APLNextSupervisor – Processing Gains

August 6, 2014

What is Multithreading?

A computer's central processing unit (CPU) can only execute one instruction at a time. As processors become faster, each instruction takes less time. To speed up a process, either the CPU needs to make calculations faster, or the process needs to contain less calculations (simplify code).

However, when running a process that must repeat the same calculation repeatedly, multithreading can also be used to increase speed. Most modern CPUs have multiple "cores" and sometimes multiple "threads" per core. If a computer has four threads, the computer can pass instructions to each thread independently to make calculations. Therefore, if the same calculation must be repeated one hundred times, each thread only needs to complete 25.

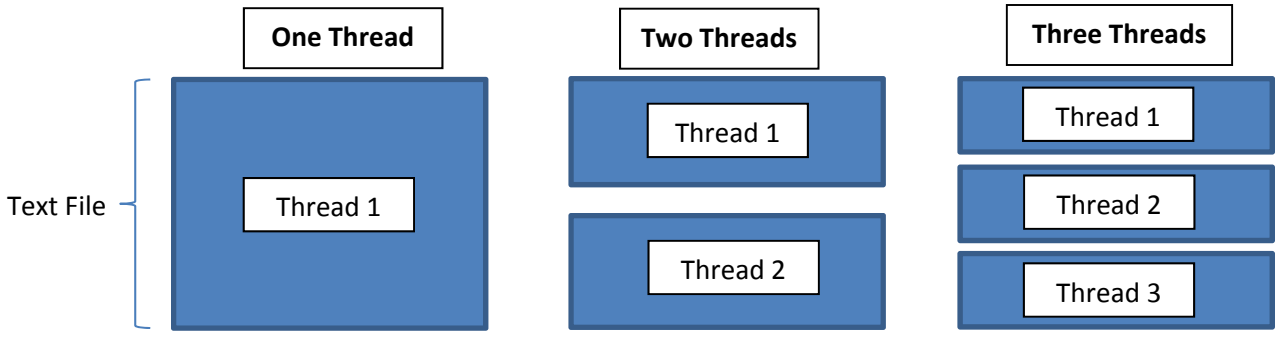
This results in less processing time since the CPU effectively processes four calculations simultaneously.

Example Where Multithreading Speeds Processing

When an insurance company changes their prices, they are interested in the impact of the change on each of their individual policyholders. Some may have an increase while others may have a decrease. To quantify the price change to each of these policyholders, the insurance company may use a computer program to re-rate each policy at the prior rate and again at the new rate. This involves a loop with many calculations through the application that has the potential to execute hundreds of thousands of times, leading to long processing times.

Let the input to the application be a text file with one row for each policy that must be rated. The APL application has to loop through each line, read in characteristics, and then processes them to find premium amounts. If the application is run on two threads, the text file can be split in half. Each thread now only has half of the text file to go through. The processing time is effectively cut in half (a relatively small amount of extra time will be needed to create the multithreading process in the beginning and to compile the results from each thread together at the end). If more threads are available, the text file can be split further, causing each individual thread to have to process fewer rows.

Figure 1: The input text file is split according to the number of threads selected.



