Arquitecturas Paralelas I

Computação Paralela em Larga Escala

LESI - 4º Ano

Medição e Optimização de Desempenho

João Luís Ferreira Sobral
Departamento do Informática
Universidade do Minho

Novembro 2003

Desempenho de Aplicações Paralelas/Distribuídas

- A análise do desempenho da aplicação, através de *modelos de desempenho*, possibilita a comparação de algoritmos, análise de escalabilidade e a identificação de estrangulamentos nas aplicações, antes de investir um esforço substancial na implementação.

Qual a definição de desempenho?

- Existem múltiplas alternativas: tempo de execução, eficiência, escalabilidade, requisitos de memória, débito, latência, custos de projecto, desenvolvimento e implementação, requisitos memória e de hardware em geral, portabilidade, potencial de reutilização, etc.
- Geralmente o peso de cada um dos elementos anteriores varia de aplicação para aplicação.

Lei de Amdahl

- Se um algoritmo paralelo tiver um componente sequencial, esse componente irá limitar os ganhos proporcionados por uma execução paralela: se s for a fracção sequencial do algoritmo então o ganho máximo que pode ser obtido com esse algoritmo é $1/s$.
- Esta lei reforça a ideia de que se devem preferir os algoritmos com um grau superior de paralelismo: *think parallel*

Modelo de desempenho

- Um modelo de desempenho deve explicar as observações disponíveis e prever circunstâncias futuras, sendo geralmente definido como uma função da dimensão do problema, do número de processadores, do número de tarefas, etc.
- O *tempo de execução* de um programa pode ser definido como o tempo que decorre desde que o primeiro processador inicia a execução até o último processador terminar. O tempo de execução pode ser decomposto no tempo de computação, de comunicação e no tempo de ócio:

$$T_{exec} = T_{comp} + T_{comn} + T_{ócio}$$

- O *tempo de computação* é o tempo despendido na computação, excluindo o tempo de comunicação e de ócio. Quando existe uma versão sequencial do algoritmo esta pode ser utilizada para estimar T_{comp}.

Desempenho de Aplicações Paralelas/Distribuídas

Modelo de desempenho (continuação)

- O *tempo de comunicação* é o tempo que o algoritmo despende a enviar e receber mensagens. Podendo ser calculado através da latência (t_s) e do débito da comunicação ($1/t_w$). O tempo necessário para a transmissão de uma mensagem de L palavras pode ser aproximado por:

$$T_{mens} = t_s + t_w L$$

Adicionalmente, se existir partilha do débito da comunicação pode ser introduzido um factor que modela essa competição, por exemplo, se o débito for dividido pelo número de processadores temos:

$$T_{mens} = t_s + t_w L(1/P)$$

t_s e t_w podem ser obtidos experimentalmente, através de um teste de ping-pong e de uma regressão linear

- O *tempo de ócio* é de medição bastante mais complexa, uma vez que depende da ordem pela qual são realizadas as operações. O tempo de ócio surge quando um processador fica sem tarefas, podendo ser minimizado com uma distribuição de carga adequada ou sobrepondo a computação com a comunicação.

- Exemplo: tempo de execução de cada iteração no método de Jacobi, assumindo uma matriz de NxN em P processadores e uma partição em colunas, N/P colunas por processador.

T_{comp} = operações por ponto *x* nº de pontos por processador *x* t_c
$= 6 \times (N \times N/P) \times t_c$ (t_c = tempo de uma operação)
$= 6t_cN^2/P$

T_{comn} = mensagens por ponto *x* tempo para cada mensagem
$= 2 \times (t_s + t_wN)$

$T_{ócio} = 0$, uma vez que este problema é balanceado

$T_{exec} = T_{comp} + T_{comn} + T_{ócio}$
$= 6t_cN^2/P + 2t_s + 2t_wN$
$= O(N^2/P+N)$

Desempenho de Aplicações Paralelas/Distribuídas

Modelo de desempenho (continuação)

- O tempo de execução pode não ser o mais indicado para avaliar o desempenho. O ganho e a eficiência são duas medidas relacionadas.

