GPU based imager for radio astronomy

GTC2014, San Jose, March 27th 2014

S. Bhatnagar, National Radio Astronomy Observatory, NM, USA
P. K. Gupta, NVIDIA-India, Pune
M. Clark, NVIDIA-US, CA
Introduction

• Sanjay Bhatnagar
  – Algorithms R&D Scientist at the National Radio Astronomy Observatory, Socorro, New Mexico

• Motivation:
  – Deploy all compute-intensive imaging operations on the GPU
  – Why? How? Listen on...
Overview of the talk

• Introduction to NRAO
  – What is it? Why it is?

• Quick intro. to the scientific projects that pose the HPC problem

• Overview of RA imaging
  – What is needed for imaging with current & future telescopes

• Details of the current hot-spots
  – Motivate the three Proof-of-Concept (PoC) projects
  – Progress so far, future plans
National Radio Astronomy Observatory

• A NSF funded national observatory

• To build and operate large radio facilities
  – Operate three of the largest ground-based radio telescopes
    » EVLA, ALMA, VLBA, GBT
  – Central Development Lab (CDL):
  – Digital Correlators for EVLA, ALMA, VLBA; digital back-end for the GBT
  – Off-line software for calibration, imaging and astronomical computing
    » Open source
    » Most widely used RA imaging software world-wide
    » Runs on laptops, desk-tops, Compute Clusters
    » Now exploring using GPUs to mitigate compute and memory footprint hotspots
The Very Large Array (NM, USA)

- Very Large Array
- 27 antennas
- Antennas movable on rails
  1 – 27 Km radius
- Spread over 27 Km radius
- Size of the “lens” 30 Km
- Frequency range 300 MHz – 50 GHz
Atacama Large MM Array (ALMA), Chile

- In partnership with EU & Japan
- At an altitude of 16,500 ft
- 50 antennas
- Effective size of the lens: 3 Km
- Frequency range 100 GHz – 950 GHz
- Re-configurable with antennas movable on a special transporter.
Very Long Baseline Array (VLBA), US

- 10 antennas
- Antennas across the US
- Size of the “lens” few 1000 Km
- Frequency range 100 GHz – 950 GHz
- Angular resolution milli arcsec
Other RA Observatories in the world:

- VLA
- CARMA
- LWA
- ALMA
- MeerKAT
- LOFAR
- PAPER
- WSRT
- GMRT
- ATCA
- ASKAP
- SKA (to be built)
- MWA
Other RA Observatories in the world

Imaging with all of these pose the combined HPC + Big Data problems
Interferometric Imaging: Big Data

• Uses the technique of Aperture Synthesis to synthesize a “lens” of size equal to the maximum separation between the antennas

• Not a direct imaging telescope: Data is in the Fourier Domain

• Image reconstruction using iterative algorithms
  – Data volume with existing telescopes: 10 – 200 TB
    with SKA Telescope: Exa Bytes

Effective data i/o for image reconstruction: 10x
Interferometric Imaging: HPC

• Sensitivity improvements of 10x or more in modern telescopes
  – What was an ignorable/too weak an effect earlier, now limits the imaging performance of modern telescopes

• Need more compute-intensive imaging algorithms
  – Tera/Peta Flops now. Exa Flops SKA (soon....ish)

• Orders of magnitude more expensive algorithms for imaging using many orders of magnitude more data

   Imaging algorithms in CASA for EVLA and ALMA
   The “aw-imager” for LOFAR (NL) – a modified version of CASA Imager
   ASKAP Imager (AU) optimized for large cluster computing
Interferometric Imaging

- **Bottom line for computing**
  - Tera–Exa(SKA) scale computing using Tera–Exa(SKA) Bytes of data to make Giga-Pixel images

- **Full end-to-end processing will require a cluster**
Computing hot spots

• Gridding / de-gridding: PoC - 1
  – Irregularly sampled data to a regular grid for FFT

• Computing Convolution Functions (CF): PoC - 2
  – Computing convolution kernels for Gridding
    » Pre-compute-n-cache OR On-demand computing
    » Memory foot print issues

• Wide-band Image Reconstruction: PoC - 3
  – Requires convolutions of many large images
    » Pre-compute-n-cache OR
    » On-demand Computing using GPU
Scientific projects: Deep Imaging

