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

Aula 7: Medição de Desempenho

Celso L. Mendes, Stephan Stephany

LAC /INPE

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

Medição de Desempenho

- Métrica Principal: Tempo
  - Tempo de execução de uma aplicação (ou de um trecho) é o aspecto de maior interesse em termos de desempenho
  - Pode ser utilizado para a derivação de outras métricas (por exemplo, Flops/s)
  - Em geral, interesse maior no *wall-time* (ou *elapsed-time*)
    - Relevante quando apenas uma aplicação executa na CPU
  - Procedimento típico de medição:

    t1 = ler_relogio()
    <trecho a ser medido>
    t2 = ler_relogio()
    Tempo = t2 – t1

Medição de Desempenho (cont.)

- Possível exemplo trivial de uso:

```
call cpu_time(t1)
do i=1,n
        x(i) = a*y(i)
enddo
call cpu_time(t2)
tloop = t2 – t1
```

- Problemas… Quais?
  - Se *n=1.000.000* ? E se *n=1* ?
  - Como obter uma medição confiável?

Medição de Desempenho (cont.)

- Primeiro possível problema: *n=1.000.000*
  - Trecho efetivamente medido:

    *x(1) = a*y(1)*
    *x(2) = a*y(2)*
    . . .

    *x(1.000.000) = a*y(1.000.000)*
  - Elementos de *x* e de *a* podem não estar na memória ainda: page-fault!
  - Tempo medido seria dominado pelo tratamento dos page-faults
  - Caso o trecho seja repetido (com os arrays já em memória), o novo tempo medido da segunda vez poderia ser muito menor!
  - Possível solução: executar duas vezes, medir apenas a segunda passada

Medição de Desempenho (cont.)

- Segundo possível problema: *n=1.000.000*
  - Alguns elementos de *x* e de *a* podem estar ou não em cache
  - Caso todos os elementos estejam em cache: duração curta
  - Caso a maioria dos elementos não esteja: duração longa
  - Tempos medidos em múltiplas execuções do código pode variar!
    - Duração medida seria dominada pelo tratamento dos cache-misses
  - Possível solução: executar múltiplas vezes, obtendo a média dos tempos medidos para cada vez (e talvez mínimo e máximo)

Medição de Desempenho (cont.)

- Terceiro possível problema: compilador !
  - Suponha que, por algum motivo, os valores de a(i) não sejam mais usados no programa
    - Compilador “otimizante” pode eliminar totalmente o código do loop!
    - Improvável neste exemplo trivial, mas razoavelmente comum em alguns benchmarks simples
  - Possível solução: montar o código tal que todos os valores produzidos no trecho medido sejam utilizados posteriormente (mesmo que após o trecho medido)

Medição de Desempenho (cont.)

- Quarto possível problema: *n=1*
  - Trecho efetivamente medido:

```
do i=1,1
        x(1) = a*y(1)
enddo
```

  - Código de máquina gerado: poucas instruções (menos que 10)
  - Supondo operandos em registros ou na cache L1: duração ≈ 10 ciclos
  - Num processador a 2 GHz: ciclo = 0.5 ηs → duração ≈ 5 ηs
  - ➢ Logo, é preciso ter um relógio capaz de medir trechos da ordem de ηs
    - Incomum na prática!
  - Possível solução: executar o trecho múltiplas vezes e medir duração total, dividindo-a então pelo número de vezes executadas (Quantas vezes? Ver próxima aula)

Outros Problemas

- Medição de tempo com rotinas internas a outras
  - Exemplo: *main()* → $sub_1()$ → $sub_2()$ → … → sub_N
  - Duração *inclusiva* de sub_K: $T_I(sub_K) = T(sub_K, sub_{K+1}, \ldots, sub_N)$
    - Inclui o tempo em $sub_k()$ e em todas as rotinas chamadas por $sub_k()$
  - Duração *exclusiva* de sub_K: $T_E(sub_K) = T_I(sub_K) - T_I(sub_{K+1})$
    - Exclui o tempo de todas as rotinas chamadas por $sub_k()$
  - Implementação de relógios para medir durações de cada nível:
    - Pode ser feita utilizando uma pilha
      - Chamada à rotina: lê relógio, empilha valor;
      - Retorno da rotina: lê relógio, desempilha valor, calcula duração (diferença)

Outros Problemas (cont.)

- Sistemas com múltiplos nós
  - Caso comum: uma imagem do Sistema Operacional em cada nó
  - Relógios independents em cada nó
  - Logo, valores de tempo lidos em um nó podem ser inconsistentes com os obtidos em outro nó
    - Problemático para correlacionar/medir eventos em nós distintos

Exemplo: P_A P_B (P_A,P_B em nós distintos)

T1=le_relogio()

MPI_Send(…, P_B, T1, …) → MPI_Recv(…, T1, …)

T2=le_relogio()

T_{MSG} = T2 – T1 → ERRADO !

Outros Problemas (cont.)

- Sistemas com múltiplos nós (cont.)
  - Abordagem típica: medir T_{MSG} através de *ping-pong*

Exemplo:

P_A		P_B
T1=le_relogio()		
MPI_Send(…, P_B, …)	→	MPI_Recv(…)
MPI_Recv(…)	←	MPI_Send(…, P_A, …)
T2=le_relogio()		

$T_{12} = T2 - T1$

$T_{MSG} = T_{12} / 2$ → Correto ! T1 e T2 obtidos no mesmo nó

Outros Problemas (cont.)

- *Traces* de Execução
  - Relatório detalhado de eventos ocorridos em cada processador
  - Trace = sequência de registros: {tipo, timestamp, dados}
  - Formato exato pode variar, mas deve seguir alguma descrição, para permitir pós-processamento

Exemplo: P_A

```
sub1(buf1, n)
MPI_Send( …, P_B, buf1, …)
MPI_Recv(…)
A=B+C
sub2(buf2,m,n)
MPI_Send( …, P_C, buf2, …)
. . .
```

Trace de P_A:

{call,t1,sub1,buf1,n}
{return,t2,sub1}
{s_begin,t3,P_B,nbytes}
{s_end,t4}
{r_begin,t5}
{r_end,t6,P_X,nbytes}
{call,t7,sub2,buf2,m,n}
{return,t8,sub2}
{s_begin,t9,P_C,nbytes}
{s_end,t10}

Outros Problemas (cont.)

- *Traces* de Execução (cont.)
  - Em geral, um *trace* obtido em cada processador
    - Podem ser combinados num único arquivo, por conveniência
  - Obtenção dos tempos t_i : instrumentação do código fonte
    - Pode ser manual ou automática
  - *t1, t2, …, tN* são obtidos localmente em P_A
    - Problemático correlacionar com tempos de P_X, X≠A
    - Se for necessário correlacionar, "ajustar" relógios no início
      - Ajustar também os registros dos traces: garantir *causalidade* (uma mensagem não pode ter T_{recv} menor que T_{send})
  - Diversas estatísticas da execução podem ser calculadas a partir do processamento dos traces