

# 并行与分布式计算基础：第四讲

杨超

chao\_yang@pku.edu.cn

2019 秋



# 内容提纲

- ① 上次课程回顾
- ② 并行计算三大定律
- ③ 共享内存并行计算模型 (1)

# 上次课程回顾

- ① 上次课程回顾
- ② 并行计算三大定律
- ③ 共享内存并行计算模型 (1)

# 课程基本情况

- 课程名称：并行与分布式计算基础
- 授课教师：杨超 (chao\_yang@pku.edu.cn, 理科 1 号楼 1520)
- 课程助教：尹鹏飞 (pengfeiyin@pku.edu.cn)

## 授课内容 (暂定)

- 引言
- 硬件体系架构
- 并行计算模型
- 编程与开发环境
- MPI 编程与实践
- OpenMP 编程与实践
- GPU 编程与实践
- 前沿问题选讲

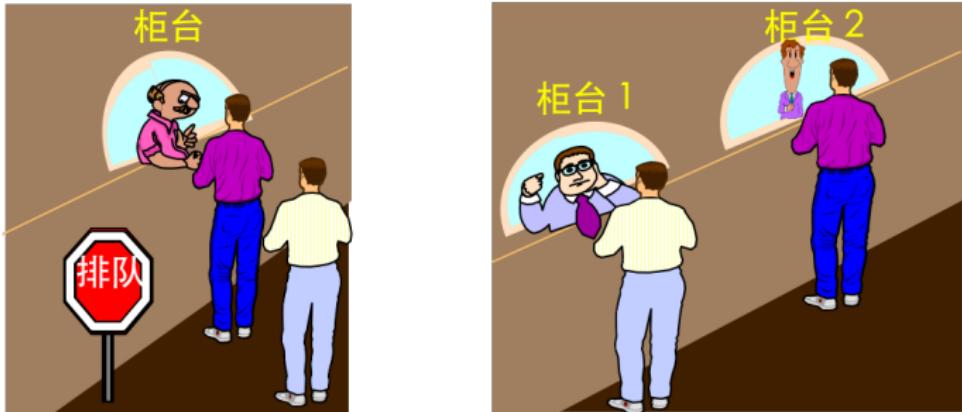
# 上课时间 (地点: 二教 211)

上课时间	星期一	星期二	星期三	星期四	星期五
第 1 节 (8:00-8:50)					
第 2 节 (9:00-9:50)					
第 3 节 (10:10-11:00)				单周	
第 4 节 (11:10-12:00)				单周	
第 5 节 (13:00-13:50)		每周			
第 6 节 (14:00-14:50)		每周			
第 7 节 (15:10-16:00)					
第 8 节 (16:10-17:00)					
第 9 节 (17:10-18:00)					
第 10 节 (18:40-19:30)					
第 11 节 (19:40-20:30)					
第 12 节 (20:40-21:30)					

# 计算、串行计算、并行计算

计算已成为科技创新的第三大手段，已在各行各业中大显身手：

- 串行计算将问题被分为一系列独立指令，按照先后顺序逐一执行；
- 并行计算则将问题分为能并发执行的若干部分，分别串行执行。



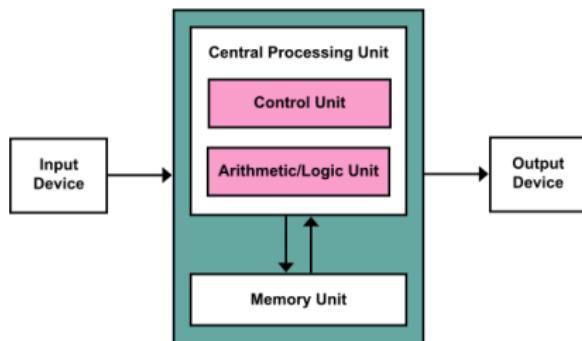
并行计算的主要目的：加速求解问题的速度、提高求解问题的规模等。

# 当代计算机的设计蓝图

- 存储程序计算机：可重新编程，比固定程序计算机更通用。
- Princeton 架构：将指令（即程序）与数据共同存储在内存。  
又称冯诺依曼架构，是现代计算机体系架构的基础。

