CAP-387(2016) – Tópicos Especiais em Computação Aplicada: Construção de Aplicações Massivamente Paralelas

Aula 19: Speedup e Lei de Amdahl

Celso L. Mendes, Stephan Stephany

LAC / INPE

Emails: celso.mendes@inpe.br, stephan.stephany@inpe.br

Speedup

- Conceito: Speedup = Ganho de Desempenho
  - Pode ser o resultado de uma otimização qualquer
  - Exemplos:
    - Utilização de um novo algoritmo num programa sequencial
    - Uso de chaves de compilação
    - Uso de mais processadores (speedup *paralelo*)
  - Caso Geral:

    Speedup = tempo_original / tempo_otimizado

Speedup Paralelo

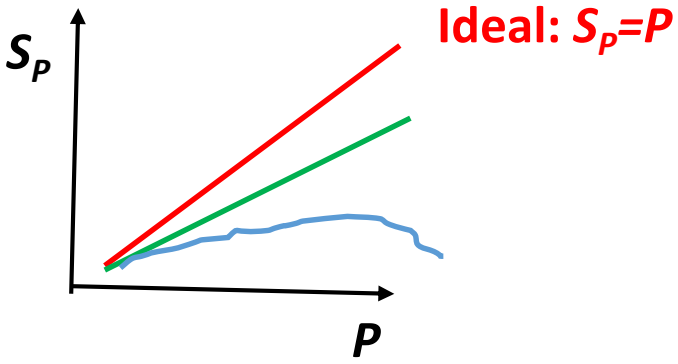
- Ganho de desempenho pelo uso de mais processadores
  - Tempo de execução com 1 processador: T_1
  - Tempo de execução com P processadores: T_P
  - Speedup Paralelo com P processadores: $S_P = T_1 / T_P$

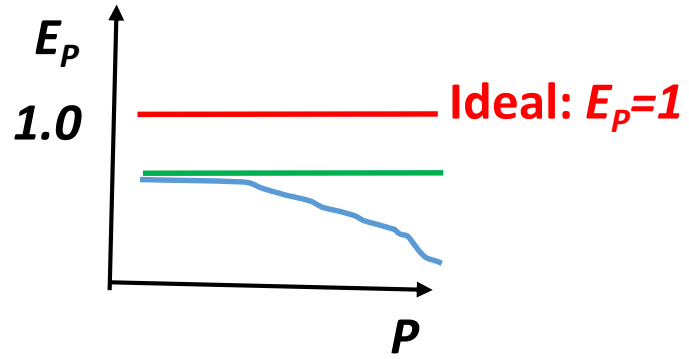
  Situação ideal: $T_P = T_1 / P \rightarrow S_P = P$
  - Difícil de se obter na prática
    - Paralelização introduz overheads que não existiam antes
  - Caso mais comum: $S_P < P$
  - Casos patológicos: $S_P > P$ (speedup *superlinear*)
    - Resultante de fatores especiais, tais como efeitos de cache, etc.

Eficiência Paralela

- Definição: $E_P = S_P / P$
  - Medida do grau de efetividade dos processadores em uso

  Situação ideal: $S_P = P \rightarrow E_P = 1.0$
  - Caso mais comum: $S_P < P \rightarrow E_P < 1.0$
  - Em geral, tanto S_P como E_P são funções de P





Lei de Amdahl

- Determina limites no speedup de programas reais
- Programa sequencial: $T_S = T_A + T_B$

  A: Parte estritamente sequencial

  B: Parte paralelizável (pode-se assumir totalmente paralelizável)

  f: fração paralelizável = $T_B / (T_A+T_B)$, $f < 1.0$
- Execução com *P* processadores: $T_P = T_A + T_B / P$
- Speedup: $S_P=T_S/T_P=(T_A+T_B)/(T_A+T_B/P)$

  com $P \rightarrow \infty$: $S_{P-->\infty}= (T_A+T_B)/T_A = 1 / (1-f)$ **$S_P(max) = 1 / (1-f)$**

Lei de Amdahl (cont.)

