Task 04

1 Problem 1.1 [2 marks]

Implement a three stage pipeline, where each pipeline stage is executed by an independent thread. Use the lock-free producer-consumer implementation introduced in the lecture for all buffers needed. Each of the three pipeline stages should perform some dummy workload. The amount of work done should be determined by means of three different compile-time constants, `WORKLOAD1`, `WORKLOAD2`, and `WORKLOAD3`, respectively. The program should create the buffers and the three threads, then it should provide input data to the initial buffer and consume the overall results from the final buffer.

You may assume that the maximum data you will ever provide as input is a compile time constant named `N_DATA`. The initial and the final buffer may be large enough to hold all data. All intermediate buffers should be significantly smaller.

As a starting point for your implementation you may wish to use the pipeline template provided on vision. Notice that the provided C code already contains dummy workload execution through a function named `process`. That function assumes that the processed data is never negative!

2 Problem 1.2 [3 marks]

Modify your solution to 1.1 so that your program measures the average latency, the minimum latency, the maximum latency as well as the throughput achieved when processing `N_DATA` many data items.

Use the system function `clock_gettime` for measuring execution times.

Devise your experiments carefully in order to obtain as accurate as possible runtime figures.

Analyse the impact that the following variations have on the observed values:

- increase the overall number of data item processed
- decrease the workload of all pipeline stages
- keep `WORKLOAD1`, decrease the other two

Run systematic experiments and produce graphs of your results. Try to explain your findings. A short text should suffice.

3 Problem 1.3 [2 marks]

Modify your solution to 1.2 by changing your `push` and `pop` methods so that they use mutexes for implementing mutual exclusion.

Repeat your experiments and contrast them to your findings from 1.2.

Hand-out: 24/02/2016 Hand-in: 9/03/2016

4 Problem 1.4 [3 marks]

Modify your solution into a data-parallel solution in the way described in the lecture on pipelining. Each of your three worker threads should now perform all three pipeline stages on each data item they process.

Try to keep the experimental setup close to the setup used earlier. In particular, do keep your input buffer and your output buffer. Rerun your experiments with this solution and interpret the results. How can you optimise the overhead that stems from the use of the input and output buffer? Measure and document the effect of any optimisation you try.