DATA STRUCTURES AND ALGORITHMS MULTI-THREADED MANDLEBROT

School of arts media and computer games

Computer games applications development

Andrew Peters

PURPOSE OF THE APPLICATION

The purpose of the application is to be able to run the mandlebrot set using different threads working in parallel.

The task creation and computing the image should be done in parallel in order t speed up the program.

THE USE OF THREADS

The threads are very important in this application because they allow the different jobs to be carried out at the same time. If the threads were not there then the program would have to wait for each slice of the mandlebrot to be made instead of doing it all at one time.

HOW IS IT PARALLEL

There is one function that gets called in main which runs the two threaded functions.

A task will be put into the task queue using one of the thread functions

The other thread function takes the task at the front of the queue and computes it.

They will both run together until the image has been competed.

HOW THE THREADS COMMUNICATE

The threads are communicating through a condition variable. This has been done by using a unique lock and making the worker thread to wait. A task will be put into the queue by the add task method and notify one of the threads waiting that there is a task ready. This then allows the worker thread to take a task and run it.

CPU SPECIFICATION

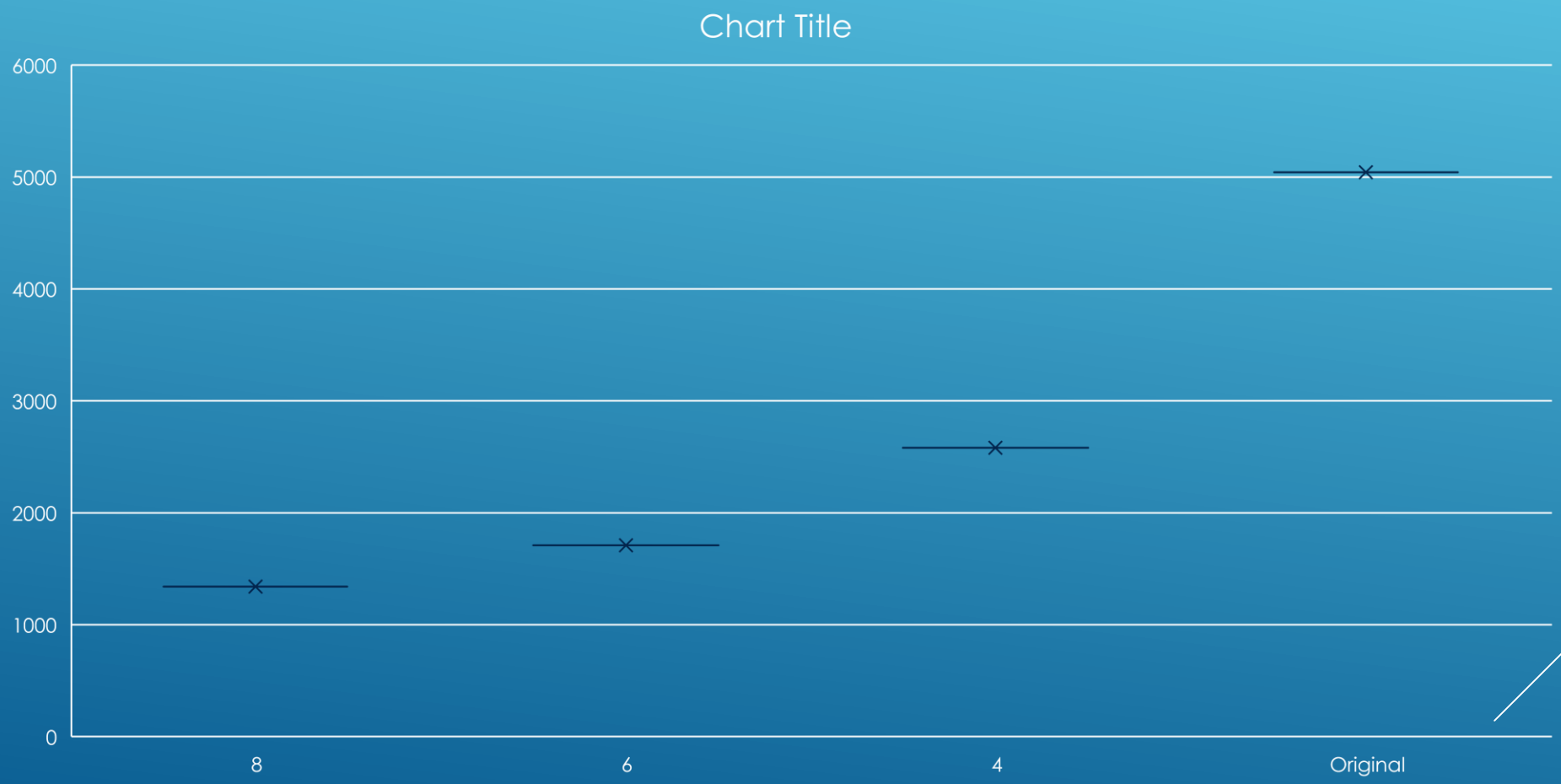
Intel core i5 2400

3.4ghz

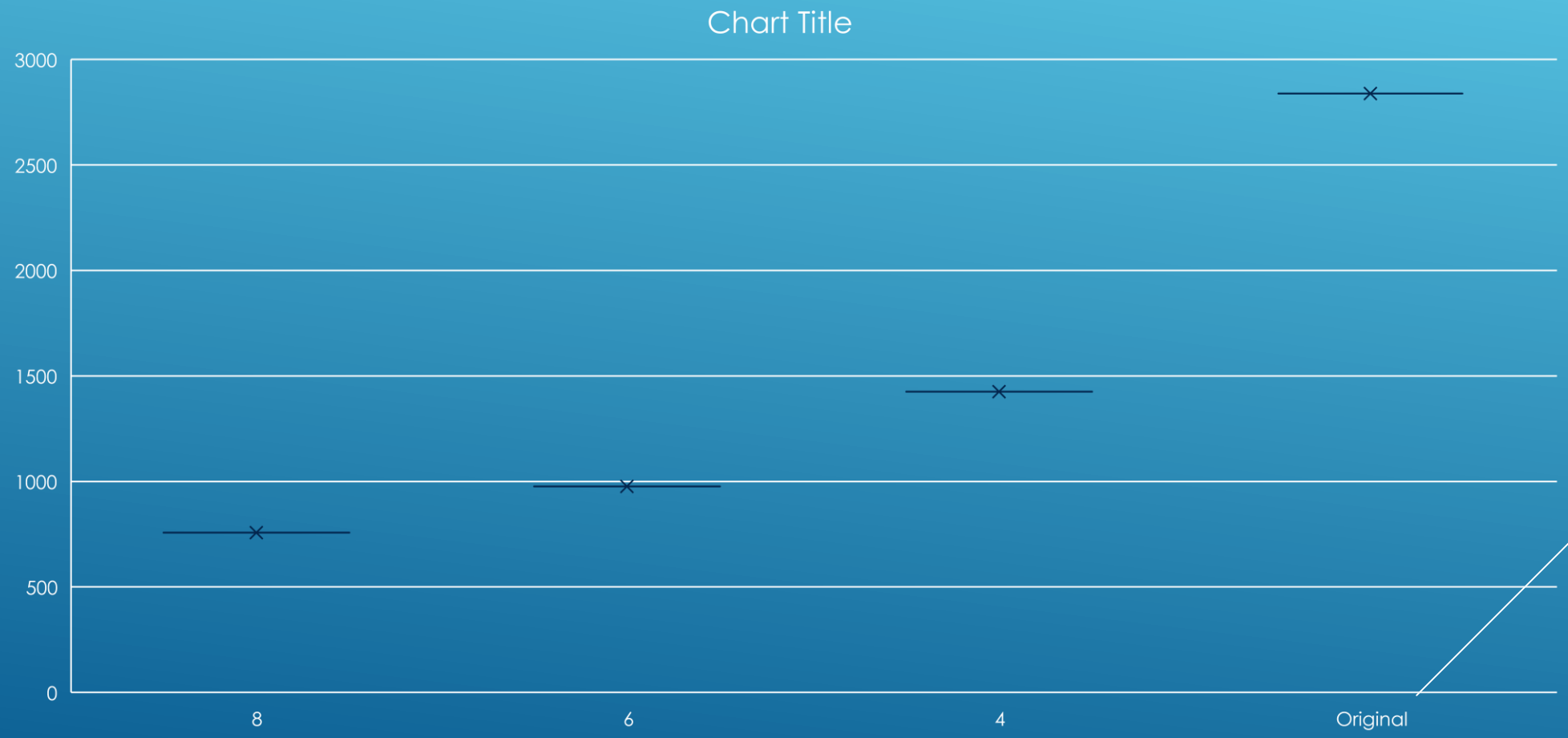
4 cores

8 threads

RESOLUTION 1024X576 500 ITERATIONS



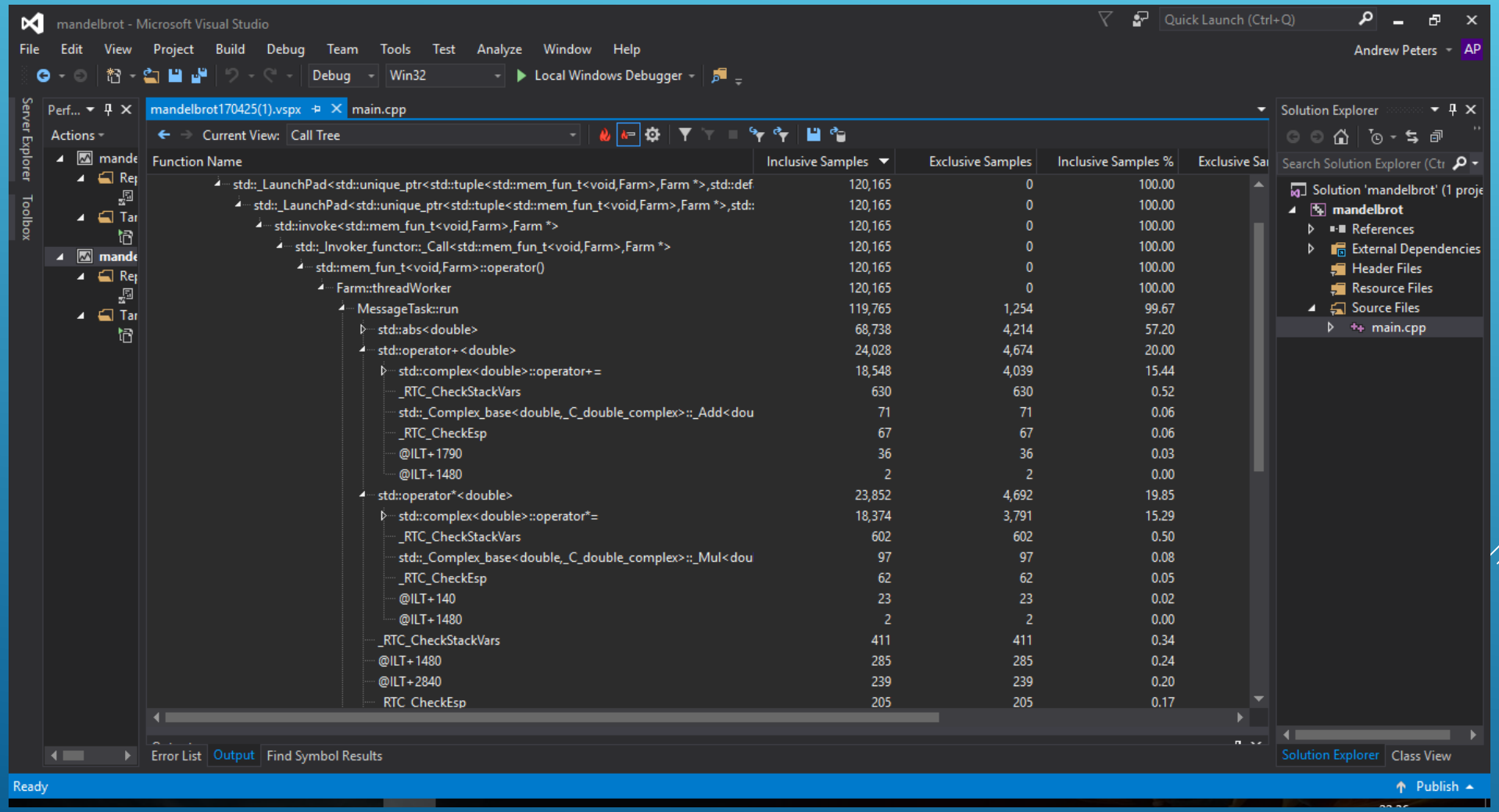
RESOLUTION 2048X1124 50 ITERATIONS



WHAT THE DATA SHOWS

This previous images shows how much faster the program runs after parallelisation and increasing speeds with the amount of threads being used. The difference in speed begins to slow down after using 6 threads this is because it is almost at is optimum speed.

PROFILER


mandelbrot - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Andrew Peters
Debug
Win32
Local Windows Debugger
mandelbrot170425(1).vspx
main.cpp
