MA Ph.D. Qualifying Exam

Fall 2010

Directions:

Use careful reasoning to develop the answers to each of the following THREE questions. The soundness of your explanations count more than the completeness of your answers. Be sure to state any assumptions you make in your solutions. Note that it's fine not to simplify numerical answers. You may use the LZGS text for reference for this exam.

1. Explain in detail why the mean residual life of the customer in service, if any, at an arrival instant in an M/G/1 queue is equal to:

$$(S/2)[1+C_X^2]$$

2. Consider a query-processing system for a large social networking site. Assume the query processing system is compute-bound, and makes I/O requests to a large backend processing system. At any given point in time, let N be the number of users interacting with the site. The users issue five types of queries. Let p_i denote the probability that an incoming query is of type i, D_i denote the average total processing time for the query (including the processing time required to send requests to the backend system), and R_i (N) denote the average total backend time for type i queries when N clients are interacting with the system. Each query type i also generates a small amount of further processing time, with average $S_i << D_i$, that occurs sometime after the query completes and the response is sent to the user. These further small processing times are needed for asynchronous garbage collection, which shares the cpus with other queries that are being processed at the later time. A total of 32 cores perform the query processing (and garbage collection), with perfect load balancing across the cores.

   a) Provide a model that computes the query processing throughput for the social networking site for given values of N, the average user think time Z, and for each type of query, $\{p_i,D_i,S_i,R_i\}$.

   b) Considering both low ("non-peak") values of N and high ("peak") values of N, corresponding to utilization of each core under 30% and utilization of each core over 80%, respectively, what is the quantitative impact of the asynchronous garbage collection compared with an alternate (hypothetical) implementation in which the garbage collection is performed synchronously before the reply returns to the user (i.e., the average total processing time for a query of type i is equal to $D_i + S_i$, and no further processing is required after the query completes)?

SEE NEXT PAGE.

3. In hardware-supported fine-grained multi-threading, each core can execute K threads concurrently. The core issues a total of one instruction per clock cycle, and cycles among the threads that can issue in round robin order, issuing one instruction from each thread in turn. When a thread issues a Load or Store request that misses in the L1 cache, the cache line is requested from the L2 cache. If the L2 cache doesn't have the cache line, the request is forwarded to the main memory module that contains the cache line. For this problem, assume all requests that miss in the L2 cache can be served by the main memory. A cache block from main memory is returned to the L2 cache and then to the L1 cache. Assume a multi-ported L1 cache with negligible contention, and negligible contention for the communication paths between the L1 cache, L2 cache, and main memory. However, assume the L2 cache can serve only one request at a time. Similarly, each memory module only serves one request at a time. Contention for the L2 cache and main memory is not negligible.

   There are four cases for which a thread can't issue an instruction. One is that the thread is waiting for an L2 cache request. The second is that the thread is waiting for the completion of an instruction that it previously issued, due to a data dependence in the next instruction it will issue. The third case is that the thread waits a few cycles to ensure that the next instruction will complete execution after the previously issued instructions (i.e., due to the in-order execution requirement). The fourth case is when the thread is waiting for a lock or barrier or some other hardware-supported synchronization event. Note that during these four types of thread "stalls", the core will cycle among the other threads that have instructions to issue. Hence, multithreading generally improves system performance. Note also that total cpu processing rate decreases when more threads are waiting for L2 cache requests.

   Given the following workload and system input parameters, develop any additional parameters and notation needed, as well as a model that predicts the speedup for a core with K threads compared to a core with one thread.

K:	number of threads
M	number of memory modules
S_{L2}	time to read or write a cache line in the L2 cache, in cpu clock cycles
S_M	time to read or write a cache line in main memory, in cpu clock cycles
f_1	fraction of instructions that are Load/Store instructions that miss in the L1 cache
f_2	fraction of requests to the L2 cache that miss in the L2 cache
t	the average number of cycles per instruction that don't miss in the L1 cache during which the thread is stalled for data dependence and/or in-order execution
N	number of cores, each with their own L1 cache, that share the L2 cache and main memory
l_1	average time to transmit a request/reply between the cpu and the L2 cache, in cpu clock cycles
l_2	average time to transmit a request/reply between the L2 cache and main memory, in cpu clock cycles