CMPT 300

Assignment 2
Context Switching

Avneet Kaur
avneetk@sfu.ca
301209987

Minimal Function Call

The cost of a minimal function call was calculated by using the timer code provided in hr-timer.c. The CLOCK_PROCESS_CPUTIME was used since it measures the time taken by that particular process. The measurement methodology to calculate the cost of a minimal function call was to call an empty function, requiring no argument and having no return value. The time before and after the function call was recorded using clock_gettime(). The time taken by the function call was then found by calculating the difference between the start and end times.

This process was repeated 10,000 times to get a stable value. As seen in Figure 1, the time taken was higher in the first run, and then decreased for the subsequent runs. The average cost of a minimal function call was found to be ≈ 300 ns

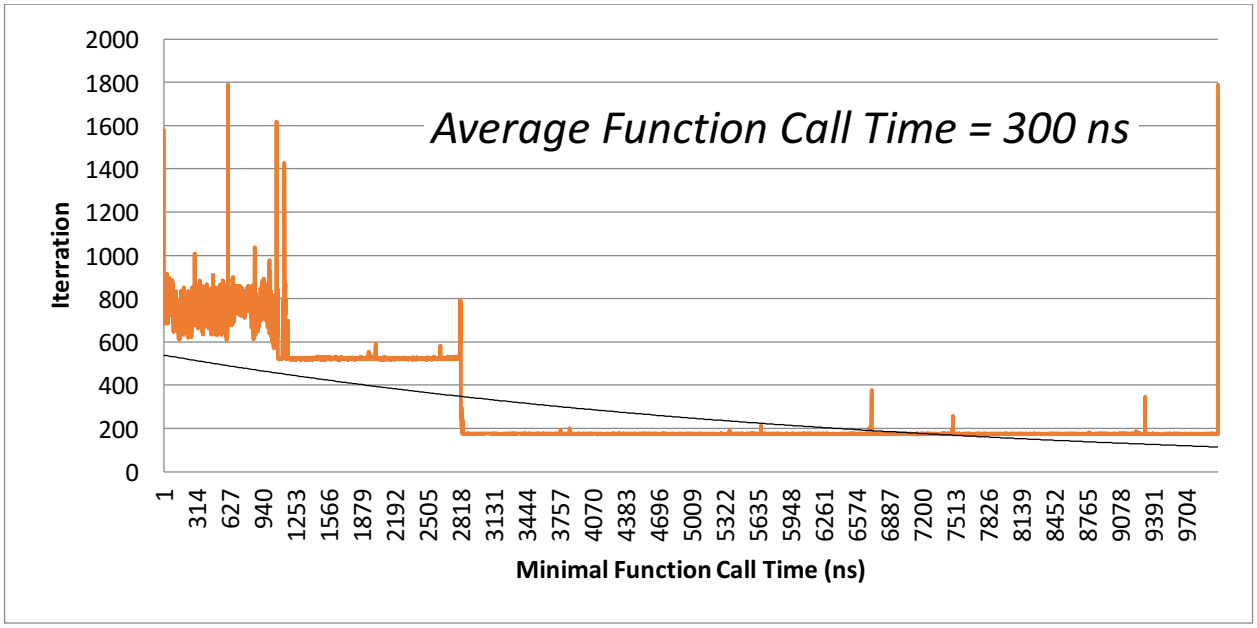

Figure 1: Minimal Function Call

Minimal System Call

The cost of a minimal function call was calculated by using the timer code provided in hr-timer.c. The CLOCK_PROCESS_CPUTIME was used since it measures the time taken by that particular process. The measurement methodology to calculate the cost of a minimal system call was to call run the getpid() system call, which just returns the process ID of the current process and is a very cheap function. The time before and after the system call was recorded using clock_gettime(). The time taken by the system call was then found by calculating the difference between the start and end times.

