Concurrency and Parallel Programming Assignment 2

David van Erkelens and Jelte Fennema
Department of Computer Science
University of Amsterdam

November 19, 2012

1 Wave

1.1 Comparison with self defined threads

Both graphs show that for some unknown reason the largest problem size performs worse than a problem size smaller. The problemsize 10^6 performs best in both cases and the speedup is about the same except when using 8 threads.
OpenMP gets a good speedup when using 10^4, this is definitly not the case when using self defined threads.
When looking at all problemsizes, OpenMP gets the best speedup for small problemsizes and self defined threads get the best speedup out of the larger problemsizes.

1.2 Chunksize and scheduling technique

This test was done with an i_max of 10^6, a t_max of 20000 and using eight threads.

1.2.1 Static

When using a small chunk sizes, the static scheduler spends quite a bit of its time deviding the chunks over the threads. When using chunck sizes betweent

2500 to 250000 it performs the best. After the 250000 the performance goes down again since the load can't be balanced as well anymore.

1.2.2 Dynamic

When using really tiny chunk sizes the performance of the dynamic scheduler is utterly terrible. With a chunk size of one this scheduler timed out, which means it went over the 900 seconds and with a chunk size of 5 it still spends most of its time deviding the chunks. After that it stagnates a bit, but is still slower than the static scheduler with most of the chunk sizes. After the 250000 the performance of this scheduler goes down as well.

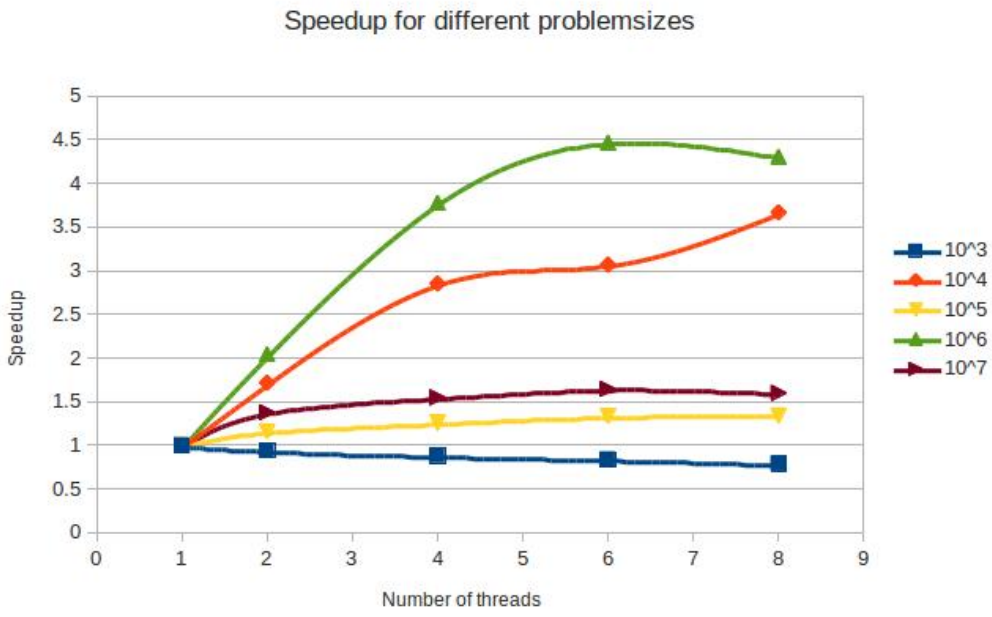
1.2.3 Guided

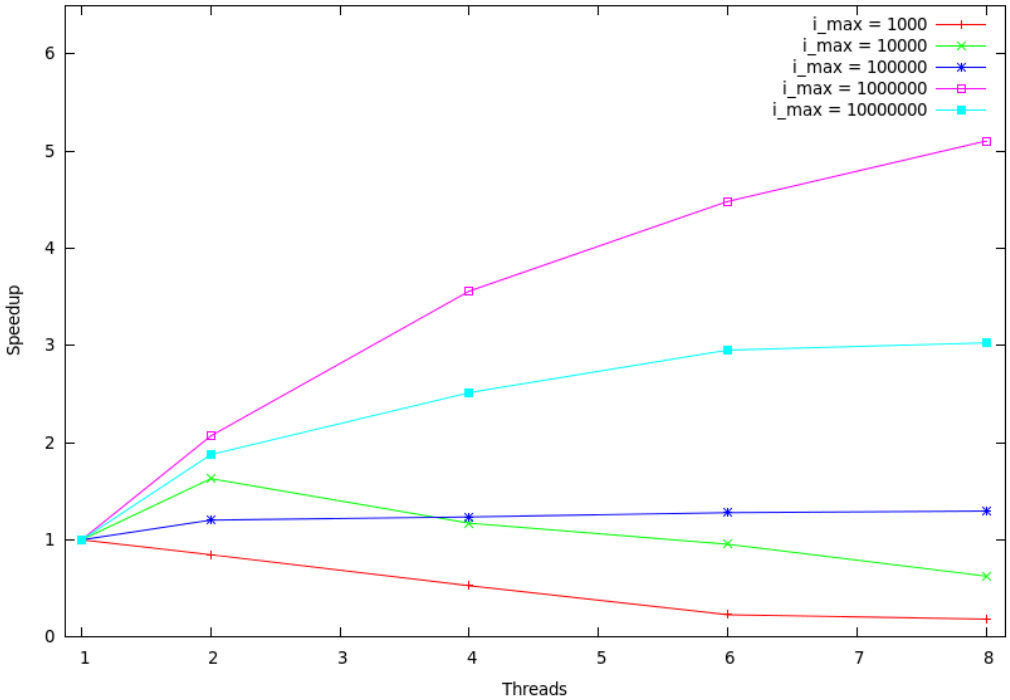
The performance of this scheduler doesn't really get influenced by the chunk size. Only the logical slowdown when the chunk size gets really large can be noticed.

