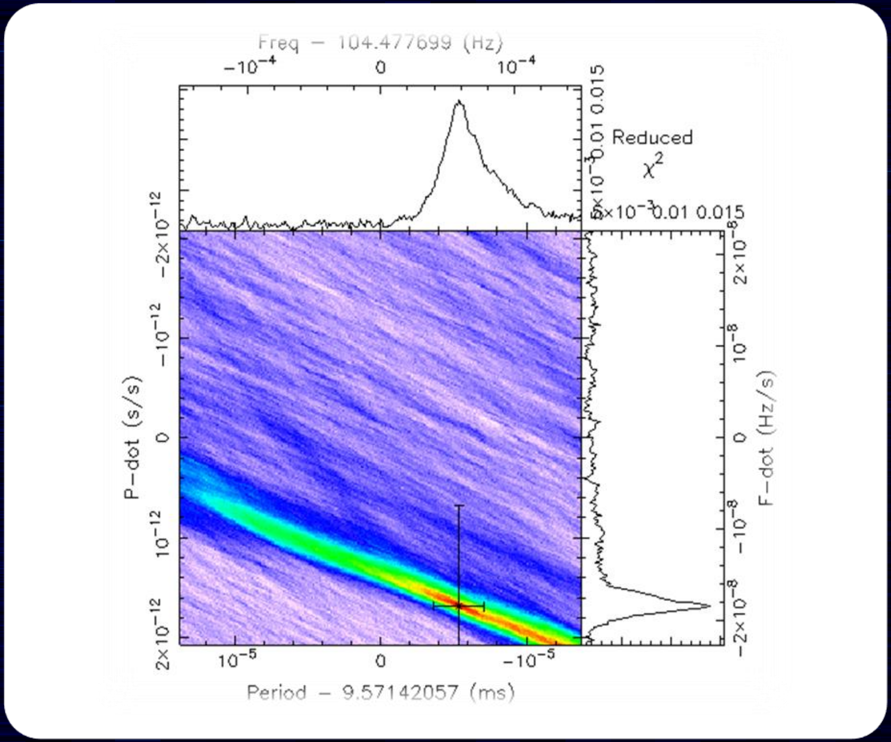
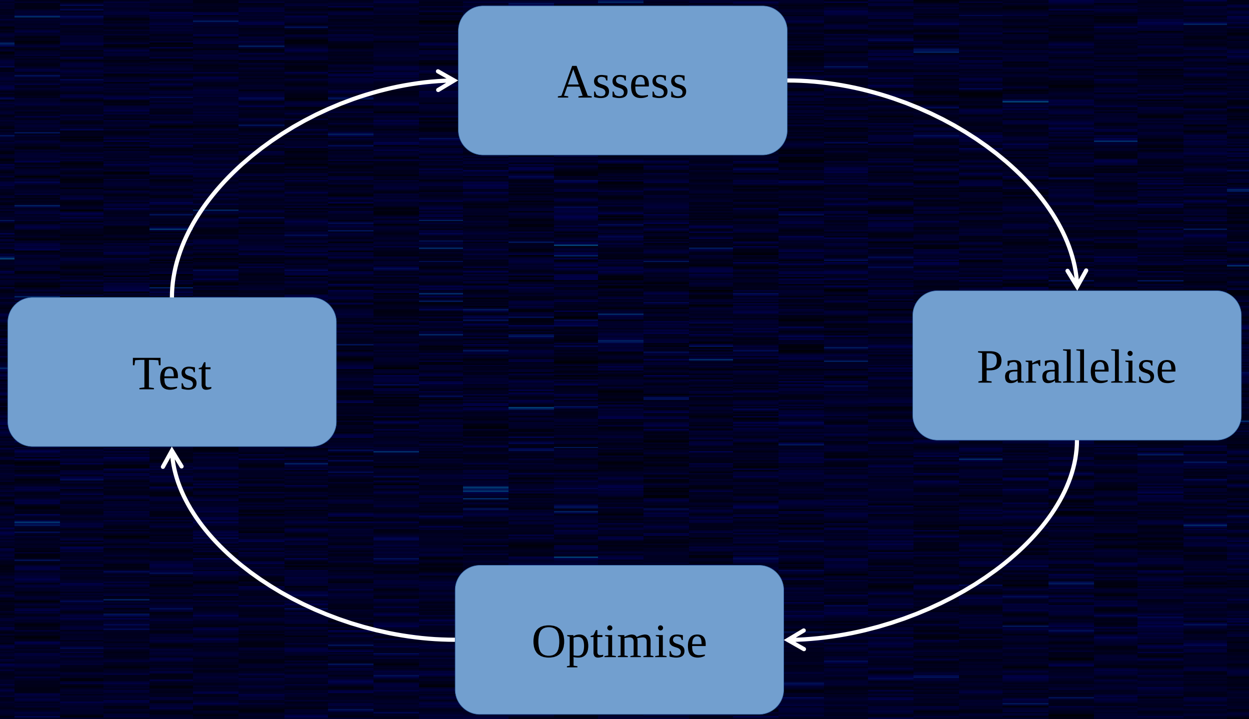
Accelerating the acceleration search
a case study



