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

Master 1 Informatique and Master 1 MOSIG

UE Parallel Algorithms and Programming

F. Desprez (Inria), J-F. Méhaut (UGA/Polytech), T. Ropars (UGA/IM^{2}AG), E. Saillard (Inria)

TP1: February 28, 2017

1 Compiling and executing OpenMP programs

The source files of this lab are in the `/home/jfmehaut/TP1.tar.gz` file.

```
$ cd
$ cp /home/jfmehaut/TP1.tar.gz .
$ tar xvfz TP1.tar.gz
$ cd TP1
```

On the idchire machine, you can compile with the Intel Compiler (icc) or with the GNU (gcc) compiler. You start by compiling the file `exo1.c`.

```
$ gcc -fopenmp -o exo1 exo1.c
$ icc -openmp -o exo1 exo1.c
```

The binary program can be executed with the command line

```
$ ./exo1 8
I am the master thread 0 and I start
Starting Region 1
Region 1  thread 0 of team 8 (max_num_threads is 8)
Region 1  thread 7 of team 8 (max_num_threads is 8)
Region 1  thread 4 of team 8 (max_num_threads is 8)
Region 1  thread 2 of team 8 (max_num_threads is 8)
Region 1  thread 1 of team 8 (max_num_threads is 8)
Region 1  thread 6 of team 8 (max_num_threads is 8)
Region 1  thread 5 of team 8 (max_num_threads is 8)
Region 1  thread 3 of team 8 (max_num_threads is 8)
End of Region 1
Starting Region 2
Region 2  thread 3 of team 4 (max_num_threads is 8)
Region 2  thread 0 of team 4 (max_num_threads is 8)
Region 2  thread 2 of team 4 (max_num_threads is 8)
Region 2  thread 1 of team 4 (max_num_threads is 8)
End of Region 2
Region 3  thread 0 of team 2 (max_num_threads is 8)
Region 3  thread 1 of team 2 (max_num_threads is 8)
```

```
End of Region 3
I am the master thread 0 and I complete
$
```

1. Can you explain the output of this OpenMP program?
2. If you run several times this `exo1` program, can you compare and analyze the different outputs?

2 Performance Analysis of Matrix and Vector Operations

In this exercise, we will study how basic numerical functions can be simply parallelized with OpenMP. These numerical functions are implemented in the file `exo2.c`.

The `vector` and `matrix` types are defined in this file. These types are based on the constant N (`#define N 512`).

1. How much memory is used for `vector` and `matrix` variables?
2. How much memory is used all the global variables of the `exo2` program?

The `init_vector` and `init_matrix` initialize vectors and matrixes. The computing functions are:

- addition between two vectors (`add_vectors`), scalar product (`dot`) of two vectors
- multiplication between a matrix and a vector (`mult_mat_vector`)
- multiplication between two matrixes (`mult_mat_mat`)

The main function of this program calls the differents functions. For each call, the number of processor cycles is obtained with the intrinsic _rdtsc ().

The nominal frequency of the Intel Processor is 2.6 GhZ. You will analyse the performance with 2, 4, 8 and 16 threads.

1. Compute the execution time of each computing function.
2. Compute the number of floating point operation of each function.
3. Compute the number of floating point operations per second (MFLOPS or GFLOPS).
4. Analyze the speedups with 2, 4, 8 and 16 threads.
5. You can compile with the `-O2` option. Analyze the performance results with this option.

3 Dynamic Loop Scheduling

M is a lower triangular matrix. It means that all the values of M matrix above the diagonal are zero. All the values below (and also) the diagonal are useful for the computation.

1. Implement the sequential function of `mult_mat_vect_tri_inf` with M which is triangular lower.
2. Implement the parallel OpenMP function `mult_mat_vect_tri_inf1` with **static** scheduling.
3. Implement the parallel OpenMP function `mult_mat_vect_tri_inf2` with **dynamic** scheduling.
4. Implement the parallel OpenMP function `mult_mat_vect_tri_inf3` with **guided** scheduling.
5. Implement the parallel OpenMP function `mult_mat_vect_tri_inf3` with **runtime** scheduling.
6. Draw a figure with the speedups for 2, 4, 8 and 16 threads.
7. Can you study the impact of the $CHUNK$ constant on the performance results?