Parallel Programming Design Notes

Bill Rubin

February 23, 2010

- **Input Queue:** Each thread has its own input queue of numbers. All accesses to each queue must be made thread-safe (serialized).
- **Worker Thread Overview:** A worker thread takes the first number from its input queue, processes it (actually does some *Hard Work*, see below), and pushes the resulting number onto the input queue of the next thread. The last thread has no "next thread", so it just throws away the result.
- **Enqueueing Event:** After a thread pushes a number onto a queue, it notifies the (possibly waiting) worker thread by setting an "item enqueued" event.
- **Main Thread:** The main thread generates 1,000 random numbers and pushes them, one at a time, onto the queue of the first thread. Then it requests the first thread to terminate, by setting a terminate event for that thread, then waits for that thread to terminate, and then exits.
- **Worker Thread Event Handling:** A worker thread begins by waiting for an event, either "item enqueued" or "terminate thread". When either event occurs, the worker thread iteratively processes the queue elements until the queue is empty. Then …
- **Orderly Termination:** … if the event was "item enqueued", the thread waits for another event, but if the event was "terminate thread", the thread requests the next thread to terminate (if this isn't the last thread), waits for that to happen, and then exits itself. In this way, all worker threads and the main thread terminate in an orderly way in reverse order. (N-th thread terminates first; main last.)
- **Hard Work:** The initial random number is between 1 and 10,000. Every number on every queue is in that range. The hard work done by each thread is as follows: Given input integer k, square it, and then take the residue modulo 9999 of that. Repeat this 10^6 times. This takes about 10 mS on one of my 2.5 GHz processors.
- **Minimal contention among threads:** The design is such that all threads essentially operate independently and in parallel on scalars. The only contention among threads is in the relatively short time where queues are being accessed.
- **Performance** (rough measure): With one worker thread, a run consists of 2×10^6 floating operations for each of 10^3 numbers, or 2×10^9 operations. This run takes about 10 seconds. Therefore, we have 2×10^8 FLOPS. But that one worker thread can run on only one processor at a time, and there are 4 processors in my Xeon quad. Therefore, the total measured power is $4 \times 2 \times 10^8$ FLOPS, or 8×10^8 FLOPS, or close to 10^9 FLOPS or **1 GFLOPS**. Reasonable, for a total processor power of 10 GHz.