Módulo 1.1

Introdução ao caso de estudo / modelo de desempenho do (C)PU

Objetivos:
- introdução ao caso de estudo multiplicação de matrizes: C = A x B
- introdução ao modelo de desempenho do (C)PU: Texe = #I x CPI x Tcc

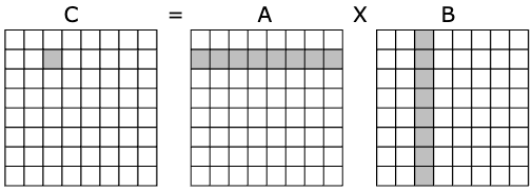
Introdução:
Este é o primeiro módulo (1.1) de um conjunto de 3 módulos (1.1 a 1.3) que utilizam um algoritmo de multiplicação de matrizes para ilustrar o impacto das várias grandezas (#I, CPI) no tempo de execução de um programa. Uma multiplicação de matrizes pode ser implementada de várias formas. Neste conjunto de módulos vamos utilizar, como base, uma implementação com três ciclos aninhados, começando pela variante mais comum, designada por ijk (posteriormente será identificada por DOT), para matrizes A, B e C quadradas com dimensão NxN:

```
for(int i=0; i<N; i++)
  for(int j=0; j<N; j++)
    for(int k=0; k<N; k++)
      C[i][j] += A[i][k] * B[k][j]; // nota: assume matriz C inicializada com 0s
```

Numa execução sequencial qualquer ordem dos ciclos i, j e k é correta, mas essa ordem origina diferenças significativas de desempenho que serão analisadas nas próximas aulas. Esta primeira variante (ijk) será designada por DOT, pelo facto de cada elemento da matriz C resultar do produto interno de uma linha da matriz A com uma coluna da matriz B, tal como ilustram a expressão/algoritmo e a figura seguintes:

$$C_{ij} = DOT_{linha\_A_i, coluna\_B_j} = \sum_{k=0}^{n-1}(A_{ik} * B_{kj})$$

```
Para cada linha de A
  Para cada coluna de B
    C linha,coluna = DOT linha de A,coluna de B
```



Recursos disponíveis:
Nesta disciplina irá ser usado o cluster computacional SeARCH, mais especificamente dois nós com 20 núcleos computacionais, na partição "`-cpar`". Para executar o código deste módulo (e dos seguintes), deve aceder à máquina `s7edu.di.uminho.pt` por `ssh`, usando as credenciais recebidas por email. O código da multiplicação de matrizes pode ser executado num dos nós disponíveis com o seguinte comando: `srun --partition=cpar <<caminho>>/gemm <<N>> 1`, onde N é o tamanho da matriz (número de linhas).
Na página de *elearning* da disciplina está disponível uma implementação da multiplicação de matrizes (versão 1 – gemm1) que deverá ser usada nesta aula. Este ficheiro deve ser copiado (com `scp`) para a máquina `s7edu`. A edição e compilação do código deve ser realizada na máquina `s7edu`, mas a execução deverá ser realizada num nó de computação da partição "`-cpar`", usando o comando `srun` já referido. Para utilizar a biblioteca PAPI deve carregar o módulo correspondente com `module load papi/5.4.1` (basta fazer na máquina `s7edu`).

Exercício 1: Estimativa de desempenho

a) Utilizando a notação O-grande, qual a complexidade deste algoritmo? De que forma isso se vai traduzir no tempo de execução do algoritmo com a duplicação do valor de N? Qual a componente do modelo de desempenho que deverá ser mais afetada com esse aumento de N (#I, CPI ou Tcc)?

b) Visualize o código *assembly* gerado para a função `gemm` (`gcc -S gemm.c`) e estime o número de instruções que serão executadas em função de N (sugestão: basta olhar para o *assembly* do ciclo mais interior!). Calcule o ganho esperado se for gerado código otimizado (estime visualizando o *assembly* gerado com `gcc -O2 -S gemm.c`)?

Exercício 2: Medição

a) O código disponibilizado para esta sessão utiliza a biblioteca PAPI (https://icl.utk.edu/papi/) para obter o número de instruções executadas (#I, PAPI_TOT_INS), o número total de ciclos (#CC, PAPI_TOT_CYC) e o tempo de execução da multiplicação de matrizes. Execute o programa `gemm` para matrizes com tamanhos de 128x128, 256x256 e 512x512 e anote os valores no obtidos na tabela seguinte (basta compilar usando a `makefile` fornecida e executar o comando `srun` passando o N correspondente na linha de comandos do programa `gemm`, ou seja, 128, 256 e 512).
Compare o valor do #I obtido com a estimativa que fez para cada tamanho. Calcule também o aumento efetivo do #I sempre que o N duplica.

	linhas	Tempo medido (usec)	#CC	#I	#I estimado	CPI calculado
	128					
	256					
	512					

b) Complete a tabela calculando o CPI médio para estes três tamanhos de matrizes. Explique porque que é que esse valor é menor do que 1. Como varia o valor do CPI em função do tamanho da matriz? Qual o impacto desta variação do CPI no tempo de execução?

Exercício 3: Otimização

a) Repita o exercício anterior, compilando agora com otimização (basta alterar `-O0` para `-O2` na `makefile`; fazer `make clean` e `make`). Houve ganhos no tempo de execução relativamente ao exercício anterior? Qual a componente do modelo de desempenho principal responsável por esse ganho? O ganho obtido corresponde ao esperado (calculado em 1b)?

	linhas	Tempo medido (usec)	#CC	#I	#I estimado	CPI calculado
	128					
	256					
	512					

b) Como varia o CPI entre as versões com e sem otimização. Qual a explicação? Qual o impacto no tempo de execução.