Abhimanyu Singh
4/19/21

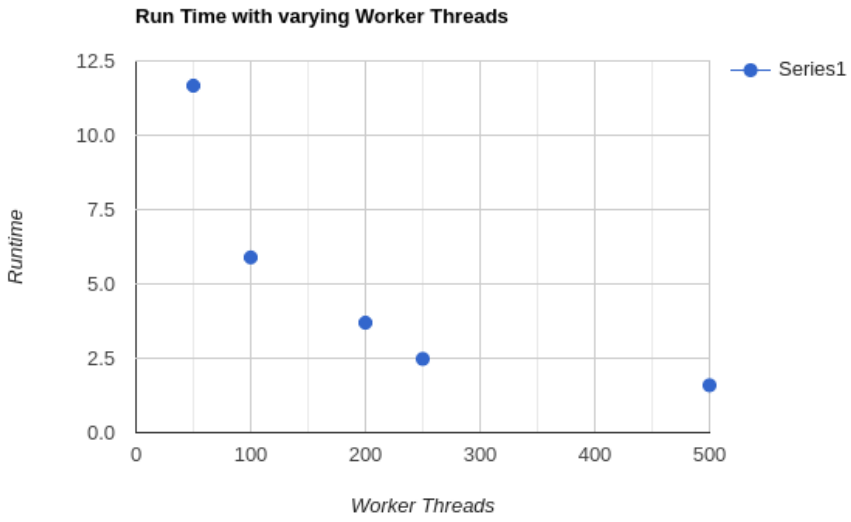
PA5: High Concurrency with a single worker thread

YOUTUBE LINKS: https://youtu.be/EbAHPKfJtRA

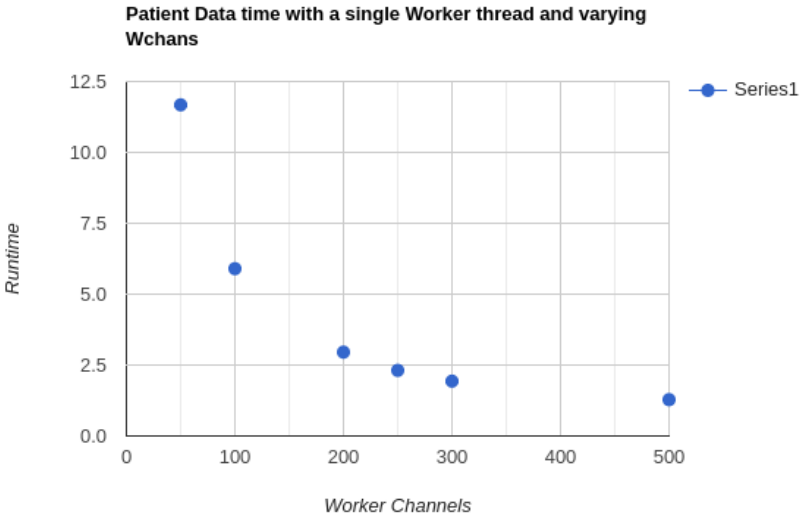
Timings:

