Jaron Smith, Noah Krill

Dr. Xiao

Intro to Parallel Processing

15 November 2020

Fantasy Football Term Project

Data Summarization and Algorithm

For this project, we decided to analyze the average amount of fantasy points given up by NFL defenses to Quarterbacks in the 2020-2021 season. We drew our data from Lineups.com, pulling the average FPPG (Fantasy points per game) given up by each team according to the Yahoo.com scoring standard. Our program is designed to use this data and find the average amount of points given up to quarterbacks, implementing the MPI API to parallelize the program.

Implementation

In implementing this program, we stored the values of points given up in an array to be processed. The number of threads to be used in this program is determined at compile time, and MPI_Scatter is used to split the given values across these threads, allowing for the average to be calculated in parallel. After this, the average of the values given to each thread is calculated using the MPI_Average function. This function receives the array of values of each thread and the number of values (stored as the integer local_size), adding together the values then dividing the sum by local_size. This value is stored in the variable subavgs. After calculating this average for each of the threads, MPI_Gather is used to retrieve the subavgs values from each of the threads, combining them into an array named subavg. Finally, the average of the averages from each of the threads is calculated using the MPI_Average() function, giving the final result.

Results

Performance	
Num Threads	Time Taken
1	0.000051
2	0.000108
4	0.000145
8	0.000231

Speedup	
Num Threads	Percent
1	1
2	0.4722222222
4	0.3517241379
8	0.2207792208

Efficiency	
Num Threads	Percent
1	1
2	0.2361111111
4	0.08793103448
8	0.0275974026

As seen in the results above, the program is not scalable when parallelized. Although a critical area was used and the data split and analyzed over multiple threads, the program did not see any increase in the categories of speed up or efficiency. This may be due to a number of factors, but most likely being from the possibility that the data set was too small to see the impacts of parallelism become apparent. The dataset that we drew our statistics from consisted of only 414 values, giving us a relatively small sample size to test our program on.

Our program worked to find the average fantasy points per game given up by defenses to quarterbacks in the NFL, according to the Yahoo.com scoring guidelines. It used C as the primary language, and implemented MPI as a means to parallelize the code. The end result did not see an improvement in performance from parallelizing the code, possibly due to the small sample of data used. Noah Krill put together the program, and Jaron wrote the report/presentation for this project.