By Chris Laidler

Optimization cycle



Profile

- Identify the function or functions in which the application is spending most of its execution time.
- CPU code:
  - gprof
  - valgrind
  - oprofile
- Identifying hotspots

Parallelize

- Use existing libraries
- Code to expose parallelism

Optimizing CUDA code

- Using CPU Timers
  - CudaDeviceSynchronize()
  - cudaEventSynchronize()
- Using CUDA GPU Timers
  - cudaEventCreate(&start)
  - CudaEventElapsedTime()
- Bandwidth
  - How, when

Data Transfer Between Host and Device

- Minimize data transfer between the host and the device. Even if that means running kernels on the GPU that do not demonstrate any speedup compared with running them on the host CPU.
- Keep it in device memory
- Batch many small transfers into one larger transfer
- Use page-locked (or pinned) memory
  - CudaHostAlloc()

Asynchronous and Overlapping memory Transfers with Computation

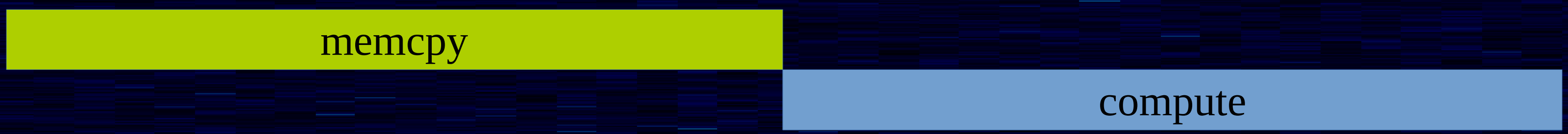
- A stream is simply a sequence of operations that are performed in order on the device. Operations in different streams can be interleaved and in some cases overlapped - a property that can be used to hide data transfers between the host and the device.
  - cudaStreamCreate(&stream1);
  - Default stream - no explicit synchronization is needed always sequential.
- Some devices are capable of concurrent copy and compute
  - cudaMemcpy() is blocking
  - cudaMemcpyAsync() is a non-blocking
- kernel<<<grid, block, 0, stream2>>>(data...);

Concurrent copy and execute

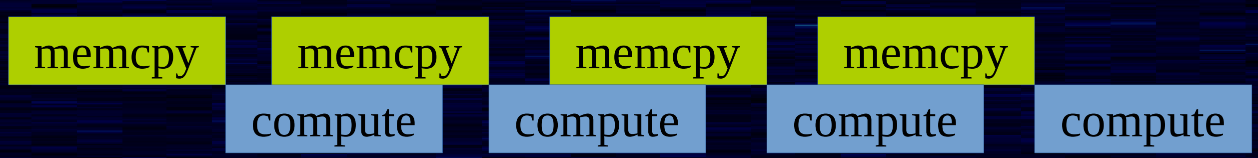
- cudaStreamCreate(&stream1);
- cudaStreamCreate(&stream2);
- cudaMemcpyAsync(a_d, a_h, size, cudaMemcpyHostToDevice, stream1);
- kernel<<<grid, block, 0, stream2>>>(otherData_d);

