Team 16 - Max Falk Nilsson, Per Albin Mattsson, Jakob Sennerby

How to run the program

To chose between sequential, pthread and OpenMP mode, use the “-mode” flag followed by a string for the three different alternatives:
SEQ: Sequential mode
PTHREAD: Pthread mode
OMP: OpenMP mode

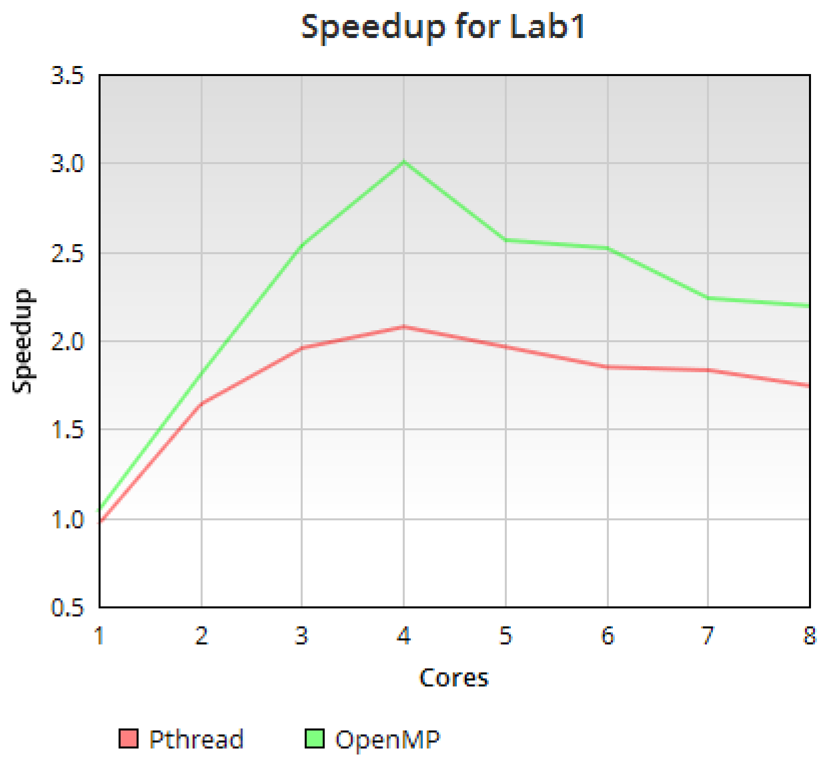
To chose the number of threads use the “-threads” flag followed by the number of threads.

Example:
“./demo -mode OMP -threads 4”

The example above runs OpenMP with 4 threads.

Experiments and plot

The experiments was conducted on a Macbook Pro mid 2012. Intel Core i7 (I7-3615QM), 2.3 GHz 6 MB L3 Cache.



Questions

A. What kind of parallelism is exposed in the identified method?

Since the same work is done by the different threads but on different data this is data parallelism.

B. How is the workload distributed across the threads?

Since we choose how to distribute the threads when using Pthreads the workload is divided as evenly as possible. When there is an odd number of threads one thread will get more work than the others (if any).
When running the program with OpenMP we use the default scheduling method which is a static scheduling so it will split up the work between threads as evenly as possible much like we do with pthreads.

C. Which number of thread gives you the best results? Why?

Since the computer we conducted the experiments on has 4 cores the optimal number of threads is also 4. Even though the computer has hyperthreading which gives 8 logical cores, 4 still gave the best result. This is because less threads would not make use of all the cores and more threads creates too much overhead in this case.

D. Which version (OpenMP, Pthreads) gives you better results? Why?

As you can see on the graph OpenMP gave us the best results. We are using the same type of scheduling in OpenMP as with Pthreads so this should not make any difference. So the compiler is probably optimizing the code for OpenMP more and making it more efficient than the Pthread code, there could also be differences in the overhead for the libraries.
When executing the Pthreads version the cpu usage is much lower than when executing the OpenMP version. The cpu usage when executing the OpenMP version is nearly 100% on all cores but with Pthreads it is about 75% on each core. Which would explain why OpenMP is faster.