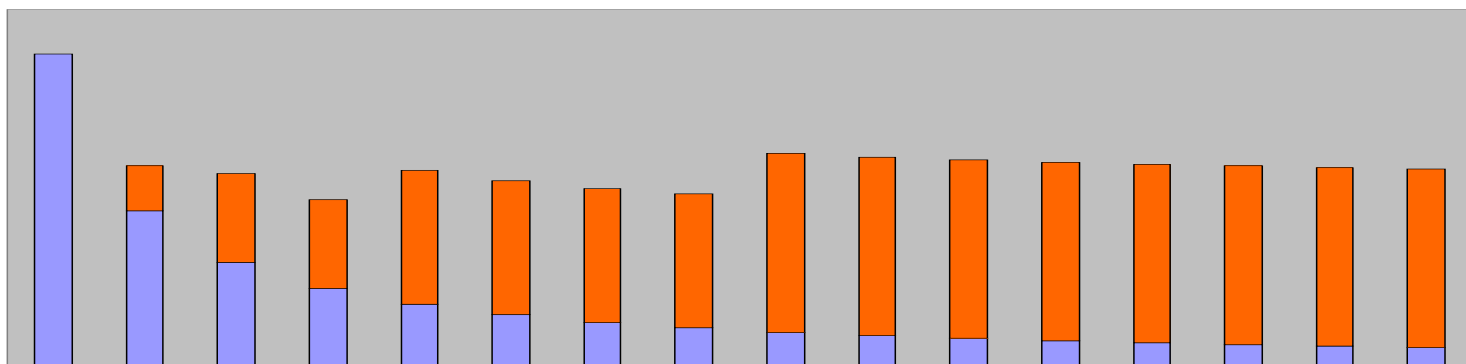
$$\varphi(n)/p$$



$$\kappa(n,p)$$

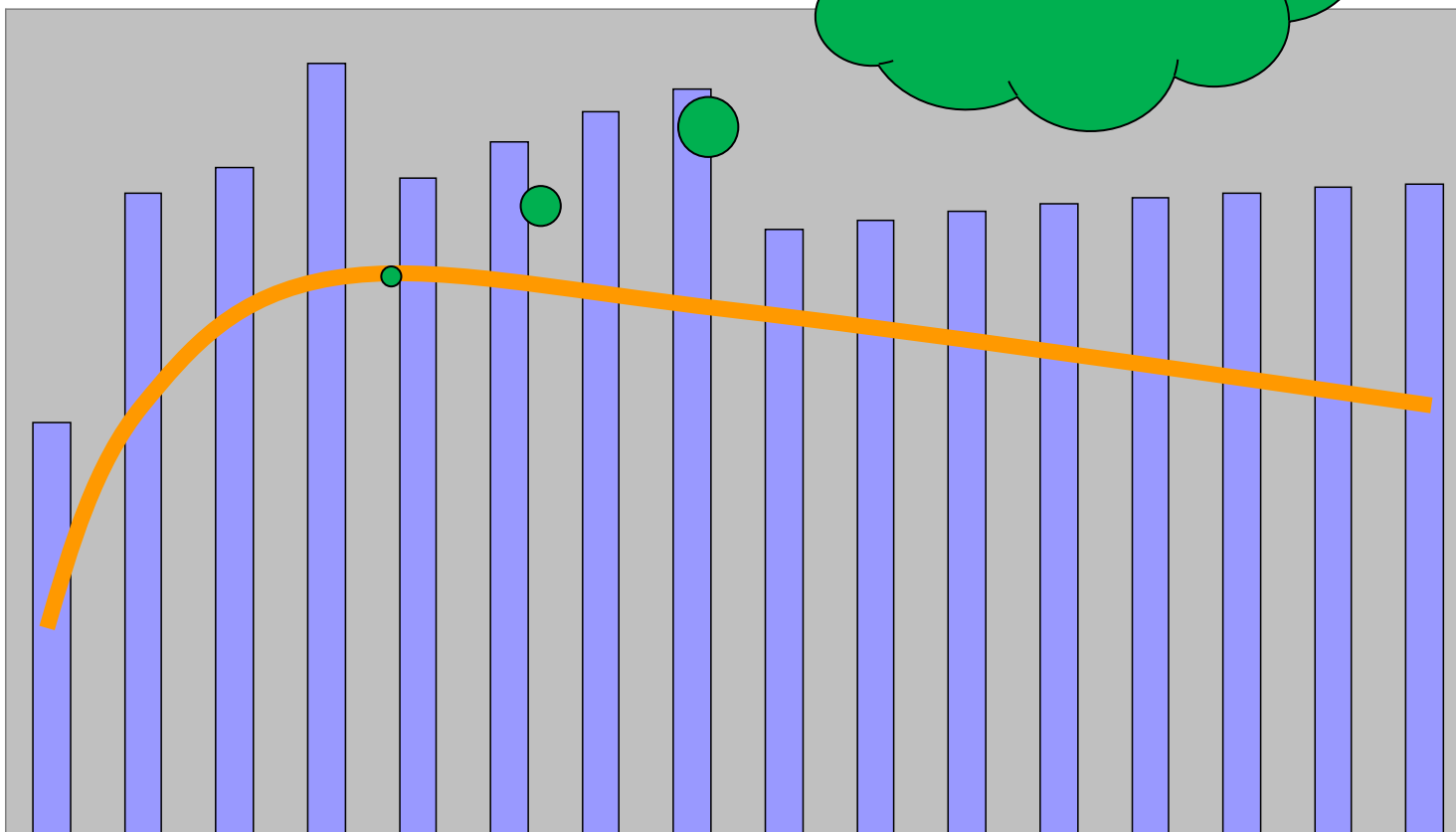


$$\varphi(n)/p + \kappa(n,p)$$



# 加速比图

“弯曲折线”  
拐点的位置？



# 效率

$$E_p = S_p/p$$

- 实际使用处理能力的比例。
- 具有线性加速比的程序效率为100%。



$$0 \leq \varepsilon(n, p) \leq 1$$

$$\varepsilon(n, p) \leq \frac{\sigma(n) + \varphi(n)}{p\sigma(n) + \varphi(n) + p\kappa(n, p)}$$

所有项 $\geq 0 \Rightarrow \varepsilon(n, p) \geq 0$ 。

$$0 \leq \varepsilon(n, p) \leq 1。$$



# 目录

- 加速比和效率
- Amdahl定律
- 可扩展性



# Amdahl定律

$$\begin{aligned}\psi(n, p) &\leq \frac{\sigma(n) + \varphi(n)}{\sigma(n) + \varphi(n) / p + \kappa(n, p)} \\ &\leq \frac{\sigma(n) + \varphi(n)}{\sigma(n) + \varphi(n) / p}\end{aligned}$$

Let  $f = \sigma(n) / (\sigma(n) + \varphi(n))$

$$\psi \leq \frac{1}{f + (1 - f) / p}$$





# Amdahl定律

$$y \leq \frac{1}{f + (1 - f) / p}$$

- $f$  是计算中必须串行执行的操作的比例,  $0 \leq f \leq 1$ 。
- 使用  $p$  个处理器执行计算所能达到的最大加速比。
