Module 2.1 Exercises

Problem 1: Identify data and task parallelism aspects of finding the area under the curve of a polynomial on a range of $[x_1, x_2]$. Include a justification to explain why a part or the entire problem is data or task parallel. For each aspect of parallelism include a discussion of why it is not the other type of parallelism too.

Solution: This has both data and task parallelism.

Data Parallelism:

> (1) There are significant data parallelism aspects of the problem because each processing element can be assigned a sub-space to integrate within the given range $[x_1, x_2]$. It is important for the students to break the problem down into the smaller parts and explain why they did that. Also, it is important for them to explain why it is not task parallelism. It is not task parallelism because the same task is being performed on the data, each processing element is computing the area for a small sub-set of the range.
>
> (2) There is data parallelism in computing local sums if a multi-level communication approach is used (see below). Again it is important that students explain why this is data parallelism and why it is not task parallelism.

Task Parallelism: Once all of the processing elements compute the area of their section these areas must be summed into a single value. Naively, this could be done by having all processing elements send their values to a single processing element who would then compute the sum. A better way is to use a multi-level communication structure where local sums are computed by receiving processing elements who ultimately get down to a single value at the end of the communication. This is task parallelism because each processing element is computing a local area, then local sums (only if multi-level communication is used), and then one processing element computes the final sum providing the area under the curve.

Problem 2: Recall that Gaussian elimination can be used to solve a set of linear simultaneous equations. A simple implementation is to start at the top row and first column and convert the matrix into the identity matrix and update the extra solution row with each step. The process starts with the leftmost column and works to the right of the matrix. Once the left part is the

identity matrix the extra solution row will contain the values for each of the *n* variables in the set that solve the problem. Assume that there are 5 linear equations with 5 unknowns yielding a 5x6 matrix (the 6th column (rightmost column)) is the solution set column and is not converted into the identity matrix). Devise a schedule of operations and identify if each operation is data or task parallel. Include rational as to why each operation is data or task parallel.

Solution: Multiple solutions for this based on how the schedule and operations are performed. Key points are below.

Task Parallelism:

- Finding the value to make the current row's leftmost non-0 element a 1.
- Finding the multiples for the current row to use with the other rows to zero-out that column in the other rows.

Data Parallelism:

- Once the values to zero out the other rows are known those computations can be performed in parallel.

Problem 3: Recall that Gaussian elimination can be used to solve a set of linear simultaneous equations. One problem with this approach is that as the size of the set of equations grows the inaccuracies of floating point mathematical operations make the final solution less accurate. There are several ways to address or minimize this. For this problem assume that you have a routine, MINIMIZE_ERROR(), that can minimize error of the matrix but that it needs to work on the entire matrix. There is a routine GAUSSIAN_STEP() that performs one step of Gaussian elimination. Also, assume that you need 20 steps of Gaussian elimination to find the solution and that the maximum allowable error will occur if MINIMIZE_ERROR() is not called at least once every 5 Gaussian elimination steps. Devise a schedule of operations and identify if each operation is data or task parallel. Include rational as to why each operation is data or task parallel.