GPU's for Radio Imaging

V. Vamsi Krishna (Kapteyn Astronomical Institute)
Dr. Panos Labropoulos (Kapteyn/ASTRON)
Prof. Leon V.E. Koopmans (Kapteyn)
Lofar Eor Project

● Innovative radio telescope
● Low Frequency Array telescope (dipoles)
● Wide field array
● EOR one of the Key Science projects
  ○ Own cluster with 80 nodes.
  ○ 2 GPUS.
  ○ http://www.astro.rug.nl/eor
● http://lofar.org/
Radio Imaging

- Signals from Sources (e.g. galaxies)
- Next Gen Antennas (e.g. LOFAR, SKA, ...)
- Image acquired after Processing (RFI elimination, Calibration).

\[ V(u, v, w) = \iiint I(l, m) e^{-2\pi i (ul + vm + w(\sqrt{1 - l^2 - m^2} - 1))} \frac{dldm}{\sqrt{1 - l^2 - m^2}} \]

\[ V(u, v, w) = \iiint I(l, m) e^{-2\pi i (ul + vm)} dldm \]

w-term often neglected because in many cases \( l^2 + m^2 < < 1 \)
Basic Radio Imaging

- Basic method of Imaging
  - Grid the data on a regular grid
  - Find Fourier Transform

- Things to be parallelized
  - Gridding (gridding kernel)
  - FFT (CUFFT)
Casa Imager

- DeFacto imaging software package
- Uses FFTW for the FFT calculation
- FFTW uses multicore cpu
- Uses casacore just like the gpu imager for loading the data
System Config

- **2 x Tesla M1060 (GPU)**
  - Compute aProcessors @ 602 MHz
  - 4GB Memory
- **8 x Intel(R) Xeon(R) CPU E5520 @2.27GHz**
  - Hyper Threading support
  - 12 GB Ram
Performance

About 5-6 Times faster
W-Projection
Intro

$V(u, v, w) = \iiint I(l, m) e^{-2\pi i (ul + vm)} dldm$

- Assumptions not true anymore for current arrays (e.g. LOFAR).
- Have to include the $w$-term.

$V(u, v, w) = \iiint e^{i2\pi w(\sqrt{1-l^2-m^2}-1)} dldm \iiint \frac{I(l, m)e^{-2i(ul+vm)}}{\sqrt{1-l^2-m^2}} dldm$
W-Stacks

- Cuda kernels for applying correction at each image plane (created for each w-slice).
- Average all w-plane images to obtain final image.
- 10 times faster than the widely used casa imager.
Simulation Results
Results

Without W-Projection

With W-Projection
Performance

GPU vs CASA

**Time in Seconds**

**Number of W-Planes**

- **GPU**
- **CASA**

Comparison of GPU and CASA performance with varying numbers of W-Planes.
MVDR

Minimum Variance Distortionless Response
\[ I(l, m) = \frac{1}{KP^2} \sum_{k=0}^{K} a_k(l, m)^H R_k a_k(l, m) \]

Bartlett beamformer

\[ I(l, m) = \sum_{k=0}^{K} W_k^H R_k W_k \]

\[ W_{mvdr} = \frac{a^H R^{-1}}{a^H R^{-1} a} \quad a_k(l, m) = \begin{pmatrix} e^{i2\pi(u_{k01}l+v_{k01}m)} \\ e^{i2\pi(u_{k02}l+v_{k02}m)} \\ \vdots \\ e^{i2\pi(u_{k0P}l+v_{k0P}m)} \end{pmatrix} \]

MVDR

Issues to consider:

- Computationally intensive
- $O(n^2)$ complex operations for each pixel
- Diminish the PSF side lobe effects.
- Visibility matrix not always invertible.
- Regularize the matrix
- Use Cholesky Decomposition to find the regularisation factor.
Block Diagram

Main Process

File Writer

Imager

Epoch Selector

Image Accumulator

Final Image

Block of Threads

Shared Memory

Image (GridDim)
Implementation (GPU)

- Each thread calculates $aH^* R_{inv}[i]$ (vector).
- Each thread calculates $(aH^* R_{inv})[i]*a$ (scalar).
- A weight vector $W$ is created $w[i] = aH^*R[i]/$scalar.
- Once all components are updated find $wH^* R * W$ (each thread computes one component as above).
- Pixel is accumulated in shared memory and saved.
- An Image for each "epoch" is saved.
- Select correct images based on regularization term.
- Averaged Image saved.
GPU Kernel

- $a^H R_{inv}^a$ (scalar, shared memory)
- $a$ (vector, shared memory)

Num Antennas

Threads

$R$ inverse (global memory)
Implementation (CPU)

- Launch 16 threads
- Each thread calculates portion of the Image.
- Kernel is similar to the GPU version.
- Number of threads on GPU num_antennas per pixel.
Regularization
Results
Results

FFT

MVDR
Results
Performance

**Graph: GPU vs CPU**

- **Y-axis:** Time in Minutes
- **X-axis:** Size of Image
- Data points for:
  - CPU MP
  - GPU
  - GPUx2

As the size of the image increases, the time taken by CPU MP, GPU, and GPUx2 to process the image increases significantly, with GPUx2 showing the highest time.
Conclusion

- We have shown that GPUs can be used to accelerate Radio Imaging software.
- We are able to look at more compute intensive methods of imaging.
Future Work

- Implement standard functions used in Imaging software packages to be feature complete.
- Experiment with other imaging techniques eg QPMVDR.
Thank You