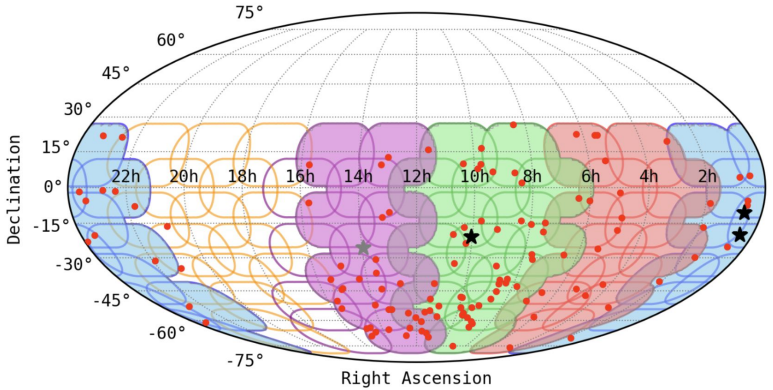
Speeding up the Pulsar Search Pipeline

Piyush Panchal (SCITAS, EPFL)

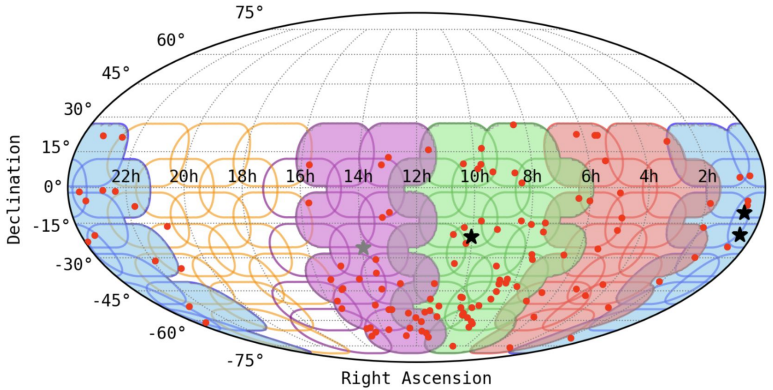
Introduction

- 4 PB data collected in SMART Pulsar survey (MWA)



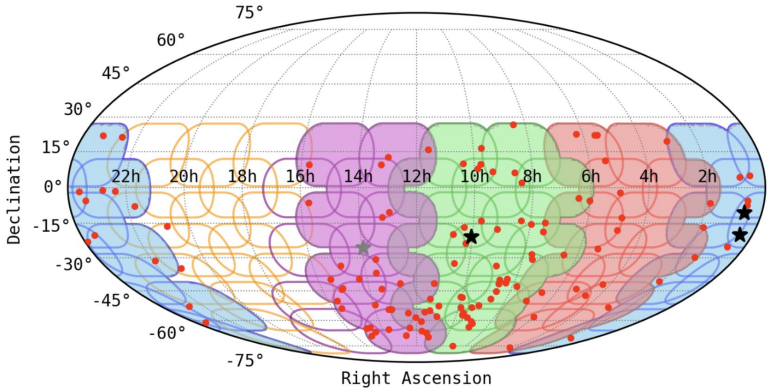
Introduction

- 4 PB data collected in SMART Pulsar survey (MWA)
- Shallow processing (~10% data) using Presto: ~300K CPU hours



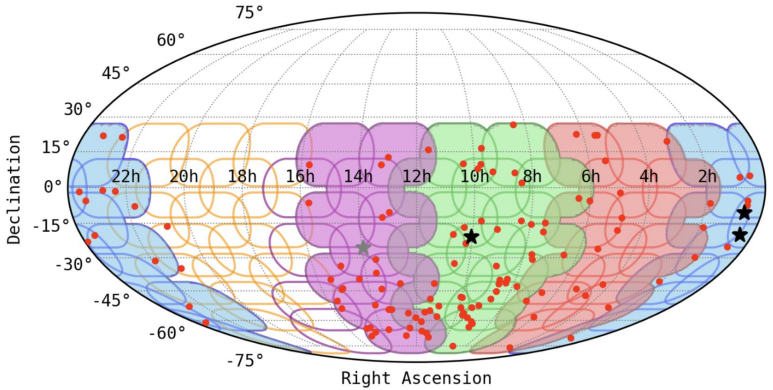
Introduction

- 4 PB data collected in SMART Pulsar survey (MWA)
- Shallow processing (~10% data) using Presto: ~300K CPU hours
- Problem: Full processing using Presto in ~23 years



