Lecture 8 - AstroAccelerate

GPU accelerated signal processing for next generation radio telescopes

Wes Armour, Karel Adamek, Cees Carels, Kate Clark, Sofia Dimoudi, Mike Giles, Jan Novotny, Nassim Ouannough, Scott Ransom, Jayanta Roy, Jack White

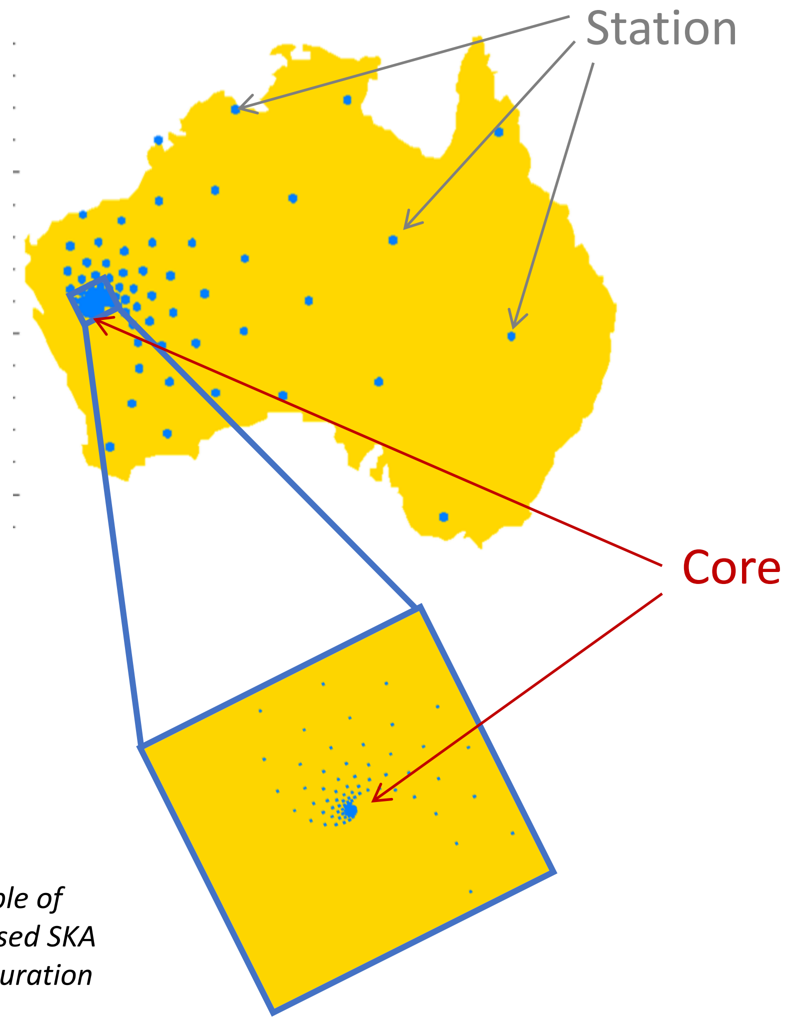
Part One

SKA

SQUARE KILOMETRE ARRAY

A brief introduction to

What is SKA?



Example of proposed SKA configuration

What is SKA?

*SKA is a **ground based radio telescope** that will span continents.*

What does SKA stand for?

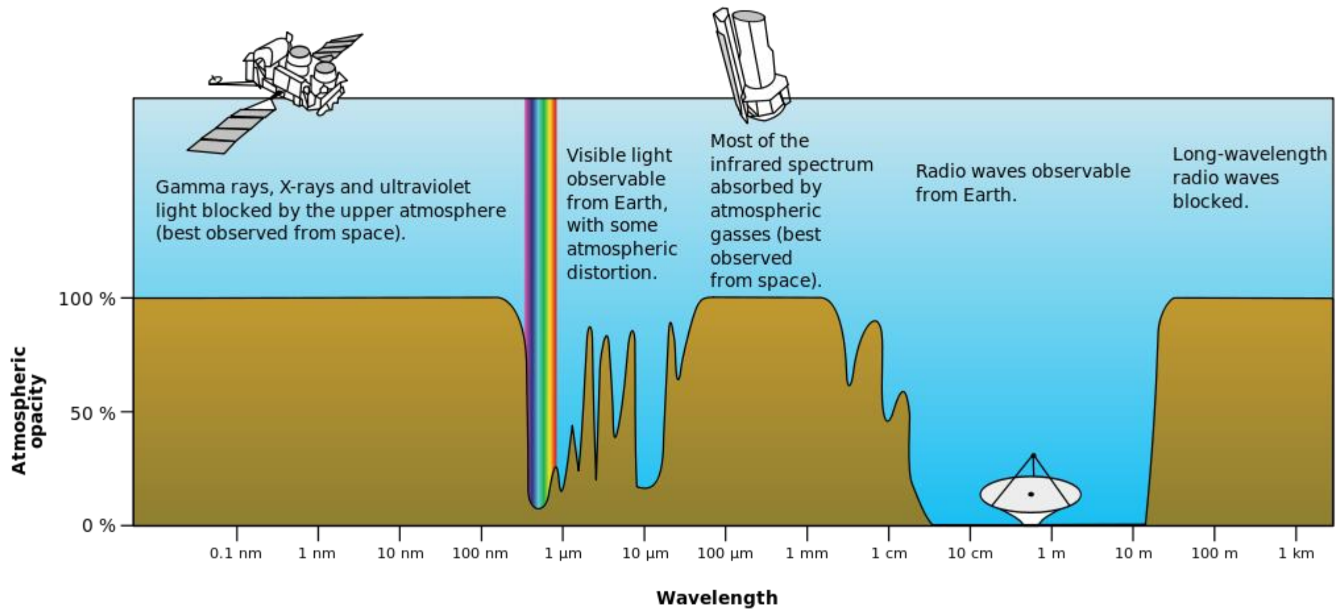
*Square Kilometre Array, so called because it will have an **effective collecting area of a square kilometre**.*

Where will SKA be located?

*SKA will be built in **South Africa and Australia.***

Graphic courtesy of Anne Trefethen

What is SKA?



SKA is a ground based telescope. This means that it is most sensitive to the radio range of frequencies. The radio range of frequencies that can be observed from here on Earth is very wide, specifically SKA will be sensitive to frequencies in the range of 50MHz to 20GHz (**wavelengths 15 mm to 6 m**). This makes SKA ideal for **studying lots of different science cases.**

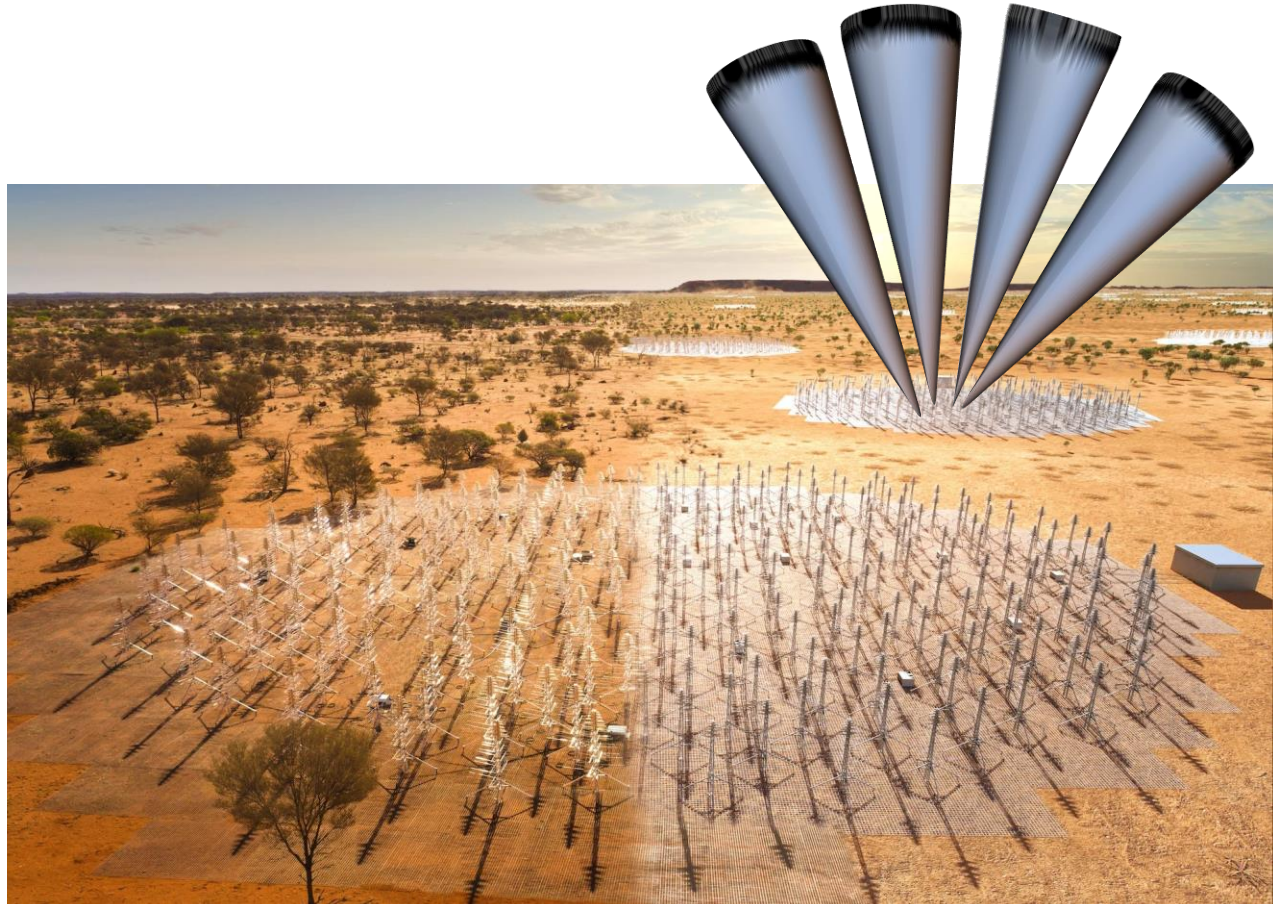
Image source Wikipedia. Authors: NASA (original); SVG by Mysid

What is SKA?

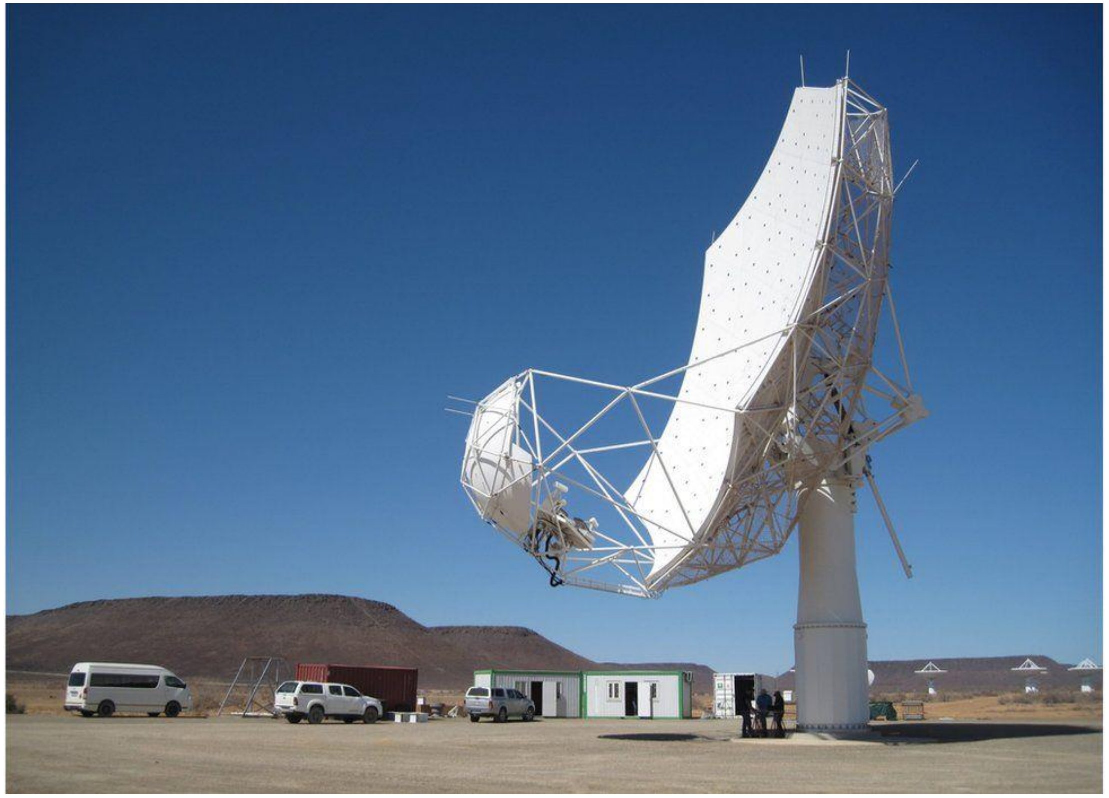
SKA will have the ability to use all of its antennas to **produce images of the radio sky in unprecedented accuracy and detail.**

It will also be able to use combinations of antennas to perform **multiple observations of different regions of the sky at the same time**.

In this scenario data from each beam can be computed in parallel.



SKA science



SKA will study a wide range of science cases and aims to answer some of the fundamental questions mankind has about the universe we live in.

- How do galaxies evolve
  - What is dark energy?
- Tests of General Relativity
  - Was Einstein correct?
- Probing the cosmic dawn
  - How did stars form?
- The cradle of life
  - Are we alone in the Universe?

SKA time domain - signal processing

UNIVERSITY OF OXFORD

MANCHESTER 1824 The University of Manchester

INAF ISTITUTO NAZIONALE DI ASTROFISICA NATIONAL INSTITUTE FOR ASTROPHYSICS

AUT UNIVERSITY AUCKLAND, NEW ZEALAND

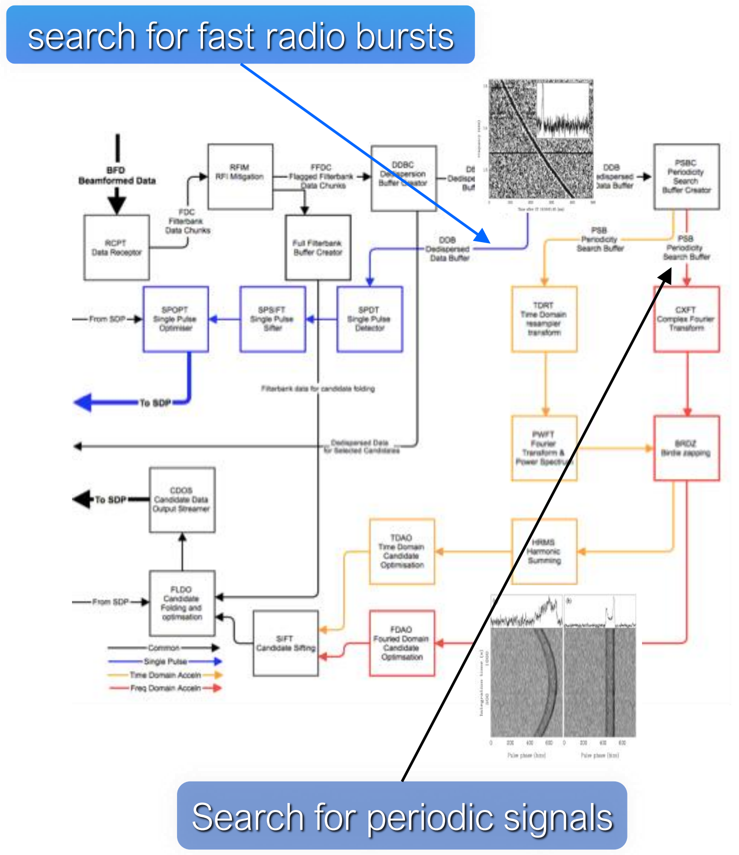
ASTRON

Max-Planck-Institut für Radioastronomie

Time Domain Team

The time domain team is an international team led by Oxford and Manchester.

