Parallel Computing, 2022S

Assignment 2

Solving Heat Equation in 2D

The heat equation is a Partial Differential Equation

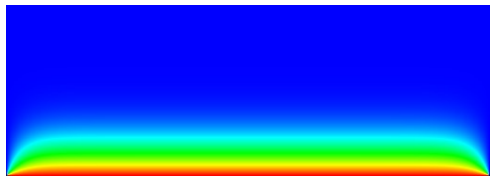
$$\frac{\partial u}{\partial t} = \alpha\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)$$

Jacobi iterative method:

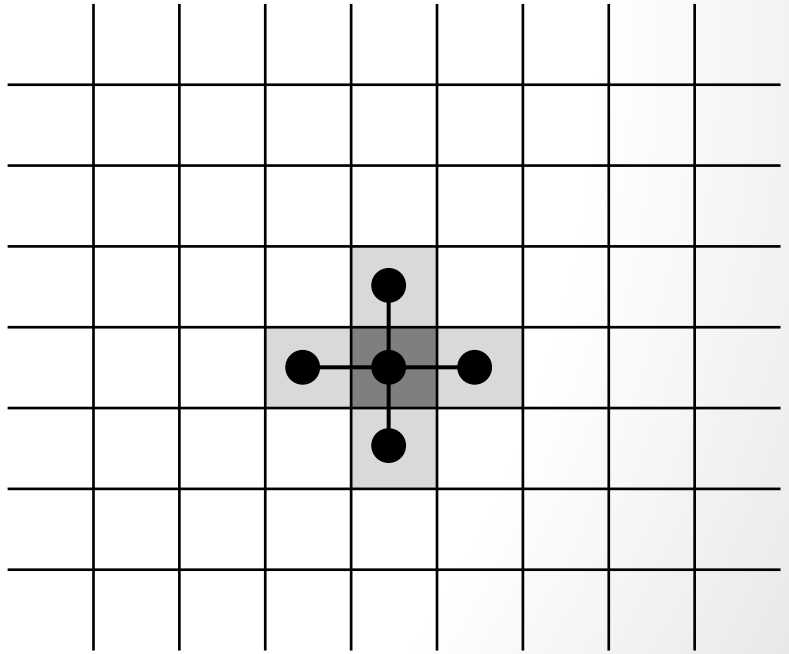
$$v_{m.l}^{n+1} = \frac{1}{4}\left(v_{m+1,l}^{n} + v_{m-1,l}^{n} + v_{m.l+1}^{n} + v_{m.l-1}^{n}\right) - \frac{h^2}{4} f_{ml}$$

Zero

- Working NxM matrix
- Five-point stencil
- Calculation of averages



Example output

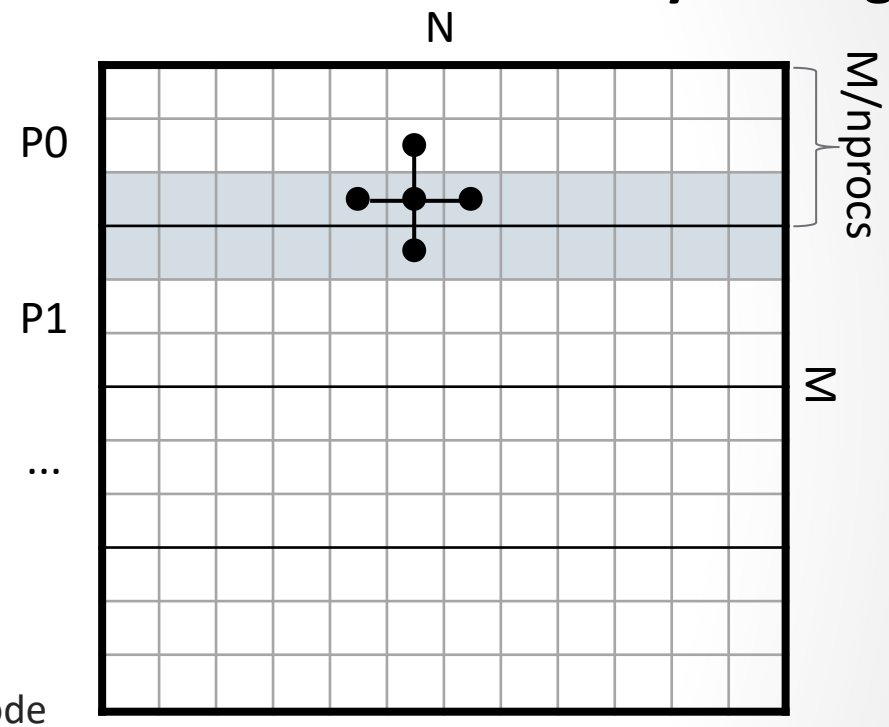


Five-point stencil

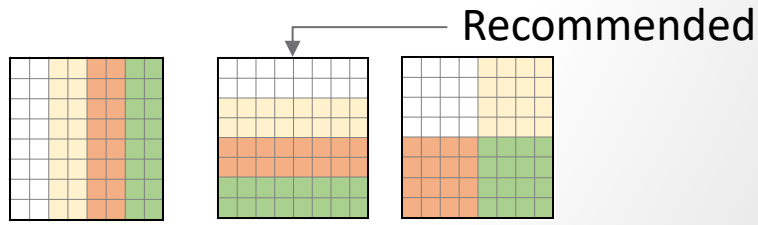
Your Task

Realize an efficient parallel implementation of the serial code for iteratively solving heat equation using MPI.

