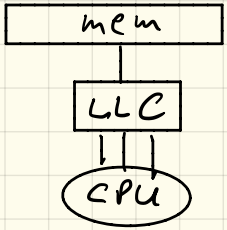
Design of Parallel & High Performance Computing

Reasoning about Performance II

Roofline II - Balance Principles - Analysis Scheduling

Roofline Model (Williams et al. 2008)

Computer:



bandwidth β [bytes/cycle]

peak perf. π [flops/cycle]

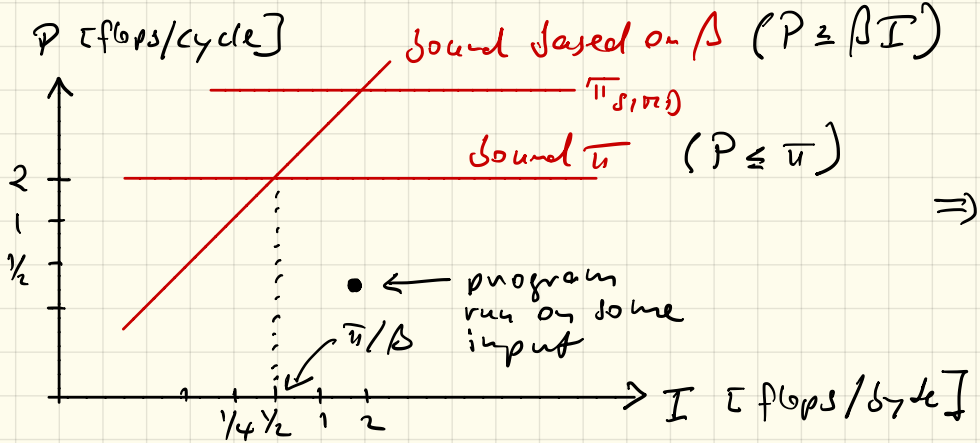
Program:

$I = W/Q$ [flops/byte]

T = runtime [cycles]

$P = W/T$ (performance) [flops/cycle]

Roofline plot: (example $\pi = 2$, $\beta = 4$)



reason: $\frac{P}{I} = \frac{W/T}{W/Q} = \frac{Q}{T} \leq \beta$

$P \leq \beta \cdot I$

$\Rightarrow \log P \leq \log I + \log \beta$

$I = 1 \Rightarrow P \leq \beta$

Operational Intensity: Upper bounds

$I(n) = W(n) / Q(n)$ — #flops / #bytes mem ↔ cache

x, y: length n, A, B, C: $n \times n$, α: scalar

		asymptotic	best	exact
daxpy	$y = \alpha x + y$	$O(1)$	$O(1)$	$\leq 1/12$
dgemv	$y = Ax + y$	$O(1)$	$O(1)$	$\leq 1/4$
fft		$O(\log n)$	$\Theta(\log Z)$	——
dgemm	$C = AB + C$	$O(n)$	$\Theta(\sqrt{Z})$	$\leq \frac{n+1}{16}$

Z = cache size

should be sharp if $3n^2 \cdot 8 \leq Z$

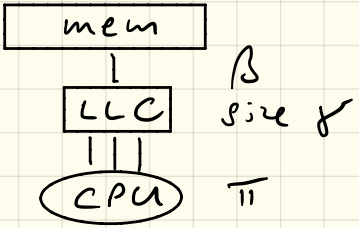
Example: $Z = 128\,\text{kB}$

$\Rightarrow n \leq 700$

Cache to slides → measured roofline plots

6. Balance Principles I (Kung 86)

Computer:



Program:

Q_γ: data transfer mem ↔ LLC

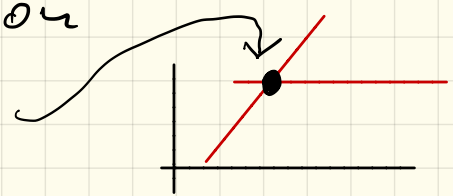
W: work = # ops

The computer is called **balanced** if

compute time = data transfer time

i.e., assuming perfect utilization

$$\frac{W}{\pi} = \frac{Q_\gamma}{\beta} \iff \frac{\pi}{\beta} = \frac{W}{Q_\gamma}$$



assume $\pi \to a\pi$ increases; how to rebalance?

a.) $\beta \to a \cdot \beta$ ✓ but β grows slower than π!

b.) $\gamma \to x \cdot \gamma$ what is x?

Example 1: Matrix multiplication $C = AB + C$

algorithm with optimal $I = \frac{W}{Q_\gamma} = \Theta(\sqrt{\gamma})$

$$\frac{\pi}{\beta} = \frac{W}{Q_\gamma} = \Theta(\sqrt{\gamma}) \qquad \pi \to a\pi, \ \gamma \to a^2 \cdot \gamma$$

Example 2: FFT / Sorting

algorithm with optimal $I = \Theta(\log \gamma)$

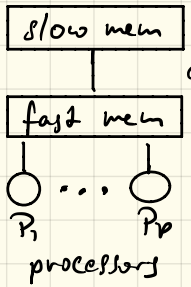
$$\frac{\pi}{\beta} = \frac{W}{Q_\gamma} = \Theta(\log \gamma) \qquad \pi \to a\pi, \ \gamma \to \gamma^a$$

Both are unrealistic.

