Little's Law (1961)

Example 1: In a Starbucks, <u>on average</u>

- every minute 2 customers enter and leave
- every customer spends 8 minutes in the store

How many people are inside? $16 = 2 \times 8$

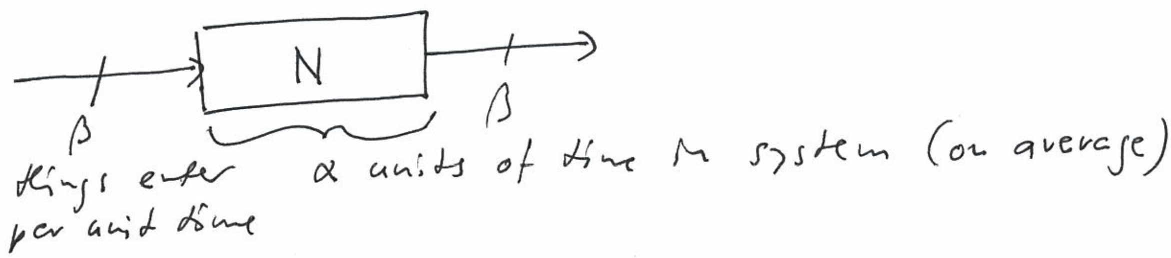
Example 2: In your wine cellar, <u>on average</u>

- there are 600 bottles
- you drink and buy 50/year

How long is every bottle in the cellar? $12 = 600/5$

Little's Law: Given a stable (arrival rate = departure rate) system

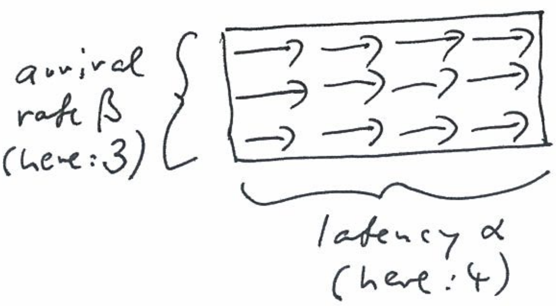
N = # things in system



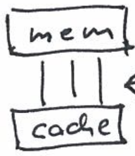
Then: $N = \alpha\beta$

Seems trivial, but the interesting aspect is the independence of input/output distribution.

Useful visualization:



Example: Memory system



$\leftarrow$ latency $\times$ throughput $=$ $\underbrace{\text{concurrency}}_{\text{bytes "in flight"}}$

latency: halves/ 9 years

throughput: doubles/ 3 years

$\underbrace{\hspace{6em}}$ x4 every 9 years $\rightarrow$ makes case for parallel processing

Core 2: $\beta = 2$ B / cycle

$\alpha \approx 100$ B / cycle

$\Rightarrow N = 200$ B $= 50$ floats or ints

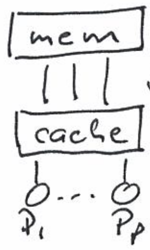
Performance Bounds: Roofline model

(Williams et al. 2008)

Which resources in a microarchitecture bound performance?

- peak performance [ops/cycle]
- memory bandwidth [bytes/cycle]
- <other>

Platform model:



bandwidth β $\leftarrow$ carefully measured
- raw bandwidth unattainable
- Stream benchmark may be conservative

peak perf. π

Algorithm/program model:

$$\text{operational intensity} = \frac{W}{Q} = \frac{\#\text{ops}}{\#\text{bytes transferred mem} \leftrightarrow \text{cache}} \left[\frac{\text{ops}}{\text{byte}}\right]$$

"ops" could be
- floating point ops (adds/mults) for numerical code
- comparisons (for sorting)
- <other>

