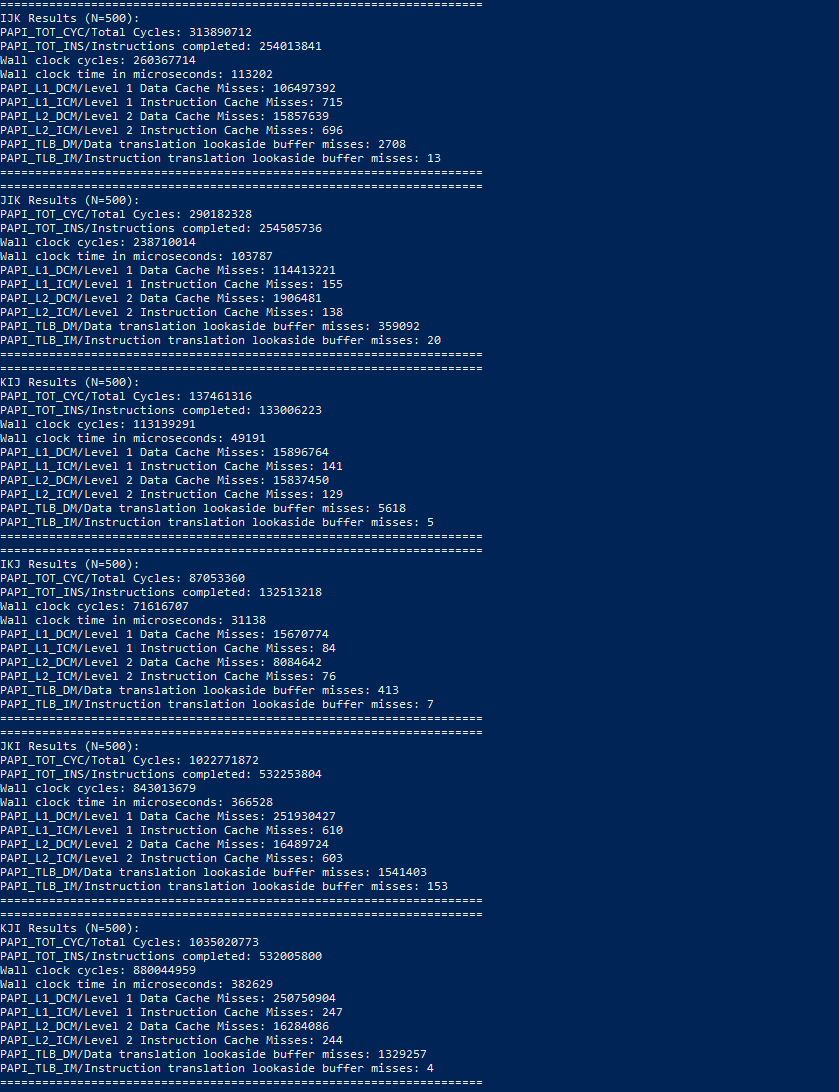
Michael Thorman

CSCI490/Fall 2021

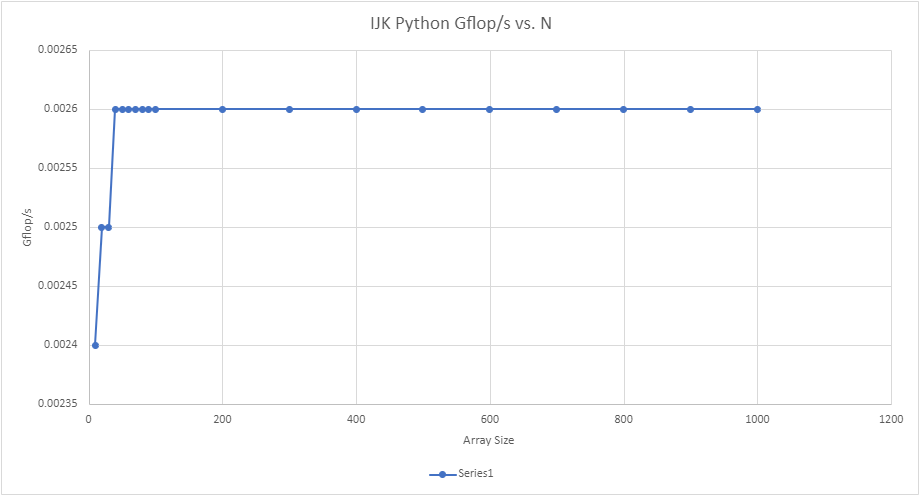
9/23/2021

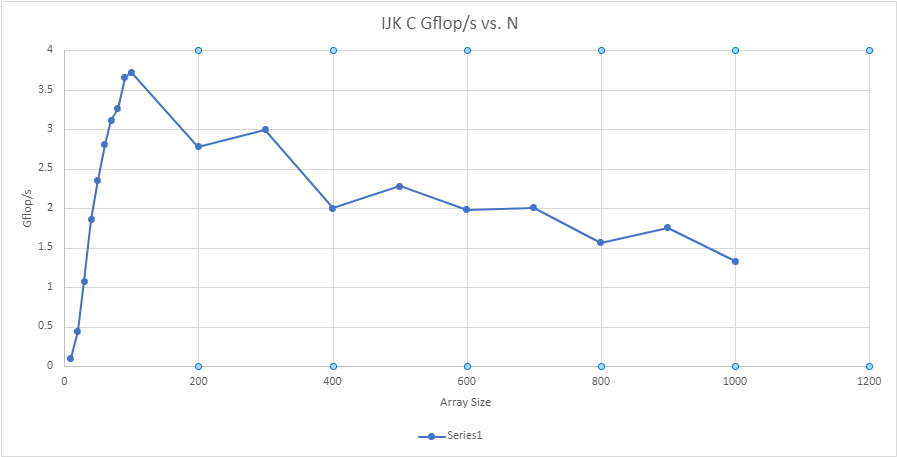
Lab 2

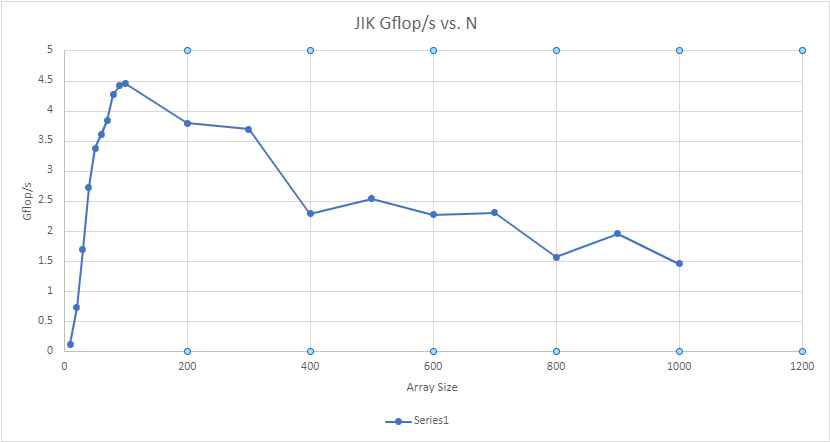
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Matrix Multiplication Algorithm | Execution Time (N=500) | Execution Time Average | Gflops = 2N^3/time | Actual Gflops/Peak Gflops (Big Red 3 peak performance = 736 Gflops) |
| Ijk (python) | 96.564760 secs  96.678939 secs  96.530001 secs  96.986018 secs  96.660906 secs | 96.6841248 secs | 2(500) ^3/96.6841248 = 2,585,739.90835773 flops = 0.0026 Gflops | 0.0026/736 = 0.00000353  X100% = 0.00035% of peak performance |
| Ijk (C) | 0.113202 secs  0.104463 secs  0.103581 secs  0.104694 secs  0.103162 secs | 0.1058204 secs | 2(500) ^3/0.1058204 = 2,362,493,432.26825829 flops = 2.3625 Gflops | 2.362/736 = 0.0036  X100% = 0.36% of peak performance |
