Comparison of Vector and GPU Implementations

W. D. Cotton (NRAO), October 17, 2022

***Abstract*—This memo examines the relative performance of different vector instruction sets on a test problem imaging a moderate size VLA dataset. This problem is dominated by computation susceptible to parallelization and vectorization. The 8 float vectors of AVX showed a 30 percent improvement over the the 4 float vector SSE. AVX2 showed a variable improvement over basic AVX. Use of a GPU for gridding and "degridding" showed a substantial improvement over the multi–threaded, vectorized tests. Lacking a GPU, substantial gains can be made using the most advanced vector instruction available. Allowing more threads than available cores can reduce the run times for the vectorized implementation by 10-15%.**

***Index Terms*—Vectorization, GPU, Interoferometric Synthesis**

I. Introduction

MOST Computer processors current in use have vector capabilities that can be exploited to improve performance. These are instructions that work on blocks of data the width of the memory bus. For 128 bit wide buses these allow 4 floats or 2 doubles to be added or multiplied in parallel at the cost of a single add or multiply. These were implemented in a series of instruction sets labeled "SSE". More recently, 256 bit wide memory buses support up to 8 parallel floating adds/multiplies or 4 doubles using Advanced Vector eXtensions (AVX and AVX2) instruction sets. The current generation of 512 bit memory buses supports 16 parallel floating adds or multiplies using AVX512 instruction sets. The more recent vector extension have included richer instruction sets including features such as gather/scatter. Significant processing gains can be had if these capabilities can be utilized.

The simplest way to utilize these features is using optimization capabilities of the compilers controlled by simple compiler directives (-msse, -mavx, -mavx2, -mavx512f). One down side of this is that an attempt to execute an unsupported instruction causes the program to abort. Also, the useful memory diagnostic tool, valgrind, doesn't work for AVX512. Compiler technology lags significantly behind the hardware technology and generally don't fully exploit the vector hardware.

Frequently, a reorganization of an algorithm may be helpful in improving the efficiency of vectorization. When the compiler doesn't directly produce optimal vectorization, it can be done explicitly using "intrinsics" functions which are formally c function calls but map to a single instruction directly manipulating vector registers. There are public domain vectorized math libraries (sse_mathfun.h, avx_mathfun.h, avx2_mathfun.h, avx512_mathfun.h) implementing heavily used functions such as sincos and exp. These are distributed as header files that expand inline and are quite efficient. See https://github.com/aff3ct/MIPP/tree/master/src/math to obtain.

In addition to vector instructions universally available, some higher performance systems include attached processors such as GPUs. Parallel processing using GPUs has proven effective for continuum imaging [1]. This memo evaluates the relative performance of different levels of vectorization and GPUs using the Obit package [2] [1]. Timing results using VLA data are given.

National Radio Astronomy Observatory, 520 Edgemont Rd., Charlottesville, VA, 22903 USA email: bcotton@nrao.edu

II. Obit Vectorization

Previous Obit memos on parallelization, vectorization and GPUs are [1], [3], [4], [5], [6], [7], [8], [9], and [10]. Since the same software has to run on hardware of various capabilities and an attempt to execute missing instructions ends badly, the various vectorization specific sections of code are wrapped in #ifdefs with priority given to the most advanced version available. These are controlled by the compiler flags -DHAVE_SSE, -DHAVE_AVX, -DHAVE_AVX2, -DHAVE_AVX512, -DHAVE_GPU together with the flags to enable the various instruction sets. A lot of "hand rolling" has gone into the gridding and "degridding" routines and many members of The ObitFArray and ObitCArray classes. These routines also make heavy use of threading to maximize the use of available cores.

III. Timing Examples

A test data set of a deep VLA C+D configuration observation is used to evaluate the various vectorization/GPU implements on a variety of different platforms ranging from a laptop to a rack mounted compute server. The test used Obit wide-band imager MFImage, described in [11] and then compared the resulting CLEAN image with a "master" version computed on yet another machine. Numerical noise will cause the results of a different ordering of a set of operations to be different, comparison with a master result allows evaluation of this.