It aims to deliver an end-to-end signal processing pipeline for time domain science performed by SKA (see right).



search for fast radio bursts
BFD Beamformed Data
RCPT Data Receptor
FDC Filterbank Data Chunks
RFIM RFI Mitigation
FFDC Flagged Filterbank Data Chunks
Full Filterbank Buffer Creator
DDBC Dedispersion Buffer Creator
DDB Dedispersed Data Buffer
PSBC Periodicity Search Buffer Creator
PSB Periodicity Search Buffer
SPOPT Single Pulse Optimiser
SPSIFT Single Pulse Sifter
SPDT Single Pulse Detector
From SDP
To SDP
Filterbank data for candidate folding
Dedispersed Data for Selected Candidates
TDRT Time Domain resampler transform
CXFT Complex Fourier Transform
PWFT Fourier Transform & Power Spectrum
BRDZ Birdie zapping
HRMS Harmonic Summing
TDAO Time Domain Candidate Optimisation
FDAO Fourier Domain Candidate Optimisation
CDOS Candidate Data Output Streamer
FLDO Candidate Folding and optimisation
SIFT Candidate Sifting
Common
Single Pulse
Time Domain Accel'n
Freq Domain Accel'n
Pulse phase (bins)
Search for periodic signals

Image courtesy of Aris Karastergiou

SKA time domain - signal processing

Our work focussed on vertical prototyping activities.

We delivered accelerated algorithms for many-core technologies, such as GPUs to perform the processing steps within the signal processing pipeline with the aim of achieving real-time processing for the SKA.

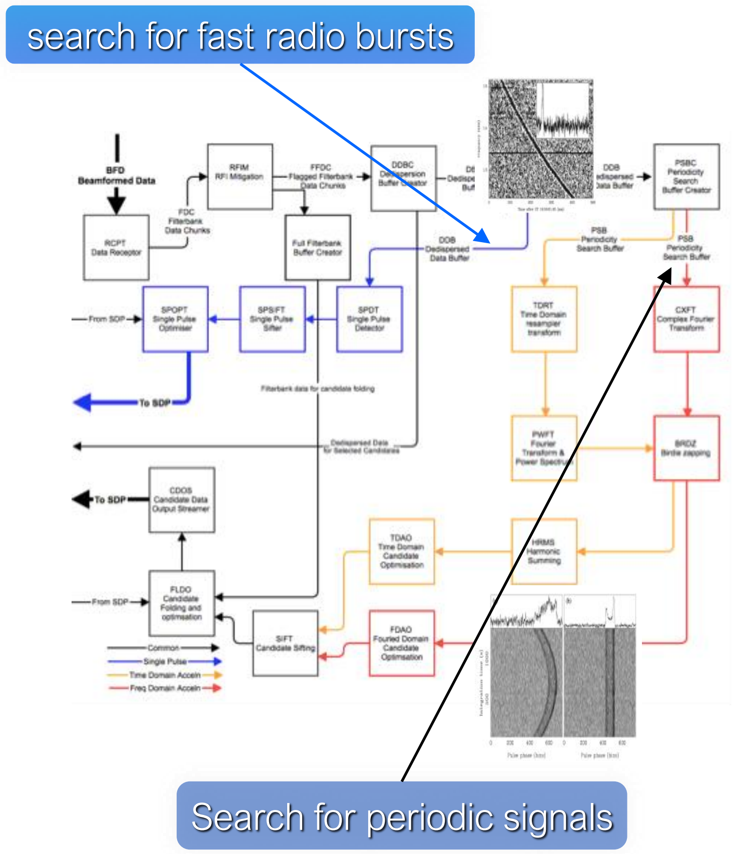

search for fast radio bursts
BFD Beamformed Data
RCPT Data Receptor
FDC Filterbank Data Chunks
RFIM RFI Mitigation
FFDC Flagged Filterbank Data Chunks
DDBC Dedispersion Buffer Creator
Full Filterbank Buffer Creator
DDB Dedispersed Data Buffer
PSBC Periodicity Search Buffer Creator
PSB Periodicity Search Buffer
From SDP
SPOPT Single Pulse Optimiser
SPSIFT Single Pulse Sifter
SPDT Single Pulse Detector
To SDP
Filterbank data for candidate folding
Dedispersed Data for Selected Candidates
TDRT Time Domain resampler transform
CXFT Complex Fourier Transform
PWFT Fourier Transform & Power Spectrum
BRDZ Birdie zapping
CDOS Candidate Data Output Streamer
TDAO Time Domain Candidate Optimisation
HRMS Harmonic Summing
FLDO Candidate Folding and optimisation
SIFT Candidate Sifting
FDAO Fouried Domain Candidate Optimisation
Common
Single Pulse
Time Domain Acceln
Freq Domain Acceln
Pulse phase (bins)
Search for periodic signals

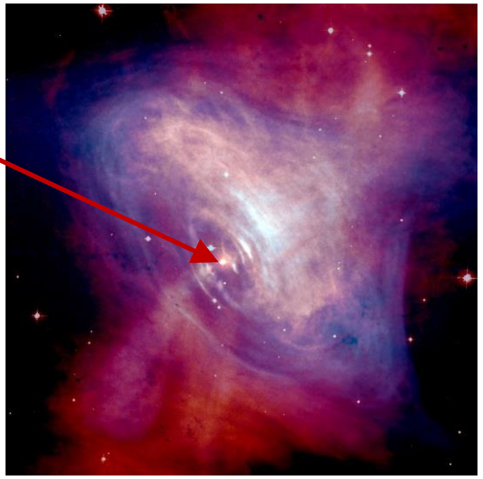
Image courtesy of Aris Karastergiou

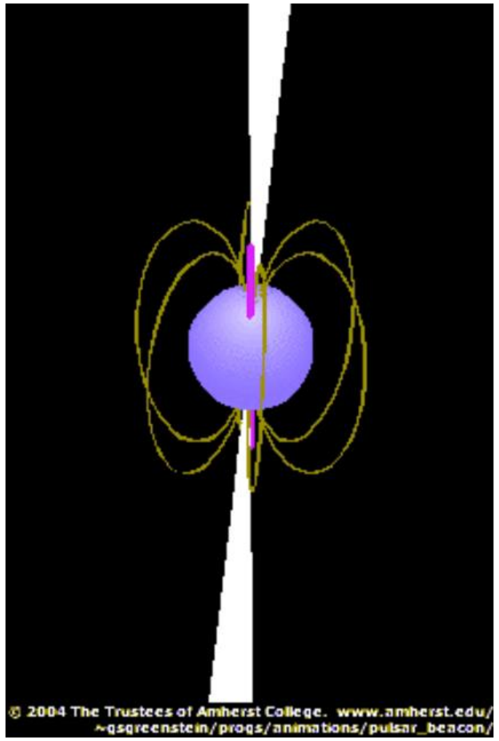
SKA time domain science - Pulsars

Pulsars are magnetized, rotating neutron stars. They emit synchrotron radiation from the poles, e.g. Crab Nebula.

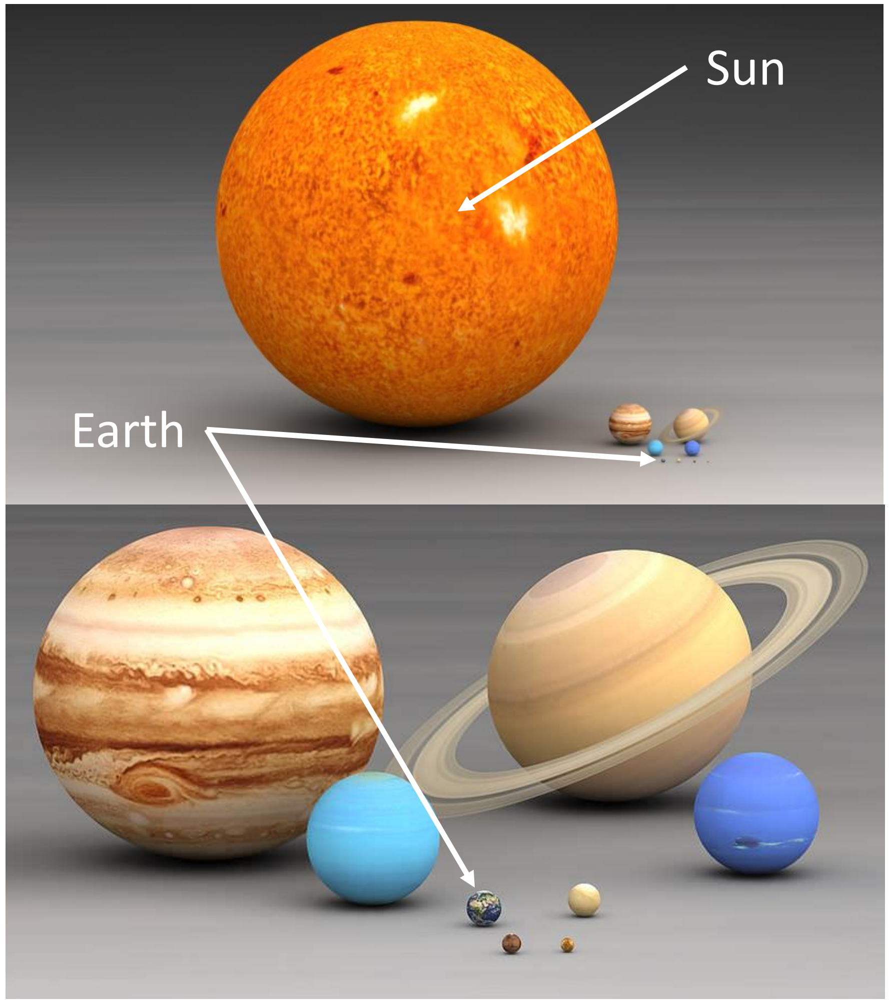
Their magnetic field is offset from the axis of rotation as such (as observed from here on Earth, they act as cosmic lighthouses.

They are extremely periodic and so make excellent clocks!


Hester et al.

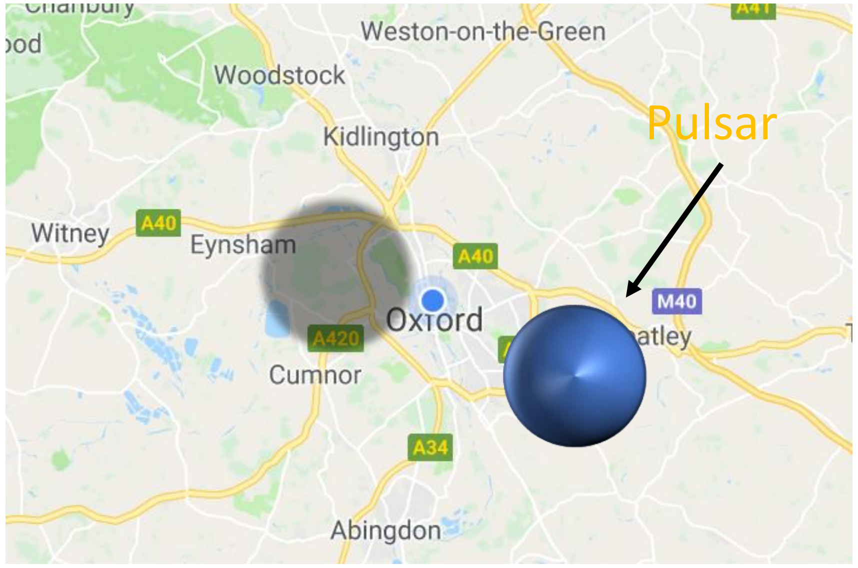

© 2004 The Trustees of Amherst College. www.amherst.edu/
~gsgreenstein/progs/animations/pulsar_beacon/
Image: Amherst College

Pulsars – size and scale



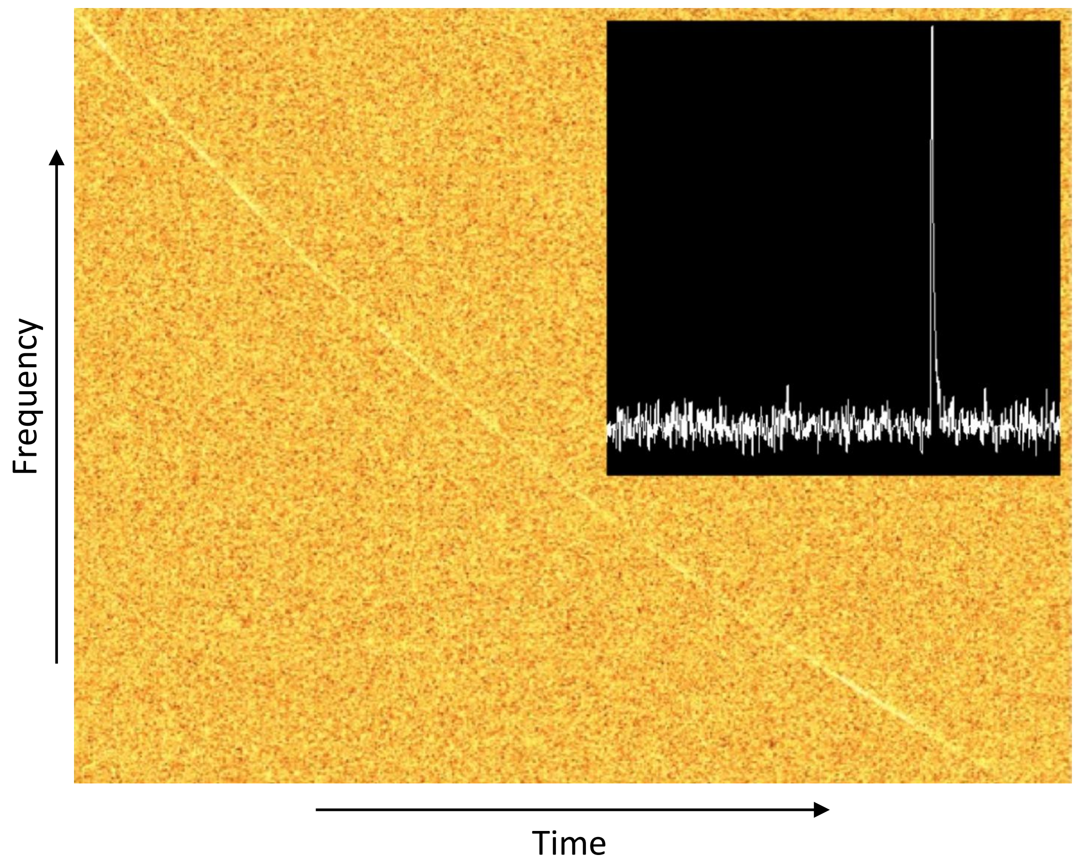
Pulsars are typically 1-3 Solar masses in size, they have a diameter of 10-20 Kilometres and a pulse period ranging from milliseconds to seconds.

