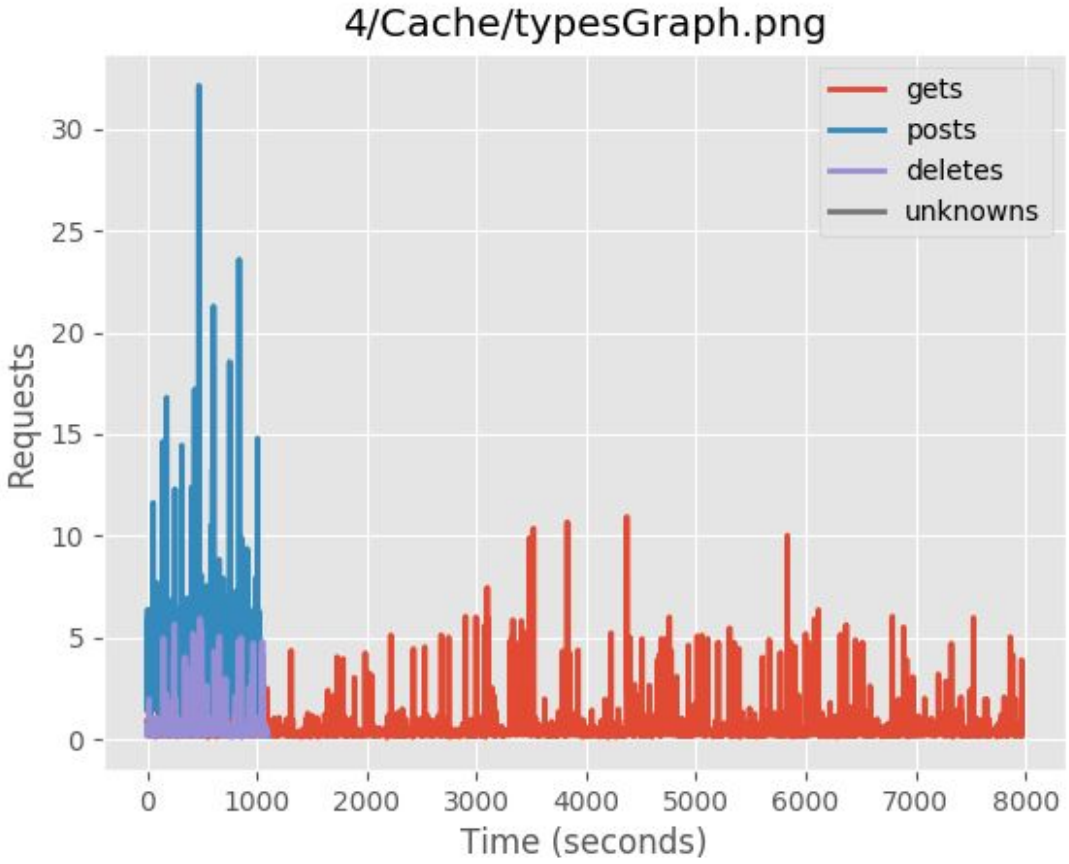
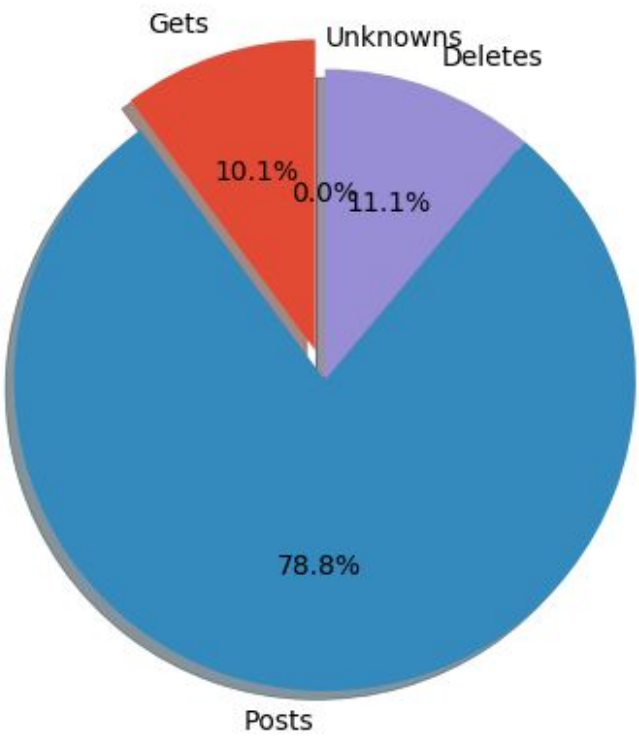
Michael Laucella
Multicore Programming