This process was repeated 10,000 times to get a stable value. As seen in Figure 2, the time taken was higher in the first run, and then stayed constant for the subsequent runs. The average cost of a minimal system call was found to be ≈ 112 ns

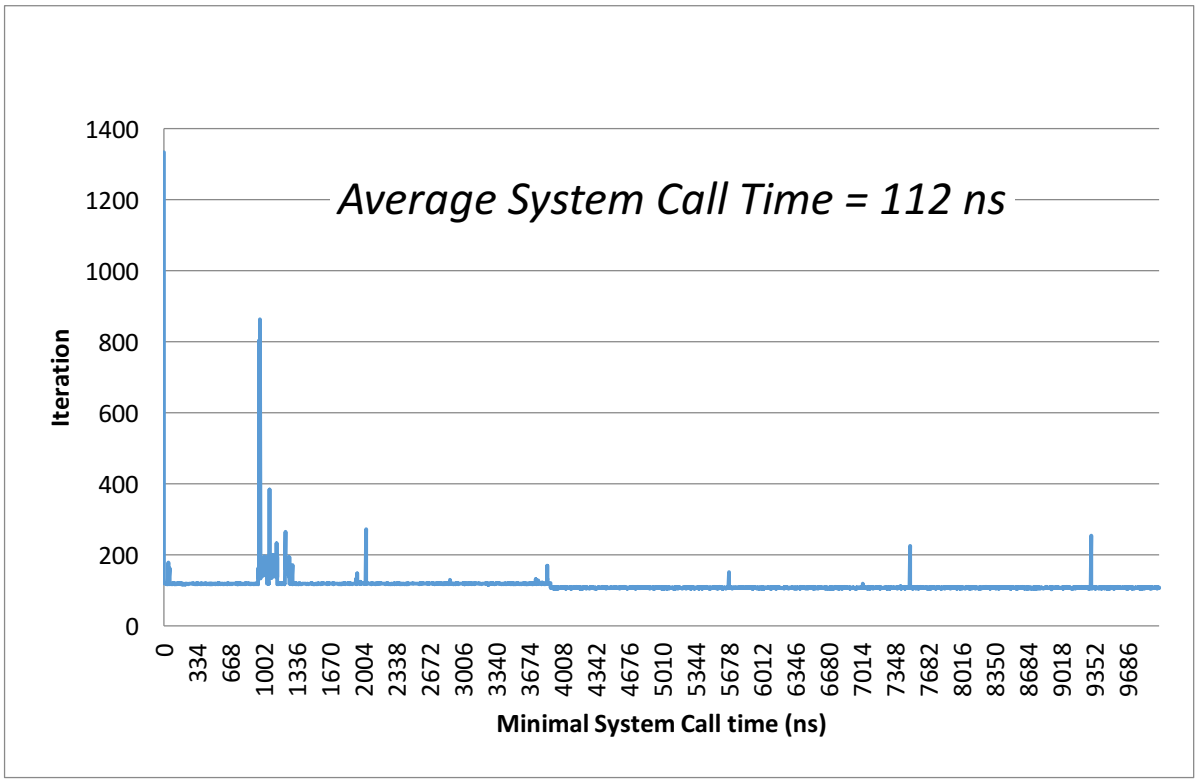

Figure 2: Minimal System Call

Cost of Process Switching

The cost of a process switch was calculated by making use of the timer code provided in hr-timer.c. The CLOCK_PROCESS_CPUTIME was used since it measures the time taken by that particular process. The measurement methodology to calculate the cost of a process switch was to create two pipes, one for the parent process and another for the child process. The two processes then communicated through the pipe by receiving and sending a single-byte message. In addition, in order to ensure the occurrence of process switching in a multi-core computer, the CPU affinity of the process was changed and restricted to a single core.
The start time was recorded when the parent process sends a message to the child process. Similarly, the end time was recorded when the parent process received a message back from the child process. The time taken by the process switch was then found by calculating the difference between the start and end times.