- Limite do Speedup: $S_P(max) = 1 / (1-f)$
  - Mesmo com número infinito de processadores !
  - Limite do speedup depende da fração paralelizável f

f	$S_P(max)$
0,5	2,0
0,8	5,0
0,9	10,0
0,95	20,0
0,99	100,0

→ **50% serial, 50% paralelizável**

***P*=∞: Speedup = 2**

→ **1% serial, 99% paralelizável**

***P*=∞: Speedup = 100**

Considerações Práticas

- Speedup máximo: $S_P(max)=1/(1-f)$ com $P \rightarrow \infty$

Porém, P=∞ é irrealizável fisicamente: logo, $S_P(max)$ é inatingível !

→ E sobre $S_P(max)/2$? Qual o valor de P necessário?

Em outras palavras, qual o valor de $P_{1/2}=Q$ tal que $S_Q=S_P(max)/2$?

$S_Q = 1/(2*(1-f))= 1/(2-2f) = 1/(2-2T_B/(T_A+T_B)) = (T_A+T_B)/2T_A$

Mas, por definição, $S_Q=(T_A+T_B)/(T_A+T_B/Q)$

Logo, $(T_A+T_B)/2T_A=(T_A+T_B)/(T_A+T_B/Q) \rightarrow T_A+T_B/Q = 2T_A \rightarrow Q=P_{1/2}=T_B/T_A$

Exemplo: $T_A=10s, T_B=90s \rightarrow f=0,9$; $S_P(max)=1/0,1=10$ com $P \rightarrow \infty$

$P_{1/2}=Q=90/10=9 \quad S_9=(T_A+T_B)/(T_A+T_B/9)=(10+90)/(10+90/9)=5$

Considerações Práticas (cont.)

Exemplo-2: $T_A=1s, T_B=99s \rightarrow f=0,99$; $S_P(max)=1/0,01=100$ com $P \rightarrow \infty$

$P_{1/2}=Q=99/1=99 \quad S_{99}=(T_A+T_B)/(T_A+T_B/99)=(1+99)/(1+99/99)=50$

→Visão concreta: $P_{1/2}$ é o número de processadores que faz com que a parte paralelizável venha a ter o mesmo peso que a parte serial

Este esquema se aplica a casos onde $T(P)=T_{Overhead}+T_{Trab.Útil}(P)$

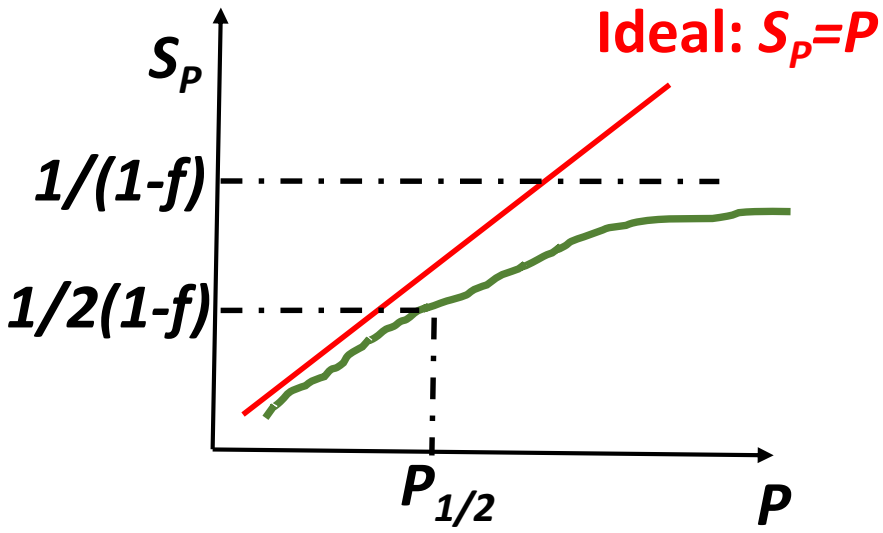
No caso atual, $T_{Overhead}=T_A$, o qual não pode ser otimizado

Idealmente, com $P \rightarrow \infty$, $T_{Trab.Útil} \rightarrow 0$, e $T \approx T_{Overhead}$

Na prática, com $P=P_{1/2}$, $T_{Overhead}=T_{Trab.Útil}$ e então $T=2.T_{Overhead}$

Considerações Práticas (cont.)