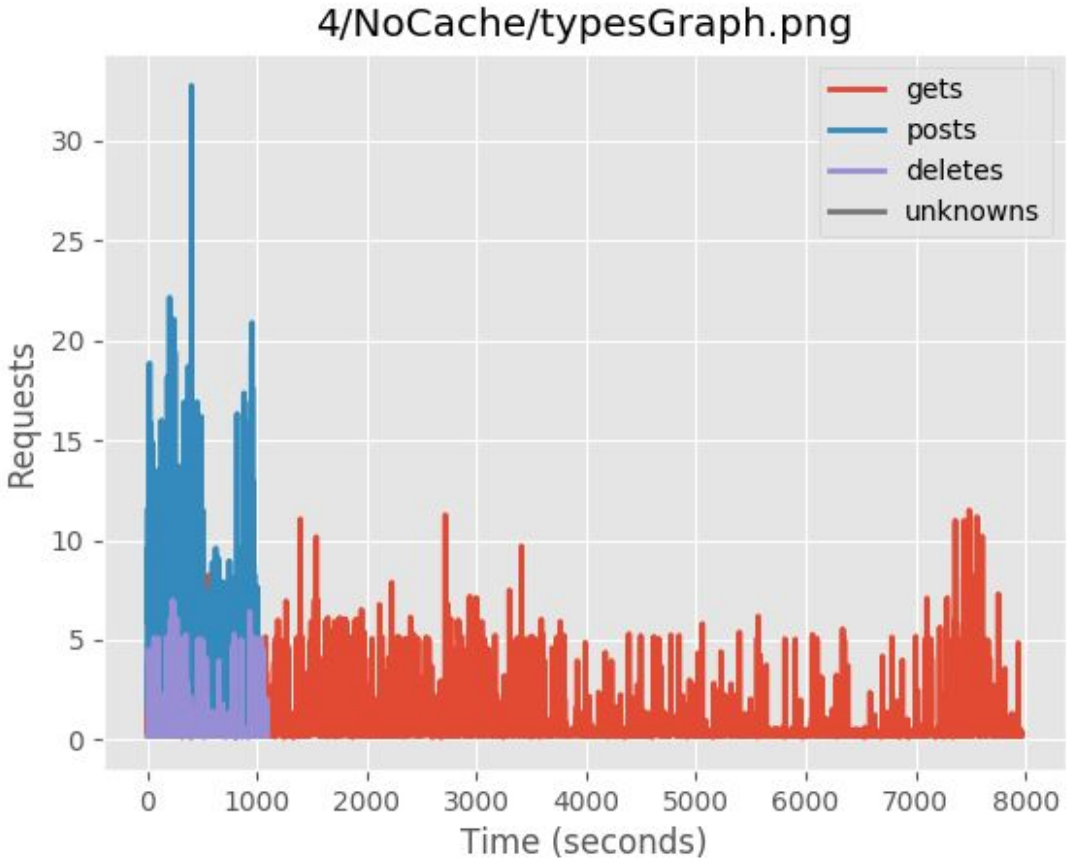
# LAB-4

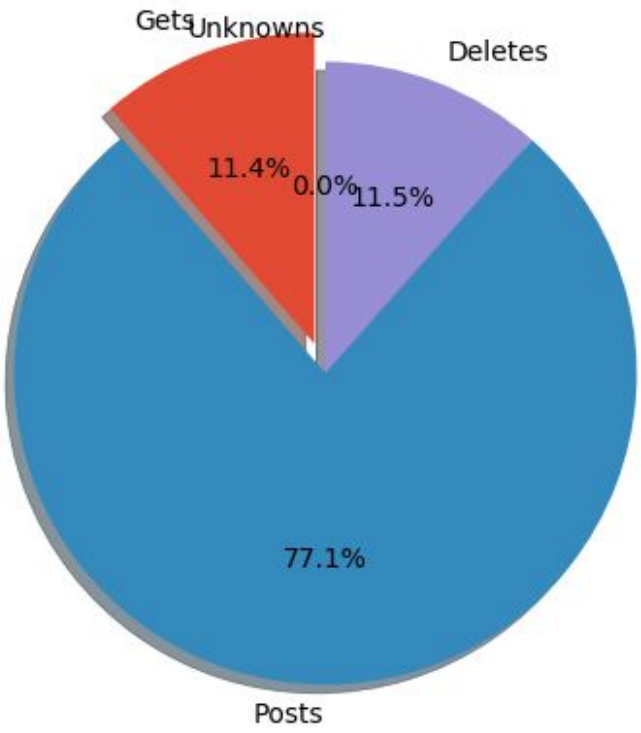
**note large segments of thread server cpp/hpp have been simplified, when running now statistics have now been integrated so no need for stats flag. This is also addressed in the makefile