## 冯诺依曼体系架构

- 控制单元：解释指令；
- 处理单元：执行指令；
- 内存：存储数据和指令；
- 输入/输出：与外界交互。



## 主要缺陷

- 仅具有单一的线性内存，指令与数据仅在使用时才隐式区分；
- 总性能往往受限于内存的读写总线所能提供的延迟和带宽。

# 提高处理器性能的主要手段

## 改善指令和存储机制

- 精简指令集、多级存储。

## 指令级并行 (Instruction Level Parallelism, ILP)

- 超标量、流水线、乱序执行。

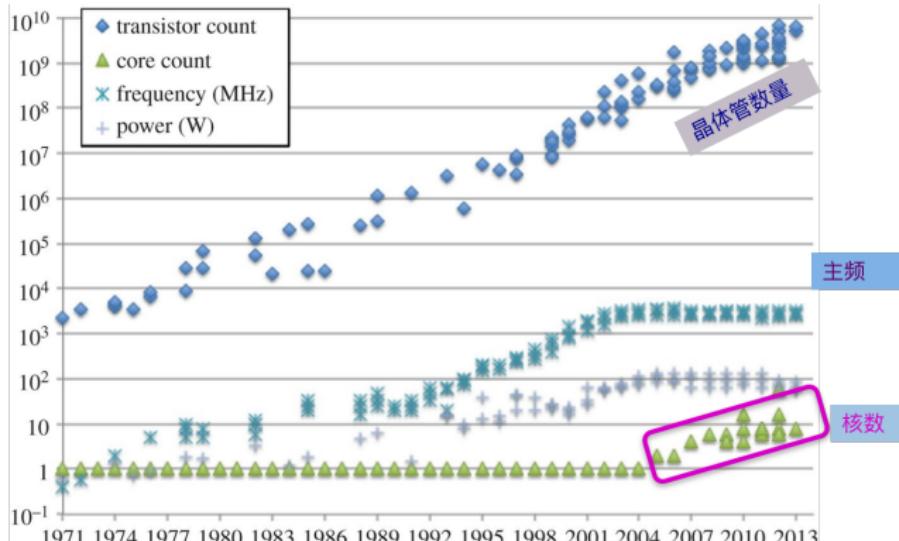
## 数据级并行 (Data Level Parallelism, DLP)

- 向量化 (如：乘加指令)。

## 任务级并行 (Task Level Parallelism, TLP)

- 多核/众核，超线程。

# 功耗墙 (Energy Wall) 与多核/众核



Courtesy: Giles & Reguly, 2014

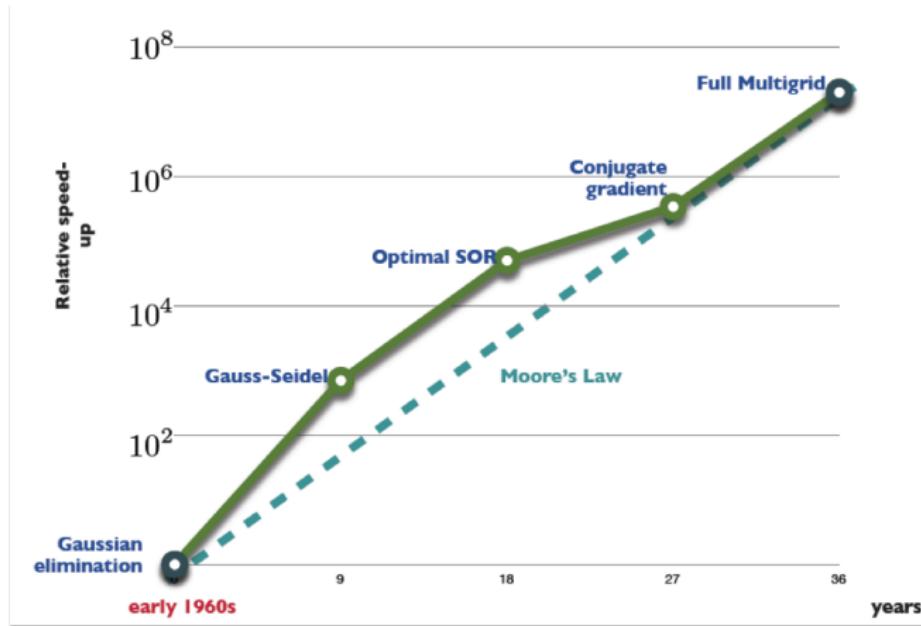
提高核数是维持性能提高并控制功耗增长的有效途径：  
单核 → 双核 → 多核 → 众核 → ...

# 硬件发展趋势



# 算法的发展规律

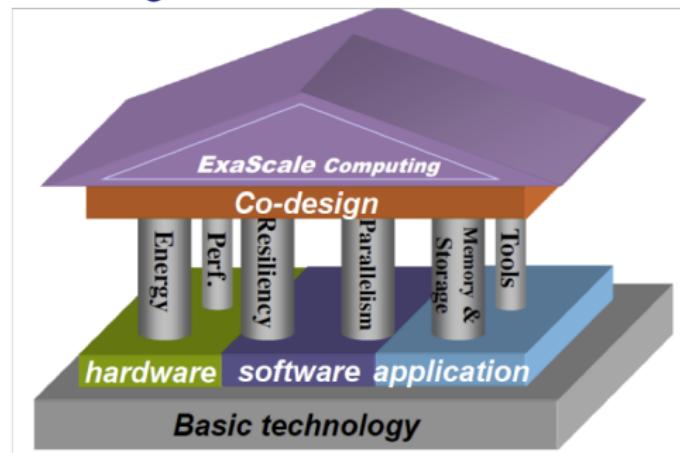
算法的贡献也遵循类似摩尔定律的发展规律！



Algorithms can often speed up science as much as the Moore's law.  
——摘自2005年美国总统报告, PITAC.