- 除非串行政程序所有操作都被并行化, 否则可能的加速比将非常有限 —— 不管可用的内核数是多



# 例 1

- 可以并行化串程序的90%。无论使用多少内核p，并行化都是“完美”的。  $T_{\text{serial}} = 20$ 秒

- 并行部分的执行时间：

$$0.9 \times T_{\text{serial}} / p = 18 / p$$

- 不可并行部分的执行时间：

$$0.1 \times T_{\text{serial}} = 2$$

- 总并行执行时间：

$$T_{\text{parallel}} = 0.9 \times T_{\text{serial}} / p + 0.1 \times T_{\text{serial}} = 18 / p + 2$$

- 加速比

$$S = \frac{T_{\text{serial}}}{0.9 \times T_{\text{serial}} / p + 0.1 \times T_{\text{serial}}} = \frac{20}{18 / p + 2}$$

$$S \leq T_{\text{serial}} / (0.1 \times T_{\text{serial}}) = 20 / 2 = 10$$



## 例 2

- 程序95%的执行时间发生在一个可以并行执行的循环内。如果使用8个CPU执行程序的并行版本，期望的最大加速比为多少？

$$\psi \leq \frac{1}{0.05 + (1 - 0.05) / 8} \cong 5.9$$



# 例 3

- 如果并行程序中25%的操作必须串行执行，最大可达到的加速比是多少？



# 例 3

- 假设我们已经实现了一个串程序的并行版本，其时间复杂度为 $\Theta(n^2)$ ，其中 $n$ 是数据集大小。
- 假设输入数据集和输出结果所需的时间是  $(18000+n)$  微秒
- 程序的计算部分可以并行执行；其执行时间为  $(n^2/100)$  微秒。
- 问在大小为10000的问题上，这个并行程序可以实现的最大加速比是多少？

