GPU-Based Visibility Gridding for Faceting

W. D. Cotton (NRAO), September 5, 2022

***Abstract*—This memo discusses the use of GPUs for convolving radio interferometer visibilities onto a regular grid allowing the use of FFTs. This convolutional gridding is one of the more compute intensive parts of interferometric synthesis imaging but has a number of properties making its implementation on GPUs more difficult. An implementation of gridding using faceting in Obit is described and examples are given of the application to a realistic data set showing it to be competitive with optimized CPU based gridding. On a machine with a recent GPU (4352 cores), GPU based gridding was nearly three times as fast as the AVX512 implementation on 24 cores. On an older machine whose GPU has 2560 cores, the GPU gridding was comparable with the AVX2 implementation with 72 cores.**

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

I. Introduction

RADIO interferometric "visibilities" are samples of the spatial coherence function of an incoming wavefront at random locations in the aperature plane. It is desirable to convert these onto a regular grid to allow using FFT algorithms to convert to the image plane. This is usually done by a convolutional gridding of the randomly spaced samples onto a regular grid [1] [2]. Each visibility is convolved with a kernel with desirable properties and summed onto the regular grid. This operation is one of the dominant compute intensive portions of imaging, especially in the iterative deconvolution schemes such as CLEAN [3][4] .

Graphics Processing Units (GPUs) are widely available and relatively cheap and have potentially enormous computing power with typically, many thousands of cores. It is desirable to apply this power to gridding. Unfortunately, the gridding problem is not a good match to GPUs' abilities and compromises must be made. A previous attempt at GPU based gridding is described in [5]. This memo evaluates such a technique using the Obit package [6] [1]. Examples using the MeerKAT array are described.

II. Parallelization Basics

Over a decade ago the speed of individual computer processors stopped making dramatic improvements with time. Instead, the number of processors (AKA "cores") in a given machine have increased. In addition, each core is a "vector" processor that can do multiple operations in parallel with the vector size the width of the memory bus. The current state of the art is 512 bit memory buses which allow up to 16 floating point operations to be done at the cost of one. To take advantage of this compute power, a given application must be "parallelized" allowing different parts of the problem to be handled by different cores in a series of "threads" using vectorization.

This puts constraints on the algorithms used when the inputs of one thread depends on the results computed in another, known as a "dependency". In the limit of a very large number of parallel threads (thousands), this can be a serious constraint.

III. Synthesis Imaging Overview

The following gives a top level view of synthesis imaging in the context of the CLEAN deconvolution technique as practiced in Obit. The primary focus of this memo is on convolutional gridding but the following describes how it fits into the overall process.

A. Sky Curvature and Faceting

One of the fundamental problems to be dealt with in syntheses imaging, especially at lower frequencies, is that the images produced by the simple imaging process are flat while the sky is curved [7] [8]. Away from the location where the image intersects the sky, the image can become increasingly defocused. There are several solutions used to deal with this but the one used in Obit is "faceting"[9]; creating a mosaic of facets which are all small enough that the defocusing is negligible while collectively covering the desired field of view.

At frequencies near 1 GHz, typically dozens to hundreds of facets are needed to cover the field of view of the array's antenna pattern. This allows a simple means of parallelization as the same data, with some modification, is convolved onto each of the grids. Faceting also allows the use of very compact convolution kernels reducing the cost of gridding individual facets. Faceting has the additional advantage that after the initial set is formed, only facets with emission at an interesting level need be formed in each major cycle.

B. Convolutional Gridding

Since each visibility measurement is treated as a delta function in aperature (AKA "uv") space, the convolution of it with the convolution kernel amounts to multiplying the convolution kernel, sampled on the regular grid, by the complex visibility. This product is then summed into the grid.

For reasons of efficiency, the convolution kernel needs to be relatively compact. By the convolution theorem, a convolution in the uv plane has the same effect as a multiplication in the image plane. The form of the convolution function is relatively arbitrary so can be chosen to have desirable features. The kernel generally used in Obit is an anti-aliasing filter, in the uv plane an oblate spheroid wave function [1] [2] which in the image plane approximates a rectangular top hat function greatly reducing out of band signals. This function is

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

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

implemented as a 7x7 kernel with separable u and v functions. To improve performance, the kernel is tabulated every 200th of a pixel and used in the form of a lookup table.

