University of Tromsø

Department of computer science

INF-3201: Parallel Programming

Assignment 1
Mandelbrot with MPI

Camilla Stormoen, cst042

September 22, 2016

1 Content

2 Introduction

This report describes the implementation of the Mandelbrot set using MPI and a discussion of different approach when implementing static and dynamic MPI.

2.1 Requirements

- First requirement is to implement two parallel versions of the given Mandelbrot code, using MPI. One is using a static load balance scheme, the other using a dynamic scheme. None of them should use a shared memory.

- Next requirement is that the solution cannot use any shortcuts that reduce the functionality of the program, it must produce the same CRC and all the data has to be sent to the master node.

- Last requirement was to evaluate the implementation using profiling and tracing tools.

3 Technical Background

3.1 Uvrocks and Cluster

A computer cluster consists of connected computers that work together, but they can be viewed as a single system. Computer clusters have each node set to perform the same task, controlled and scheduled by the software. [1]

Uvrocks is a small cluster at UiT.

3.2 Mandelbrot

The Mandelbrot set is a set of complex numbers. All the complex numbers c in the sequence is limited, meaning it will not diverge when iterating from z=0. [2].

$$z_o = 0, \qquad c = complex\ number$$

$$z_{n+1} = z_n^2 + c$$

Figure 1 shows how Mandelbrot can be viewed as an image with a set of zoom sequences.

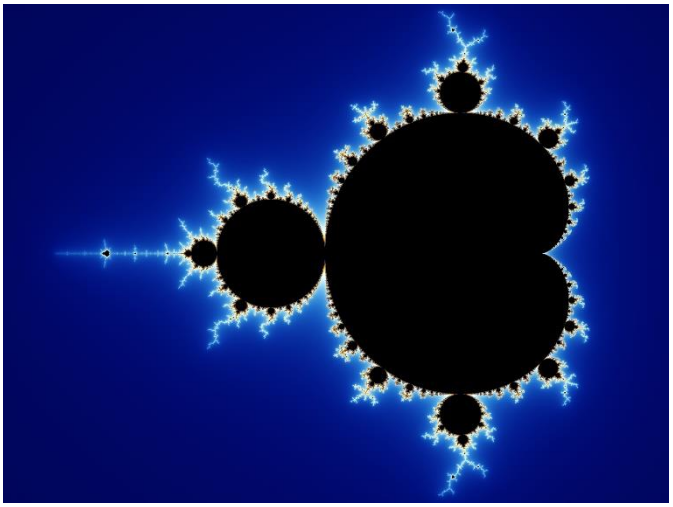

Figure 1 *Shows how Mandelbrot can be viewed as an image*

3.3 MPI – Message Passing Interface

Message Passing Interface is a standardized and portable message-passing system designed to function on a wide variety of parallel computers. [3]

MPI was originally designed for distributed memory architectures. There are many reasons for using MPI, like standardization, portability, performance opportunities, functionality and availability.

3.3.1 Static MPI

In static MPI are all the processes specified before execution and the system will execute a fixed number of processes.

3.3.2 Dynamic MPI

In dynamic MPI are processes and their execution happen during the execution of other processes.

4 Design

4.1 Static MPI

The static MPI starts with all of the processors being determined at the beginning of the execution. The workload of each processor is divided into rows, so there is one row

per processor as seen in Figure 2. The slaves divide the work and calculate their given row and send their result back to the master.

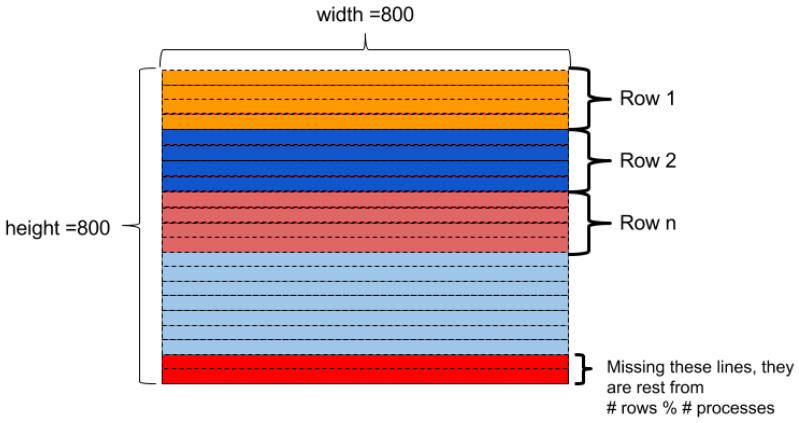

Figure 2 *Shows how the design of the static MPI implementation*

If there are an odd number of processors, some lines will not be calculated because they are not assigned to any processor. Therefor must the last processor take a bigger amount of lines than the other processors.

