Institut Superior de l'Aeronatique et de l'Espace

1MAE011 - Scientific Computing 2 – High Performance Computing

Marked Seminar Report

Parallel Resolution of an Equivalent 2D Laplacian Problem

Submitted by:
Harshini AICH
Akash SHARMA

Submitted on:
09th May 2022

1. Introduction to the Problem

The following report presents and discusses the results obtained from a parallel computing code. The code is built to solve the Poisson equation over a 2D domain by splitting it into subdomains, such that number of subdomains = number of processes.

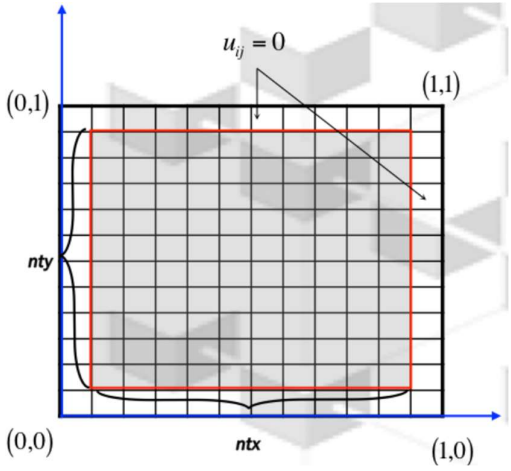
1.1. Partial Differential Equation (PDE) to be solved

$$\Delta u(x,y) = f(x,y) \text{ in } \Omega = [0,1]\times[0,1]$$
$$u(x,y) = 0 \text{ on } \partial\Omega$$
$$f(x,y) = 2\cdot\left(x^2 - x + y^2 - y\right)$$
$$u_{exact} = xy(x-1)(y-1)$$

The problem we are dealing with is the Poisson problem with the function value going to zero at the domain boundaries.

The source term f is a quadratic expression. The exact solution to this problem is also known.

1.2. Domain Discretization



- nty = ntx = number of inner nodes
- h = 1/(ntx+1)
- A 5-point stencil is employed.
- This means the value u_{ij}^{n+1} is calculated from u_{i+1j}^{n}, u_{i-1j}^{n}, u_{ij+1}^{n} and u_{ij-1}^{n}.
- The Jacobi iteration is employed to form the numerically approximate problem.

$$u_{ij}^{n+1} = c_0\left(c_1\left(u_{i+1j}^{n} + u_{i-1j}^{n}\right) + c_2\left(u_{ij+1}^{n} + u_{ij-1}^{n}\right) - f_{ij}\right)$$

where

$$c_0 = \frac{h^2}{4}$$

$$c_1 = c_2 = \frac{1}{h^2}$$

The domain is subdivided into subdomains such that eah processor can work on a part of the problem and communicate with each other to reach the global solution.

2. Computation Results

The code is modified as below to implement the broadcast method of communication between the subdomains.

```
40 ! TO DO : uncomment one of the two following lines to perform the proper communication
41 CALL MPI_BCAST(ntx, 1, MPI_INTEGER, 0, MPI_COMM_WORLD, code)
42 !!CALL MPI REDUCE(ntx,sum ntx, 1, MPI INTEGER, MPI SUM,0, MPI COMM WORLD, code)
```

The code is then run for several values of ntx and np to test convergence and performance.

2.1. Computer Configuration

Sockets available	1
Computer Cores available per socket	4
Threads per core	2
Operating System	Linux
CPU/Processor Frequency	3600 MHz
L3 Cache Size	8192 kB

Performance without multi-threading
= 1 (socket) * 4 (cores) * 3.6 GHz (cycles/second) * 4 (double-precision FLOPs/second)
= **57.6 GFlops/s**

Performance with multi-threading
= 1 (socket) * 4 (cores) * 3.6 GHz (cycles/second) * 4 (double-precision FLOPs/second) * 2(threads/core)
= **115.2 GFlops/s**

2.2. Small Mesh Results & Observations

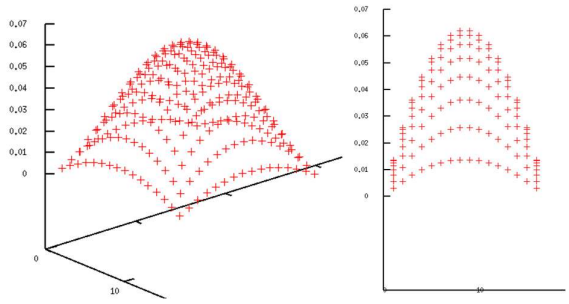
	np 1	**np 2**	**np 4**
ntx 16	Iterations: **947** Error: **$9.8874*10^{-11}$**	Iterations: **947** Error: **$9.8874*10^{-11}$**	Iterations: **947** Error: **$9.8874*10^{-11}$**
ntx 32	Iterations: **3288** Error: **$9.9904*10^{-11}$**	Iterations: **3288** Error: **$9.9904*10^{-11}$**	Iterations: **3288** Error: **$9.9904*10^{-11}$**
ntx 64	Iterations: **11609** Error: **$9.9968*10^{-11}$**	Iterations: **11609** Error: **$9.9968*10^{-11}$**	Iterations: **11609** Error: **$9.9968*10^{-11}$**

The increase in the number of internal nodes (ntx) is seen to result in an increase of iterations required and in the final global error.

