

# Introduction to Parallel Processing

## Home Assignment #2

Lecturer: Dr. Guy Tel-Zur

Topic: Parallel Computations with MPI

Goal: Solve 2D heat equation – Laplace equation steady state

Student: Shir Chen 203869698

### General Explanations:

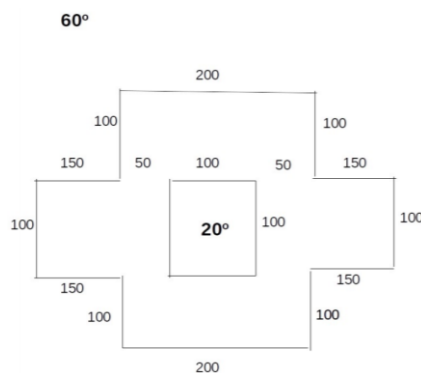
The Laplace equation applies to many physical problems, such as temperature.

For temperature, it is the steady state heat equation.

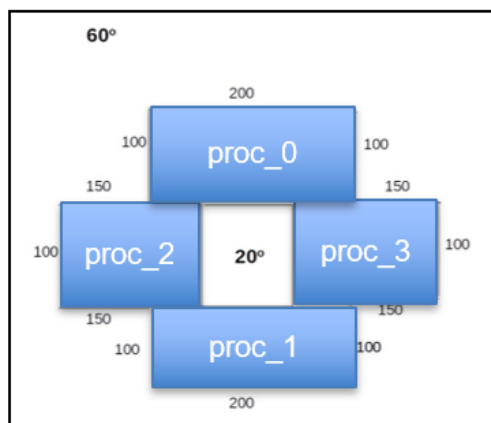
In the Laplace equation each point is the average of its neighbors. Therefore, we can iteratively converge to that steady state by repeatedly computing new values at each point from the average of neighbors' points. We keep on iterating until the difference is small enough for us to tolerate. This is the Jacobi Iteration method:

$$M_{k+1}(i,j) = \frac{M_k(i-1,j) + M_k(i,j-1) + M_k(i+1,j) + M_k(i,j+1)}{4}$$

In this code we solve the heat equation for the following geometric figure problem:



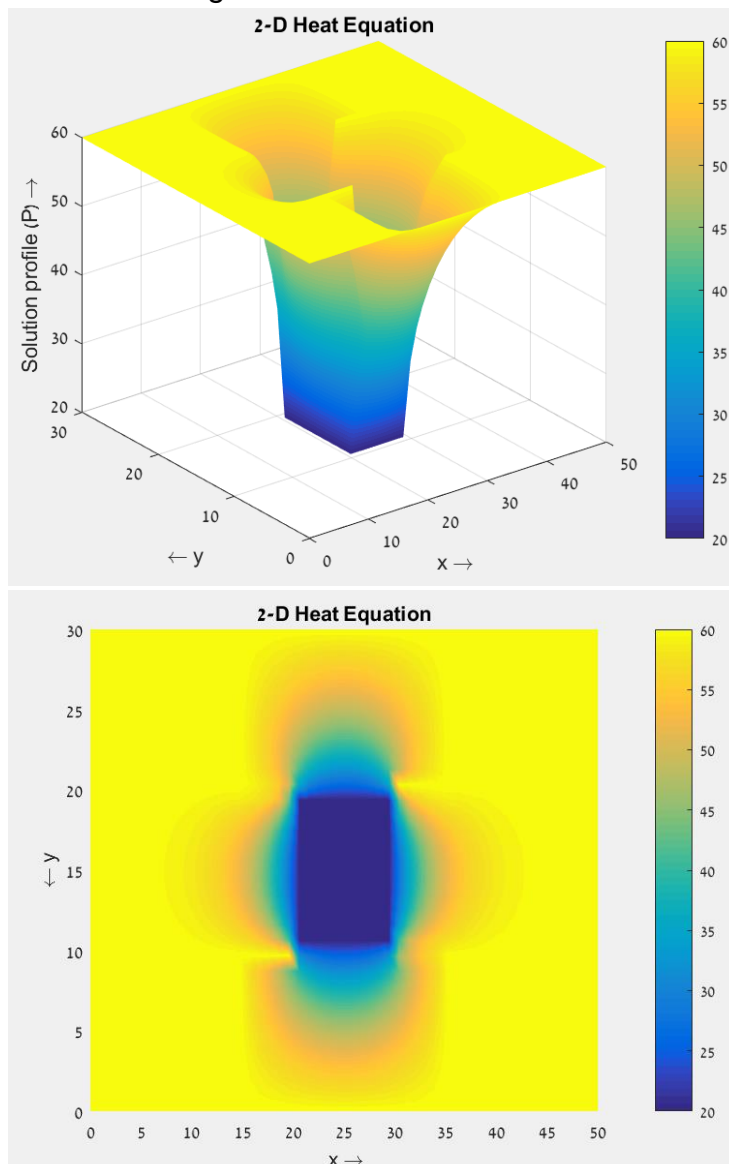
We were asked to use only 4 processes, due to that requirement I selected to use the following domain decomposition:



Using this domain decomposition, each process is calculating the same data amount, 200X100 matrix, and each process must communicate with two other processes but only 50X1 array data is transferred between them.