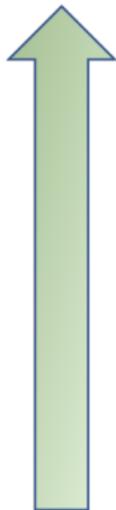
# 并行算法与软件的研究价值

- 算法和软件是应用和计算机之间的桥梁
- 算法是软件的灵魂，不同类型的应用所需的算法可能不同
- 不同的算法各自适用于不同的计算机
  - ❖ 比如：传统的FFT算法在向量机上很好，但在分布式系统上不够理想
- 需要多方面专家协同设计（Co-design）



# 并行计算研究的几个主要视角

应用



- 应用视角：数学建模，算法设计与分析等。
- 算法视角：算法的并行化以及相关的主要原则等。
- 编程视角：采用何种方式编程实现并行算法。
- 性能视角：通过建立并行计算模型，指导算法设计、编程实现、性能优化等。
- 硬件视角：并行计算机体系架构。

硬件

课程从应用与硬件切入，通过讨论算法与性能，最终聚焦并行编程。

# 并行计算三大定律

① 上次课程回顾

② 并行计算三大定律

③ 共享内存并行计算模型 (1)

# 一些约定/假定

- 任务 (task): 并行计算所处理的对象。
- 工作量 (workload): 处理某任务的所需的各种开销的总和。
- 处理器 (processor): 并行计算中所使用的最基本的处理器单元。
- 执行率 (execution rate): 每个处理器单位时间能完成的工作量<sup>1</sup>。
- 执行时间 (execution time): 处理某任务所需的时间。
- 加速比 (scalability): 当处理器个数增多时，完成某固定工作量任务所需执行时间的减少倍数。
- 理想加速比 (ideal scalability): 处理器个数增多的比例。
- 并行效率 (parallel efficiency): 加速比  $\div$  理想加速比  $\times 100\%$ .

注 1: 这里假定每个处理器的执行率相同。

# 阿姆达尔定律

## 阿姆达尔定律 (Amdahl's Law, 1967)

记  $\alpha \in [0, 1]$  是某任务无法并行处理部分所占的比例. 假设该任务的工作量固定, 则对任意  $n$  个处理器, 相比于 1 个处理器, 能够取得的加速比满足:  $S(n) < 1/\alpha$ .

**证明:** 设每个处理器的执行率是  $R$ , 处理该任务的总工作量是  $W$ . 记  $T(1)$  和  $T(n)$  分别为使用 1 个和  $n$  个处理器处理该任务所需要的时间, 则有

$$T(1) = \frac{W}{R}, \quad T(n) = \frac{\alpha W}{R} + \frac{(1 - \alpha)W}{nR}.$$

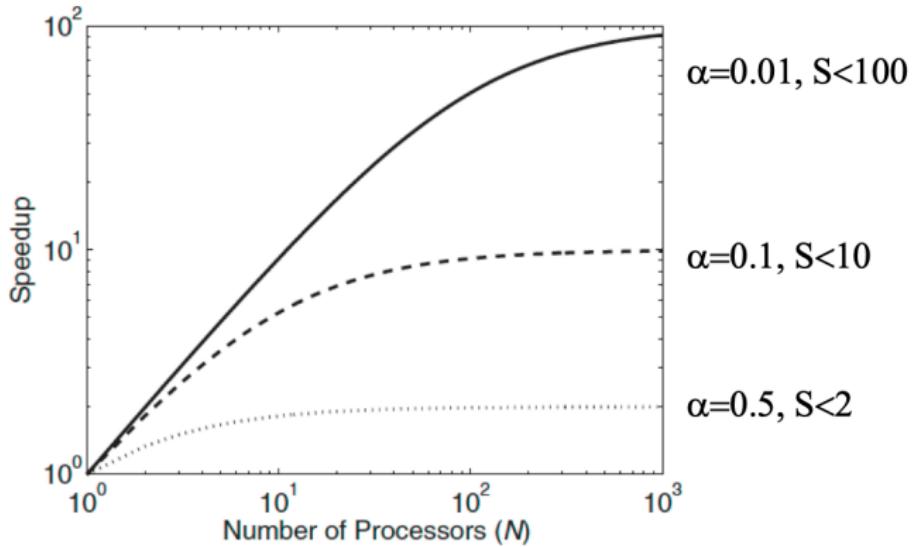
据此, 可计算出使用  $n$  个处理器相比于 1 个处理器的加速比

$$S(n) = \frac{T(1)}{T(n)} = \frac{1}{\alpha + \frac{1-\alpha}{n}} < \frac{1}{\alpha}.$$

