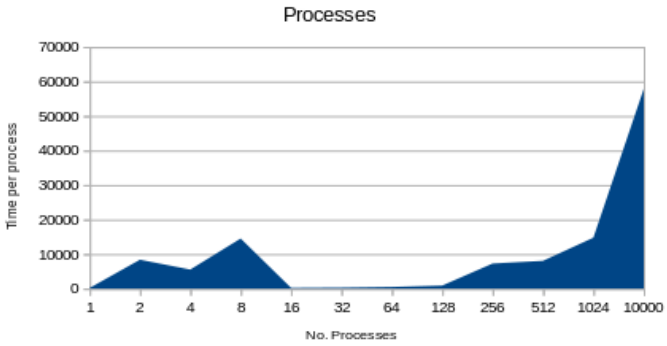
Processes

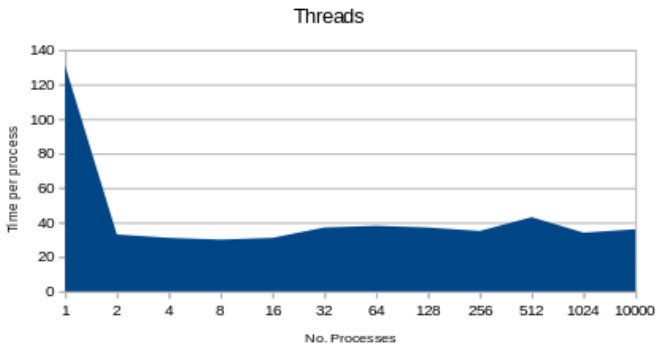
1 proc: fork= 123 (mics) wait= 18 (mics) sum= 141 (mics)
per proc: fork= 123 (mics) wait= 18 (mics) sum= 141 (mics)
bash-4.1$ 2 proc: fork=16685 (mics) wait= 3 (mics) sum=16688 (mics)
per proc: fork= 8342 (mics) wait= 2 (mics) sum= 8344 (mics)
4 proc: fork=21996 (mics) wait= 3 (mics) sum=21999 (mics)
per proc: fork= 5499 (mics) wait= 1 (mics) sum= 5500 (mics)
8 proc: fork=115915 (mics) wait= 3 (mics) sum=115918 (mics)
per proc: fork=14489 (mics) wait= 0 (mics) sum=14490 (mics)
16 proc: fork= 3270 (mics) wait= 4 (mics) sum= 3274 (mics)
per proc: fork= 204 (mics) wait= 0 (mics) sum= 205 (mics)
32 proc: fork= 8941 (mics) wait= 9 (mics) sum= 8950 (mics)
per proc: fork= 279 (mics) wait= 0 (mics) sum= 280 (mics)
64 proc: fork=29417 (mics) wait= 12 (mics) sum=29429 (mics)
per proc: fork= 460 (mics) wait= 0 (mics) sum= 460 (mics)
128 proc: fork=111035 (mics) wait= 24 (mics) sum=111059 (mics)
per proc: fork= 867 (mics) wait= 0 (mics) sum= 868 (mics)
256 proc: fork=1854439 (mics) wait= 38 (mics) sum=1854477 (mics)
per proc: fork= 7244 (mics) wait= 0 (mics) sum= 7244 (mics)
512 proc: fork=4101828 (mics) wait= 85 (mics) sum=4101914 (mics)
per proc: fork= 8011 (mics) wait= 0 (mics) sum= 8012 (mics)
1024 proc: fork=15085815 (mics) wait= 144 (mics) sum=15085959 (mics)
per proc: fork=14732 (mics) wait= 0 (mics) sum=14732 (mics)
10000 proc: fork=578706313 (mics) wait= 1742 (mics) sum=578708055 (mics)
per proc: fork=57871 (mics) wait= 0 (mics) sum=57871 (mics)

Threads

1 proc: thread create= 78 (mics) wait= 54 (mics) sum= 131 (mics)
per proc: thread create= 78 (mics) wait= 54 (mics) sum= 131 (mics)
2 proc: thread create= 53 (mics) wait= 14 (mics) sum= 67 (mics)
per proc: thread create= 26 (mics) wait= 7 (mics) sum= 33 (mics)
4 proc: thread create= 97 (mics) wait= 27 (mics) sum= 124 (mics)
per proc: thread create= 24 (mics) wait= 7 (mics) sum= 31 (mics)
8 proc: thread create= 188 (mics) wait= 56 (mics) sum= 244 (mics)
per proc: thread create= 23 (mics) wait= 7 (mics) sum= 30 (mics)
16 proc: thread create= 399 (mics) wait= 103 (mics) sum= 502 (mics)
per proc: thread create= 25 (mics) wait= 6 (mics) sum= 31 (mics)
32 proc: thread create= 970 (mics) wait= 201 (mics) sum= 1171 (mics)
per proc: thread create= 30 (mics) wait= 6 (mics) sum= 37 (mics)
64 proc: thread create= 2058 (mics) wait= 392 (mics) sum= 2450 (mics)
per proc: thread create= 32 (mics) wait= 6 (mics) sum= 38 (mics)
128 proc: thread create= 3891 (mics) wait= 794 (mics) sum= 4685 (mics)
per proc: thread create= 30 (mics) wait= 6 (mics) sum= 37 (mics)
256 proc: thread create= 8398 (mics) wait= 5573 (mics) sum=13971 (mics)
per proc: thread create= 33 (mics) wait= 22 (mics) sum= 55 (mics)
512 proc: thread create=18375 (mics) wait= 3554 (mics) sum=21929 (mics)
per proc: thread create= 36 (mics) wait= 7 (mics) sum= 43 (mics)
1024 proc: thread create=28882 (mics) wait= 6343 (mics) sum=35225 (mics)
per proc: thread create= 28 (mics) wait= 6 (mics) sum= 34 (mics)
10000 proc: thread create=289517 (mics) wait=69297 (mics) sum=358814 (mics)
per proc: thread create= 29 (mics) wait= 7 (mics) sum= 36 (mics)

t per thread	t per proc
1	131	141
2	33	8344
4	31	5500
8	30	14490
16	31	205
32	37	280
64	38	460
128	37	868
256	35	7244
512	43	8012
1024	34	14732
10000	36	57871





The time taken to execute each process is not linear with the number of processes created. Looking at the above graph, the time taken to execute a thread stays linear with the number of threads created.

We decided to use **threads** because we found that threads have a linear time to complete, which is unaffected by the number of threads.
Also, inter-process communication is harder and slower than inter-thread communication.