UNIVERSIDADE DE COIMBRA FACULDADE DE CIÊNCIAS E TECNOLOGIA ***Departamento de Engenharia Informática***	**Trabalho Prático N.º 1** **Computação de Alto Desempenho** **2009/10 – 2º Semestre** MEI **Prazo: 14-4-2010**
<u>Nota</u>: A fraude denota uma grave falta de ética e constitui um comportamento não admissível num estudante do ensino superior e futuro profissional. Qualquer tentativa de fraude pode levar à reprovação na disciplina tanto do facilitador como do prevaricador.	

Programação com Memória Partilhada

Objetivos do Trabalho

Criação e otimização de um programa paralelo programado com memória partilhada. Estudo do respetivo desempenho.

Descrição do Trabalho

A avaliação de um polinómio consiste em atribuir um valor concreto às variáveis desse polinómio. Por exemplo, dado

$$p(x) = x^4 + 8x^3 + 2x^2 + x + 1$$

a avaliação para x=3, resulta em

$$p(3) = 3^4 + 8\cdot 3^3 + 2\cdot 3^2 + 3 + 1 = 319$$

A forma mais simples de calcular o valor da atribuição, que designaremos de "básica", passa por calcular sequencialmente todos os termos (monómios) envolvidos, i.e., primeiro o 3^4, ao qual adicionamos depois o resultado de $8\cdot 3^3$, etc. Embora simples, este método é muito ineficiente, porque repete múltiplas vezes os mesmos cálculos (3^4, 3^3, etc.). De forma a evitarmos este problema, podemos recorrer a um método muito mais eficiente, conhecido como "método de Horner", que reorganiza os termos do polinómio:

$$\begin{aligned} p(x) &= x^4 + 8x^3 + 2x^2 + x + 1 \\ &= \left(x^3 + 8x^2 + 2x^1 + 1\right)x + 1 \\ &= \ldots \\ &= \left(\left(\left(x + 8\right)x + 2\right)x + 1\right)x + 1 \end{aligned}$$

Para um polinómio de grau n, este método requer no máximo n adições e n-1 multiplicações. Pelo contrário, o método básico requer até $(n^2+n)/2$ multiplicações e n adições, embora este número possa ser reduzido para $2n$-1 multiplicações, se calcularmos as potências iterativamente (i.e., x, depois x^2, depois x^3, etc.).

Tarefas

Neste trabalho, os alunos terão de implementar um (ou mais) algoritmo(s) paralelo(s) para avaliação de polinómios. Para tal, poderão basear-se numa variante do *prefix sum problem* (também conhecida por *scan*, *prefix reduction*, ou soma parcial). A ideia é multiplicar determinados elementos de um vetor de acordo com o padrão da Figura 1. Estas multiplicações ocorrem em O(log n) passos síncronos.

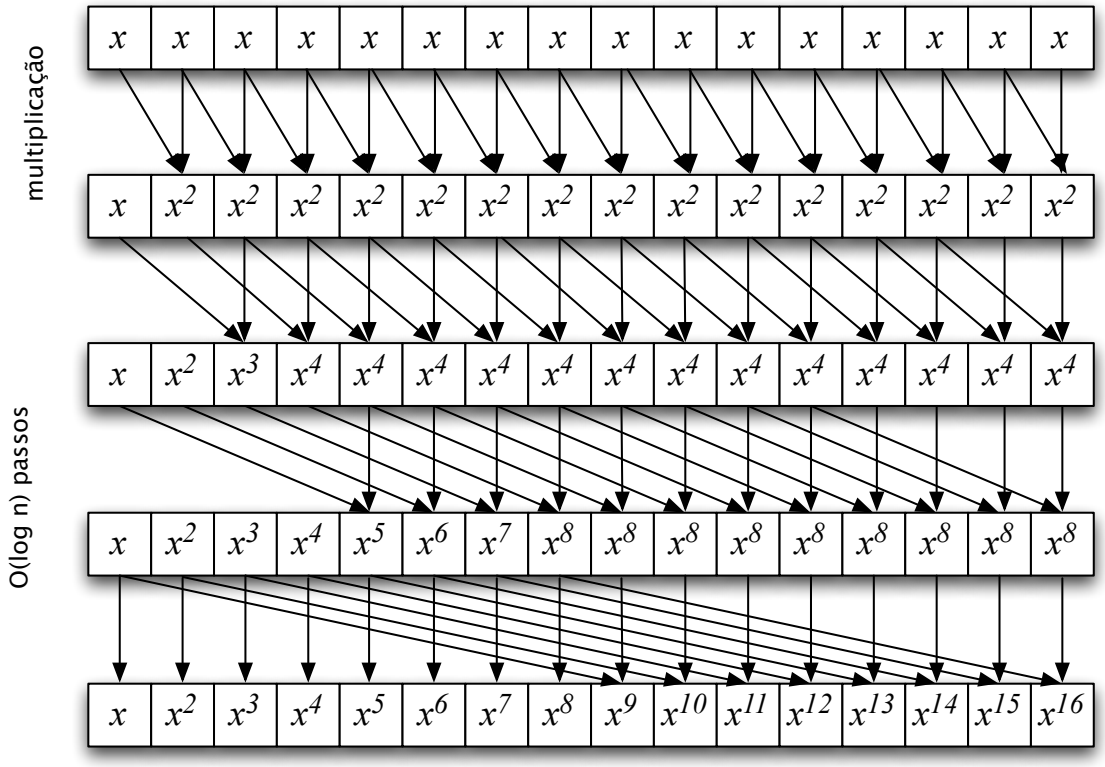

Figura 1 – Variante do *prefix sum problem*

Os alunos terão de criar uma implementação deste ou de outro algoritmo paralelo, utilizando, para isso, uma tecnologia à escolha, de entre as seguintes:

- Java
- Pthreads
- OpenMP
- Cilk
- SSE

É permitido combinar estas tecnologias para otimizar os resultados.

