Kendall Wahnschaffe

**Whitepaper**

**Introduction**

There are many factors to take into consideration when running processes. Some factors that play a key role are:

* The number of processors a system has
* The type of processing done by the program (CPU intensive or IO intensive)

We will be performing various tests, and try to answer the two questions:

* 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 determining the number of threads that an IO intensive program should use?

**Background**

The way we will be testing these questions is by testing two separate files

* MatrixMult: A matrix multiplication program that can multiply an N by M matrix (a CPU intensive program). This program is written to run on any number of threads, as specified by the user.
* doIOjobs: This program performs a constant amount of input/output functions (IO intensive)

We will be gathering information by running these two programs on different types of systems (multiple processors, hyperthreaded systems, different cores) while manipulating both the work load and number of threads. By running sufficient tests of 1 to 228 threads, we will be able to answer the questions of this paper.

**Results**

One of the factors that was constant across all systems was that the speed up from 1 thread to 2 threads was almost always 2. However, increasing the speed up beyond that tended to quickly plateau (see Figure 1).

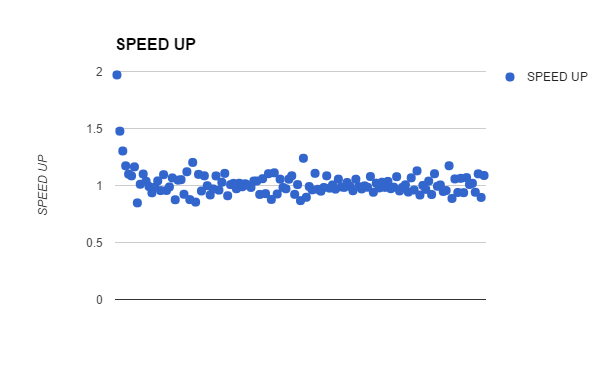


Figure 1.

**Conclusion**

The results for CPU intensive operations seemed to perform the best when you keep the workload evenly distributed between the cores. Generally, having one operation per core provided the most speedup. Having more than one thread per core did not provide a significant improvement.

The IO intensive programs, however, did have a significant improvement when more threads were added (see Figure 2). Although it also plateaued, the rate at which it did so was significantly slower than that of CPU intensive programs. The improvement also lasted much longer.