

CS-342

Operating Systems

Section: 1

PROJECT 2

Mehmet Hasat Serinkan

21901649



Mehmet Eren Balasar

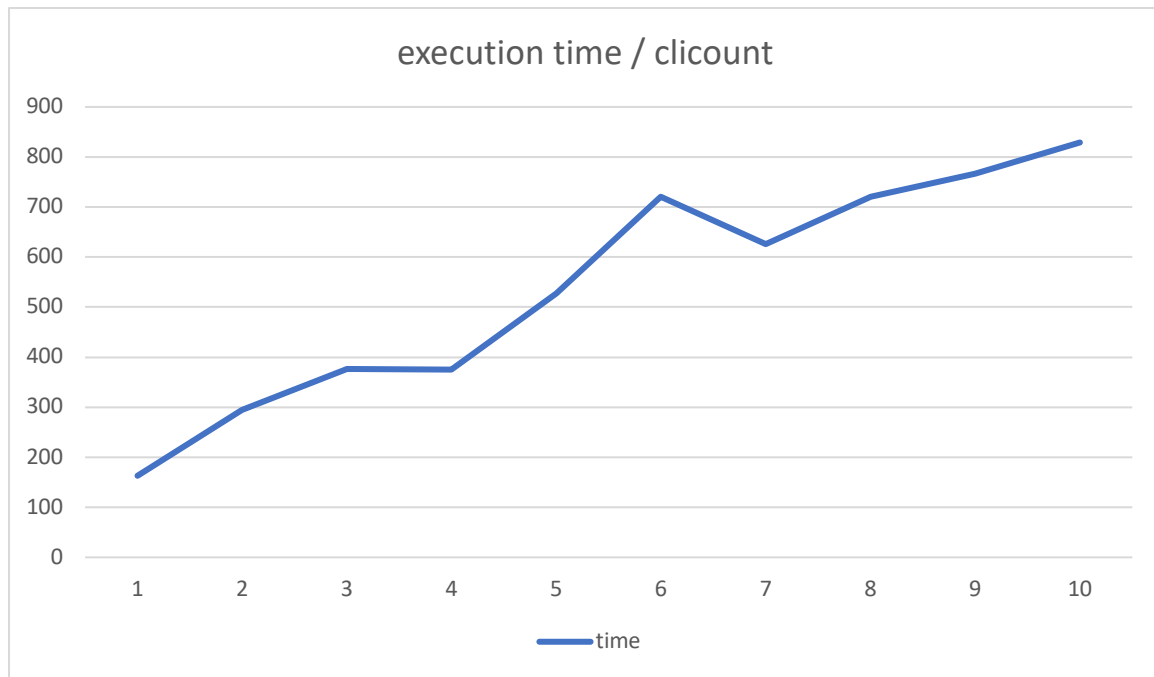
22001954

19.11.2023

Part 1

The effect of clicount (number of threads in client) (vsize = 64 dcount = 2 tcount = 5)



Execution Time 	ms
Clicount(number of Threads) 	
1	163
2	295
3	376
4	375
5	527
6	721
7	626
8	721
9	766
10	829