# Amdahl定律的局限性

- 忽略了  $\kappa(n,p)$
- 高估了可实现的加速比。

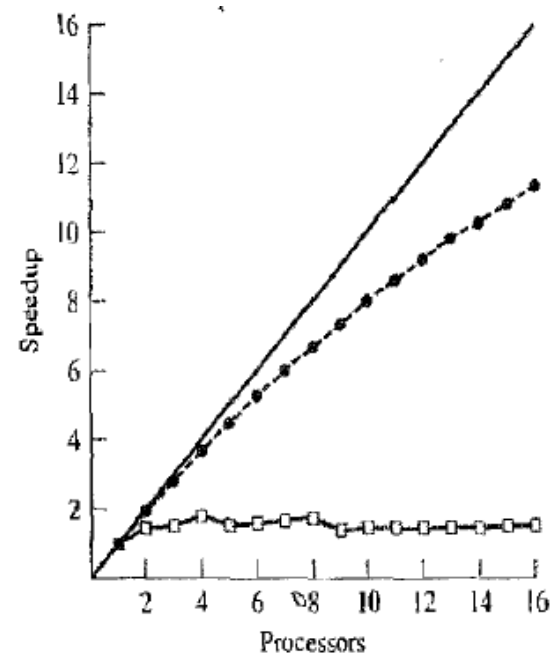


## 例 3(2)

- 考虑通信
- 基于类超立方互联网络
- 通信时间:  $10000[\log p] + (n/10) \mu\text{sec}$
- 加速比曲线应该是怎样的?

$$\psi \leq \frac{(28\,000 + 1\,000\,000)}{(42\,000 + 1\,000\,000/p + 140\,000 \lceil \log p \rceil)}$$

- 按照Amdahl定律预测的加速比(虚线)高于考虑通信开销的加速比预测 (实线)。

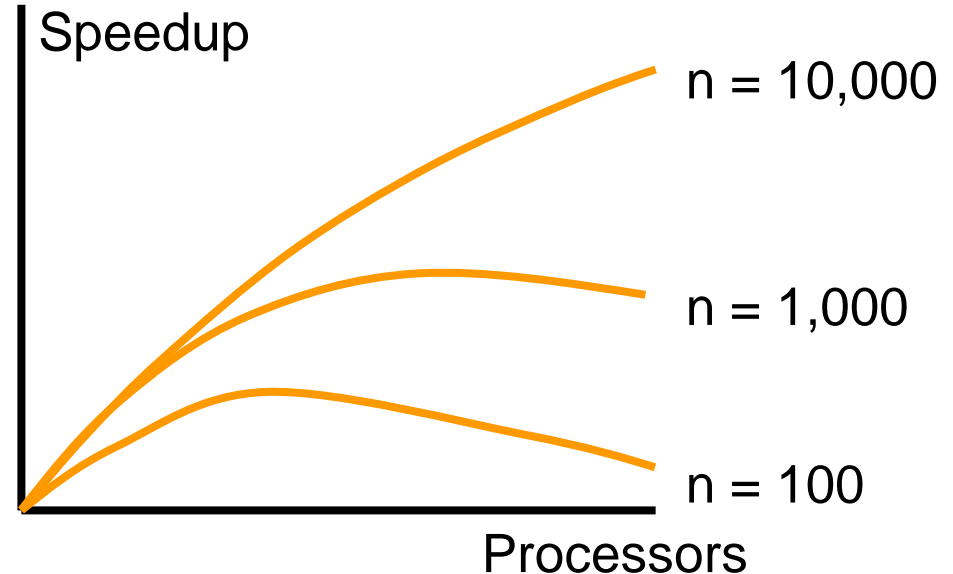


# Amdahl效应

- 通常，函数  $\kappa(n,p)$  的复杂度比函数  $\phi(n)/p$  低。
- 随着  $n$  的增加， $\phi(n)/p$  支配了  $\kappa(n,p)$ 。
- 对于固定数量的处理器，随着  $n$  的增加，加速比也会增加。