Staged concurrent copy and execute

- Sequential



- Concurrent



Device Memory Spaces

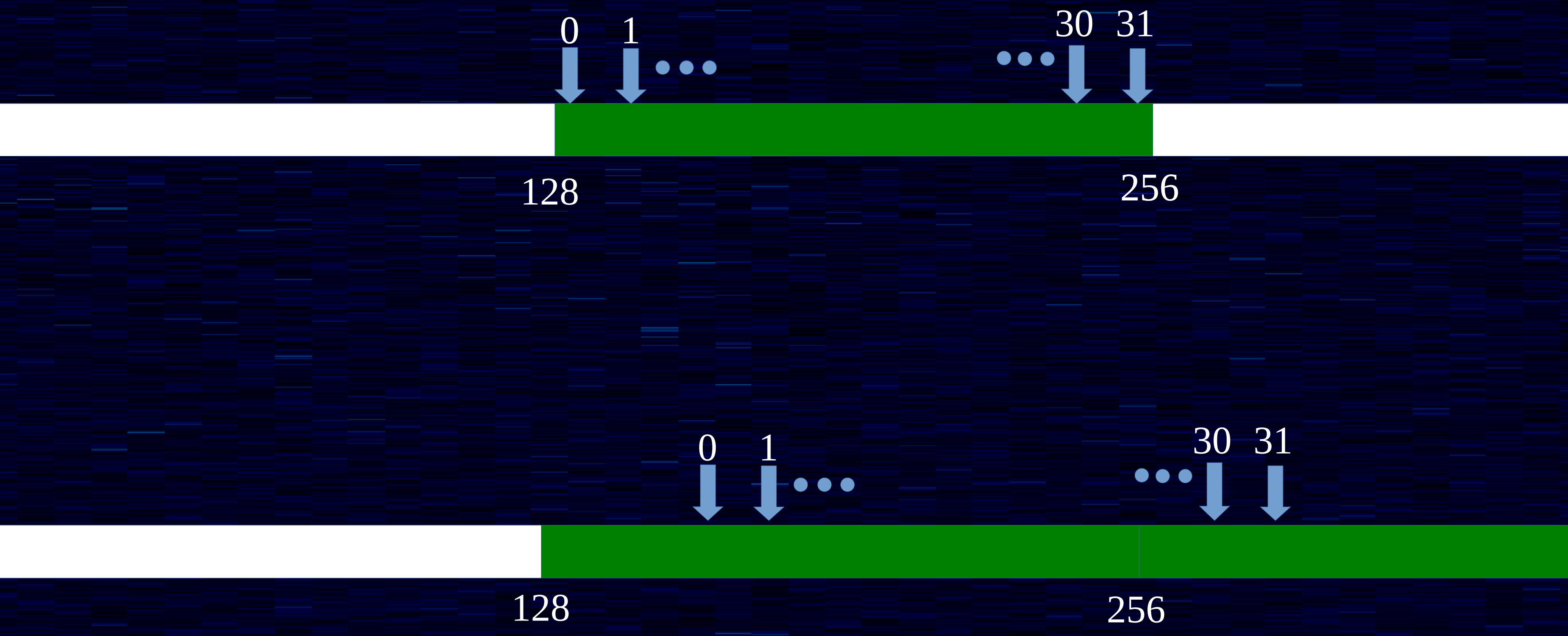
- Coalesced Access to Global Memory
- Global memory loads and stores by threads of a warp are coalesced by the device into as few as one transaction when certain access requirements are met.
- the concurrent accesses of the threads of a warp will coalesce into a number of transactions equal to the number of cache lines necessary to service all of the threads of the warp.
- By default, all accesses are cached through L1, which as 128-byte lines.

Global memory accesses

- 2.x cached through L1, which has 128-byte lines.
- 3.x is only cached in L2.
  - L1 is reserved for local memory accesses.

