Parallel program design — simple example cont.

Example 4

Suppose we have an array with large quantities of floating point data stored in it. In order to have a good feeling of the distribution of the data, we can find the histogram of the data. To find the histogram of a set of data, we can simply divide the range of data into equal sized subintervals, or bins, determine the number of values in each bin, and plot a bar graphs showing the sizes of the bins.

Parallel program design — simple example cont.

Example 4 cont.

As a very small example, suppose our data are

$$A = [1.3, 2.9, 0.4, 0.3, 1.3, 4.4, 1.7, 0.4, 3.2, 0.3,$$
$$4.9, 2.4, 3.1, 4.4, 3.9, 0.4, 4.2, 4.5, 4.9, 0.9]$$

- For A, $A_{min} = 0.3$, $A_{max} = 4.9$, $A_{count} = 20$.
- Let's set the number of bins to be 5, and the bins are the $[0, 1.0), [1.0, 2.0), [2.0, 3.0), [3.0, 4.0), [4.0, 5.0)$, $bin_{width} = (5.0 - 0)/5 = 1.0$.
- Then $bin_{count} = [6, 3, 2, 3, 6]$ is the histogram — the output is an array the number of elements of data that lie in each bin.

Parallel program design — simple example cont.

Example 4 cont.

```
for (i = 0; i < data_count; i++){
    bin = Find_bin(data[i], ...);
    bin_count[bin]++;
}
```

The `Find_bin` function returns the bin that `data[i]` belongs to.

Parallel program design — simple example cont.

Example 4 cont.

- Now if we want to parallelize this problem, first we can decompose the dataset into subsets. Given the small dataset, we divide it into 4 subsets, so that each subset has 5 elements.
  - Identify tasks (decompose): i) finding the bin to which an element of data belongs; ii) increment the corresponding entry in `bin_count`.
  - The second task can only be done once the first task has been completed.
  - If two processes or threads are assigned elements from the same bin, then both of them will try to update the count of the same bin. Assuming the `bin_count` is shared, this will cause **race condition**.

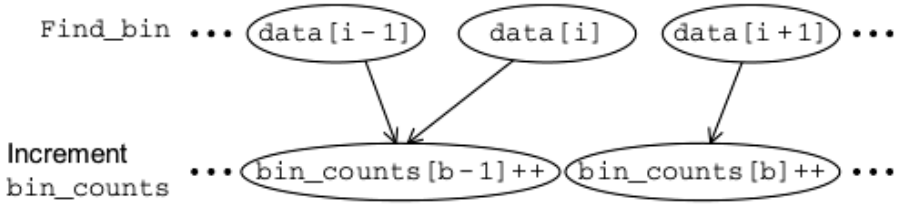

Figure: Tasks and their communications.

WITS UNIVERSITY

Parallel program design — simple example cont.

Example 4 cont.

- A solution to race condition in this example is to create local copies of `bin-count`, each process updates their local copies, and at the end add these local copies into the global `bin_count`.

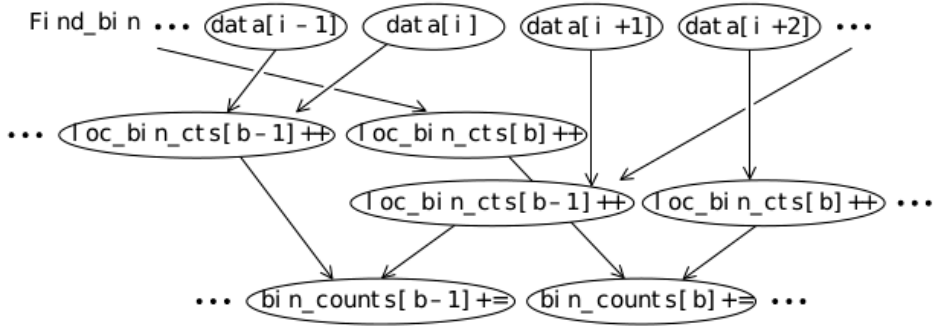

Figure: Tasks and communications.

Parallel program design — simple example cont.

Example 4 cont.

- In summary, the parallelization approach is to
  - Elements of data are assigned to processes/threads so that each process/thread gets roughly the same number of elements;
  - Each process/thread is responsible for updating its `local_bin_counts` array based on the assigned data elements.
  - The local `local_bin_counts` are needed to be aggregated into the global `bin_counts`.
    - If the number of bins is small, the final aggregation can be done by a single process/thread.
    - If the number of bins is large, we can apply parallel addition in this step too.