Parallel and Distributed Algorithms (PRL) – 2021/2022
Odd-Even Merge Sort

Pavel Yadlouski (xyadlo00)

1 Communication protocol & network

In this project size of the problem (amount of number to sort) is hard-coded to 8. From this perspective, processor network would look in the following way:

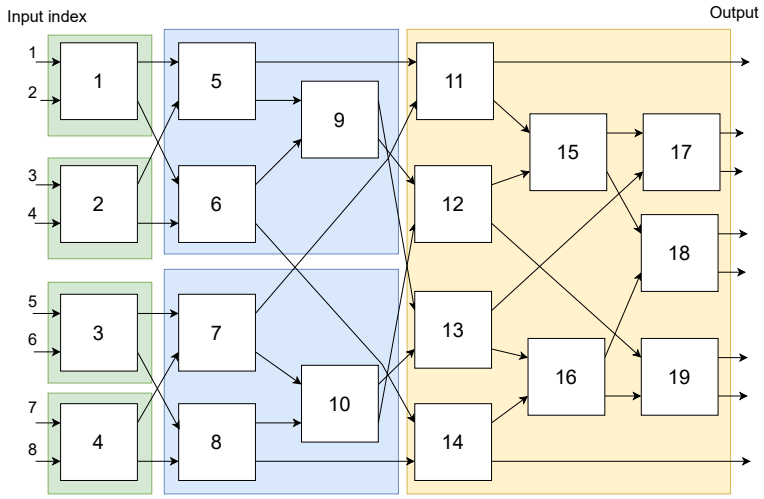

Figure 1: Processor network

One processor is represented as a square with a its rank inside. As you can see, number of processors required for sorting the sequence of 8 numbers without any improvements is 19 + 1 control processor. From the Figure 1 we can see how each processor communicate with each other processors. Form top-level point of view, this network contains 3 building block:

1. Blocks for sorting and merging sequence 1 x 1 (green boxes)
2. Blocks for sorting and merging sequence 2 x 2 (blue boxes)
3. Blocks for sorting and merging sequence 4 x 4 (yellow boxes)

On initial step processor 0 (control processor) would send corresponding numbers to processes 1 - 4 as shown on the Figure 1. After that each processor sends sorted sequence to corresponding processors on the next level and etc. From code perspective, each block has following tags on the input "ports":

Each processor is waiting with block function `MPI_Recv(...)` for message from non-blocking `MPI_Isend(...)` function.

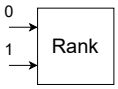

Figure 2: Processor input tags

2 Analyses

Following steps are considered while analysing this algorithm without any improvements for sorting sequence of length $n = 2^m$ where m is some power.

1. The first phase of comparison requires 2^{m-1} of Computation Element CE blocks (minimal building block of type 1 x 1)
2. The second phase requires 2^{m-2} blocks of type 2 x 2 (each block is of 3 CE blocks)
3. The third phase requires 2^{m-3} blocks of type 4 x 4 (each block is of 9 CE blocks)
4. And so on

Time taken by this algorithm is $t(n) = O(m^2) = O(log^2(n))$

Total cost of the algorithm is $c(n) = O(n * log^2(n))$

3 Summary

Odd-Even Merge Sort algorithm has total cost as $c(n) = O(n*log^2(n))$. This cost is not optimal, but still can be used because it is easy enough to implement. Moreover on some hardware architectures (e.g GPU architecture) this algorithm is useful even with not optimal cost.

Possible improvements

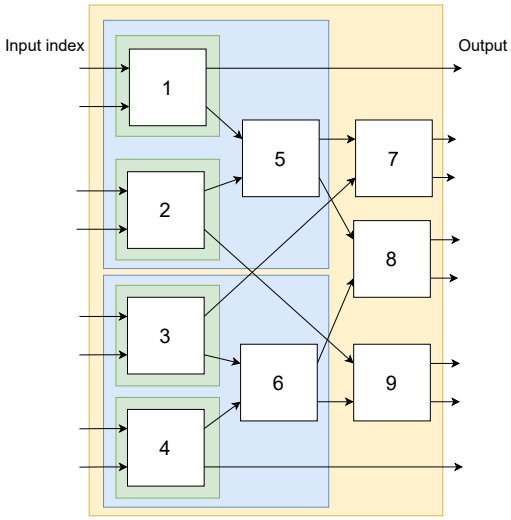

Figure 3: Theoretical concept of processor network

One of the problems of naive network architecture in Figure 1 is that after computation, processor is doing nothing useful. On of the theoretical improvements for overcoming this problem can be "merging" of the block of different type to cascade in the way Figure 3.

Adding all connections between processors would make the figure very confusing, so not adding these connections. But in general, those connections would include self loops based on used number of block types. In this case number or processors required for running the algorithm is given by number k of CE in the biggest block type (in this case, biggest block type is 4 x 4) and one control processor, so $p(n) = k + 1$.

This concept is just theoretical and requires in-dept analyses, but I don't have time for such research.