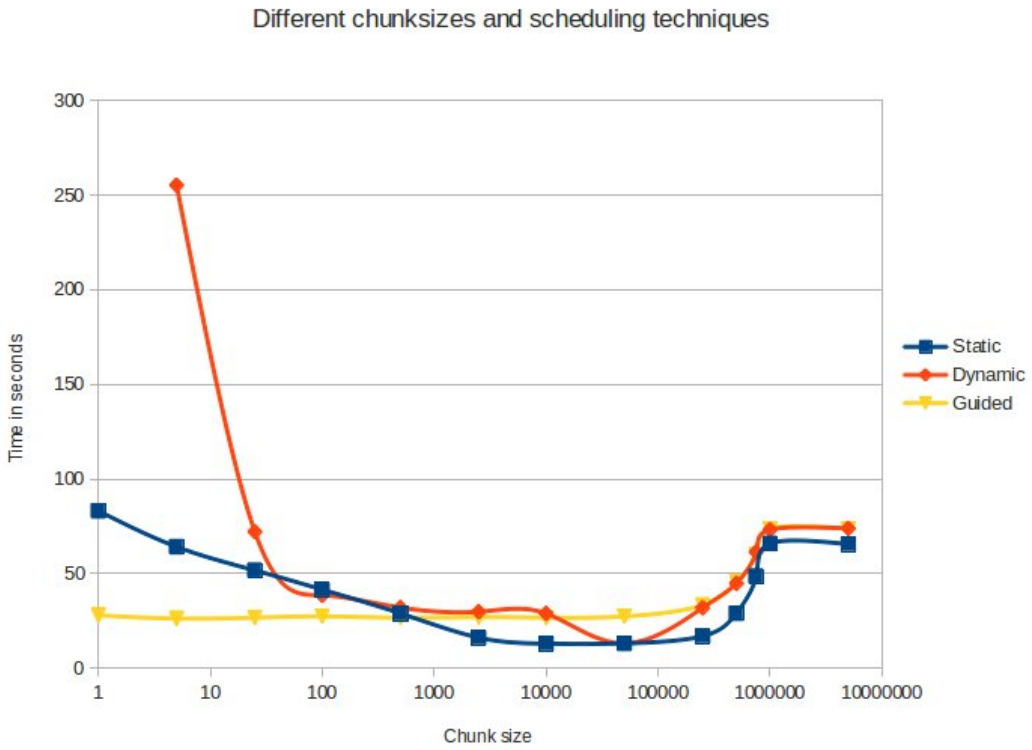
1.2.4 Conclusion

The static scheduler has the best performance when the chunk size is chosen somewhere between about a 1/400 and a 1/4 of the iteration maximum. The dynamic scheduler performce mediocre when comparing it to both other schedulers, although it can perform better than the guided scheduler when the chunk size is chosen just right. The guided scheduler performs better than the static with small chunk sizes, so when you don't care about specifying a good chunk size it is a good choice.

1.3 Graphs


Speedup for different problemsizes
Speedup
Number of threads
10^3
10^4
10^5
10^6
10^7


i_max = 1000
i_max = 10000
i_max = 100000
i_max = 1000000
i_max = 10000000
Speedup
Threads



2 Mandelbrot

For assignment 3.2, a sequential version of the algorithm to calculate the Mandelbrot set was provided. The assignment was to parallelise this algorithm using OpenMP.
The algorithm contains serveral loops. We choose to only parallelise the first for-loop. Since the last for-loops only execute a simple increment, the overhead of paralellisation isn't worth the speedup. Furthermore, we had to take the private variables into account: most of the variables cannot be accessed by multiple threads at once, so they have to be private so every thread can calculate with a variable of its own.
The last problem we encountered, was the fact that the for-loop to be paralellised was a for-loop iterating with doubles. OpenMP doesn't allow this. So we edited the for-loop to iterate with integers, and we transformed the iteration value into a first private value. This way, the algorithm is parillised.