## Instructions How to Compile and Execute:

1. On a linux machine run: `xhost +`  
this command enables your connection to the hobbit node (adds you to the list allowed to make connection to the X server)
2. Next connect to the hobbit with your username with the following command:  
`ssh -X (username)@hobbit(xx).ee.bgu.ac.il`
3. Move to the c file folder: `cd (your folder)`
4. MPI compile your file: `mpicc -o (exefilename) (yourfilename).c`
5. MPI run the created executable file: `mpirun -np (numofproc) ./(exefilename)`
6. The code prints every iteration current error and when we reach the required number of iterations or the error is less than what we expected it update an output txt file with the final heat data. The file `outputMatrix.txt` can be found in the same directory.
7. The `outputMatrix.txt` file can be read by MATLAB program and show the heat map on a 3D axis using the `surf` function. The MATLAB code is also attached to this assignment.



## Code tracing and profiling using Jumpshot:

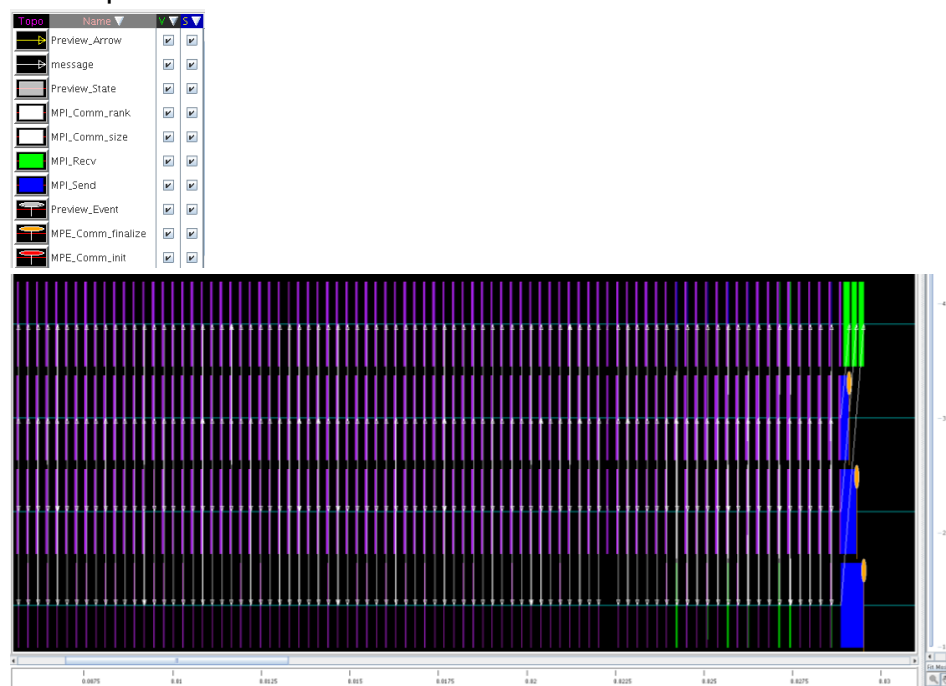
### Running Jumpshot:

