Programming Assignment 2 Report

Ziyi (Francis) Yin and Rafael Orozco

March 2021

1 Algorithm Descriptions

1.1 Broadcast

For broadcast, we implemented the naive algorithm. In which the processor with rank-0 sends the double to each processor in sequence. This runs in O(n) time. The rank-0 processor needs O(n) memory but then sends the proper O(n/p) array to each processor.

1.2 Scatter

For our implementation of scatter was based on the parallel algorithm shown in class. This algorithm uses hypercubic permutations to organize communication. One problem that we encountered was the case when p was not a power of 2. For this we took the advice shown in class and just implemented the algorithm with $p < p' = 2^d$ and then ignore all communications that are attempted to processors that dont exist.

1.3 ParallelPrefix

For our implementation of Parallel prefix, we made sure to make proper use of the OP constant so that we could toggle on SUM and PRODUCT operator. Although in the end, the polynomial evaluator only made use of the product operator.

1.4 Polynomial Evaluator

1.4.1 Cereal Polynomial Evaluator

For our cereal polynomial evaluator, we made sure to use high quality horner's milk and then added the cereal.

1.4.2 MPI Polynomial Evaluator

For the parallel polynomial evaluator, we followed the instructions in the slides which can be summarised as such. First set the entries in each processor to x, which was previously broadcast-ed to all processors. With the exception of the processor rank-0 which will set the first element in its array to 1. Then we will run parallel prefix with PRODUCT op in order to get the powers of x. Then we will multiply the constants to each entry and then finally sum over all entries using the parallel sum algorithm that we implemented in PA1.

2 Plots

In the following plots we will show run-time results of a constant problem with increasing number of processors. If we fix with $N = 1000000$ then we have our broadcast time increasing in Fig 1. If the next plot Fig **??**, we see the run-time of parallelprefix. The n we chose were 1,2,4,8,16,32 .

