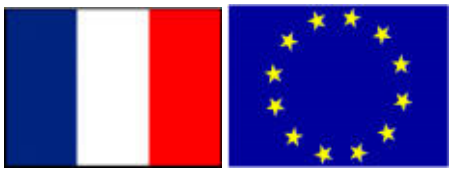
CentraleSupélec



SG6 - HPC

Optimisation & OpenMP multithreading

TD/Lab 1

Stéphane Vialle

CentraleSupélec université PARIS-SACLAY ÉCOLE DOCTORALE Sciences et technologies de l'information et de la communication (STIC) RISEGrid Research Institute for Smarter Electric Grids GrandEst ALSACE CHAMPAGNE-ARDENNE LORRAINE

CentraleSupélec

Optimisation & OpenMP multithreading

1. **Execution of the sequential computing kernels**
   → *First performance measurements*
2. **Parallelization of the kernel 4 with OpenMP**
   → ***ikj loop order***
3. **Parallelization of the kernel 5 with OpenMP**
   → ***BLAS (OpenBLAS)***
4. **Parallelization of the kernel 3 with OpenMP**
   → ***ijk + TB + loop unrolling***

CentraleSupélec

Experimentation on one CPU core

1.1 - Run the *MatrixProduct* application with 2048x2048 matrixes:

- Check SIZE definition in **kernel.h file** (should be 2048)
- Compile (*make*) with ***–O3 –funroll-loops*** options in the *Makefile*
- Execute kernel 0 up to 5:

  ./MatrixProduct –k 0 (about 51s)

  ……

  ./MatrixProduct –k 5 (about 0.5s)

k0 compiled with –O0
executes in 200s/0,09 Gflops

- Observe the differences between the successive source codes
- For each run: collect the the Gflops (printed on screen) in an Excel file (or equivalent software)

1.2 – Draw performance curves for 2048x2048 matrixes

- Y axis should start at 0 Gflops
- You can use log scale on Y axis (it could be better, with base 2…)
- Set the identifier of each kernel tested on the X axis
- Set a legend specifying the matrix size for the plotted curve

CentraleSupélec

Experimentation on one CPU core

1.3 – Conduct 2 new experiment series:

- Run 1024x1024 matrixes, using **k0** up to **k5**
- Run 4096x4096 matrixes, using **k3** up to **k5**

1.4 – Draw performance curves of the new experiments

- Superimpose the new curves to the previous one on the same figure
- Set a legend specifying the matrix size for the different plotted curves

1.5 – Performance analysis

- What are the important steps in the optimization process ?
- Which version of the kernel has the most regular performances, independent of the pb size ? **Why** ?

CentraleSupélec

Optimisation & OpenMP multithreading

1. Execution of the sequential computing kernels
   → *First performance measurements*
2. **Parallelization of the kernel 4 with OpenMP**
   → ***ikj loop order***
3. Parallelization of the kernel 5 with OpenMP
   → *BLAS (OpenBLAS)*
4. Parallelization of the kernel 3 with OpenMP
   → *ijk + TB + loop unrolling*

CentraleSupélec

Parallelization of kernel 4 (*ikj loop*)

2.1 – Parallelization of *i-loop* of *kernel 4* with 2048x2048 matrixes

- **Use OpenMP directives to efficiently parallelize *kernel 4***
  → Parallelize the outer loop of *kernel 4* (the *i-loop*)
  → Complete the **kernel.c file**
  → Be careful with the loop indexes….
- **Re-compile and run with different number of threads:**
  → ./MatrixProduct –k 4 –nt *x*
  with *x*: 1, 2, 16 threads
- **Check the rightness of your parallelization:**
  → the 3 printed matrix values must not change
  (they are independent of the number of threads)

CentraleSupélec

Parallelization of kernel 4 (*ikj loop*)

2.2 – Performance measurement and analysis

- Each Kyle machine has two 8-cores Intel processors:
  → 16 physical cores per machine
  → 32 logical cores per machine
- **Collect performances** (Gflops) reached by each run
- **Draw the performance curve** of this parallelization
- **Draw the ideal performance curve**
  → What should be the ideal performance curve ?
- **Compare ideal and experimental performances curves**

2.3 – Speedup and efficiency

- Compute and draw the **speedup curve**, function of the nb of used cores
  → compare to the ideal speedup curve
- Compute and draw the **efficiency curve**, function of the nb of used cores
  → Does your parallelization use efficiently the ressources of the node ?

