

Prepare 13 : White Paper

James Olney



December 6, 2017

CS 345 Operating Systems

James Olney

CS345

12 White Paper

Introduction/executive summary

This is an analysis and study of how the number of threads can affect performance. We tested this on Linux with tests for CPU, and I/O systems.

Body

Below is the data gathered from the testing on the linux server.

CPU

In CPU, the performance in real-time was pretty much the same, with the number of threads not changing much even when adding 1000 threads! The user and sys didn’t change more than 28% until the use of 1000x1000 matrices, but didn’t seem affected by the number of threads. When the number of threads matched the matrices, it would max out at 7s in real time, up to 100 threads. But this might have varied with more testing since 1 thread for 100x100 matrices had 6 seconds versus the 7 seconds for 100 threads for 100x100(see below).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| cpu |  |  |  |  |  |
|  | matrices | threads | real | user | sys |
| 2 | 4x4 | 1 | 0m0.002s | 0m0.000s | 0m0.002s |
| 2 | 4x4 | 4 | 0m0.007s | 0m0.000s | 0m0.005s |
| 2 | 4x4 | 10 | 0m0.002s | 0m0.000s | 0m0.002s |
| 2 | 4x4 | 25 | 0m0.005s | 0m0.000s | 0m0.003s |
| 2 | 5x5 | 1 | 0m0.002s | 0m0.000s | 0m0.002s |
| 2 | 5x5 | 7 | 0m0.002s | 0m0.001s | 0m0.001s |
| 2 | 5x5 | 10 | 0m0.002s | 0m0.000s | 0m0.002s |
| 2 | 5x5 | 20 | 0m0.002s | 0m0.002s | 0m0.001s |
| 2 | 10x10 | 1 | 0m0.002s | 0m0.001s | 0m0.000s |
| 2 | 10x10 | 5 | 0m0.003s | 0m0.001s | 0m0.002s |
| 2 | 10x10 | 7 | 0m0.003s | 0m0.001s | 0m0.002s |
| 2 | 10x10 | 10 | 0m0.002s | 0m0.001s | 0m0.001s |
| 2 | 10x10 | 20 | 0m0.003s | 0m0.002s | 0m0.001s |
| 2 | 100x100 | 1 | 0m0.006s | 0m0.005s | 0m0.001s |
| 2 | 100x100 | 5 | 0m0.005s | 0m0.006s | 0m0.002s |
| 2 | 100x100 | 7 | 0m0.005s | 0m0.005s | 0m0.002s |
| 2 | 100x100 | 20 | 0m0.005s | 0m0.004s | 0m0.003s |
| 2 | 100x100 | 100 | 0m0.007s | 0m0.007s | 0m0.004s |
| 2 | 1000x1000 | 1 | 0m4.594s | 0m4.580s | 0m0.004s |
| 2 | 1000x1000 | 5 | 0m2.586s | 0m5.155s | 0m0.007s |
| 2 | 1000x1000 | 7 | 0m2.588s | 0m4.877s | 0m0.007s |
| 2 | 1000x1000 | 20 | 0m2.357s | 0m4.635s | 0m0.007s |
| 2 | 1000x1000 | 100 | 0m2.321s | 0m4.481s | 0m0.035s |
| 2 | 1000x1000 | 1000 | 0m2.562s | 0m4.820s | 0m0.132s |

For I/O, the number of threads increased the performance the more that was added. It seems that 100 was a sweet spot where it improved the real-time to the point that even 1000 threads was about the same as 100 made, varying only about 1% in real time. User and sys performance again only varied about 5-15 seconds and didn’t seem to vary much when the threads were added(see below).

IO

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| i/o |  |  |  |  |  |
|  |  | Threads | real | user | sys |
|  |  | 1 | 0m53.745s | 0m0.011s | 0m0.01s |
|  |  | 2 | 0m36.274s | 0m0.009s | 0m0.013s |
|  |  | 3 | 0m22.681s | 0m0.011s | 0m0.01s |
|  |  | 4 | 0m19.265s | 0m0.007s | 0m0.014s |
|  |  | 5 | 0m11.142s | 0m0.009s | 0m0.019s |
|  |  | 6 | 0m13.803s | 0m0.01s | 0m0.011s |
|  |  | 7 | 0m13.149s | 0m0.011s | 0m0.011s |
|  |  | 8 | 0m12.301s | 0m0.008s | 0m0.013s |
|  |  | 9 | 0m8.463s | 0m0.01s | 0m0.012s |
|  |  | 10 | 0m9.098s | 0m0.013s | 0m0.012s |
|  |  | 11 | 0m7.478s | 0m0.011s | 0m0.013s |
|  |  | 12 | 0m7.326s | 0m0.01s | 0m0.011s |
|  |  | 13 | 0m8.033s | 0m0.011s | 0m0.011s |
|  |  | 14 | 0m7.319s | 0m0.01s | 0m0.011s |
|  |  | 15 | 0m7.161s | 0m0.01s | 0m0.013s |
|  |  | 50 | 0m3.516s | 0m0.008s | 0m0.019s |
|  |  | 100 | 0m2.981s | 0m0.012s | 0m0.019s |
|  |  | 500 | 0m2.982s | 0m0.009s | 0m0.032s |
|  |  | 900 | 0m2.984s | 0m0.016s | 0m0.044s |
|  |  | 1000 | 0m2.985s | 0m0.015s | 0m0.058s |

Summary/conclusions

Number of threads only made a big difference on the IO but not much on the CPU, and even then, more than 100 threads seems to be overkill. The number of threads on the CPU only made a difference when matching the matrices.

# References

Chicago, U. o. (2017). *Threads.* Retrieved from CS: https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/4\_Threads.html

Lee, E. A. (2006). *The Problem with Threads.* https://www2.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-1.pdf: Electrical Engineering and Computer Sciences University of California at Berkeley.

Olszewski, M., Ansel, J., & Amarasinghe, S. (2009). *Kendo: efficient deterministic multithreading in software.* Washington, DC: ASPLOS XIV.

Robison, A. (2007, April 6). *Why Too Many Threads Hurts Performance, and What to do About It.* Retrieved from codeguru: https://www.codeguru.com/cpp/sample\_chapter/article.php/c13533/Why-Too-Many-Threads-Hurts-Performance-and-What-to-do-About-It.htm