High Performance Computing 1: Homework #2

Due Oct. 23, 2022 at 11:59 pm

Submission Instructions

Code may be written in `C`, `C++`, or `Fortran`. Clarity of code and plots will be a component of the credit you receive. Please `tar` or `zip` the components listed below into a SINGLE bundled file and submit it electronically through UBLearns. In the bundled submission please include:
(a) A SINGLE PDF file containing your results description, graphs, and any tables of the program output.
(b) Your source code, text files generated, and processing scripts.
(c) `README` file the states the contents of the `tar` or `zip` file and the steps to compile and run the different numerical experiments.

Problem 1

One of the many ways to determine the value for π is to solve the following integral:

$$\int_0^1 \frac{1}{1+x^2}\, dx = \frac{\pi}{4}.$$

Here, we will use numerical integration to estimate π, by integrating the function $f(x) = \frac{4}{1+x^2}$ over the domain $0 \leq x \leq 1$. To do this, divide the x domain (0,1) into evenly spaced increments (bins). Then, estimate the area of each local bin by solving $f(x_i)\Delta x$ for each bin. Let Δx be the width of each bin and solve $f(x_i)$ at the midpoint (x_i) of each bin. Then, sum the areas from each local bin to approximate the total area under the curve.

1. Write a serial program to numerically calculate the value for π. What is the value of pi calculated?
2. Now, parallelize your code using MPI. Be sure to test your program to make sure your results for pi are correct.
3. Using your parallelized code, conduct multiple runs (at least 3) in order to examine *strong scaling* on up to 32 cores at CCR. In your analysis, plot the number of cores on the horizontal axis and the runtime on the vertical axis.
4. Using your parallelized code, conduct multiple runs (at least 3) in order to examine *weak scaling* on up to 32 cores at CCR. In your analysis, plot the number of cores on the horizontal axis and the runtime on the vertical axis.
5. Describe the results from strong and weak scaling. What is the speedup?

* This Snipshot is the result for the Serial Code to Calculate the the value of Pie. I also compared the Calculated value of pie, with respect to the pie from Standard Library and generated error percentage from the Standard value of pie.

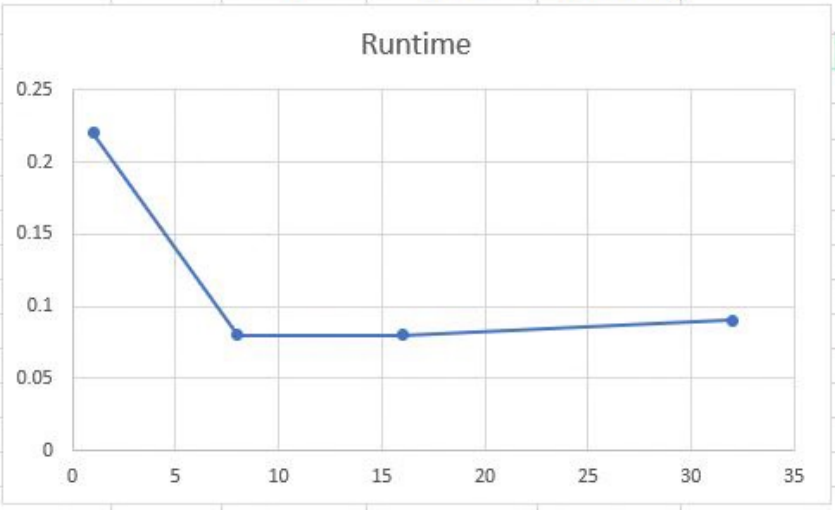
```
avishwak@srv-p22-12:~/Desktop/HPC_1_practise/hw_2/1_pi/serial_code
[avishwak@vortex1:~/Desktop/HPC_1_practise/hw_2/1_pi/serial_code]$ ls
pie  pie.cpp
[avishwak@vortex1:~/Desktop/HPC_1_practise/hw_2/1_pi/serial_code]$ g++ pie.cpp -o pie
[avishwak@vortex1:~/Desktop/HPC_1_practise/hw_2/1_pi/serial_code]$ ./pie

Value of Pie from Std lib : 3.14159
Number of intervals(bins) are : 10000
Calculated value of Pi : 3.14149
Error : 0.0100002 %
[avishwak@vortex1:~/Desktop/HPC_1_practise/hw_2/1_pi/serial_code]$
```