Various timing tests were performed using the range of vector capabilities available in each platform and a GPU where available. The tests use SSE, AVX, AVX2, AVX512 vectorization and GPU as available.

A. Test Data

The test VLA S band data-set ([12]) has 1,576,289 averaged visibilities with 16 spectral windows of 31 channels each. Imaging was to a radius of 0.25° needing 13 facets. The resultant image was 1439 × 1439 pixels 1.″25 in size with 15 subbands and a 7.″5 restoring beam. CLEANing used

[1] http://www.cv.nrao.edu/∼bcotton/Obit.html

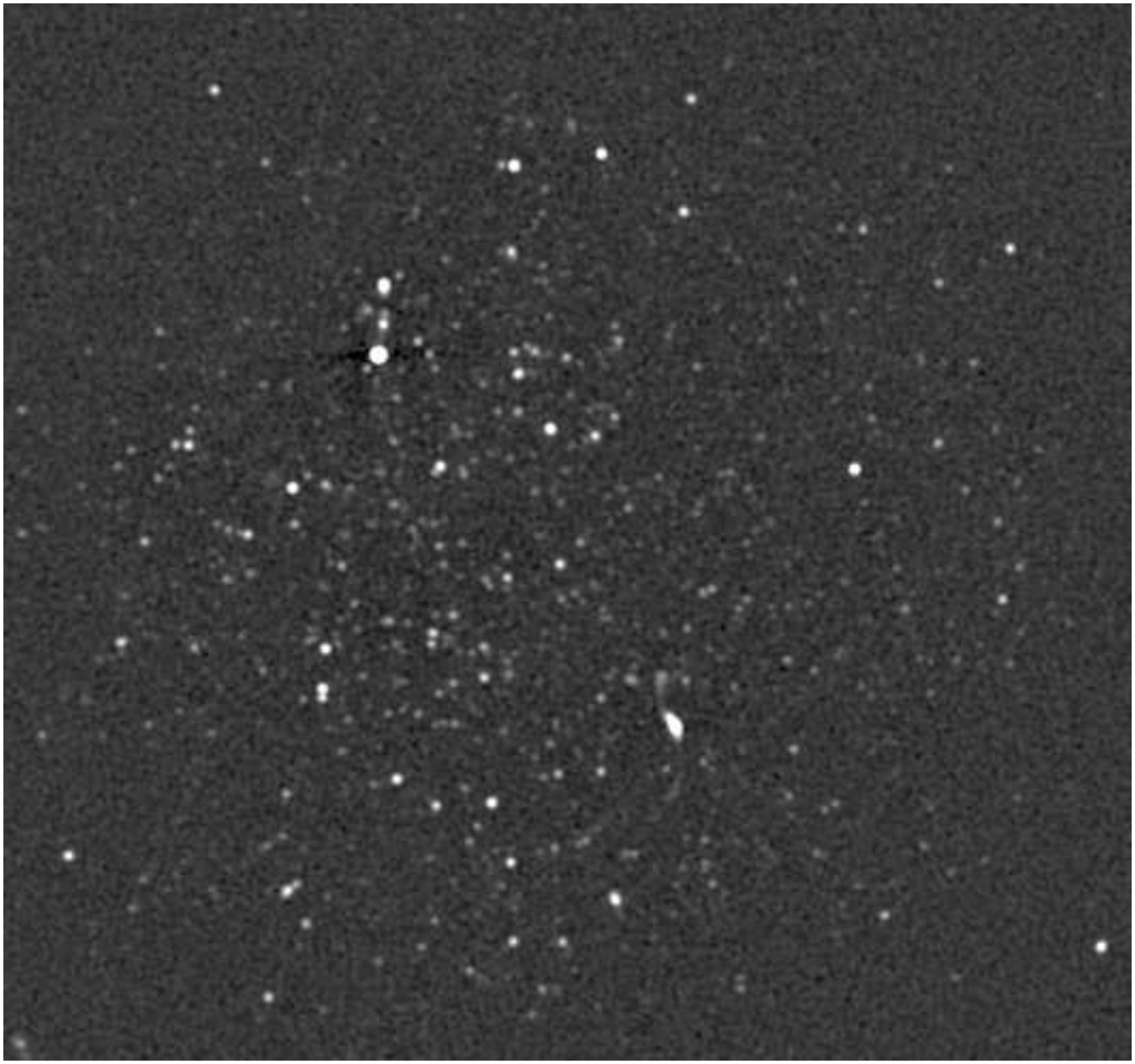

Fig. 1. Greyscale rendition of the inner part of the field being imaged, from [12]. The restoring beam is 7.″5.

5000 components in 10 major cycles; a CLEAN mask was used containing the brighter sources. The master image was computed on a 16 core (Intel(R) Xeon(R) CPU E5-2687W 0 @ 3.10GHz) work station using AVX which took 1167 sec real and 10017 sec of CPU time. The central part of the field imaged is shown in Figure 1.

B. Test Machines

A number of different computing platforms were compared.

- **leopard**
  Leopard is a laptop running RHEL8 with 6 × Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz cores, 16 GByte of RAM and a nvme disk. It has a 256 bit bus and supports up to AVX2.
- **smeagle**
  Smeagle is a RHEL7 work station with 24 (hyper-threaded) cores of Intel Xeon Gold 6136 CPU @3.00 GHz with 256 GByte of RAM, 150 GBytes of which were in a RAM disk for all files. Smeagle has a 512 bit memory bus and supports AVX512. This machine has an NVIDIA GeForce RTX 2080 Ti GPU with 68 Multiprocessors with 64 cores each (4352 cores total) and a clock speed of 1508 kHz. The CUDA capability is 7.5 with 10 GByte of global memory.
- **cheeta**