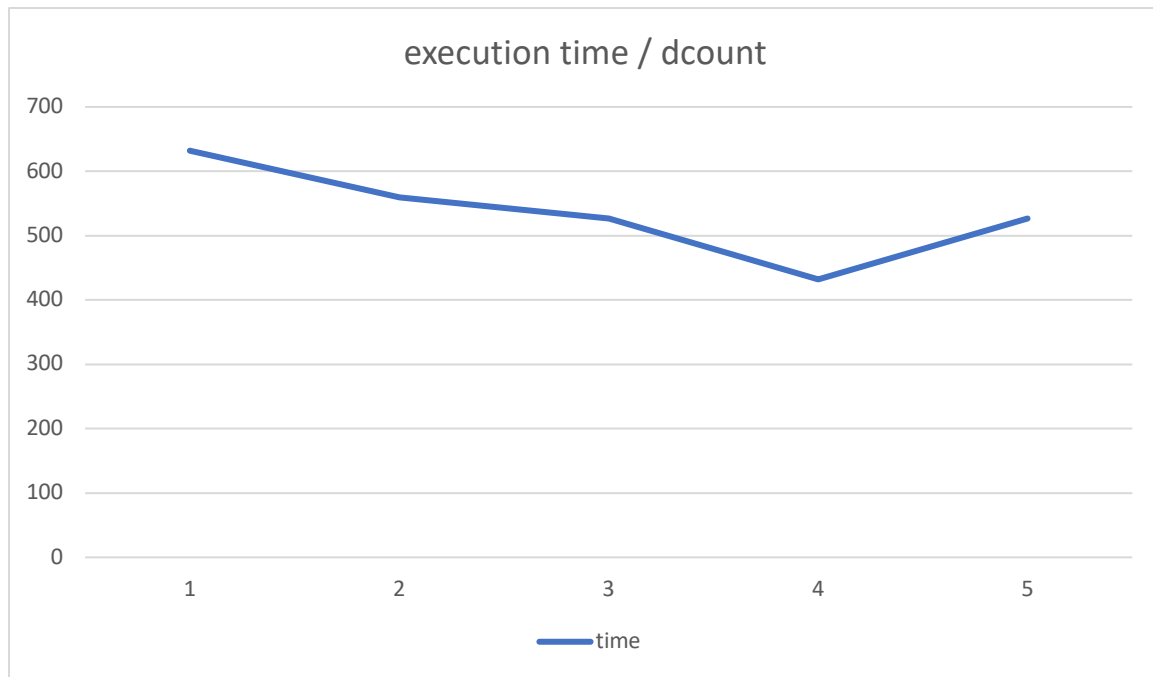


The increase in execution time with more client threads is primarily due to resource saturation, where the fixed number of server worker threads (5) can only handle so many requests simultaneously, leading to queuing and delays. Additionally, more threads mean greater synchronization overhead for managing access to shared resources, increased context switching by the operating system, and contention among threads for writing to the data files. This combination of factors causes a general trend of increased execution time as the number of client threads rises. The dip in time observed between certain thread counts may be due to variations in the test conditions or the specific nature of the requests being processed.

Part 2

The effect of dvalue (number of data files in server) (vsize = 64 tcount = 5 clicount = 5)