4.2 Dynamic MPI

In dynamic MPI will the work load be distributed among the processors at runtime. The master will first assign a task to each slave, and they will calculate their task and return the result. When they return, and if there are more packets to be sent, they are assigned a new task. This routine will go on until all the workload has been calculated and returned to master.

The workload is divided into squares, as seen in Figure 3. These squares are regulated by a size and is given a package-number. When the slave gets a packet, they will calculate where to start by the given packet-number.

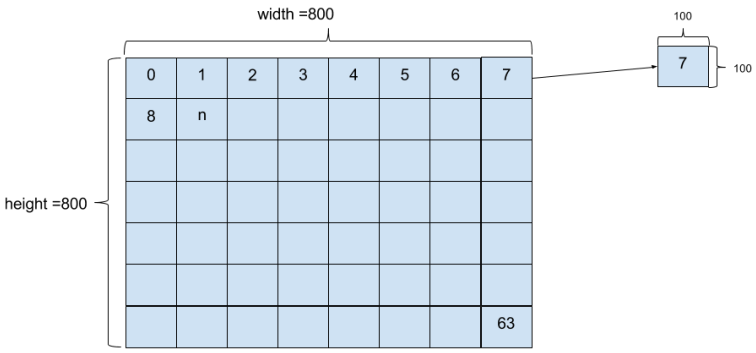

Figure 3 *Shows how the design off the dynamic MPI implementation*

5 Implementation

The implementation is written in C language and run on a cluster (uvrocks) at UiT, executed using the Linux terminal from a computer. The testing and profiling is provided from the source-code and the output-file is examined through Vampir.

6 Discussion

6.1 Static MPI

In the static MPI is design with one row to each process. However, it could also be separated into either per pixel or in squares. Dividing the workload into squares would probably be the best solution because of the workload in the middle of the image. The processor that have the centre of the Mandelbrot image as workload would have extremely more work to calculate than the processor that is on the edge of the picture.

The overall goal to the static MPI is to reduce the execution time of a concurrent program while minimizing the communication delays, but it has the disadvantage that the workload is being set at the beginning when the process is created and cannot be changed during process execution to make changes in the system load.

6.2 Dynamic MPI

The dynamic MPI has also the choice between separating the image with either squares, rows or pixels. The version, which is described in the design section, is supposable the best choice of these three choices.

When each chunk is being divided into squares with same size, will the processors with chunks that have less calculations receive more packets than the processors with packets with big calculations. The middle of the Mandelbrot image will here as well, be the point where there is most calculations. It will therefore be more work for the chunks in the middle than the chunks on the sides.

6.3 Evaluation

All the tests are run 10 times and the average is plotted in the graphs. Even though the tests are the average off 10 executions each, will there be an overhead since the cluster was used by other students at the same time. The best time was the static MPI run on a quad-processor with 20 or 40 hosts.

6.3.1 Execution with dual core

The graphs from Figure 4 shows that the sequential code use around 15-16 seconds to execute while the three others are around the same area. The static and the dynamic (where it's 400 packets with a size of size 1600) use approximately the same time at 4 nodes, but the static MPI is much faster if the number of hosts increase while the dynamic is quite steady around 2-3 seconds.