fig:- Serially Calculated pie

```
avishwak@srv-p22-12:~/Desktop/HPC_1_practise/hw_2/1_pi/strong_scaling
[avishwak@vortex1:~/Desktop/HPC_1_practise/hw_2/1_pi/strong_scaling]$ ls
16core  1core  32core  8core  pie.cc  qpie.sh
[avishwak@vortex1:~/Desktop/HPC_1_practise/hw_2/1_pi/strong_scaling]$ cat ./1core/JOB_pi_2_-10197186.out
------------------------------------------------------
Number of processors : 1            Nx : 100000000
STD_Pi : 3.14159
Num_Pi : 3.14159
Error : 9.99957e-07 %
Total Time : 0.220000 seconds.
Parallel Time : 0.185580969
Serial time :  0.0344190311
------------------------------------------------------
DONE !!!!
[avishwak@vortex1:~/Desktop/HPC_1_practise/hw_2/1_pi/strong_scaling]$ cat ./8core/JOB_pi_2_-10197187.out
------------------------------------------------------
Number of processors : 8            Nx : 100000000
STD_Pi : 3.14159
Num_Pi : 3.14159
Error : 9.99983e-07 %
Total Time : 0.080000 seconds.
Parallel Time : 0.0369172096
Serial time :  0.0430827904
------------------------------------------------------
DONE !!!!
[avishwak@vortex1:~/Desktop/HPC_1_practise/hw_2/1_pi/strong_scaling]$ cat ./16core/JOB_pi_2_-10197189.out
------------------------------------------------------
Number of processors : 16            Nx : 100000000
STD_Pi : 3.14159
Num_Pi : 3.14159
Error : 1.00001e-06 %
Total Time : 0.080000 seconds.
Parallel Time : 0.0218858719
Serial time :  0.0581141281
------------------------------------------------------
DONE !!!!
[avishwak@vortex1:~/Desktop/HPC_1_practise/hw_2/1_pi/strong_scaling]$ cat ./32core/JOB_pi_2_-10197190.out
------------------------------------------------------
Number of processors : 32            Nx : 100000000
STD_Pi : 3.14159
Num_Pi : 3.14159
Error : 1e-06 %
Total Time : 0.090000 seconds.
Parallel Time : 0.0145640373
Serial time :  0.0754359627
------------------------------------------------------
DONE !!!!
[avishwak@vortex1:~/Desktop/HPC_1_practise/hw_2/1_pi/strong_scaling]$
```

fig: Results for Strong Scaling

strong scaling		
No. of Cores	Runtime	speedup
1	0.22	1
8	0.08	2.75
16	0.08	2.75
32	0.09	2.444444444



* For Strong Scaling as observed,
As the No. of Cores increases for a fixed Problem Size, the runtime decreases until it reaches the Saturation point, beyond which the time does not reduce any further.

* talking about Speed up of Strong Scaling, as we can see, as the number of Cores increases Speed up factor increases, until it reaches a Saturation point.

⇒ Results for Weak Scaling

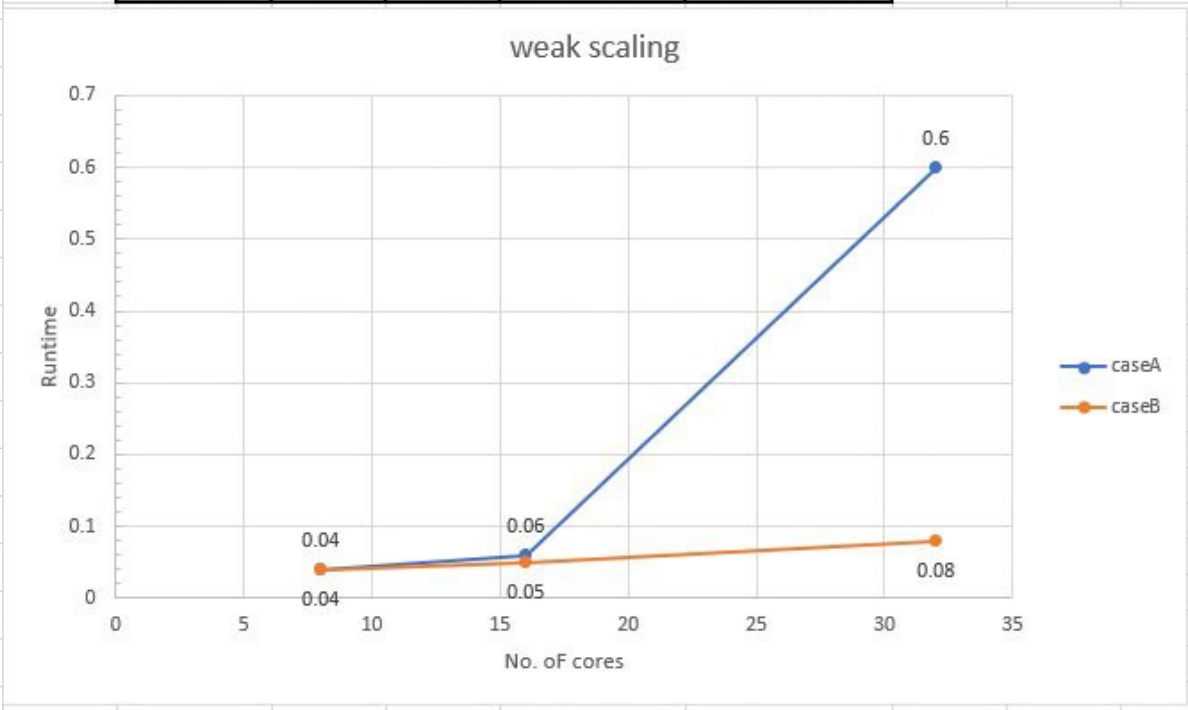
Case A :

```
Host: vortex.cbls.ccr.buffalo.edu
[avishwak@vortex1:~/Desktop/HPC_1_practise/hw_2/1_pi/weak_scaling]$ pwd
/user/avishwak/Desktop/HPC_1_practise/hw_2/1_pi/weak_scaling
[avishwak@vortex1:~/Desktop/HPC_1_practise/hw_2/1_pi/weak_scaling]$ cat ./8core/JOB_pi_2_-10173975.out
-----------------------------------------------------
Number of processors : 8           Nx : 100000
STD_Pi : 3.14159
Num_Pi : 3.14158
Error : 0.001 %
Total Time : 0.040000 seconds.
Parallel Time : 0.000911951065
Serial time :  0.0390880489
-----------------------------------------------------
DONE !!!!
[avishwak@vortex1:~/Desktop/HPC_1_practise/hw_2/1_pi/weak_scaling]$ cat ./16core/JOB_pi_2_-10174001.out
-----------------------------------------------------
Number of processors : 16           Nx : 1000000
STD_Pi : 3.14159
Num_Pi : 3.14159
Error : 0.0001 %
Total Time : 0.060000 seconds.
Parallel Time : 0.00739598274
Serial time :  0.0526040173
-----------------------------------------------------
DONE !!!!
[avishwak@vortex1:~/Desktop/HPC_1_practise/hw_2/1_pi/weak_scaling]$ cat ./32core/JOB_pi_2_-10173977.out
-----------------------------------------------------
Number of processors : 32           Nx : 100000000
STD_Pi : 3.14159
Num_Pi : 3.14159
Error : 1e-06 %
Total Time : 0.600000 seconds.
Parallel Time : 0.528697968
Serial time :  0.0713020325
-----------------------------------------------------
DONE !!!!
[avishwak@vortex1:~/Desktop/HPC_1_practise/hw_2/1_pi/weak_scaling]$
```