1. **Distribute data over all processes**
   - Vertical, horizontal, 2d?
   - Also initialize in a distributed fashion
   - Each process gets only a part of the matrix
   - Support N != M
     - e.g., have larger last block
2. **Use point-to-point communication to communicate the overlapping regions**
3. **Make sure that you get the termination criteria correctly**
   - `diffnorm` has to be the same as if executed with the sequential code
4. **Verify/compare to the sequential version**
   - **Use collectives to transfer data to rank 0, and then compare!**
     - **You can use the verify function from a2-helpers.hpp**
   - **Results must be the same!**



Overlapping (halo/ghost) regions



Other configurations

Hints: First support N==M and then improve

Useful MPI routines: MPI_Send/Recv, MPI_Allreduce, MPI_Gatherv

Your Task

Use MPI to develop an efficient parallel code that can utilize all available nodes on a cluster

Develop anywhere, test on ALMA

1. Use up to all nodes and all cores
2. Allow different sizes of matrix and also support M != N
   - But you can assume that matrix is larger than some minimum

3. Try the following code parameters:
   a. `mpirun ./a2 --m 2688 --n 4096 --epsilon 0.001 --max-iterations 1000`
   b. `mpirun ./a2 --m 2688 --n 4096 --epsilon 0.001 --max-iterations 2000`
   c. `mpirun ./a2 --m 1152 --n 1152 --epsilon 0.001 --max-iterations 1000`

4. Try the following cluster configurations:
   a. 1 node, 16 processes
   b. 2 nodes, 32 processes
   c. 4 nodes, 64 processes
   d. 6 nodes, 96 processes

- Achieve a speedup of 12+ using all available nodes on the Alma system (using 3.a. and 4.d) and observe the speedup when using other configurations.
- Try to explain the differences (if any) in the performance as M, N, and max-iterations change

  Note that in most cases the code will reach the maximum number of iterations in the current setup

Your Task

Write a report and include:

1. Speedup results (graphs + tables)
   - Measure performance (computation and gather separately – as pointed out in the sequential code)
     - You can consider avoiding measuring MPI_Init() time
     - Use MPI_Wtime() for time measurements
   - Compare speedups for different matrix sizes (given in the last slide)
2. Source codes + Slurm job scripts
3. Discussion
   - Performance
   - Communication
     - and effects of the communication on the performance
   - Bottlenecks?
   - Possible improvements?

Sequential Code, Template, Compilation

Given in Moodle

- Main part of the source code: `a2.cpp`
- Helper functions and data structures: `a2-helpers.cpp`
- Compile (Alma):
  - `g++ -O2 -o a2 a2.cpp` (sequential version)
  - `mpic++ -O2 -o a2 a2.cpp` (MPI)
- Run (Usage):
  - Sequential
    - **`./a2 --n <columns> --m <rows> [--max-iterations <int> --epsilon <double>]`**
  - To run your MPI program
    - **`mpirun –np ./a2 --n <columns> --m <rows> [--max-iterations <int> --epsilon <double>]`**
    - **Use Slurm to run on Alma**

Sequential Code Notes (a2-helpers.hpp)

Contains a custom data structure “Mat”

- Contiguous 2d vector
- Note the ”(“ and “)” instead of “[“ and ”]”

```
Mat mat(height, width);
for (int i=0; i<mat.height; i++) {
   for (int j = 0; j < mat.width; j++) {
      mat(i,j) = f(x);
   }
}
```

And a set of helper functions that you do not need to modify:

- `void process_input(int argc, char **argv, int& N, int& M, int& max_iterations, double& epsilon);`
  - Process input arguments
- `void save_to_disk(Mat& U, std::string filename);`
  - Save data to disk (so it can be converted to an image)
- `void print_mat(Mat& m);`
  - Prints to std output
- `bool verify(Mat& a, Mat& b);`
  - Compares two give matrices
- `void heat2d_sequential(int max_iterations, double epsilon, Mat& U, int& iteration_count);`
  - Sequential version of the code (for verification)

Slurm

https://slurm.schedmd.com/

Job scheduling system for clusters

→ You submit a job script which tells the system how to execute your MPI program on the cluster

sbatch

- Submit a batch script to Slurm

squeue

- view information about jobs located in the Slurm scheduling queue

sstat

- Display various status information of a running job/step

sinfo (-N)

