Análise de Desempenho em Programas Paralelos

Tópicos:
- Introdução
- Aquisição de Dados de Desempenho
- Processamento/Visualiz.de Dados de Desempenho
- Ferramentas de Análise de Desempenho

Referência: Foster, Ian *Designing and Building Parallel Programs* Addison-Wesley, 1995

(versão online em http://www.mcs.anl.gov/dbpp)

Introdução

Fases da Análise de Desempenho de um Programa:
- Aquisição de Dados de Desempenho
  - Feita durante a execução do programa
  - Dados de desempenho podem ser armazenados num arquivo
- Transformação ou Redução dos Dados
  - Redução da massa total de dados adquiridos
  - Pode ser feita durante a execução ou a posteriori
- Visualização dos Dados
  - Exibir os dados de desempenho, para análise visual
  - Informar ao programador os gargalos de desempenho

Introdução (cont.)

Fatores a considerar:
- Precisão dos dados adquiridos:
  - Depende dos recursos disponíveis para a aquisição
- Simplicidade das ferramentas utilizadas:
  - Ferramentas automáticas são mais convenientes
- Flexibilidade das ferramentas utilizadas:
  - Ferramentas que podem ser expandidas são mais úteis
- Perturbação causadas no experimento:
  - Sempre há um overhead se a aquisição for via software
  - Idealmente, aquisição deveria ser por hardware
- Modelo de abstração:
  - Dados de desempenho adquiridos devem ser facilmente relacionados com o programa-fonte original

Aquisição de Dados de Desempenho

Principais Técnicas de Aquisição:
- Perfil de Execução (*profiling*)
- Contadores
- Rastreio da Execução (*tracing*)

- Perfil de Execução:
  - Mostra o tempo gasto em diferentes partes do programa
  - Implementado através de amostragens periódicas do contador de programa
  - Técnica originada em programas seqüenciais

Aquisição de Dados de Desempenho (cont.)

Exemplo (programa seqüencial):

```
{aldebaran}/tmp/celso% cc -p -o serial serial.c
{aldebaran}/tmp/celso% serial
Enter a, b, and n
0 2 100000
With n = 100000 trapezoids, our estimate
of the integral from 0.000000 to 2.000000 = 2.671024
{aldebaran}/tmp/celso% /usr/ccs/bin/prof
 %Time Seconds Cumsecs  #Calls   msec/call  Name
 66.7    0.02    0.02        1     20.      main
 33.3    0.01    0.03   100001      0.0001  f
```


Aquisição de Dados de Desempenho (cont.)

Vantagens do uso de perfis:
- Podem ser obtidos de forma automática
- Fornecem rapidamente uma visão “geral” da execução

Desvantagens do uso de perfis:
- Não capturam variações temporais durante a execução
- Eventos periódicos podem acabar sendo mascarados

Problemas adicionais no caso de programa paralelo:
- Em princípio, 1 arquivo de perfil para cada processador
- Problema de desempenho em um processador pode ser causado pelo comportamento de outro processador!

Aquisição de Dados de Desempenho (cont.)

- Contadores :
  - Registros que gravam o número de ocorrências de um certo fenômeno
  - Dois tipos principais:
    - Contadores de Eventos: contam eventos específicos
    - Contadores de Intervalo: contam número de ciclos de relógio entre dois eventos
  - Podem ser implementados manualmente, ou por código gerado pelo compilador, automaticamente
  - Exemplos: Número de mensagens enviadas, duração de um loop, número de chamadas a uma sub-sotina, etc.

Aquisição de Dados de Desempenho (cont.)

Algumas observações:

- Perfis e Contadores fornecem recursos complementares (Ex: Perfil → Duração total gasta numa sub-rotina, Contad. → Número de chamadas ∴ duração média)
- Como em perfis, contadores também ignoram variacões temporais ao longo da execução do programa
- Como em perfis, é difícil relacionar contagens obtidas em processadores distintos, no caso de prog. paralelos
- Quase todos os processadores recentes oferecem recursos de suporte em hardware

Aquisição de Dados de Desempenho (cont.)

- Rastreio da Execução (*tracing*):
  - Visão detalhada da execução do programa
  - Na execução, a cada ocorrência do evento de interesse é gerado um registro, com o valor corrente do *clock*
  - Outras informações possíveis no registro: tipo de evento, número do processador, duração, etc.
  - *Trace*: seqüência de registros gerados na execução; pode ser armazenado num arquivo
  - Programa original + código p/ geração de registros = programa instrumentado
  - Instrumentação pode ser feita pelo programador ou automaticamente

Aquisição de Dados de Desempenho (cont.)

Exemplo de arquivo de *trace*:

Trace Record	Description
11 0 1558 2 0 1078	Timer data
8 0 2287 2 1 1 8	Receive
11 0 2237 2 0 1078	Timer data
11 0 2500 2 0 1841	Timer data
4 0 8186 2 8 2 12	Send
11 0 8186 2 0 1841	Timer data
11 0 8502 2 0 1858	Timer data
4 0 8872 2 4 2 12	Send
11 0 8872 2 0 1858	Timer data

Tipo Proc Clock …

Processamento e Visualização de Dados de Desempenho

- Objetivo:
  - A partir dos dados de desempenho adquiridos, gerar dados mais simples e exibí-los ao programador
- Apresentação Típica:
  - Forma gráfica
- Com Perfis e Contagens:
  - Forma gráfica mais conveniente: histograma
  - Histogramas podem ser 1-D ou 2-D (ou até 3-D!)
    - Ex 1-D: Duração total × Sub-rotina
    - Ex 2-D: Duração total × {Sub-rotina, Processador}

Processamento e Visualização de Dados de Desempenho (cont.)

- Com *Traces*:
  - Histogramas 1-D/2-D/3-D, após filtragem dos dados
  - *Gantt charts* (carta de barras horizontais):
    - Eixo horizontal representa o tempo ao longo da execução
    - Cada barra horizontal corresponde a um processador
    - Forma da barra (largura, cor, etc.) representa o estado do proc. Obs: Assume-se que os *clocks* dos proc. estão sincronizados
    - Conexões entre barras horizontais representam mensagens
    - Também conhecidas como diagrama espaço/tempo
    - Relações causa-efeito ficam evidentes no diagrama
    - Não podem ser usadas c/ número alto de processadores

Ferramentas de Análise de Desempenho

- Ferramentas de domínio público:
  - Paragraph:
    - Gant charts, histogramas, etc.
    - Feita para exibir *traces* gerados por PICL
  - Upshot
    - Feita especialmente para programas com MPI
    - Maior flexibilidade de uso que Paragraph
  - Pablo
    - Mais poderosa (e mais complexa); dados em formato SDDF
    - Baseada num editor gráfico para processamanto de *traces*
    - Versão atual: SvPablo (ligação com o código-fonte)