Amdahl 效应示例

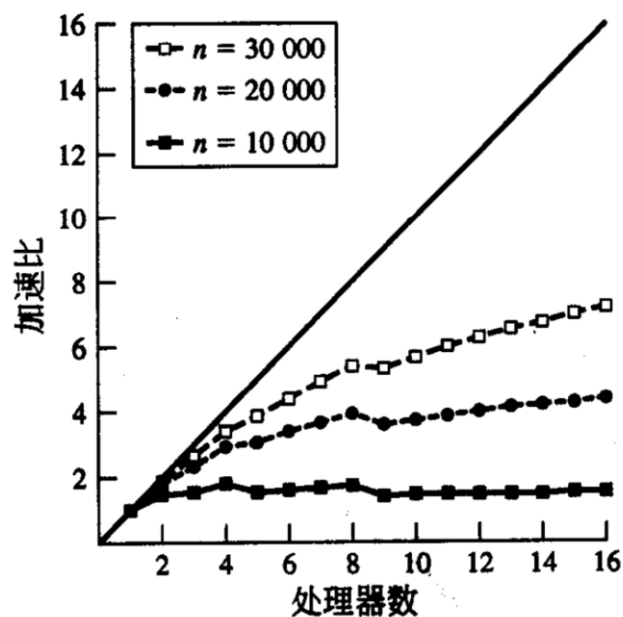
$$y(n, p) \in \frac{S(n) + f(n)}{S(n) + f(n)/p + k(n, p)}$$





# 示例

- 串行部分计算时间:  $(18000+n) \mu\text{sec}$
- 并行部分计算时间:  $(n^2/100) \mu\text{sec}$
- # 通信  $\log n$
- 通信时间:  $10000[\log p] + (n/10) \mu\text{sec}$
- $n=20000, 30000$ 时的加速比曲线?



# 另一个视角

- 以上分析，我们将问题大小视为常数，分析了执行时间是如何随着处理器数量的增加而减少的
- 另一个场景：使用更快的计算机来解决更大的问题实例
- 把时间当作一个常数，允许问题的大小随着处理器数量的增加而增加



# Gustafson-Barsis定律

$$\psi(n, p) \leq \frac{\sigma(n) + \varphi(n)}{\sigma(n) + \varphi(n) / p}$$

$$\text{Let } s = \sigma(n) / (\sigma(n) + \varphi(n) / p)$$

$$\psi \leq p + (1 - p)s$$



# Gustafson-Barsis定律

- 从并行执行时间开始
- 估计解决同一问题的顺序执行时间
- 问题大小是  $p$  的一个递增函数
- 预测比例加速度



# 例 1

- 一个运行在10个处理器上的应用程序在串行代码中花费了3%的时间。该应用程序的比例加速度是多少？

$$\psi = 10 + (1 - 10)(0.03) = 10 - 0.27 = 9.73$$



## 例 2

- 如果一个程序要在8个处理器上达到7的比例加速度，那么并行程序在串行代码中可以花费的最大比例是多少？

$$7 = 8 + (1 - 8)s \Rightarrow s \approx 0.14$$



# 课堂测试

- 一个在32个处理器上执行的并行程序花了1%的时间在串行代码上。这个程序的比例加速度是多少？



# Karp-Flatt指标

- Amdahl定律和Gustafson-Barsis定律忽略了函数 $\kappa(n,p)$ 的影响
- 可能高估加速比或比例加速度
- Karp和Flatt提出了另一个度量指标





# 实验确定的串行比例

$$e = \frac{\sigma(n) + \kappa(n, p)}{\sigma(n) + \varphi(n)}$$

并行计算中固有的串行部分+  
处理器通信和同步开销

---

单处理器执行时间

$$e = \frac{1/\psi - 1/p}{1 - 1/p}$$

实验确定的串行比例



# 实验确定的串行比例

- 考虑了并行开销
- 检测在加速比模型中被忽略的其他开销或低效率来源
  - 进程启动时间
  - 进程同步时间
  - 不平衡的工作负载
  - 架构开销

$$e = \frac{\sigma(n) + \kappa(n, p)}{\sigma(n) + \varphi(n)}$$

$$e = \frac{1/\psi - 1/p}{1 - 1/p}$$



# 例 1

p	2	3	4	5	6	7	8
$\psi$	1.8	2.5	3.1	3.6	4.0	4.4	4.7

在8个CPU上加速比仅为4.7的主要原因是什么？

e	0.1	0.1	0.1	0.1	0.1	0.1	0.1
---	-----	-----	-----	-----	-----	-----	-----

由于常数e是固定的，因此较大的串行部分是导致加速比较低的主要原因。



## 例 2

p	2	3	4	5	6	7	8
$\psi$	1.9	2.6	3.2	3.7	4.1	4.5	4.7

在8个CPU上加速比仅为4.7的主要原因是什么？

e	0.070	0.075	0.080	0.085	0.090	0.095	0.100
---	-------	-------	-------	-------	-------	-------	-------