  Cheeta is a rack mounted RHEL7 compute server which has 72 (hyperthreaded) cores of Intel Xenon CPU E5-2695 v4 @ 2.1 GHz with 256 GByte of RAM, 150 GBytes of which were in a RAM disk for all files. Cheeta

TABLE I
TIMING AND IMAGE COMPARISON

Test	Real sec.	CPU sec.	Max	RMS
leopard SSE	2084	10154	2.192e-4	3.196e-7
leopard AVX	1596	7706	2.656e-4	3.319e-7
leopard AVX2	1315	6036	2.656e-4	3.319e-7
smeagle SSE	1500	21960	2.251e-4	2.206e-7
smeagle AVX	1037	14997	0	0
smeagle AVX2	934	12683	0	0
smeagle AVX512	876	11824	4.416e-4	1.458e-6
smeagle GPU	319	[365]	2.795e-3	3.576e-5
cheeta SSE	974	34283	2.476e-4	4.864e-7
cheeta AVX	700	20841	2.476e-4	4.863e-7
cheeta AVX2	683	18586	2.476e-4	4.863e-7
cheeta GPU	462	[575]	2.947e-3	3.580e-5

Notes:
Column “Real” is the total wall clock time.
Column “CPU” is the total CPU time in brackets [] if a GPU used.
Column “Max” is the fractional ratio of the most discrepant image pixel.
Column “RMS” is the RMS fractional ratio over image pixels.

TABLE II
MORE THREADS THAN CORES

Test	# core	# thread	Real sec.	CPU sec.	Ratio
leopard AVX2	6	6	1315	6036	1.00
leopard AVX2	6	12	1246	6048	1.06
leopard AVX2	6	24	1215	6125	1.08
smeagle AVX512	24	24	876	11824	1.00
smeagle AVX512	24	48	782	11234	1.12
smeagle AVX512	24	96	739	11242	1.18
cheeta AVX2	72	72	683	18586	1.00
cheeta AVX2	72	144	641	17294	1.07
cheeta AVX2	72	216	591	17163	1.16

Notes:
Column “# core” is the number of cores used.
Column “# thread” is the number of threads used.
Column “Real” is the total wall clock time.
Column “CPU” is the total CPU time in brackets [] if a GPU used.
Column “Ratio” is the ratio of real time wrt no. thread=no. core.

has a 256 bit memory bus and supports AVX2. This machine has two GPUs (only one was used for these tests) NVIDIA GeForce GTX 1080 with 20 Multiprocessors with 128 cores each (2560 cores total) and a clock speed of 1508 kHz. The CUDA capability is 6.1 with 7 GByte of global memory.