| Jik (C) | 0.103787 secs  0.095321 secs  0.092875 secs  0.09245 secs  0.094428 secs | 0.0957722 secs | 2(500) ^3/0.0957722 =  2,610,360,835.39899887 flops = 2.6104 Gflops | 2.6104/736= 0.0035  X100% =0.35% of peak performance |
| Kij (C) | 0.049191 secs  0.046182 secs  0.046554 secs  0.046227 secs  0.046691 secs | 0.046969 secs | 2(500) ^3/0.046969 = 5,322,659,626.56220060 flops = 5.3227 Gflops | 5.3227/736 =0.007%  X100% =0.72% of peak performance |
| IKJ (C) | 0.031138 secs  0.029726 secs  0.030117 secs  0.029766 secs  0.030235 secs | 0.0301964 secs | 2(500) ^3/0.0301964 = 8,279,132,611.83452332 flops= 8.2791 Gflops | 8.2791/736=0.0112  X100%=1.12% of peak performance |
| Jki (C) | 0.366528 secs  0.322037 secs  0.324889 secs  0.321241 secs  0.33177 secs | 0.333293 secs | 2(500) ^3/0.333293=  750,090,760.98207883 flops = 0.7501 Gflops | 0.7501/736 =0.001  X100%=0.10% of peak performance |
| Kji (C) | 0.382629 secs  0.328555 secs  0.32373 secs  0.32807 secs  0.323685 secs | 0.3373338 secs | 2(500) ^3/0.3373338 = 741,105,694.12255754 flops = 0.7411 Gflops | 0.7411/736=0.001  X100% =0.10% of peak performance |