A Simple Access Pattern

- A Simple Access Pattern



Memory Hierarchy

- Shared Memory
  – Minimize Bank conflicts
- Texture Memory
- Constant Memory
- Registers

Occupancy

- Occupancy: number of warps running concurrently on a multiprocessor divided by maximum number of warps that can run concurrently
- Limited by resource usage:
  - Registers
  - Shared memory
- Higher occupancy does not necessarily lead to higher performance
  - Low occupancy kernels cannot hide memory latency

Case Study

Finding pulsars

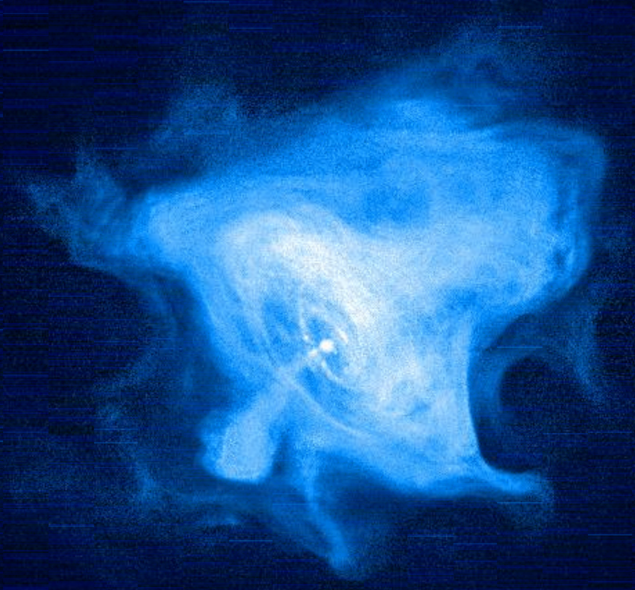
Pulsars

Neutron stars

- Mass ~1.4 $M_{\odot}$
- Radius: 10 – 80 km
- Density: 10^{14} grams/cm^3
- Rapidly rotating

  Up to 716 Hz
- Highly Magnetized

  10^8 - 10^{15} Gauss



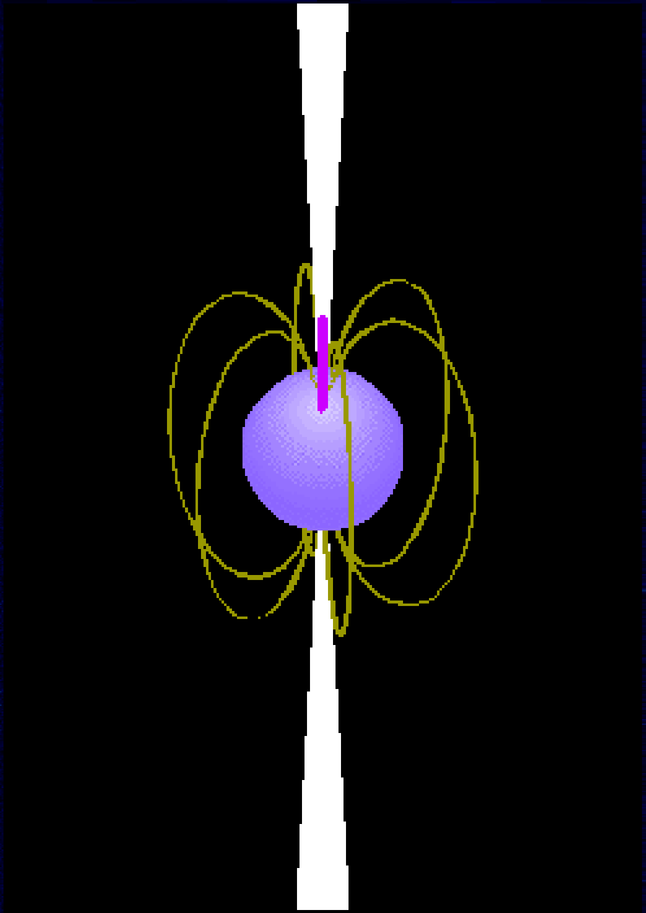
Pulsars

Neutron stars

- Mass ~1.4 $M_\odot$
- Radius: 10 – 80 km
- Density: 10^{14} grams/cm^3
- Rapidly rotating

  Up to 716 Hz
- Highly Magnetized

  10^8 - 10^{15} Gauss


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

The rotating magnetic field induces an electric field which accelerates charged particles that are then beamed from the poles of the star. If one of these beams pass over us we can detect them as a broadband periodic signal.

So how do we find new pulsars?