C. Run Times

Run times and image comparisons are shown in Table I. The image comparisons are with respect to the master image and are the image difference as a fraction of the master image value.

TABLE III
CPU/REAL RATIO

computer	no. core	SSE	AVX	AVX2	AVX512
leopard	6	4.87	4.8	4.6	
smeagle	24	14.6	14.5	13.6	13.5
cheeta	72	35.2	29.8	27.2	

Notes:
Column “no. core” is the number of cores used.
Column “SSE” is the CPU/Real ratio of SSE.
Column “AVX” is the CPU/Real ratio of AVX.
Column “AVX2” is the CPU/Real ratio of AVX2.
Column “AVX512” is the CPU/Real ratio of AVX512.

D. More Threads than Cores

The tests described in Section III-C show a high degree of parallelization. As shown by the utility top, the “degridding” phase of the CLEAN operation comes close to using all of the capability of the machine but the gridding utilizes roughly half of the total capability. The previous tests allowed a single thread per core (hyperthreaded or true). A number of tests using the most advanced vector operation in each machine were performed allowing a number of threads a multiple of the number of cores. The timing results are shown in Table II.

IV. DISCUSSION

Table I shows that the CPU times used greatly exceeded the run time showing the effects of multi-threading; each thread was also vectorized. These ratios are shown in Table III. NB: the GPU tests are not shown because the computation in the GPU is not accounted for. Leopard has 6 real cores and the CPU/Real ratios of nearly 5 reflect this. The “cores” in smeagle and cheeta are hyperthreaded, meaning there were only half that number but extra sets of registers were allocated to speed context switches; this gives a minor performance gain. The hyper threading explains why the CPU/Real ratios are around half of the number of “cores”. The vast majority of the computing in these tests was done in parallel sections.

Tables I and IV show that using the AVX instructions gives a substantial improvement over only the SSE instructions, roughly 1/3 for the various machines tested. This shows that the test problem was dominated by operations that could benefit from vectorization, the common case for interferometric imaging.

Both AVX and AVX2 process 8 floats in parallel but AVX2 has a stronger instruction set. This made a big difference for leopard, an additional 30% but hardly made any difference for cheeta with smeagle in between. It may be a coincidence but this correlates with age of the computer, leopard being the youngest and cheeta the oldest.

Only smeagle supported AVX512 which, while substantially faster than SSE, the AVX512 run was only 6% faster than the AVX2 time. The memory access will still be in 512 bytes

TABLE IV
TIMING COMPARISON

computer	SSE/AVX	SSE/AVX2	SSE/AVX512	SSE/GPU
leopard	1.29	1.59		
smeagle	1.44	1.61	1.71	4.70
cheeta	1.39	1.43		2.11

Notes:
Column "SSE/AVX" is the ratio of SSE to AVX run times.
Column "SSE/AVX2" is the ratio of SSE to AVX2 run times.
Column "SSE/AVX512" is the ratio of SSE to AVX512 run times.
Column "SSE/GPU" is the ratio of SSE to GPU run times.

blocks and this suggests that the second 256 was still in cache when processed by the AVX2 instructions.

The run time for the 2 GPU based runs was substantially faster than any of the vector runs. This is especially true of smeagle who's GPU had roughly twice the number of cores as cheeta.