- Requires high resolution, high dynamic range imaging
  - i.e. Big Data
  - Compute-intensive algorithms
- Large area surveys of the radio sky

- Dynamic range
  - Ratio of strongest to weakest Source: $10^6$
- Dynamic range of raw images $10^2 - 3$
- Need high resolution to reduce background sky-noise ("confusion noise")
Scientific projects: Deep Imaging

- Requires high resolution, high dynamic range imaging
  - i.e. Big Data
  - Compute-intensive algorithms
- Large area surveys of the radio sky

- Dynamic range
  - Ratio of strongest to weakest Source: $10^6$
  - Dynamic range of raw images $10^2 - 3$
  - Need high resolution to reduce background sky-noise ("confusion noise")
Scientific projects: Fast Imaging

- Transient sky
  - Storing data at high time resolution for later processing is not an option
  - Needs fast (near real-time) imaging → as a trigger to store bursts of data

- A short blip in time
  - Spike of ms in 10s of hours of data

- Data rate too high to be recorded at ms resolution

- Need fast imaging as a trigger to record short bursts of data

- Need interferometric imaging to localize on the sky

Thornton et al., Science, 2013
Scientific projects: Fast Imaging

- Transient sky
  - Storing data at high time resolution for later processing is not an option
  - Needs fast (near real-time) imaging → as a trigger to store busts of data

  - A short blip in time
    - Spike of ms in 10s of hours of data
  - Data rate too high to be recorded at ms resolution
  - Need fast imaging as a trigger to record short busts of data
  - Need interferometric imaging to localize on the sky

Thornton et al., Science, 2013

10s of hours

ms
Aperture Synthesis Imaging: Why?

- Single dish Resolution too low for many scientific investigations
  - Limited collecting area + resolution limits sensitivity at low frequencies

Single dish resolving power
\[
\frac{\text{Wavelength}}{\text{Dish Diameter}}
\]

Biggest steerable single dish
\[= 100 \text{ m}\]
Aperture Synthesis Imaging: Why?

- Single dish Resolution too low for many scientific investigations
  - Limited sensitivity/limits sensitivity at low frequencies

Synthesis Array resolving power

\[
\frac{\text{Wavelength}}{\text{Max. separation between antennas}}
\]

Max. separation in VLA = 35 km

Resolution: ~ 350x better
**Aperture Synthesis Imaging: How?**

- **Aperture Synthesis** or **Fourier Synthesis Technique**
  - An interferometric imaging technique (Nobel Prize in '74)
  - Many antennas separated by 10s – 100s Km
  - Each pair of antennas measure **one** Fourier Component

- Synthesized aperture equal to the largest separation between antennas
Aperture Synthesis Imaging: How?

- Aperture Synthesis
  - An interferometric imaging technique (Nobel Prize in '74)
  - Many antennas separated by 10s – 100s Km
  - Each pair of antennas measure another Fourier Component

- Synthesized aperture equal to the largest separation between antennas
Aperture Synthesis Imaging: How?

- **Aperture Synthesis**
  - An interferometric imaging technique (Nobel Prize in '74)
  - Many antennas separated by 10s – 100s Km
  - Each pair of antennas measure *another (one)* Fourier Component

- Synthesized aperture equal to the largest separation between antennas
Aperture Synthesis Imaging: How?

- Aperture Synthesis
  - An interferometric imaging technique (Nobel Prize in '74)
  - Many antennas separated by 10s – 100s Km
  - All pairs with one antenna measure N-1 Fourier Component = 26

- Synthesized aperture equal to the largest separation between antennas
Aperture Synthesis Imaging: How?

- Aperture Synthesis
  - An interferometric imaging technique (Nobel Prize in '74)
  - Many antennas separated by 10s – 100s Km

  - **All** pairs with **all** antenna measure $N(N-1)/2$ Fourier Component = 351

- Synthesized aperture equal to the largest separation between antennas
Aperture Synthesis Imaging: How?

- Aperture Synthesis
  - Use **Earth Rotation Synthesis** to fill the Fourier plane
  - **All** pairs with **all** antenna measures \(N(N-1)/2\) Fourier Component

  \[\text{Measure } N(N-1)/2 \times 2 \text{ Fourier components over 2 integration time} = 702\]

- Synthesized aperture equal to the largest separation between antennas
Aperture Synthesis Imaging: How?

