1

Perf Analysis

$$\text{Speedup} = \frac{\text{Sequential Exec Time}}{\text{Parallel Exec Time}} = \psi(n, p) \quad (\text{psi})$$

problem size (n), # processors (p)

Sigma — $\sigma(n)$ = the sequential portion of comp (time)

phi — $\varphi(n)$ = portion that is parallel (time) can be run in parallel
(this time is still the seq exec time of the)

kappa — $\kappa(n,p)$ = time required for parallel overhead (time)

speedup (psi) — $$\psi(n,p) \le \frac{\text{seq time}}{\text{parallel time}} = \frac{\sigma(n) + \varphi(n)}{\sigma(n) + \frac{\varphi(n)}{p} + \kappa(n,p)} = \frac{T_1}{T_p}$$

$$\text{Efficiency} = \frac{\text{Seq. exec time} \times 1}{\text{processors used} \times \text{parallel exec time}} \triangleq \frac{\text{proc}\cdot\text{hours}}{\text{proc}\cdot\text{hours}} \triangleq \%$$

epsilon — $$\varepsilon(n,p) \le \frac{(\sigma(n) + \varphi(n)) \times 1}{p\underbrace{\left(\sigma(n) + \frac{\varphi(n)}{p} + \kappa(n,p)\right)}_{\text{parall exec time}}} = \frac{\sigma(n) + \varphi(n)}{p\sigma(n) + \varphi(n) + p\kappa(n,p)}$$

$\boxed{0 \le \varepsilon(n,p) \le 1}$ is a %

Amdahl's law explained

1.5

1 | can be parallelized | serial |

X Y

$$f = \frac{Y}{X+Y} \qquad Y = f(X+Y)$$

$$1-f = \frac{X+Y}{X+Y} - \frac{Y}{X+Y} =$$

$$1-f = \frac{X}{X+Y} \qquad X = (1-f)(X+Y)$$

$$T_1 = X+Y \qquad T_p = \frac{X}{p} + Y$$

$$\psi = \frac{T_1}{T_p} = \frac{X+Y}{\frac{X}{p}+Y} = \frac{(1-f)(X+Y) + f(X+Y)}{\frac{(1-f)(X+Y)}{p} + f(X+Y)}$$

$$= \frac{(1-f)+f \quad \cancel{(X+Y)}}{\frac{(1-f)}{p} + f \quad \cancel{(X+Y)}} \leq \boxed{\frac{1}{f + \left(\frac{1-f}{p}\right)}}$$

Amdahl's Law

Speedup $\psi(n,p) \le \dfrac{\sigma(n)+\varphi(n)}{\sigma(n)+\varphi(n)/p + \kappa(n,p)} \le \dfrac{\sigma(n)+\varphi(n)}{\sigma(n)+\frac{\varphi(n)}{p}}$

b/c if $\kappa(n,p) > 0 \Rightarrow$ drop from denom.
inequality still holds if u drop a "+" term from denom.

sequential fraction $\boxed{f = \dfrac{\sigma(n)}{\sigma(n)+\varphi(n)}}$ and substitute into speedup eq $\psi(n,p)$

$$1-f = \underbrace{\frac{\sigma(n)+\varphi(n)}{\sigma(n)+\varphi(n)}}_{1} - \underbrace{\frac{\sigma(n)}{\sigma(n)+\varphi(n)}}_{f} = \boxed{\frac{\varphi(n)}{\sigma(n)+\varphi(n)} = (1-f)}$$

$$\psi(n,p) \le \frac{\sigma(n)+\varphi(n)}{\sigma(n)+\frac{\varphi(n)}{p}} = \frac{1}{\dfrac{\sigma(n)+\varphi(n)/p}{\sigma(n)+\varphi(n)}} = \frac{1}{\underbrace{\dfrac{\sigma(n)}{\sigma(n)+\varphi(n)}}_{f} + \underbrace{\dfrac{\varphi(n)}{\sigma(n)+\varphi(n)}}_{(1-f)} \cdot \dfrac{1}{p}}$$

$$\boxed{\psi(n,p) \le \frac{1}{f + \frac{(1-f)}{p}}}$$ } Amdahl's Law

this assumes that we're trying to solve a fixed sized problem as fast as possible. provides an upper bound.

