L8 PERFORMANCE ANALYSIS

①

(QUINN) accurately predict the performance of a parallel algorithm to decide whether it is worth coding/debugging it.

analyze execution time of a parallel program: (1) helps understand the barriers to higher performance
(2) predict the improvement due to increased #procs.

AIMS

general formula for the speedup achievable by a parallel program

performance prediction formulas:

(1) Amdahl's law:
(helps decide whether a serial program merits parallelization)

(2) Gustafson-Barsis law:
(evaluate the performance of a parallel program)

(3) Karp-Flatt metric:
(helps decide whether the principal barrier to speedup is the amount of inherently sequential code, or parallel overhead)

(4) Isoefficiency metric:
(evaluate the scalability of a parallel algorithm executed on a parallel computer)

SPEEDUP & EFFICIENCY

$$\text{Speedup} = \frac{\text{serial exec. time}}{\text{parallel exec. time}}$$ // RATIO

op^s performed by a parallel alg. fall into 3 categories:

(1) serial computations
(2) parallel "
(3) inter-proc. com/tion, parallel overhead

using these 3 categories we can produce a simple model for speed-up.

$\psi(n,p)$ speedup achieved solving a pb of size n on p proc^s.

$\sigma(n)$ inherently sequential (serial) portion of the computation

$\varphi(n)$ portion of the computation that can be executed in parallel

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

Serial program on 1 proc. requires time $\sigma(n)+\varphi(n)$ to execute the computations.
(no inter-proc com/tion required)

consider best possible parallel exec. time: (on p procs)

- serial portion takes $\sigma(n)$ time
- parallel portion takes $\varphi(n)/p$ time (IDEAL)
- $+\ \kappa(n,p)$ time for inter-proc. comm/tion

ideal assumption: if parallel exec. time is larger, then speedup is smaller.

<u>IN SUMMARY</u>

$$\psi(n,p) \leq \frac{\sigma(n)+\varphi(n)}{\sigma(n)+\frac{\varphi(n)}{p}+\kappa(n,p)}$$

general speedup formula

adding procs reduces comp/tion of parallel portion, but increases com/tion time.

at some point the comm/tion time increase is larger than the comp. decrease so denom $\uparrow$ $\rightarrow$ $\psi(n,p)\downarrow$

- computation component decreases as $\#$proc^s $\uparrow$
- communication component increases as $\#$proc^s $\uparrow$

FOR A FIXED PB SIZE, THERE IS AN OPTIMAL # PROC^S, MINIMIZING overall EXEC TIM

$$= \frac{\text{speedup}}{p}$$

$$\underline{\text{efficiency}}_{\text{(of a } \| \text{ program)}} = \frac{\text{serial exec. time}}{(\#\text{proc}^{s}) \times \text{parallel exec. time}} =$$

it is a measure of processor utilization

efficiency of a $\|$ code solving pb of size n on p proc^{s}

$$\varepsilon(n,p) \leq \frac{\sigma(n)+\varphi(n)}{p[\sigma(n)+\kappa(n,p)]+\varphi(n)}$$

property: $0 \leq \varepsilon(n,p) \leq 1$

AMDAHL's LAW

since $\kappa(n,p) > 0$ we can discard it

$$\psi(n,p) \leq \frac{\sigma(n)+\varphi(n)}{\sigma(n)+\varphi(n)/p}$$

denote by f the inherently sequential portion of the computation

$$f = \frac{\sigma(n)}{\sigma(n)+\varphi(n)} \qquad \text{i.e. } \varphi(n) = \sigma(n)\left(\frac{1}{f}-1\right)$$

THEN $$\psi(n,p) \leq \frac{\sigma(n)/f}{\sigma(n)+\sigma(n)\left(\frac{1}{f}-1\right)/p} = \frac{1}{f+(1-f)/p}$$

Amdahl's Law

if f is the fraction of op^s in a computation that must be performed sequentially ($0 \leq f \leq 1$)

then the max speedup ψ achievable by a parallel computer with p proc^s

is $$\psi \leq \frac{1}{f + (1-f)/p}$$

~> upper bound on the speedup achievable when using p proc^s

~> determination of asymptotic speedup (as #procs increases)

ex1 10% ~> serial
90% ~> parallel

max speedup on 8 proc^s?

$$\psi(n,8) \leq \frac{1}{0.1 + (1-0.1)/8} \simeq 4.7$$

ex2 25% ~> ~~parallel~~ serial

max possible speedup?

$$\psi(n,p) \leq \frac{1}{0.25 + (1-0.25)/p}$$

$$\xrightarrow{p \to \infty} 4$$

⚠ Amdahl's law neglects parallel overhead.

Amdahl effect

typically $\kappa(n,p)$ has smaller complexity than $\sigma(n)$

e.g. $\Theta(n \log n)$, $\Theta(n^2)$

$f(x)$ is $\Theta(g(x))$ if $\exists\, x_0, M, N$ s.t.

$\forall x > x_0, \quad N|g(x)| < |f(x)| < M|g(x)|$

~> g bounds f above and below for sufficiently large x

~> gives tighter bounds on runtime than O-notation

conseq. increasing size of pb, increases the computation time ~~slower~~ faster than it increases the communication time

Amd. effect

~> for a fixed # procs speedup is usually an increasing fct of the pb. size.

hypothetical pb