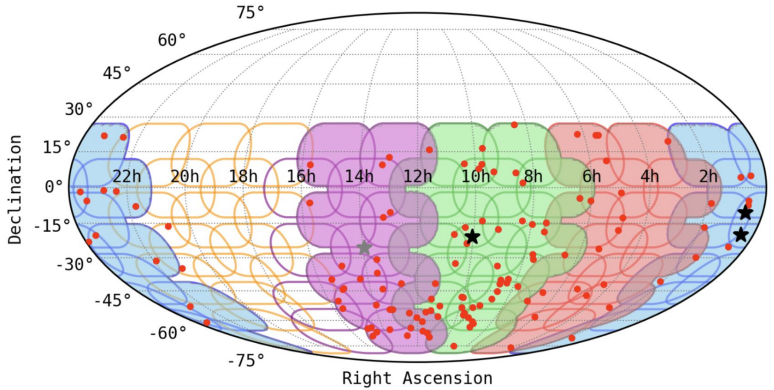
Introduction

- 4 PB data collected in SMART Pulsar survey (MWA)
- Shallow processing (~10% data) using Presto: ~300K CPU hours
- Problem: Full processing using Presto in ~23 years
- Motivation: Pulsars help studying extreme physics



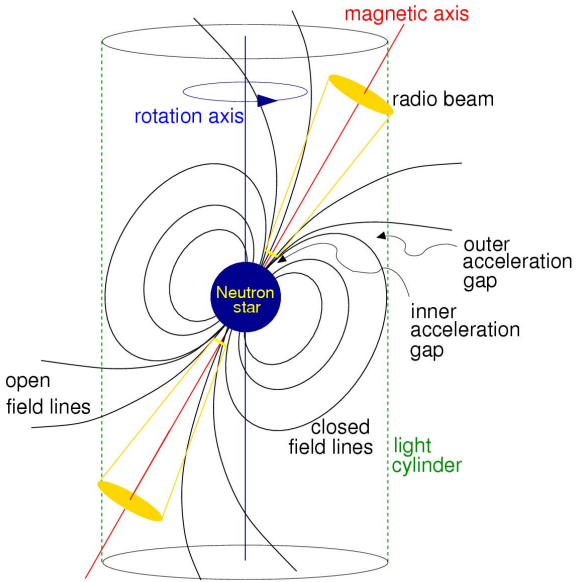
Introduction

- 4 PB data collected in SMART Pulsar survey (MWA)
- Shallow processing (~10% data) using Presto: ~300K CPU hours
- Problem: Full processing using Presto in ~23 years
- Motivation: Pulsars help studying extreme physics
- Goal: GPU acceleration to enable full/faster processing



