Project 2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Part 1 of 1 -** | 105.0 Points |  |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | | Question 1 of 7 | 20.0 Points |  |  | | --- | |  |   Fill in the blanks so that the set of experiments below can let you reason about the impact of multithreading on the GETFILE server when handling large files.   Assume FS1 and FS2 are appropriately selected file sizes.   1. RUN\_MIXED( 1 , MIXED\_FILES, M1, M2)  2. RUN\_MIXED( 10 , MIXED\_FILES, M1, M2)  3. RUN\_MIXED( 100 , MIXED\_FILES, M1, M2)  4. RUN\_MIXED( 1000 , MIXED\_FILES, M1, M2) | | |  |  | | --- | --- | | Question 2 of 7 | 20.0 Points |  |  | | --- | |  |   Fill in the blanks so that the set of experimental runs shown below can produce some meaningful results about the impact of multithreading on the server.   Assume that T1-T4 correspond to appropriately chosen thread-count values.   1. RUN\_FIXED(T1, FIXED\_SIZE, 1048576 )  2. RUN\_FIXED(T2, FIXED\_SIZE, 1048576 )  3. RUN\_FIXED(T3, FIXED\_SIZE, 1048576 )  4. RUN\_FIXED(T4, FIXED\_SIZE, 1048576 ) | | |  |  | | --- | --- | | Question 3 of 7 | 20.0 Points |  |  | | --- | |  |   You need to perform two sets of experiments to understand the impact of multithreading in situations when the workload fits in the file cache (i.e., in memory) vs. when it does not.   Each set of experiment consists of 4 runs. Fill in the blanks to represents each of these runs such that you end up with meaningful data.   Assume the hardware platform has 20kB cache, 20MB main memory, and 20GB disk.    A. First set of experiments:  1. RUN\_FIXED(50, FIXED\_SIZE, 10240);  2. RUN\_FIXED(50, FIXED\_SIZE, 15360);  3. RUN\_FIXED(50, FIXED\_SIZE, 1048576);  4. RUN\_FIXED(50, FIXED\_SIZE, 10485760);   B. Second set of experiments:  1. RUN\_FIXED(50, FIXED\_SIZE, 15728640);  2. RUN\_FIXED(50, FIXED\_SIZE, 20971500);  3. RUN\_FIXED(50, FIXED\_SIZE, 41943040);  4. RUN\_FIXED(50, FIXED\_SIZE, 83886080); | | |  |  | | --- | --- | | Question 4 of 7 | 20.0 Points |   In no more than 2-3 sentences explain what you observe happening in the following situation, and provide your rationale as to why you think this trend is occurring.   For a request patter consisting of repeated requests for the same 10kB file, how is throughput impacted by adding more parallelism in the GETFILE server?   |  | | --- | |  |   As the thread increases in the beginning, we add concurrency so the throughput increases, however, as we keep adding more thread, the overhead for thread context switch outweighs the added concurrency benefit and becomes the dominant factor, so that we could see the throughput decreases and remains at some level. | | |  |  | | --- | --- | | Question 5 of 7 | 20.0 Points |   In no more than 2-3 sentences explain what you observe happening in the following situation, and provide your rationale as to why you think this trend is occurring.   What are the differences between the curves representing throughput as a function of number of threads obtained when requesting repeated the same sequence of 1KB files (1st curve) vs. 10MB files (2nd curve). Why do you think these differences occur?  Assume the same machine characteristics as in Q3.   |  | | --- | |  |   For 1KB curve, the file is stored in cache and it is very small, adding threads initially would increase concurrency and increase throughput, however, as more threads are added, the context switch time would actually become dominate and outweighs the benefits of multithreading as the file is small enough and stored in the cache and does not need context switch for single thread's one time access, thus throughput would decrease. For 10MB curve, the file is stored in the memory, accessing the 10MB file is memory bound and one thread takes a lot more context switches to finish and the throughput would eventually be flat as the overhead of context switch amortizes over long period of time. | | |  |  | | --- | --- | | Question 6 of 7 | 5.0 Points |  |  | | --- | |  |   (**EXTRA CREDIT:**) How many cores do you think are available on the machine used to run the experiments that generated the data for this assignment?   Answer: 4 cores. | | |  |  | | --- | --- | | Question 7 of 7 | 0.0 Points |   If you wish to provide additional clarification to any of you answers, use the space below.   |  | | --- | |  |   For Question 3.     Two sets of experiments are designed as following:   The first one is trying to test file size difference between cache and memory bound. The second experiment is to test file size difference between memory and disk bound.    For Question 6,    The run experiment is set with only 1 byte fixed size in order to make sure context switch overhead would not be a distraction factor in the experiment.    RUN\_FIXED(1, FIXED\_SIZE, 1);          RUN\_FIXED(2, FIXED\_SIZE, 1);      RUN\_FIXED(3, FIXED\_SIZE, 1);          RUN\_FIXED(4, FIXED\_SIZE, 1);       RUN\_FIXED(5, FIXED\_SIZE, 1);       RUN\_FIXED(6, FIXED\_SIZE, 1);       RUN\_FIXED(8, FIXED\_SIZE, 1);       RUN\_FIXED(10, FIXED\_SIZE, 1);      RUN\_FIXED(11, FIXED\_SIZE, 1);      RUN\_FIXED(12, FIXED\_SIZE, 1);      RUN\_FIXED(16, FIXED\_SIZE, 1);      RUN\_FIXED(18, FIXED\_SIZE, 1);      RUN\_FIXED(20, FIXED\_SIZE, 1);      RUN\_FIXED(21, FIXED\_SIZE, 1);      RUN\_FIXED(29, FIXED\_SIZE, 1);      RUN\_FIXED(30, FIXED\_SIZE, 1);      RUN\_FIXED(31, FIXED\_SIZE, 1);      RUN\_FIXED(39, FIXED\_SIZE, 1);      RUN\_FIXED(40, FIXED\_SIZE, 1);       RUN\_FIXED(41, FIXED\_SIZE, 1);       RUN\_FIXED(49, FIXED\_SIZE, 1);       RUN\_FIXED(51, FIXED\_SIZE, 1);       RUN\_FIXED(61, FIXED\_SIZE, 1);    The results show that the throughput reaches peak when thread count is from 3 to 8. It is possible that the core number is within that range. Since not all the thread counts are sampled in the experiment, I picked up 4 as the possible core number. | |