# Projet HPC

Jean-Pierre Mansour - Ezekiel Rembangouet - Paul Pasquet

December 2024

## 1 Introduction

**1.** *How many cores are available in the computer?*
Total processors: 8 Physical cores: 4

**2.** *How many processors (or socket or node)?*
Sockets: 1 NUMA nodes: 1

**3.** *What is the maximum frequency speed?*
Maximum frequency: 4800 MHz

**4.** *What is the maximum amount of RAM Memory?*
Le maximum de RAM mémoire disponible est 32 GB.

**5.** *What is the operating system currently running?*
Operating system architecture: $x86\_64$ (64-bit) It is running on Linux

**6.** *What is the performance in Tflop/s of the computer?*
We have: Frequency = 3 GHz, Each core has 2 FPUs, with no vector capacity.
Performance per core = Frequency × (nb of FPUS):
Performance per $core = 3\,GHz \times 2 = 6\,GFLOP/s$
Total performance = Performance per core×Number of cores. Then,
Total performance$= 6\,GFLOP/s \times 8 = 48\,GFLOP/s = 0.048\,TFlop/s$
In conclusion, Performance: $0.048TFlop/s$.

## 2 Tests with small mesh and questions

**1.** *Verify convergency is achieved for each case and write down the error and the number of iterations necessary (read on screen ≪ Global error ≫) for different values of NTX, and different number of processes (1, 2 and 4).*
**1.1** *The case ntx = 16*

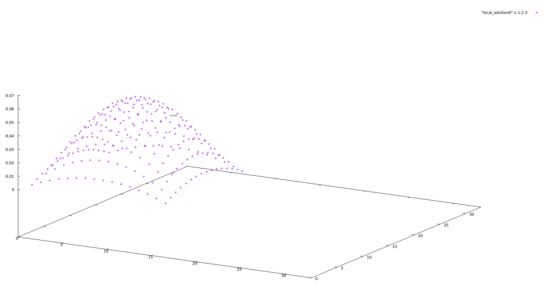

Figure 1: Solution for ntx = 16 and one process

With $ntx = 16$ and one process, we obtain the graph of Figure 1 and we have convergency after 947 iterations in $2.36 \times 10^{-3}s$. The global error is $9.89 \times 10^{-11}$.

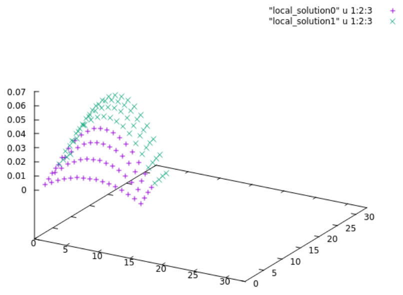

Figure 2: Solution for ntx = 16 and two processes

With $ntx = 16$ and two processes, we obtain the graph of Figure 2 and we have convergency after 947 iterations in $5.20 \times 10^{-3}s$. The global error is $9.89 \times 10^{-11}$.

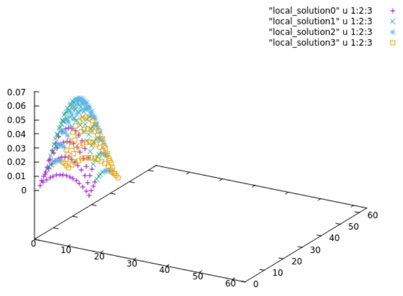

Figure 3: Solution for ntx = 16 and four processes

With $ntx = 16$ and four processes, we obtain the graph of Figure 3 and we have convergency after 947 iterations in $2.65 \times 10^{-3}s$. The global error is $9.89 \times 10^{-11}$.

**1.2** *The case ntx = 32*

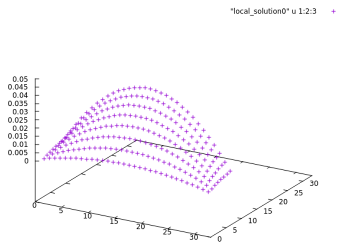

Figure 4: Solution for ntx = 32 and one process

With $ntx = 32$ and one process, we obtain the graph of Figure 4 and we have convergency after 3288 iterations in $2.85 \times 10^{-2}s$. The global error is $9.99 \times 10^{-11}$.

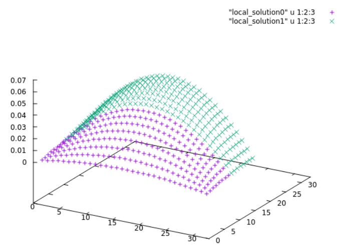

Figure 5: Solution for ntx = 32 and two processes

With $ntx = 32$ and two processes, we obtain the graph of Figure 5 and we have convergency after 3288 iterations in $2.23 \times 10^{-2} s$. The global error is $9.99 \times 10^{-11}$.

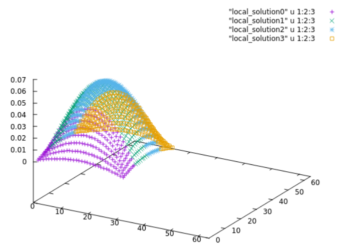

Figure 6: Solution for ntx = 32 and four processes

With $ntx = 32$ and four processes, we obtain the graph of Figure 6 and we have convergency after 3288 iterations in $1.03 \times 10^{-2} s$. The global error is $9.99 \times 10^{-11}$.

