Seminar - Processes

Professor version

Prof. Leonardo Mostarda,
School of Science and Technology,
Camerino University, Italy
Version 2.0, 1 March 2013

Question 1

In this problem you are to compare reading a file using a single-threaded file server and a multithreaded server. It takes 15 msec to get a request for work, dispatch it, and do the rest of the necessary processing, assuming that the data needed are in a cache in main memory. If a disk operation is needed, as is the case one-third of the time, an additional 75 msec is required, during which time the thread sleeps. How many requests/sec can the server handle if it is single threaded? If it is multithreaded?

Solution 1

In the single-threaded case, the cache hits take 15 msec and cache misses take 90 msec. The weighted average is 2/3 x 15 + 1/3 x 90. Thus the mean request takes 40 msec and the server can do 25 per second. For a multithreaded server, all the waiting for the disk is overlapped, so every request takes 15 msec, and the server can handle 66 2/3 requests per second.

Question 2

Would it make sense to limit the number of threads in a server process?

Solution 2

Yes, for two reasons. First, threads require memory for setting up their own private stack. Consequently, having many threads may consume too much memory for the server to work properly. Another, more serious reason, is that, to an operating system, independent threads tend to operate in a chaotic manner. In a virtual memory system it may be difficult to build a relatively stable working set, resulting in many page faults and thus I/O. Having many threads may thus lead to a performance degradation resulting from page thrashing. Even in those cases where everything fits into memory, we may easily see that memory is accessed following a chaotic pattern rendering caches useless. Again, performance may degrade in comparison to the single-threaded case.

Question 3

We described a multithreaded file server, showing why it is better than a single-threaded server and a finite-state machine server. Are there any circumstances in which a single-threaded server might be better? Give an example.

Solution 3

Yes. If the server is entirely CPU bound, there is no need to have multiple threads. It may just add unnecessary complexity. As an example, consider a telephone directory assistance number for an area with 1 million people. If each (name, telephone number) record is, say, 64 characters, the entire database takes 64 megabytes, and can easily be kept in the servers memory to provide fast lookup.

Question 4

Compare threads and Processes.

Solution 4

- Multithreading leads to performance gain(creation and switching are faster)
- A multithread system maintains minimum information in order to allow different threads to share the same CPU
- Switching can take place in the user space
- Processes require Interprocess Communication (IPC) (pipes, message queues, shared memory) which require kernel intervention
- Thread are not automatically protected against each other (no concurrency transparency)
- Developing multithreading application requires additional intellectual effort