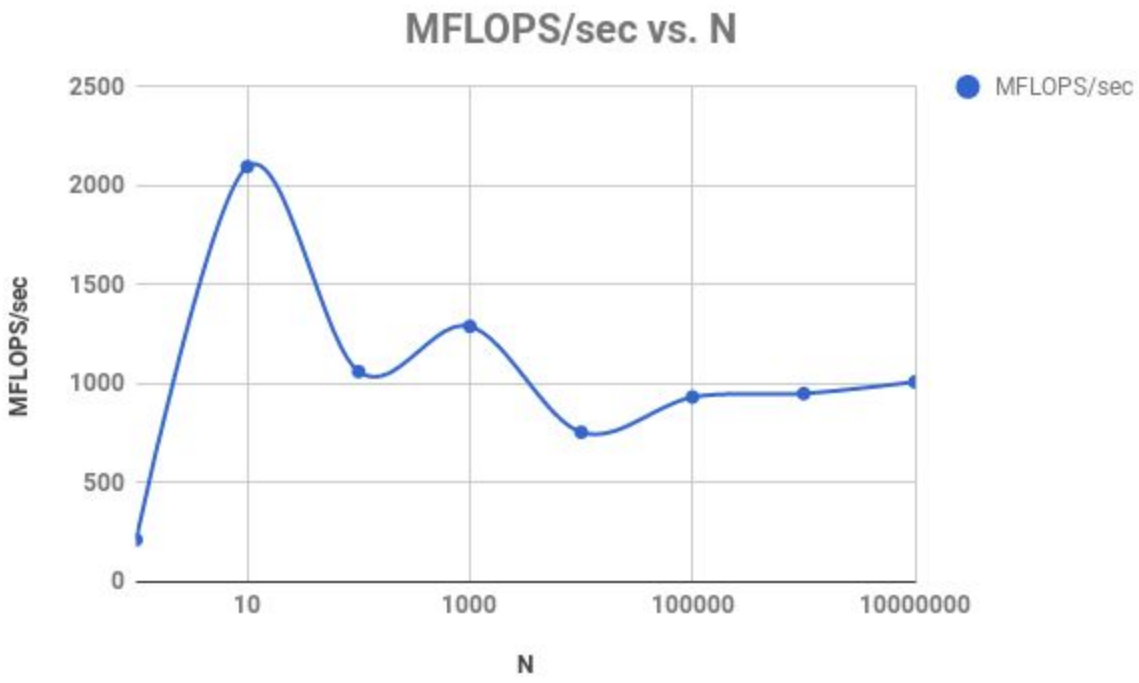
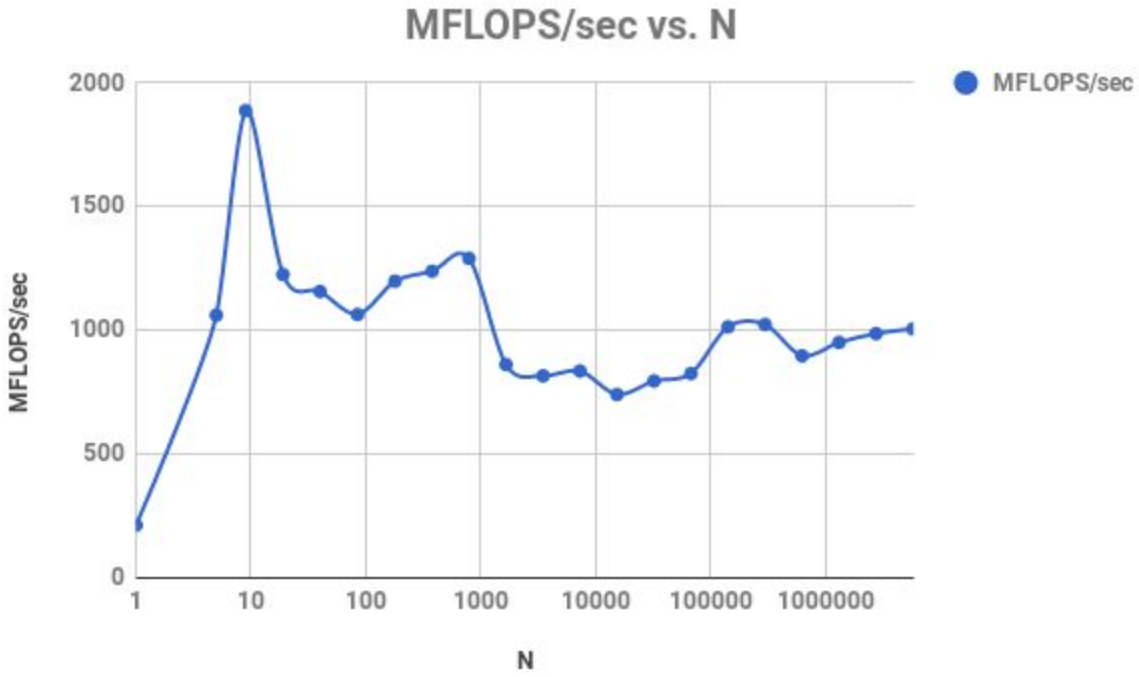
I performed testing on a machine in the Linux lab (Intel(R) Xeon(R) CPU E5-1620 v3 @ 3.50GHz, 15.5 GB RAM, 4 cores, 8 threads, Kernel Linux 2.6.32-696.18.7.el6.x86_64, std=c99 compiler) as the code was not compatible with Windows (it contained UNIX specific time functions). I performed multiple tests: Keeping R constant (R = 100) and varying N to 10,000,000, by multiplying N by 10 (as it was mentioned in class that errors may come up if other increments of N are taken) and by taking specific values of N usually taken in other vector triad tests found online (for comparison purposes), on both the linux machine and the ACI-ICS interactive cluster (aci-i). I used the -O3 flag. The -O3 flag is supposed to optimize the code and the dummy() function is included so that the compiler doesn't throw away the loop for optimization purposes. After setting the dummy() function and the -O3 flag up, I observed massive MFLOPS improvements for the same N and R.