Meaning that they are very small, very dense and rotate extremely quickly.



https://commons.wikimedia.org/wiki/File:Planets_and_sun_size_comparison.jpg (Author: Lsmpascal)

SKA time domain science - FRBs



Fast Radio Bursts (FRBs), were first discovered in 2005 by Lorimer et al.

They are observed as **extremely bright single pulses that are extremely dispersed** (meaning that they are likely to be far away, maybe extra galactic).

Now hundreds of FRBs have been observed or found in survey data. They are of unknown origin, but **likely to represent some of the most extreme physics in our Universe.**

Hence they are extremely interesting objects to study.

Credit: FRB110220 Dan Thornton (Manchester)

SKA time domain - data rates

The SKA will produce vast amounts of data. In the case of time-domain science we expect the telescope to be able to place ~2000 observing beams on the sky at any one time (there are trivially parallel to compute).

The telescope will take 20,000 samples per second for each of those beams and then it will measure power in 4096 frequency channels for each time sample. Each of those individual samples will comprise of 4x8 bits, although we are only really interested in one of the 8 bits of information.

Doing the math tells us that we will need to process 160GB/s of relevant data. This is approximately equal to analysing 50 hours of HD television data per second.

The most costly computational operations in data processing pipeline are

DDTR ~ $O(n_{dms} * n_{beams} * n_{samps} * n_{chans})$

FDAS ~ $O(n_{dms} * n_{beams} * n_{samps} * n_{acc} * \log(n_{samps}) * 1/t_{obs})$

Requiring ~2 PetaFLOP of Compute!

SKA time domain – data challenges

Because we would like to **monitor interesting and exotic events as they occur we need to process data in real-time** (or as near to as possible).

So storing the data and processing later isn't feasible. The data rates mean transporting data offsite would be challenging and costly.

So processing must happen close to the telescope. But how do we put a computer capable of processing big-data streams in real-time close to the telescope?

Connectivity, power, operation all pose significant problems.

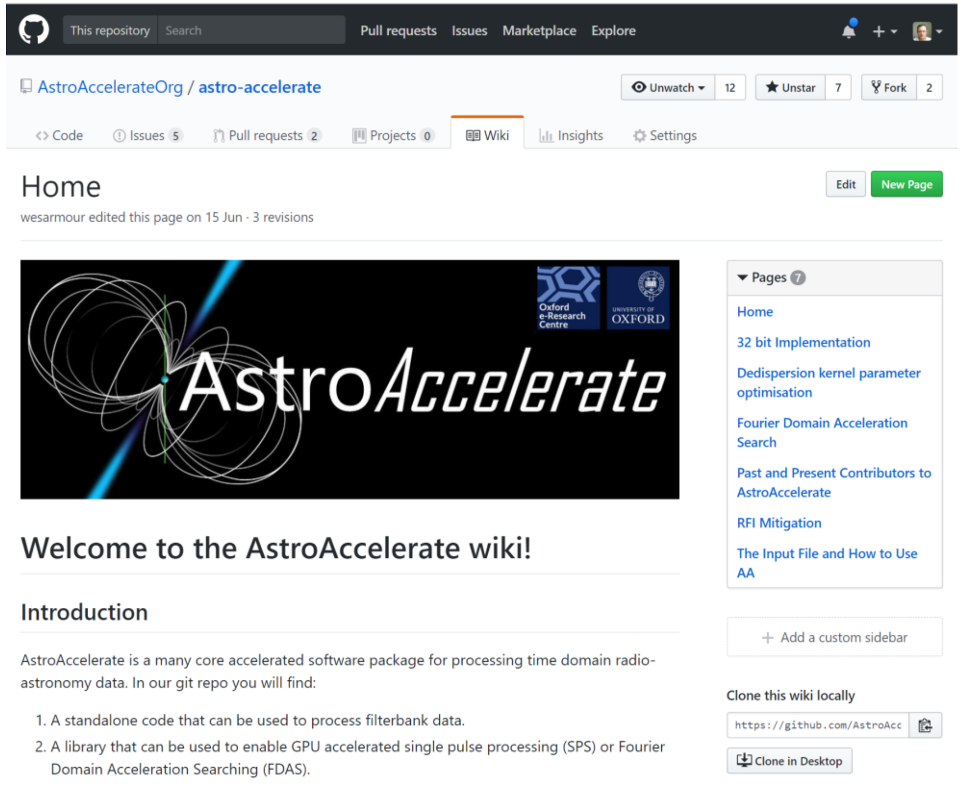
Part Two

Astro*Accelerate* – A case study

AstroAccelerate

AstroAccelerate is a GPU enabled software package that focuses on achieving real-time processing of time-domain radio-astronomy data. It uses the CUDA programming language for NVIDIA GPUs.

The massive computational power of modern-day GPUs allows the code to perform algorithms such as de-dispersion, single pulse searching and Fourier Domain Acceleration Searching in real-time on very large data-sets which are comparable to those which will be produced by next generation radio-telescopes such as the SKA.



Home

wesarmour edited this page on 15 Jun · 3 revisions

Welcome to the AstroAccelerate wiki!

Introduction

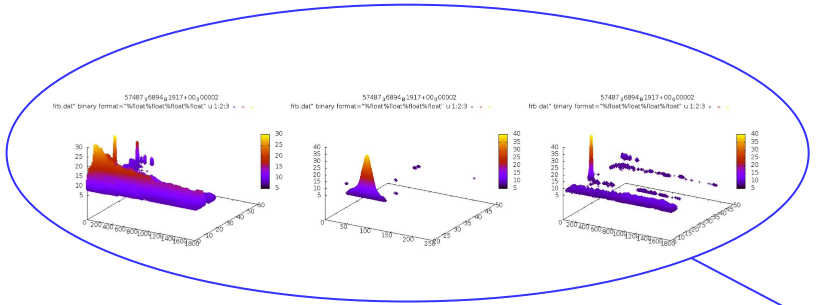
AstroAccelerate is a many core accelerated software package for processing time domain radio-astronomy data. In our git repo you will find:

1. A standalone code that can be used to process filterbank data.
2. A library that can be used to enable GPU accelerated single pulse processing (SPS) or Fourier Domain Acceleration Searching (FDAS).

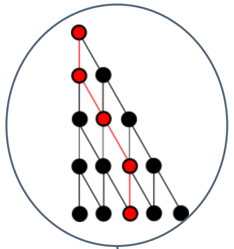
https://github.com/AstroAccelerateOrg/astro-accelerate

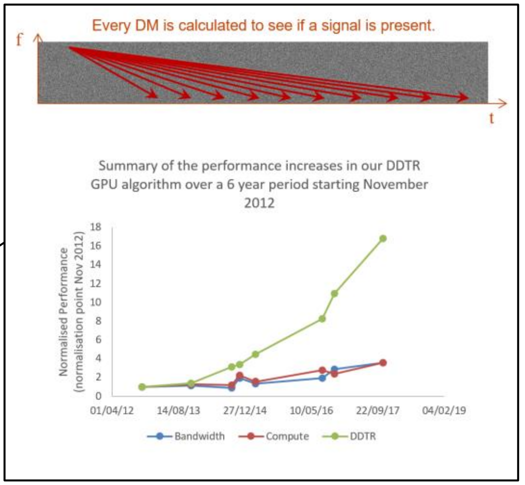
AstroAccelerate - Signal Processing

Radio Frequency Interference Mitigation



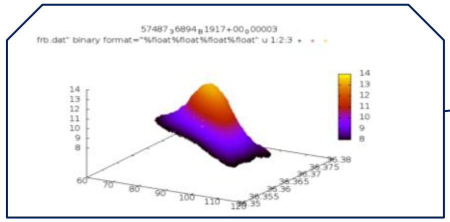
Harmonic Sum

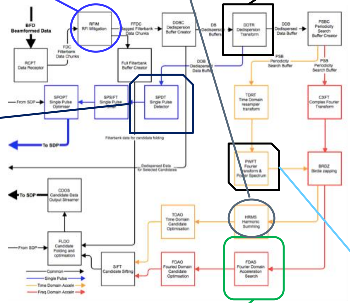



Every DM is calculated to see if a signal is present.
f
t
Summary of the performance increases in our DDTR GPU algorithm over a 6 year period starting November 2012
Normalised Performance (normalisation point Nov 2012)
01/04/12
14/08/13
27/12/14
10/05/16
22/09/17
04/02/19
Bandwidth
Compute
DDTR

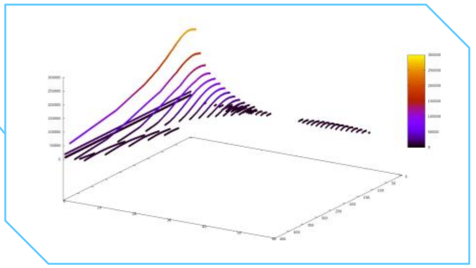
De-dispersion

Single Pulse Search

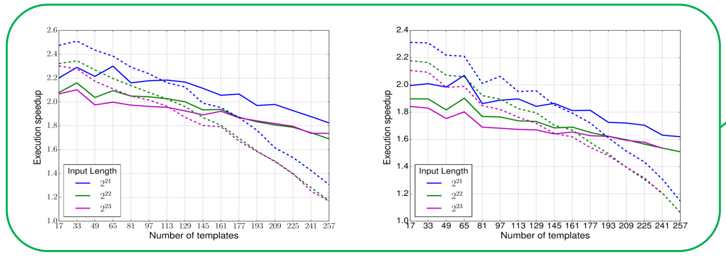




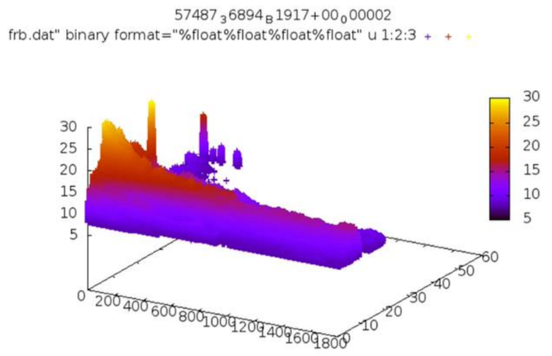
Periodicity Search

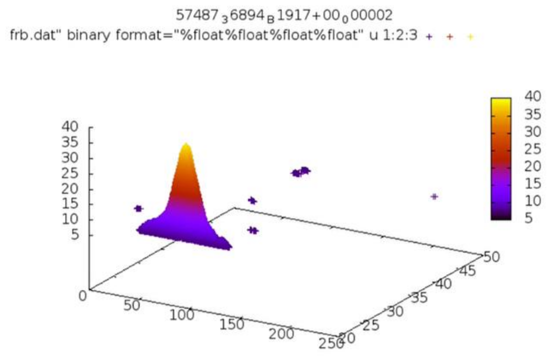


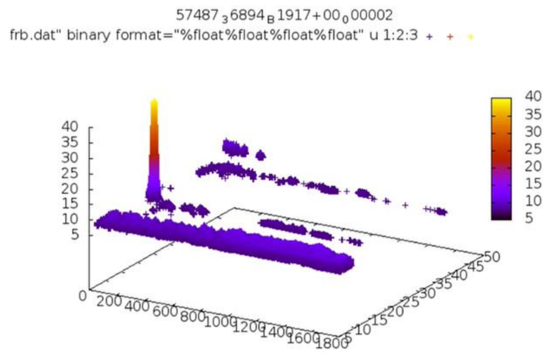
Fourier Domain Acceleration search



AstroAccelerate - Features

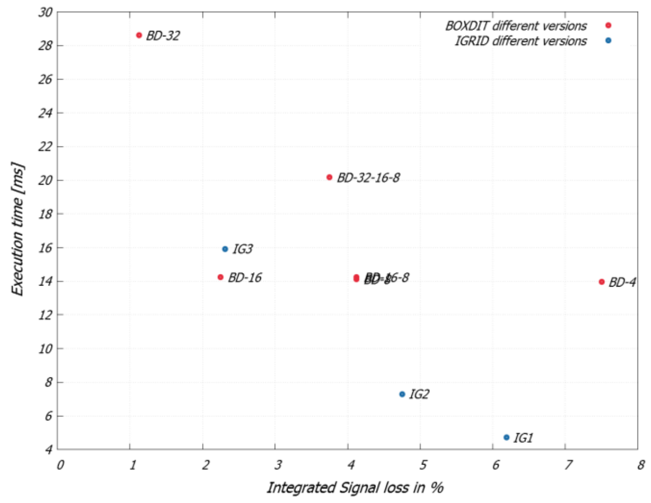






AstroAccelerate has the following features…

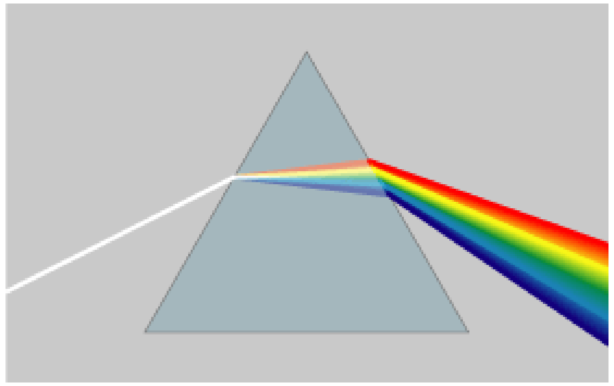
- Zero DM and basic RFI Mitigation
- DDTR
- Single Pulse Search
- Fourier Domain Acceleration Search
- Periodicity search with harmonic sum



Case study 1: Real-time de-dispersion for the SKA

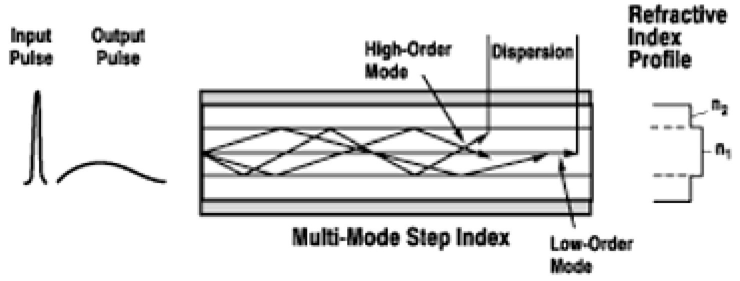
Mike Giles (Oxford)
Jan Novotný (Oxford)
Karel Adámek (Oxford)
Kate Clark (NVIDIA)
Tom Bradley (NVIDIA)
Tim Lanfear (NVIDIA)

What is dispersion?



