Laborator 3 - Tanislav Cosmin-Gabriel, Grup 237

**Implementation details**

The program receives as command line arguments the mode to run in, with 0 meaning iterative, 1 meaning scatter-gather, and 2 meaning manual asynchronous transfers.

The iterative implementation reads the digits of the 2 numbers (from least significant to most significant), padding them both with zeros until their lengths are equal to the maximum of their individual lengths + 1 (to leave space in case the last addition produces a carry. Then, it iterates over all the digits of the 2 numbers and adds them, taking into account the carry of each addition past addition and passing the new carry forward. After doing all the additions are done, the result is written into the output file.

The scatter-gather implementation reads the digits of the 2 numbers in the same fashion as the iterative implementation, but the numbers are padded with zeros until their individual length + 1 is divisible by the number of processes. The digits are scattered to each process (including the master one). Each process adds the digits of the received slice of numbers. The carry is then received and if it is not zero, then a part of the calculation is re-done to add the new carry. The produced carry is then sent to the next process. After each process finished its calculations, all the result slices are gathered and written into the output file.

The manual implementation reads the digits of the 2 numbers in slices (similar to how scatter-gather would slice the digits when sending them to the individual processes) and then immediately sent to each process (except the master one).

The slave processes starts receiving the 2 numbers and the carry. They wait for the 2 numbers, and check if the carry has been received. If the carry has been received, the calculation is done one single time. If not, the calculation is done without a carry, and then the carry is waited for. If the received carry is non-zero, part of the calculation is re-done to add the new carry.

**Results (average over 100 runs)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Case** | **Time in ms** | | |
| **Iterative** | **SG** | **Manual** |
| Numar 1 = “123456789123456789” = Numar2 | 0 | 0.5 | 0.4 |
| N\_1=1000 si N2=1000 | 0 | 0.7 | 0.7 |
| N\_1=100 si N\_2=100000 | 5 | 6 | 7.5 |
| N\_1=100 si N\_2=1000000 | 60 | 67 | 60 |

**Conclusions**

It doesn’t make much sense to split the addition operation across multiple processes, because each process needs the carry of the past one to finish its calculation. Iterative addition takes generally less time, and it doesn’t have the overhead of data transmission (which is a big issue if the processes are ran on different machines, across a network).