Execution Time 	ms
Dvalue(number of files) 	
1	632
2	559
3	527
4	432
5	393

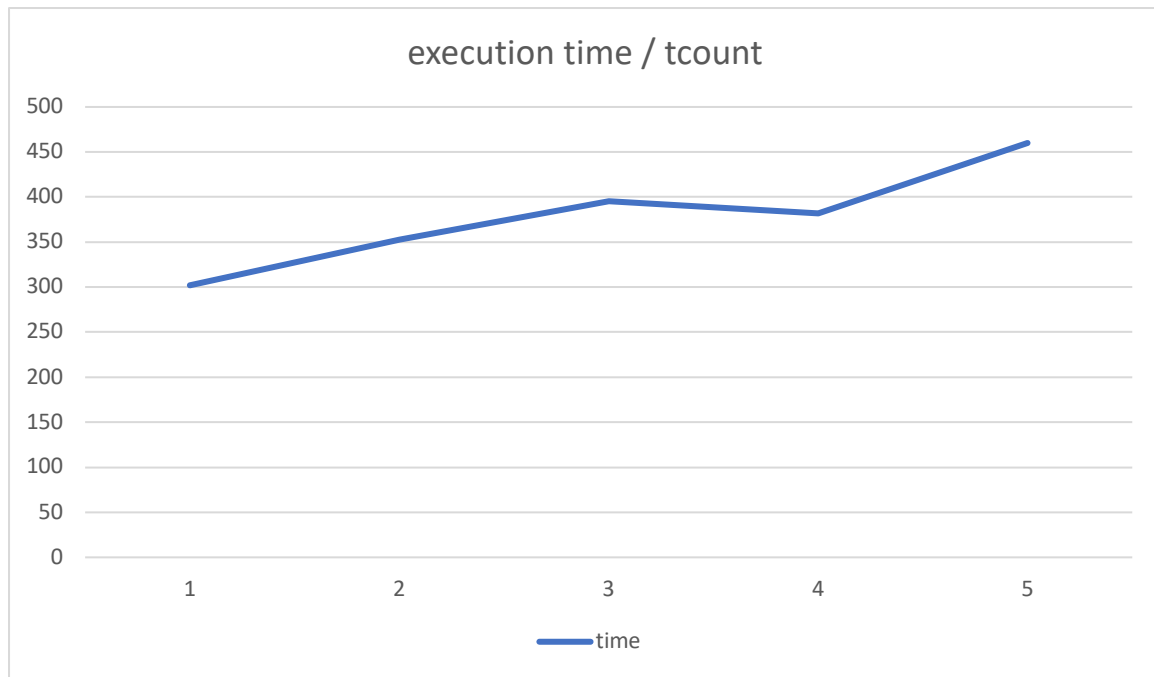


The decrease in execution time as the number of data files in the server increases can be attributed to reduced contention and better load distribution across the files. With only one file, all operations are queued for that single resource, creating a bottleneck. As more files are added, operations on different keys can be processed in parallel across these files, significantly improving throughput. The trend shown in the data suggests that up to a certain point, adding more files allows the server to handle more operations concurrently, which decreases the overall execution time. However, the slight uptick in execution time when going from four to five files may indicate overhead from managing an additional file or other system-specific factors that slightly reduce performance gains from increased parallelism. The most probable reason why there is not a strict decrease when data file count is increased might be about the keys we have used in request files. They are mostly mapped to same 3 files so even if you increase the data file count above 3, there is no performance increase.

Part 3

The effect of tcount (number of threads in server) (vsize = 64 dvalue = 5 clicount = 5)

Execution Time 	ms
tcount(number of Threads) 	
1	302
2	353
3	395
4	382
5	460



The fluctuating execution time with varying server worker threads suggests a complex interplay between parallel processing benefits and overheads from thread management and resource contention. Initially, more threads mean more overhead, which could increase execution times. However, at four threads, there seems to be an optimal point where the server efficiently handles multiple operations, decreasing the execution time. At five threads, the execution time increases again, likely due to the overhead and contention outweighing the benefits of additional parallelism. This indicates that there is a limit to how much concurrency improves performance before it becomes detrimental, possibly due to the server's specific workload, synchronization costs, and the overhead of managing an increased number of threads.

Part 4) Execution time of request

Execution time when dcount = 5 v-size=64 t=5

Put 1.8 ms

Get 0.9 ms

Del 1 ms

Execution time when dcount = 5 v-size=1024 t=5

Put 5 ms

Get 0.9 ms

Del 1.1 ms

Execution time when dcount = 5 v-size=64 t=5

Put 1 ms

Get 0.9 ms

Del 3 ms

Execution time when dcount = 5 v-size=1024 t=5

Put 1.1 ms

Get 2.2 ms

Del 0.5 ms

The key observation is that the execution time for PUT requests increases significantly when the value size (vsize) is larger (from 1.8 ms with vsize=64 to 5 ms with vsize=1024). This is expected as larger data sizes typically take longer to write to storage due to increased I/O time. GET requests, which read

data, show a small increase in execution time with larger value sizes, likely due to the longer time needed to read more data from storage. However, DELETE operations, which simply remove entries, are less affected by value size, showing mixed execution times across the examples. This variability must be attributed to the deletion algorithm we use, which is simply setting a flag to true.