→Visão prática: $P_{1/2}$ é um número "razoável" de processadores, com o qual se consegue obter bons ganhos de desempenho a um custo ainda não-proibitivo



$P > P_{1/2}$: ganhos vão se tornando cada vez menos lucrativos!

Pipelining e $N_{1/2}$

- Ganho de desempenho do Pipeline: *(N.K)/(K+N-1)*
  - *N*: Número de tarefas, *K*: Número de estágios

    Ou, no caso de processamento vetorial com pipelining:
    - *N*: comprimento de vetores; *K*: estágios das unidades funcionais

Conforme já visto, quando $N \gg K \rightarrow Speedup_{PIPELINE} \approx K$

$N_{1/2}$: Valor de *N* tal que Speedup=*K/2*

$(N.K)/(K+N-1)=K/2 \rightarrow 2.N.K=K^2+N.K-K,\ N.K=K^2-K,\ N_{1/2}=K-1$

Exemplo: *K=10* → $N_{1/2}=9$: *T(sem-Pipeline)=10.9=90; T(com-Pipeline)=10+9-1=18*

$Speedup_{PIPELINE} = 90/18=5$

Por outro lado, se N=10.000: T(sem)=10.000.10=100.000 T(com)=10+10.000-1=10.009

$Speedup_{PIPELINE} = 100.000/10.009 = 9,99$

Lei de Gustafson

- Extensão da Lei de Amdahl – casos práticos
  - Idéia: Paralelismo é usado para resolver problemas maiores
- Novo cenário de execução: $T_S = T_A + T_B$ (caso original)

  aumentando o problema P vezes →

$$T_{S'} = T_A + P \cdot T_B$$

  - T_S : Tempo sequencial da execução do problema original
    - Duas partes: T_A (parte estritamente sequencial) e T_B (parte paralelizável)
  - $T_{S'}$: Tempo sequencial da execução do problema aumentado
    - Parte estritamente sequencial independe do tamanho do problema
    - Parte paralelizável aumenta proporcionalmente ao tamanho do problema

Lei de Gustafson (cont.)

- Novo cenário de execução: $T_S = T_A + T_B$

  aumentando o problema *P* vezes →

$T_{S'} = T_A + P \cdot T_B$

Nova execução paralela, *P* procs.: $T_{P'} = T_A + P \cdot T_B / P = T_A + T_B$

Speedup em escala: $S' = T_{S'} / T_{P'} = (T_A + P \cdot T_B) / (T_A + T_B) = (1-f)+P.f$

Speedup em escala (Gustafson): $\boxed{S' = (P-1)\, f + 1}$

- Não é mais função de *f* apenas; bem mais próximo do speedup ideal (*P*)
- Aumentando *P* (número de procs) e tamanho do problema, speedup aumenta!

Ref.: J.L.Gustafson, *Reevaluating Amdahl's Law*, Communications of the ACM, V. 31, N. 5, May 1988, pp. 532-533, doi: 10.1145/42411.42415

Lei de Gustafson (cont.)

- Implicações da Lei de Gustafson
  - Speedup em escala *S'* bem mais próximo do ideal (*P*)

f	*S' , P=2*	*S' , P=10*	*S' , P=100*
0,5	1,5	5,5	50,5
0,8	1,8	8,2	80,2
0,9	1,9	9,1	90,1
0,95	1,95	9,55	95,5
0,99	1,99	9.91	99,1

Leis de Amdahl × Gustafson

- Lei de Amdahl: escalabilidade *forte*
  - Aumento de *P* é usado para resolver o mesmo problema
  - Execução paralela leva tempo mais curto que a serial
  - *P* crescente: tamanho de problema fixo, T_P diminui
  - Exemplo: dinâmica molecular (número de átomos fixo)
- Lei de Gustafson: escalabilidade *fraca*
  - Aumento de *P* é usado para resolver problema *P* vezes maior
  - Execução paralela leva o mesmo tempo que a serial
  - *P* crescente: tamanho de problema crescente, T_P fixo
  - Exemplo: previsão de tempo (grade de maior resolução)