Case B:

```
avishwak@srv-p22-12:~/Desktop/HPC_1_practise/hw_2/1_pi/weak_scaling_caseB
[avishwak@vortex1:~/Desktop/HPC_1_practise/hw_2/1_pi/weak_scaling_caseB]$ ls
16core  32core  8core
[avishwak@vortex1:~/Desktop/HPC_1_practise/hw_2/1_pi/weak_scaling_caseB]$ cat ./8core/JOB_pi_2_-10195461.out
-----------------------------------------------------
Number of processors : 8           Nx : 100000
STD_Pi : 3.14159
Num_Pi : 3.14158
Error : 0.001 %
Total Time : 0.040000 seconds.
Parallel Time : 0.00089097023
Serial time :  0.0391090298
-----------------------------------------------------
DONE !!!!
[avishwak@vortex1:~/Desktop/HPC_1_practise/hw_2/1_pi/weak_scaling_caseB]$ cat ./16core/JOB_pi_2_-10195497.out
-----------------------------------------------------
Number of processors : 16           Nx : 200000
STD_Pi : 3.14159
Num_Pi : 3.14159
Error : 0.0005 %
Total Time : 0.050000 seconds.
Parallel Time : 0.00196504593
Serial time :  0.0480349541
-----------------------------------------------------
DONE !!!!
[avishwak@vortex1:~/Desktop/HPC_1_practise/hw_2/1_pi/weak_scaling_caseB]$ cat ./32core/JOB_pi_2_-101954
JOB_pi_2_-10195482.err  JOB_pi_2_-10195482.out  JOB_pi_2_-10195495.err  JOB_pi_2_-10195495.out
[avishwak@vortex1:~/Desktop/HPC_1_practise/hw_2/1_pi/weak_scaling_caseB]$ cat ./32core/JOB_pi_2_-10195482.out
-----------------------------------------------------
Number of processors : 32           Nx : 400000
STD_Pi : 3.14159
Num_Pi : 3.14159
Error : 0.00025 %
Total Time : 0.080000 seconds.
Parallel Time : 0.00301098824
Serial time :  0.0769890118
-----------------------------------------------------
DONE !!!!
[avishwak@vortex1:~/Desktop/HPC_1_practise/hw_2/1_pi/weak_scaling_caseB]$
```

weak scaling					
	No. of core	Runtime	No. of bins	No.of bin per core	speedup
Case A	8	0.04	1.00E+05	1.25E+04	0.75
	16	0.06	1.00E+06	6.25E+04	0.5
	32	0.6	1.00E+08	3.13E+06	0.05
Case B	8	0.04	1.00E+05	1.25E+04	0.75
	16	0.05	2.00E+05	1.25E+04	0.6
	32	0.08	4.00E+05	1.25E+04	0.375
Case 0	1	0.03	1.25E+04	1.25E+04	



* For Weak Scaling, I ran 2 Cases:

  Case 1: Increasing number of Cores, also increasing number of bins (Elements) to Calculate the integration (i.e. Value of pie) where Number of bin per Core also increases.

  Case 2: Increasing number of Cores, also increasing number of bins (Elements) to Calculate the integration (i.e. Value of pie), but here we are keeping Number of bins per Core as Constant.

* On observing we can see if the number of bins per core increases we get increasing graph.
* And if Number of bins per Core is kept Constant, we have almost Constant graph (Slight variation in result is possibly due to hardware overhead's while Compiling)
* Speed up for Weak Scale decreases, due to it's increasing trend as observed from Case A.

Problem 2

Use MPI to parallelize the program that you wrote for Homework #1 that solves for the area of the Mandelbrot set, where x spans $[-2, 2]$ and y spans $[-1, 1]$. There are many ways of partitioning the work (i.e., how to divide the problem over the number of processes). For this exercise, partition the complex plane into contiguous rectangular blocks.

