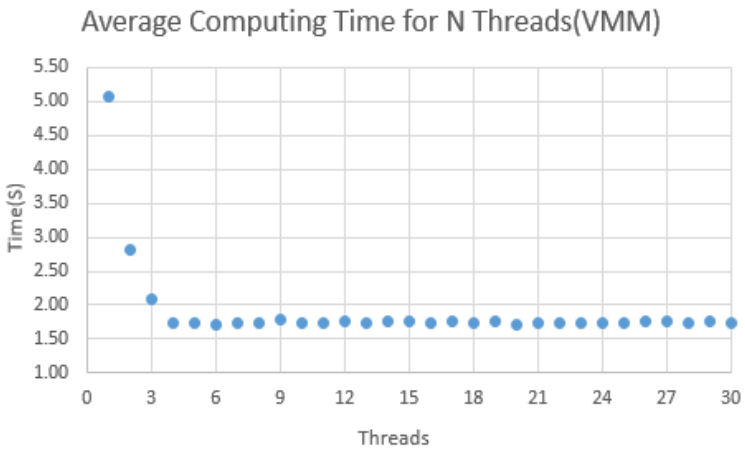
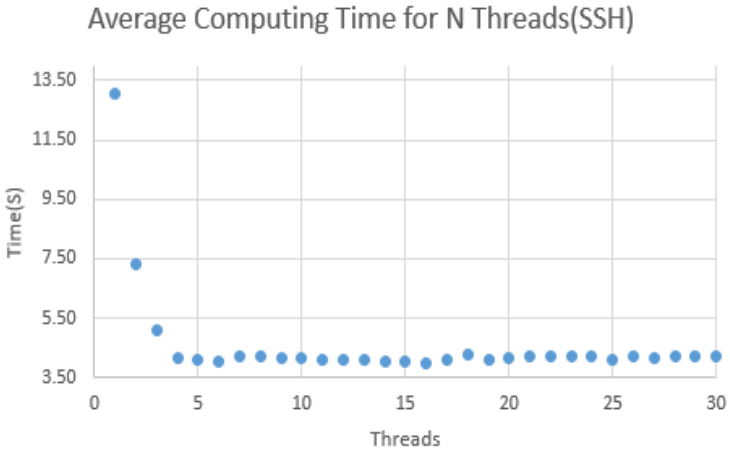
Darrell Percey & Andrew Boven