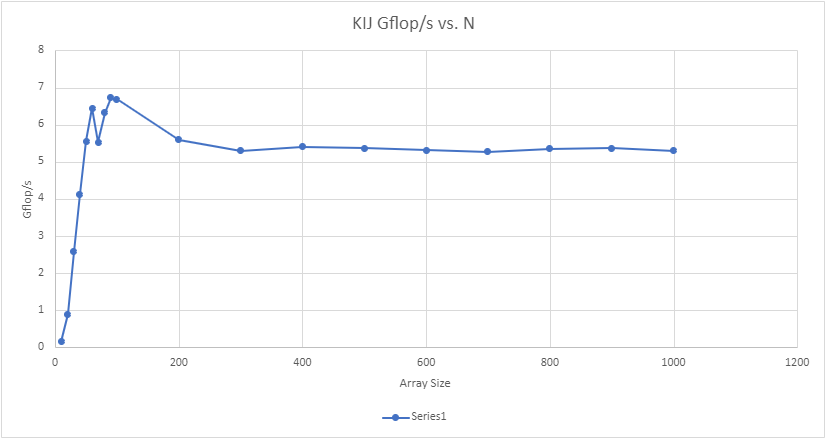


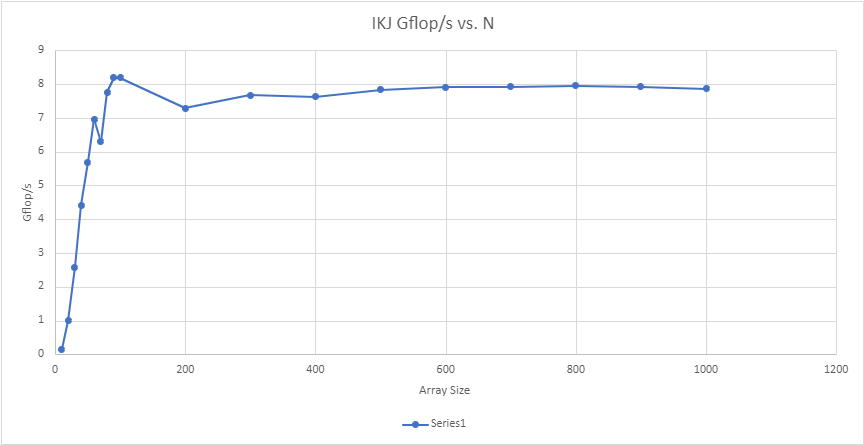


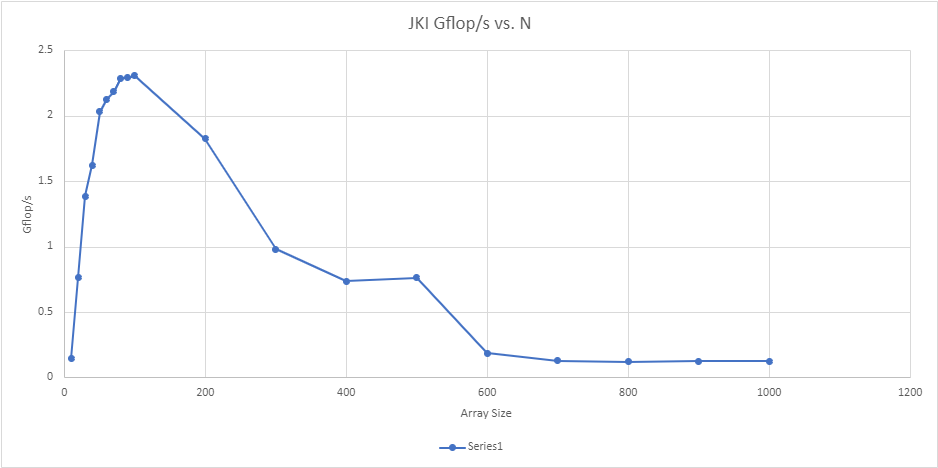


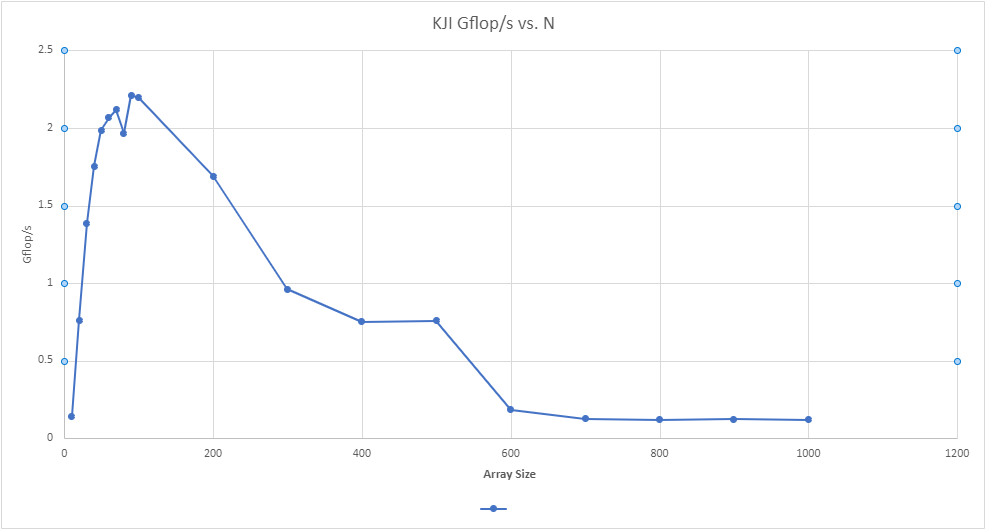












The matrix multiplication algorithm which computes C= C+A&B in the shortest period of time and has the largest most Gflop/s is IKJ using the C language. The next algorithm with the shortest computation time is KIJ with 5.3227 Gflop/s. Algorithms IJK and JIK had similar flops, 2,362,493,432.26825829 and 2,610,360,835.39899887, respectively. Moreover, algorithms JKI and KJI performed the worst using the C language, with 750,090,760.98207883 flops and 741,105,694.12255754 flops, respectively. Lastly, the IJK algorithm using Python performed the worst of all algorithms using C with flops = 2,585,739.90835773. As far as rankings are concerned, IKJ(C)>KIJ(C)>JIK(C)>IJK(C)>JKI(C)>KJI(C)>IJK(Python). The larger N gets the faster the IKJ and KIJ algorithms work than other iterations (around 40x faster when comparing cycles per inner loop iteration to array size (N). When comparing actual Gflops to peak performance of Big Red 3, IKJ (C) is 1.12% of peak performance, while IJK (python) is only 0.00035% of peak performance.

As you can see from the scatter plots, IKJ and KIJ are able to reach a larger number of Gflop/s and sustain Gflop/s as N increases, while JKI and KJI are not able to reach the same number of Gflop/s and as N increases Gflop/s decreases. IJK and KIJ’s Gflop/s also decreases as N increases, but at a slower rate than JKI and KJI. Lastly, IJK Python is never able to reach a high Gflop/s, but does stay consistent around 0.0026 Gflop/s.

