IJCSN International Journal of Computer Science and Network, Volume 3, Issue 4, August 2014
ISSN (Online) : 2277-5420 www.IJCSN.org
Impact Factor: 0.274

Performance Evaluation for Threads Execution in Single and Multi Core Processors

[1] Vilma Tomço, [2] Igli Tafa, [3] Elona Pashaj

[1] Mathematical and Statistics Department, Tirana University

[2] Computer Science, Faculty of Information and Technology, Polytechnic University Tirana

[3] Computer Science, Faculty of Information and Technology, Polytechnic University Tirana

Abstract - It was the year 2000 when firstly got introduced a new technology that would format the entire thinking and working with computers. This new technology was called multi core and was firstly invented by Intel. From that year on cooperating with a computer was easier and faster. Our paper is intended to describe the benefits of multi core processors based on an experiment comparing this technology with the one before.

Keywords - *Linux, C language, threads, parallelism, multi-core, single-core, processor.*

1. Introduction

As far as we remember the first math problems our teachers showed us were about two workers doing a job versus one worker doing the same job. We would calculate the time needed by the two workers to finish the job and then calculate the time needed by one worker to finish it. The results were clear. This job was finished for almost half time by two workers working together. We wanted to bring this example to demonstrate that the roots of every evolution are this small problems we face every day. Multi core technology is somehow an incarnation of the example we brought above. [4][8]From the first AMD processors developed by the 2000s until the last technologies we are facing today including i5 and i7, we could say Intel is demonstrating to be a "limit breaker" in this area. The greater this technology becomes the easier is going to be for everybody to solve problems in a short time. We think that every experiment is very important for successor developmentsin this area. So we are offering in this paper our view on multi core processors. We are having an experiment with threads because they are the smallest units programed to work independently. Our experiment will be done on a mathematical base. So to have it short, we are working on Linux. A program in C language using threads is executed in both kind of states, multi core and single core. By the term multi core we mean dual core. Our environment uses an Intel core duo processor. We used above the term mathematical. That's because we are comparing these two kind of states by making divisions and calculating reports on the delay time. Let's see what happens.

2. Related Works

Our first attempt to find works done on the same field was a theoretical article[8]. This article presents multi core technologies and its benefits. [9]is another article on the field. It focuses not only on the benefits but also on the disadvantages of multi-core technology. It also brings down some statistics on the delay time optimization but without showing any experiment. [5] is another article I really liked. It analyses the scalability of 7 different applications scalability working on Linux. We could say this article is way far from our league. [3] Has done an article on Linux multi-core scalability. He speaks of parallelism as the base of scalability and shows some experiments results working with multiple media files encoders. Another article on the field is[2]. It speaks about multi core technologies demonstrating with schematic views what multi-core means. It shows also codes in C language about synchronizing threads in a multi core processor. [6] shows us a development article on Multi-Threaded Programming with Posix Threads. We are focusing on this article to create our personal C code with threads. Multithread Programming guide is an article [7] that we are also referring because of its codes in C language about multithread programming.

3. Theory of Experiment

Some theoretical concepts of the experimental phase that we will use are listed below in the next section.

3.1 Definition of Single versus Multi-Core

A single core processor is a calculation unit inside a computer with only one processing unit. Instead a multi

core processor is a computing component with more than one processing units.[1]Below are shown sketches that demonstrate this difference[2].As we notice dual core processor accepts two different threads at a given moment. Single core can't do that.

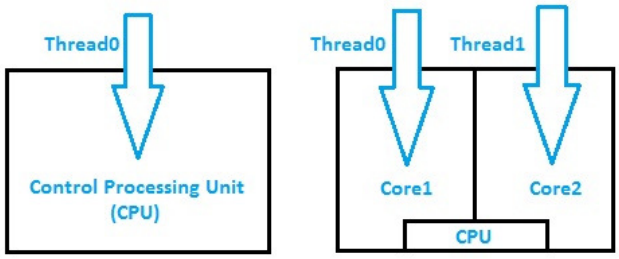

Fig. 1. Single Core CPU versus Dual Core CPU

3.2 Parallelism Basics of Multi-Core

When we say multi-core what we really mean is two or more parallel events happening at the same time[3]. The computer will not get faster by just executing a single operation. But it will improve when executing these kind of single operations in parallel always in different cores. Because our experiment will be done in a computer that doesn't support hyper-threading. An annotation has to be done here. We explain it below.

3.3 Multi-Core versus Multi-Thread

Multi core means two or more units in one CPU. Multi or hyper-thread means that each of this units can handle two or more different smaller units[3]. We will not enter in details of hyper-threading because our experiment doesn't cover this field.

3.4 Environments of Experiment

We are setting our experiment in Linux Operating Systems. We could have chosen Windows but we preferred not. That is because Windows is not familiar with GCC compiler. This compiler is projected to work on Linux Ubuntu, version 12.04. Kernel model we are using is Kernel 3.5.The programming language we are working with for the experiment is C language. We have chosen it for the simple reason it's our most familiar language.

3.5 Experimental Phase

The next section shows the experiment we have done and the calculations on the results

4. Algorithm

Below are shown two pseudo-codes for the performance in both cases: single-core and multi-core.

