## http://summer.bghcdn.ogqcorp.com/image/1615/1280There Is No Right Answer.

The debate about how many threads to use is alive and well among the forums of the internet, though there seems to be some convergence around the idea that it is better to measure than to theorize. This is due to the fact that there is no general, one size fits all answer. In harmony with this notion, this paper sets out to answer the following questions:

Figure These are not the threads you’re looking for

1. For a matrix multiplication application (say that 5 times fast), how many threads should be used for n processors where n ?
2. For an I/O bound application, how many threads should be used for n processors where n ?

## Matrix Multiplication

This data was gathered by averaging the result of three runs with the number of threads in the x-axis. There is a clear taper after the threads > n. For this program, running no more than n threads are recommended. Since these threads are CPU bound rather than I/O bound, we expect that one thread per core is going to give us the best performance.

## Simulated I/O

This data was gathered in a similar manner to the Matrix Multiplication threads. Here the results begin to taper off around 11 threads, regardless of the number of cores in the system. We expect this to be the case as the threads are I/O bound, and thus the CPU can switch contexts and run additional threads while we are waiting for the previous threads to complete.

## Conclusions

For primarily CPU bound applications, use roughly the same number of threads as there are cores in your CPU. (Double this if your cores are hyper threaded). For primarily I/O bound applications, your results will vary wildly. As a general rule, you will want more threads for this than for CPU bound applications, but the exact number depends on the program you are using. Keep in mind that there is no right theoretical answer. The correct answer is always to measure your results on your production machine and adjust accordingly.