Design of Parallel & High Performance Computing

Reasoning about Performance I

Amdahl's Law – PRAM – α-β Model – Little's Law
Operational Intensity – Roofline I

1. Amdahl's Law (1967)

Gene Amdahl (1922–2015)
computer architect & entrepreneur

A program runs in time T_1 on one processor. A fraction f, $0 \le f \le 1$, of it is sequential. Let T_p be the runtime on p processors. Then

$$T_p \geq \frac{(1-f)T_1}{p} + f\,T_1$$

picture T_1: $\overbrace{\quad}^{fT_1}\overbrace{\qquad\qquad}^{(1-f)T_1}$

speedup: $S_p \le \frac{T_1}{T_p} \le \frac{1}{\frac{1-f}{p} + f}$

efficiency: $E_p = \frac{S_p}{p} \le \frac{1}{1-f+fp}$

$p \to \infty$: $T_\infty \geq f\,T_1$

$$S_\infty \le \frac{1}{f}$$

$$E_\infty = 0 \quad \text{if } f \neq 0$$

Is Amdahl's Law optimistic or pessimistic?

Pessimistic:

a.) AL fixes the problem size, but more processors usually means larger problems. Take this into account: $T_1(n)$, $f(n)$, ...

Gustafson's Law (1988)

$$S_\infty(n) \le \frac{1}{f(n)} \xrightarrow[n\to\infty]{} \infty \quad \text{if } f(n) \to 0$$

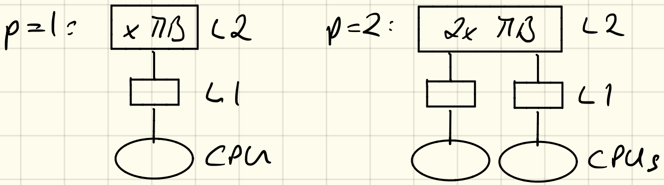
i.e. if sequential part $\to 0$ for large n

Terms:

- strong scaling: behavior of $S_p(n)$ for fixed n and $p \to \infty$
- weak scaling: behavior of $S_p(n)$ for $n, p \to \infty$

5.) AL assumes that by increasing p all other resources stay the same. If this is not the case, superlinear speedup is possible: e.g.

- data caches scale: working set suddenly fits into cache, e.g.



- memory bandwidth scales: $p=2$ threads may have twice the bandwidth

Optimistic

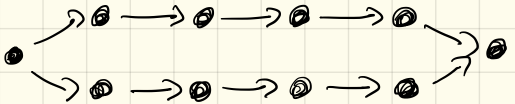
a.) Ignores overhead of parallelization (e.g. creating threads) which increases with p.

b.) Assumes perfect load balancing.

So in reality: $S_p(n) = \frac{T_1(n)}{T_p(n) + A_p(n)}$

Overhead

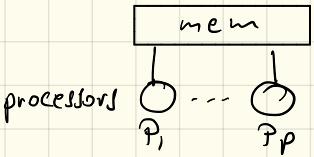
c.) Programs often have no sequential or infinitely parallelizable part. Example:



For this we need better models that take graph structure into account.

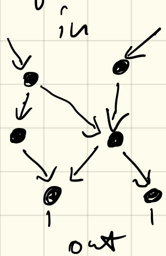
2. PRAM model

Computer:



all processors can access memory in unit time

Program: DAG (directed acyclic graph)



nodes: unit time ops
edges: dependencies

$W(n)$ = # nodes (work)
$D(n)$ = longest path from in to out (depth, span)

