Exam 2: Applying Foster's Methodology

Author: Gabriel Hofer

Course: CSC-410 Parallel Programming

Instructor: Dr. Karlsson

Due: November 23, 2020

Computer Science and Engineering
South Dakota School of Mines and Technology

Buidling the Histogram

Partitioning

We will partition the array into $n/chunksize$ subsections where n is the size of the array of floats and the parameter $chunksize$ is the number of floats per subsection of the original array. If n is not evenly divisible by $chunksize$, the remaining subsection will have fewer elements.

Each subsection of the array is assigned a task. And, each task corresponds to a unique bin in the histogram. Task i is assigned to bin j such that i=j (i.e. tasks and bins have the same rank/identifier). See figure 1.

Before processing the data, each task has a local variable, cnt, which is initialized to zero. cnt will store the number of floating-point numbers in the bin that is associated with its task.

However, the floating-point numbers in a subsection don't necessarily 'belong' in the bin associated with their task number. We need communication in order to 'move' the floating-point numbers to the right tasks/bins.

Communication

Each task will iterate through its own subsection. For any floating-point number f in the subsection, the task calculates which bin it belongs to. The correct bin is calculated by dividing the floating-point number by the bin_width using integer division: $target\_bin = f/bin\_width$. After traversing through the whole sub-array, the task then sends a message to all bins. The messages contain the amount that each non-local task should update their local count ('cnt') variable. See figure 2.

Agglomeration

We could have created a task/process for each floating-point number in the array. This is actually possible if the bin_width is set to 1. However, it's preferable to choose a larger value for bin_width because it will reduce the number of messages sent between processes. Notice that there will be $O(chunksize^2)$ messages sent between tasks for updating each tasks cnt. Also, the point of the histogram is to combine data points to show groups of similar data. A histogram with so many bins is less likely to be useful.

Mapping

Since the number of tasks and the number of processors are the same, we simply assign task t to a process p such that t=p. This mapping was illustrated in Figure 1.

Combining Data in the Root Process

The root task will construct a plot from the cnt of each task. For this to happen, the data needs to be retrieved and sent to the root. Once, each task has iterated through its own array, they will ping the root message, indicating that the task has processed all of its floating point numbers and that all messages to other processors have been received. Once the root process has been pinged this message from all of the processors/tasks, then the root will send back a pong to all of the tasks. The purpose of this pong is to tell every task to send their 'cnt' variable to the root task. The reason we need to do a ping-pong before each task simply sends its cnt

variable to the root is that we need to make sure that every message has been completed (i.e. synchronized) before we send the cnt to the root.