The error must be below the threshold at each node for two consecutive time steps, in order to reach convergence. The number of iterations required to achieve this convergence is expected to be higher as the number of nodes increases.

The global error is calculated as a summation of the error at each node and is also expected to increase with the total number of nodes.

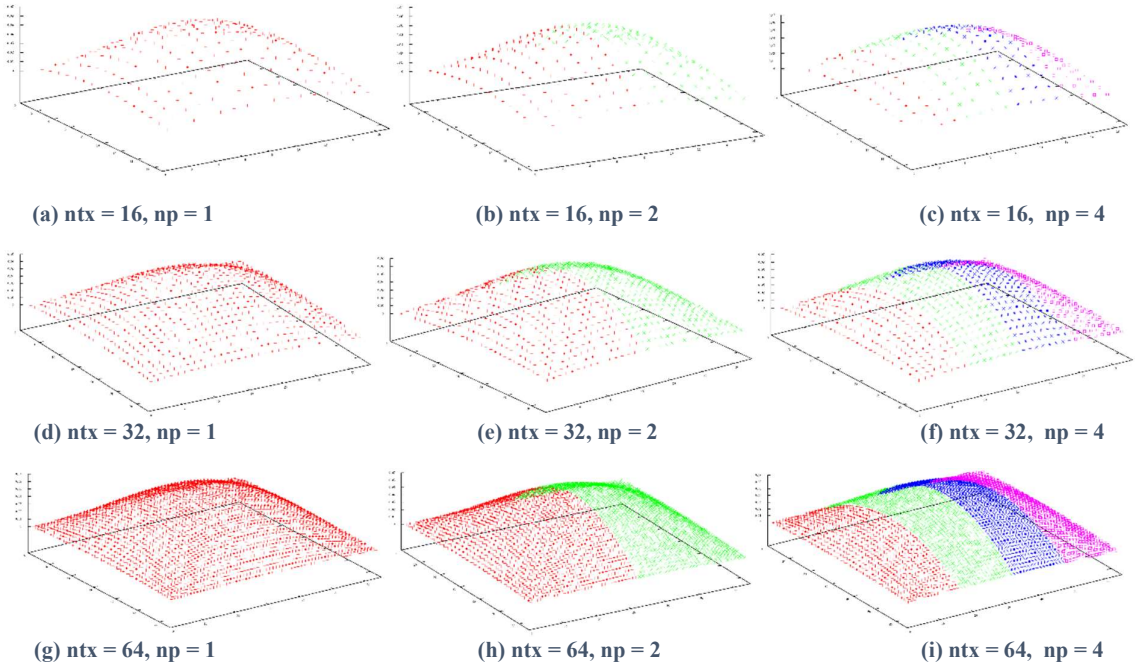
2.3. Small Mesh Graphs



Plot showing the solution for the mesh of 16x16 nodes and the solution value

The plots are displayed below for scaled axes to better show the solution. The plot size of the mesh and the solution has also increased (from 16 to 32 and then 64). This is because the nodes are numbered and not representative of actual geometric distance.

The number of nodes in the solution in seen to increase as a greater number of internal points have been considered.



(a) ntx = 16, np = 1 (b) ntx = 16, np = 2 (c) ntx = 16, np = 4

(d) ntx = 32, np = 1 (e) ntx = 32, np = 2 (f) ntx = 32, np = 4

(g) ntx = 64, np = 1 (h) ntx = 64, np = 2 (i) ntx = 64, np = 4

2.4. Performance Results

	np 1	**np 2**	**np 4**
ntx 1000	Iteration time: **0.823** secs User time: **0.865** secs	Iteration time: **0.479** secs User time: **1.006** secs	Iteration time: **0.474** secs User time: **1.994** secs
ntx 5,000	Iteration time: **21.168** secs User time: **21.518** secs	Iteration time: **12.151** secs User time: **24.679** secs	Iteration time: **10.464** secs User time: **42.405** secs
ntx 10,000	Iteration time: **84.310** secs User time: **85.600** secs	Iteration time: **48.546** secs User time: **93.435** secs	Iteration time: **40.892** secs User time: **165.814** secs
ntx 15,000	Iteration time: **192.498** secs User time: **195.043** secs	Iteration time: **109.577** secs User time: **222.041** secs	Iteration time: **104.751** secs User time: **423.482** secs

The iteration time is the internal computation time as calculated by the Poisson algorithm code. The user time is the total processing time spent by all the cores on the algorithm. This includes the time to compute the approximate solution, compute the exact solution and perform auxiliary functions like deallocate.

This explains why the difference between User Time = np * Iteration Time increases with the number of processors and the number of nodes.

Speedup is defined as the ratio between the sequential and the parallel processing time. This is obtained for each value of ntx by looking at the iteration times. The sequential processing time refers to the one with 1 processor (np=1). Speedup evolution follows an increasing trend with increasing number of processors.

	Speedup with np2	**Speedup with np4**
ntx 1000	1.718	1.736
ntx 5,000	1.742	2.022
ntx 10,000	1.736	2.061
ntx 15,000	1.756	1.837