Lab Guide 0

Support material for this preparatory session

Objectives:

- Write a basic matrix multiplication code in C
- Get accustomed with the lab session environment (gcc compiler, Linux, perf tool, remote cluster access)

1. Simple Matrix Multiplication in C (see annex 1 if you need help)

Write a simple C code to implement a matrix multiplication. Assume that all matrices are of equal size and rectangular.

2. Performance evaluation with the Linux perf tool (see annex 2 and 3 if you need help to install the perf tool or to execute the matrix case study on the University cluster)

To obtain the application profile with `perf`, run the application with `perf record ./a.out` to sample the execution data at fixed time intervals (4000 samples per second, by default). The profile is written into a file named `perf.data`. Use `perf report` to generate a [flat] view profile with the application hotspots. In simple cases, these two steps can be replaced by `perf stat [-e XXX] ./a.out` which gets the performance metrics over the entire program (the `-e XXX` option can be used to specify a specific set of metrics, see `perf list` for a complete list of available metrics on your machine).

To get a better profile and more accuracy, the code (C program) should be compiled with `-g -fno-omit-frame-pointer`.

Annex 1 - Matrix Multiplication Algorithms (in Portuguese)

(see https://en.wikipedia.org/wiki/Matrix_multiplication for more information)

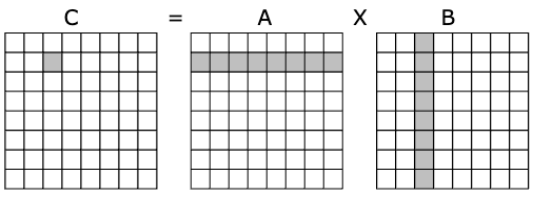
Uma multiplicação de matrizes pode ser implementada de várias formas. Nas sessões práticas da disciplina vamos utilizar, como base, uma implementação com três ciclos aninhados, começando pela variante mais comum, designada por ijk (posteriormente será identificada por DOT), para matrizes A, B e C quadradas 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 aulas. Esta primeira variante (ijk) será designada por DOT, pelo facto de cada elemento da matriz C resultar do produto escalar (***dot** product*) de uma linha da matriz (um vetor) A com uma coluna da matriz B (um segundo vetor), 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
```



Annex 2. Perf installation on Ubuntu (follow similar steps for other Linux distributions)

Install these packages:

```
sudo apt install linux-tools-common
sudo apt install linux-tools-5.15.0-48-generic  # update for your kernel version
```

Annex 3: (simple) Instructions for using the search cluster (in portuguese)

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 comando `srun`.
Na página de *elearning* da disciplina será disponibilizada (depois desta primeira sessão) uma implementação muito simples da multiplicação de matrizes que poderá ser usada nas próximas aulas. O ficheiro pode 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.

a) **Copy local file to remote machine (don't forget the two points at the end) :**

```
scp <local file name> <student_id>@s7edu.di.uminho.pt:
```

b) **Login:** `ssh <student_id>@s7edu.di.uminho.pt`

c) **Load the gcc environment:** `module load gcc/7.2.0`

d) **Compile:** `gcc …`

e) **Run:** `srun --partition=cpar perf stat -e instructions,cycles <<full_path>>/a.out`