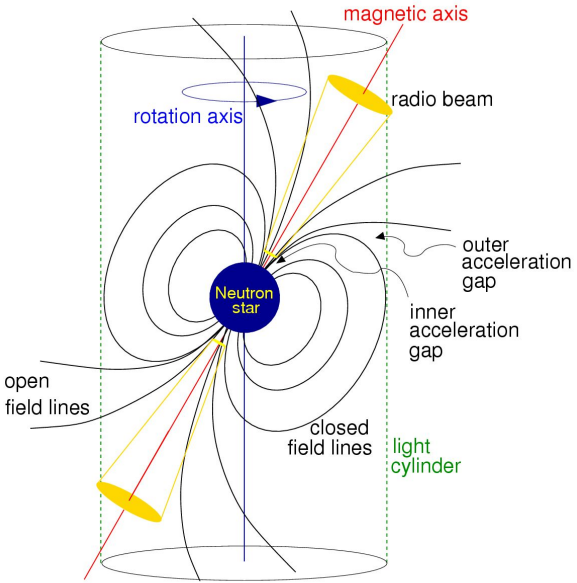
Pulsar Searching

- Pulsar: Rotating Neutron star, regular EM radiation from poles



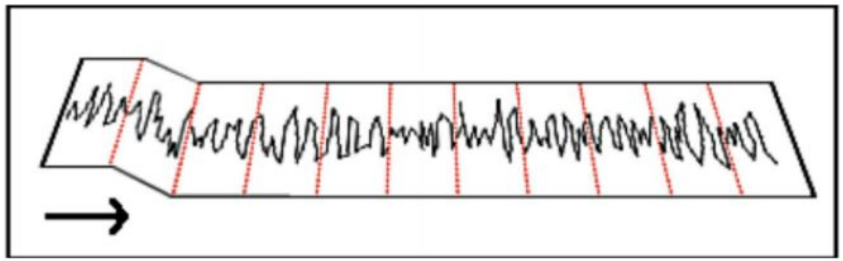
Pulsar Searching

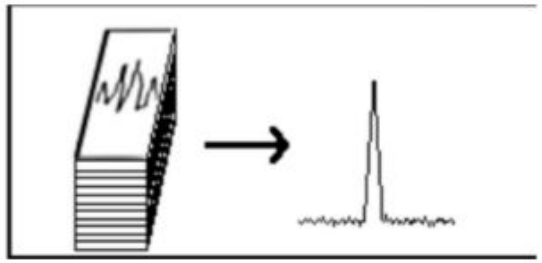
- Pulsar: Rotating Neutron star, regular EM radiation from poles
- Signal dispersion from interstellar medium, low SNR



Pulsar Searching

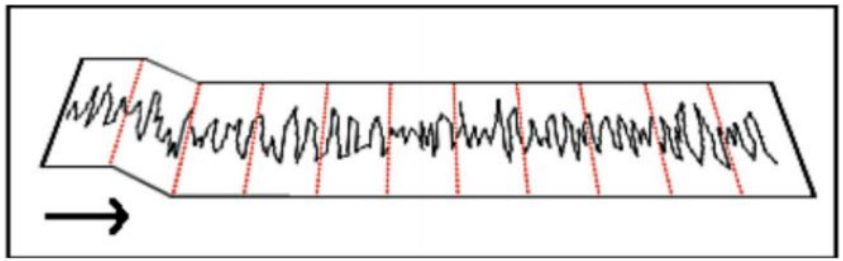
- Pulsar: Rotating Neutron star, regular EM radiation from poles
- Signal dispersion from interstellar medium, low SNR
- Detection steps (post beamforming)
  - De-dispersion
  - Frequency domain search
  - Folding

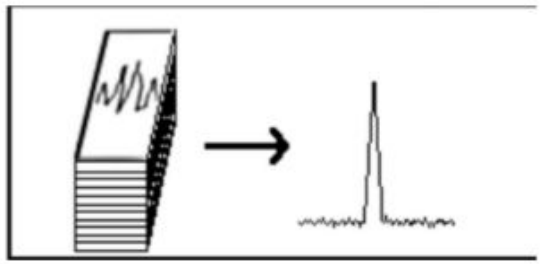




Pulsar Searching

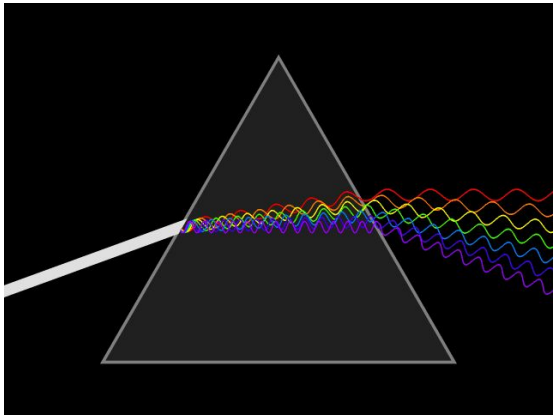
- Pulsar: Rotating Neutron star, regular EM radiation from poles
- Signal dispersion from interstellar medium, low SNR
- Detection steps (post beamforming)
  - **De-dispersion**
  - **Frequency domain search**
  - Folding

