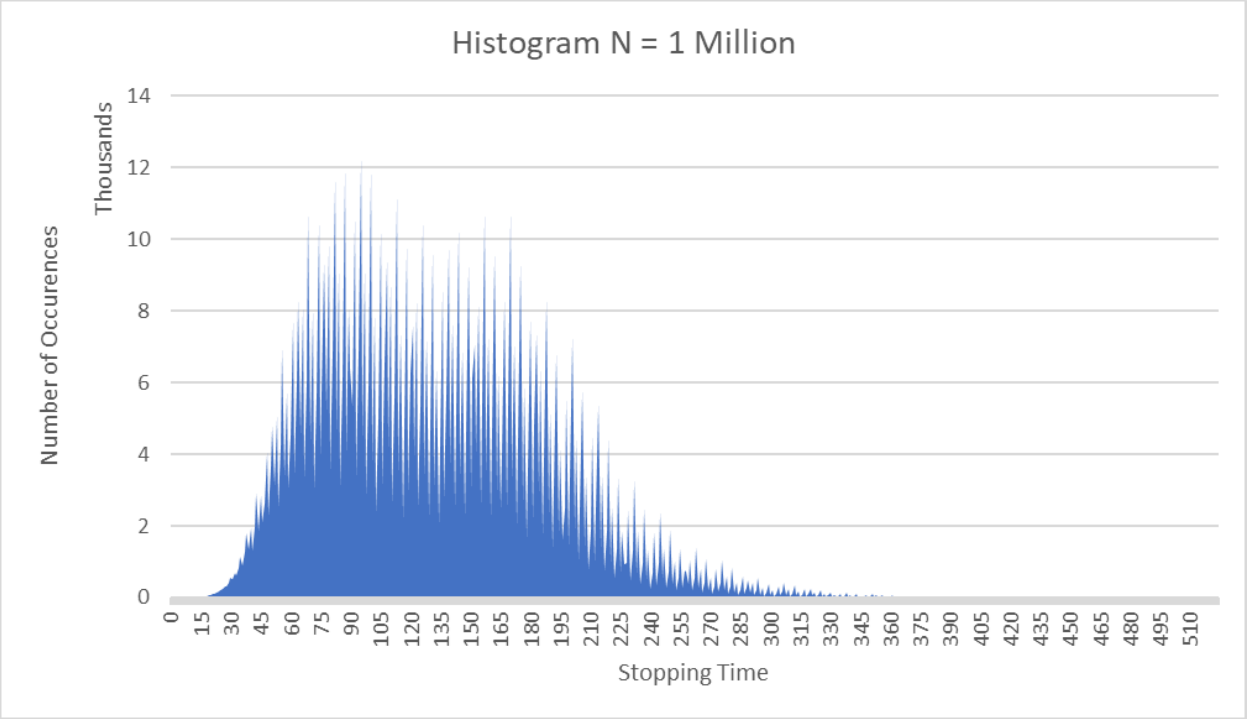
Report For Project 2
Systems and Networks

Andrew Bare, Dante Gordon
10/15/2023

Experiment Description

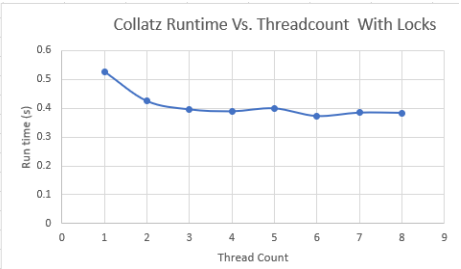
The Collatz conjecture posits that by either dividing an even number by two or multiplying an odd number by three and adding one continuously, eventually every real number will converge at one. Our task was to experiment with this function and collect data relating to how many iterations each series took to converge.The experiment was run on a personal computer with 12 GB of installed RAM and an Intel i5-8400 CPU 2.80GHz with 6 cores. We ran the collatz program in the WSL terminal open in VS code with N = 1 million with T = 1 - 8 for 10 individual trials. We then took the average of the trials for each value of T. We plotted these values in excel and generated a plot of the results, with and without locks.

