Arrays and performance in C

Ying Pei Lin

Fall 2024

Random access

Before measuring the time required to access random elements in an array, we need determine the accuracy of the clock. This can be completed by calling the `clock_gettime()` function two times and calculate the time difference between the two calls. The following code snippet is used to test the accuracy of the clock and the result is shown in Figure 1.

```c
void test_clock_accuracy() {
    for(int i = 0; i < 1000; i++) {
        clock_gettime(CLOCK_MONOTONIC, &t_start);
        clock_gettime(CLOCK_MONOTONIC, &t_stop);
        long wall = nano_seconds(&t_start, &t_stop);
        printf("%ld ns\n", wall);
    }
}
```

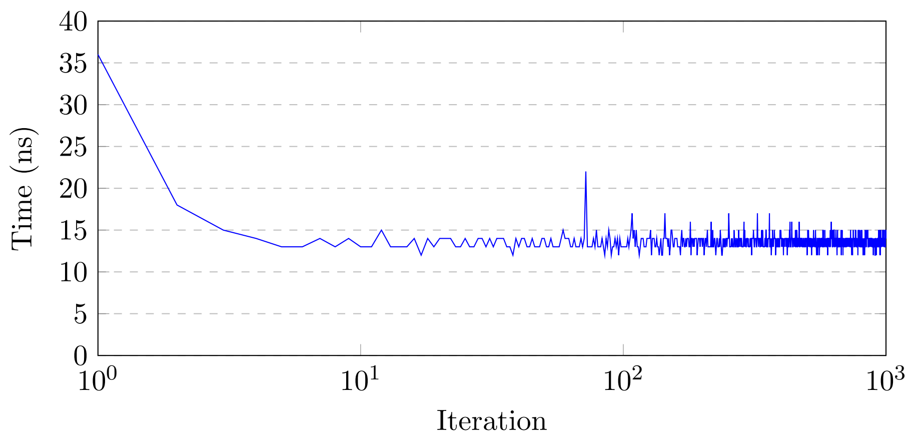

Figure 1: Time difference between two calls of clock_gettime()

The time difference between two calls of `clock_gettime()` is about 14 ns. The difference includes the time required to execute the system call, read the hardware clock and return the result. Additionally, the CPU will need to load the function into memory. If the function is not inside the cache (cache miss), the CPU will need to fetch the function from the main memory, which is slower, and this is why the first few iterations are slower than the rest.

Besides, when the CPU is executing multiple threads, it has to manage context switching between processes, distribute its computational resources across all active tasks, and handle other system overheads, which can cause delay. It will take longer to execute the function. Therefore, the workload of the CPU can also affect the time it takes to execute the function.

The following plot is the comparison between using only the terminal to run the program and running the program while web browsing and IDE is open. To avoid the influence from the CPU workload and ensure the accuracy of the time measurement, we should run the program in a quiet enviroment with minimal background processes.

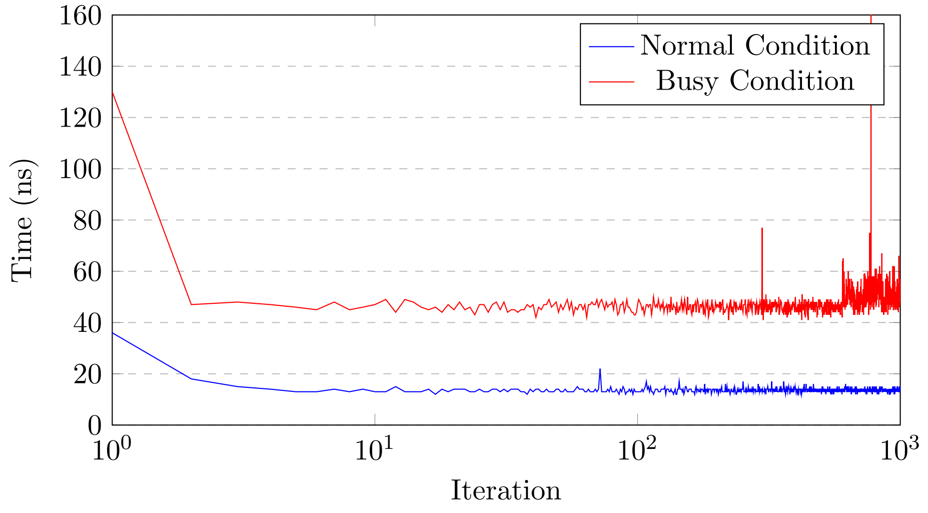

Figure 2: Comparison between running the program with and without CPU workload

Search

Duplicates