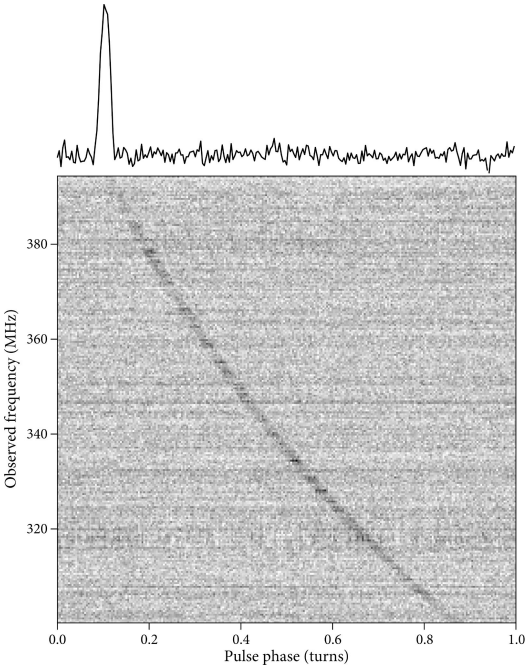




De-Dispersion

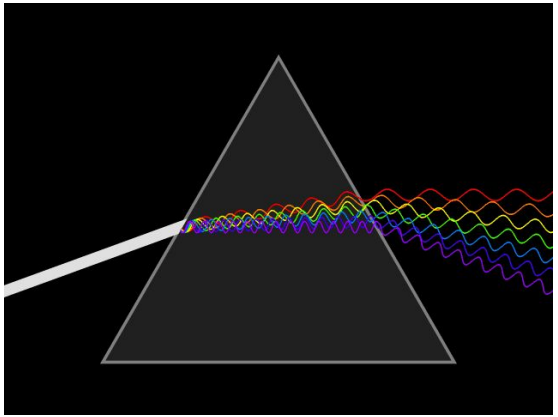
- Dispersion: frequency dependent delay from interstellar medium

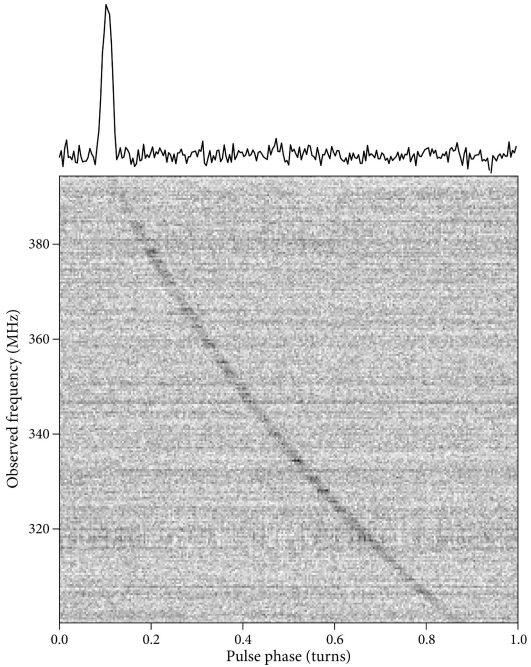




De-Dispersion

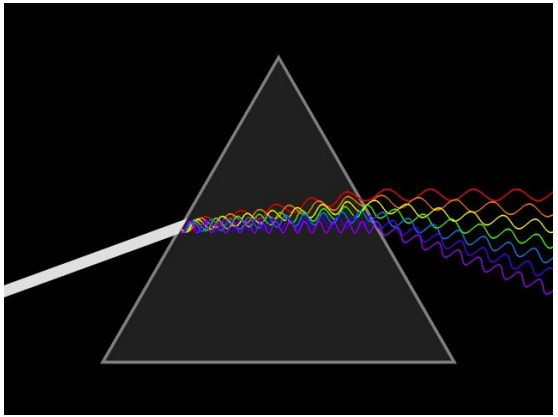
- Dispersion: frequency dependent delay from interstellar medium
- De-dispersion for trial DM values