Part A:

1. Fix the grid size to a high resolution. Split the domain into two equal subdomains, divided evenly about the *vertical* axis. Assign each partition of the model to a separate MPI process, making a total of two MPI processes for the problem.
2. Estimate the area of the Mandelbrot set.
3. Study the load balancing. Record the time it took for each process. Divide the maximum time by the minimum time. How do the times on each process compare?
4. Plot the results for each subdomain of Mandelbrot set separately. Then, plot the Mandelbrot set for the whole domain.

Part B:

1. Using the same total size for the high-resolution grid above, now split the domain into four equal subdomains, divided evenly about the horizontal axis. Assign each partition of the model to a separate MPI process, making a total of four MPI processes for the problem.
2. Estimate the area of the Mandelbrot set.
3. Study the load balancing. Record the time it took for each process. Divide the maximum time by the minimum time. How do the times on each process compare?
4. Plot the results for each subdomain of Mandelbrot set separately. Then, plot the Mandelbrot set for the whole domain.
5. Describe the results. Compare the timing results in Parts A and B.

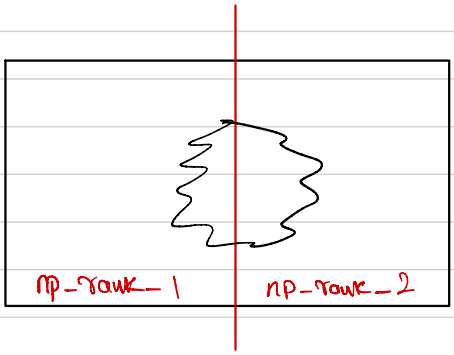
Problem 3

Answer the following using Amdahl's Law, $T_p = T_1 F_s + \frac{T_1}{p} F_p$. Let $T_1 = 4000$ hours, $F_s = .10$, and $F_p = .90$. Show your work.

1. Calculate the parallel execution time, T_p, for a simulation run on 96 cores:

2. Is perfect speedup achieved? Why or why not.

Problem 2:- PART A.



* We basically have to split Mandelbrot domain in two parts & assign each domain to one processor. & Collect results from each part & form a whole picture.

* The way, I wrote my Code is, I am calling for 3 processor. & their tasks are as follows,

⇒ Processor rank == 2 ⇒ Calculate & plot the Right-hand Side of Mandelbrot & Send all the information to rank == 0

⇒ Processor rank == 1 ⇒ Calculate & plot the Left-hand Side of Mandelbrot & Send all information to rank == 0

⇒ Processor rank == 0 ⇒ Collect information from Rank == 1 & Rank == 2 & plot a final image.

On running my Code, I get results as follows:-

```
avishwak@srv-p22-13:~/Desktop/HPC_1_practise/hw_2/2a_mandelbrot/trial_4
[avishwak@vortex2:~/Desktop/HPC_1_practise/hw_2/2a_mandelbrot/trial_4]$ ls
JOB_pi_2_-10197669.err  mandelbrot_2.mpi                              mandelbrot_set_grid_1000x800_rank_rank_2.ppm
JOB_pi_2_-10197669.out  mandelbrot_set_grid_1000x800_rank_rank_0.ppm  qmandelbrot.sh
mandelbrot_2.cc         mandelbrot_set_grid_1000x800_rank_rank_1.ppm
[avishwak@vortex2:~/Desktop/HPC_1_practise/hw_2/2a_mandelbrot/trial_4]$ cat JOB_pi_2_-10197669.out
From Rank : 2      count1 : 40862
From Rank : 2      Time Taken for calculation : 1.51699 seconds

From Rank : 1      count1 : 109919
From Rank : 1      Time Taken for calculation : 3.92993 seconds


From Rank : 0
Num of process : 3

fraction of mandelbrot Set is 0.188476
Area of Mandelbrot is 1.50781
Range in x-axis (Real axis): xmin = -2 ; xmax = 2
Range in y-axis (img axis) : ymin = -1 ; ymax = 1
Length in x-axis : 4
Length in y-axis (img axis) : 2
Number of grid points in x:1000
Number of grid points in y:800
Area of mandlebrot is: 1.50781
DONE !!!!
[avishwak@vortex2:~/Desktop/HPC_1_practise/hw_2/2a_mandelbrot/trial_4]$
```

* As Observed My Area = 1.50781, which is inline with my results from Homework 1.

* Now Notice time taken by each processor,

rank == 1 {L.H.S} time = 3.92993 s

rank == 2 {R.H.S} time = 1.51699 s

* Ratio of Max time to Min time = $\frac{t_{max}}{t_{min}} = \frac{3.92993}{1.51699} = 2.5906$