  O *ganho* indica o factor de redução do tempo de execução proporcionado por P processadores, sendo dado pelo quociente entre o tempo de execução do melhor algoritmo sequencial e o tempo de execução do algoritmo paralelo.

  $$G = Tseq / Tpar,$$

  A *eficiência* indica a facção de tempo em que os processadores efectuam trabalho útil:

  $$E = Tseq / (P \times Tpar)$$

- Exemplo para o caso do método de Jacobi:

$$E = \frac{6t_c N^2}{6t_c N^2 + 2Pt_s + 2Pt_w N}$$

Análise de escalabilidade

- As expressões do tempo de execução e da eficiência podem ser utilizadas para efectuar uma análise qualitativa do desempenho. Por exemplo, podem ser efectuadas as seguintes observações relativamente ao método de Jacobi:

  1. O tempo de execução diminui com o aumento do número de processadores, mas está limitado pelo tempo necessário para a troca de duas colunas da matriz entre processadores;

  2. O tempo de execução aumenta com N, t*c*, t*s* e t*w*;

  3. A eficiência diminui com aumentos de P, t*s* e t*w*;

  4. A eficiência aumenta com aumentos de N e de t*c*.

- *Escalabilidade de problemas de dimensão fixa*. Analisa as variações de T*exec* e de *E* em função do número de processadores. Em geral, *E* decresce de forma monótona, podendo T*exec* vir a crescer, se o modelo de desempenho incluir uma parcela proporcional a uma potência positiva de P.

Desempenho de Aplicações Paralelas/Distribuídas

Análise de escalabilidade (continuação)

- *Escalabilidade de problemas de dimensão variável*. Por vezes pretende-se resolver problemas de maior dimensão, mantendo a mesma eficiência, ou mantendo o mesmo tempo de execução ou utilizando a memória disponível. A *função de isoeficiência* indica qual o aumento necessário na dimensão do problema para que a eficiência permaneça constante, quando se aumenta o número de processadores.

Estudo experimental e avaliação das implementações

- A computação paralela tem uma forte componente experimental, uma vez que geralmente os problemas são demasiados complexos para que uma realização apenas baseada em teoria possam ser eficiente. Adicionalmente, os modelos de desempenho devem ser calibrados com resultados experimentais, por exemplo para determinar t*c*.

- Um dos principais problemas do estudo experimental é garantir que são obtidos resultados precisos e reprodutíveis. Nomeadamente, devem ser realizadas várias medições e deve-se verificar se o relógio tem resolução e precisão suficientes. Geralmente os resultados não podem variar mais do que um pequeno montante: 2 – 3%.

- Os vários tempos de execução podem ser obtidos recolhendo o *perfil de execução* (p.ex., contadores do número de mensagens, volume de informação transmitido e dos váios tempos de execução), o que pode ser efectuado através da inserção de código específico ou de ferramentas próprias. Note-se que existe sempre alguma intrusão provocada por este código ou ferramentas.

- Por vezes ocorrem *anomalias* no ganho, podem este ser superlinear (i.é., superior ao número de processadores), o que pode ser derivado do efeito das *caches*.

Técnicas de avaliação do perfil de execução das aplicações (*profiling*)

- **Sondagem**: A aplicação é periodicamente interrompida para recolher informação estatística sobre a execução

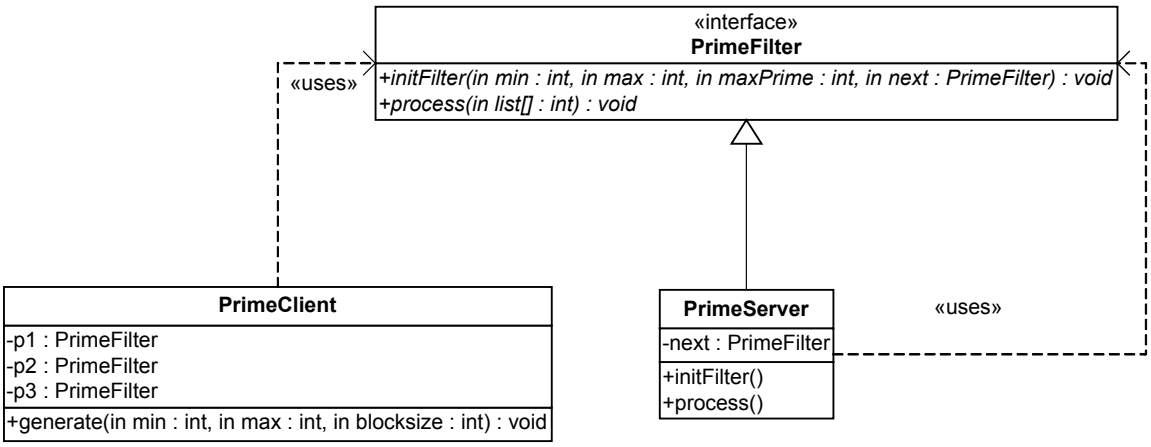
- **Instrumentação**: É introduzido código (pelo programador ou pelas ferramentas) na aplicação para recolha de informação sobre os vários eventos.