- Aperture Synthesis
  - Use **Earth Rotation Synthesis** to fill the Fourier plane
  - **All** pairs with **all** antenna measures $N(N-1)/2$ Fourier Component

- Measure $N(N-1)/2 \times 10$ Fourier components over 10 integrations = **7020**

- Synthesized aperture equal to the largest separation between antennas
Aperture Synthesis Imaging: How?

- Aperture Synthesis
  - Use **Earth Rotation Synthesis** to fill the Fourier plane
  - All pairs with all antenna measures $N(N-1)/2$ Fourier Component

- Fourier Components measured over 10 hr: $O(10^{11-12})$

- Data Size: 10s – 100s TB now Up to Exa Bytes for SKA-class telescopes
- Data not on a regular grid.
Interferometric Imaging

- Aperture Synthesis Imaging
  - Indirect imaging: data in the Fourier domain
  - Incomplete sampling → artifacts in the image

Sky  

Telescope  

Data Visibilities

Raw Image  

Dirty Image

Telescope Transfer Function  

Point Spread Function

*(Convolution)*

Fourier Transform
Interferometric Imaging

- Raw image (FT of the raw data) is dynamic range limited

\[
\text{Dynamic range: } > 1 : 1000, 000
\]

- Processing: Remove telescope artifacts to reconstruct the sky brightness

- Image reconstruction is a High-Performance-Computing-using-Big-Data problem

\[
\text{Dynamic range: } 1 : 1000
\]
Interferometric Imaging

• Image reconstruction is an ill-posed Inverse Problem

\[ \mathbf{D} = \mathbf{A} \mathbf{I}^{\text{true}} \]

D: The Raw Data  
A: The Measurement Matrix  
\( \mathbf{I}^{\text{true}} \): The True Sky Brightness distribution

• Recover \( \mathbf{I}^{\text{true}} \) given \( \mathbf{D} \)

\[ \mathbf{A}^{-1} \mathbf{D} = \mathbf{I}^{\text{True}} \]

• \( \mathbf{A} \) is singular \( \implies \) Non-linear (iterative) algorithms required to reconstruct the Sky Brightness distribution

• Typically 10 iterations, using ALL the data in each iteration

Raw Data: 10 – 100 TB  Effective data volume: 100 – 1000 TB
The Computing Problem

- Basic computing steps
  1. Use FFT to transform to the image domain: **Gridding + FFT**
  2. Image-plane deconvolution of the PSF: **Search and subtract on images**
  3. Inverse transform to the data domain: **De-gridding + Inv. FFT**

---

**Image Domain**

- Iterative in nature

**Data Domain**

- Use all data

- Resample: Regular grid to Irregularly sampled data

---

**Use all data**

**Image deconvolution**

**Iterative in nature**

---

**Resample**

**On regular grid**

---

FFT

FFT\(^{-1}\)
The Computing Problem

- Basic computing steps
  1. Use FFT to transform to the image domain: Gridding + FFT
  2. Image-plane deconvolution of the PSF: Search and subtract on images
  3. Inverse transform to the data domain: De-gridding + Inv. FFT
Computing architecture

- Make images on the GPU
  - Use GPU as a Gridding + FFT server
  - CPU host for image reconstruction
Computing architecture

- Make images on the GPU
  - Use GPU as a Gridding + FFT server
  - CPU host for image reconstruction + GPU as a image convolution server
Computing architecture

- Fast imaging
  - 100s of image in milli seconds on the GPU
  - Search for peak on the GPU
  - If peak found, send a trigger to the host to save the data buffer on the disk
The Computing Problem: Why Gridding?

- Use FFT to transform to the image domain
  - Raw data is not on a regular grid
The Computing Problem: Why Gridding?

• Use FFT to transform to the image domain
  – Raw data is not on a regular grid
The Computing Problem: Why Gridding?

- Use FFT to transform to the image domain
  - Raw data is not on a regular grid
The Computing Problem: Why Gridding?

- Use FFT to transform to the image domain
  - Raw data is not on a regular grid

Requires re-sampling ALL the data on a regular grid
Using 2D Convolutional resampling → 2D Interpolation
Gridding – How?