Código Disponível e a Entregar

Os alunos terão acesso a código escrito em C, não só para fazer a leitura dos polinómios e valores a avaliar, mas também para escrever os resultados para a saída padrão[1]. A Figura 2 representa o formato dos dados de entrada. O polinómio em questão é $p(x) = x^4 + 8x^3 + 2x^2 + x + 1$ e terá de ser avaliado para 3 valores: x=3, x=2 e x=11. Os alunos devem notar que tanto o grau do polinómio (4), como o número de pontos a avaliar (3) são indicados no ficheiro.

[1] A leitura e a escrita dos valores processa-se na íntegra para e a partir da memória, o que poderá constituir um problema para polinómios excecionalmente grandes. A resolução deste eventual problema ficará a cargo dos alunos.

```
4 1 8 2 1 1
3
3
2
11
```

Figura 2 – Formato de entrada de dados

O resultado deverá ser a avaliação do polinómio para os múltiplos valores dados, de acordo com a função evaluate() que é dada. Não deve ser adicionado qualquer texto ao formato desta função. Cada programa executável deve executar um algoritmo diferente. Se os alunos criarem três versões (p.ex., básico, Horner e paralelo), devem ter três executáveis diferentes e devem listar estes nomes no relatório.

Os alunos não deverão alterar os formatos de entrada, ou de saída.

Os alunos que quiserem utilizar o Java para realizar o trabalho, terão de replicar as funcionalidades já disponíveis em C.

Entre os ficheiros disponibilizados, encontram-se vários exemplos de polinómios, bem como um script de python, que permite gerar novos ficheiros de polinómios. Em Unix pode-se usar a seguinte linha de comando, para gerar um polinómio de grau 400 e com 200 valores para avaliar:

```
echo 400 200 | python generate-inputs.py > nome-ficheiro
```

Todos os coeficientes e valores a avaliar são inteiros e estão dentro de certos limites, sendo trivial alterar esses comportamentos, se for necessário criar polinómios diferentes.

É importante referir, que os resultados dos diferentes algoritmos (por exemplo, básico e Horner) podem diferir, para os **mesmos** valores de entrada. Isto resulta do facto de as operações de vírgula flutuante não serem nem associativas nem distributivas.

Os alunos devem entregar uma **Makefile**, ou equivalente, que permita compilar o código fonte e construir os executáveis de forma automática.

Avaliação

A avaliação do trabalho será feita com base nos seguintes critérios:

- Correção. Os alunos deverão garantir que o seu algoritmo funciona corretamente. Note-se que os trabalhos serão testados com alguns polinómios específicos para verificar este item.
- Desempenho. Os alunos deverão preocupar-se em minimizar o tempo de execução do seu algoritmo. Os desempenhos dos diferentes trabalhos serão comparados contra uma implementação base (que será diferente conforme se trate de Java ou de C).
- A análise do desempenho será um dos aspectos mais fundamentais na avaliação do trabalho. Serão valorizados os trabalhos que incluam comparações de desempenho de mais do que uma implementação ou de otimizações efetuadas a uma implementação sequencial. É importante realçar que a validade das otimizações a realizar deverá ser demonstrada com medições de desempenho. Para além disto é importante manter a clareza do código de forma a que o programa seja de fácil compreensão.

Relatório

O relatório deverá ser o mais curto possível, não deve incluir informação redundante (que já esteja suficientemente coberta no enunciado do trabalho, por exemplo) e deverá ser bem estruturado.

Um dos aspetos mais importantes do relatório reside na avaliação do desempenho do algoritmo paralelo. Quais os ganhos de desempenho da solução paralela relativamente ao grau do polinómio, ou relativamente ao n.º de threads, por exemplo.

Os alunos não se devem esquecer de indicar os nomes dos programas que criarem, para além da linha de comando necessária para os correr.

Prazo de Entrega

Ver cabeçalho. No total dos dois trabalhos os alunos podem atrasar-se 4 dias.

Entrega do Trabalho

O código do trabalho, deverá ser guardado num ficheiro ZIP. Dentro desse ficheiro ZIP deverá ainda juntar um ficheiro **README.TXT** com TODA A INFORMAÇÃO NECESSÁRIA PARA O DOCENTE EXECUTAR, CONFIGURAR E TESTAR O SEU TRABALHO SEM A PRESENÇA DO(S) ALUNO(S). Trabalhos que não contenham esse ficheiro README com todas as instruções necessárias não serão avaliados. Trabalhos que não executem corretamente também não serão avaliados.

Deverá fazer o upload desse ficheiro ZIP no moodle: http://classes.dei.uc.pt

O relatório poderá ser incluído em formato PDF dentro do ficheiro ZIP. No entanto, é obrigatório entregar o relatório no cacifo de correio do docente da cadeira, até no máximo 1 dia útil após a entrega do trabalho. A chave de inscrição no moodle para CAD é cad$2010.

Bom Trabalho!