Université Grenoble Alpes, Grenoble INP, UFR IM^{2}AG

Master 1 Informatique and Master 1 MOSIG

UE Parallel Algorithms and Programming

TD # 1

2018

Exercise 1: Processor benchmarking and application profiling

A 40-MHz processor was used to execute a test program with several instructions. The number of cycles for each type of instructions is given in Figure 1.

Instruction type	Number of instructions	Cycles
Integer arithmetics	45000	1
Data transfert	32000	2
Floating point operation	15000	2
Control	8000	2

Figure 1: Application benchmark.

Questions

1. If CPI (Cycles per Instruction) is the average number of cycles per instruction, compute it for this program.
2. Compute the number of MIPS (*Million of Instructions per second*) obtained on this processor.
3. Compute the number of MFLOPS (*Million of floating point Instructions per second*) obtained on this processor.
4. Compute the execution time using two different methods.

Exercise 2: Ratio between cost of memory accesses and cost of computing operations

Reminder from the lecture

We suppose that we have two levels of memory hierarchy (fast and slow). At the beginning data are available in the slow (main) memory. We note:

m:	the number of memory elements (words) exchanged between the fast memory and the slow one.
t_m:	the time per operation in slow memory
f:	the number of arithmetic operations
t_f:	the time per arithmetic operation $<< t_m$
q:	the average number of flops of the algorithm per access to the slow memory ($q = \frac{f}{m}$), called **computational intensity**.

We suppose that accessing fast memory has negligible cost.

Questions

1. Give the minimal execution time of an algorithm when data are available in fast memory.
2. Give the effective execution time (including accesses to slow memory)
3. Give the effective execution time as a function of the ration between the time per operation in slow memory, the time per arithmetic operation and the average number of flops per access to slow memory.

Exercise 3: matrix-vector product

Figure 2 describes the sequential algorithm for the matrix-vector product

```
read x(1 : n) in fast memory
read y(1 : n) in fast memory
for i = 1 : n
   read line i of A in fast memory
   for j = 1 : n
      y(i) = y(i) + A(i, j) * x(j)
Write y(1 : n) in main memory
```

Figure 2: Sequential matrix-vector product algorithm.

Questions

1. Give the number of accesses to slow memory m
2. Give the number of arithmetic operations f
3. If $q = \frac{f}{m}$ is the computational intensity, give it for the matrix-vector product.
4. Compute the total execution time of the matrix-vector product.
5. Explain why the matrix-vector product is bounded by the speed of the slow memory.

Exercise 4: matrix-matrix product

Figure 3 describes the sequential algorithm for the matrix-matrix product ($C = C + A * B$).

Questions

1. Give the number of arithmetic operations f
2. Give the number of accesses to the slow memory m
3. If $q = \frac{f}{m}$ is the computational intensity, give it for the matrix-matrix product.

```
for i = 1 : n
  read line i of A in fast memory
  for j = 1 : n
    read C(i, j) in fast memory
    read column j of B in fast memory
    for k = 1 : n
        C(i, j) = C(i, j) + A(i, k) * B(k, j)
    write C(i, j) in main memory
```

Figure 3: Sequential matrix-matrix product algorithm.

Exercice 5: Blocked matrix-matrix product

Let A, B, and C be matrices of size $n \times n$ splitted in $N \times N$ sub-blocks of size $b \times b$ (where $b = \frac{n}{N}$ is called the block size). Figure 4 gives the sequential algorithm for the blocked matrix-matrix product ($C = C + A * B$).

```
for i = 1 : N
  for j = 1 : N
    read block C(i, j) in fast memory
    for k = 1 : N
        read block A(i, k) in fast memory
        read block B(k, j) in fast memory
        C(i, j) = C(i, j) + A(i, k) * B(k, j) // matrix-matrix product over 3 blocks
    write block C(i, j) in main memory
```

Figure 4: Sequential blocked matrix-matrix product algorithm.

Questions

1. Give the number of accesses to the slow memory m
2. If $q = \frac{f}{m}$ is the computational intensity, compute it for the blocked matrix-matrix product.
3. If M_{fast} is the size of the cache memory, give the optimal block size.