Lab Machine:

With R = 100 :

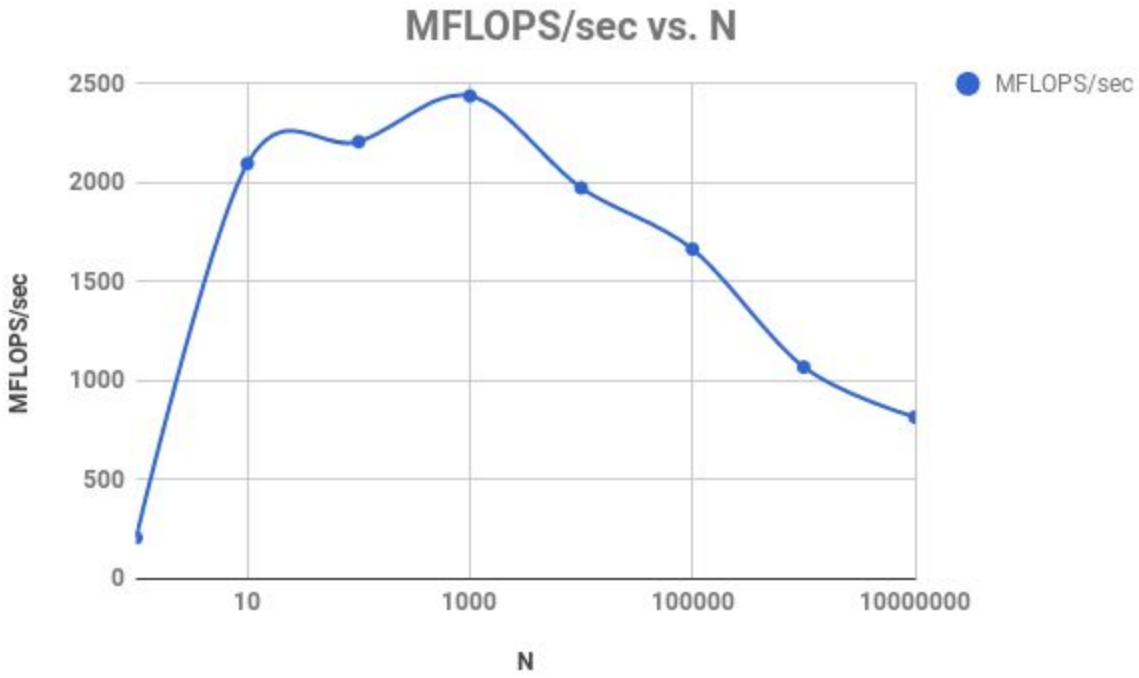


With R = 100 (more detailed):