Once the data are convolved to the regular grid, it is Fourier transformed to the image plane to produce the "dirty" image. The point spread function (psf = the instrumental response to a point source) is produced by replacing the complex visibilities with 1+0i and performing the same operation to produce the "dirty" beam.

By the convolution theorem, convolving with this kernel is the equivalent of multiplying the image by the Fourier transform of the kernel. The Fourier transform of the convolving kernel is divided into the resultant image to correct for the artifacts introduced in the gridding.

C. CLEAN

Sampling of visibilities in the uv plane is always incomplete and uneven and this affects the quality of the image. Conceptually, this is dealt with by a "sampling" function which is zero where the visibility has not been measured and a positive value ("weight") where it has. The weight is a function of the quantity and quality of the data going into the visibility sample.

In practice, the dirty image is the convolution of the true sky with the dirty beam. This generally requires a deconvolution to obtain the best representation of the true sky. Because the sampling function contains zeroes, a non linear deconvolution is needed. The CLEAN algorithm is widely used for this as it is a very simple and robust algorithm (although does not always converge to a plausible solution). The result of CLEAN is a set of sky model components (delta functions, Gaussians, wavelets...).

A typical image/deconvolution consists of the following:

1) Grid the visibility set onto a regular axis.
2) Fourier Transform (FFT) the grid to produce dirty image and beam. Correct for the effects of gridding convolution.
3) Partial deconvolution of dirty/residual image with dirty beam to produce a partial sky model (CLEAN components).
4) Fourier transform the sky model to the locations of visibility measurements and produce a residual dataset.
5) Repeat steps 1–4 until the image residuals have reached the noise level.
6) Restore the final residual image with the CLEAN components convolved with the Gaussian approximation of the core of the dirty beam.

When the desired field of view is larger than can be adequately imaged using a single facet, a mosaic of facets covering the field of view can be imaged and deconvolved in parallel. The deconvolution of each facet needs to include the artifacts ("sidelobes") resulting from emission in other facets.

D. Sky Model Subtraction

An evaluation of the instrumental response to the partial sky model is needed in the deconvolution described above. In the simple case of a single facet, the Fourier transform of the grid containing the sky model can be interpolated to the uv locations of the visibility measurements. For multiple facets, this FFT/interpolate needs to be done independently for each facet. This operation is commonly referred to as "degridding"

Visibility datasets from modern instruments tend to be large and many passes through the data are undesirable. In the case of large numbers of facets, it is desirable to use a DFT ('Discrete Fourier Transform') based sky model calculation in which the response to each component in the CLEAN model is calculated for each visibility sample and summed. This may involve a HUGE amount of computing but modern hardware, especially GPUs are REALLY good at this [10] and may outperform the grid/FFT based method.

IV. GPUs

Graphic Processing Units (GPUs) are devices attached to computers with their own memory and processing units. They were originally optimized for video games but have a relatively general set of arithmetic instructions and can be used for general purpose computing. Data must be transferred to and from the device to be employed from a program running in the host CPU.

GPUs have a large number (thousands) of relatively slow processors that are deployed in blocks of threads to loop through a given problem. Memory latency is very long but if access is sequential through a large block of memory, such as an image, this latency is largely hidden.

A. Sky Model ("degridding"

The bulk of the computing for a DFT degridding consists of the sincos calculation of the dot product of each sky model's components sky coordinates with the u,v,w uv plane coordinates of each visibility. This operation has no dependencies, involves a large amount of computing per data point and there are optimized sincos[2] vector (SSE,AVX) and GPU functions.

B. Gridding

Gridding visibilities onto a regular grid presents several difficulties for GPU processing. The first is that each of the thousands of threads may be updating the same grid. An error occurs if the value in a cell changes between the time when a thread reads it and when the new value is rewritten. More recent NVIDIA devices have an "atomic add" function which assures that the cell value is not changed between read and write but this slows the processing, especially when there are many collisions.