Amdahl's Law con't

$$\psi \le \frac{1}{f + \frac{(1-f)}{p}}$$ look at asymptotic behavior as $p \to \infty$

$$\psi(n,p) \le \lim_{p \to \infty} \left(\frac{1}{f + \frac{(1-f)}{p}} \right) = \frac{1}{f}$$ $\Rightarrow$ f needs to be close to 0.

ie, seq frac need to be small.

---

examples: suppose 90% of program is parallelizable

$\Rightarrow f = .1$, on 8 procs

$$\frac{1}{.1 + (.9)/8} = 4.7$$

on ∞ procs $\frac{1}{.1} = 10$ max achievable.

---

limitation of Amdahl's

Suppose $\sigma(n) = (18{,}000 + n)\ \mu S$

$\varphi(n) = \frac{n^2}{100}\ \mu S$

$n = 1000$

$$f = \frac{18{,}000 + n}{(18{,}000 + n) + \frac{n^2}{100}} = \frac{19{,}000}{19{,}000 + 10{,}000}$$

now lets say $n = 1000$

$$= .655 \text{ (seq frac)}$$

$$\psi \le \frac{(18{,}000 + n) + \frac{n^2}{100}}{(18{,}000 + n) + \frac{n^2}{100p}} =$$

$n = 10{,}000$

$$f = \frac{28{,}000}{28{,}000 + 1{,}000{,}000} = .027 \text{ (seq frac)}$$

$\Rightarrow$ making n larger can make f smaller $\Rightarrow$ larger speedup.

however this also ignores overhead introduced due to parallelization.

Further limitations of Amdahl's Law (ignores comm overhead)
continuing prev example

Suppose there are $\lceil \log n \rceil$ places where comm takes place, and at
each place

$T_c = 10{,}000 \lceil \log p \rceil + n/10 \ \mu s \Rightarrow$ total comm time is

$\lceil \log n \rceil \left(10{,}000 \lceil \log p \rceil + n/10 \right) \mu s$. for $n = 10{,}000 \Rightarrow$

Kappa. $K = \boxed{14 \left(10{,}000 \lceil \log p \rceil + 1000 \right) \mu s} = O\left(\lceil \log n \rceil \left(\lceil \log p \rceil + n \right) \right)$
$= O\left(\lceil \log n \rceil \lceil \log p \rceil + n \lceil \log n \rceil \right)$

therefore including all factors

$$\psi \leq \frac{28{,}000 + 1{,}000{,}000}{42{,}000 + \frac{1{,}000{,}000}{p} + 140{,}000 \lceil \log p \rceil}$$

} want the denom to be small $\Rightarrow$ bigger speedup

$p \to \infty$ small gets bigger

Amdahl Effect if $O(\kappa(n,p)) < O(\varphi(n))$

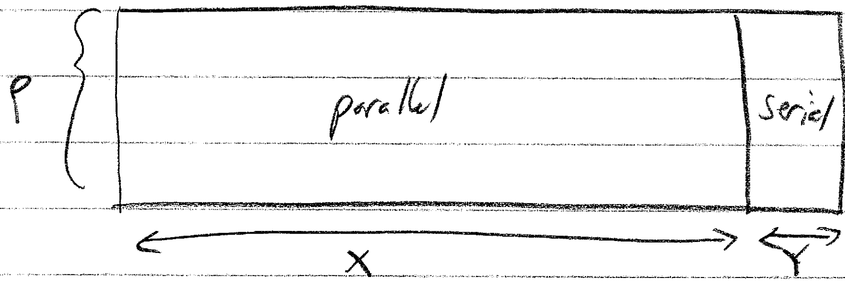
then making n bigger improves speedup for a given p.