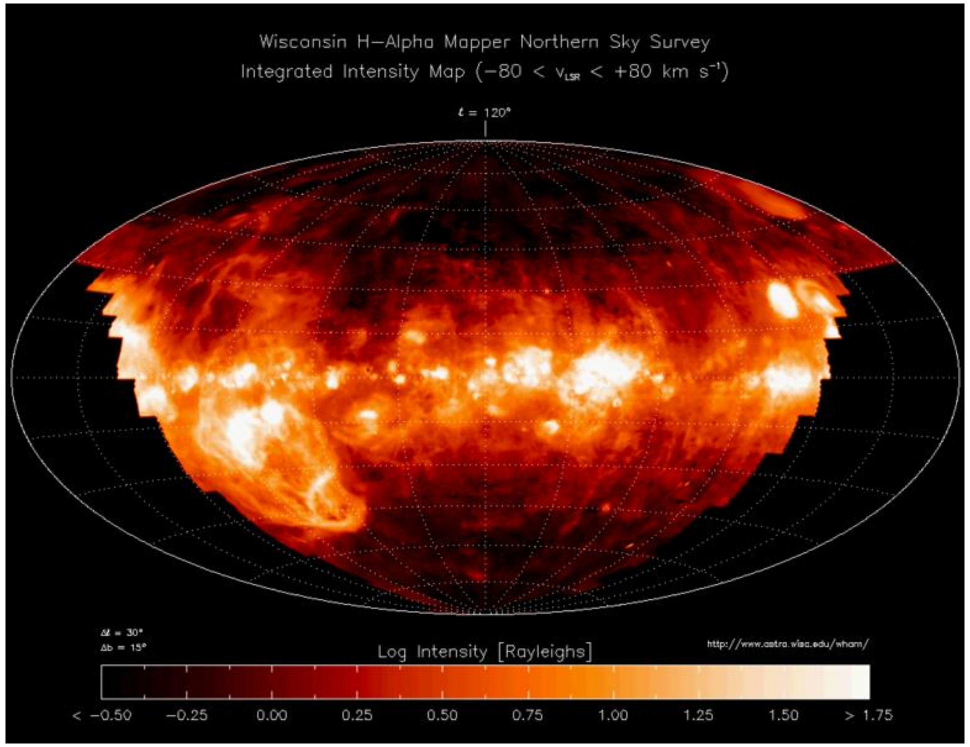
Group velocity dispersion occurs when pulse of light is spread in time due to its different frequency components travelling at different velocities. An example of this is when a pulse of light travels along an optical fibre.

Chromatic dispersion is something we are all familiar with. A good example of this is when white light passes through a prism.



Dispersion by the ISM

The interstellar medium (ISM) is the matter that exists between stars in a galaxy.


Wisconsin H-Alpha Mapper Northern Sky Survey
Integrated Intensity Map (−80 < v_{LSR} < +80 km s^{-1})
ℓ = 120°
Δℓ = 30°
Δb = 15°
Log Intensity [Rayleighs]
http://www.astro.wisc.edu/wham/
< −0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50
> 1.75

Haffner et al. 2003

In warm regions of the ISM (~8000K) electrons are free and so can interact with and affect radio waves that pass through it.

The dispersion measure - DM

The time delay, $\Delta\tau$, between the detection of frequency f_{high} and f_{low} is given by:

$$\Delta\tau = \mathrm{C}_{DM} \times \mathrm{DM} \times \left(\frac{1}{f_{low}^2} - \frac{1}{f_{high}^2}\right)$$

Where C_{DM} is the dispersion constant. DM is the dispersion measure:

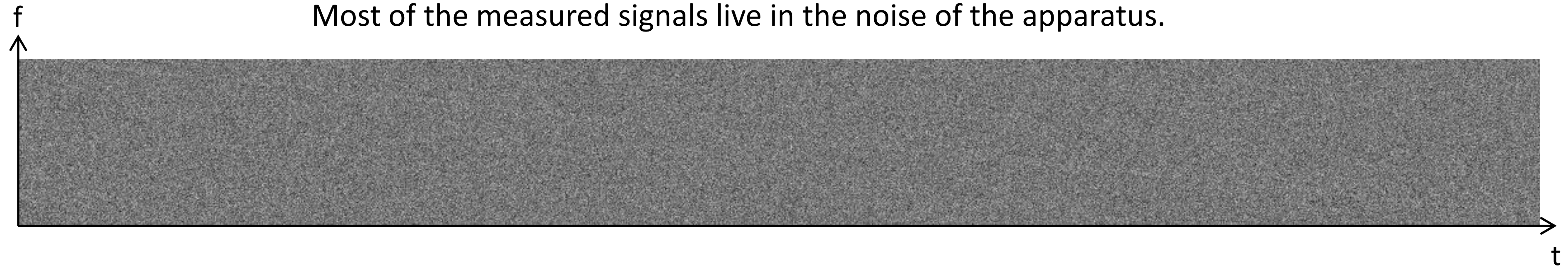
$$DM = \int_0^d n_e dl$$

This is the free electron column density between the radio source and observer.

We can measure $\Delta\tau$ and f and so can study DM

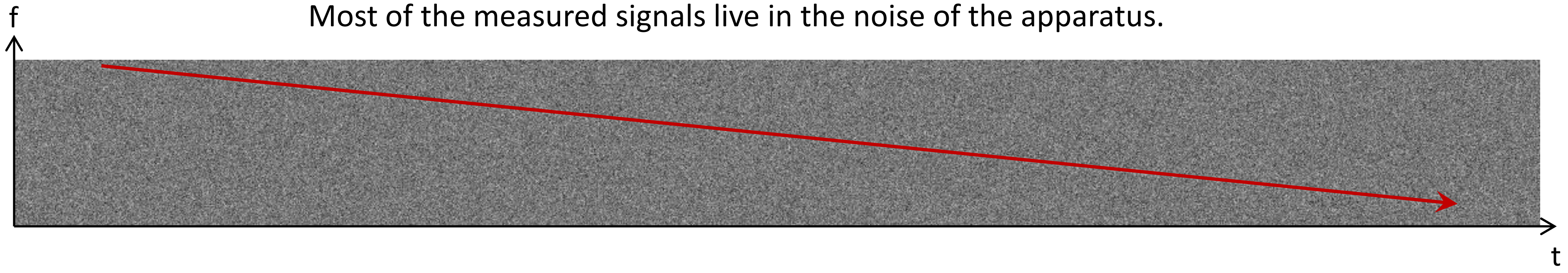
Experimental Data

Most of the measured signals live in the noise of the apparatus.

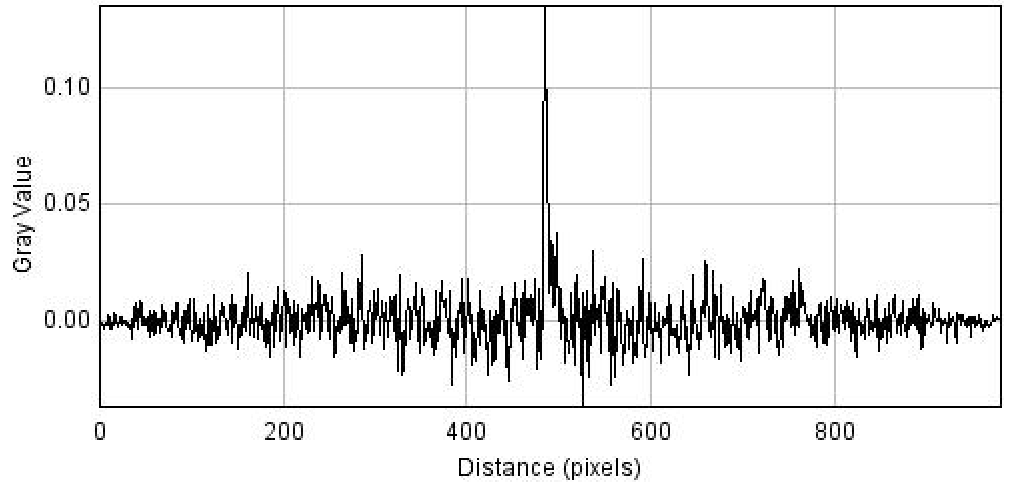


Experimental Data

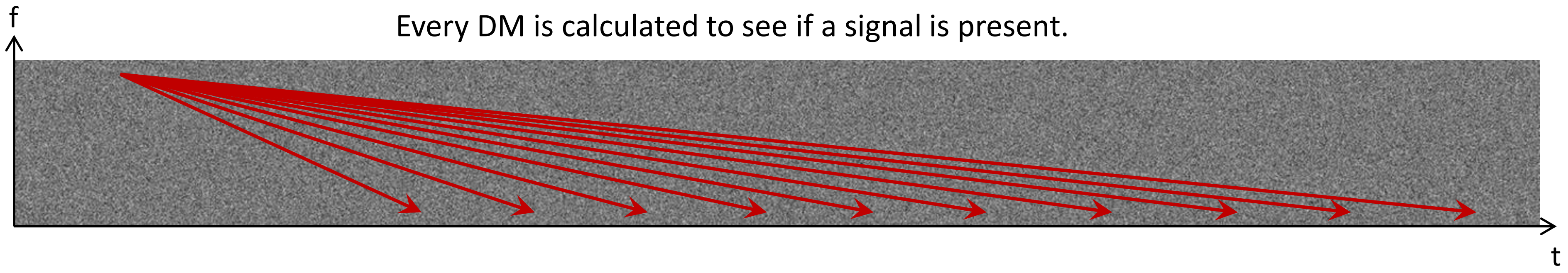
Most of the measured signals live in the noise of the apparatus.



Hence frequency channels have to be “folded”



De-dispersion



In a blind search for a signal many different dispersion measures are calculated.

This results in many data points in the (f,t) domain being used multiple times for different dispersion searches.

This allows for data reuse in a GPU algorithm.

All of this must happen in real-time i.e. the time taken to process all of our data must not exceed the time taken to collect it

De-dispersion Transform

Our DDTR is an implementation of incoherent brute force de-dispersion.

1. We brute force optimise the tuneable parameters of the code, such as the thread block size and number of registers used.

2. It utilises GPU shared memory and typically achieves 60-80% of peak throughput.

3. It uses SIMD in word to process multiple time samples per machine word for data less than or equal to 16 bits.

```
for (i = 0; i < SNUMREG; i++)
{
        local = 0;
        unroll = ( i * 2 * SDIVINT );
        for (j = 0; j < UNROLLS; j++)
        {
                stage = *(int*) &f_line[j][( shift[j] + unroll )];
                local += stage;
        }
        local_kernel_one[i] += (local & 0x0000FFFF);
        local_kernel_two[i] += (local & 0xFFFF0000) >> 16;
}
```

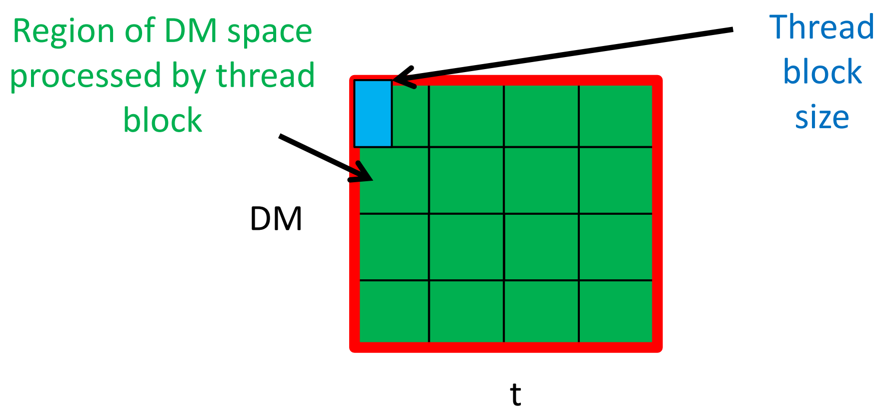
https://github.com/AstroAccelerateOrg/astro-accelerate/blob/master/lib/device_dedispersion_kernel.cu

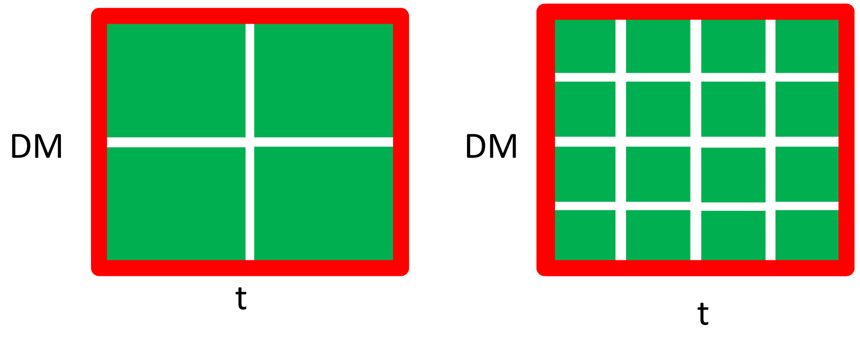
De-dispersion Transform – 1. tuning

Each thread processes a tunable number of time samples, each **de-dispersion trial associated with one time sample is stored in a GPU register**.

Along with this **the number of time samples per thread block and the number of de-dispersion trials (which is where data reuse comes from) are tuned.**

Finally the code performs **a tunable number of SIMD in word operations** which are periodically unloaded to a floating point accumulator.



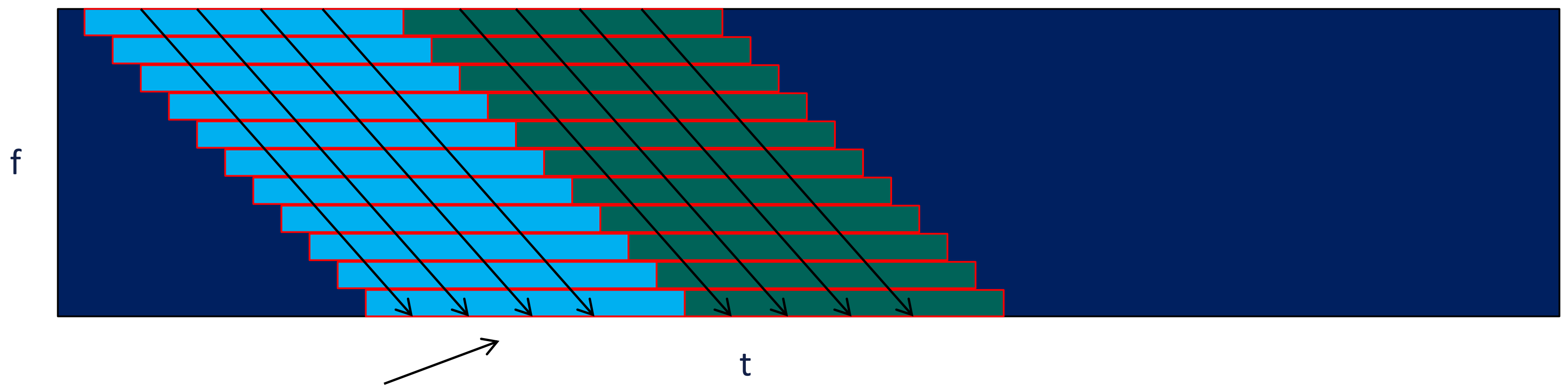


Optimising the parameterisation

De-dispersion Transform – 2. shared memory

Exploiting registers and fast shared memory…

Each dispersion measure for a given frequency channel needs a shifted time value.



Constant DM's with varying time.

In practice a thread will process multiple time samples and a thread block will also process neighboring DM trials to increase data reuse.

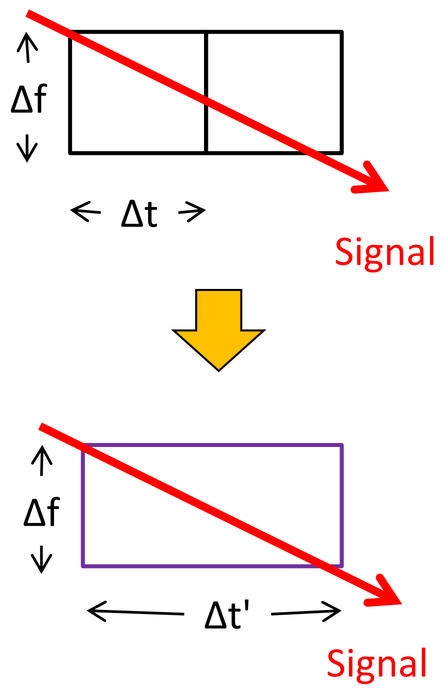
Incrementing all of the registers at every frequency step ensures a high data reuse of the stored frequency time data in the L1 cache or shared memory.

