CS-214 Systems Programming
Professor Francisco

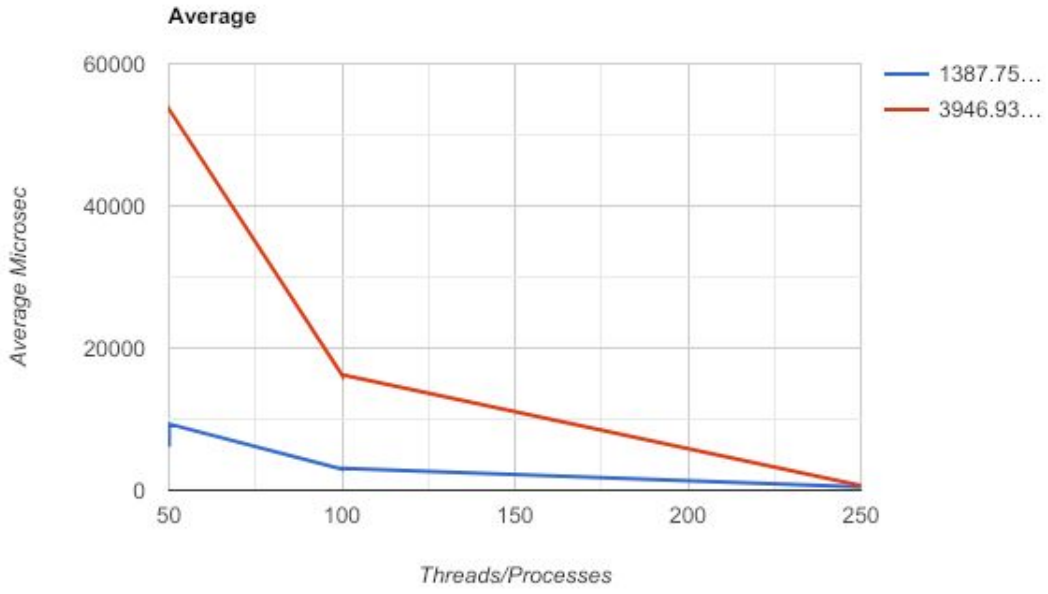
**Group Members:**

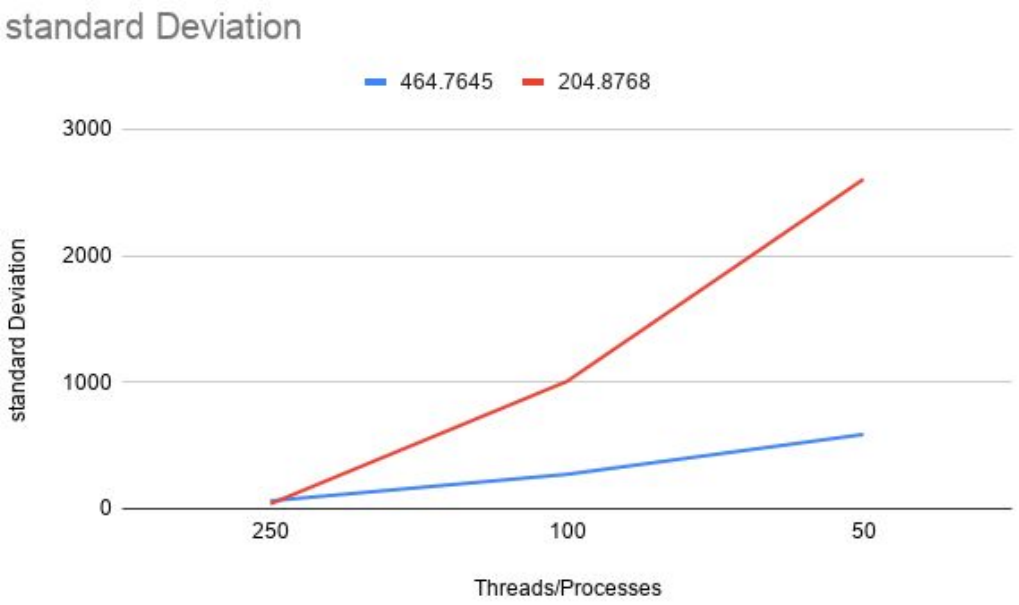
| | |
|---|---|
| Richard Travers | NetID: rdt58 |
| Syed Abbas Haider | NetID: sah300 |


**All test times are in microseconds.**

To get our data, we ran each test through a loop 150 times, and took the average, minimal time, maximum time, and standard deviation once the for-loop completed. We ran our program 2 times for both thread mode and Process mode to gather the appropriate data.