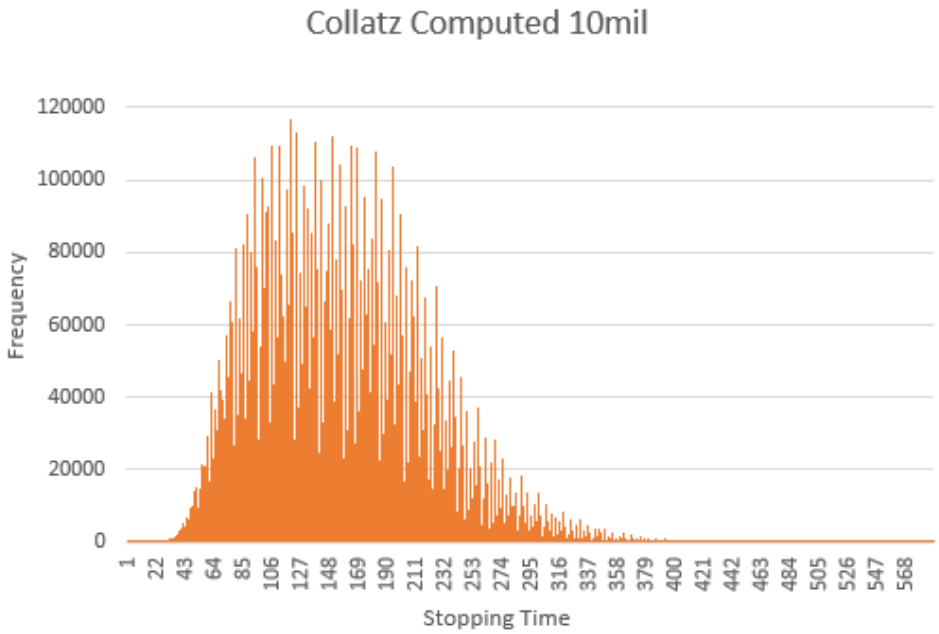
Project 2: Graph and Report

Virtual Machine (4 core utilization) N= 10mil					SSH Server N= 10mil				
Threads	Time1	Time2	Time3	Average Time	Threads	Time1	Time2	Time3	Average Time
1	5.060808	5.053600	5.119521	5.077976	1	13.058672	13.095585	13.060463	13.071573
2	2.797468	2.785722	2.840315	2.807835	2	7.369580	7.294465	7.380273	7.348106
3	2.088622	2.074136	2.064967	2.075908	3	5.155200	5.119622	5.019452	5.098091
4	1.715849	1.761239	1.719601	1.732230	4	4.099947	4.137663	4.173538	4.137049
5	1.703033	1.736648	1.728783	1.722821	5	4.195069	4.008021	4.102614	4.101901
6	1.730830	1.707523	1.719092	1.719148	6	4.080519	4.001613	4.096992	4.059708
7	1.732888	1.754903	1.712725	1.733505	7	4.250603	4.177625	4.152508	4.193579
8	1.719846	1.720837	1.729759	1.723481	8	4.221646	4.122530	4.272390	4.205522
9	1.836976	1.760445	1.771564	1.789662	9	4.168794	4.147672	4.187931	4.168132
10	1.729473	1.739115	1.729473	1.732687	10	4.056736	4.138716	4.210228	4.135227
11	1.724095	1.758388	1.757015	1.746499	11	4.157254	4.038055	4.043029	4.079446
12	1.745869	1.747293	1.796386	1.763183	12	4.135602	4.011478	4.136862	4.094647
13	1.713172	1.735360	1.756775	1.735102	13	4.184531	3.993550	4.124774	4.100952
14	1.759045	1.732315	1.784016	1.758459	14	4.037414	4.120123	4.018357	4.058631
15	1.765170	1.728531	1.796349	1.763350	15	4.029071	4.110808	4.079044	4.072974
16	1.741087	1.730637	1.742453	1.738059	16	3.939209	3.926791	4.087944	3.984648
17	1.753346	1.743524	1.753014	1.749961	17	4.046647	3.981051	4.207645	4.078448
18	1.743904	1.749822	1.731230	1.741652	18	4.270210	4.210751	4.368954	4.283305
19	1.753343	1.763619	1.758759	1.758574	19	4.146445	4.043691	4.126300	4.105479
20	1.714162	1.719218	1.729596	1.720992	20	4.202658	4.173950	4.120238	4.165615
21	1.736759	1.725672	1.743003	1.735145	21	4.108797	4.154791	4.311115	4.191568
22	1.739862	1.732891	1.720942	1.731232	22	4.204364	4.278279	4.260045	4.247563
23	1.712735	1.728522	1.774750	1.738669	23	4.223241	4.163366	4.189689	4.192099
24	1.735785	1.742466	1.740387	1.739546	24	4.263154	4.218310	4.206651	4.229372
25	1.759296	1.717632	1.726095	1.734341	25	4.123944	4.102880	4.140250	4.122358
26	1.724782	1.741636	1.797574	1.754664	26	4.327845	4.070719	4.196808	4.198457
27	1.781676	1.732918	1.734305	1.749633	27	4.197767	4.151283	4.170337	4.173129
28	1.736050	1.728555	1.744313	1.736306	28	4.307702	4.139014	4.215340	4.220685
29	1.753325	1.750535	1.751809	1.751890	29	4.246065	4.250569	4.144548	4.213727
30	1.714029	1.745452	1.717859	1.725780	30	4.199129	4.268661	4.205682	4.224491

Graphs







Report:

Each amount of threads I tested with multiple test runs just to get an average time. The times were all in the general same range. You can see after computing with 4 threads the computing plateaus. This is due to the threads being able to run on its own core. In the VMM there was 4 cores available for the program so it could process in parallel. However if we had a single core(&single processer) system then it would have relatively the same computing time for the entire range of 1-30 threads. Multithreaded programs if properly programmed can increase the performance of the program greatly. The problem most likely will be decided on which parts of the program can be ran on a thread separate from the program. Also you must take in account the user's CPU(even though the standard is a quad core nowadays).