* Results from Each Processors are,

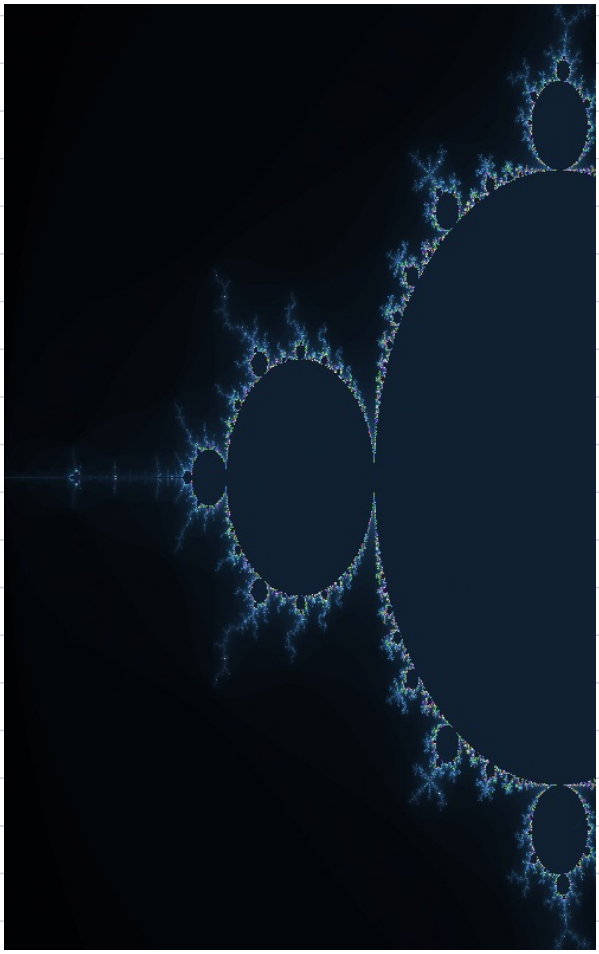


fig: Result from Rank 1

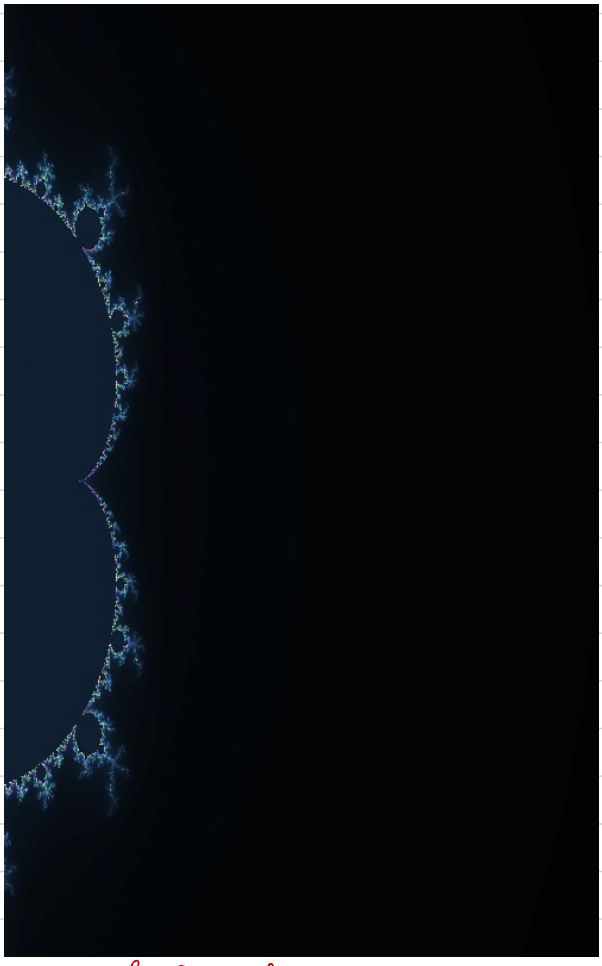


fig: Result from Rank 2

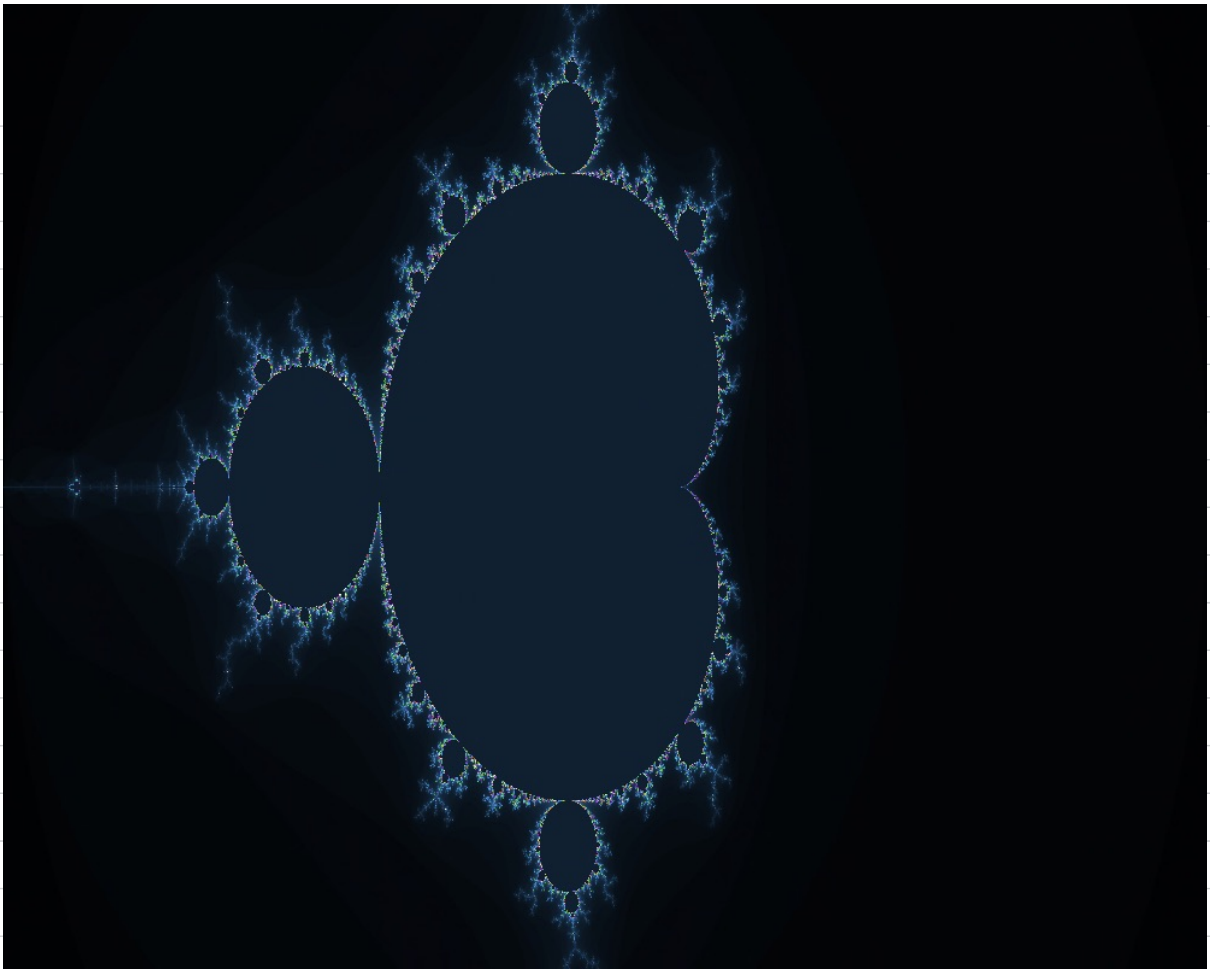
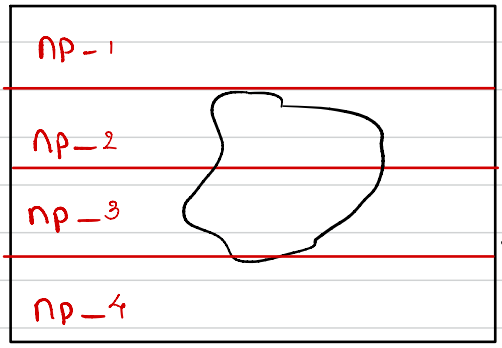


fig: Result from Rank 0

PART B

PART B:



* For Part B, we have to split the domain into 4 parts,
* The way I Coded, I call for 5 processor's & rank ==1, 2, 3 & 4 will Calculate the each divided Section's of the domain, & also send those Calculated information to the processor, with rank ==0.
* Rank ==0 takes information from all the 4 processes & then form the final picture.

```
avishwak@srv-p22-13:~/Desktop/HPC_1_practise/hw_2/2b_mandelbrot/trial_2_4split
[avishwak@vortex2:~/Desktop/HPC_1_practise/hw_2/2b_mandelbrot/trial_2_4split]$ ls
JOB_mandelbrot_-10198625.err                mandelbrot_set_grid_1000x800_rank_rank_1.ppm
JOB_mandelbrot_-10198625.out                mandelbrot_set_grid_1000x800_rank_rank_2.ppm
mandelbrot.cc                               mandelbrot_set_grid_1000x800_rank_rank_3.ppm
mandelbrot.mpi                              mandelbrot_set_grid_1000x800_rank_rank_4.ppm
mandelbrot_set_grid_1000x800_rank_rank_0.ppm  qmandelbrot.sh