However sometimes you're not interested in a given problem faster, you're interested in solving a more detailed problem (ie, larger n) in the same amount of time. This is the focus of

Gustafson's Barsis' Law $\longrightarrow$

$\boxed{\text{G-B-law explained.}}$

$\cancel{\psi(n,p) \le \dfrac{\sigma(n)+\varphi(n)}{\sigma(n)+\varphi(n)/p}}$ (here excluding $\kappa(n,p)$)



$$s = \frac{Y}{X+Y} \qquad 1-s = \frac{X}{X+Y}$$

$$Y = s(X+Y) \qquad X = (1-s)(X+Y)$$

$$T_1 = pX + Y, \quad T_p = X+Y$$

$$\psi = \frac{T_1}{T_p} = \frac{pX+Y}{X+Y} = \frac{p(1-s)(X+Y)+s(X+Y)}{X+Y}$$

$$= \frac{(p(1-s)+s)\cancel{(X+Y)}}{\cancel{(X+Y)}} = p(1-s)+s = p-sp+s$$
$$= p+s-sp$$

$$\boxed{\psi \le p + (1-p)s}$$

assumes that single CPU has enough memory.
called scaled speedup b/c starts w/ parall first

G-B example

Ex1 $T_p = 220s \quad p = 64 \qquad s = 5\% = .05$

$\Psi = p + (1-p)s = 64 + (1-64)(.05) = 60.85 \times$

Ex2 $p = 16{,}384$ need $\Psi = 15{,}000$ what can s be?

$15000 = 16{,}384 + (1 - 16{,}384)s$, solve for s

$s = .084 \Rightarrow 8.4\%$

Compare to Amdahl's

$$\Psi = \frac{1}{f + \frac{(1-f)}{p}} = 1500 = \frac{1}{f + \frac{(1-f)}{16{,}384}}$$

$$\Psi\left(f + \frac{1-f}{p}\right) = 1$$

$$f + \frac{1-f}{p} = \frac{1}{\Psi}$$

$$pf + 1 - f = \frac{p}{\Psi}$$

$$f(p-1) = \frac{p}{\Psi} - 1$$

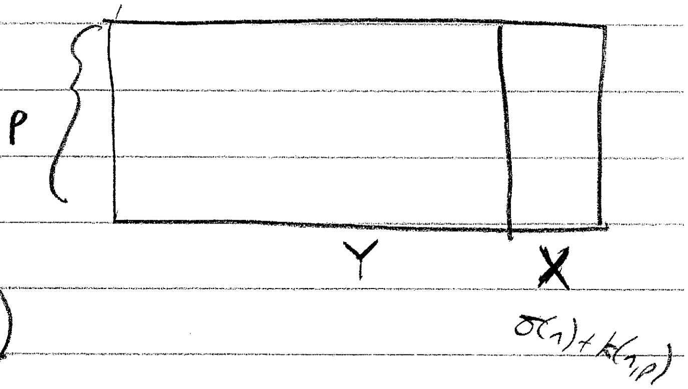
$$f = \frac{\frac{p}{\Psi} - 1}{p-1} = \frac{\frac{16{,}384}{1500} - 1}{16{,}384 - 1}$$

Karp-Flatt Metric

because GB and Amdahl's ignore $\kappa(n,p)$
Can overestimate speedup

$$T(n,p) = \sigma(n) + \frac{\varphi(n)}{p} + \kappa(n,p)$$

$$T(n,1) = \sigma(n) + \varphi(n)$$



Let

$$\boxed{e = \frac{\sigma(n) + \kappa(n,p)}{T(n,1)}}$$

experimentally determined sequential fraction

$$T(n,1)\,e = \sigma(n) + \kappa(n,p)$$

$$T(n,p) = T(n,1)\,e + \frac{T(n,1)(1-e)}{p}$$

$$\psi = \frac{T(n,1)}{T(n,p)} \Rightarrow T(n,1) = \psi\, T(n,p)$$

$$\cancel{T(n,p)} = \psi \cancel{T(n,p)}\, e + \frac{\psi \cancel{T(n,p)} (1-e)}{p}$$

$$1 = \psi e + \frac{\psi(1-e)}{p}$$

$$p = \psi e p + \psi(1-e) = \psi e p + \psi - \psi e = \cancel{e(\psi p - \psi)}$$

$$p = \psi e (p-1) + \psi$$

$$\frac{p-\psi}{\psi(p-1)} = e = \frac{\frac{1}{p\psi}(p-\psi)}{\frac{1}{p\psi}\psi(p-1)} = \boxed{\frac{\frac{1}{\psi} - \frac{1}{p}}{1 - \frac{1}{p}}} = e$$

K-F good because it takes $K(n,p)$ into acct and also that the work $\varphi(n)$ may not be evenly divided among p processors (bad imbalance)

for fixed problem size (Eff goes down typically)

e can tell you whether Eff ↓ is due to:

1.) limited parallelism or
2). increases of algorithmic / architectural overhead

take away { if e stays constant as p↑ then limited parallelism
if e ↑ as p↑ then parallel overhead.