Another problem is that the latency of GPU memory access is many cycles and the relatively random arrangement of visibility samples does not result in large scale sequential access through the grid. Thus, memory access can be a major bottleneck.

These two problems push the design of a GPU gridding routine in opposite directions. The dependency/atomic add

[2]The bulk of the work is the same for a sine and cosine evaluation and if both are needed, a sincos function is more efficient.

problem is reduced by having the bulk of the threads hitting widely disjoint sections of the grid while the memory latency is reduced by having the different threads hitting the same regions of the grid.

A subtle complication is that the order of execution of blocks of cuda code is not defined and thus the order in which visibilities are accumulated onto their respective grids. Due to the limited precision of floats, numerical operations done in different orders will give slightly different results. Two runs of the same operation on the same data will not give identical results.

V. GPU Gridding in Obit

Obit uses NVIDIA GPUs programmed in cuda. The nature of imaging in Obit using tasks like MFImage [11] naturally lends itself to parallelization. With faceting, the same data is gridded onto the different grids using a very compact convolution kernel. Furthermore, the frequency space is divided into bins that are gridded and imaged separately and each consists of a large number of individual frequency channels, of order of a thousand total for big continuum problems. These frequency channels are closely spaced in uv space, hence the same area of the grid reducing the latency problem. The many facets and frequency bin grids can be used to reduce the dependency problem. There are a number of axes along which the data can be split.

1) Visibility - the data from a given baseline at a given time.
2) Channel - the many frequency channels in a dataset.
3) Facet - the many sub fields of view
4) Frequency bin - the sub band set of images

Cuda allows the looping to be defined along up to 3 axes. In the Obit implementation these are 1) visibility, 2) channel, 3) facet. Which frequency bin is determined by the frequency channel. If there are eight or more facets, the maximum number of samples per block of 1024 threads is 8, 16, 8 on each of the axes. For a smaller number of facets, these are 32,32, 1. This seems to be a good compromise between reducing the problems of dependence and latency. Two processing streams were used to overlap host/device data transfers and computation.

VI. Timing Examples

A comparison of execution times was made for GPU and CPU based gridding for a large MeerKAT L band dataset on a difficult portion of the sky to image; this continuum image is dominated by a number of extended HII regions. Testing used Obit task MFImage which allows separate control of using the GPU for gridding and degridding. Each test was run repeatedly, once with GPU gridding and once with an optimized multi threaded, vectorized CPU gridding; each test used a GPU for degridding. Otherwise the data, software and control parameters were the same. A final run was made without using a GPU to compare the basic speeds of the test machines.

A. Test Data

The test MeerKAT dataset has 1,869,089 visibilities with 8 spectral windows of 119 channels each. Imaging was done in Stokes I to a radius of 0.5° with outlying facets to 1° centered on sources brighter than 50 mJy in SUMMS using a total of 42 facets and 14 $\times$ 5% fractional bandwidth frequency bins. CLEANing used 100,000 point components and autoWindowing [12] to set the CLEAN window. The GPU was allowed to use up to half the total global memory for grids.

The field of view contained multiple bright and extended sources which required many cycles of autoWindowing to construct the CLEAN window. This is a difficult CLEANing case with over 24 Jy of emission in the field.

B. Test Machines

Furthermore, the tests were run on two different CPU/GPU combinations. One machine is cheeta 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 scratch files. Other disk was SSD. Cheeta 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.

The other test machine is smeagle with 24 (hyperthreaded) 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 scratch files. The other disk was software RAID-5. 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.

C. Timing Tests

Timing tests were run using a GPU for both gridding and degridding, degridding only and for neither. The test involves a seriously nonlinear component, CLEAN deconvolution, meaning the total amount of work was not constant as differences in the order of numeric operations will cause the numerical results to differ enough to take different path to the final solution. This is especially true of tests using the GPU for gridding as the order of execution of threads is not defined. The various test results are given in Table I. The resultant images are visually indistinguishable when blinked against each other.