 # 1 Varying worker threads vs varying worker channels @ -p 15 -n 15K -b 1024 -h 5

a

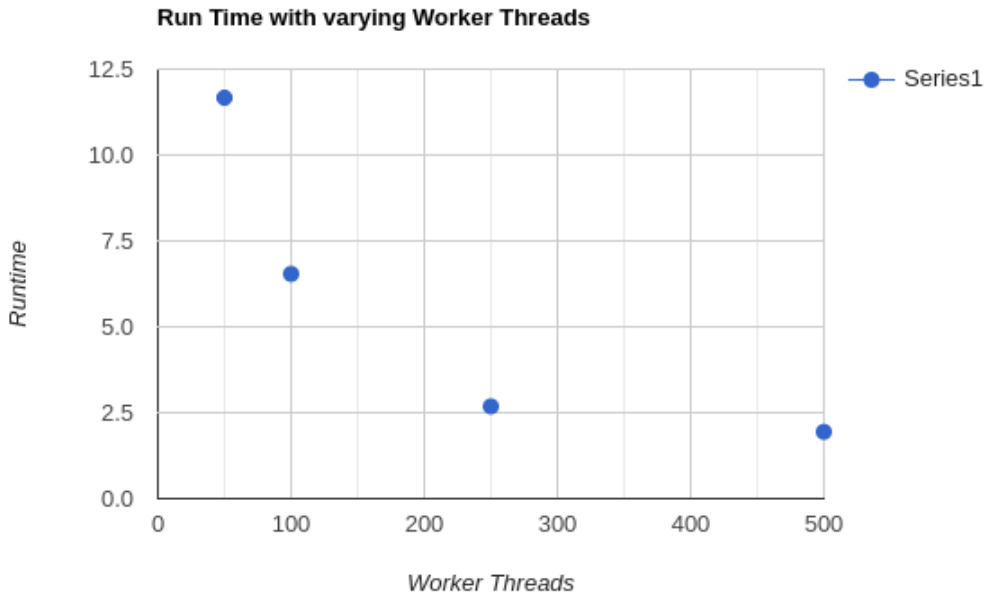


B

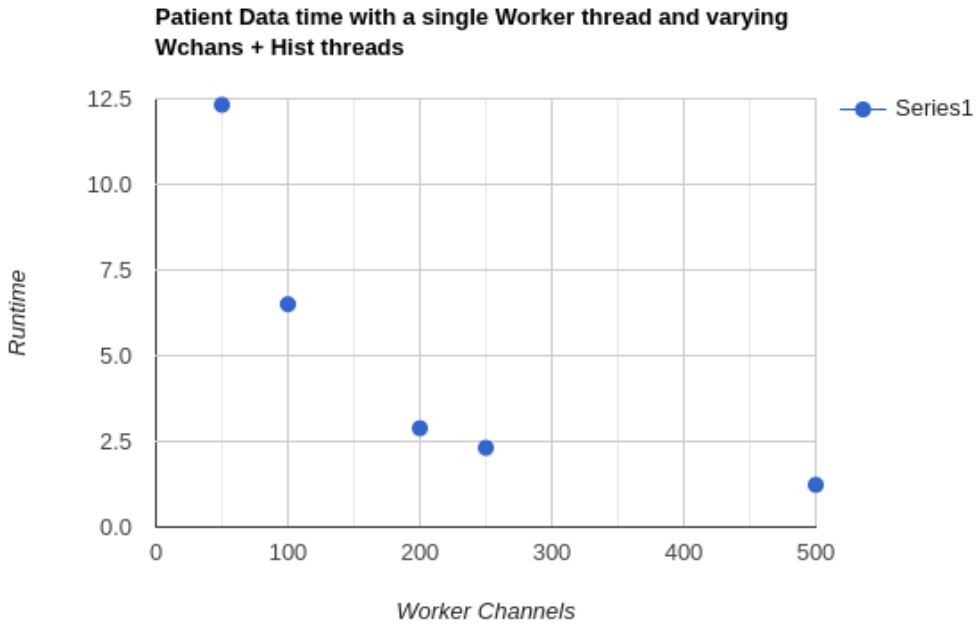


#2 Varying worker threads vs varying worker channels with various hist threads H(5, 50) -> W(50, 500)

a



B



The runtime difference between this PA and last PA's implementation was not very significant, and the differences between these two implementations skewed in both ways depending on the other programs that I was running such as zoom screen record. In the tests for my data I made sure to replicate the same machine with all other apps and programs closed. This resulted in a slight speedup, comparing 500 Worker Threads **@ 1.8 seconds** to 500 Worker Channels with a single thread **@ 1.25 seconds.** The speedup is definitely there but more interesting is that the bottleneck does not occur sooner, with the 500 Worker threads the bottleneck started to show around 350 Worker threads, with the PA5 implementation the bottleneck starts to show up beyond 500 Worker Channels, this is due to PA4 having less space for concurrent processes, when cwrite() is done then random server time limits the speedup after a certain amount of threads, with PA5 implementation we overcome this obstacle all while using just a single thread. This efficient implementation shines through not because its slightly faster, but because we eliminate the nonconcurrency of our threads and channels, now if we were to implement a way to speed this up more we could rely on more threads that work in concurrency, although it would not make a large difference unless we were requesting much larger chunks of data, on the order of 100s of MB.