[avishwak@vortex2:~/Desktop/HPC_1_practise/hw_2/2b_mandelbrot/trial_2_4split]$ cat JOB_mandelbrot_-10198625.out
From Rank : 4      count2 : 12692
From Rank : 4      Time Taken for calculation : 0.526912 seconds

From Rank : 1      count1 : 12481
From Rank : 1      Time Taken for calculation : 0.563581 seconds

From rank : 0 ::   Count1 received from rank : 1
From Rank : 2      count2 : 62628
From Rank : 2      Time Taken for calculation : 2.28357 seconds

From rank : 0 ::   Count2 received from rank : 2
From Rank : 3      count2 : 62980
From Rank : 3      Time Taken for calculation : 2.36419 seconds

From rank : 0 ::  Count3 received from rank : 3
From rank : 0 ::   Count4 received from rank : 4

From rank : 0
Total Number of process : 5
fraction of mandelbrot Set is 0.188476
Area of Mandelbrot is 1.50781
Range in x-axis (Real axis): xmin = -2 ; xmax = 2
Range in y-axis (img axis) : ymin = -1 ; ymax = 1
Length in x-axis : 4
Length in y-axis (img axis) : 2
Number of grid points in x:1000
Number of grid points in y:800
Area of mandlebrot is: 1.50781
DONE !!!!
[avishwak@vortex2:~/Desktop/HPC_1_practise/hw_2/2b_mandelbrot/trial_2_4split]$
```

* As Observed My Area = 1.50781 , which is inline with my results from Homework 1.