# 阿姆达尔定律的推论



Gene M. Amdahl  
(1922-2015)



484 Spring Joint Computer Conf., 1967

by DR. GENE M. AMDAHL  
International Business Machines Corporation  
Sunnyvale, California

Validity of the single processor approach to achieving large scale computing capabilities

# 卡普挑战

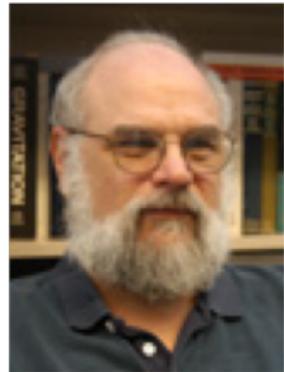


NCUBE 10 (1,024 processors)

## Background:

In 1983, there was a new tech trend of **massively parallel computing** with over 1,000 processors.

People started to wonder how to use them efficiently.



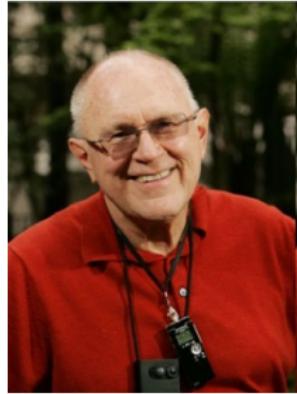
Alan H. Karp

## 卡普挑战 (Karp Challenge, 1985)

打赌 100 美元没人能在十年内为三个实际应用实现 200 倍的并行加速。

- 结果：两年之内，没有胜者（大多是模型问题，加速比最多数十）。

# 戈登贝尔奖



Gordon Bell  
(1934-)

In 1987, Alan Karp was approached by Gordon Bell, who was a founding assistant director of the **CISE Directorate at NSF**.

Bell persuaded Karp to replace the **Karp Challenge** with the **Gordon Bell Prize**.

The Gordon Bell Prize **does not require a speedup of 200X** or any particular number but emphasizes on **technical innovations in HPC applications**.

## 戈登贝尔奖 (Gordon Bell Prize, 1987)

每年奖励 1000 美元给在高性能计算应用取得杰出成就的团队。

- 2006 年起由 ACM 负责，2011 年起奖金提升至 1 万美元。

# 首届戈登贝尔奖



John L. Gustafson  
(1955-)

In 1987 the first Gordon Bell Prize was awarded to **Robert Benner, John Gustafson and Gary Montry** from Sandia National Laboratories.

Achieved **400~600** speedup on NCUBE 10 in 3 applications:

- \* Beam Stress Analysis
- \* Surface Wave Simulation
- \* Unstable Fluid Flow Modeling

In fact, they also won the Karp Challenge!

## REEVALUATING AMDAHL'S LAW

JOHN L. GUSTAFSON

# 古斯塔法森定律

## 古斯塔法森定律 (Gustafson's Law, 1988)

记  $\alpha \in [0, 1]$  是某任务无法并行处理部分所占的比例. 假设该任务的工作量可以随着处理器个数缩放, 从而保持处理时间固定. 则对任意  $n$  个处理器, 相比于 1 个处理器, 能够取得的加速比  $S'(n)$  不存在上界.