1. mpecc -mpilog main.c -o exe
2. mpirun -n 4 ./exe
3. clog2TOSlog2 exe.clog2
4. jumpshot exe.slog2

```
At iteration 98, diff is 1.435707e+01
At iteration 99, diff is 1.433008e+01
At iteration 100, diff is 1.430503e+01
Output updated
Writing logfile....
Enabling the Default clock synchronization...
Finished writing logfile ./exe.clog2
[bagirov@hobbits ~/Desktop]$ clog2TOSlog2 exe.clog2
clog2TOSlog2: Command not found.
[bagirov@hobbits ~/Desktop]$ clog2TOSlog2 exe.clog2
clog2TOSlog2: Command not found.
[bagirov@hobbits ~/Desktop]$ clog2TOSlog2 exe.clog2
GSL_LIBRARIES is set. GSL_LIBRARIES = /opt/mpi/lib
Drawables: WARNINGS Equal Drawables?
State[ Primitive[ InfoBox[ TimeBox(0.011926889419555664,0.011926889419555664) Category=objDef( evt=160,161), Category[ Index=81, name=UnknownType-81, topoState, color=(153,102,51,255,true), isUsed=tru
e, width=1, vis=true, search=true, ratios=0.0,0.0, count=0 ] ] ] (0.011926889, 2) (0.011926889, 2) ] bsize=32 ]
State[ Primitive[ InfoBox[ TimeBox(0.011926889419555664,0.011926889419555664) Category=objDef( evt=160,161), Category[ Index=81, name=UnknownType-81, topoState, color=(153,102,51,255,true), isUsed=tru
e, width=1, vis=true, search=true, ratios=0.0,0.0, count=0 ] ] ] (0.011926889, 2) (0.011926889, 2) ] bsize=32 ]
Drawables: WARNINGS Equal Drawables?
State[ Primitive[ InfoBox[ TimeBox(0.011926889419555664,0.011926889419555664) Category=objDef( evt=160,161), Category[ Index=81, name=UnknownType-81, topoState, color=(153,102,51,255,true), isUsed=tru
e, width=1, vis=true, search=true, ratios=0.0,0.0, count=0 ] ] ] (0.011926889, 2) (0.011926889, 2) ] bsize=32 ]
State[ Primitive[ InfoBox[ TimeBox(0.011926889419555664,0.011926889419555664) Category=objDef( evt=160,161), Category[ Index=81, name=UnknownType-81, topoState, color=(153,102,51,255,true), isUsed=tru
e, width=1, vis=true, search=true, ratios=0.0,0.0, count=0 ] ] ] (0.011926889, 2) (0.011926889, 2) ] bsize=32 ]
Drawables: WARNINGS Equal Drawables?
State[ Primitive[ InfoBox[ TimeBox(0.025085926055908203,0.025085926055908203) Category=objDef( evt=160,161), Category[ Index=81, name=UnknownType-81, topoState, color=(153,102,51,255,true), isUsed=tru
e, width=1, vis=true, search=true, ratios=0.0,0.0, count=0 ] ] ] (0.025085926, 2) (0.025085926, 2) ] bsize=32 ]
State[ Primitive[ InfoBox[ TimeBox(0.025085926055908203,0.025085926055908203) Category=objDef( evt=160,161), Category[ Index=81, name=UnknownType-81, topoState, color=(153,102,51,255,true), isUsed=tru
e, width=1, vis=true, search=true, ratios=0.0,0.0, count=0 ] ] ] (0.025085926, 2) (0.025085926, 2) ] bsize=32 ]
Drawables: WARNINGS Equal Drawables?
State[ Primitive[ InfoBox[ TimeBox(0.025085926055908203,0.025085926055908203) Category=objDef( evt=160,161), Category[ Index=81, name=MPI_Send, topoState, color=(0,0,255,255,true), isUsed=tru
e, width=1, vis=true, search=true, ratios=0.0,0.0, count=0 ] ] ] (0.025085926, 2) (0.025085926, 2) ] bsize=32 ]
State[ Primitive[ InfoBox[ TimeBox(0.025085926055908203,0.025085926055908203) Category=objDef( evt=160,161), Category[ Index=81, name=MPI_Send, topoState, color=(0,0,255,255,true), isUsed=tru
e, width=1, vis=true, search=true, ratios=0.0,0.0, count=0 ] ] ] (0.025085926, 2) (0.025085926, 2) ] bsize=32 ]
SLOG-2 Header:
version = SLOG 2.0.0
numOfChildrenPerNode = 2
freeLeafBytesize = 65536
maxTreeDepth = 1
maxBufferBytesize = 65536
Categories is #BInfo(580 @ 102907)
MethodDefs is #BInfo(0 @ 0)
liveTimes is #BInfo(212 @ 103482)
treeRoot is #BInfo(3220 @ 99687)
treeDir is #BInfo(180 @ 103719)
Annotations is #BInfo(0 @ 0)
postamble is #BInfo(0 @ 0)

Number of Drawables = 2825
Number of Unmatched Events = 0
Total Bytesize of the logfile = 288304
timeElapsed between 1 & 2 = 17 msec
timeElapsed between 2 & 3 = 717 msec
```