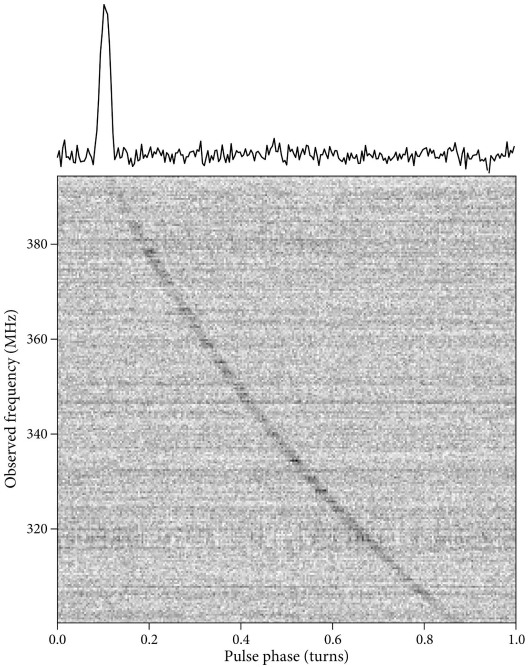




De-Dispersion

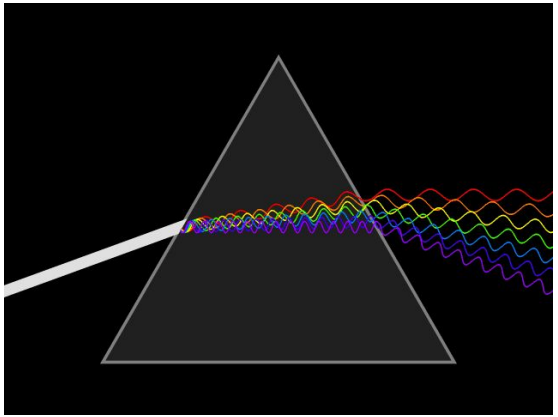
- Dispersion: frequency dependent delay from interstellar medium
- De-dispersion for trial DM values
  - **Time domain**: shifting time signals and adding
  - **Frequency domain**: shifting in fourier domain

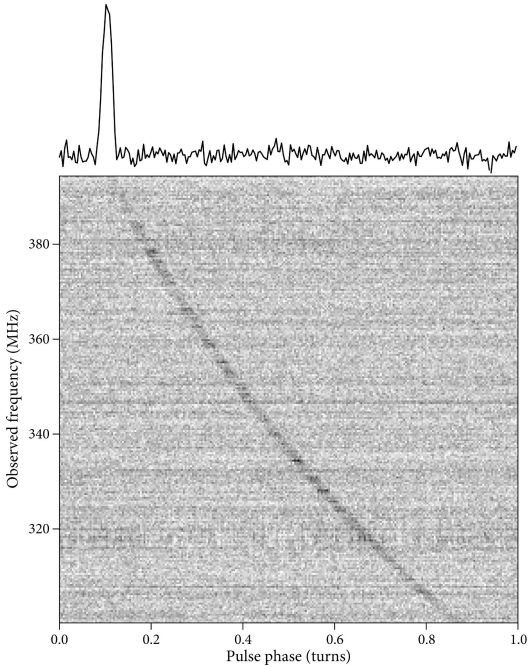




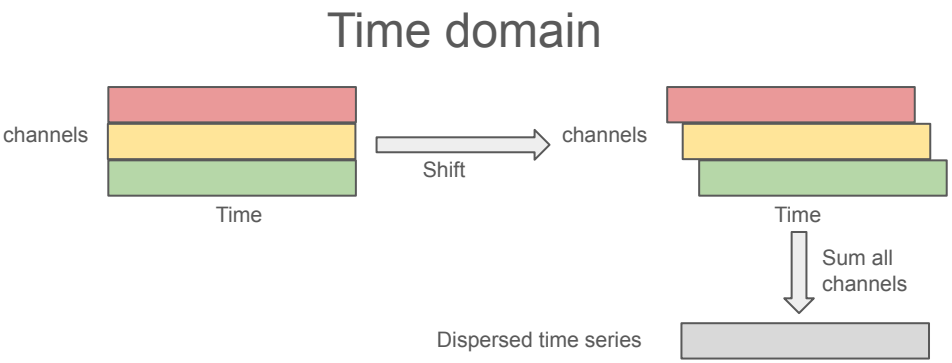
De-Dispersion

- Dispersion: frequency dependent delay from interstellar medium
- De-dispersion for trial DM values
  - **Time domain**: shifting time signals and adding
  - **Frequency domain**: shifting in fourier domain
- Full search on SMART dataset: ~15000 DM values