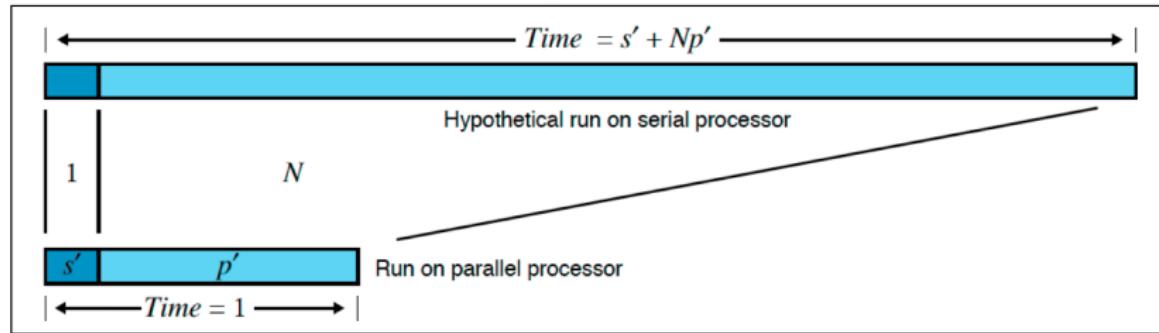
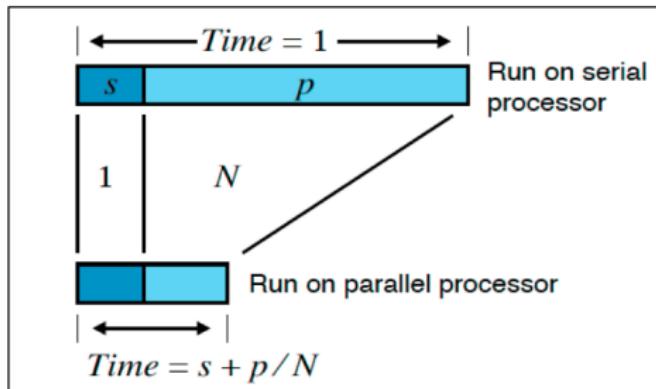
证明: 设每个处理器的执行率是  $R$ , 单处理器情况下该任务的基准工作量是  $W$ . 在处理时间固定的情况下, 可以得知采用  $n$  个处理器时该任务的缩放工作量  $W'$  应为

$$W' = \alpha W + (1 - \alpha)nW.$$

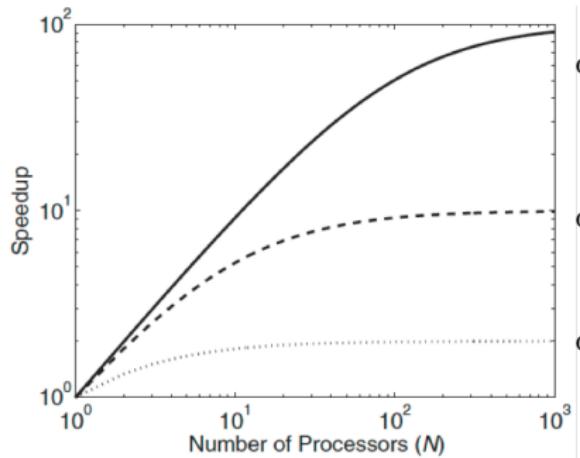
据此, 可计算出使用  $n$  个处理器相比于 1 个处理器的加速比

$$S'(n) = \frac{\frac{W'}{R}}{\frac{W}{R}} = \alpha + (1 - \alpha)n.$$

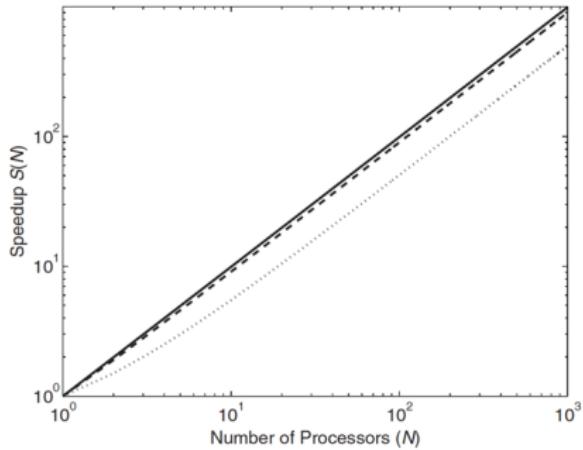
# 从阿姆达尔加速比到古斯塔法森加速比



# 从阿姆达尔加速比到古斯塔法森加速比



$\alpha=0.01$   
 $\alpha=0.1$   
 $\alpha=0.5$



# 孙-倪定律

## 孙-倪定律 (Sun-Ni's Law, 1990)

记  $\alpha \in [0, 1]$  是某任务无法并行处理部分所占的比例. 假设该任务的可并行部分随着处理器个数  $n$  按照因子  $G(n)$  缩放, 则对任意  $n$ , 相比于 1 个处理器, 能够取得的加速比  $S^*(n)$  满足

$$S^*(n) = \frac{\alpha + (1 - \alpha)G(n)}{\alpha + (1 - \alpha)\frac{G(n)}{n}}.$$

证明: 设每个处理器的执行率是  $R$ , 单处理器情况下该任务的基准工作量是  $W$ . 使用  $n$  个处理器时该任务的缩放工作量为

$$W^* = \alpha W + (1 - \alpha)G(n)W.$$

据此, 可计算出使用  $n$  个处理器相比于 1 个处理器的加速比

$$S^*(n) = \frac{\alpha W \frac{1}{R} + (1 - \alpha)G(n)W \frac{1}{R}}{\alpha W \frac{1}{R} + (1 - \alpha)G(n)W \frac{1}{nR}} = \frac{\alpha + (1 - \alpha)G(n)}{\alpha + (1 - \alpha)\frac{G(n)}{n}}.$$