* Now Notice time taken by each processor,

rank == 1 {First Domain} time 1 = 0.563581 Seconds
rank == 2 {Second Domain} time 2 = 2.28357 Seconds
rank == 3 {Third Domain} time 3 = 2.36419 Seconds
rank == 4 {Fourth Domain} time 4 = 0.526912 Seconds

* Ratio of Max time to Min time $= \frac{t_{max}}{t_{min}} = \frac{time3}{time4} = \frac{2.36419}{0.526912} = 4.4868$

* From Part A & Part B, & on observing time of each domain, we can say that the domain where values tends to infinity, they tend to take major fraction of time to calculate

* Results from each Processors are :-

fig: Result from rank 1 ⇒

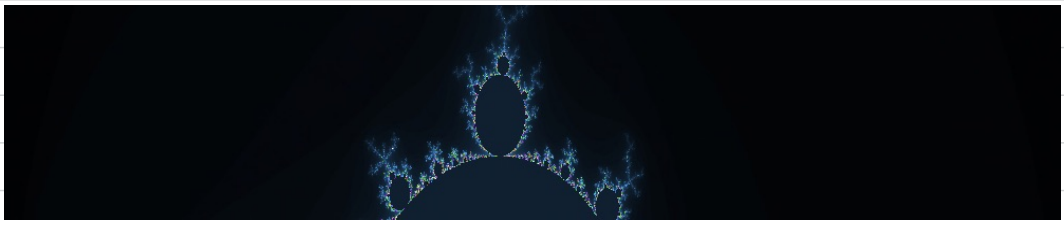


fig: Result from rank 2 ⇒

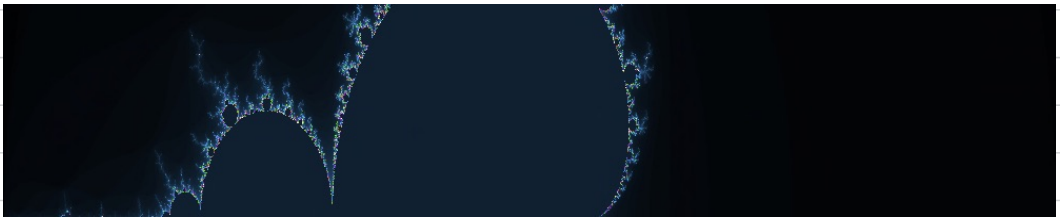


fig: Result from rank 3 ⇒

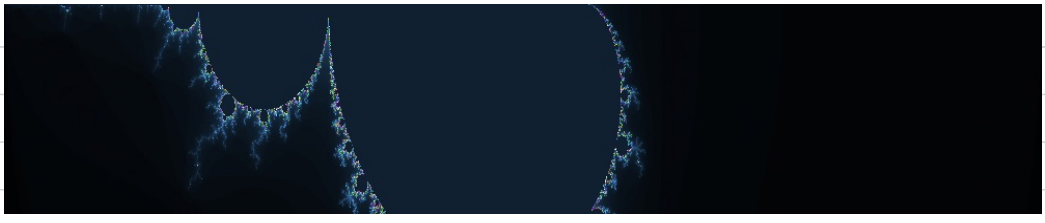
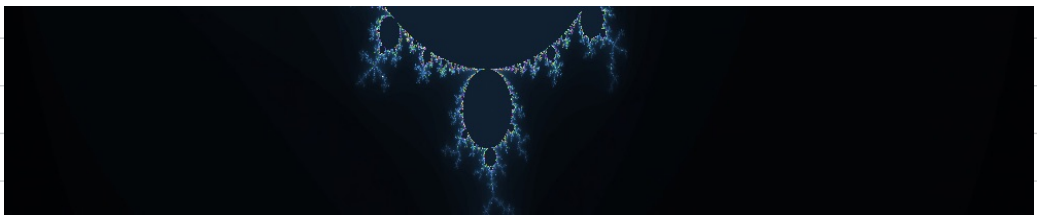


