MPI_Proj1 Results

Azlan Shah Bin Abdul Jalil

903108282

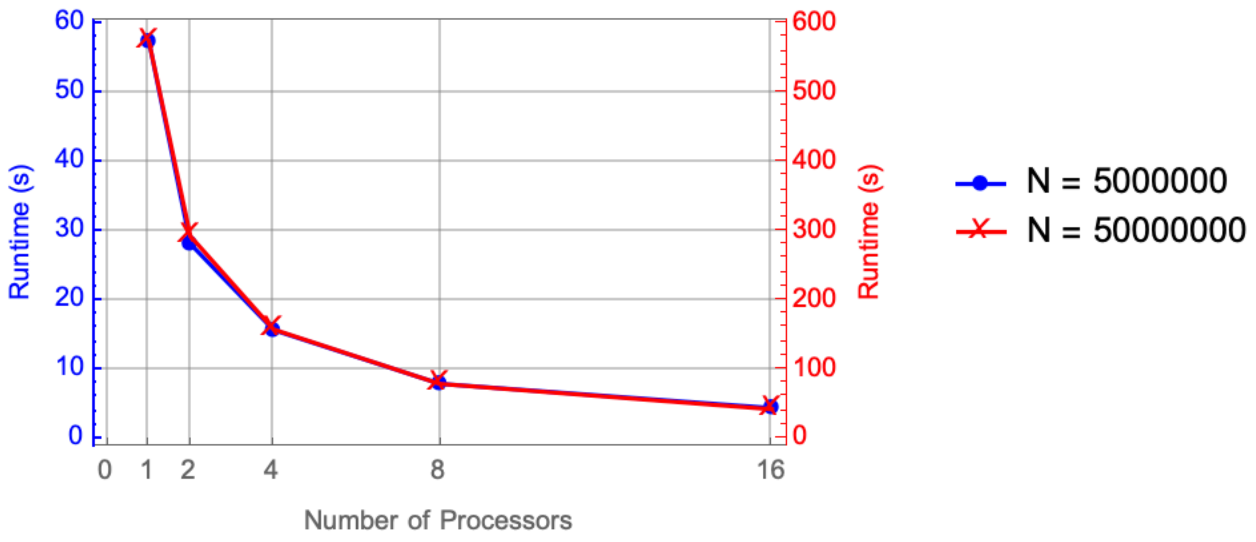

Figure 1: Runtime for variable number of processors

Figure 1 shows two curves of the runtime against number of processors for the dartboard algorithm. The trend is the same whether the algorithm was running with element $N = 5 * 10^6$ or with $N = 5 * 10^7$. Note that both cases were done with the same $R = 100$. As the processors doubled, the speed-up will also be doubled. As an example, for $N = 5 * 10^6$, the runtime was $57.46s$ with one processor while with two processors, the runtime was $28.15s$. The runtime was halved every time we double the number of processors, up to 16 where it only took $4.45s$ for the algorithm to finish.

The trend of the runtime halving every time we double the processor continue even when we increase the number of elements N. It is difficult to determine exactly the runtime formula, but we can say the lower bound of $T(N,P) = \Theta\left(\frac{R*N}{P}\right)$. We know the algorithm will call the dboard() function R times and each time it's called, every processor will process $\frac{N}{P}$ random numbers. This linear relation between the runtime and the number of elements or rounds does get reflected in the final results.

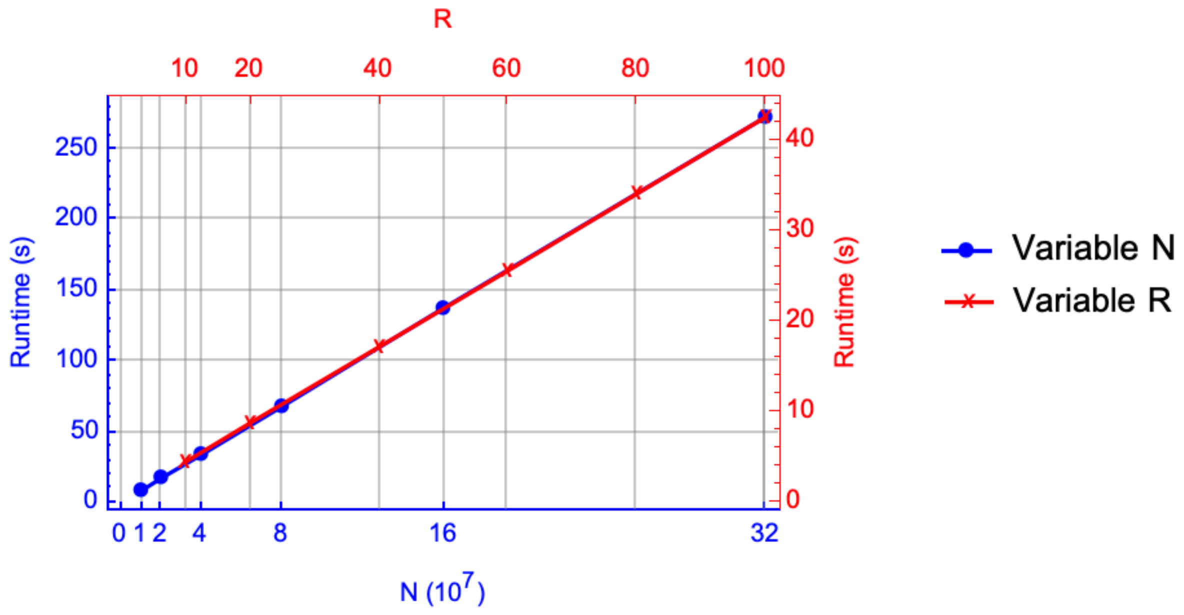

Figure 2: Runtime for variable N and R

As described earlier, the runtime has a linear relation with the number of elements, N and the number of rounds, R. Figure 2 above shows that for a fix P (processor used which was $P = 16$), we can see a linear graph for both cases of variable N (R fixed which was $R = 100$) and of variable R (N fixed which was $N = 5 * 10^7$). The blue portion of the graph is the variable N and we can clearly see that as we double the number of elements, the runtime will also be doubled. For example, when $N = 1 * 10^7$ runtime was $8.53s$, while when $N = 32 * 10^7$ runtime was $272.76s$ which is exactly 32 times longer. The same is true for variable R (the red portion of the graph). When $R = 10$, the runtime was $4.52s$ and when $R = 100$, the runtime was $45.47s$ which is 10 times longer.

From the two figures, we can conclude that the runtime is inversely proportional to the number of processors and linearly proportional to the number of elements and rounds. Thus, our lower bound formula for runtime is correct which is $T(N,P) = \Theta\left(\frac{R*N}{P}\right)$. Our algorithm is simple enough that we can create a serialize version (only using one processor) and running it on P number of processors simultaneously and we will get the same runtime as the parallel version. This is possible since technically each darts thrown are independent and no variables are actually needed to pass between processors (even though we did pass some variables just for exercise purposes). However, we did prove the relationship between runtime and $N, R\ and\ P$ using MPI programming.