This procedure was repeated 10,000 times in order to get a stable value. As seen in Figure 3, the time taken was higher in the first run, and then stayed constant for the subsequent run.
The average cost of a process switch was found to be ≈ 1283 ns

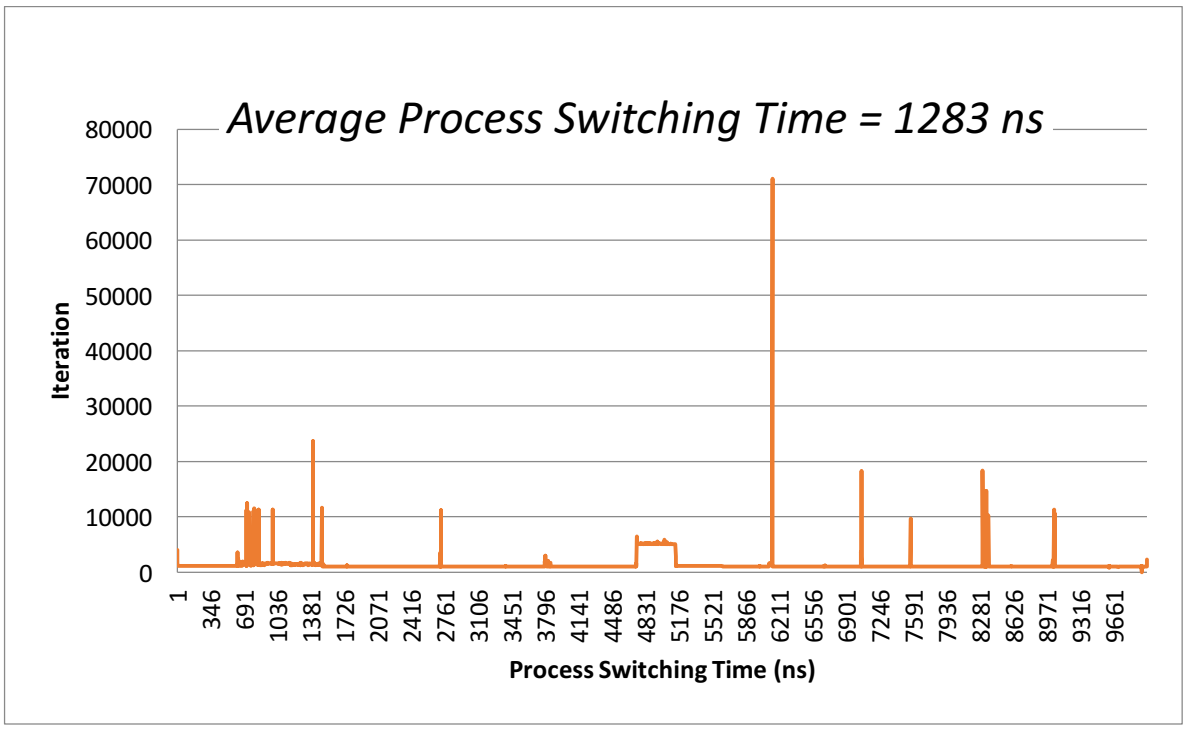

Figure 3: Process Switching Cost

Cost of Thread Switching

The cost of a thread switch was calculated by making use of the timer code provided in hr-timer.c. The CLOCK_PROCESS_CPUTIME was used since it measures the time taken by that particular process. The measurement methodology to calculate the cost of a thread switch was to create two threads, and make use of mutually exclusive locks to ensure that only one thread would make progress at a time. This was done by using a shared variable and making use of mutex locks. When the lock was released by one thread, the second would change the value of the shared variable to 0 if was changed to 1 by the first thread. In addition, in order to ensure the occurrence of thread switching in a multi-core computer, the CPU affinity of the main process was changed and restricted to a single core.