7. Balance principles II (Czechowski et al. 2011)

Goal: more detailed principles for multicores, algorithm/architecture codesign, assessment of HW trends

Computer:



α: latency, β: throughput, size γ, transaction size λ

each peak perf π

Algorithm:

PRAM: W = work, D: depth

$Q_{\gamma,\lambda}$: mem. transfers of size λ

assumptions: optimal W, W/Q

How to get $Q_{p,\gamma,\lambda}$ from $Q_{\gamma,\lambda}$?

- there are general bounds and some direct results

Processor is balanced if

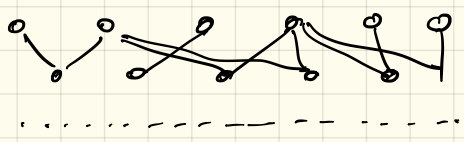
$T_{mem} \leq T_{comp}$ ("compute-bound")

Hong:

$$\frac{Q \cdot \lambda}{\beta} \leq \frac{W}{p \cdot \pi} \iff \frac{p\pi}{\beta} \leq \frac{W}{Q\lambda}$$

Derivation principles:

1.) Estimate T_{mem}: Idea divide DAG in levels



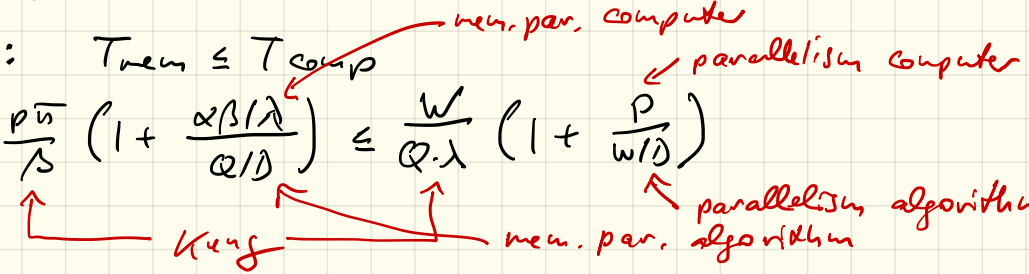
In each level α-β model: $\alpha + \frac{q_i \cdot \lambda}{\beta}$

$$\Rightarrow T_{mem} \approx \sum_{i=1}^{D} \left(\alpha + \frac{q_i \cdot \lambda}{\beta}\right) = \alpha \cdot D + \frac{Q \cdot \lambda}{\beta}, \qquad Q = Q_{P,\gamma,\lambda}$$

2.) Estimate T_{comp}: Brent's lemma

$$T_{comp} \approx \left(D + \frac{W}{P}\right) \cdot \frac{1}{\pi}$$

Balance: $T_{mem} \leq T_{comp}$



$$\Leftrightarrow \frac{P\pi}{\beta}\left(1 + \frac{\alpha\beta I \lambda}{Q I D}\right) \leq \frac{W}{Q \cdot \lambda}\left(1 + \frac{P}{W I D}\right)$$

Example 1: matrix multiplication

use $Q \geq \frac{W}{\sqrt{\varepsilon} \cdot \lambda \sqrt{\gamma p}}$ (Irony et al. 2004) $\Rightarrow I(n) = O\left(\sqrt{\frac{\gamma}{p}}\right)$

and $\eta \ll W$, $\eta \ll Q$ ($\ll$ means much smaller)

$\Rightarrow$ balance principle $\frac{p\eta}{\beta} \leq O\left(\sqrt{\frac{\gamma}{p}}\right)$

$\eta \to a \cdot \eta$ rescales: $\beta \to a\beta$ or $\gamma \to a^2\gamma$

$p \to a \cdot p$ " : ($\beta \to a\beta$ and $\gamma \to a\gamma$) or ($\gamma \to a^3\gamma$)

Example 2: sorting/FFT

$\Rightarrow \frac{p\eta}{\beta} \leq O\left(\log \frac{\gamma}{p}\right)$

Back to Schrödinger $\to$ Evolution of balance

Greedy scheduling: Analysis

Reminder: $T_1 = W$, $T_\infty = D$ $\quad T_p \geq T_1/p$, T_∞ $\quad T_p \leq D + W/p = T_\infty + T_1/p$

Greedy scheduler:

complete steps $\leq T_1/p$
incomplete steps $\leq T_\infty$ (every incomplete step reduces critical path by 1)

$\Rightarrow T_p \leq T_1/p + T_\infty$

How far from optimal?

Assume T_p^* is optimal:

$T_p^* \geq \max\{T_1/p, T_\infty\}$

$T_p \leq T_1/p + T_\infty \leq 2\max\{T_1/p, T_\infty\} \leq 2T_p^*$ — go at most factor 2 away

enough parallelism: $W/D = T_1/T_\infty \gg p \iff T_\infty \ll T_1/p$

$\Rightarrow T_p \leq T_1/p + T_\infty \approx T_1/p$

Work stealing scheduling: Analysis

Theorem: The scheduler achieves $T_p \leq T_1/p + O(T_\infty)$

Proof sketch: Every processor is either working or stealing

Total time working: T_1

Total time stealing: Every steal has $1/p$ chance to reduce the critical path; hence $O(pT_\infty)$

$\Rightarrow$ total time: $T_1 + O(pT_\infty)$

$\Rightarrow$ time per processor: total time $/p = T_1/p + O(T_\infty)$

Note: space requirement $S_p \leq pS_1$.