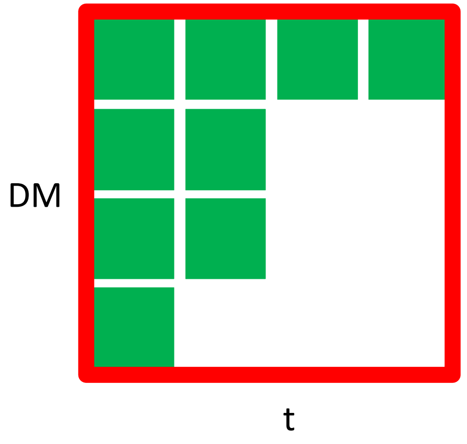
De-dispersion Transform – 2. time binning

One issue with using a shared memory based algorithm is that for high DM trials (those that represent distant objects, forming long broad curves in our input frequency-time data) **we need to store increasing lengths of constant frequency varying time data in shared memory.**

This ultimately limits the highest DM trial that can be searched at full time resolution.

To overcome this we've added a time binning (scrunching) kernel that decimates data in time. This has the effect of decreasing time resolution and allows us to search to arbitrary high DM trials.





Has the added advantage of reducing the amount threads that are needed to process a region of (DM,t) space, speeding up the code.

De-dispersion Transform – 3. SIMD in word

We exploit the fact that **one frequency-time sample of SKA data will be 8 bits**.

We **pack the data in such a way so that we can perform two de-dispersion trials per integer operation**.

We convert the **unsigned char to an unsigned short and pack as ushort2**, we **mask this as an int and add ints**.

Once a single trial nears the maximum allowable value for a ushort we store the value in a floating point accumulator. This has the effect of increasing the speed of the code and also it's precision.

Recorded telescope data (t_n = 8 bits) is stored in global as a uchar array

char[] = [t_0,t_1,t_2,t_3,t_4,t_5,t_6 …]

This is converted to ushort when loaded though the texture pipe (doubling the size of the array stored because it is now interleaved with 8 bits of zeros

ushort[] = [0 t_0, 0 t_1, 0 t_2, 0 t_3, 0 t_4, 0 t_5, 0 t_6, …]

Masking this with an int allows us to add two samples per one instruction issued.

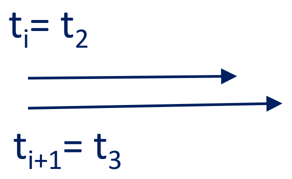
De-dispersion Transform – 3. SIMD in word

In reality we have to odd/even interleave the data to ensure correct byte alignment within shared memory banks (4 bytes wide).

ushort2[] = [0 t_0,0 t_1][0 t_1,0 t_2][0 t_2,0 t_3][0 t_3,0 t_4]…

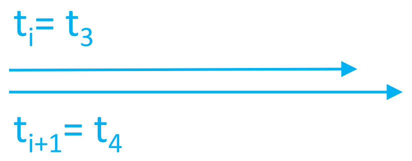
For thread with an even shift (lets say 2)…

(t_0,t_1) (t_1,t_2) (t_2,t_3) (t_3,t_4) (t_4,t_5) (t_5,t_6)



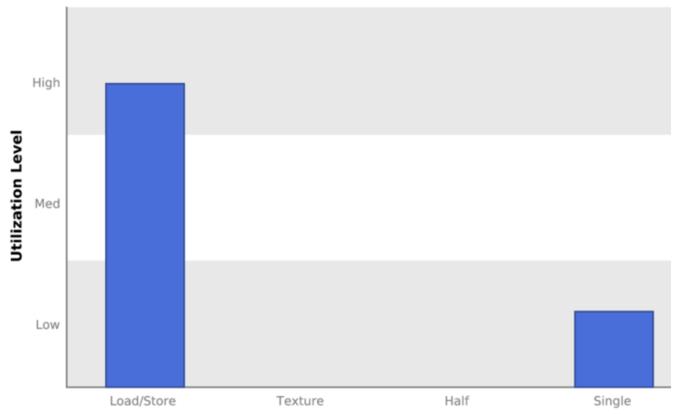
For thread with an odd shift (lets say 3)…

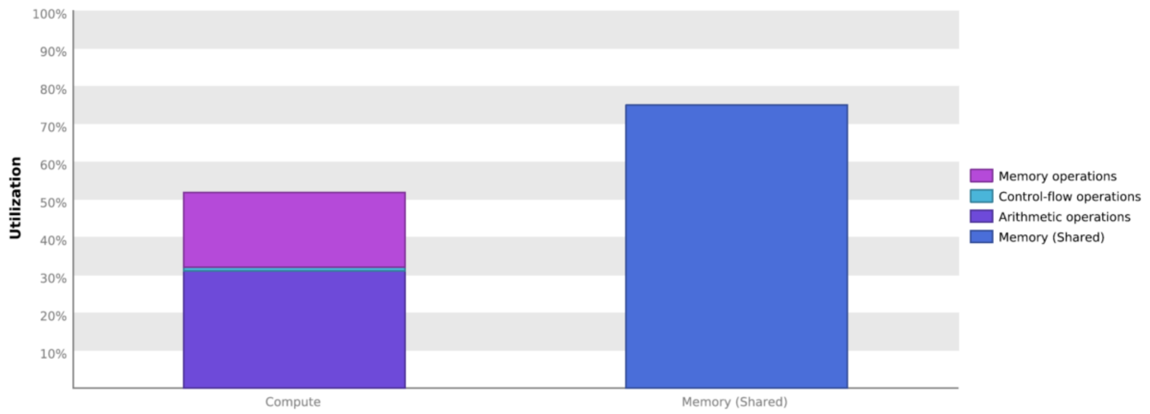
(t_0,t_1) (t_1,t_2) (t_2,t_3) (t_3,t_4) (t_4,t_5) (t_5,t_6)

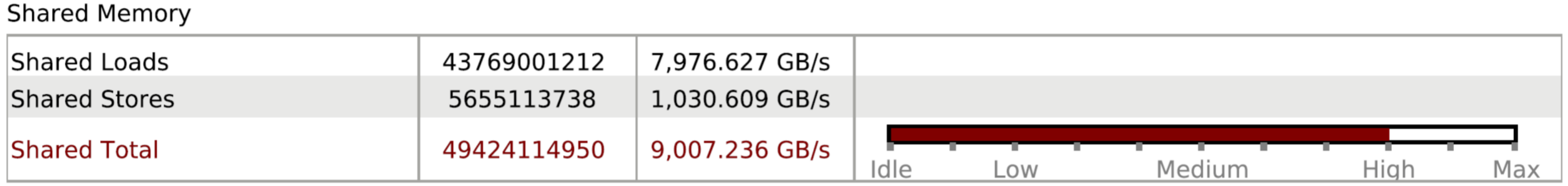


Now each thread computes the correct two time values and at double data rate

De-dispersion Transform - results





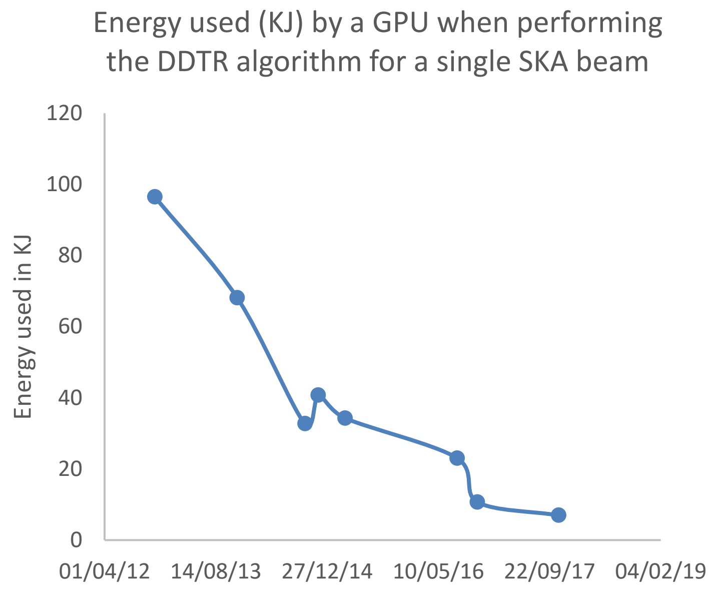

Shared Memory

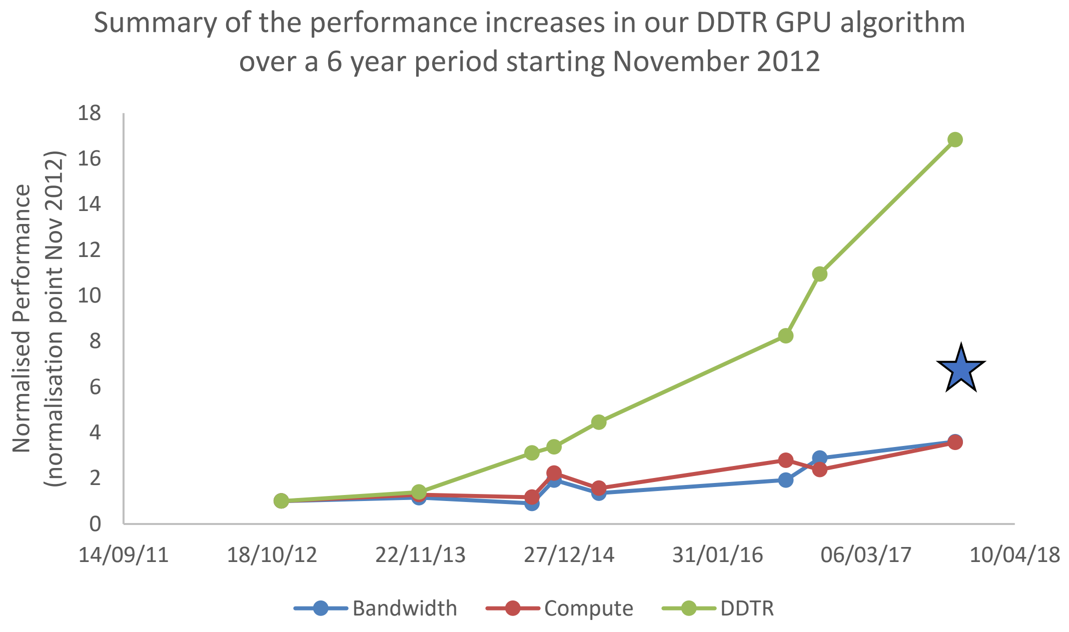
Shared Loads	43769001212	7,976.627 GB/s	
Shared Stores	5655113738	1,030.609 GB/s	
Shared Total	49424114950	9,007.236 GB/s	Idle Low Medium High Max

Results showing the shared memory utilisation, which is this codes limiting factor. **We achieve 75% of peak throughput**, limited by load/store.

The total shared memory bandwidth throughput achieved on a **TITAN V is 9 TB/s**.

De-dispersion Transform - results





These two plots demonstrate how we have reduced power consumption and increased performance for the DDTR algorithm over a six year period.

The bule star indicates the performance of our initial (optimised) code running on current hardware.
Demonstrating how invested effort algorithm optimisation over a long period can deliver significant gains.

De-dispersion Transform – cost / benefit analysis

But is it worth the effort?

Estimated runtime for DDTR in the PSS pipeline (conservative 25%)
Estimate of speed increase compared to initial code ~17x

➡ Total PSS pipeline acceleration ~ 4x

So to deliver the science in the same wall clock time you'd need 4x the GPU capacity.

Even if you're prepared to wait 4x longer… Energy efficiency has increased by 14x
Very rough estimate of PSS OpEx saving ~ £1M
Estimate of total effort ~ 1.0FTE for four years ~ £250K (FEC)

Hence a £750K saving in OpEx costs alone (this is a conservative estimate).

(You can't just go out and buy this at a later date. Domain expertise in both radio astronomy data processing and many-core acceleration are needed)

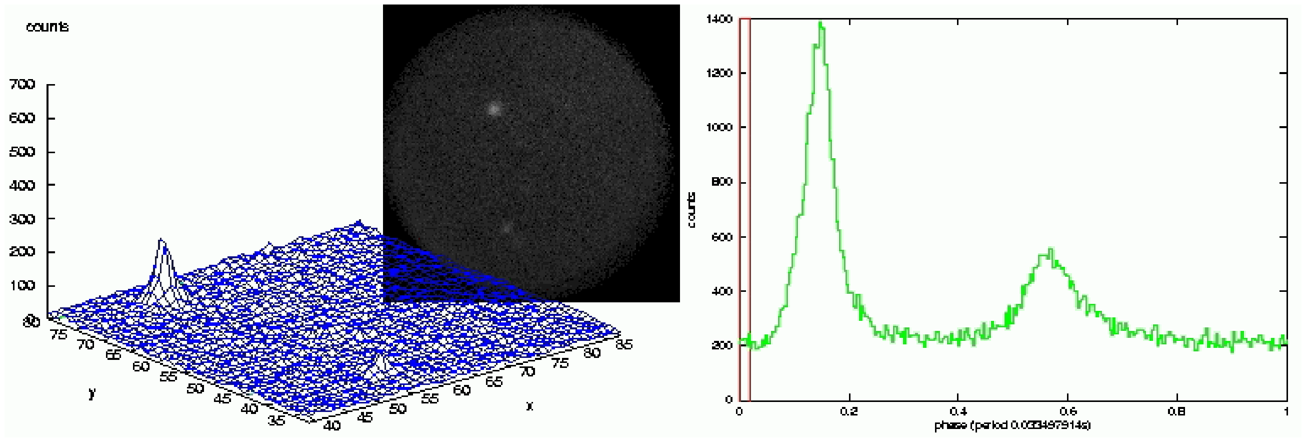
Conclusions – Comparisons of GPUs

Technology	Kepler (K40)	Kepler (K80)	Kepler (780Ti)	Maxwell (980)	Maxwell (Titan X)	Pascal (Titan XP)	Pascal P100	Volta V100	Volta Titan V
Fraction of real-time	1.035	2.5	2.88	2.3	3.3	6.1	8.1	12.5	10.9
Watts per beam (Average)	127W	76 W	~70W	~61W	~64W	~43W	~24W	13W	10W
Cost per beam (capital, accelerator only)	£3K?	£4K?	£250	£200	£240	~£200	~£420	~£530	~£270
Cost per beam (2 year survey, GPU only, based on 1KWh costing £0.2)	~£430	~£265	~£245	~£213	~£224	~£151	~£84	~£45	~£35

Improvement between generations comes from a combination of advances in both the hardware and algorithm