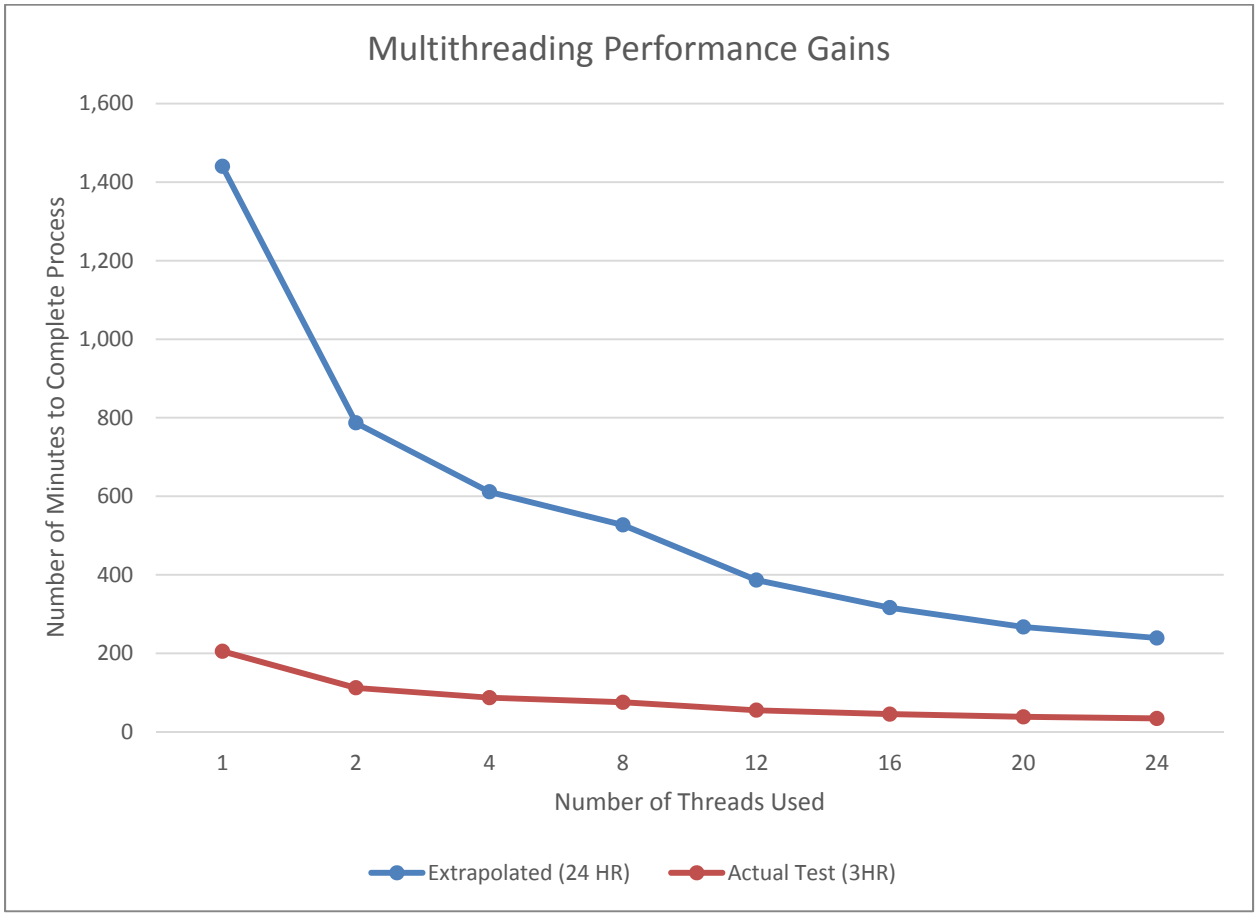
Processing Time Gains

To illustrate the realized time gains multithreading provides, an application similar to the one described above was run multiple times, each using a different number of threads. The process on one thread took 3 hours and 25 minutes to complete using one thread. The data has been extrapolated to a 24 hour process to see the gains of using multithreading on a more time consuming process or larger dataset.

Figure 2: Table showing incremental time savings from using multithreading.

	Extrapolated to 24 Hour Process						Tested 3 Hr 25 Min Process					
Threads	Run Time in Hours and Minutes	Extrapolated (24 HR)	Incremental Time Saving (Minutes)	Incremental Time Saving (Percent)	Cumulative Time Saving (Minutes)	Cumulative Time Saving (Percent)	Run Time in Hours and Minutes	Actual Test (3HR)	Incremental Time Saving (Minutes)	Incremental Time Saving (Percent)	Cumulative Time Saving (Minutes)	Cumulative Time Saving (Percent)
1	24 Hr 0 Min	1,440					3 hr 25 min	205				
2	13 Hr 7 Min	787	653	45%	653	45%	1 hr 52 min	112	93	45%	93	45%
4	10 Hr 11 Min	611	176	22%	829	58%	1 hr 27 min	87	25	22%	118	58%
8	8 Hr 47 Min	527	84	14%	913	63%	1 Hour 15 min	75	12	14%	130	63%
12	6 Hr 26 Min	386	140	27%	1054	73%	55 Min	55	20	27%	150	73%
16	5 Hr 16 Min	316	70	18%	1124	78%	45 Min	45	10	18%	160	78%
20	4 Hr 27 Min	267	49	16%	1173	81%	38 Min	38	7	16%	167	81%
24	3 Hr 59 Min	239	28	11%	1201	83%	34 Min	34	4	11%	171	83%

Figure 3: Chart showing the time savings from multithreading.



Conclusion

Multithreading in APL2000 is a useful tool that can greatly decrease the time it takes to run a program. However, to use multithreading, the instructions sent to each thread must be independent; processing on each thread must not rely on the results of another. The example program is very insurance specific, but any process in which the work can be broken up into multiple pieces has the potential to be multithreaded and benefit.