---

finally Isoefficiency Metric.

take $\psi(n,p) \leq \dfrac{\sigma(n) + \varphi(n)}{\sigma(n) + \frac{\varphi(n)}{p} + K(n,p)}$ do a bunch of algebra substituting + manip

$T(n,1) \leq \dfrac{\varepsilon(n,p)}{1-\varepsilon(n,p)} T_0(n,p)$ where $\varepsilon(n,p)$ is efficiency and $T_0(n,p)$ is the Total amount of time spent by all processes doin work not done by sequential algorithm i.e. sequential code, IPC, redundant computations.

all the overhead × #p doing it.

$T_0(n,p) = (p-1)\sigma(n) + pK(n,p)$

1 proc must have done the seq stuff anyway

the objective is to have efficiency const as we scale the system.

$\Rightarrow$ let $C = \dfrac{\varepsilon(n,p)}{1-\varepsilon(n,p)}$ (constant).

Iso Efficiency Relation.

$$\Rightarrow \quad T(n,1) \geq C\, T_0(n,p), \quad C = \frac{\varepsilon(n,p)}{1-\varepsilon(n,p)}$$

$$T_0(n,p) = (p-1)\sigma(n) + p\kappa(n,p)$$

in order for $\varepsilon(n,p)$ to be constant as $p\uparrow$, n must increase as well by an amount sufficient to keep inequality true.

sense $n\uparrow$ eventually you may hit an upper limit of MM available per processor. as such, this will be considered the bottleneck.

Example 1

$$T(n,1) \geq C\, T_0(n,p)$$

part of seq that can't be done in ∥ $=0$

reduction sum.

$O(n)$ seque

simple linear sum

$$n \geq C\left[(p-1)\cancel{\sigma(n)} + p\kappa(n,p)\right]$$

↳ $\log p$ comm steps (tree pattern)

$$n \geq C(p \log p)$$

in terms of memory require $M(n) = n$ 3 list of n #'s

constant we ignore in big O

$$\frac{M(T_0(n,p))}{p} = \frac{M(p\log p)}{p} = \frac{p\log p}{p} = \log p$$

if you double p and you double n then each proc still has the same # of elements $\frac{n}{p}$ but you have 1 more step in reduction.

$$n \geq Cp\log p$$

plus a little more

$$2n \geq 2p(\log 2p)$$

Iso Example. Floyd's (2D row decomp).

$$T(n,1) \geq C T_o(n,p) = C(p-1)\sigma(n) + pk(n,p)$$

$T(n,1) = n^3$ } b/c of that tripple nested loop.

$\sigma(n)$ can all be done in # $= 0$ $\quad k(n,p) = n$ bcast $\times \log p \frac{\text{steps}}{\text{bcast}}$

$$n^3 \geq C\left(\cancel{(p-1)}\overset{0}{0} + p n^2 \log p\right)$$ $\times\ n$ data elements $= n^2 \log p$

$n \geq \underbrace{Cp\log p}$ $\Rightarrow$ looks good. double p, little more than double n

but $M(n) = n^2$

$M(n) = C^2 p^2 \log^2 p$.

$$\frac{M(n)}{p} = \frac{C^2 p^2 \log^2 p}{p} = C^2 p \log^2 p \Rightarrow \frac{\text{mem}}{\text{proc}} \text{ increase fast as } p\uparrow$$

2D checker board.

$$k(n,p) = 2 \text{ bcasts} \times \log p \frac{\text{steps}}{\text{bcast}} \times n \text{ iterations} \times \frac{n}{\sqrt{p}} \frac{\text{elements}}{\text{bcast}}$$

b/c not whole entire row.

from above

$$n^3 \geq Cpk(n,p) = Cn^2\log p \quad \frac{2n^2 \log p}{\sqrt{p}}$$

$$n^3 \geq \frac{2pn^2 \log p}{\sqrt{p}} \Rightarrow n \geq \frac{2p\log p}{\sqrt{p}} \Rightarrow n \geq 2\sqrt{p}\log p$$

better.

$M(n) = n^2 = Cp\log^2 p$