|  |  |  |  |
| --- | --- | --- | --- |
|  | PAPI\_TOT\_CYC | PAPI\_TOT\_INS | IPC |
| IJK | 313890712 | 254013841 | 0.8092 |
| JIK | 290182328 | 254505736 | 0.8771 |
| KIJ | 137461316 | 133006223 | 0.9676 |
| IKJ | 87053360 | 132513218 | 1.5222 |
| JKI | 1022771872 | 532253804 | 0.5204 |
| KJI | 1035020773 | 532005800 | 0.5140 |

IKJ has more instructions per cycle than any other algorithm, while JKI/KJI have the least instructions per cycle. Therefore, IKJ is more efficient.

|  |  |  |  |
| --- | --- | --- | --- |
|  | PAPI\_L1\_DCM | PAPI\_L1\_ICM | Miss/Inst |
| IJK | 106497392 | 715 | 0.4193 |
| JIK | 114413221 | 155 | 0.4496 |
| KIJ | 15896764 | 141 | 0.1195 |
| IKJ | 15670774 | 84 | 0.1183 |
| JKI | 251930427 | 610 | 0.4733 |
| KJI | 250750904 | 247 | 0.4713 |

IKJ/KIJ have the lowest data cache miss rate, while JKI/KJI have the highest data cache miss rate. Therefore, IKJ/KIJ have the least amount of latency while accessing the main memory. There are more misses per iteration using JKI and KJI. Additionally, the cycle count is constant as N increases for IKJ and KIJ.

|  |  |  |  |
| --- | --- | --- | --- |
|  | PAPI\_L2\_DCM | PAPI\_L2\_ICM | Miss/Inst |
| IJK | 15857639 | 696 | 0.062 |
| JIK | 1906481 | 138 | 0.00749 |
| KIJ | 15837450 | 129 | 0.119 |
| IKJ | 8084642 | 76 | 0.061 |
| JKI | 16489724 | 603 | 0.031 |
| KJI | 16284086 | 244 | 0.031 |

|  |  |  |
| --- | --- | --- |
|  | PAPI\_TLB\_DM | PAPI\_TLB\_IM |
| IJK | 2708 | 13 |
| JIK | 359092 | 20 |
| KIJ | 5618 | 5 |
| IKJ | 413 | 7 |
| JKI | 1541403 | 153 |
| KJI | 1329257 | 4 |