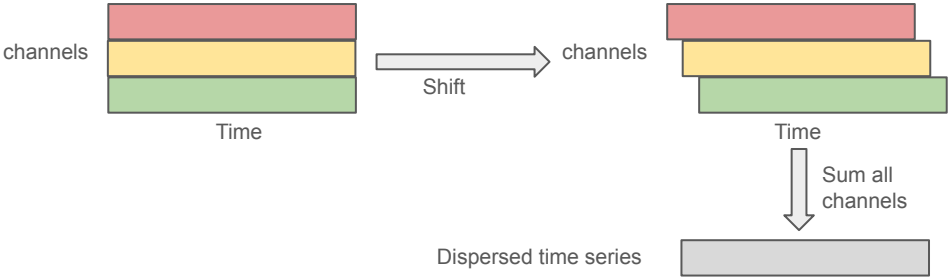


Time vs Frequency Domain De-Dispersion

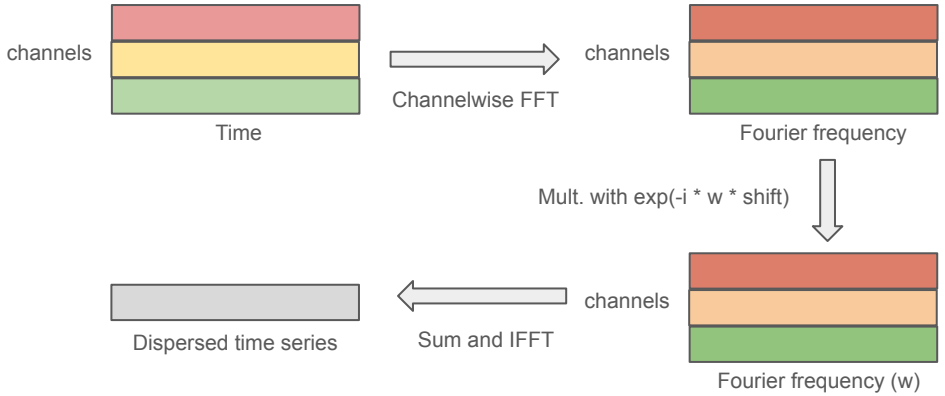


Time vs Frequency Domain De-Dispersion

Time domain

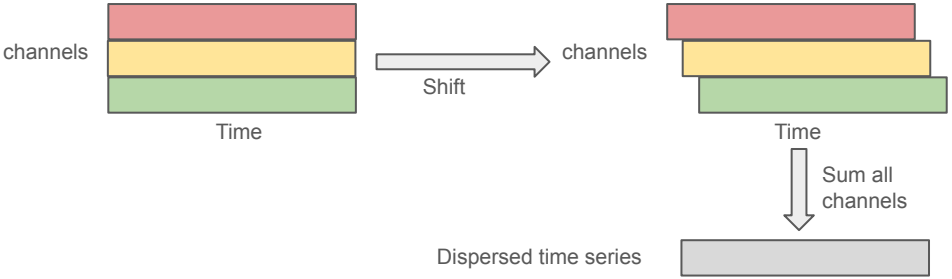


Frequency domain



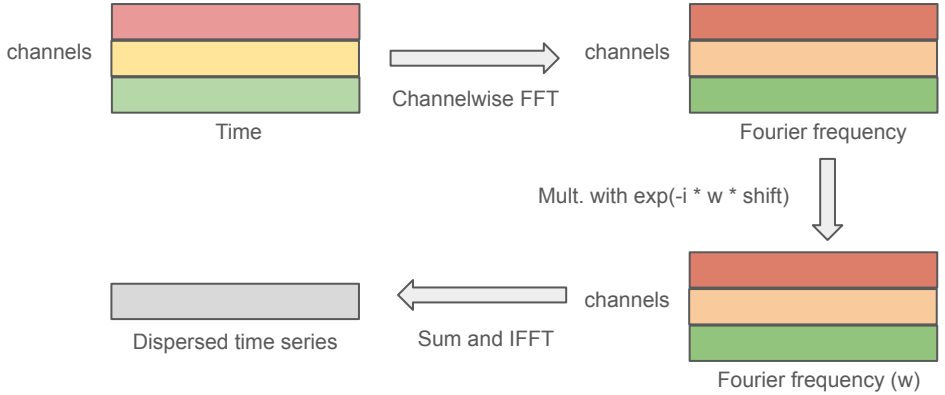
Time vs Frequency Domain De-Dispersion

Time domain



- Memory bound
- #DM FFTs

Frequency domain



- Compute bound
- #channels FFTs
- Can skip IFFT

GPU Implementation

Existing CUDA based implementation: dedisp (github.com/ajameson/dedisp)

GPU Implementation

Existing CUDA based implementation: dedisp (github.com/ajameson/dedisp)

My contributions to dedisp (github.com/piyushplcr7/dedisp_tests):

- FITS input (SMART data format), optimized reads, handling float input type
- Compatibility with newer CUDA version
- Bug fixes: failing CuFFT calls, incorrect Fourier freq., non integer delay
- Usable CLI similar to Presto

GPU Implementation

Existing CUDA based implementation: dedisp (github.com/ajameson/dedisp)

My contributions to dedisp (github.com/piyushplcr7/dedisp_tests):

- FITS input (SMART data format), optimized reads, handling float input type
- Compatibility with newer CUDA version
- Bug fixes: failing CuFFT calls, incorrect Fourier freq., non integer delay
- Usable CLI similar to Presto

Next steps:

- Porting to AMD GPUs
- MPI + GPU implementation for full input (~589 GB)