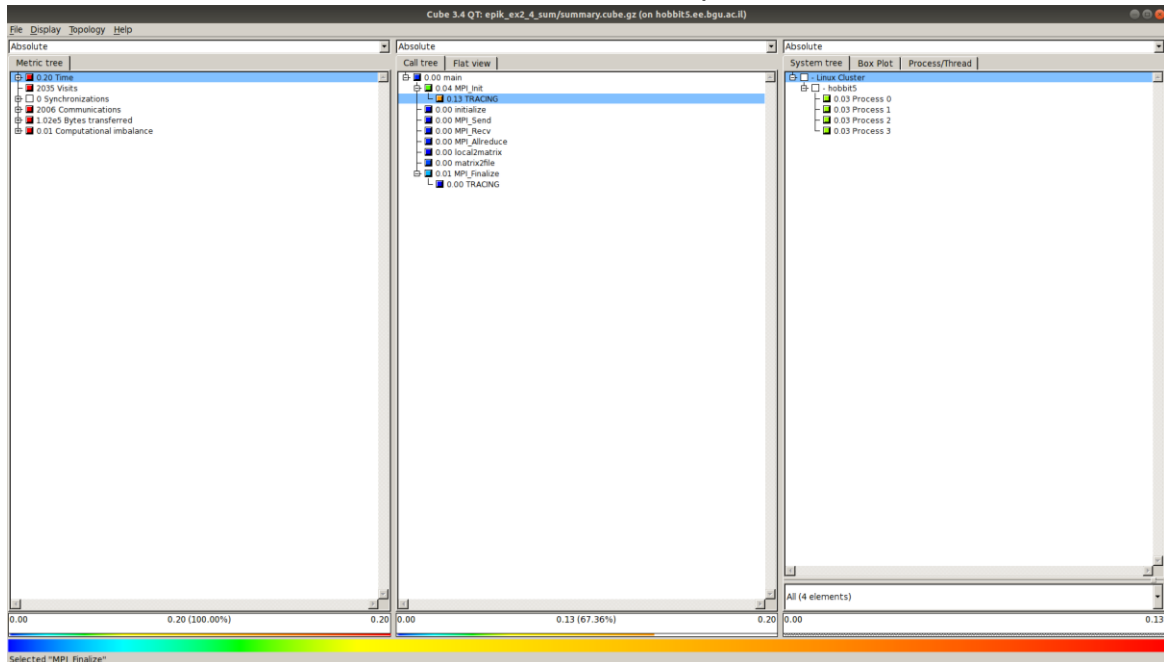
- We can see that we have many messaging between all the processes.
- Every iteration each process communicates with two other processes. It sends and receives the 50X1 array of the neighbors' points that are not included in the matrix of the current process.
- When all iterations are completed the three workers processes sends their final calculated matrix to the master process. Therefore, we see MPI\_Recv only for the master process and MPI\_Send for all the other workers processes.



## Code tracing and profiling using Scalasca:

### Running Scalasca:

5. scalasca -instrument mpicc -o exe ./main.c
6. scalasca -analyze mpirun -np 4 ./exe
7. scalasca -examine ./epik\_exe\_4\_sum



- Respectively MPI\_SEND and MPI\_Recv takes almost no time at all.
- On the other hand, MPI\_Init is being done by all processes and has significant executable time.

### Timing and Conclusions:

```
At iteration 65, diff is 1.566486e+01
At iteration 66, diff is 1.560811e+01
At iteration 67, diff is 1.554952e+01
At iteration 68, diff is 1.549595e+01
At iteration 69, diff is 1.544069e+01
At iteration 70, diff is 1.539013e+01
At iteration 71, diff is 1.533793e+01
At iteration 72, diff is 1.529013e+01
At iteration 73, diff is 1.524074e+01
At iteration 74, diff is 1.519549e+01
At iteration 75, diff is 1.514897e+01
At iteration 76, diff is 1.510575e+01
At iteration 77, diff is 1.506132e+01
At iteration 78, diff is 1.502056e+01
At iteration 79, diff is 1.497839e+01
At iteration 80, diff is 1.493937e+01
At iteration 81, diff is 1.489938e+01
At iteration 82, diff is 1.486247e+01
At iteration 83, diff is 1.482418e+01
At iteration 84, diff is 1.478998e+01
At iteration 85, diff is 1.475245e+01
At iteration 86, diff is 1.471886e+01
At iteration 87, diff is 1.468396e+01
At iteration 88, diff is 1.465186e+01
At iteration 89, diff is 1.461849e+01
At iteration 90, diff is 1.458778e+01
At iteration 91, diff is 1.455584e+01
At iteration 92, diff is 1.452643e+01
At iteration 93, diff is 1.449583e+01
At iteration 94, diff is 1.446763e+01
At iteration 95, diff is 1.443828e+01
At iteration 96, diff is 1.441123e+01
At iteration 97, diff is 1.438305e+01
At iteration 98, diff is 1.435707e+01
At iteration 99, diff is 1.433609e+01
At iteration 100, diff is 1.430503e+01
output updated
execution time = 0.107933
```

The execution time is 0.107933 seconds, meaning it took only about 108 ms to perform the total iterations of calculations and data transformation.

As the number of parallel processes increase the run time of the calculation drops. In addition, the domain decomposition allows me to transfer relatively small arrays, so the messaging time is also not high. The last send and receive of the total data calculated by each process is the only significant amount of data transferred between the processes. We can also see that in the code tracing using jumpshot.