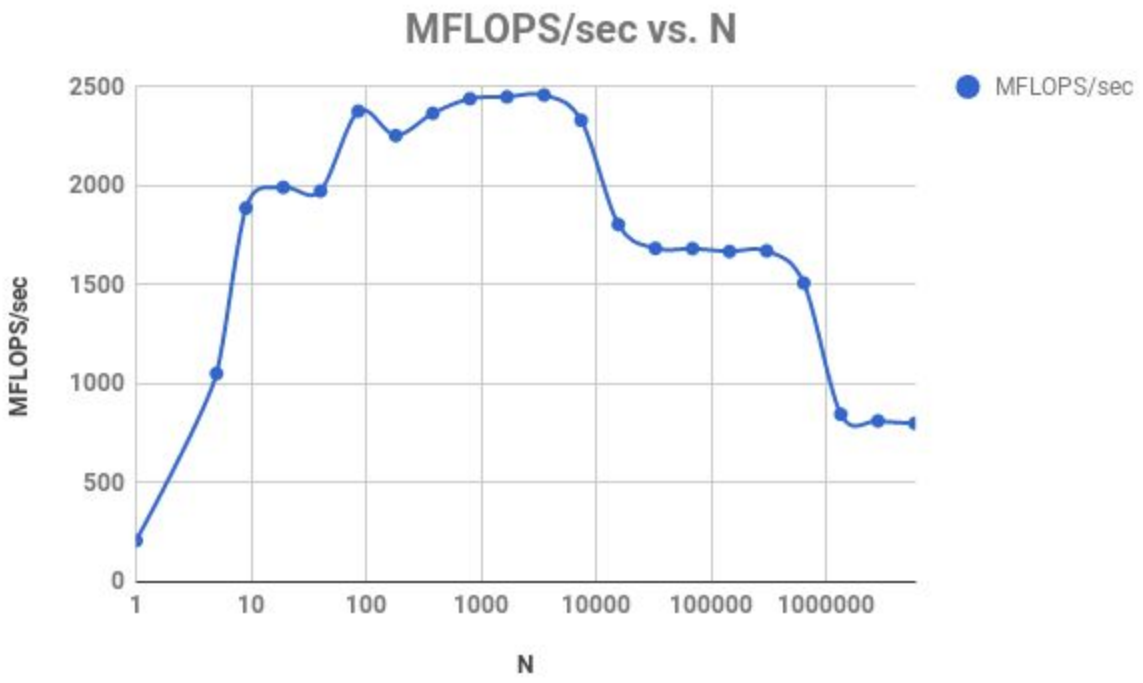


ACI Cluster:

With R = 100 :



With R = 100 (more detailed):



The constant-N graphs are not particularly interesting as the vector triad test is usually useful when varying N. R is only there so that the running time can be accurately computed (especially in the cases where N is small). The local machine curves obtained are different from those in the book. The R-constant curve was similar to other online vector triad tests performed on the Intel Xeon CPU. What should be expected is bad performance on small values of N that improves as we make N bigger until we reach an MFLOPS maximum. From there, performance should drop and stagnate on multiple levels. The reason for the bad performance for low values of N is because of system caches that are not well populated and “warmed-up”. As we increase N, the caches (L1, L2, L3) become more efficient until they reach maximum efficiency (their limit). As each cache reaches its limit (In turn L1, L2 and L3), we reach the stagnation levels. The caches become overshadowed by the extremely large value of N. In our case, our core seems to be pretty efficient at handling very small values of N and does not behave like most cores. The cluster behaves as expected, with MFLOPS increasing, reaching a maximum then decreasing and reaching “plateaux” or stagnations. It was very surprising however how MFLOPS for the linux machine was comparable to the cluster for large N values. I attempted to test for extremely large values of N (>100,000,000) by changing N from int to long. I did not get very interesting results doing that as there was no change in the observable pattern. Since the vector triad test also uses int (not long), I did not include that data.