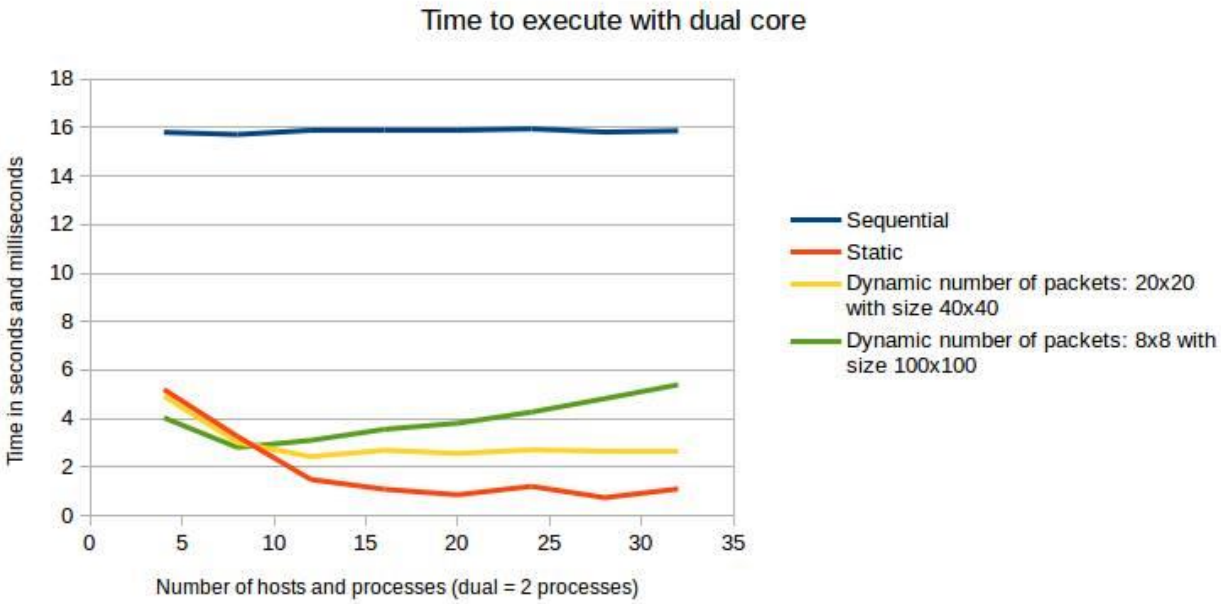

Figure 4 *Shows the time to execute with dual core*

6.3.2 Execution with quad core

Figure 5 shows the execution time with quad core and here as well will the sequential implementation have around 15-16 seconds execution-time. The static MPI will also here be the best solution, except if number of hosts is over 50.

The dynamic MPI with 64 packets with size 1000 will not execute after 16 nodes because there will be more processes than packages to send and calculate.

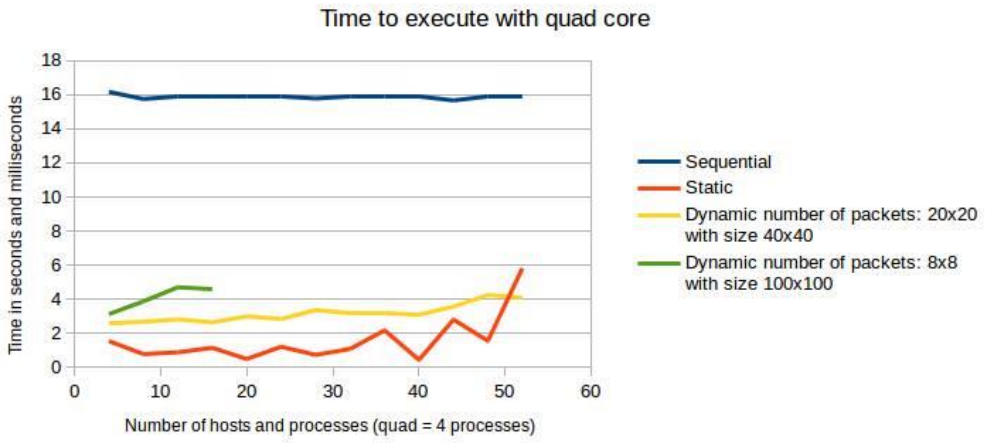

Figure 5 *Shows the time to execute with quad core*

6.3.3 Execution with all the processors

Figure 6 shows the time to execute Mandelbrot with all the processors. It differs from the other graphs where in this figure, the dynamic MPI where number of packets are 400 with a size of 1600, is the fastest implementation. The other dynamic MPI will not execute because of the problems with more processes than workload.

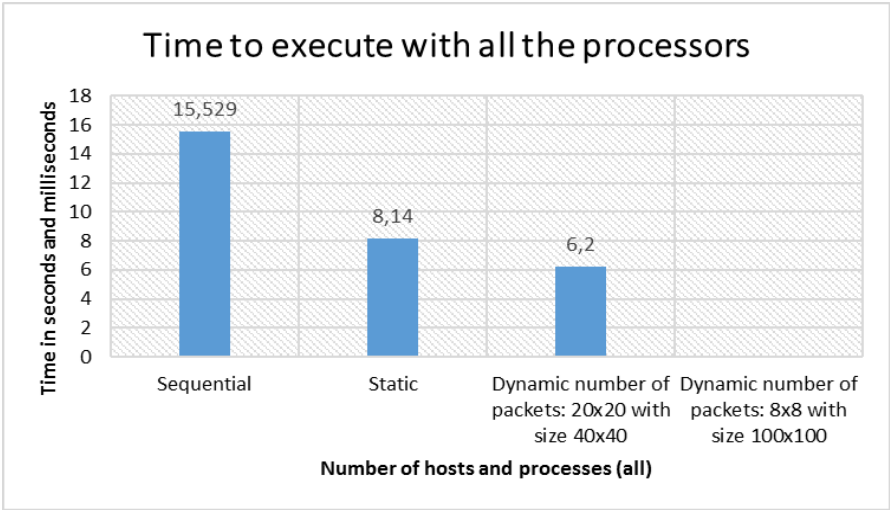

Figure 6 *Shows the time to execute with all the processors*

6.3.4 Evaluating work load with Vampir-tool

When profiling, the output is examined by the Vampir-tool, but this is only a demo, so it won't trace the whole file. Figure 7 and Figure 8 shows the tracing, and process 0 is master, so it is correct that this process has mostly receives and not many send. As for the other processes, they are slaves, so they would have a lot of calculations and then send-operations like in the figures.