- View information about Slurm nodes and partitions

scancel

- Used to signal jobs or job steps that are under the control of Slurm (cancel a job with some ID)

Job Script

```
#!/bin/bash
#SBATCH -N 4
#SBATCH --ntasks 16
mpirun --report-bindings ./a2
```

4 nodes

MPI tasks

16 processes over 4 nodes

Optional -> lets you know where processes were executed

The result will be saved in slurm-6235.out when complete

```
> sbatch jobscript.sh
Submitted batch job 6235

> squeue
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
 6236       all jobscrip johndoe  R       0:11      4 alma[01-04]
```

Alma System

Frontend + 6 worker nodes

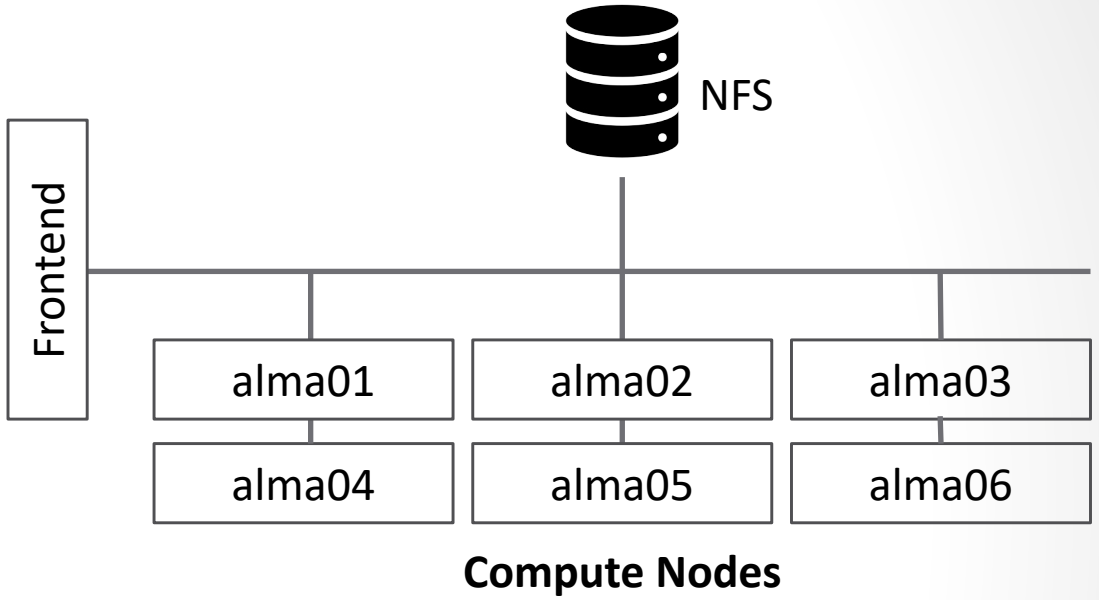
Node:

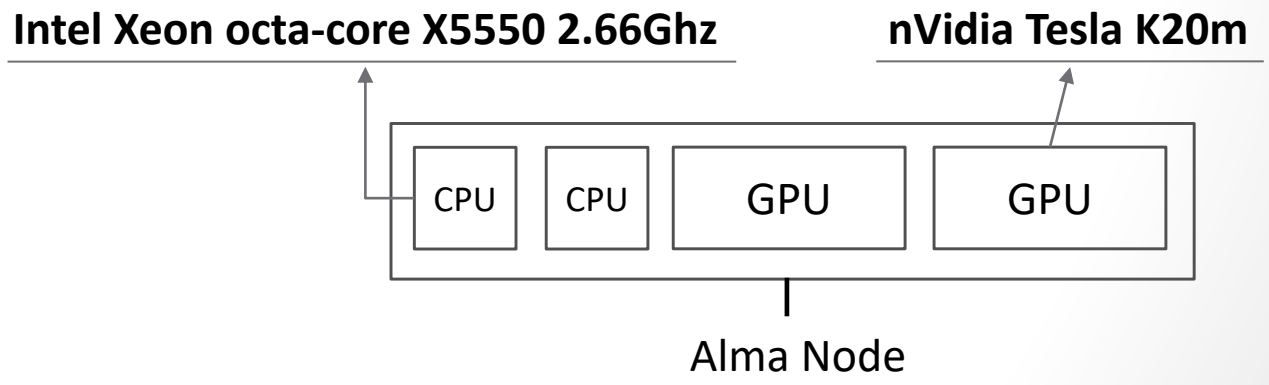
- 2x Intel Xeon octa-core X5550 2.66Ghz
- 2x nVidia Tesla K20m
- 128GB RAM
- CentOS

Access via Frontend nodes

Shared file system

Submit jobs scheduled to run on compute nodes





http://www.par.univie.ac.at/teach/doc/alma.html