AARHUS UNIVERSITET
INGENIØRHØJSKOLEN

Exercise: Concurrency Patterns – The Pipeline Pattern

In this exercise you will become acquainted with the pattern. You will use this pattern to speed up an initially sequential operation on string compression and manipulation.

A new string compression algorithm has been devised. It is so groundbreaking that it is not really trusted – laboratory tests must be conducted to validate the results of the algorithm.

The algorithm for compression of a string is as follows:

- Given a string consisting of a sequence of alphanumeric characters, any subsequence of identical characters of length > 1 shall be replaced with one occurrence of the character followed by the number of occurrences of that character.

Example: **"ABBBCABCCCAACCCAAABBBCABBAAAA" →**
"AB3CABC3A2C3A3B3CAB2A4"

Obviously, this is a lossless and extremely effective compression algorithm bound to revolutionize everything overnight and rule the world! But first, we must implement it and prove it right!

Exercise 1:
Implement a regular, sequential program which does the following:

- Generates 100.000 strings each consisting of a random sequence of the three first letters of the alphabet (that's A, B and C), 25.000 characters long.
- Compresses each string as specified above
- Calculates the compression ratio of each string and finally returns the average compression ratio.

Exercise 2:
Implement the same program as in exercise 1, this time using the Pipeline pattern. Compare the execution time of the two implementations – which is faster?

Exercise 3:
Analyse the execution time of each of the steps in the pipeline. Identify the bottleneck and create one or more parallel step(s) to reduce the bottleneck. Again, compare the execution time of the two pipeline implementations – did you get a performance gain?