CentraleSupélec

Parallelization of kernel 4 (*ikj loop*)

2.4 – Load balancing on virtual cores ?

- Measure performances reached with:
  16, 17, 19, 21, 23, 25, 27, 29, 31, 32 threads
- Draw the **performance curve**
- Do the performances increase regularly ?
- **What do you think about the load balancing of this parallelization?**

CentraleSupélec

Parallelization of kernel 4 (*ikj loop*)

2.5 – Parallelization of *k-loop* (only) of *kernel 4* with 2048x2048 matrixes

- Is it possible to use a simple « parallel for » ? Why ?
- **Design an efficient solution to parallelize the *k-loop***
- Re-compile and run with different number of threads
- **Check the rightness of your parallelization**

2.6 – Performance analysis

- Draw the performance curve, using 1, 2, 4, 8, 16, 32 threads
- Compare to the performance curve of the *i-loop* parallelization

CentraleSupélec

Parallelization of kernel 4 (*ikj loop*)

2.7 – Parallelization of *j-loop* (only) of *kernel 4* with 2048x2048 matrixes

- Parallelize efficiently the j-loop only
- Check the rightness of the computations

- Measure performances
- Compare to the previous performance curves

- **Explain the gradation of the 3 performance curves**

CentraleSupélec

Optimisation & OpenMP multithreading

1. Execution of the sequential computing kernels
   → *First performance measurements*
2. Parallelization of the kernel 4 with OpenMP
   → *ikj loop order*
3. **Parallelization of the kernel 5 with OpenMP**
   → ***BLAS (OpenBLAS)***
4. Parallelization of the kernel 3 with OpenMP
   → *ijk + TB + loop unrolling*

CentraleSupélec

Parallelization of kernel 5 (*BLAS*)

3.1 – Parallelization of *BLAS library call* (kernel 5) with 4096x4096 matrixes

- **Propose a general strategy** to parallelize kernel 5 with OpenMP
- Look at the BLAS/dgemm documentation of the course
- **Implement a solution assuming:**
  SIZE = q.NbThreads
- **Check the rightness** of your solution on 4096x4096 matrixes

3.2 – Generic parallelization of kernel 5

- Extend your solution to support matrixes of any size:
  SIZE = q.NbThreads + r
- Dsitribute the rest *r* on *r* threads (one more line on *r* threads)
- Each thread has to make only one call to dgemm
- **Check the rightness** of your solution on **4099x4099** matrixes

CentraleSupélec

Parallelization of kernel 5 (*BLAS*)

3.3 – Performance measurement and analysis with 4096x4096 matrixes

- **Measure performances (Gflops)** on 1, 2, 4, 8, 16, 32 threads
- Draw performance curve
- Compare to ideal performance curve

- **Draw Speedup and efficiency curves**, function of the used cores
  Consider the sequential refecrence is the kernel 5 running on one core

- **Compare to the curves obtained with the *i-loop* of kernel 4**

  → Do you observe the same optimal number of threads ?
  Why ?

  → What is the solution reaching the best performances (Gflops) ?
  What is the solution reaching the best speedup ?
  Explanation ?

CentraleSupélec

Optimisation & OpenMP multithreading

1. **Execution of the sequential computing kernels**
   → *First performance measurements*
2. **Parallelization of the kernel 4 with OpenMP**
   → ***ikj loop order***
3. **Parallelization of the kernel 5 with OpenMP**
   → ***BLAS (OpenBLAS)***
4. **Parallelization of the kernel 3 with OpenMP**
   → ***ijk + TB + loop unrolling***

CentraleSupélec

Parallelization of kernel 3

4.1 – Parallelization of kernel 3 (TB + accumulator array)

- Parallelize the *k-loop* of *kernel 3*
- Check the rightness of your solution
- Parallelize the *i-loop* (only)
- Parallelize the *j-loop* (only)

4.2 – Performance analysis

- Measure performances of each solution
- Draw performance curves
- Does the performance gradation of the solutions seem logic ?

CentraleSupélec

Optimisation & OpenMP multithreading

(Data Center for Education)

End