fig: Result from rank 4 ⇒



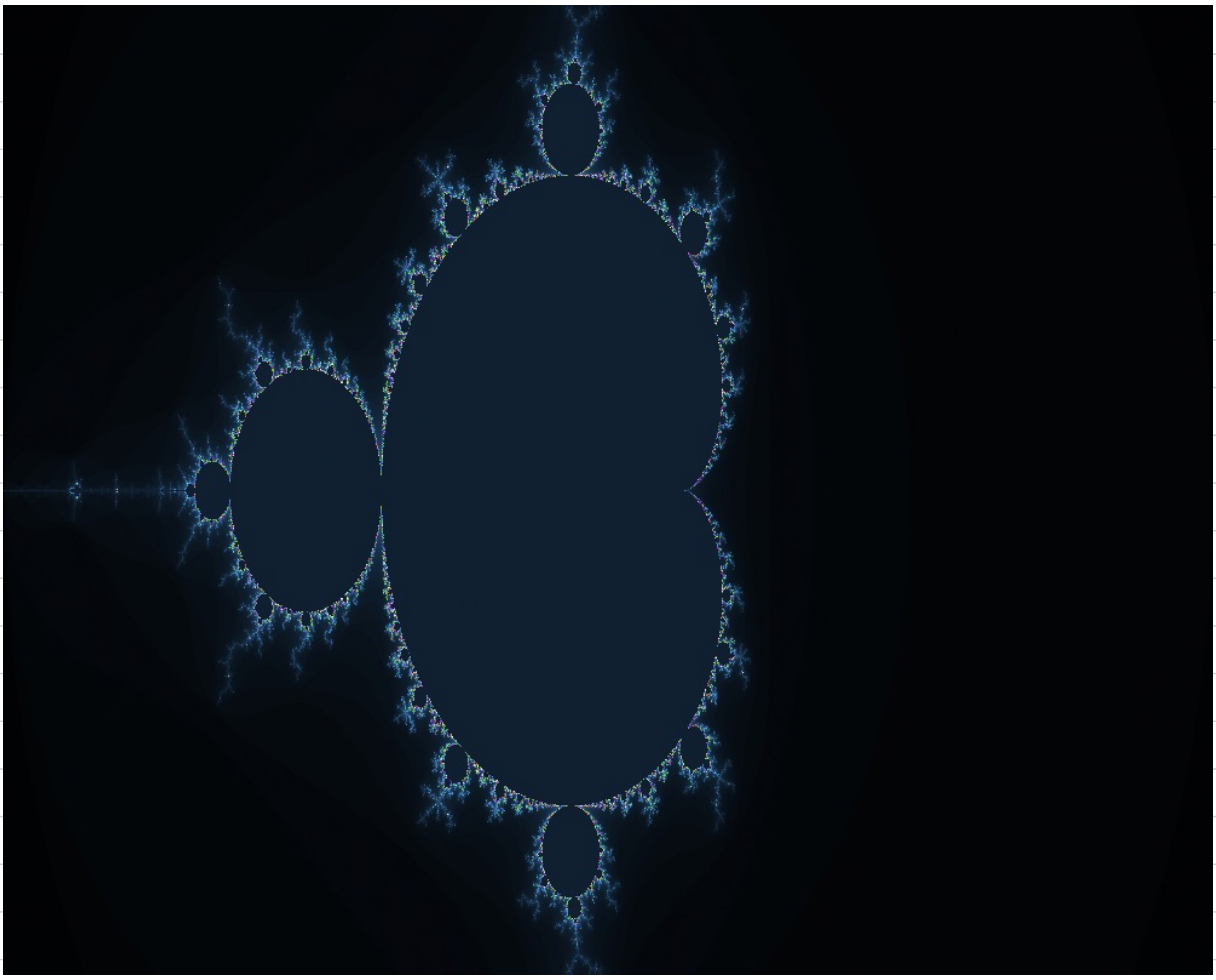


fig: Result from Rank 0

Problem 3

Answer the following using Amdahl's Law, $T_p = T_1 F_s + \frac{T_1}{p} F_p$. Let $T_1 = 4000$ hours, $F_s = .10$, and $F_p = .90$. Show your work.

1. Calculate the parallel execution time, T_p, for a simulation run on 96 cores:

2. Is perfect speedup achieved? Why or why not.

0.3] ① Given :- $T_1 = 4000$ hours

$F_s = 0.10$

$F_p = 0.90$

By Amdahl's Law, $T_p = T_1 F_s + \frac{T_1}{P} F_p$

$= 4000\,(0.10) + \frac{4000}{96}\,(0.90)$

$= 437.5$

② Speed up ⇒ , $S = \frac{\text{Single processor execution time}}{\text{Multi processor execution time}}$

$= \frac{4000}{437.5}$

$S = 9.1428$

For ideal case, we can say $T_{p,\,ideal} = \frac{T_1}{P} = \frac{4000}{96} = 41.6667$

Then, we can say, Efficiency, $E = \frac{S}{P}$

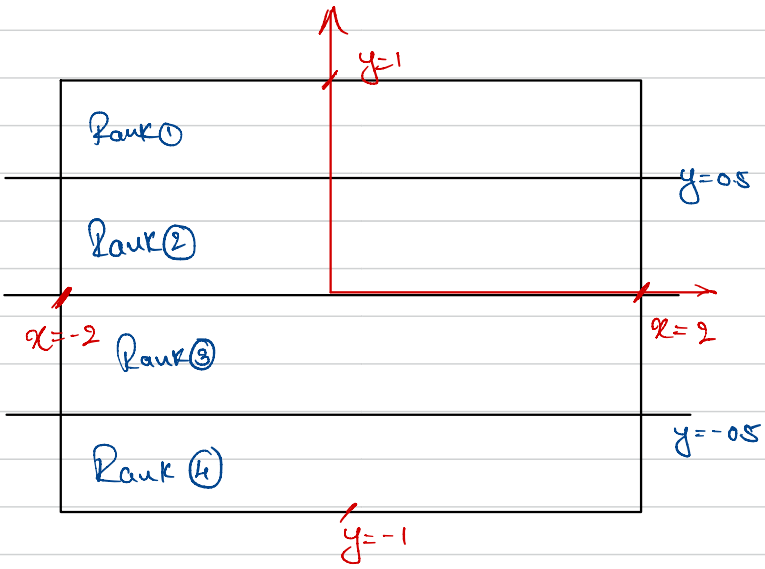
$= \frac{9.1428}{96}$

$= 0.09523$

$= 9.523\ \%$

So, 9.523 % efficiency is very less & we can say computational resources are not used optimally, using fewer number of cores could increases the efficiency & also free-up lot many cores to be used for other purposes.

Rough Page.



Q] What is "tag" in MPI_Send & MPI_Recv.?

Q] How to broadcast Multiple "variables."? → MPI_SCATTER & MPI_GATHER.

Q] Submitt Multiple Jobs on the SBATCH.? Changing Nx. → 10, 100, 1000, ⋮

Q] graph result discussion? 1st Runtime? total Runtime?