Case study 2: Fourier Domain Acceleration Searching for the SKA

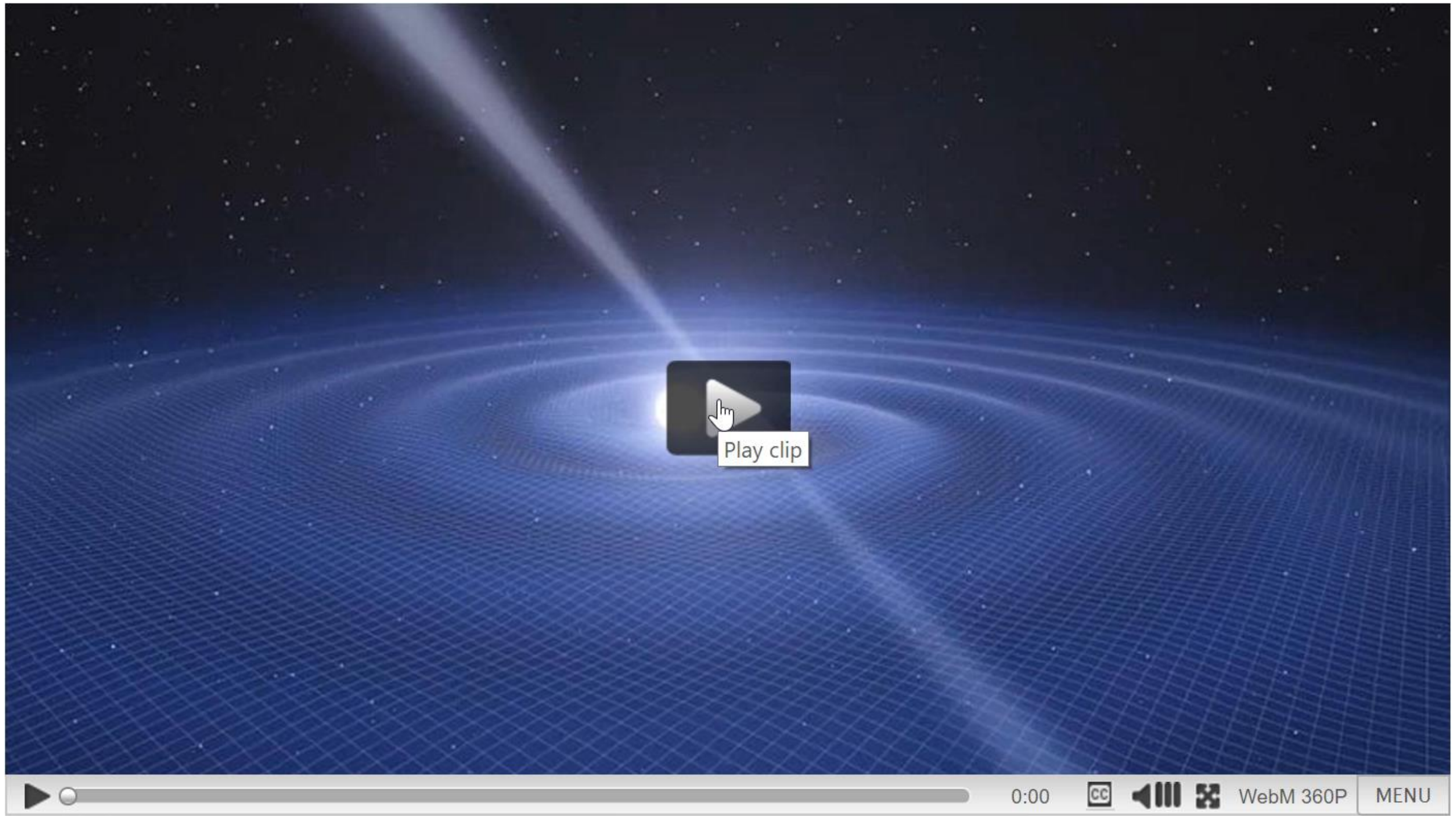
Sofia Dimoudi, Karel Adámek, Jack White, Mike Giles
(Oxford)



Light curve and slow motion picture of the **solitary** pulsar located in the centre of the Crab Nebula.

Image taken with a photon counting camera on the 80cm telescope of the Wendelstein Observatory, Dr. F. Fleischmann, 1998

Binary pulsars and gravitational waves



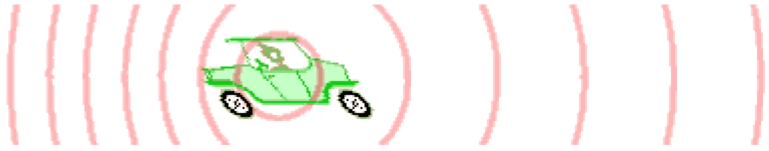
http://www.eso.org/public/videos/eso1319a/ Author: ESO/L. Calçada

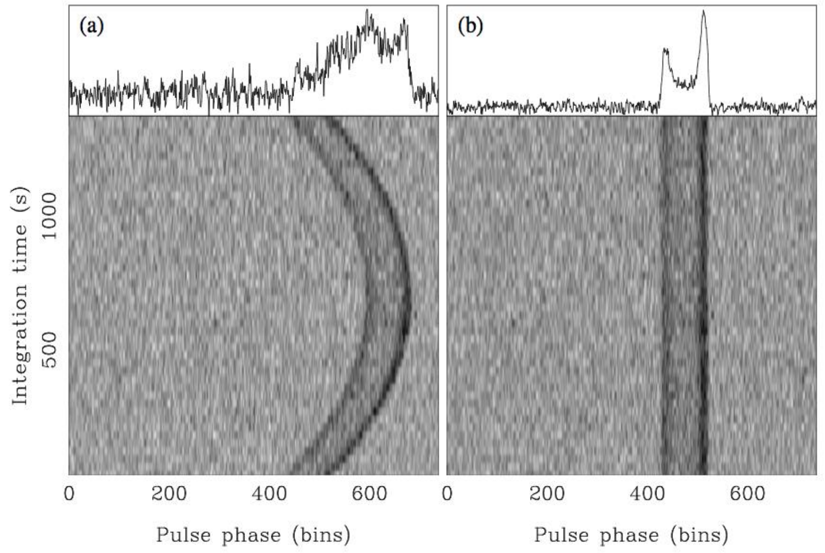
Fourier Domain Acceleration Search - FDAS

Signals from binary systems can undergo a **Doppler shift due to accelerated motion experienced over the orbital period**.

Much like the sound of a siren approaching you and then speeding away.

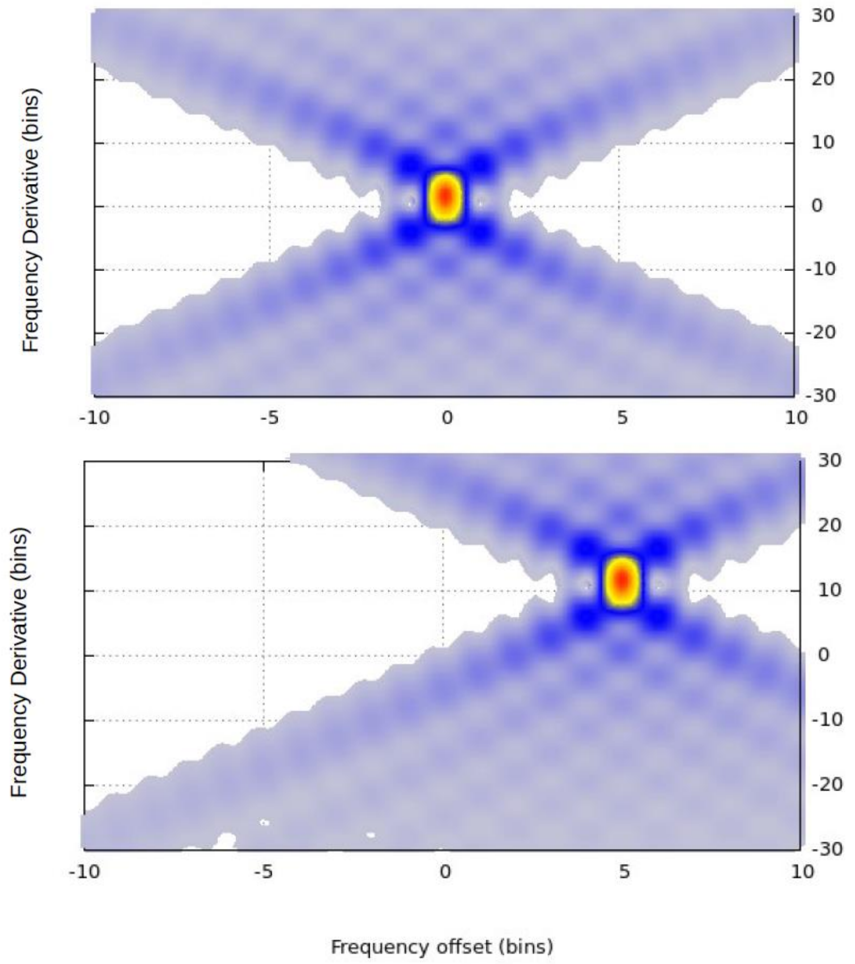
This can be corrected by using a matched filter approach.





Ransom, Eikenberry, Middleditch: AJ, Vol 24, Issue 3, pp. 1788-1809

FDAS Example



The two plots illustrate the effect of orbital acceleration.

The **first plot shows a signal without acceleration**, the signal is centred on its frequency and lies on the f-dot template corresponding to zero acceleration.

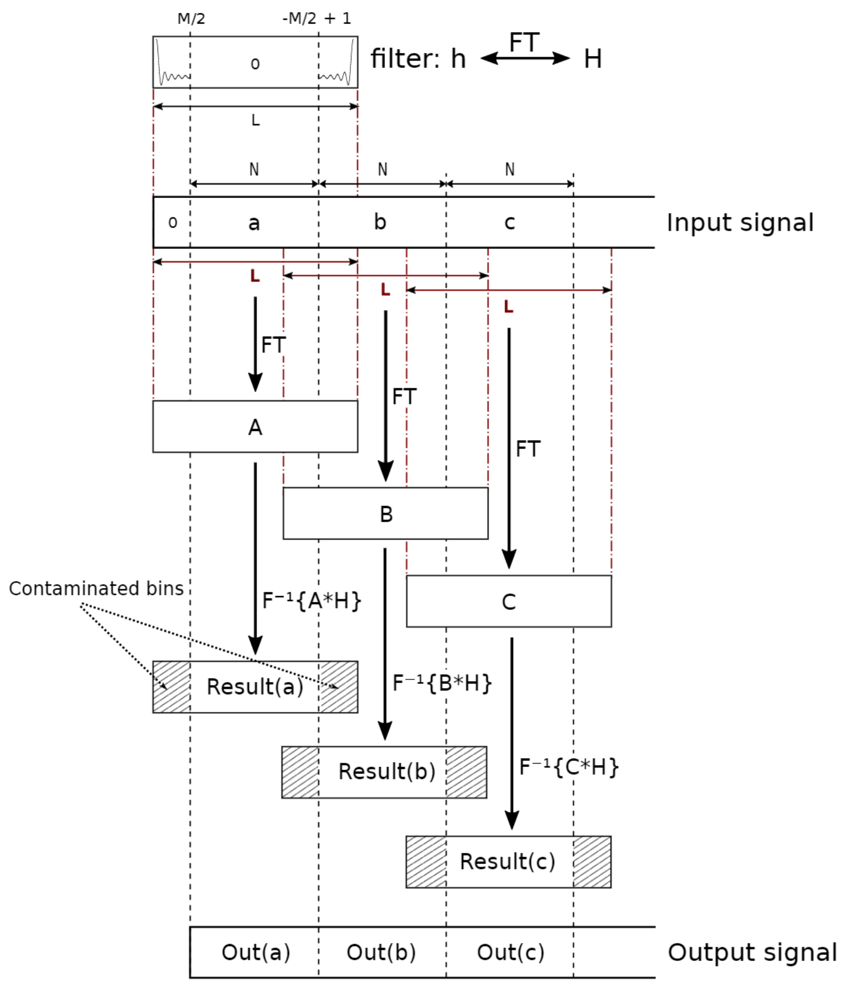
The **second plot shows a signal with a frequency derivative**, and has drifted from the original frequency by a number of bins.

Fourier Domain Acceleration Search

Use overlap-save algorithm to compute cyclic N-point convolution of template with signal segment.

Avoids the need for synchronisation because contaminated ends of convolved data are discarded (as opposed to overlap-add).

Code calculates the convolution, powers and extracts peaks.



GPU Fast Convolution via the Overlap-and-Save Method in Shared Memory: https://dl.acm.org/doi/10.1145/3394116

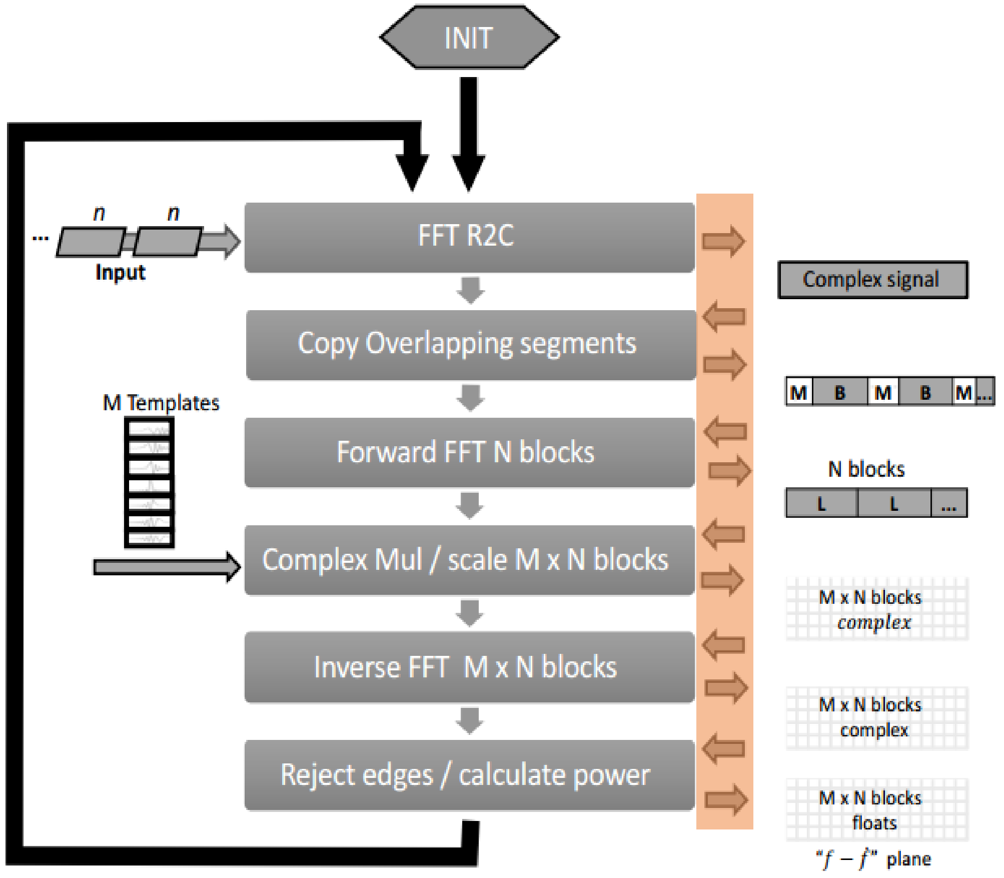
For more info: Kundur: https://www.comm.utoronto.ca/~dkundur/course_info/real-time-DSP/notes/8_Kundur_Overlap_Save_Add.pdf

