Performance

martes, 15 de marzo de 2022 06:56 a. m.

Ideally :

$$t_p^{ideal} = \left[\frac{t_s}{n_p}\right] \begin{matrix} \rightarrow \text{sequential time} \\ \rightarrow \text{number of processors} \end{matrix}$$

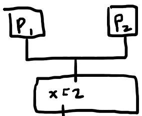
When working in parallel. From this, we can calculate Overcharged Time:

$$t_{sc} = [\, t_p - t_p^{ideal} \,]$$

t_p → usually bigger than t_p^{ideal}

→ this time can tell :

- In Shared Memory
  → Means we have problems in communication. There are many synchronization techniques that involve Blockers to Critical Sections. Example



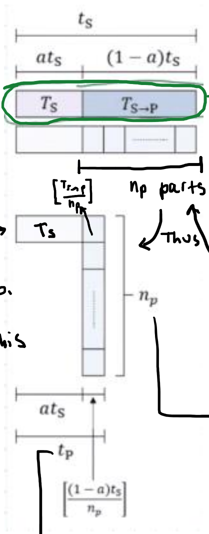
→ Critical Section (Memory) — increases Overcharged Time

  → Blocking serializes our program, which also involves dependency
- In Distributed Memory
  → Each processor has its memory, and thus only with Messaging the Overcharge Time increases.
  → The more Data we need to message through the network, the more Overcharge Time we will have

$$\uparrow \text{Data} = \uparrow t_{sc}$$

→ Ambdahl's Law

Is a specific case of the General Form



→ the whole length is the time it takes with 1 processor, we will call it t_s

↳ Divided into two:

T_s : time that naturally is executed in series, that cannot be parallelized

$T_{s\to p}$: time of execution that can be parallelized

Since we can parallelize this, we can divide it between the n_p processors: $\left[\frac{T_{s\to p}}{n_p}\right]$

$\left[\frac{T_{s\to p}}{n_p}\right]$ n_p parts

T_s Thus

Let's calculate an expression that determines the scalab. when we change to this

→ and the time diagram changes

→ Thus, we can rewrite T_s and $T_{s\to p}$

$a \in [0,1]$, represents the portion of data that cannot be parallel

i.e. $a = 0.4$ } 60% can be parallel, 40% can't be parallel

$t(n_p, n)$

And now we have :

T_s: $a t_s$

$T_{s\to p}$: $(1-a)\,t_s$

Lets now calculate acceleration

↱ $t(1, n)$

Lets now calculate acceleration

$$S = \frac{t_s}{t_p} = \frac{T_s + T_{s\to p}}{T_s + \left[\frac{T_{s\to p}}{n_p}\right]}$$

$t_s \to t(1, n)$

with $T_s = at_s$ and $T_{s\to p} = (1-a)\,t_s$, we now have

$$S = \frac{at_s + (1-a)t_s}{at_s + \left[\frac{(1-a)t_s}{np}\right]}$$

} we have $\frac{t_s \text{ (num)}}{t_s \text{ (denom)}}$, thus we can cancel out t_s term.

$$= \frac{a + (1-a)}{a + \frac{(1-a)}{np}}$$

$$\boxed{S = \frac{1}{\left[a + \frac{(1-a)}{np}\right]}}$$

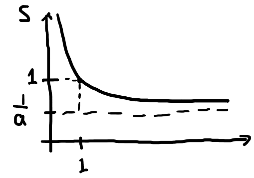
Amdahl's Law: a relationship of how time changes from 1 to np processors

If we say $np \to \infty$, then

$$\lim_{np\to\infty} S = \lim_{np\to\infty} \frac{1}{\left[a + \underbrace{\frac{(1-a)}{np}}_{\frac{(1-a)}{\infty} \approx 0}\right]} = \boxed{\frac{1}{a}}$$

the acceleration limit we can achieve

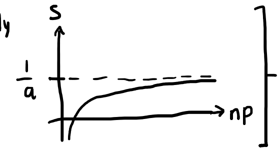
If we plot Amdahl's Law



→ when a is small, S increases fast

→ when a is large, S decreases fast (limit)

actually



] Strong Scalab.

when $np = 1$, $S = \frac{1}{\left[a + \frac{(1-a)}{1}\right]}$

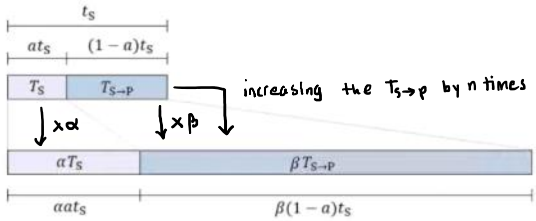
Let's say $a = 0.1$ (10%),

$S_{MAX} = \frac{1}{a} = \frac{1}{0.1} = 10$ } maximum scalability we can achieve in a problem where 10% of total time not parallel: no matter how many processors we put, acceleration (s) will be $S = 10$

Let's say $a = 0.01$ (1%)

$S_{MAX} = \frac{1}{a} = \frac{1}{0.01} = 100$ } The problem itself has almost no dependencies (a small), thus 100 of S.