- Gridding/De-gridding $\leftrightarrow$ 2D Interpolation via convolutional re-sampling
- Convolution Function $\leftrightarrow$ 2D Weighting Function
Gridding – How?

- Gridding/De-gridding $\leftrightarrow$ 2D Interpolation via convolutional re-sampling
- Convolution Function $\leftrightarrow$ 2D Weighting Function

![Diagram showing gridding and de-gridding process with convolution function and weighting function](image)
Gridding – How?

- Gridding/De-gridding $\leftrightarrow$ 2D Interpolation via convolutional resampling
- Convolution Function $\leftrightarrow$ 2D Weighting Function

![Diagram showing 2D Convolution process with N x N Complex Multiply and N x N Complex Additions.]
Gridding – How?

- Gridding/De-gridding $\leftrightarrow$ 2D Interpolation via convolutional resampling
- Convolution Function $\leftrightarrow$ 2D Weighting Function

![Diagram showing 2D Convolution Function](image)
Gridding – How?

- Gridding/De-gridding $\leftrightarrow$ 2D Interpolation via convolutional resampling
- Convolution Function $\leftrightarrow$ 2D Weighting Function

Massively Parallel H/W should help: PoC 1
Status-3: Gridding – PoC-1 (on-going)

- Divide the grid into sub-grids, each with multiple pixels

- Map each sub-grid to a CUDA Block of threads
  - One thread per sub-grid pixel

- For each data $D_i$
  
  Calculate the range of the CF centered on $D_i$
  
  If this block in range
  
  For all threads in range
  
  $\text{local\_Grid}_j + = D_i \times \text{Cf}_{ij}$

Write local_Grid to GMEM Grid
Status-3: Gridding – PoC-1 (on-going)

For Data 1
Calculate the range of the CF centered on $D_i$
If this block is in range
For all threads in range
local_GRID$_j += D_i \cdot C_{f_{ij}}$
Write local_GRID to GMEM

Sub-grids/Blocks of 10x10 threads
Status-3: Gridding – PoC-1 (on-going)

For Data 2
- Calculate the range of the CF centered on \( D_i \)
- If this block in range
  - For all threads in range
  - \( \text{local}_{Grid} \) += \( D_i \cdot Cf_{ij} \)
- Write local_{Grid} to GMEM Grid

Sub-grids/Blocks of 10x10 threads
Status-3: Gridding – PoC-1 (on-going)

For Data 3:
Calculate the range of the CF centered on $D_i$
If this block in range
For all threads in range
$\text{local}_i \leftarrow D_i \ast \text{Cf}_i$
Write $\text{local}_i$ to GMEM Grid

Sub-grids/Blocks of 10x10 threads

\begin{align*}
\text{local}_i & \leftarrow D_i \ast \text{Cf}_i \\
\text{Write local}_i \text{Grid to GMEM Grid}
\end{align*}
Status-3: Gridding – PoC-1 (on-going)

- No thread contentions
  - No atomic-operations required

- But current implementation is limited by global memory accesses

- Solution: reduce GMEM access
  - Coarse-grid the raw data / sort the raw data
  - Copy data required for each block to SMEM

- Cache-coherent access
  - Re-arrange GMEM buffers
Convolution Functions (CFs)

- Convolution Functions encode the physics of the measurement process
- Prolate Spheroidal: As anti-aliasing operator
- W-Term: Account for Fresnel Propagation term
- A-Term: Account for antenna optics
- Final function: Convolution of all three
  - $PS \ast W\text{-Term} \ast A\text{-Term}$
  - $N \times N = 10 \times 10$ – few $\times$ 100$\times$100

In use in
casa (nrao),
“aw-imager” for lofar (nl),
askap imager (au)
Compute CFs on the GPU: PoC - 2

• CFs as tabulated functions in the computer RAM

• Minimize the quantization errors, by over-sampling
  – Typical over-sampling 10x – 100x

• Memory footprint gets prohibitive
  – Total Memory = $10^{3-8} \times 1000s$ of CF = 10s – 100s GB

• PoC-2: Can we compute the CFs on-the-fly (as against compute-n-cache)?
  – Compute + Multiply + FFT
**Status-1: CF Computations – PoC-2**

- Negligible I/O – mostly computing

\[ T_0 \]

\[ \text{FFT} \]