Fourier Domain Acceleration Search

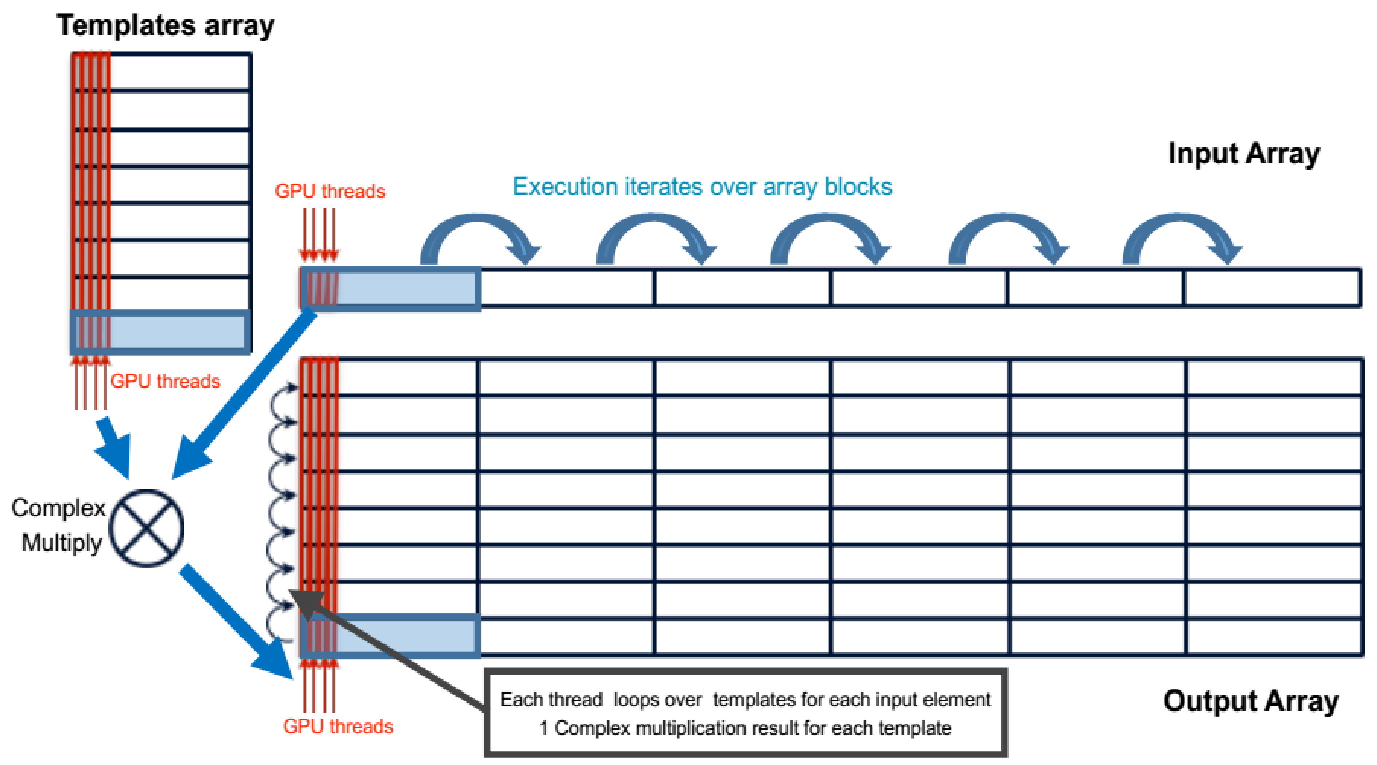
Using **cuFFT means many transactions to device memory on the GPU** (represented by grey arrows on the right of the diagram).

This causes the computation to be **limited by global memory bandwidth** (the lowest common denominator on a GPU).

This means that a cuFFT based implementation is very slow.



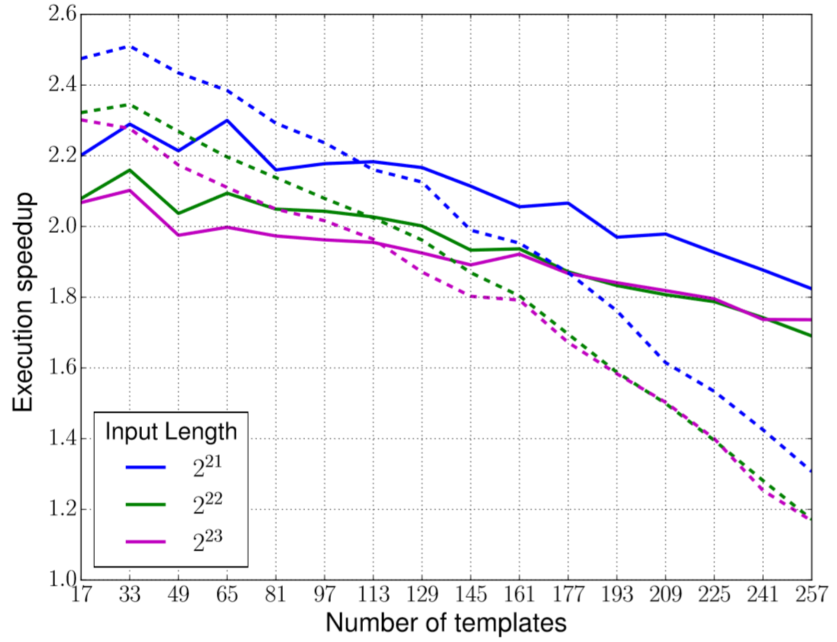
Fourier Domain Acceleration Search

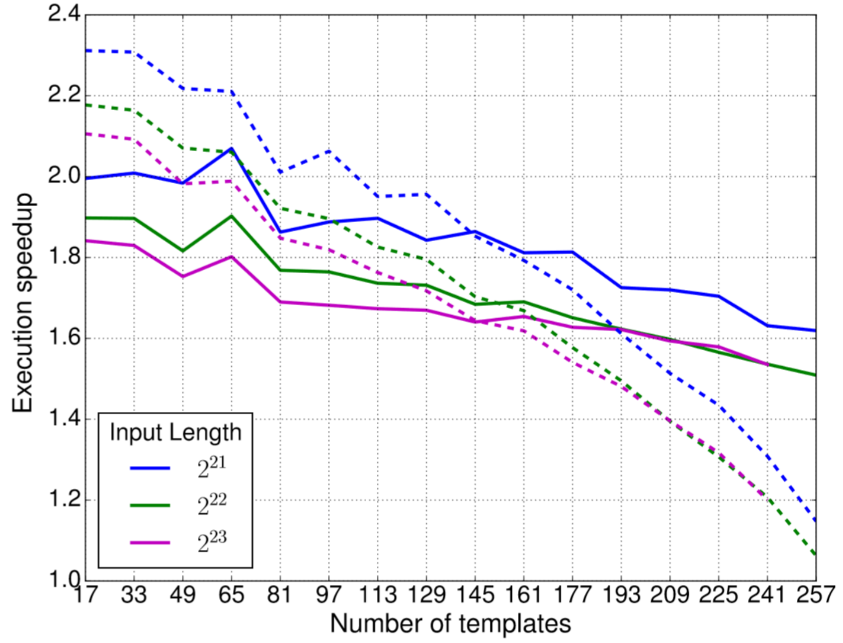

Templates array
GPU threads
Execution iterates over array blocks
Input Array
GPU threads
Complex Multiply
GPU threads
Each thread loops over templates for each input element
1 Complex multiplication result for each template
Output Array

By writing our own custom I/FFT codes to work on shared memory we can perform the FFT, pointwise multiply and scale, IFFT and edge rejection all in one kernel.

Karel Adámek, Sofia Dimoudi, Mike Giles, and Wesley Armour. 2020. GPU Fast Convolution via the Overlap-and-Save Method in Shared Memory. *ACM Trans. Archit. Code Optim.* 17, 3, Article 18 (August 2020). https://doi.org/10.1145/3394116

Fourier Domain Acceleration Search





Initial results from our tests on a Tesla P100. In the SKA region of interest – signal length 2^{23}, template size of 512 (solid line) and no interbinning (left graph)

Further optimisation achieved approximately a 3.5x speed increase on a V100

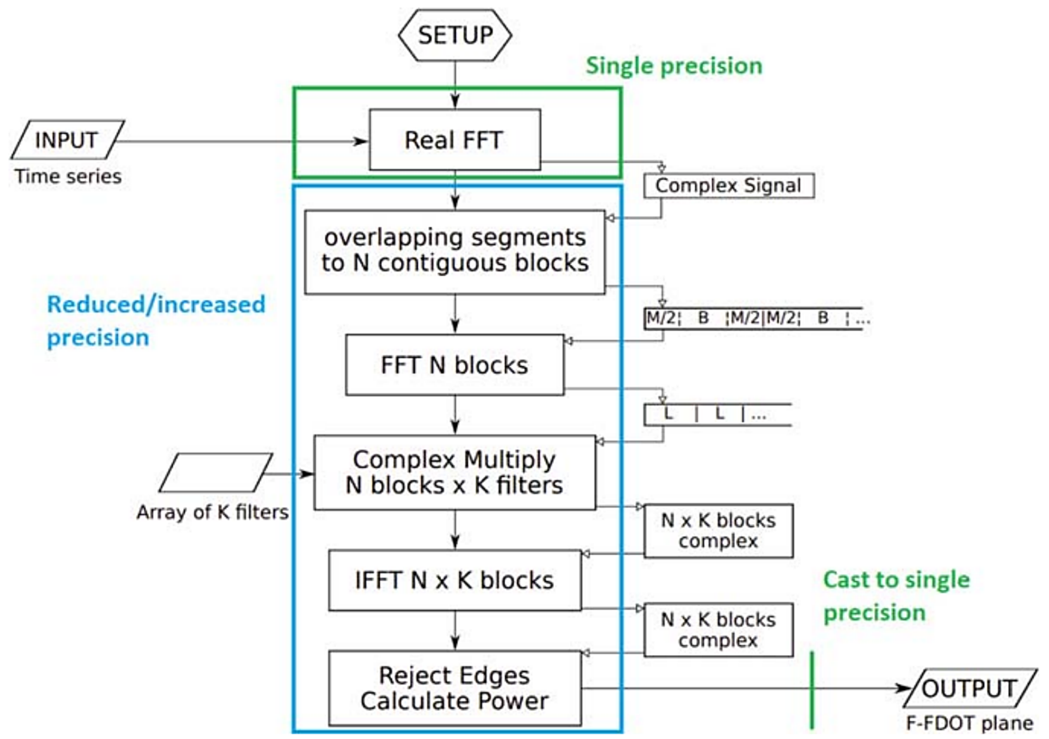
Sofia Dimoudi *et al* 2018 *ApJS* **239** 28 https://iopscience.iop.org/article/10.3847/1538-4365/aabe88

Karel Adámek, Sofia Dimoudi, Mike Giles, and Wesley Armour. 2020. GPU Fast Convolution via the Overlap-and-Save Method in Shared Memory. *ACM Trans. Archit. Code Optim.* 17, 3, Article 18 (August 2020). **https://doi.org/10.1145/3394116**

FDAS further optimisation

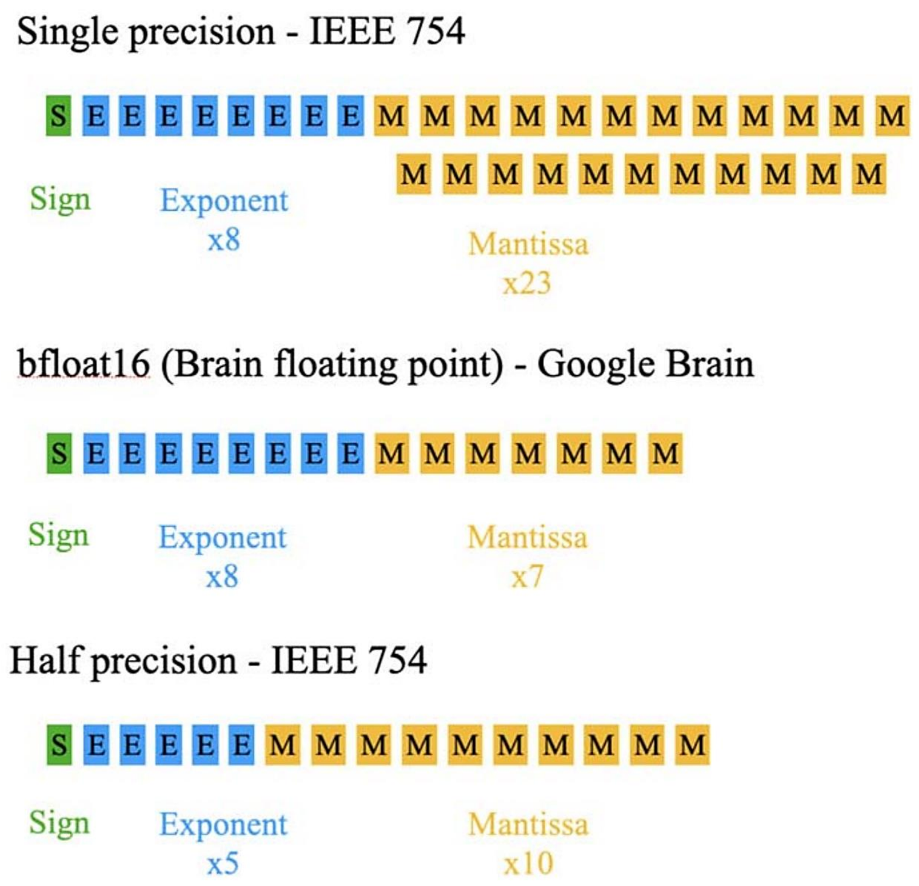
Jack's work has focused on implementing **mixed precision for FDAS**.

By reducing the precision in the convolution part of the algorithm (where FFTs can be shorter and we have far smaller accumulated errors) we are able to double the bandwidth throughput for the convolution parts of the code.



FDAS further optimisation

Jack used **bfloat16 (16 bits) for the convolution part of FDAS**. This is because it has the same range as float (fp32) so there is no need to scale numbers, and due to the fact that the exponent is the same as fp32 we are easily able to type convert between the two (on GPUs the SFU performs type conversion and it is a limited resource).

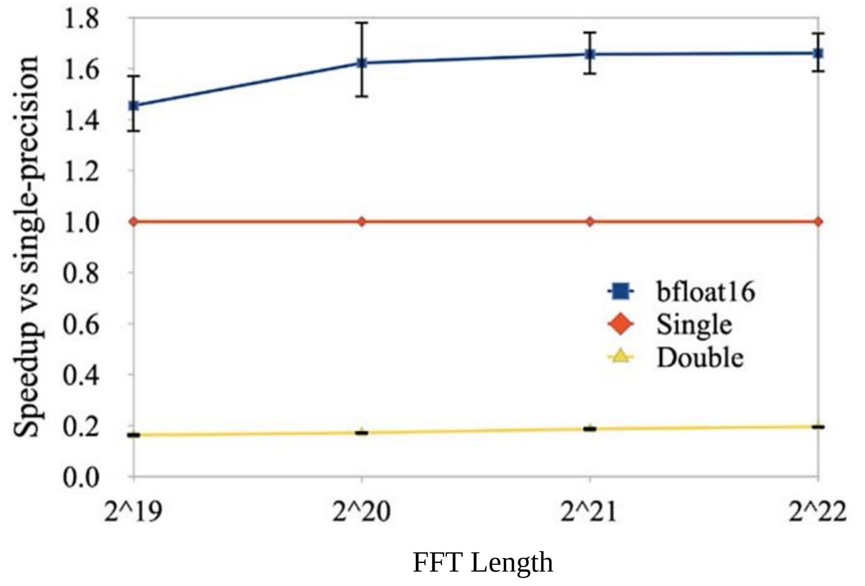


Single precision - IEEE 754
S E E E E E E E E M M M M M M M M M M M M
M M M M M M M M M M M
Sign
Exponent x8
Mantissa x23
bfloat16 (Brain floating point) - Google Brain
S E E E E E E E E M M M M M M M
Sign
Exponent x8
Mantissa x7
Half precision - IEEE 754
S E E E E E M M M M M M M M M M
Sign
Exponent x5
Mantissa x10

FDAS further optimisation

The work provides roughly a 1.7x speed increase compared to a single precision implementation.