- A técnica de instrumentação tende a produzir melhores resultados mas também tende a causar mais interferência nos tempos de execução

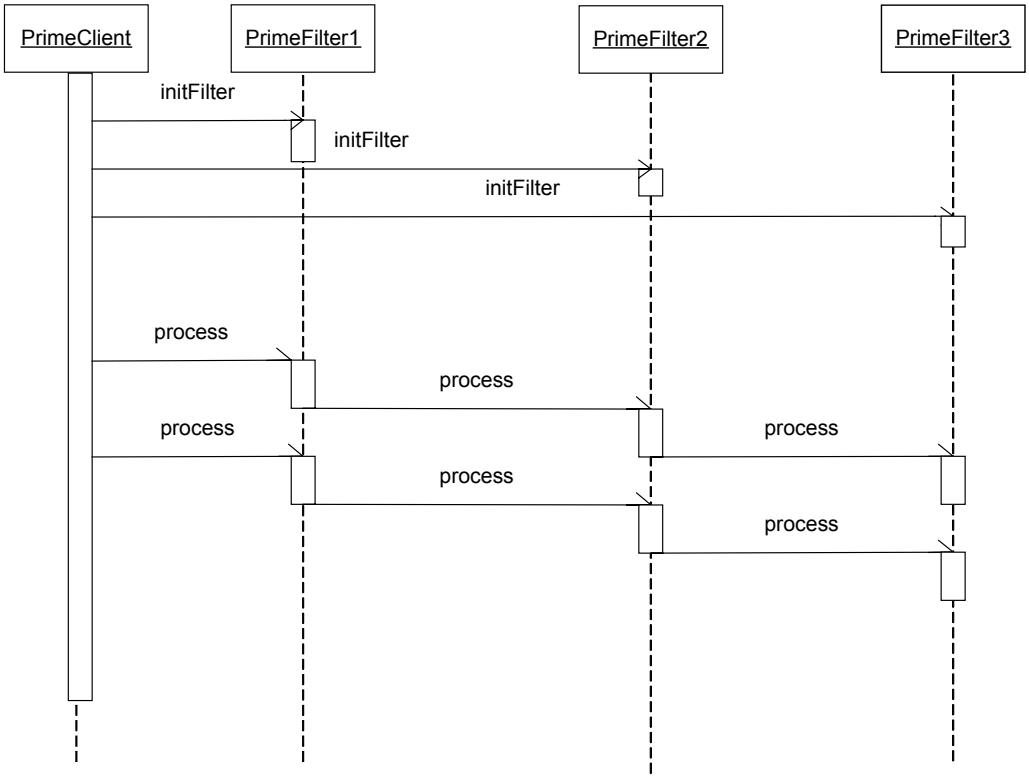
Desempenho de Aplicações Paralelas/Distribuídas

Caso de Estudo: Crivo de Eratosthenes (cálculo dos números primos)

- Diagrama de classes:



- Diagrama de interacção



Desempenho de Aplicações Paralelas/Distribuídas

Exercícios

1 Transforme o código sequencial do crivo em código paralelo através da introdução de invocações assíncronas de métodos entre os filtros. Compare com os resultados da versão sequencial.

2 Altere o código do crivo por forma a que seja possível determinar o tempo que demora a execução (sugestão: utilize o último filtro do crivo para contabilizar o tempo de execução).

3 Corra a aplicação em vários processadores e verifique os tempos de execução para as seguintes condições (utilize 10 000 000 de números):

- 3.1 Todos os filtros no pe0
- 3.2 Todos os filtros no pe12
- 3.3 Um filtro em cada máquina (pe0, pe10 e pe11)
- 3.4 Um filtro em cada máquina (pe10, pe11 e pe12) e o gerador em pe0

4 Altere o número de primos atribuído a cada filtro (por. exemplo, utilize 500, 100 e 3163 primos para os três filtros)

5 Altere o número de primos enviado em cada mensagem (experimente as seguintes quantidades: 5 000, 10 000, 50 000, 200 000, 1 000 000)