Computer Benchmarking

By

Neal Friedman

In this modern age, almost every American seems to own at least one computer; whether that be a personal computer, tablet, phone, or calculator. Computers have become a necessary part of all our lives. Part of the reason computers have become so widespread is how efficient they are at processing information. All of this comes down to the mighty central processing unit, or CPU. The CPU is the brain of the computer. Now if you have a computer, you are most likely running some sort of program on this computer. Programs are compilation of a series of instructions written in a special kind of coding language designed for computers. The CPU can run these instructions at an amazing rate. But how fast are we talking really?

There are many ways to measure the performance of a computer. For the basic user that may just be how long your computer takes to open Adobe Acrobat Reader (which at times can seem ungodly long). For a more frequent user that could mean how long it takes for their game to load. For the programmer, this means how long a program may take to run. Lastly, for supercomputing this could mean how long it takes to perform a series of difficult calculations.

One of these kinds of difficult calculations is done using floating point numbers. In computing, floating point numbers are an approximation of a real number in a way that can support a wide range of values. It is the computers way of using scientific notation. Numbers in scientific notation look like this 5.9x10^12 which would mean 5.9 trillion. Scientific notation is used for very large or very small numbers. Performing calculations with these numbers on a computer takes a lot of work. Thus the method of measuring computer performance called FLOPS was born.

FLOPS or FLoating-point Operations per Second is a measure of computer performance used in scientific calculations that make heavy use of floating-point calculations. In these cases, measuring the FLOPS is more accurate than the generic measure of instructions per second.

The program I have written for you measures the performance of a series of additions using floating point numbers. This program takes a table with size of your choosing. You may pick how many rows by how many columns you would like to have. When it is run it will fill this table with floating point numbers. Then it will add up all the numbers in the table once by going down by rows and adding and then once by columns. It will calculate the time in nanoseconds it took to perform the addition by rows and then the time it took to add by columns. After the additions are complete, the program will compare the times and display which form of addition was faster.

From my analysis, it seems that adding by rows is almost always faster. It becomes very apparent when handling very large tables (for example, 10000 by 10000 sized tables). I believe the reasoning for this is due to how cache memory works and how Java handles two dimensional arrays. When java pulls an element from a row it pulls the elements near it as well. This will cause multiple cache hits before a cache miss. The cache miss will then cause the program to pull another chunk of elements from the row. When adding by columns, the program is still grabbing chunks of elements from the row. This causes many more cache misses, thus making adding by columns a slower process. On my system, the difference is about 13 seconds in a 10000 by 10000 size table. In conclusion, adding by rows is faster due to the mechanisms behind cache memory. When adding by rows it is much faster because there will be many cache hits before a cache miss.