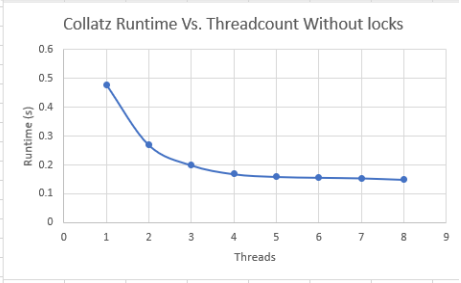
Histogram Of Collatz Values For N =1,000,000



Experimental Data

1												
2	Dante PC											
3	Threads	1	2	3	4	5	6	7	8	9	10	Ti me
4	1	0.5228	0.54228	0.5185	0.536	0.5235	0.5336	0.5223	0.5266	0.5188	0.5301	0.527448
5	2	0.4130	0.4088	0.4182	0.4231	0.4514	0.4183	0.4182	0.4256	0.4363	0.4377	0.4251
6	3	0.3790	0.3844	0.4109	0.425	0.3893	0.4008	0.4008	0.3943	0.3864	0.3865	0.39574
7	4	0.3809	0.3674	0.3775	0.3979	0.3682	0.3768	0.4345	0.3705	0.4274	0.3871	0.3888
8	5	0.3779	0.4205	0.4161	0.415	0.3932	0.3697	0.3789	0.43	0.4045	0.3883	0.39941
9	6	0.3751	0.3816	0.366	0.3701	0.3332	0.3508	0.3705	0.3843	0.4156	0.3802	0.3727
10	7	0.3795	0.35173	0.3834	0.3904	0.3555	0.3811	0.3876	0.3674	0.4514	0.397	0.384503
11	8	0.3770	0.4027	0.3774	0.3702	0.3997	0.3876	0.3956	0.393	0.3731	0.3612	0.3837
12	Dante PC NO LOCKS											
13	Threads	1	2	3	4	5	6	7	8	9	10	Time
14	1	0.4824	0.4668	0.4824	0.4795	0.4809	0.4779	0.4834	0.4621	0.4834	0.48	0.47788
15	2	0.2754	0.2712	0.2613	0.2818	0.2623	0.2657	0.2605	0.2728	0.2692	0.2704	0.26906
16	3	0.1925	0.1997	0.2123	0.1984	0.1873	0.1965	0.2015	0.1919	0.2076	0.201	0.19887
17	4	0.1589	0.1626	0.1866	0.1627	0.16464	0.1597	0.1704	0.1775	0.1729	0.1675	0.168344
18	5	0.1612	0.1541	0.1609	0.1571	0.1573	0.1665	0.1561	0.1594	0.1483	0.1691	0.159
19	6	0.1617	0.1563	0.1518	0.1551	0.15353	0.162	0.1441	0.1559	0.1745	0.1472	0.156213
20	7	0.1627	0.1604	0.1748	0.1434	0.1512	0.1491	0.1536	0.1544	0.151	0.1409	0.15415
21	8	0.1521	0.1515	0.1512	0.1466	0.152	0.1437	0.1526	0.144	0.146	0.1512	0.14909
22												
23												
24												
25												
26												
27												
28												
29												





Parallel Execution Performance

By leveraging multiple threads to split the workload we were able to make a fairly efficient program to generate data with. As the thread count increased, the run time dropped for both runs with and without locks. Having more threads to do the computations allows the entire job to be completed sooner. There are limitations to this though. As the number of threads increases eventually the run time plateaus, and no longer improves with more threads.

Impact of Locks On Performance

In order to create a thread safe program, the use of mutex locks was required. In their absence this can lead to incorrect output as two threads may access data at the same time. One side effect of using a mutex is the compute time wasted waiting for the resource to unlock. When using mutex locks this effect was shown by observing the time to completion plateauing as more worker threads were added. Using locks raises the plateau so that the run time drop of multithreading is less pronounced.

In Conclusion from the Experiment

In conclusion, the use of multithreading decreases the run time but it will eventually plateau. The use of locks prevents race conditions while running the algorithm, but raises the plateau to a higher runtime. The Collatz conjecture is one of many mathematical concepts to be explored using machine computation, and is an excellent example of the utility of multiple threads