由于常数e不断增加，因此并行开销是导致加速比较低的主要原因



# 课堂测试

p	4	8	12
$\psi$	3.9	6.5	?

这个程序在 12 个处理器上是否有可能达到 10 倍加速比？



# 等效率指标

- 并行系统的可扩展性：衡量其随着处理器数量增加而提高性能的能力
- 可扩展的系统可以保持效率，即在增加处理器时仍然能够保持高效率
- **等效率性**：衡量可扩展性的一种方式



# 等效性的推导步骤

- 从加速比公式开始
- 计算总的并行开销
- 假设效率保持不变
- 确定串行执行时间和并行开销之间的关系



# 推导等效性关系

- 确定并行开销

$$T_o(n, p) = (p - 1)\sigma(n) + p\kappa(n, p)$$

- 将并行开销代入加速比公式

$$\psi(n, p) \leq \frac{p(\sigma(n) + \varphi(n))}{\sigma(n) + \varphi(n) + T_o(n, p)}$$

- 将 $T(n, 1) = \sigma(n) + \varphi(n)$ 代入公式, 假设效率保持不变。

$$T(n, 1) \geq CT_0(n, p) \quad \text{等效性关系}$$





# 可扩展性函数

- 假设等效关系为  $n \geq f(p)$
- 令  $M(n)$  问题规模为  $n$  时所需的内存
- $M(f(p))/p$  表明每个处理器的内存用量必须增加才能保持相同的效率。
- 称  $M(f(p))/p$  为可扩展性函数。