The start time required for time calculation was recorded at the time when the first thread releases the mutex lock. Similarly, the end time was recorded when the second thread acquires the mutex lock. The time taken by the thread switch was then found by calculating the difference between the start and end times.

This procedure was repeated 10,000 times in order to get a stable value. As seen in Figure 4, the time taken was high in the first run, and then stayed constant for the subsequent run. This could be attributed to the caching for the later runs.

The average cost of a thread switch was found to be ≈ 3323 ns

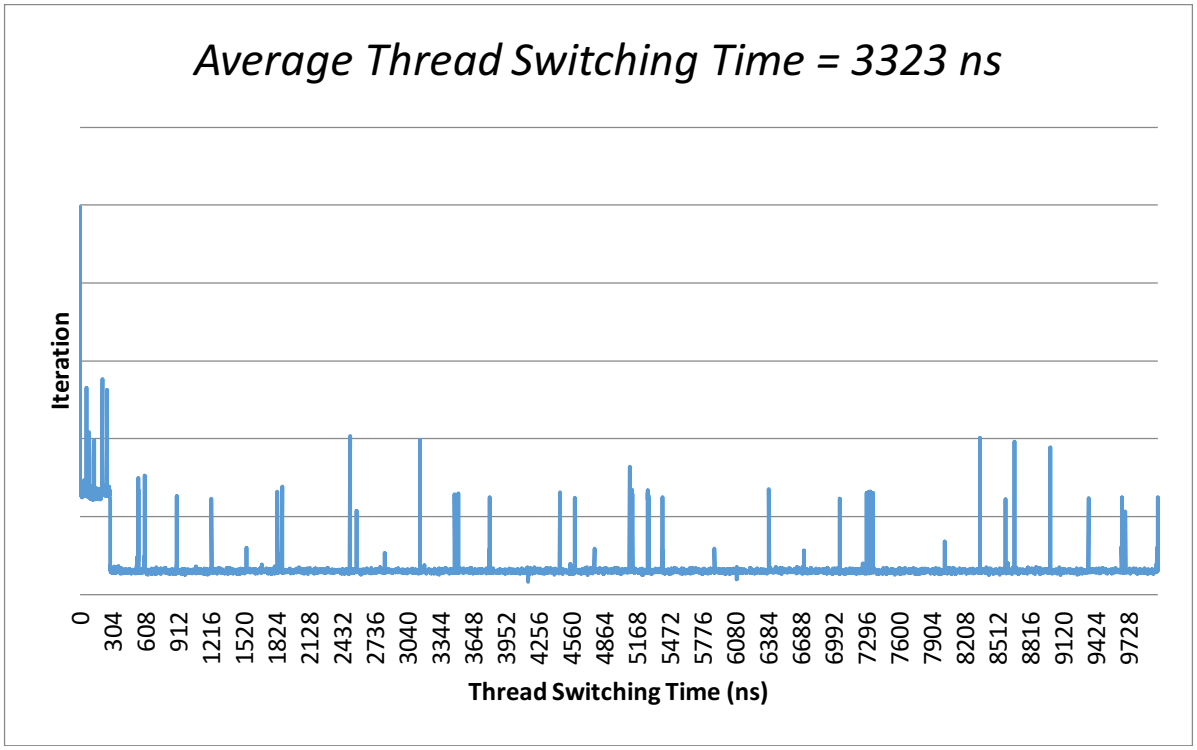

Figure 4: Thread Switching Cost

Conclusion

By calculating the costs of various operations, it was observed that the initial cost of the function was always the highest cost. This fact was true for all four kinds of operations performed. This is attributed to the fact that at the time of the first run, the data has to be fetched from the main memory, which takes a long time. In the subsequent runs however, the process already has easy access to the data since it resides in the processor cache from the previous run. Since the process and thread switching tasks were performed on a single core, the cache remained the same for all runs of these operations. In addition, function overhead for thread switching and loop overhead might also have significantly added to the costs.