- Integration with the acceleration/jerk search, using FFT directly
- GPU specific optimizations

Preliminary Results

- Optimized dedisp vs Presto vs PrestoZL

Preliminary Results

- Optimized dedisp vs Presto vs PrestoZL
- No sub-bands, no barycentering. 2 Million points, 3072 channels.

Preliminary Results

- Optimized dedisp vs Presto vs PrestoZL
- No sub-bands, no barycentering. 2 Million points, 3072 channels.
- CPU: Ryzen 7 7800X3D, GPU: RTX 4070 Ti Super, Samsung 970 Evo Plus

	Dedisp (FDD, GPU)	PrestoZL (TDD, GPU)	Presto (TDD, CPU)
1000 DMs	28.95 s	> 139.77 s (800 DM)	8 hrs+
500 DMs	23.29 s	98 s	4.75 hrs

Preliminary Results

- Optimized dedisp vs Presto vs PrestoZL
- No sub-bands, no barycentering. 2 Million points, 3072 channels.
- CPU: Ryzen 7 7800X3D, GPU: RTX 4070 Ti Super, Samsung 970 Evo Plus

	Dedisp (FDD, GPU)	PrestoZL (TDD, GPU)	Presto (TDD, CPU)
1000 DMs	28.95 s	> 139.77 s (800 DM)	8 hrs+
500 DMs	23.29 s	98 s	4.75 hrs

- Kuma (H100): 15000 DMs, 116.223 s!

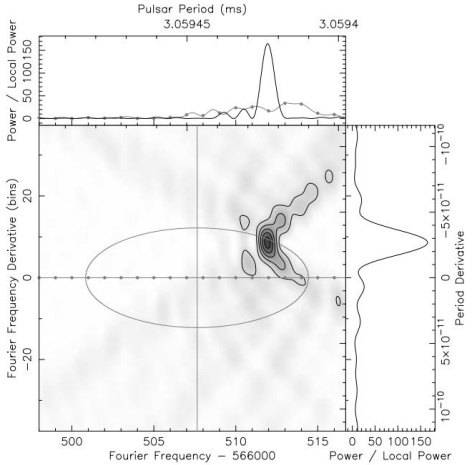
Frequency Domain Search

- **Periodicity search**: Peaks in frequency spectrum

Frequency Domain Search

- **Periodicity search**: Peaks in frequency spectrum
- **Acceleration/jerk search**: matched filtering in Fourier domain with templates

$$T^{i,j,k} := \mathcal{F}\Big(\cos(2\pi(r_i t + \frac{1}{2} z_j t^2 + \frac{1}{6} w_k t^3)) \Big)$$