As Figure 8 also shows, is that most of the time executing is used in the slaves when they are calculating the result.

By this little piece of tracing, it looks like master and slaves work as they should.

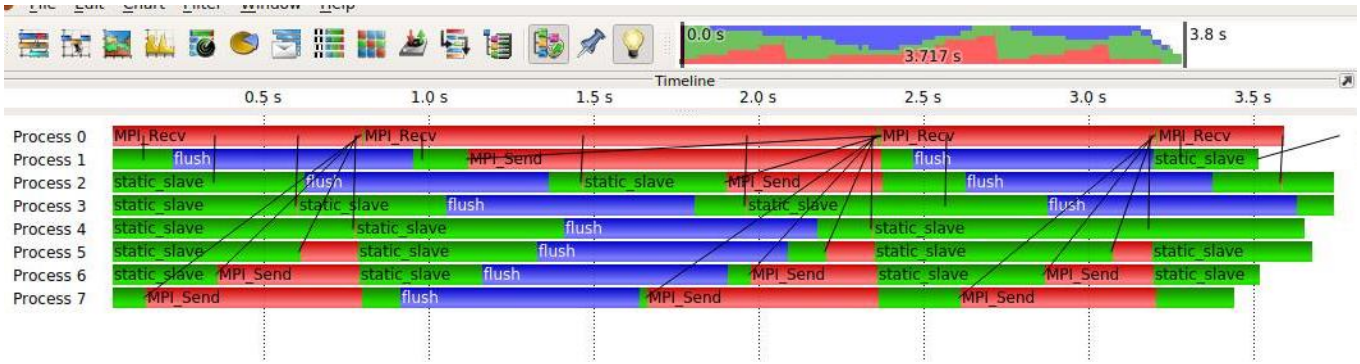

Figure 7 *Shows output from profiling in Vampir*

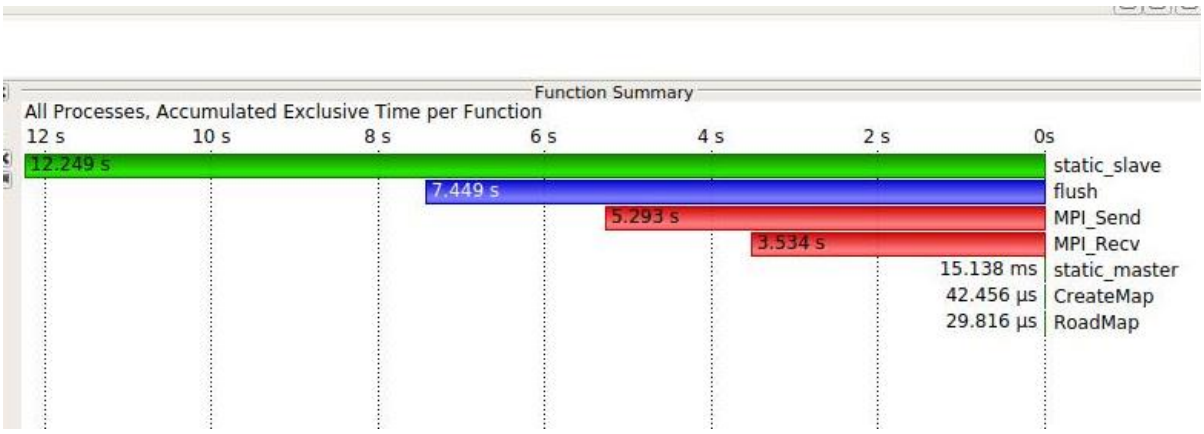

Figure 8 *Shows output from profiling in Vampir*

7 Conclusion

This report has described how to implement Mandelbrot using static- and dynamic MPI, and has also discussed which approach was the fastest. The best result was the static MPI that was executed on a quad-processor with 20 or 40 hosts.

8 References

[1] W. contributors, "Computer cluster," Wikipedia, The Free Encyclopedia., 12 September 2015. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Computer_cluster&oldid=680736707. [Accessed 05 November 2015].

[2] W. contributors, "Mandelbrot set," Wikipedia, The Free Encyclopedia., 6 September 2016. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Mandelbrot_set&oldid=737976926. [Accessed 22 September 2016].

[3] Wikipedia contributors, "Message Passing Interface," Wikipedia, The Free Encyclopedia., 21 September 2016. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Message_Passing_Interface&oldid=740555563. [Accessed 22 September 2016].