So do consider whether you really need to compute in double or float or…..



Jack White *et al* 2023 *ApJS* **265** 13 https://iopscience.iop.org/article/10.3847/1538-4365/acb351/meta

FDAS Energy optimisation

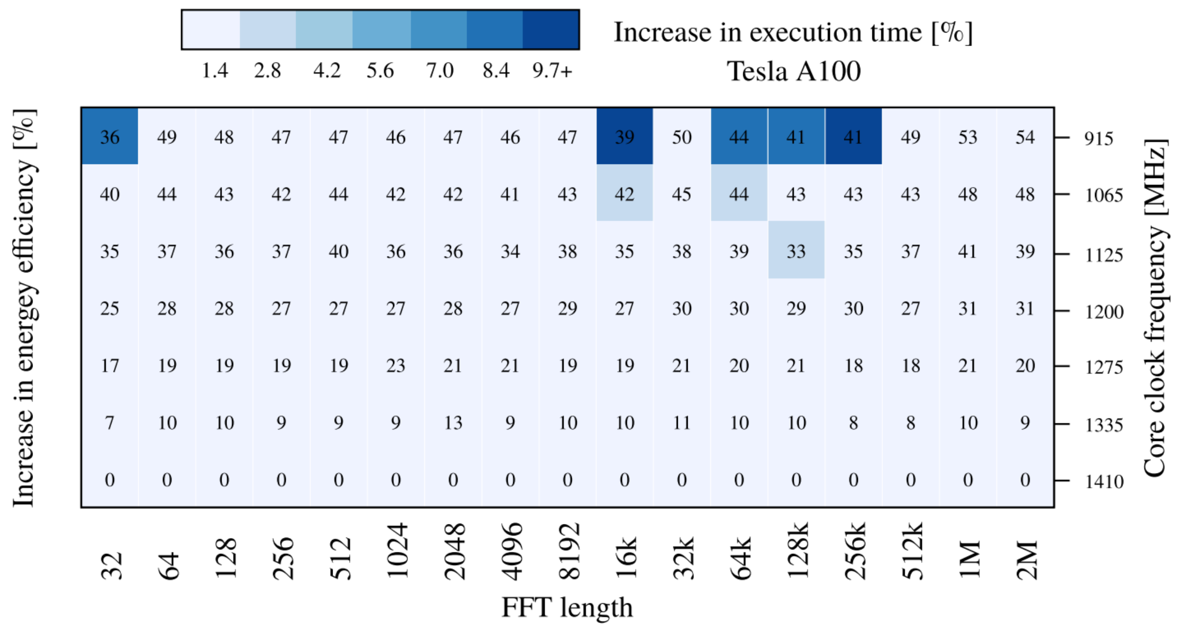
For a bandwidth bound algorithm it makes no sense to have the GPU cores running as quickly as they can.

The reason is that your memory cannot deliver data to them quickly enough to utilise all of the computation that they can perform.

FDAS Energy optimisation

Because the **FFT is bandwidth bound we are able to reduce the GPU core clock frequency without having significant impact on execution time.**

The figure (right) shows the energy saving for a given core clock frequency. The shading indicates the increase in execution time.



Core clock frequency [MHz]	32	64	128	256	512	1024	2048	4096	8192	16k	32k	64k	128k	256k	512k	1M	2M
915	36	49	48	47	47	46	47	46	47	39	50	44	41	41	49	53	54
1065	40	44	43	42	44	42	42	41	43	42	45	44	43	43	43	48	48
1125	35	37	36	37	40	36	36	34	38	35	38	39	33	35	37	41	39
1200	25	28	28	27	27	27	28	27	29	27	30	30	29	30	27	31	31
1275	17	19	19	19	19	23	21	21	19	19	21	20	21	18	18	21	20
1335	7	10	10	9	9	9	13	9	10	10	11	10	10	8	8	10	9
1410	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

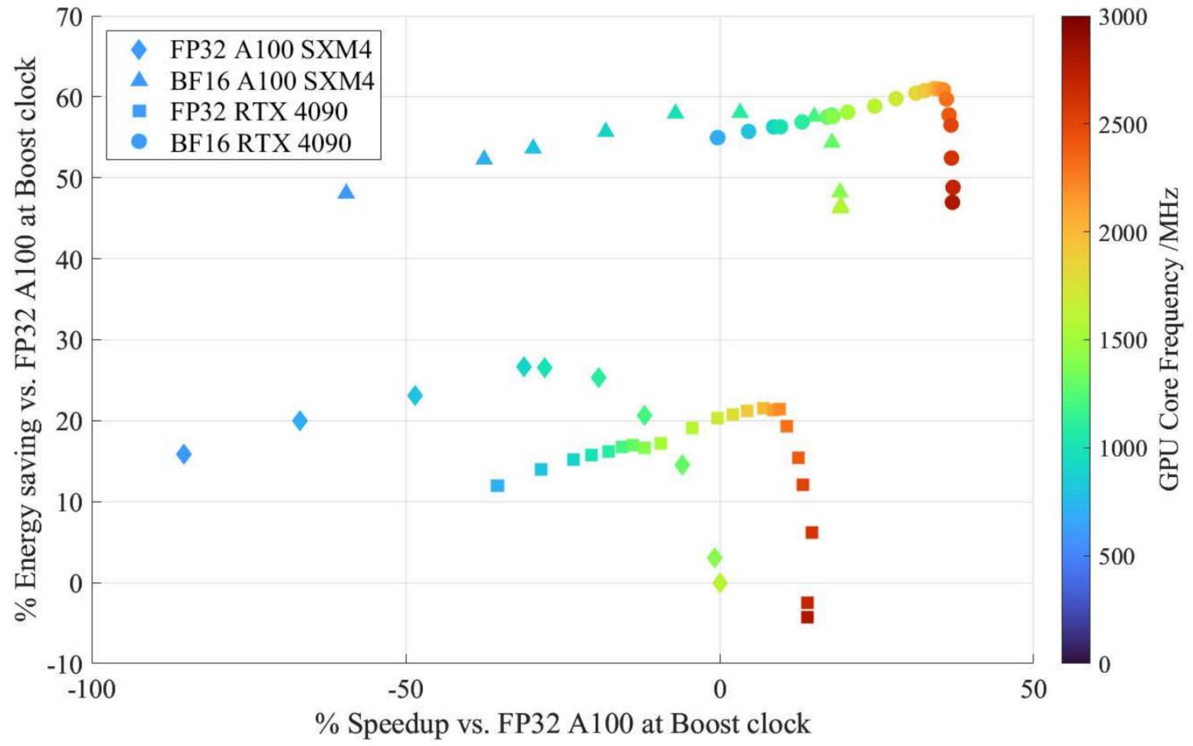
K. Adámek, et al in *IEEE Access* https://ieeexplore.ieee.org/abstract/document/9330509

FDAS Energy optimisation

We can use NVML to change the clock speed of the processing cores to slow them down.

This has the advantage of reducing the energy that the GPU uses.

By combining mixed precision with reduction in core clock frequency we are able to achieve speed increase and energy savings.



Jack White *et al* Astronomical Data Analysis Software and Systems, ADASS XXXII

A final word on measurement

Often nvidia-smi is used to establish values of physical parameters of the GPU whilst it is executing code.

If you have workloads with short duration kernels this might not be the best thing to do…

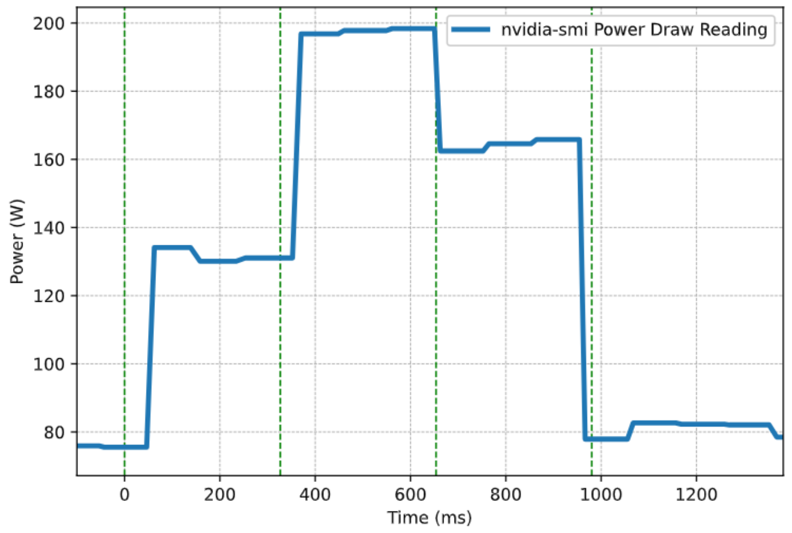


Figure 1: nvidia-smi can report drastically different power draw for the same CUDA Kernel, ranging from 80W to 200W. This is because nvidia-smi does not fully capture the power information on some GPUs. The figure shows a CUDA program that runs for 325ms on an A100 GPU. The kernel is executed 4 times, and the green dotted line indicates the beginning of each iteration.

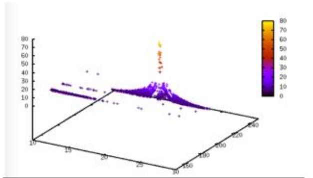
Jim Yang et. al. (to appear in SC24 proceedings) https://arxiv.org/abs/2312.02741

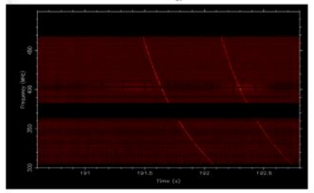
Impact

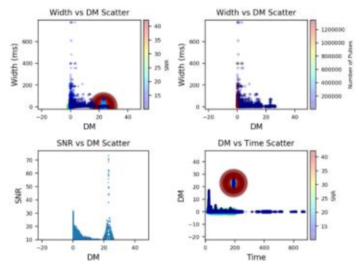
Giant Metrewave Radio Telescope
Presentation

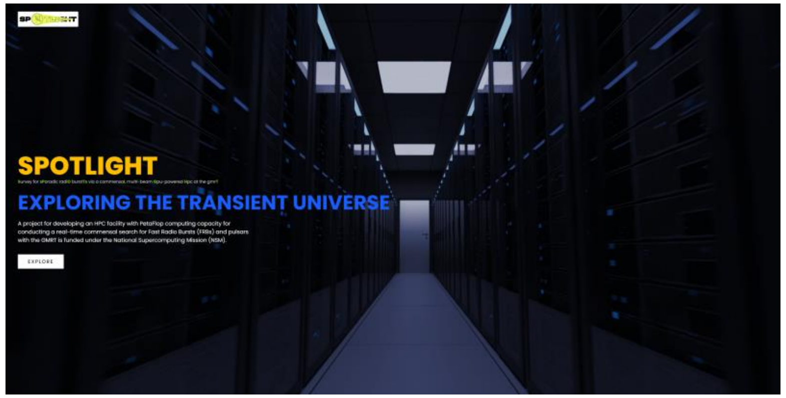
Real-time FRB search pipeline for GHRSS

- A GPU-based FRB detection pipeline for the GMRT (using AstroAccelerate credit: Wes et al.) → 1st RRAT from GMRT discovered!!
- GMRT beams can be processed in (5x faster than) real-time for DM up to 2000 pc cm^{-3} (up to z=2)









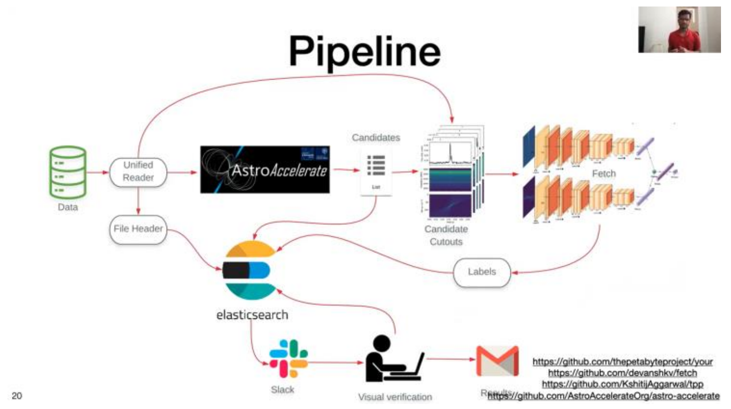
SPOTLIGHT

Survey for sPoradic radiO burstTs via a commensaL multi-beam Gpu-powered Hpc at the gmrT

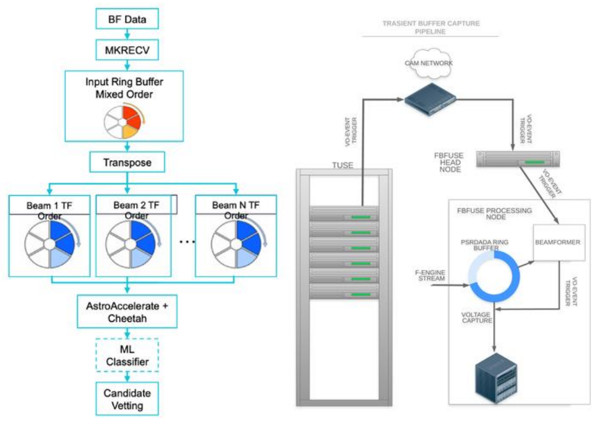
EXPLORING THE TRANSIENT UNIVERSE

A project for developing an HPC facility with PetaFlop computing capacity for conducting a real-time commensal search for Fast Radio Bursts (FRBs) and pulsars with the GMRT is funded under the National Supercomputing Mission (NSM).

EXPLORE


Pipeline
Data
Unified Reader
File Header
AstroAccelerate
Candidates
Candidate Cutouts
Fetch
Labels
elasticsearch
Slack
Visual verification
Results
https://github.com/thepetabyteproject/your
https://github.com/devanshkv/fetch
https://github.com/KshitijAggarwal/tpp
https://github.com/AstroAccelerateOrg/astro-accelerate
20

The Petabyte FRB Search Project
Presentation



MeerKAT (MeerTRAP)
paper (arxiv)

Acknowledgments and Collaborators

University of Oxford

Mike Giles (Maths)

Aris Karastergiou (Physics)

Chris Williams (Physics)

Steve Roberts (Engineering)

Sofia Dimoudi (OeRC)

Nassim Ouannoughi (OeRC)

Karel Adamek (OeRC)

Jayanth Chennamangalam (Physics)

Griffin Foster (Physics)

University of Manchester

Ben Stappers

Mike Keith

Prabu Thiagaraj

Jayanta Roy

Mitch Mickaliger

NRAO/UVA

Scott Ransom

University of Bristol

Dan Curran (Electrical Engineering)

Simon McIntosh Smith (Electrical Engineering)

ASTRON

Cees Bassa

Jason Hessels

University of Opava

Jan Novotny

NVIDIA

Kate Clark

Tim Lanfear

Tom Bradley

Max Plank

Ewan Barr

AstroAccelerate

http://www.oerc.ox.ac.uk/projects/astroaccelerate