# 三种模型的关系

## 加速比分析

$$S^*(n) = \frac{\alpha + (1 - \alpha)G(n)}{\alpha + (1 - \alpha)\frac{G(n)}{n}} \begin{cases} = \frac{1}{\alpha + \frac{1-\alpha}{n}} & \text{if } G(n) = 1, \\ = \alpha + (1 - \alpha)n & \text{if } G(n) = n, \\ > \alpha + (1 - \alpha)n & \text{if } G(n) > n. \end{cases}$$

	加速比 ( $n \rightarrow \infty$ )	并行效率 ( $n \rightarrow \infty$ )
阿姆达尔	$S(n) \rightarrow \frac{1}{\alpha}$	$E(n) \rightarrow 0$
古斯塔法森	$S'(n) \rightarrow \infty$	$E'(n) \rightarrow 1 - \alpha$
孙-倪 ( $G(n) > O(n)$ )	$S^*(n) \rightarrow \infty$	$E^*(n) \rightarrow 1$

## 应用举例

例：矩阵乘  $C = AB$ , 这里  $A, B, C$  都是  $N \times N$  阶矩阵.

- 计算复杂度:  $\mathcal{C}(N) = 2N^3$ .
- 存储复杂度:  $\mathcal{S}(N) = 3N^2$ .

由  $\mathcal{S}(N) = x$  可得  $\mathcal{S}^{-1}(x) = \left(\frac{x}{3}\right)^{\frac{1}{2}}$ , 因此

$$G(n) = \frac{\mathcal{C}(\mathcal{S}^{-1}(nx))}{\mathcal{C}(\mathcal{S}^{-1}(x))} = \frac{2\left(\frac{nx}{3}\right)^{\frac{1}{2} \cdot 3}}{2\left(\frac{x}{3}\right)^{\frac{1}{2} \cdot 3}} = n^{\frac{3}{2}},$$

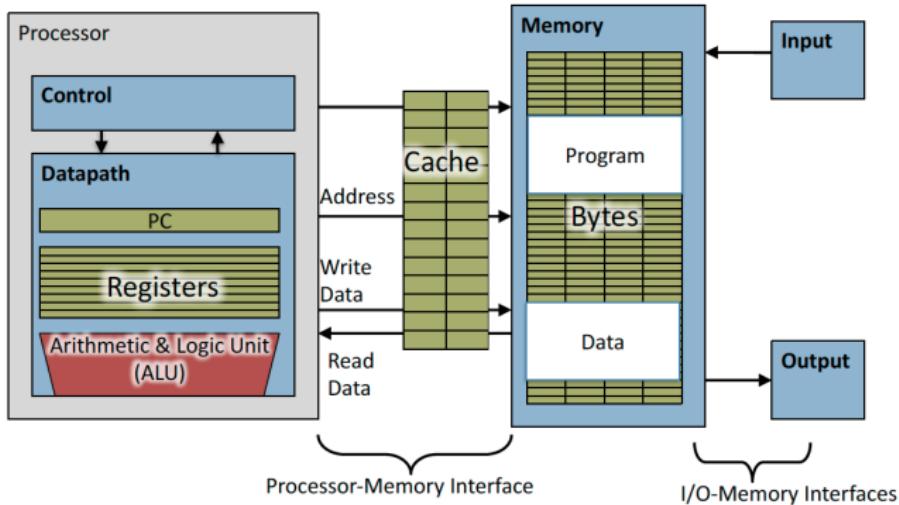
从而

$$S^*(n) = \frac{\alpha + (1 - \alpha)G(n)}{\alpha + (1 - \alpha)\frac{G(n)}{n}} = \frac{\alpha + (1 - \alpha)n^{\frac{3}{2}}}{\alpha + (1 - \alpha)n^{\frac{1}{2}}}.$$

# 共享内存并行计算模型 (1)

- ① 上次课程回顾
- ② 并行计算三大定律
- ③ 共享内存并行计算模型 (1)