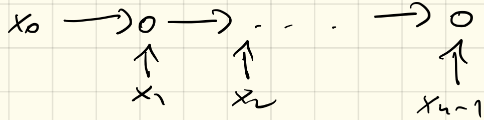
average parallelism: $W(n)/D(n)$

Examples:

a) Reduction: $x_0 + x_1 + \dots + x_{n-1}$

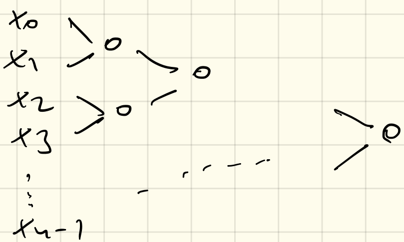
sequential: $W(n) = \Theta(n)$
$D(n) = \Theta(n)$

avg. par: $O(1)$



binary tree: $W(n) = \Theta(n)$
$D(n) = \Theta(\log n)$

avg. par: $O(n/\log n)$



b) Mergesort: L list of length n

sort(L)

    if length(L) = 1 return L
    L_1 = sort(left(L))
    L_2 = sort(right(L))
    return merge(L_1, L_2)

$W(n) = \Theta(n \log n)$

$D(n) = D(\frac{n}{2}) + \Theta(n) = \Theta(n)$

avg. par. $O(\log n)$

Note: parallel merge exists
$\Rightarrow$ shorter $D(n)$

---

c) Scan:

input: $L = (x_0, \ldots, x_{n-1})$ output: $(0, x_0, x_0 + x_1, \ldots, x_0 + \ldots + x_{n-2})$

sequential: $W(n) = D(n) = \Theta(n)$

scan(L)

    if length(L) = 1 return (0)
    sums = $(x_0 + x_1, \ldots, x_{n-2} + x_{n-1})$
    evens = scan(sums)
    odds = $(\text{evens}[i] + x_{2i} \mid i = 0 \ldots \lceil \frac{n}{2} - 1 \rceil$
    return interleave(evens, odds)

$W(n) = W(\frac{n}{2}) + \Theta(n) = \Theta(n)$

$D(n) = D(\frac{n}{2}) + \Theta(1) =$
$= \Theta(\log n)$

avg. par $\Theta(n / \log n)$

Reasoning in PRAM

Given a DAG with $W(n)$ nodes and $D(n)$ depth.

Sequential runtime: $T_1(n) = W(n)$

Time on ∞ processors: $T_\infty(n) = D(n)$

" p " $T_p(n) = ?$

$T_p(n) \geq D(n)\,,\ W(n)/p$

$\rightarrow T_p(n) \leq D(n) + (W(n) - D(n))/p$

Brent's Lemma (1974)

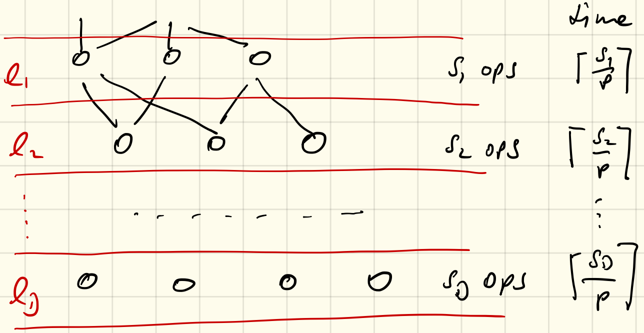
in summary:

$$W(n)/p \leq T_p(n) \leq W(n)/p + D(n)$$

(compare to Amdahl's law)

Proof of Brent's Lemma

Idea: divide DAG into levels



$\sum = W$ ops

$$T_p(n) \leq \sum_{i=1}^{D} \left\lceil \frac{s_i}{p} \right\rceil \leq \sum_{i=1}^{D} \frac{s_i + p - 1}{p}$$

$$= \frac{1}{p} W(n) + \frac{p-1}{p} D(n)$$

$$= D(n) + \frac{W(n) - D(n)}{p}$$

$$\leq D(n) + W(n)/p$$

Speedups:

$$S_p(n) = T_1(n) / T_p(n)$$

$$S_p(n) \le W(n)/D(n), \le p$$

$$S_p(n) \geq \frac{p}{\frac{D(n)}{W(n)}p + 1} \xrightarrow[n \to \infty]{} \frac{W(n)}{D(n)}$$