Current View: Call Tree
Function Name
Inclusive Samples
Exclusive Samples
Inclusive Samples %
Exclusive Sa
std::_LaunchPad<std::unique_ptr<std::tuple<std::mem_fun_t<void,Farm>,Farm *>,std::def 120,165 0 100.00
std::_LaunchPad<std::unique_ptr<std::tuple<std::mem_fun_t<void,Farm>,Farm *>,std:: 120,165 0 100.00
std::invoke<std::mem_fun_t<void,Farm>,Farm *> 120,165 0 100.00
std::_Invoker_functor::_Call<std::mem_fun_t<void,Farm>,Farm *> 120,165 0 100.00
std::mem_fun_t<void,Farm>::operator() 120,165 0 100.00
Farm::threadWorker 120,165 0 100.00
MessageTask::run 119,765 1,254 99.67
std::abs<double> 68,738 4,214 57.20
std::operator+<double> 24,028 4,674 20.00
std::complex<double>::operator+= 18,548 4,039 15.44
_RTC_CheckStackVars 630 630 0.52
std::_Complex_base<double,_C_double_complex>::_Add<dou 71 71 0.06
_RTC_CheckEsp 67 67 0.06
@ILT+1790 36 36 0.03
@ILT+1480 2 2 0.00
std::operator*<double> 23,852 4,692 19.85
std::complex<double>::operator*= 18,374 3,791 15.29
_RTC_CheckStackVars 602 602 0.50
std::_Complex_base<double,_C_double_complex>::_Mul<dou 97 97 0.08
_RTC_CheckEsp 62 62 0.05
@ILT+140 23 23 0.02
@ILT+1480 2 2 0.00
_RTC_CheckStackVars 411 411 0.34
@ILT+1480 285 285 0.24
@ILT+2840 239 239 0.20
RTC_CheckEsp 205 205 0.17
Solution Explorer
Search Solution Explorer (Ctr
Solution 'mandelbrot' (1 proje
mandelbrot
References
External Dependencies
Header Files
Resource Files
Source Files
main.cpp
Error List
Output
Find Symbol Results
Solution Explorer
Class View
Ready
Publish

PROFILER

As we can see from the previous slide all the threads from message task are being run parallel to the worker thread. This has helped increase the speed of the data being processed.

However and it is running the same number of threads in both the add task and worker functions it is not worker as many workers as there could be.

HOW CAN WE IMPROVE THIS

We could potentially increase the speed by increasing the number if worker threads and reducing the number of add task threads. This would help as it is much faster to add the tasks to the queue than it is to compute the slice of mandlebrot

QUESTIONS?