# 影响性能的主要因素是什么？



## 主要因素分析

- 冯诺伊曼体系架构 → 计算、访存之间的关系。
- 多级存储 → 多级数据访问。
- 多核/众核 → 并行计算/访存。

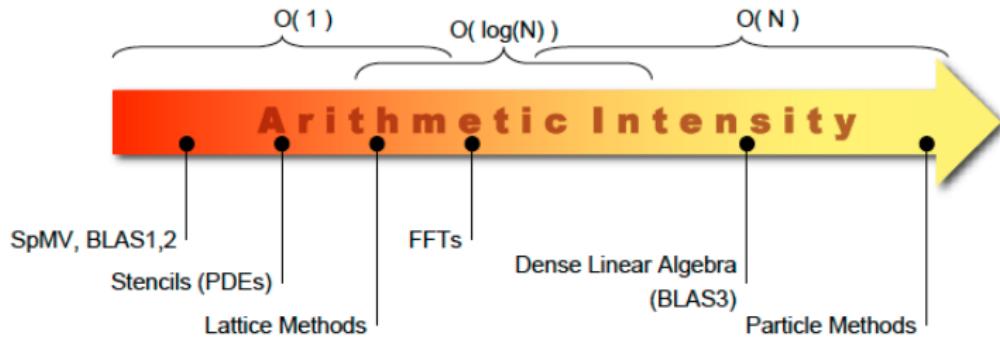
# 从计算、访存之间的关系考虑

基本思想

$$\frac{flop}{second} = \min\{\text{peak\_flops}, \frac{byte}{second} * \frac{flop}{byte}\}$$

性能              峰值              带宽      计算密度

因此，引入计算密度 (Arithmetic Intensity) :



# Roofline 模型 (2009)

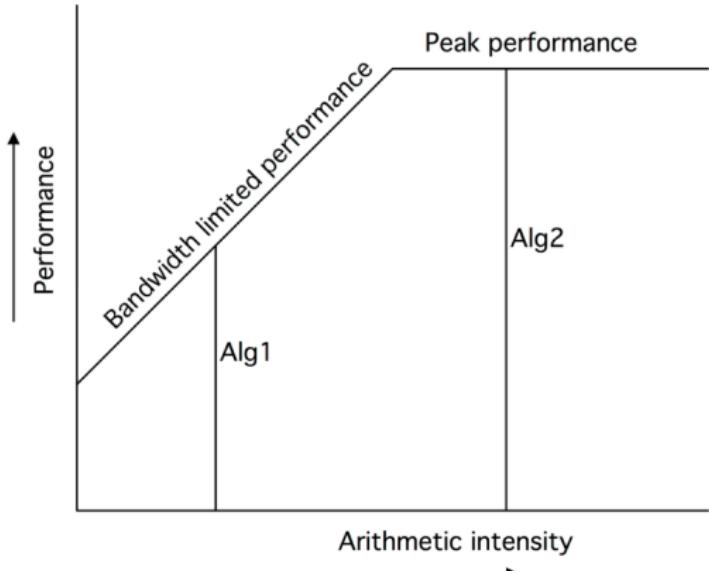
DOI:10.1145/1498765.1498785

The Roofline model offers insight on how to improve the performance of software and hardware.

BY SAMUEL WILLIAMS, ANDREW WATERMAN, AND DAVID PATTERSON

# Roofline: An Insightful Visual Performance Model for Multicore Architectures

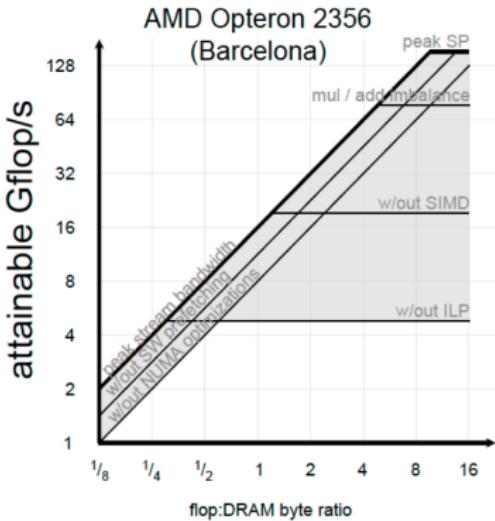
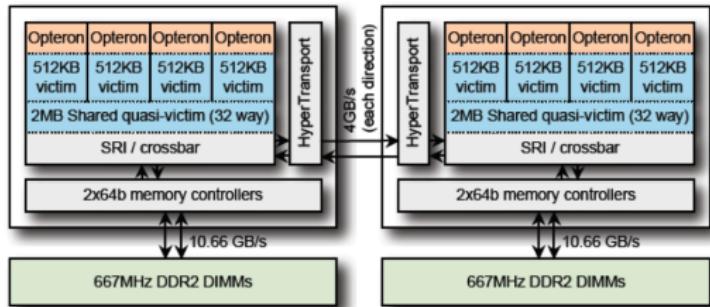
APRIL 2009 | VOL. 52 | NO. 4 | COMMUNICATIONS OF THE ACM 65