### **1.3** *The case ntx = 64*

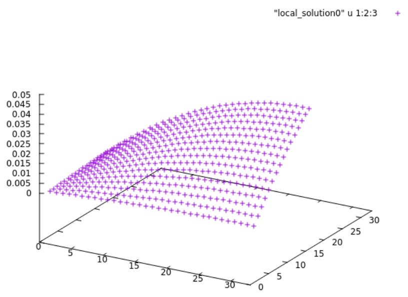

Figure 7: Solution for ntx = 64 and one process

With $ntx = 64$ and one process, we obtain the graph of Figure 7 and we have convergency after 11609 iterations in $0.26s$. The global error is $10.0 \times 10^{-11}$.

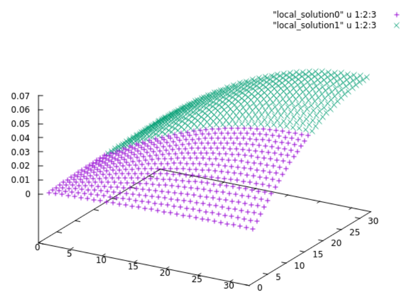

Figure 8: Solution for ntx = 64 and two processes

With $ntx = 64$ and two processes, we obtain the graph of Figure 8 and we have convergency after 11609 iterations in $0.16s$. The global error is $10.0 \times 10^{-11}$.

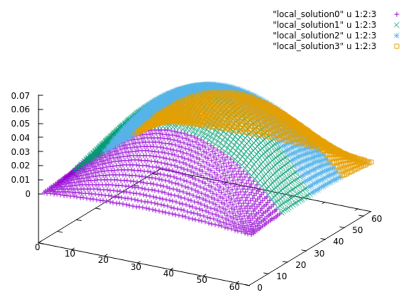

Figure 9: Solution for ntx = 64 and four processes

With $ntx = 64$ and two processes, we obtain the graph of Figure 9 and we have convergency after 11609 iterations in $0.12s$. The global error is $10.0 \times 10^{-11}$.

**1.4** *The case ntx = 500 with 100 iterations*

- With 1 process, the 100 iterations end in $0.20s$ and the global error is $9.95 \times 10^{-7}$.
- With 2 processes, the 100 iterations end in $0.11s$ and the global error is $9.95 \times 10^{-7}$.
- With 3 processes, the 100 iterations end in $6.181 \times 10^{-2}s$ and the global error is $9.95 \times 10^{-7}$.
- With 4 processes, the 100 iterations end in $6.182 \times 10^{-2}s$ and the global error is $9.95 \times 10^{-7}$.
- With 5 processes, the 100 iterations end in $5.81 \times 10^{-2}s$ and the global error is $9.95 \times 10^{-7}$.
- With 10 processes, the 100 iterations end in $8.13 \times 10^{-2}s$ and the global error is $9.95 \times 10^{-7}$.

**1.5** *The case ntx = 1000 with 100 iterations*

- With 1 process, the 100 iterations end in $0.57s$ and the global error is $2.49 \times 10^{-7}$.
- With 2 processes, the 100 iterations end in $0.31s$ and the global error is $2.49 \times 10^{-7}$.
- With 3 processes, the 100 iterations end in $0.230s$ and the global error is $2.49 \times 10^{-7}$.
- With 4 processes, the 100 iterations end in $0.234s$ and the global error is $2.49 \times 10^{-7}$.
- With 5 processes, the 100 iterations end in $0.26s$ and the global error is $2.49 \times 10^{-7}$.
- With 10 processes, the 100 iterations end in $0.29s$ and the global error is $2.49 \times 10^{-7}$.

**1.6** *The case ntx = 12000 with 100 iterations*

- With 1 process, the 100 iterations end in $157s$ and the global error is $1.74 \times 10^{-9}$.
- With 2 processes, the 100 iterations end in $58.3s$ and the global error is $1.74 \times 10^{-9}$.
- With 3 processes, the 100 iterations end in $52.4s$ and the global error is $1.74 \times 10^{-9}$.
- With 4 processes, the 100 iterations end in $51.4s$ and the global error is $1.74 \times 10^{-9}$.
- With 5 processes, the 100 iterations end in $62.0s$ and the global error is $1.74 \times 10^{-9}$.
- With 10 processes, the 100 iterations end in $58.8s$ and the global error is $1.74 \times 10^{-9}$.

By choosing a larger and larger NTX, we increase the number of nodes in our grid, which leads to a refinement of the mesh. Consequently, this increases the number of Jacobi iterations we need to compute. Therefore, this process will take longer for a finer mesh.

The numerical solution for a very fine mesh will likely better approximate the true solution. However, mesh refinement results in a more costly resolution time. It is therefore necessary to strike a balance between mesh refinement and resolution time.

Fortunately, parallelizing the computation offers the advantage of distributing the task of calculating the iterations across multiple processors that communicate with each other. This would significantly reduce the resolution time and allow us to refine the mesh.

## 3 Performance tests and questions

The consistency of the resolution time with respect to the number of processors is a strong indicator that the processors communicated effectively with each other throughout the calculation and that the iterations, which needed to be exchanged between the subdomains, were successfully communicated without interruption during the computation process.

On the other hand, the slight increase in computation time as the number of processors increases suggests that there was an overhead in communication between the processors.

The speedup shows significant improvement when increasing the number of processes from 1 to 2 or 3, indicating good scalability for small numbers of processes. However, beyond 3-4 processes, the speedup plateaus or even slightly decreases, particularly with 10 processes, due to increased communication overhead in the MPI framework. This suggests that while the algorithm benefits from parallelism initially, its scalability is limited by the cost of inter-process communication as the number of processes grows. This behavior is typical in weak scalability tests when the problem size per process becomes too small relative to the communication overhead.

In conclusion, the increase in problem size demonstrates good scalability with a small number of processes, as the speedup improves significantly. However, the speedup plateaus with a larger number of processors.