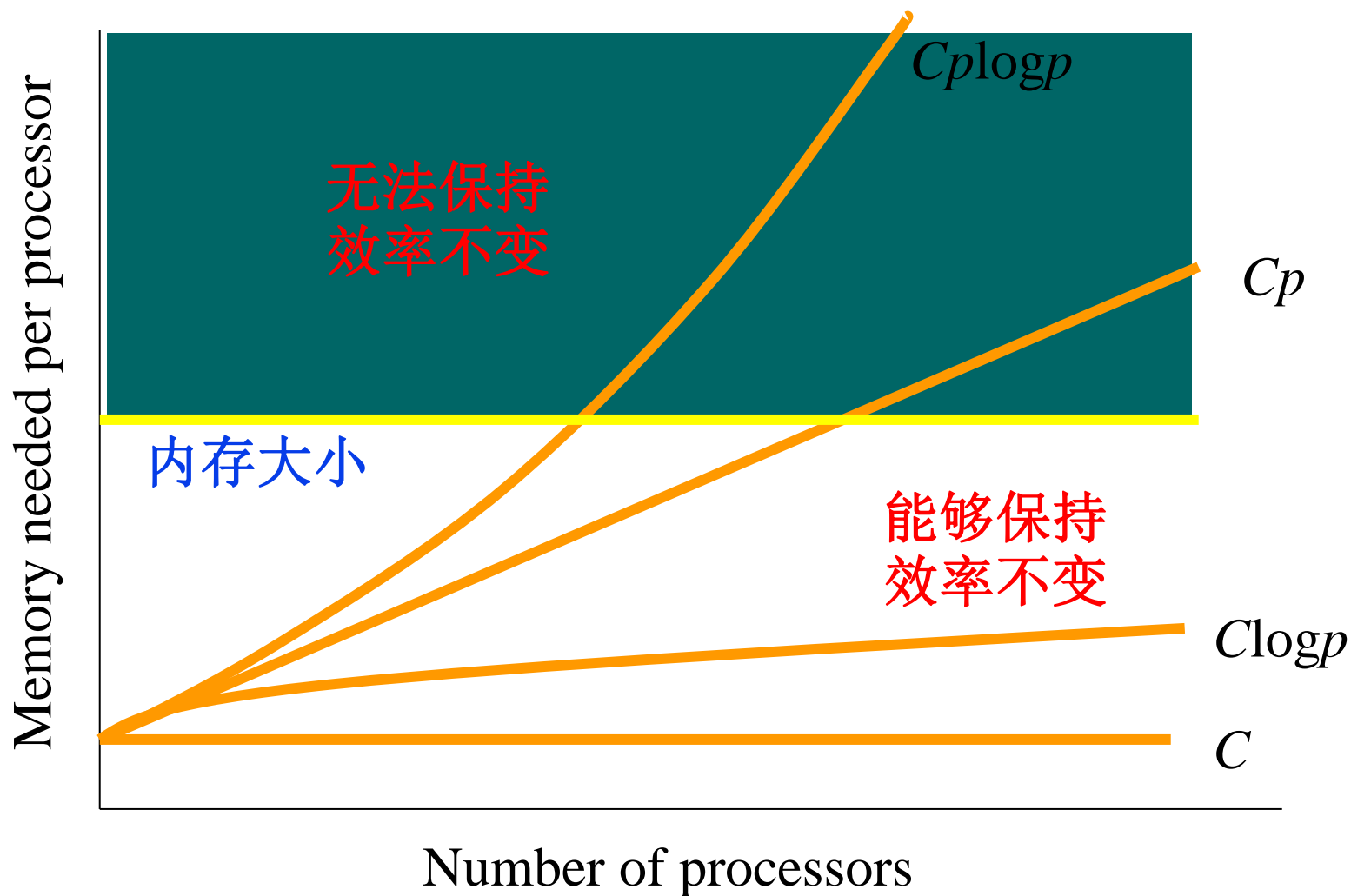


# 可扩展性函数意义

- 为了在增加 $p$ 时保持效率不变，我们必须增加 $n$
- 最大的问题规模受限于可用内存，它与 $p$ 是线性关系
- 可扩展性函数显示了每个处理器的内存使用量必须如何增长以保持效率不变
- 可扩展性函数为常数意味着并行系统是完美可扩展的



# 可扩展性函解释



# 例 1：规约 (Reduction)

- 串行算法复杂度  $T(n,1) = \Theta(n)$
- 并行算法
  - 计算复杂度 =  $\Theta(n/p)$
  - 通信复杂度 =  $\Theta(\log p)$
- 并行开销  $T_o(n,p) = \Theta(p \log p)$



# 例 1：规约 (Reduction)

- 等效性关系：  $n \geq C p \log p$
- 为了保持相同的效率水平，当  $p$  增加时，  $n$  必须如何增加？
- $M(n) = n$
- 系统具有良好的可扩展性

$$M(Cp \log p) / p = Cp \log p / p = C \log p$$



## 例 2 : Floyd算法

- 串行时间复杂度 :  $\Theta(n^3)$
- 并行计算时间:  $\Theta(n^3/p)$
- 并行通信时间:  $\Theta(n^2 \log p)$
- 并行开销:  $T_o(n,p) = \Theta(pn^2 \log p)$



## 例 2: Floyd算法

- 等效性关系:  $n^3 \geq C(p n^3 \log p) \Rightarrow n \geq C p \log p$
- $M(n) = n^2$
- 并行系统可扩展性较差

$$M(Cp \log p) / p = C^2 p^2 \log^2 p / p = C^2 p \log^2 p$$



# 例 3：有限差分

- 每轮迭代串行时间复杂度： $\Theta(n^2)$
- 每轮迭代并行通信时间复杂度： $\Theta(n/\sqrt{p})$
- 并行开销： $\Theta(n \sqrt{p})$



# 例 3：有限差分

- 等效性关系：  $n^2 \geq Cn\sqrt{p} \Rightarrow n \geq C\sqrt{p}$
- $M(n) = n^2$
- 这个算法是完美可扩展的。

$$M(C\sqrt{p}) / p = C^2 p / p = C^2$$



# 总结

- 性能术语
  - 加速比
  - 效率
- 加速比模型
  - 串行部分
  - 并行部分
  - 通信部分



# 总结

- 线性加速比无法实现的原因
  - 串行操作
  - 通信操作
  - 进程启动
  - 不平衡的工作负载
  - 架构限制



# 总结

- 并行性能分析
  - Amdahl's 定律
  - Gustafson-Barsis' 定律
  - Karp-Flatt 指标
  - Isoefficiency 指标