\{ CF set used for 100s of gridding cycles \}

\{ W-Terms \}

\{ A-Term \}

\( \times \)

\[ \text{Compute per pixel} \]

\[ \text{Cached} \]

\[ \text{Analytically computed} \]

\[ \text{No analytical expr.} \]
Status-1: CF Computations – PoC-2

- Negligible I/O – mostly computing

W-Terms

A-Term

$T_0$

$T_1$

100s – 1000s

Compute per pixel

Cached

$\times$

$\times$

100s of gridding cycles

100s of gridding cycles

$\times$

$\times$

$\times$

FFT

FFT

100s – 1000s

Compute per pixel

Cached

Analytically computed

No analytical expr.

W-Terms

A-Term

Gridding

Image Conv.

FFT

Bgrid

CF

2

FFT

Image Conv.
Status-1: CF Computations – PoC-2

- Negligible I/O – mostly computing

W-Terms

- Compute per pixel
  - Analytically computed

A-Term

- Cached
  - No analytical expr.

FFT

10s - 100s

100s - 1000s

CF set used for 100s of gridding cycles

Gridding

Image Conv.

S. Bhatnagar: GTC 2014, San Jose, USA, March 27th 2014
Status-1: Compute CFs on GPU – PoC-2

- Negligible I/O – mostly computing

\[
\{ \begin{array}{ccc}
\text{\includegraphics[width=0.1\textwidth]{image1.png}} & \includegraphics[width=0.1\textwidth]{image2.png} & \includegraphics[width=0.1\textwidth]{image3.png} \\
\end{array}\}
\]

- GPU: Pre-compute A-Term and cache it in GPU GMEM
  - Compute W-term OTF – one thread per pixel
  - Multiple A x W
  - FFT

- Sizes involves: 2K x 2K Complex images
- GPU: 1024 CFs made in ~1 ms.
  - ~20x faster than CPU
  - Room for improvement by another 2 - 3x
**Image reconstruction: PoC - 3**

- **Simplest algorithm: CLEAN**
  - Iteratively search for the peak in the Raw image and subtract the PSF image at the location of the peak

- **Most complex algorithms: Multi-scale Multi-freq. Synthesis (MS-MFS)**
  - Requires convolutions of large images +
  - Requires CLEAN

- **Use GPU as a convolution-server PoC – 3**
  Do deconvolution on the GPU (future)
Status-2: Image deconvolution – PoC-3

• **Wide-band image reconstruction**
  - Multi-Term Multi-Scale
  - $N_{\text{terms}} = 2 - 3 \quad N_{\text{scales}} = 3 - 10$

• **Computing cost**
  - $N_{\text{terms}}$ gridding cycles
  - Convolutions of $N_{\text{Terms}}^2 \times N_{\text{scales}}^2$ images
  - Search in $N_{\text{terms}}$ images
**Status-2: Image deconvolution – PoC-3**

- High memory footprint: High resolution wide-band imaging currently not possible
- Solution being pursued
  - Use GPU as an enabler technology (high resolution wide-band imaging)
  - Compute the multi-scale images OTF
  - Use GPU as a convolution-server

![Graph showing time vs. image size for FFT and multi-scale image computing](image-url)
Low DR, fast imaging needs

• Transient source localization on the sky
  – Data rates too high for store-n-process approach
  – Need fast, low-DR imaging to trigger storage of short bursts of data

• EVLA: Data dumps every 5 ms

• Computing
  – Make 119 images (DM search): 1K x 1K size
  – Trigger storage if peak > threshold

• Current CPU based processing (14 nodes x 16 cores)
  – ~10x slower than real-time
Fast-imaging GPU pipeline

• Simplify the gridded

• On-GPU FFT

• On-GPU peak detection
  – If (peak > threshold) trigger data storage

• Compute to I/O ratio $\sim O(10^{5-6})$
  – Data (@900MB per sec) goes into the GPU
  – Only trigger info. Comes out
Fast-imaging GPU pipeline estimates

- Imaging is FFT-limited

- GPU: Gridding + FFT + Peak search
  - Once per ~1 ms
  - 50 (100x?) faster than single CPU core

- Initial estimates for fast-imaging (work-in-progress):
  - 5 (2?) K20Xs become comparable to 14x16 CPU cores
    » 10x slower than real-time
  - 50 (25?) K20X GPU cluster can enable real-time processing
Conclusions, future work