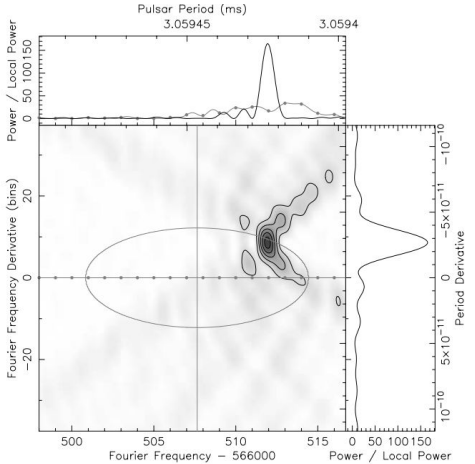


Frequency Domain Search

- **Periodicity search**: Peaks in frequency spectrum
- **Acceleration/jerk search**: matched filtering in Fourier domain with templates

$$T^{i,j,k} := \mathcal{F}\Big(\cos(2\pi(r_i t + \frac{1}{2} z_j t^2 + \frac{1}{6} w_k t^3)) \Big)$$

- Acceleration/jerk search capture effects from orbital motion of Pulsar

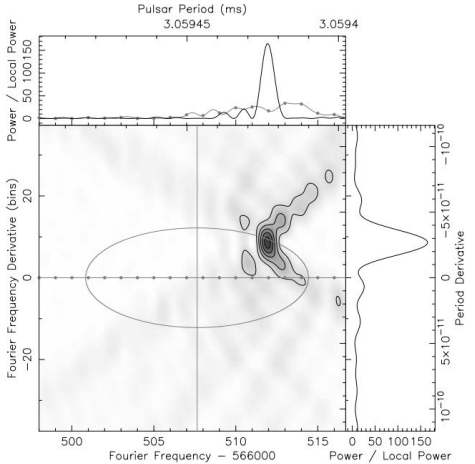


Frequency Domain Search

- **Periodicity search**: Peaks in frequency spectrum
- **Acceleration/jerk search**: matched filtering in Fourier domain with templates

$$T^{i,j,k} := \mathcal{F}\left(\cos(2\pi(r_i t + \frac{1}{2} z_j t^2 + \frac{1}{6} w_k t^3)) \right)$$

- Acceleration/jerk search capture effects from orbital motion of Pulsar
- Parallelizable algorithm, same filters for all de-dispersed time series
- Full search on SMART dataset: ~10000 filters



GPU Implementation

Existing GPU implementations (CUDA)

- jintaoluo/presto2_on_gpu : 10+ yrs old code (No jerk search)
- chrislaidler/presto : 6+ yrs old code (No jerk search)
- Astro-accelerate : Integrated code, can’t handle low frequencies efficiently
- PrestoZL : Recent, no FDD only TDD. Jerk search implemented

GPU Implementation

Existing GPU implementations (CUDA)

- jintaoluo/presto2_on_gpu : 10+ yrs old code (No jerk search)
- chrislaidler/presto : 6+ yrs old code (No jerk search)
- Astro-accelerate : Integrated code, can’t handle low frequencies efficiently
- PrestoZL : Recent, no FDD only TDD. Jerk search implemented

Progress:

- Extracted Jerk search from Astro-accelerate into a standalone executable

Next steps:

- Further optimizations in Astro-accelerate (naive copy of Presto)
- Exploring PrestoZL
- Porting to AMD GPUs

Preliminary Results

- Astro-accelerate vs Presto

Preliminary Results

- Astro-accelerate vs Presto
- Harmonic summing: false, interbinning: true. 2 Million samples

Preliminary Results

- Astro-accelerate vs Presto
- Harmonic summing: false, interbinning: true. 2 Million samples
- CPU: Ryzen 7 7800X3D, GPU: RTX 4070 Ti Super, Samsung 970 Evo Plus

	Presto (CPU)				Astro accelerate
kernels	1	2	4	8	GPU
33	0.178	0.137	0.1	0.105	0.304
357	1.318	0.766	0.431	0.268	0.492
1111	4.014	2.272	1.192	0.859	1.233
4221	19.217	9.808	5.501	3.023	4.102
25551	205.56	112.495	68.914	47.432	18.04

Questions?