However, Amdahl's Law does not take on account n (problem's size), and since n is fixed in the law, Scalability (Weak) is not considered. So if we want to scale n naturally, we need to scale it $\beta\,(T_{s\to p})$:



] we are increasing both times by a constant β, since n should increase with np processors: $n \times np$

Thus,

$at_s \to \alpha\, at_s$] represents that the problem increases

Thus,

$$\left.\begin{array}{l} a t_s \rightarrow \alpha\, a t_s \\ (1-a) t_s \rightarrow \beta (1-a) t_s \end{array}\right\}$$ represents that the problem increases with the amount of processors

Let's calculate scalability (acceleration),

$$S = \frac{t_s}{t_p} = \frac{T_s + T_{s\to p}}{T_s + \left[\frac{T_{s\to p}}{np}\right]} \quad \text{with} \quad T_s = \alpha a t_s \text{ and } T_{s\to p} = \beta(1-a)\, t_s$$

$$= \frac{\alpha a t_s + \beta(1-a) t_s}{\alpha a t_s + \left[\frac{\beta(1-a) t_s}{np}\right]}$$

Once again t_s is a common factor on both numerator and denominator, and therefore we can cancel it out,

$$= \frac{\alpha a + \beta(1-a)}{\alpha a + \left[\frac{\beta(1-a)}{np}\right]}$$

If we divide numerator and denominator by $1/\alpha$,

$$= \frac{\frac{\alpha a}{\alpha} + \frac{\beta(1-a)}{\alpha}}{\frac{\alpha a}{\alpha} + \left[\frac{\beta(1-a)}{\alpha np}\right]}$$

And we can again cancel out α,

$$= \frac{a + \left[\frac{\beta}{\alpha}\right](1-a)}{a + \left[\frac{\beta}{\alpha}\right]\frac{(1-a)}{np}} \quad , \quad \text{with } \gamma = \left[\frac{\beta}{\alpha}\right], \text{ we have}$$

$$\boxed{S = \frac{a + \gamma(1-a)}{a + \gamma\left[\frac{(1-a)}{np}\right]}}$$ General Law

→ If we have $\gamma = 1$ $(\beta = \alpha)$, we get Amdahl's Law:

$$S = \frac{a + (1-a)}{a + \left[\frac{1-a}{np}\right]} = \boxed{\frac{1}{a + \left[\frac{1-a}{np}\right]}}$$ Amdahl's Law

→ We want to scale the problem as always, that is, multiplying $n \times np$. Here, that would mean $\gamma = np$,

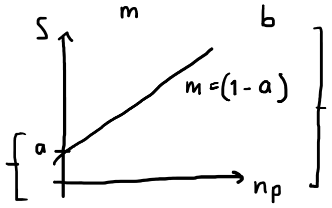
$$S = \frac{a + np(1-a)}{a + np\left[\frac{1-a}{np}\right]} = \frac{a + np(1-a)}{a + (1-a)} = a + np(1-a)$$

$$\boxed{S = (1-a)np + a}$$ Gustafson's Law

And we also can derive from General's Law, the Gustafson's Law. It has a linear form:

$$S = \underbrace{(1-a)}_{m} np + \underbrace{a}_{b}$$

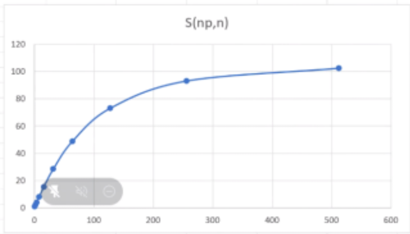
if $np = 0$
$S = a$

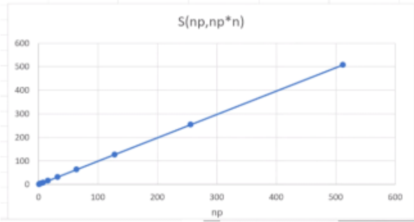


Weak Scalability

since we have γ factor

And these Scalability plots resemble what we have been doing:





Confirming that indeed strong and weak scalab have a log an linear form

these two plots are just the computation time, if you have communication, just add it at the end.

Summary

Amdahl's Law: Strong scalability, where $np \to \infty$ but n is fixed.
Gustafson's Law: Weak scalability, where $np \to \infty$ and $n = np \times n$.