- The algorithms for the three hot-spots ported on GPU (the three PoCs)

- Work in progress on the gridding algorithm
  - Minimize Global Memory transactions, other optimizations
  - Take decision about which algorithm to use

- Optimize the CF severe code and image convolution code

- Integrate to make a imaging pipeline

- Scientifically test the results
  - Measure actual run-time performance with real data

- Prototype and check Fast Imaging pipeline
  - If the estimates of run-time improvements hold up, deploy for real-time fast-imaging
Back up slides
Number of CFs required

- Not all CF terms can be computed analytically
  - Final convolution function can't be computed analytically

- No. of CFs required for wide-band, full-polarization high-dynamic range imaging is large

  Total number of CFs: 10s – 1000s

- Expensive to compute
  - Current solution: Pre-compute and cache
  - Memory Footprint issues
Computing architecture

- Make images on the GPU
  - Use GPU as a Gridding + FFT server
  - CPU host for deconvolution

- GPU as a Image Convolution + on-demand CF server
Computing architecture

• Make images on the GPU
  – Use GPU as a Gridding + FFT server
  – CPU host for deconvolution

• GPU as an Enabling Technology
  – GPU as a Convolution and Convolution Functions server
  – Where GPU RAM is not sufficient to hold all CF and buffers for MS-MFS
  – Imaging + Deconvolution loops on the Host

• GPU as a trigger for fast-transients
  – 100s of images from a given set of data
  – Image + search for transients on the GPU
Aperture Synthesis Imaging: How?

Each Pair of Antennas:

=> Measures one 2D “fringe”

“fringe spacing”, “orientation”, “amplitude”, “phase”

Complex FFT

Correlator

Multiply Signals from All antennas With all other antennas

Data Processing

Disk

Integrator
Aperture Synthesis Imaging: How?

- Aperture Synthesis
  - An interferometric imaging technique (Nobel Prize in '74)
  - Many antennas separated by 10s – 100s Km
  - Each pair of antennas measure another (one) Fourier Component

- Synthesized aperture equal to the largest separation between antennas
Gridding on the GPU: PoC - 1

- Each data point is multiplied by a NxN complex Convolution Function followed by NxN additions to the Global Grid

\[ N_{\text{data}} \times N^2_{\text{CF}} \times 8 \text{ FLOP} + \text{overheads} \]

\[ = O(10^{10-12}) \times (10 \times 10) \times 8.2 = 100 \text{ TFLOP} \]

\[ \times (100 \times 100) \times 8.2 = \sim \text{PFLOP} \]

- SKA

\[ O(10^{15}) \times \ldots \times 8.2 = \sim \text{ExaFLOP} \]

- Gridding cost dominates computing load for all imaging

- **Compute to I/O ratio:** \( 10^{2-5} \)

Massively Parallel H/W should help: PoC 1
Status-3: Gridding – PoC-1 (on-going)

- Gridding / de-gridding:
  - Dominates to cost of High Dynamic Range imaging

- Compute to i/o ratio: $10^{2-5}$
  - Dominant cost for most imaging
  - Wide-band imaging: comparable to the cost of the deconvolution step

- Scaling:
  - Run-time cost: (data volume) $\times$ (CF size)
  - W-, AW-Projection: $10^{12} \times 10^{2-5}$ FLOP
  - A-Projection: $10^{12} \times 10^{2-3}$ FLOP

- Existing literature
  - Non-imaging: Margo et al. (2013),
  - FPGA: Clarke et al. (2014)
Each element is a Convolution of 4 functions (precomputed once)

Every iteration evaluates this matrix multiply/convolution.
Status-1: CF Memory footprint – PoC-2

• Negligible I/O – mostly computing

• Tabulated CF requires oversampling to minimize quantization errors

• Memory per CF for high DR imaging with the EVLA:
  – Oversampling: 100; Pixels: 2K x 2K = 8 Mbytes
  – No. of CFs: 100 W-Terms x 100 A-Terms = 10^4
  – Total memory footprint: 80 GB

• Memory footprint for SKA several orders of magnitude larger
Status-3: Gridding – PoC-1 (on-going)

- Solutions: Load balancing
  - Non-regular sub-grids