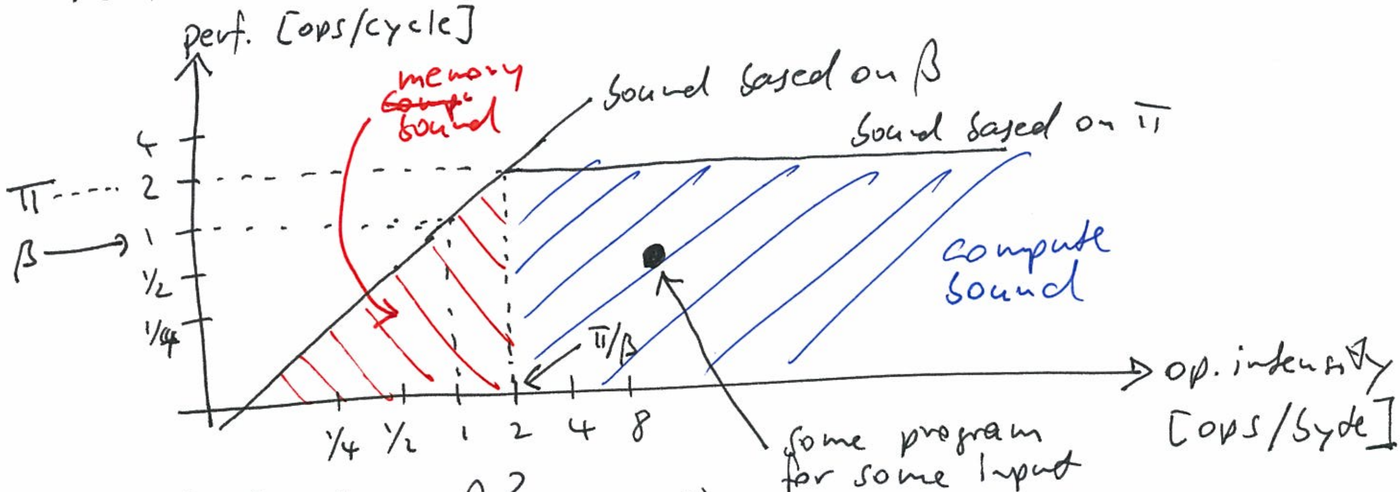
W: derive on paper or
instrument your code or
perf. counters

Q: perf. counters
[roofline model assumes cold cache]

For some functions, asymptotic results on optimal op. intensity are available.
Assume cache size of m bytes. Then the optimal $W/Q = W/Q_m$ is

FFT/sorting: $\Theta(\log m)$ (m-way mergesort)
Matrix mult.: $\Theta(\sqrt{m})$ (suitably blocked)

Roofline model: (example $\pi = 2$, $\beta = 1$)



bound based on β? (on input)
- assume program has op. intensity of $x \frac{\text{ops}}{\text{byte}}$
- the program gets at most $\beta \frac{\text{bytes}}{\text{cycle}}$

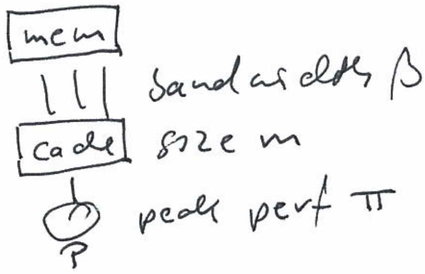
$\Rightarrow$ perf $= y \le \beta \cdot x$

$\Leftrightarrow \log_2(y) \le \log_2(x) + \log_2(\beta)$ (slope is $=1$)

$x = 1 \Rightarrow y \le \beta$

Balance principles I (Kung 86)

platform model:



algorithm model:

Q_m: data transfer mem $\leftrightarrow$ cache

W: work = #ops

The processor is called <u>balanced</u> for an algorithm (on an input) if compute time = data transfer time, i.e., assuming perfect utilization:

$$\frac{W}{\pi} = \frac{Q_m}{\beta} \iff \frac{\pi}{\beta} = \frac{W}{Q_m}$$ (compare to roofline model!)

If π is increased by $a\times$ ("times"), β can also be increased by $a\times$ to rebalance. But π grows faster than β!

Hence this question: If $\pi \to a\pi$, how much to increase m to rebalance?

Consider algorithms with optimal $\frac{W}{Q_m}$

Case 1: Matrix mult.

$$\frac{W}{Q_m} = \Theta(\sqrt{m})$$

$\Rightarrow \frac{\pi}{\beta}$ increased $\times a$ $\to$ m needs increase $\times a^2$

Case 2: FFT / sorting

$$\frac{W}{Q_m} = \Theta(\log m)$$

$\Rightarrow \frac{\pi}{\beta}$ increased $\times a$ $\to$ m needs increase to m^a

