**Parallel programming: Assignment 5**

Our solution to the problem given can be divided into two steps. The first one being dividing labor between processes, and the second one being communicating the results that each process had when they are done processing their assigned data. We will be explaining each step in a detailed manner in the next paragraphs.

1. **Division of labor**

One of the requirements for this assignment is that each process has a balanced amount of work to perform. This means that each process should have the exact same amount of information to process. To accomplish this, we need to set a start and end to each process that satisfies this requirement.

The problem presents us with two nested for loops that iterate through the array in a O(n2) time complexity. But to be more specific, let’s call the variable used to iterate on the outer for loop ‘I’. The nature of the iteration is that for each i, the inner loop will iterate i+1 times. We think that this is useful information for us to be able to find start and end points for each process.

To start, we must find the total number of iterations that are needed to solve the problem and divide that by the number of processes. This will give us the number of iterations that each process will perform. The way we do that is by adding i+1 to a total\_iter variable as we iterate through the array once.

Once we have the number of iterations that each process needs to perform, we use 3 helper functions that allow us to determine the start and end for each process. Basically, these functions will give us a start ‘i’ value and a start ‘j’ value for the loops. Once we have that set, we can let the processes run and perform the operations needed only on their part of the entire process. At the end of this step, each process will have produced a result array (that was initialized with all 0 values) with their results. This array will be used in the communication step to be able to get to the final solution.

1. **Communication between processes**

For the communication we decided to take the same approach we did in project 2. We will use the process ID to match each process to another and combine their results array. Then they will proceed to communicate to other processes until they all have the information from every process. The communication will look like this assuming we have 8 processes:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Matches: (0, 7); (1, 6); (2, 5); (3, 4)

This will make each pair have the info from the other, so that they now can share it with other pairs in a more effective way. The next step will look like this:

|  |  |  |  |
| --- | --- | --- | --- |
| Chunk 1 | | | |
| 0 | 1 | 2 | 3 |
| Matches (0, 3); (1, 2) | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Chunk 2 | | | |
| 4 | 5 | 6 | 7 |
| Matches (4, 7); (5, 6) | | | |

|  |  |
| --- | --- |
| Chunk 3 | |
| 4 | 5 |
| Matches: (4, 5) | |

|  |  |
| --- | --- |
| Chunk 4 | |
| 6 | 7 |
| Matches: (6, 7) | |

|  |  |
| --- | --- |
| Chunk 1 | |
| 0 | 1 |
| Matches: (0, 1) | |

|  |  |
| --- | --- |
| Chunk 2 | |
| 2 | 3 |
| Matches: (2, 3) | |

BY doing this, every process would have the data from the rest in just Log2P steps with P being the number of processes. We’ll take process 0 to demonstrate this communication:

1st Communication: 0 -> data from (0, 7)

2nd Communication: 0 -> data from (0, 7, 3, 4) (because 3 already holds information from 4)

3rd Communication: 0 -> data from (0, 1, 2, 3, 4, 5, 6, 7)

At every communication step, the process will need to extract the data from the result array received and add every value to the result array that they currently have. They will need to do that before they send the information to their new process match.

With this, at the end of this communication, every process will hold the final global result.