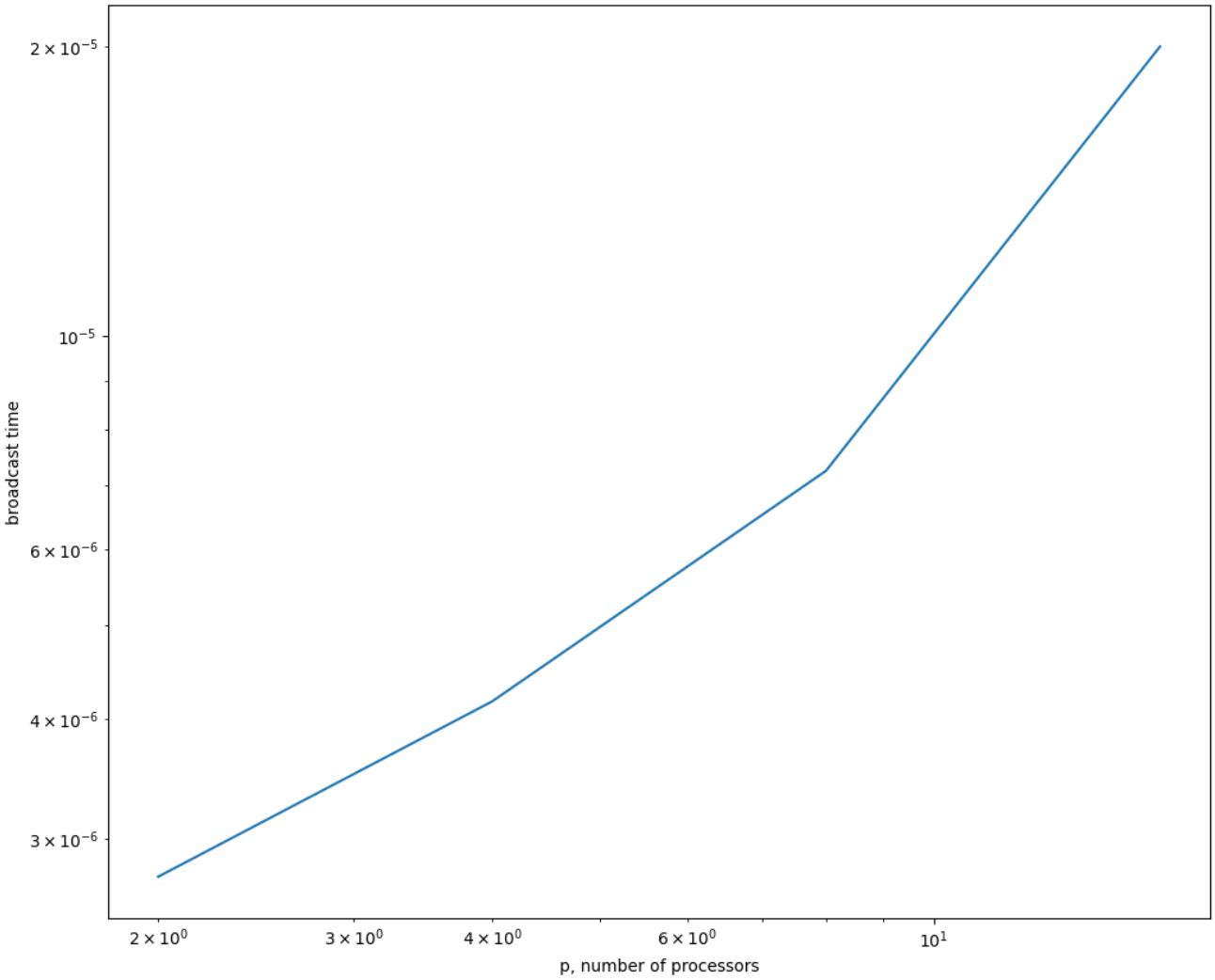

Figure 1: Broadcast time for all p, with fixed $n = 10^6$

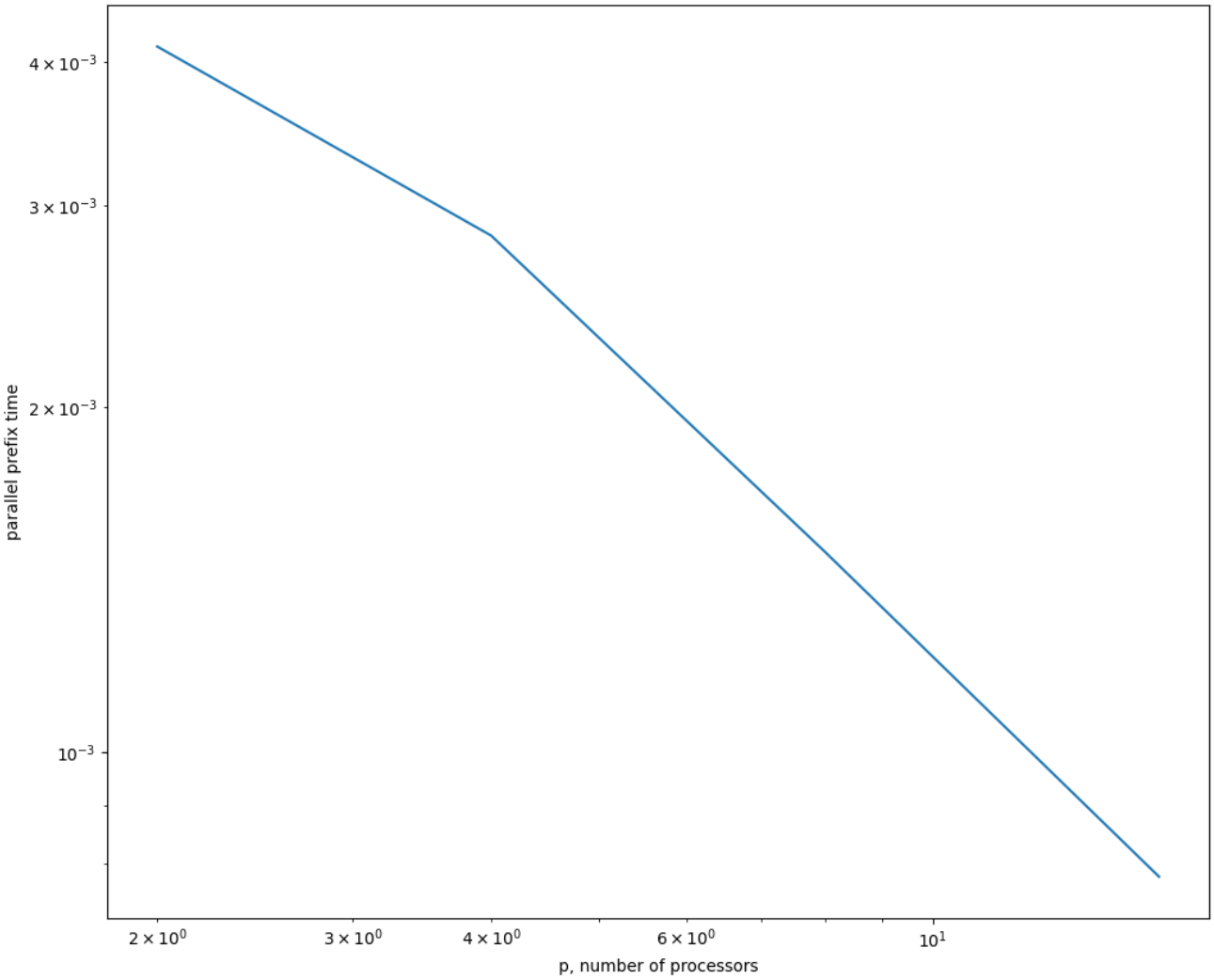

Figure 2: Parallel Prefix time for all p, with fixed $n = 10^6$

3 Observations

The first observation that we want to discuss is the presence of NAN values in the results. Because of the large numbers that get evaluated this is expected and normal, but we can still make meaningful conclusions on run time of the runs.

From our plots we can notice that as expected the run time was dominated by the computations in mpi poly evaluator.

Notice that these are loglog plots. Since these plots are straight lines, with slope -1 and 1, we can conclude that

$$T \propto \frac{N}{p}$$

which means that run time is proportional to number of elements, and inverse proportional to number of processors.