Table II shows that increasing the number of threads allowed to a multiple of the number of cores can lead to performance gains, here a 10 to 15% reduction in run time. No more threads can run sumultaneously than the number of cores but waiting for data to be read from memory into registers can be a nontrivial part of the execution time. Having more threads than cores can reduce the memory latency cost when there are other threads ready to execute. There is a limit to how many threads can be used by an algorithm. The implementation of the gridding allows a thread per facet per imaging subband; in this test up to $13 \times 15 = 195$ threads can be usefully used.

The image comparisons with the master image shown in Table I reflect the numerical noise from a changed ordering of the numerical computations. The master was computed using AVX instructions and the AVX and AVX2 runs on the other machines are similar; for smeagle they are bitwise identical. The SSE values are also similar. Both the GPU based runs had substantially larger differences from the master, the maximum difference about an order of magnitude higher and the RMS two orders of magnitude. When a GPU is not available, optimal use of available vector instructions can improve a program's performance.

REFERENCES

[1] W. D. Cotton, "GPU-Based Visibility Gridding for Faceting," *Obit Development Memo Series*, vol. 73, pp. 1–4, 2022. [Online]. Available: https://www.cv.nrao.edu/∼bcotton/ObitDoc/GPUGridv2.pdf

[2] W. D. Cotton, "Obit: A Development Environment for Astronomical Algorithms," *PASP*, vol. 120, pp. 439–448, 2008.

[3] W. D. Cotton, "Note on the Efficacy of Multi-threading in Obit," *Obit Development Memo Series*, vol. 1, pp. 1–8, 2008. [Online]. Available: https://www.cv.nrao.edu/∼bcotton/ObitDoc/Thread.pdf

[4] ——, "A Fast Sine/Cosine Routine," *Obit Development Memo Series*, vol. 14, pp. 1–9, 2009. [Online]. Available: https://www.cv.nrao.edu/∼bcotton/ObitDoc/FastSine.pdf

[5] ——, "A Fast Exp(-x) Routine," *Obit Development Memo Series*, vol. 27, pp. 1–7, 2011. [Online]. Available: https://www.cv.nrao.edu/∼bcotton/ObitDoc/FastExp.pdf

[6] ——, "Comparison of GPU and Multithreading for Interferometric DFT Model Calculation," *Obit Development Memo Series*, vol. 35, pp. 1–5, 2014. [Online]. Available: https://www.cv.nrao.edu/∼bcotton/ObitDoc/GPUDFTv2.pdf

[7] ——, "Comparison of GPU, Single- and Multi-threading for Interferometric Gridding," *Obit Development Memo Series*, vol. 36, pp. 1–14, 2014. [Online]. Available: https://www.cv.nrao.edu/∼bcotton/ObitDoc/GPUGrid.pdf

[8] ——, "AVX2: First Look," *Obit Development Memo Series*, vol. 49, pp. 1–4, 2017. [Online]. Available: https://www.cv.nrao.edu/∼bcotton/ObitDoc/AVX2.pdf

[9] ——, "Notes on icc and AVX," *Obit Development Memo Series*, vol. 61, pp. 1–2, 2019. [Online]. Available: https://www.cv.nrao.edu/∼bcotton/ObitDoc/ICCAVX.pdf

[10] ——, "AVX512: First Look," *Obit Development Memo Series*, vol. 67, pp. 1–5, 2020. [Online]. Available: https://www.cv.nrao.edu/∼bcotton/ObitDoc/AVX512.pdf

[11] W. D. Cotton, J. J. Condon, K. I. Kellermann, M. Lacy, R. A. Perley, A. M. Matthews, T. Vernstrom, D. Scott, and J. V. Wall, "The Angular Size Distribution of μJy Radio Sources," *ApJ*, vol. 856, no. 1, p. 67, Mar. 2018.

[12] J. J. Condon, W. D. Cotton, E. B. Fomalont, K. I. Kellermann, N. Miller, R. A. Perley, D. Scott, T. Vernstrom, and J. V. Wall, "Resolving the Radio Source Background: Deeper Understanding through Confusion," *ApJ*, vol. 758, p. 23, Oct. 2012.