**Graph:**
**RED:** Processes
**BLUE:** Threads






**Threads**

Test A:

Run 1:

Average: 1401.786667
Standard Deviation: 462.839009
Minimum: 1177
Maximum: 8046

Run 2:

Average: 1373.72
Standard Deviation: 466.699341
Minimum: 1200
Maximum: 8166

Test B:

Run 1:

Average: 452.166667
Standard Deviation: 46.816857
Minimum: 410
Maximum: 1371

Run 2:

Average: 457.66
Standard Deviation: 79.057922
Minimum: 412
Maximum: 1830

Test C:

Run 1:

Average: 3150.14
Standard Deviation: 200.937947
Minimum: 2761
Maximum: 5824

Run 2:

Average: 2977.04
Standard Deviation: 82.91425
Minimum: 2695
Maximum: 5641

Test D:

Run 1:

Average: 3046.986667
Standard Deviation: 187.683071
Minimum: 2719
Maximum: 5651

Run 2:

Average: 2896.593333
Standard Deviation: 73.453443
Minimum: 2723
Maximum: 5502

Test E:

Run 1:

Average: 6282.006667
Standard Deviation: 299.309653
Minimum: 5583
Maximum: 11423

Run 2:

Average: 6054.953333
Standard Deviation: 257.886169
Minimum: 5535
Maximum: 11392

Test F:

Run 1:

Average: 6267.373333
Standard Deviation: 330.833023
Minimum: 5583
Maximum: 11241

Run 2:

Average: 5961.5
Standard Deviation: 287.245975
Minimum: 5535

Maximum: 11235

**Processes**

Test A:

Run 1:

Average: 4020.74
Standard Deviation: 205.753625
Minimum: 3729
Maximum: 8013

Run 2:

Average: 3873.126667
Standard Deviation: 204
Minimum: 3680
Maximum: 9807

Test B:

Run 1:

Average: 683.16
Standard Deviation: 49.945263
Minimum: 619
Maximum: 1357

Run 2:

Average: 652.993333
Standard Deviation: 28.301262
Minimum: 608
Maximum:1305

Test C:

Run 1:

Average: 16438.026667
Standard Deviation: 563.304191
Minimum: 15713
Maximum: 32660

Run 2:

Average: 16005.126667
Standard Deviation: 304.999741
Minimum: 15480
Maximum: 31345

Test D:

Run 1:

Average: 16486.02667
Standard Deviation: 805.497863
Minimum: 15565
Maximum: 32777

Run 2:

Average: 15956.493333
Standard Deviation: 339.776385
Minimum: 15406
Maximum: 31408

Test E:

Run 1:

Average: 54200.013333
Standard Deviation: 1828.019195
Minimum: 52247
Maximum: 107340

Run 2:

Average: 53010.546667
Standard Deviation: 900.630639
Minimum: 50578
Maximum: 103779

Test F:

Run 1:

Average: 54079.68
Standard Deviation: 1571.419386
Minimum: 52247
Maximum: 106406

Run 2:

Average: 53010.566667
Standard Deviation: 908.012895
Minimum: 50578
Maximum: 10424

**Overall Results:**

Overall in our findings, using threads to search was always faster than using Processes by a large amount. Not once in our tests did the average time for threads reach 5 digits, whereas the average runtime for Processes hit this time as soon as we tried to create 200 workers in Test C.

The closest comparable times between threads and Processes were in Test B, which created about 20 workers, where threads were only about 200 microseconds faster. We hypothesize the tradeoff between threads and Processes for our 5000 sized array to be somewhere above 250 units in size, but as we were unfortunately restricted to 250 and below, we could not find the exact tradeoff.

As for the parallelism tradeoff, decreasing the size of the threads/Processes always resulted in a higher time to search, no matter how many workers were created, therefore we

hypothesize that this tradeoff is also somewhere above 250 units for size, although it is most likely higher than the threads v. Processes tradeoff as the runtimes show.