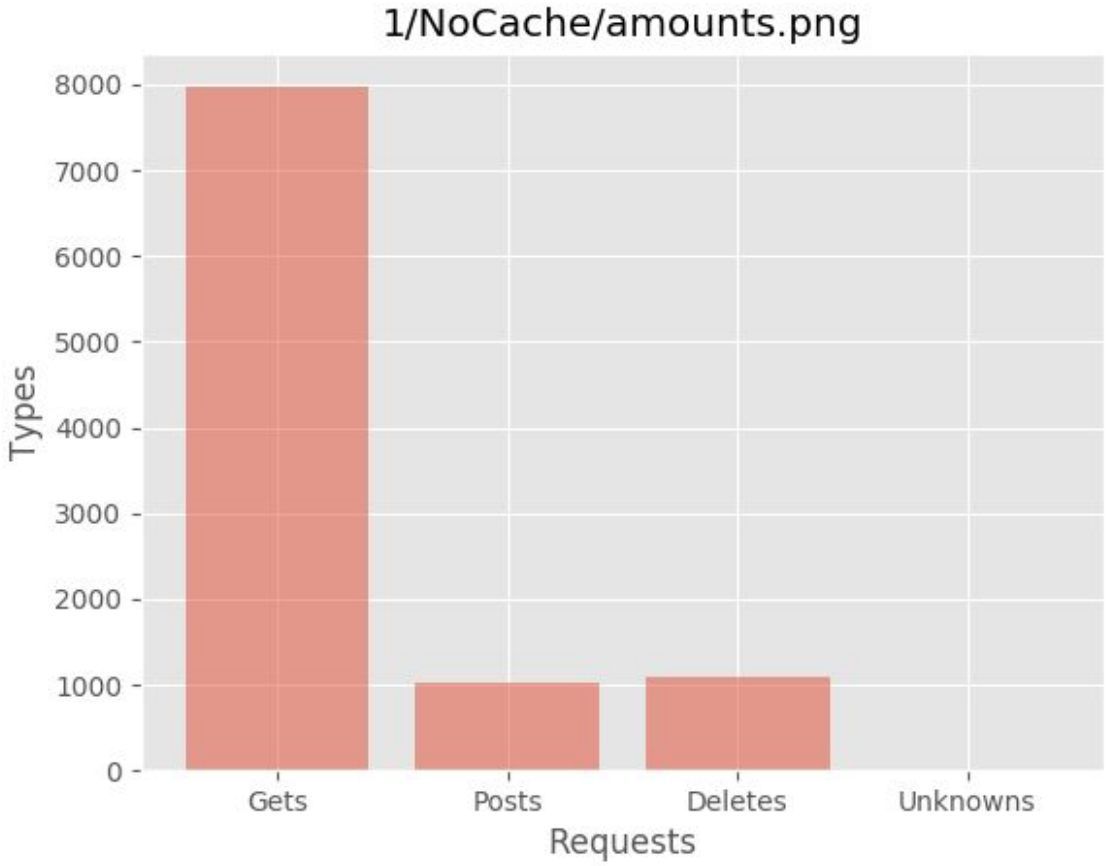
Beginning with an examination of the cached system on 4 threads (the trends hold decently throughout all the different number of threads) we see the posts have a huge impact on the time, why is it so much higher than deletes seeing as both use a write lock? Simple because the posts have to write data to the file and it can be a great deal of data whereas remove just frees the segment (doesn't overwrite the data just removes the logical structure pointing to it). Now in the Gets we can see some trends and spikes most likely based on the data set used we have 2.6k 200 and 7.4k 400 status codes this means many gets would be likely to fail as (from the amount table further below) we see there are about 8000 get requests. This would explain many of the flatter low lines in the get plot with those I would also throw in the lower lines < 5 as cached gets, this conclusion was reached after viewing the time of gets in the no cache system in which several more lines are around 5 giving the impression that it is the average file grab and read time. Analysis of the time of the no cache server shows posts had a much greater effect, a reason for this could be that although the cache is write-through the file system setup utilizes a read/write lock. If the cache picks up more of the reads then less readers can be present at any given time this would mean the writer would have to wait for less readers or even no readers to finish before writing. As mentioned before the get lines have significantly increased towards the 5 mark. Viewing the pie charts we also see that in the cached system a smaller percentage of time is used on gets 10.1% and for the no cache system 11.4% although it is only a one percent difference we see it has a large effect on the run time as the no cache average time is 1.03308 and the cache average is 0.68966 (stats.txt) which is approx a 1.4x speedup. A general view of the overall runtime can be seen below (for a better view see analyze/All_Runs_Big.png), here we can see the trend holding up even at 32 threads at the center of the graph the no cache plots are much greater than the cache plots. (viewing the charts within the analyze folders provide a greater amount of detail)