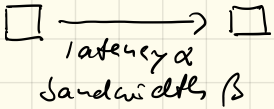
$$S_\infty(n) = \frac{W(n)}{D(n)}$$

so: if n is fixed then speedup is limited; for $n \to \infty$ speedup can be unbounded

Example: tree reduction

$$S_p(n) \geq \frac{p}{\frac{\log_2 n}{n}p + 1} \xrightarrow[p \to \infty]{} \frac{n}{\log_2 n}$$

3. α-β Model

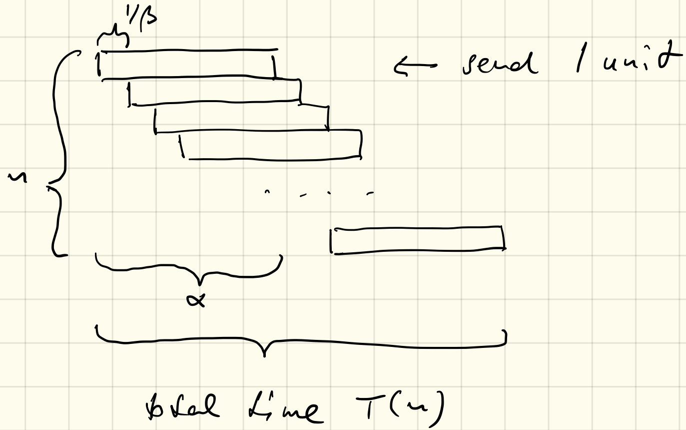


How long does it take to send a message of size n?

units:

α [cycles], β [units/cycle]

Intuition:



$$T(n) = \frac{n}{\beta} + \alpha$$

4. Little's Law

John Little (1928-), Professor MIT

In a Starbucks, on average

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

How many people are inside?

$2\cdot 8 = 16$

In your wine cellar, on average

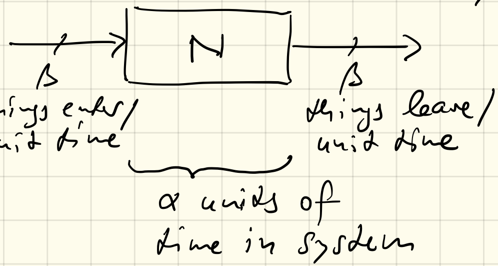
- there are 600 bottles
- you drink and buy 50/year

How long is every bottle in the cellar?

$600/50 = 12$

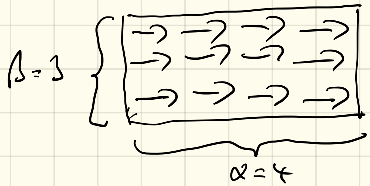
Little's Law: Given a stable system (input rate = output rate)

N = # things in system



Then: $n = \alpha\beta$

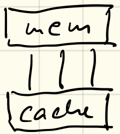
Visualization:



N = 12

Seems trivial but crucial is independence of I/O distribution

Example: Memory System



latency × throughput = concurrency (bytes in flight)

latency: halves / 9 years

throughput: doubles / 3 years

×4 every 9 years → makes case for parallel processing

			$\alpha\beta \approx$
Intel Core 2 (2006):	$\beta = 2$ bytes/cycle	$\alpha \approx 100$ β/cycle	200
Intel Haswell (2014): (oct-core)	$\beta = 23$	$\alpha = 63$	1450

