CPD project 1

Part 1: Performance evaluation of a single core

In this project, we will study the effect on the processor performance of the memory hierarchy when accessing large amounts of data. The product of two matrices will be used for this study. Use the Performance API (PAPI) to collect relevant performance indicators of the program execution.

1. Download the example file from moodle that contains the basic algorithm in C/C++ that multiplies two matrices, i.e. multiplies one line of the first matrix by each column of the second matrix (matrixproduct.cpp). Implement the same algorithm in another programming language (just one), such as JAVA, C#, Fortran, etc, of your choice.

Register the processing time for the two languages of implementation, for input matrices from 600x600 to 3000x3000 elements with increments in both dimensions of 400. Use –O2 as optimization flag in C++.

2. Implement a version that multiplies an element from the first matrix by the correspondent line of the second matrix, using the 2 programming languages selected in 1.

Register the processing time of the algorithm, in the 2 versions, for input matrices from 600x600 to 3000x3000 elements with increments in both dimensions of 400.

Register the processing time from 4096x4096 to 10240x10240 with intervals of 2048 in C/C++ version.

3. Implement a block oriented algorithm that divides the matrices in blocks and uses the same sequence of computation as in 2, using C/C++.

Register the processing time from 4096x4096 to 10240x10240 with intervals of 2048, for different block sizes (e.g. 128, 256, 512).

Notes: use “papi_avail -a” in the “terminal” to obtain a list of available counters.
The tools referred here are for Linux native. PAPI cannot obtain counters on VMs.

CPD project 1

Part 2: Performance evaluation of a multi-core implementation

Implement parallel versions of the second implementation (line x line) of the matrix product. Analyze the next two solutions, and verify their performance (MFlops, speedup and efficiency).

```
#pragma omp parallel for
for (int i=0; i<n; i++)
     for (int k=0; k<n; k++)
           for (int j=0; j<n; j++)
           {              }
```

```
#pragma omp parallel
for (int i=0; i<n; i++)
     for (int k=0; k<n; k++)
           #pragma omp for
           for (int j=0; j<n; j++)
           {              }
```

OUTCOMES

Write a report of up to 6 pages explaining the algorithm versions and analyzing the results obtained. Justify the performance parameters selected and use them to evaluate and compare the versions implemented.

To be delivered by: 17/03/2024

Presentation: week of 18/03/2024

The Report should include:

- Problem description and algorithms explanation;
- Performance metrics;
- Results and analysis;
- Conclusions.

Notes: use "papi_avail -a" in the "terminal" to obtain a list of available counters.
The tools referred here are for Linux native. PAPI cannot obtain counters on VMs.