A simple comparison is the wall clock time to make the initial set of 42 images and beams, this is the same in all tests and can be determined from the log file. This is given in Table I as “1st”; this time also includes the FFT and correction for the gridding convolution. The entire test process is dominated by the gridding when the GPU is used for degridding so the total run time divided by the number of images formed is also an estimate of the efficiency of the gridding. This ratio is given in Table I as “time/image”.

TABLE I
TIMING TESTS

Machine	GPU	1st min.	Run min.	images	time/image sec.
smeagle	Y	2.00	76.2	1796	2.54
smeagle	N	5.67	194.1	1991	5.85
smeagle	X	5.58	460.6	1899	14.55
cheeta	Y	3.48	172.2	2262	4.04
cheeta	N	3.50	151.7	2017	4.51
cheeta	X	3.83	319.6	2103	9.19

Notes:
GPU: Y or N indicates whether the gridding used a GPU, X = no GPU.
1st is the wall clock time for the initial imaging of the 42 images and beams.
Run is the total wall clock time.
images is the total number of images made.
time/image is the total run time divided by the number of images made.

VII. DISCUSSION

Table I shows a difference in the relative efficiency of GPU based gridding between smeagle and cheeta. On smeagle, the GPU gridding reduces the run time by a factor of 2.5 with a factor of 2.8 for the time to make the initial images and beams. Cheeta on the other hand showed little difference in the gridding time between using, or not, the GPU for gridding. The difference between the two machines is that cheeta has many more, if slower, CPU cores than smeagle while its GPUs have about half the number of cores. In the head-to-head CPU comparison in Table I cheeta is 40% faster that smeagle. The more capable GPU on smeagle makes a big difference. On both machines, using the GPU for degridding made a factor of 2 difference in the total run time.

The finite precision of the calculations and the nonlinear nature of the CLEAN algorithm resulted in some variance of the number of images formed in each run. For the GPU gridding based tests, repeats of the same run will results in different number of images formed. This is due to the indeterminacy of the order of thread execution. CPU based runs on the same machine are more repeatable.

REFERENCES

[1] F. R. Schwab, "Optimal Gridding of Visibility Data in Radio Interferometry," in *Indirect Imaging. Measurement and Processing for Indirect Imaging. Australia, Cambridge University Press, Cambridge, England, New York, NY, L C # QB51.3.E43 I53 1984. ISBN # 0-521-26282-8. P.333, 1983*, 1983, pp. 333–+.

[2] ——, "Optimal Gridding of Visibility Data in Radio Interferometry," in *Indirect Imaging. Measurement and Processing for Indirect Imaging*, J. A. Roberts, Ed., 1984, pp. 333–+.

[3] J. A. Högbom, "Aperture Synthesis with a Non-Regular Distribution of Interferometer Baselines," *A&A Suppl.*, vol. 15, p. 417, Jun. 1974.

[4] B. G. Clark, "An efficient implementation of the algorithm 'CLEAN'," *A&A*, vol. 89, pp. 377–+, Sep. 1980.

[5] W. D. Cotton, "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

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

[7] R. A. Perley, "Imaging with Non-Coplanar Arrays," in *Synthesis Imaging in Radio Astronomy II*, ser. Astronomical Society of the Pacific Conference Series, G. B. Taylor, C. L. Carilli, and R. A. Perley, Eds., vol. 180, 1999, pp. 383–+.

[8] ——, "Imaging with Non-Coplanar Arrays," in *Synthesis Imaging in Radio Astronomy II*, ser. Astronomical Society of the Pacific Conference Series, G. B. Taylor, C. L. Carilli, and R. A. Perley, Eds., vol. 180, 1999, pp. 383–+.

[9] W. D. Cotton, "Multi-facet CLEANing in Obit," *Obit Development Memo Series*, vol. 15, pp. 1–5, 2009. [Online]. Available: https://www.cv.nrao.edu/∼bcotton/ObitDoc/MultiClean.pdf

[10] ——, "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

[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] W. D. Cotton, "Performance Enhancement of the autoWindow technique," *Obit Development Memo Series*, vol. 9, pp. 1–3, 2009. [Online]. Available: https://www.cv.nrao.edu/∼bcotton/ObitDoc/autoWin2.pdf