Design of Parallel & High Performance Computing

Reasoning about Performance I

Amdahl's Law - PRAM - A-B Model - Little's Law
Amdahl’s Law (1967)

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

$$T_p \geq \frac{(1-f)T_1}{p} + f \frac{T_1}{p} = \frac{fT_1}{p} + \frac{(1-f)T_1}{p}.$$

picture $T_1$: $0 \leq \frac{fT_1}{p} \leq \frac{1}{T_1}$

Speedup: $Sp \leq \frac{T_1}{T_p} \leq \frac{1}{1-f+f/p}$

Efficiency: $Ep = \frac{Sp}{p} \leq \frac{1}{1-f+f/p}$

$p \to \infty$: $T_\infty \geq fT_1$

$Sp_\infty \leq \frac{1}{f}$

$Ep_\infty = 0$ if $f \neq 0$

Is Amdahl’s Law optimistic or pessimistic?

Gene Amdahl (1922–2015)

Computer architect & entrepreneur

Pessimistic:

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


$$Sp(n) \leq \frac{1}{f(n)} \to \infty \text{ if } f(n) \to 0$$

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

Terms:

- Strong scaling: behavior of $Sp(n)$ for fixed $n$ and $p \to \infty$

- Weak scaling: behavior of $Sp(n)$ 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.

\[
p = 1: \begin{array}{c}
\text{CPU} \\
\text{L1}
\end{array} \quad p = 2: \begin{array}{c}
\text{CPU} \\
\text{L1} \quad \text{L1}
\end{array}
\]

- memory bandwidth scales: \( p = 2 \) threads may have faster bandwidth

Optimistic

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

b.) Assumes perfect load balancing.

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

Overhead

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

\[
\begin{array}{c}
\bullet \\
\rightarrow \\
\rightarrow \\
\rightarrow \\
\rightarrow \\
\rightarrow \\
\rightarrow
\end{array}
\]

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

Examples:

a) Reduction: \( X_0 + X_1 + \ldots + X_{n-1} \)
- Sequential: \( T(n) = \Theta(n) \)
- Parallel: \( T(n) = \Theta(1) \)

Binary tree: \( W(n) = \Theta(n) \)
- Sequential: \( T(n) = \Theta(\log n) \)
- Parallel: \( T(n) = \Theta(n/\log n) \)

average parallelism: \( \frac{W(n)}{D(n)} \)
5) Meyesont: a list of length $n$

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

$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:** $W(n) = \Theta(n)$

```plaintext
scan(L)
if length(L) = 1 return (0)
sums = (x_0 + x_1, \ldots, x_{n-2} + x_{n-1})
evens = scan(sums)
odds = (odds circ \text{even}) + x_{n-1} | i = 0, \ldots, \frac{n-1}{2} - 1
return interleave(evens, odds)
```

```
$W(n) = W\left(\frac{n}{2}\right) + \Theta(1) = \Theta(n)$
```

```
\(J(n) = J\left(\frac{n}{2}\right) + \Theta(1) = \Theta(\log n)\)
```

```plaintext
\text{any, par } O\left(n/\log n\right)
```

Note: parallel merge exists \(\Rightarrow\) shorter $J(n)$
Given a DAG with \( W(n) \) nodes and \( D(n) \) depth.

Sequential runtime: \( T_s(n) = W(n) \)

Time on \( p \) processors: \( T_{cp}(n) = \frac{W(n)}{p} \)

But \( T_{cp}(n) \leq D(n) + \frac{(W(n) - D(n))}{p} \)

Proof of Trend's theorem

**Idea:** Divide DAG into levels

\[ T_p(n) \leq \sum_{i=1}^{D(n)} \left[ \frac{S_i}{p} \right] \leq \sum_{i=1}^{D(n)} \frac{S_i + p - 1}{p} \]

In summary:

\[ \frac{W(n)}{p} \leq T_{cp}(n) \leq \frac{W(n)}{p} + D(n) \]

(compare to Amdahl's law)
Speedups:

\[ S_p(n) = \frac{T_c(n)}{T_p(n)} \]

\[ S_p(n) \leq \frac{W(n)}{\rho(n)} \leq \rho \]

\[ S_p(n) \geq \frac{\rho}{\frac{W(n)}{\rho(n)} + 1} \rightarrow \frac{W(n)}{\rho(n)} \]

\[ S_0(n) = \frac{W(n)}{\rho(n)} \]

So: if \( n \) is fixed, then speedup is limited; for \( n \rightarrow \infty \), speedup can be unsounded.

Example: tree reduction

\[ S_p(n) \geq \frac{\rho}{1 + \frac{n}{\rho(n)} + 1} \rightarrow \frac{n}{\rho(n)} \]

3. \( \alpha - \beta \) Model

How long does it take to send a message of size \( n \) units:

\( \alpha \) [cycles], \( \beta \) [units/cycle]

Intuition:

\( \frac{1}{\beta} \)

\( \rightarrow \) send 1 unit

Total time \( T(n) \)

\[ T(n) = \frac{n}{\beta} + \alpha \]
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? \[ 2 \times 8 = 16 \]

In your wine cellar, on average
- there are 600 bottles
- you drink and buy 50/year

How long is every bottle in the cellar? \[ \frac{600}{50} = 12 \]

---

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

\[ N = \# \text{things in system} \]

\[ \text{things enter/unit time} \quad \text{things leave/unit time} \]

\[ \alpha \text{ units of time in system} \]

Then: \[ n = \frac{\alpha}{\beta} \]

Visualization:

\[ \beta = 3 \]

\[ \alpha = 4 \]

\[ N = 12 \]

 Seems trivial but crucial is independence of I/O distribution.
Example: Memory System

\[
\text{latency} \times \text{throughput} = \text{concurrency (bytes in flight)}
\]

\[
\text{x4 every 9 years} \rightarrow \text{makes case for parallel processing}
\]

Intel Core 2 (2006): \( \beta = 2 \) bytes/cycle \( \alpha = 100 \beta \) /cycle \( \frac{\alpha}{\beta} = 200 \)

Intel Haswell (2014): \( \beta = 23 \) \( \alpha = 63 \) \( \frac{\alpha}{\beta} = 1450 \)