Single-core algorithm pseudo-code

1. Thread0 enters a loop
2. Thread0 gets out of loop
3. Thread0 enters another loop
4. Thread0 gets out of loop[6][7]

Dual-Core algorithm pseudo-code

1. Thread0 enters a loop
1.1 Thread1 enters a loop
2 Thread0 gets out the loop
2.1 Thread1 gets out the loop[6][7]

5. Algorithm Schematic Blocks

a)The first section

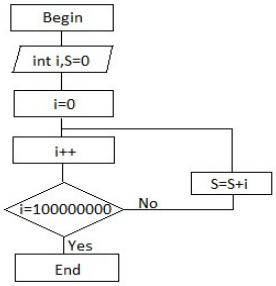

Fig 2. First Section

b)The second section

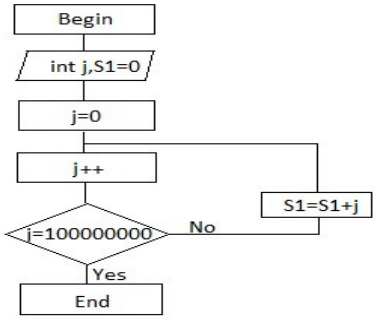

Fig. 3. Second Section

6. Test Environment

As we said before in this paper we have done the experiment in Linux Operating System. The code needed for the test is listed in the appendix by the end of this paper. For the experiment to take place we firstly had to download GCC compiler[9].It is a very important tool because it contains OpenMP library that works with threads.

Execution steps

1. Open terminal in Linux
2. Type gcc –fopenmp <program.c> -o program
3. Type ./program

4. Type echo 0 | sudo tee/ sys / devices / system/cpu/cpu1/online(this command disables one CPU)
5. Execute using ./program
6. Type echo 1 | sudo tee/ sys / devices / system/cpu/cpu1/online (this command enables the CPU firstly disabled)
7. Redo the operations listed above to get as many different tests as wished[10]

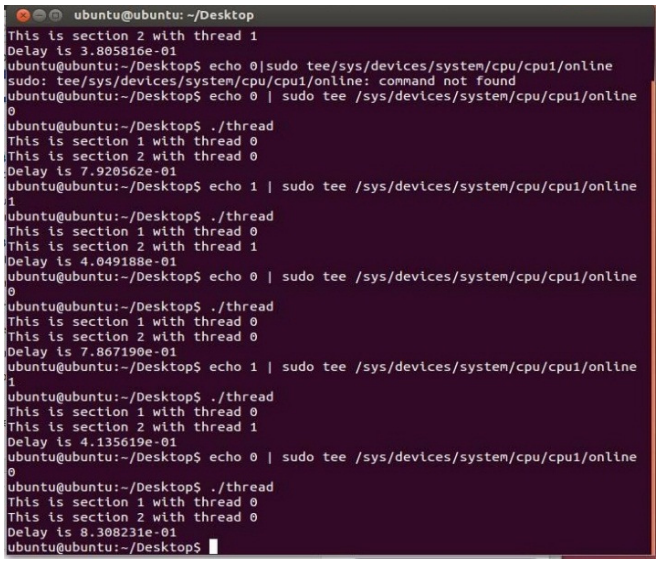

Fig. 4. Tests in terminal

7. Analytical Review

From the tests done at the paragraph 4.3 we got the results:

Table1. Calculating delay improvements

Test nr.	Single core	Dual core	Delay improvement
1	7.564532e	3.805816e	50.3%
2	7.920562e	4.049188e	51.1%
3	7.867190e	4.135619e	52.6%
4	8.308231e	4.234557e	50.9%
5	8.176235e	4.185732e	51.2%

We have calculated the delay improvements based on the formula:

- $\text{Improvement}=\frac{\textit{Dual core delay}}{\textit{Single core delay}}X100$ (1)

Observations: We see from the last column that percentages are not 50% but near this value. We thought before doing the experiment that this value would be exactly 50% because we are working on one and two processors only.

8. Conclusions

1.Dual core processors improve the work of the computer in a very distinguished way
2.The result of nearly 50% in the delay improvement demonstrates that dual-core technology saves nearly 50% time on calculations.
3. Why this value is not exactly 50% but a close value is a matter of thread overhead.

9. Future Works

Seizing the opportunity given by this article we have serious intentions to develop this paper longer on. We think to expand its idea not only in dual core technologies but also technologies that includes much more cores.

References

[1] Margaret Rouse Definition multi core processor.
[2] Multi Core Architectures by Jerney Barbic
[3] Linux multi-core scalability by Andi Kleen
[4] Multi-core processors-A necessity by Bryan Schauer
[5] An analysis of Linux scalability to many cores by Silas Boyd Wickizer
[6] Multi-Threaded Programming with Posix by Guy Kerens
[7] Multithread Programming guide is an article by Sun Microsystems
[8] Multi-core technologies by Ron Wilson
[9] Understanding and using multi-core processors by Mike Anderson
[10] Index of the Bash command Line for Linux by Linux Command Directory from O'Reilly