5. Roofline Model (Williams et al. 2008)

resources in a microarchitecture that bound performance:	associated program features
• peak performance π [flops/cycle]	• work W [flops]
• memory bandwidth β [bytes/cycle]	• data transferred cache $\leftrightarrow$ mem Q [bytes]

for a given problem: min W = work/time complexity

min Q = I/O complexity

<u>Operational intensity I</u>: Given a program, assume empty cache:

$$I(n) = W(n)/Q(n)$$

Intuition: high I $\leftrightarrow$ compute bound

low I $\leftrightarrow$ memory bound

Example: asymptotic bounds on I

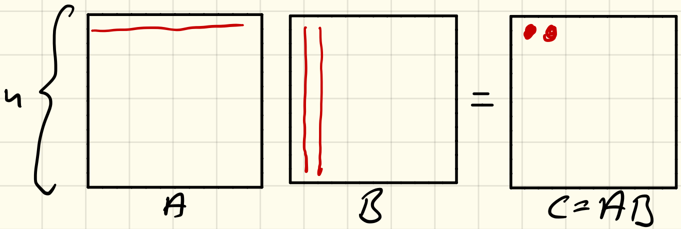
		$I(n)$
vector sum	$y = x + y$	$O(1)$
matrix-vector product	$y = Ax$	$O(1)$
fast Fourier transform		$O(\log n)$
matrix-matrix product	$C = AB$	$O(n)$

Operational Intensity: Example Matrix Multiplication

Assumptions: cache size $\gamma \ll n$, cache block = 8 doubles, 1 cache

We want to estimate Q. $W = 2n^3$ flops

1.) Triple loop:



1. entry of C: $n + 8n$ doubles

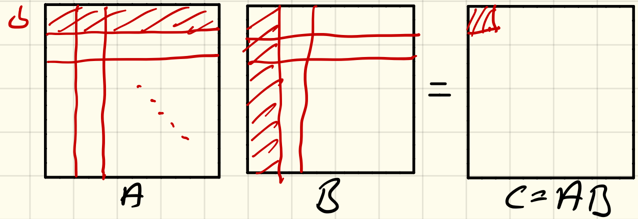
2. entry of C: same

. . . . .

total: $9n^3$ doubles

$I(n) = O(1)$

2.) Blocked: $8 \mid b$, $3b^2 \le \gamma$



1. block of C: $2nb$ doubles

2. block of C: same

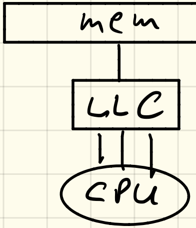
. . . .

total: $2nb \cdot \left(\frac{n}{b}\right)^2 = \frac{2n^3}{b}$ doubles

$b = \sqrt{\frac{\gamma}{3}} \Rightarrow I(n) = \Theta(\sqrt{\gamma})$

Roofline Model (Williams et al. 2008)

Computer:



bandwidth β [bytes/cycle]

peak perf. π [flops/cycle]

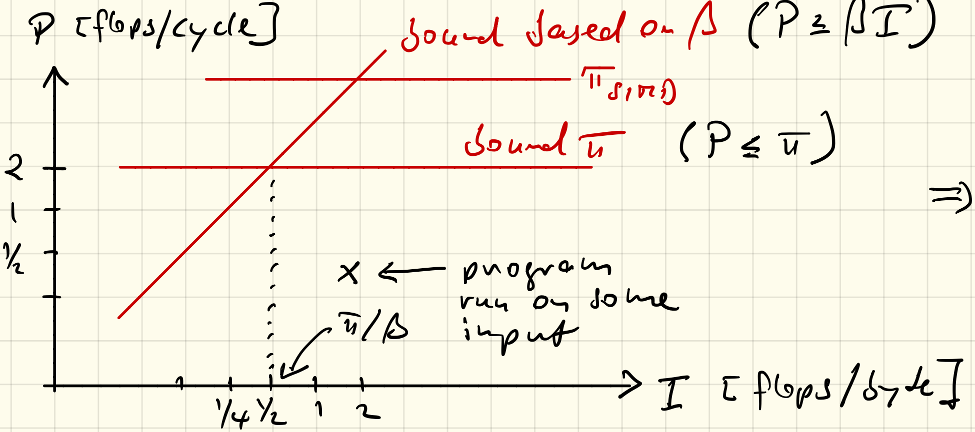
Program:

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

T = runtime [cycles]

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

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



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

$P \leq \beta \cdot I$

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

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