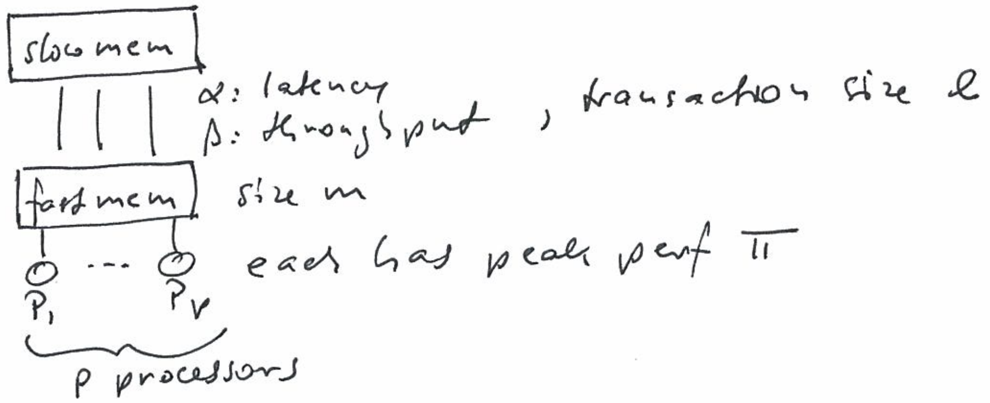
Both are unrealistic.

Balance principles II (Czechowski et al. 2011)

Goal: More detailed balance principles for multicores

Applications:
- relationship between hardware and algorithm parameters
- reasoning about architecture trends

Platform model:



Algorithm model:

PRAM: $W(n)$, $D(n)$

Memory transfers of size l: $Q_{m,l}(n)$

Assumptions:
- optimal $W(n)$
- optimal $W(n)/Q_{m,l}(n)$

Question: How to get $Q_{p;m,l}(n)$ (i.e., for p processors)

Answer:
- there are general bounds based on $Q_{m,l}(n)$ and a given scheduler
- some direct results are available

~~Algo~~ Processor is balanced for algorithm if

$$T_{mem} \leq T_{comp} \qquad \text{(Kung used "=")}$$

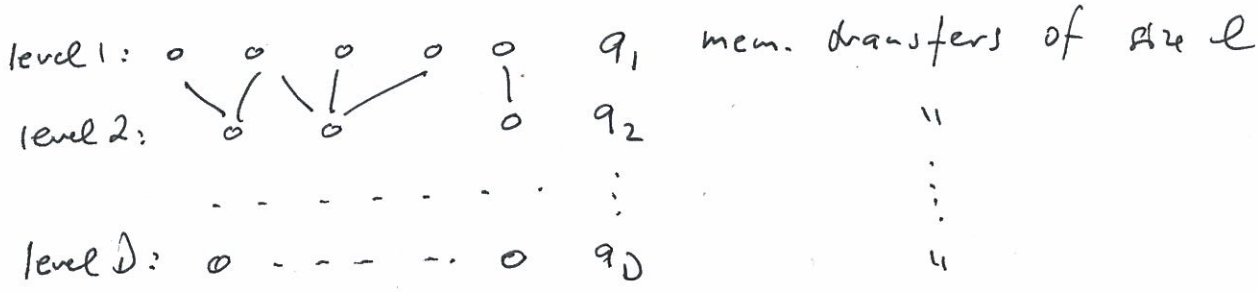
This is equivalent to "compute-bound".

Kung would get:

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

Estimating T_{mem}:

Idea: divide DAG into levels



$$\Sigma = Q_{m,\ell}$$

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

$$\Rightarrow T_{mem} \approx \sum_{i=1}^{D} \left(\alpha + \frac{q_i \cdot \ell}{\beta}\right) = \alpha D(n) + \frac{Q_{p;m,\ell}(n)}{\beta}$$

Estimating T_{comp}: Brent's theorem

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

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

mem. parallelism — comp. parallelism

$$\Leftrightarrow \frac{p\pi}{\beta}\left(1 + \frac{\alpha\beta/\ell}{Q/D}\right) \leq \frac{W}{Q \cdot \ell}\left(1 + \frac{p}{W/D}\right)$$

Kung

mem. parallelism of algorithm — comp. parallelism of algorithm

Example: Matrix mult.

$$Q \geq \frac{W}{\sqrt{2} \cdot \ell \sqrt{m/p}} \quad \text{(Irony et al. 2004)}$$

$\Rightarrow$ Balance principle $\frac{p\pi}{\beta} \leq \sqrt{\frac{m}{p}}$

Doubling p requires doubling both β and m
or $m \to 8m$ to rebalance

Note: doubling p is very different than doubling π
(even though both double peak perf.)

Other example: Comparison-based sorting (+ work stealing)

$$\frac{p\pi}{\beta} \leq O\left(\log \frac{m}{p}\right)$$