CS345

Whitepaper: Threads Guideline

John Decker

Introduction

When performing the tests for the systems below I was able to look into how each one behaved with the number of threads presented. What I found out was the fact that out of each system that a better time either correlated to the amount of threads presented or the speed of the clock. If you improved in one or the other the speed would increase. When it came to using threads that a pattern came about after at least 20 threads it would drop to a constant rate that inevitably stayed at that rate to the end. The actual change came about in certain way which are described below in both System #1 and System #2. As you will see there is a difference in speed but over all there is a pattern in how both CPU intensive and IO intensive programs are handled with threads.

System #1

Name of the Processor and Type:

Name of Computer: The old Jordan server (jordan2008) at 157.201.194.253

Type of Computer: Intel(R) Xeon(R) CPU E5345 @ 2.33GHz

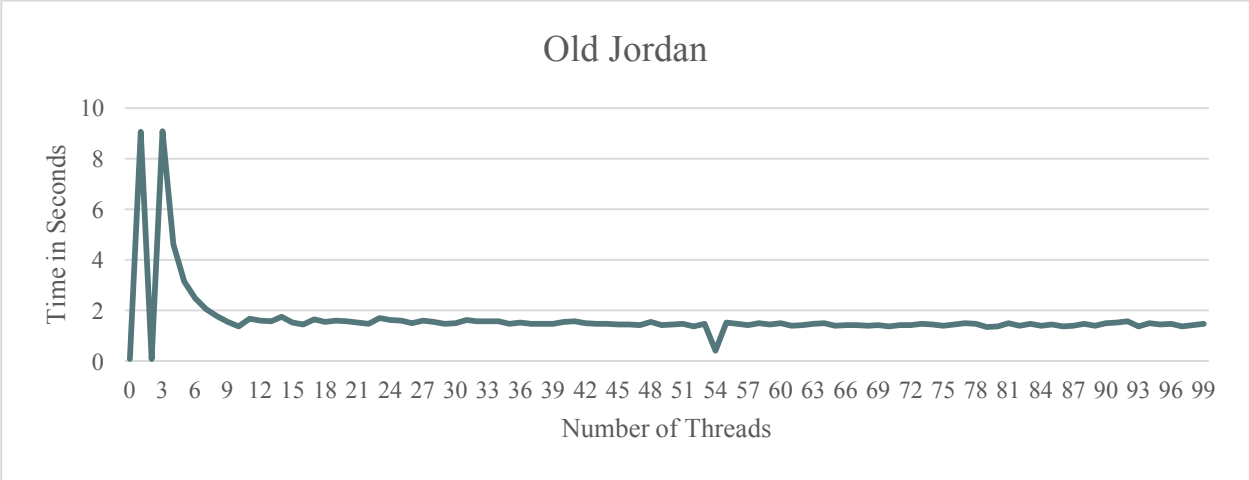
The number of processors a system has:

The system has 8 processors, non-hyper-threaded

The type of processing done by the program:

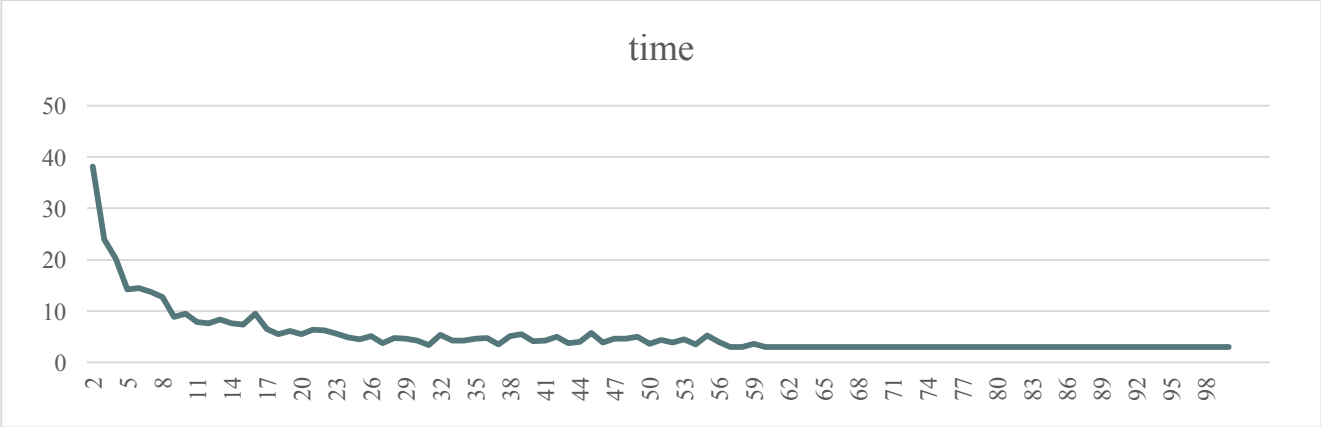
CPU Intensive: matrixMult.c

System	Type of Workload	Size of Workload	Number of Threads
Old Jordan	CPU Intensive	Matrix of 1000	0 - 99



IO Intensive: doIOjobs.cpp

System	Type of Workload	Size of Workload	Number of Threads
157.201.194.202	IO Intensive	standard	2 - 99



Questions about the following: For each system:

What should be considered (what factors should be taken into account) when determining the number of threads that a CPU intensive program should use?

What should be considered when running threads on a computer is the notion of how long and how many threads you are going to use. As an example is System #1, It is a 8 core processor and a slow clock speed. You can see that the speed holds to a constant after the 3rd thread and holds there for a while until it gets to the 54th thread where it drops down to a time of 0.394s This is evidence that the 54th thread is in some means a multiple or a off shoot of the number of iterations performed on the data.

What should be considered when determine the number of threads that an IO intensive program should use?

This on the other hand is a matter that holds for most of the machines that were tested. It would seem that any number after 62 threads it begins to even itself out up. Therefore, it would seem that in order to find the best efficiency for a IO intensive program it would be a manner that anything above half of the amount of iterations perform will lead to an efficient time.

System #2

Name of the Processor and Type:

Name of Computer: 157.201.194.237

Type of Computer: Intel® Pentium® 4 CPU, 3.4 GHz

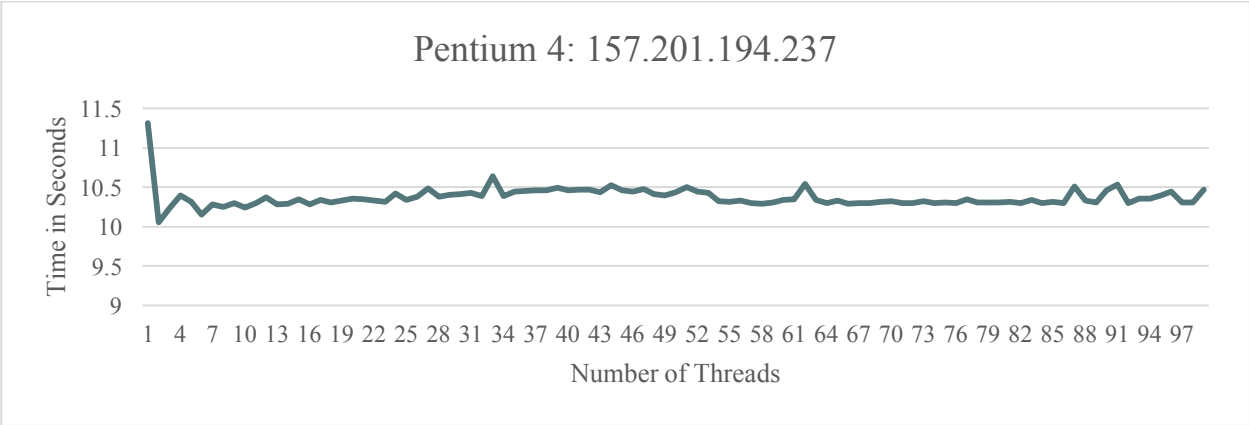
Needs the number of processors a system has:

The system has 1 processor, hyper-threaded

The type of processing done by the program:

CPU Intensive: matrixMult.c

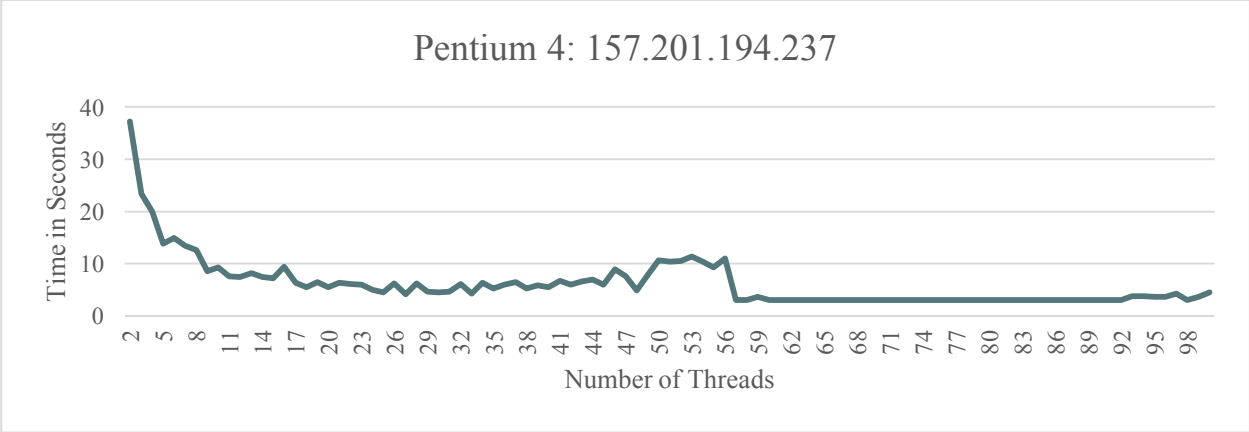
System	Type of Workload	Size of Workload	Number of Threads
157.201.194.237	CPU Intensive	Matrix of 1000	1 - 99



IO Intensive: doIOjobs.cpp

System	Type of Workload	Size of Workload	Number of Threads

157.201.194.237	IO Intensive	standard	2 - 100



Questions about the following: For each system:

What should be considered (what factors should be taken into account) when determining the number of threads that a CPU intensive program should use?

What should be considered when running threads on a computer is the notion of how long and how many threads you are going to use. As an example is System #2, It is a Pentium 4 processor, on processor and is multithreaded with a high clock speed. You can see that the speed holds to a constant after the 3rd thread and holds there until the end of the iterations in this case it would seem even with a multithreaded program that the number of processors can make the program more efficient than that of a single processor with hyper-threads.

What should be considered when determine the number of threads that an IO intensive program should use?

This on the other hand is a matter that holds for most of the machines that were tested. It would seem that any number after 54 threads that it evens itself out. Therefore, it would seem that in order to find the best efficiency for a IO intensive program it would be a manner that anything above half of the amount of iterations perform will lead to an efficient time. Which would indicate that in any system the number of iterations (which equates to the amount of iterations needed to perform on the data) divided by 2 will be the best selection to a IO intensive program.

Share some basic guidelines about how many threads to use in a multithreaded program. Some of the basic needs for a multi threaded program and the performance desired fall around two different points with the program you are creating:

1. **The correct number of threads**
   The amount of efficiency that you desire in time requires the correct number of threads to present. This could be a thread for every number of iterations that is wished to be performed or in the case of a few systems, their age it could mean finding a multiple of the iterations to be desirable.

2. **If you are using a legacy system (i.e. single core systems, low CPU speed)**
   This can be an issue as the speed that is desired is dependent on the speed of the clock and no matter how hyper-threaded you have your system you still get to a point in which the amount of threads and overhead to create threads detracts from the speed of running the iteration.