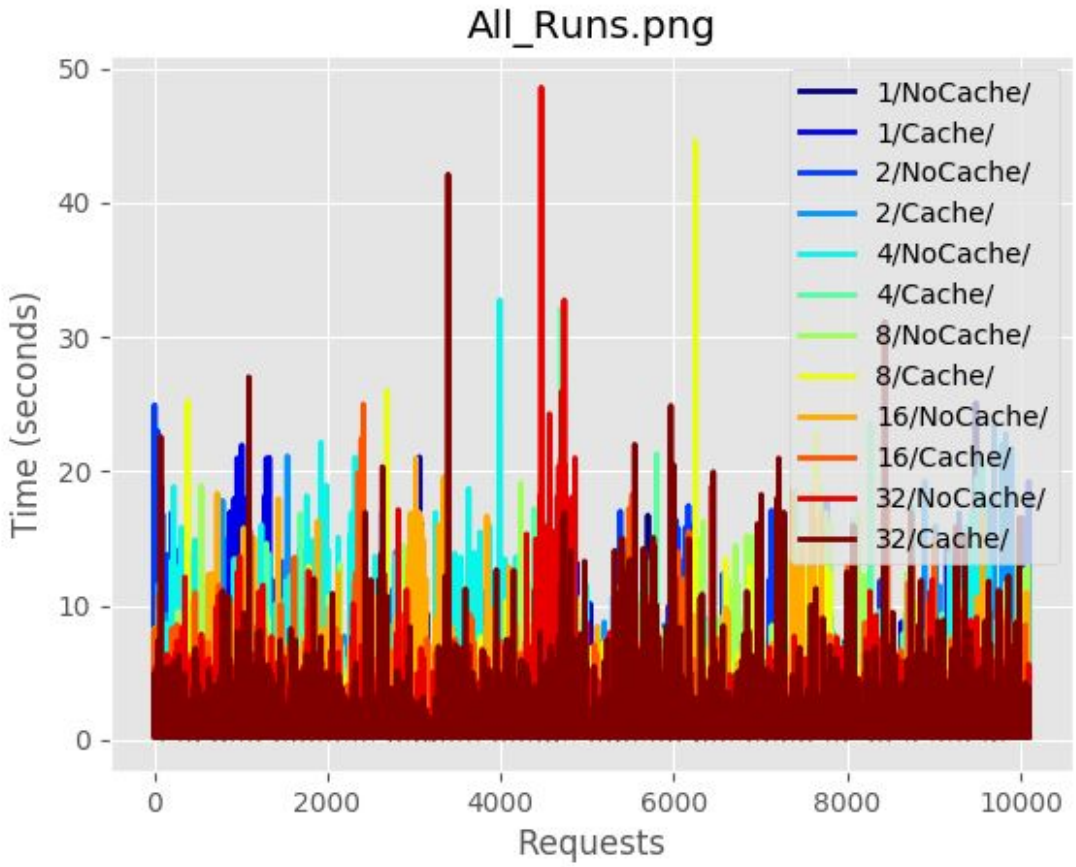

4/Cache/typesGraph.png
gets
posts
deletes
unknowns
Requests
0
5
10
15
20
25
30
0
1000
2000
3000
4000
5000
6000
7000
8000
Time (seconds)


Gets
Unknowns
Deletes
10.1%
0.0%
11.1%
78.8%
Posts


4/NoCache/typesGraph.png
gets
posts
deletes
unknowns
Requests
0
5
10
15
20
25
30
0
1000
2000
3000
4000
5000
6000
7000
8000
Time (seconds)


Gets
Unknowns
Deletes
11.4%
0.0%
11.5%
77.1%
Posts


1/NoCache/amounts.png
8000
7000
6000
5000
4000
3000
2000
1000
0
Types
Gets
Posts
Deletes
Unknowns
Requests


All_Runs.png
Time (seconds)
Requests
0
10
20
30
40
50
0
2000
4000
6000
8000
10000
1/NoCache/
1/Cache/
2/NoCache/
2/Cache/
4/NoCache/
4/Cache/
8/NoCache/
8/Cache/
16/NoCache/
16/Cache/
32/NoCache/
32/Cache/