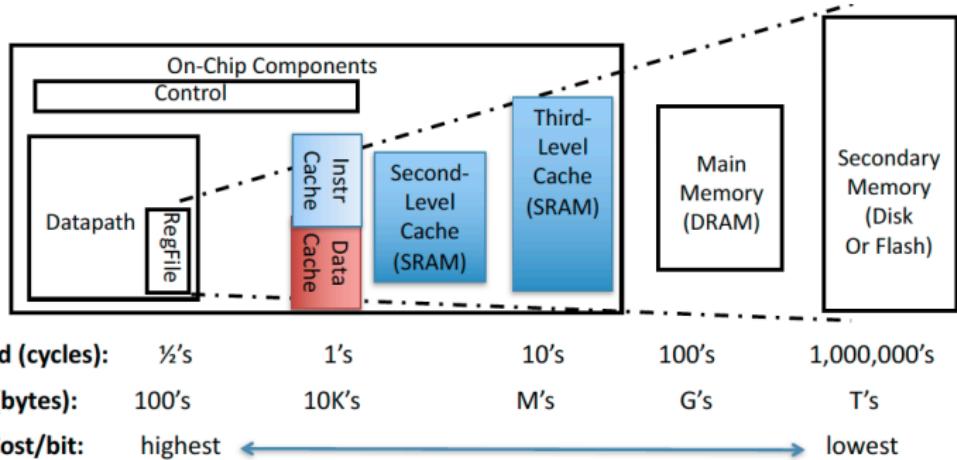
# 受约束的 Roofline 模型

举例

## AMD Opteron 2356 (Barcelona)



# 从多级存储的角度考虑



重点考虑缓存命中/失效 (hit/miss) 对时间的影响：

- 命中：数据在该级存储中找到；
- 不命中：需要在下一级存储中寻找。
- 需要引入命中率/失效率 (hit/miss rate) 的概念。

# 单层 AMAT 模型

## 单层 AMAT (Average Memory Access Time) 模型

假设只有一级缓存，AMAT 模型预测的平均访存时间为：

$$\begin{aligned} \text{AMAT} &= (1 - r)T_{\$} + r(T_{\$} + T_M) \\ &= T_{\$} + rT_M, \end{aligned}$$

其中  $T_{\$}$  为缓存访问时间 (也叫命中时间, hit time)， $T_M$  为内存访问时间 (也叫缓存失效损失, miss penalty)， $r$  为缓存失效率 (miss rate)。

假如某平台 CPU 主频为 1GHz，即 CPU 时钟周期是 1ns，且

- 缓存访问开销为 2 拍 (cycle)，即  $T_{\$} = 2\text{ns}$ ；
- 缓存失效损失为 300 拍 (cycle)，即  $T_m = 300\text{ns}$ ；
- 缓存命中率为 90%，即  $r = 0.1$ ；则

$$\text{AMAT} = 2\text{ns} + 0.1 * 300\text{ns} = 32\text{ns}.$$

# 多层 AMAT 模型

## 多层 AMAT (Average Memory Access Time) 模型

假设有两级/三级缓存， $T_1, T_2, T_3$  分别为 L1, L2, L3 缓存访问时间， $T_M$  为内存访问时间， $r_1, r_2, r_3$  分别为 L1, L2, L3 缓存的局部失效率 (local miss rate)，则两层/三层 AMAT 模型预测的平均访存时间为：

$$\begin{aligned}\text{AMAT}_2 &= T_1 + r_1(T_2 + r_2T_M) \\ &= T_1 + R_1T_2 + R_2T_M,\end{aligned}$$

$$\begin{aligned}\text{AMAT}_3 &= T_1 + r_1 [T_2 + r_2(T_3 + r_3T_M)] \\ &= T_1 + R_1T_2 + R_2T_3 + R_3T_M.\end{aligned}$$

其中  $R_1 = r_1$ ,  $R_2 = r_1r_2$ ,  $R_3 = r_1r_2r_3$  分别为 L1, L2, L3 缓存的整体失效率 (global miss rate)。

注：局部失效率指该层次缓存失效的概率，整体失效率表示该层次缓存以及其上层所有缓存同时失效的概率。

# AMAT 模型应用举例

某平台的多级缓存信息：

	L1 缓存	L2 缓存	L3 缓存	内存
访问时间 (ns)	$T_1 = 2$	$T_2 = 10$	$T_3 = 50$	$T_M = 400$
局部失效率	$r_1 = 0.1$	$r_2 = 0.5$	$r_3 = 0.4$	0
整体失效率	$R_1 = 0.1$	$R_2 = 0.05$	$R_3 = 0.02$	0

如果只有 L1 缓存：

$$AMAT_1 = 2\text{ns} + 0.1 * 400\text{ns} = 42\text{ns}.$$

如果增加 L2 缓存：

$$AMAT_2 = 2\text{ns} + 0.1 * 10\text{ns} + 0.05 * 400\text{ns} = 23\text{ns}.$$

如果再增加 L3 缓存：

$$AMAT_3 = 2\text{ns} + 0.1 * 10\text{ns} + 0.05 * 50\text{ns} + 0.02 * 400\text{ns} = 13.5\text{ns}.$$