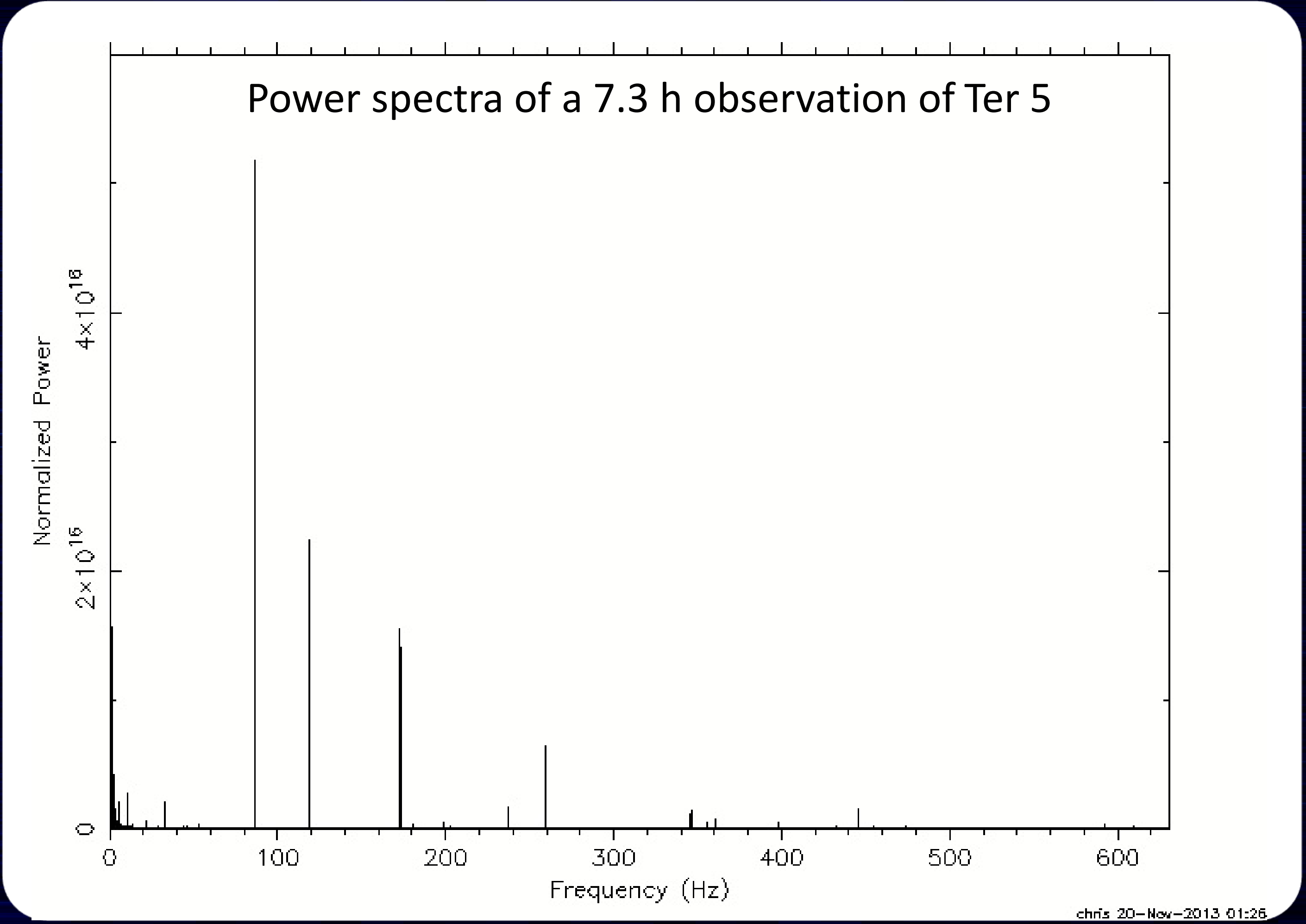
- Take a long observe with radio telescope
- High sampling rate ~12 kHz
- Remove what RIF we can
- Perform barycentric corrections
- De-disperce the observation – for a number of trial DM's

And then to find a periodic signal....

The good old Fourier Transform!

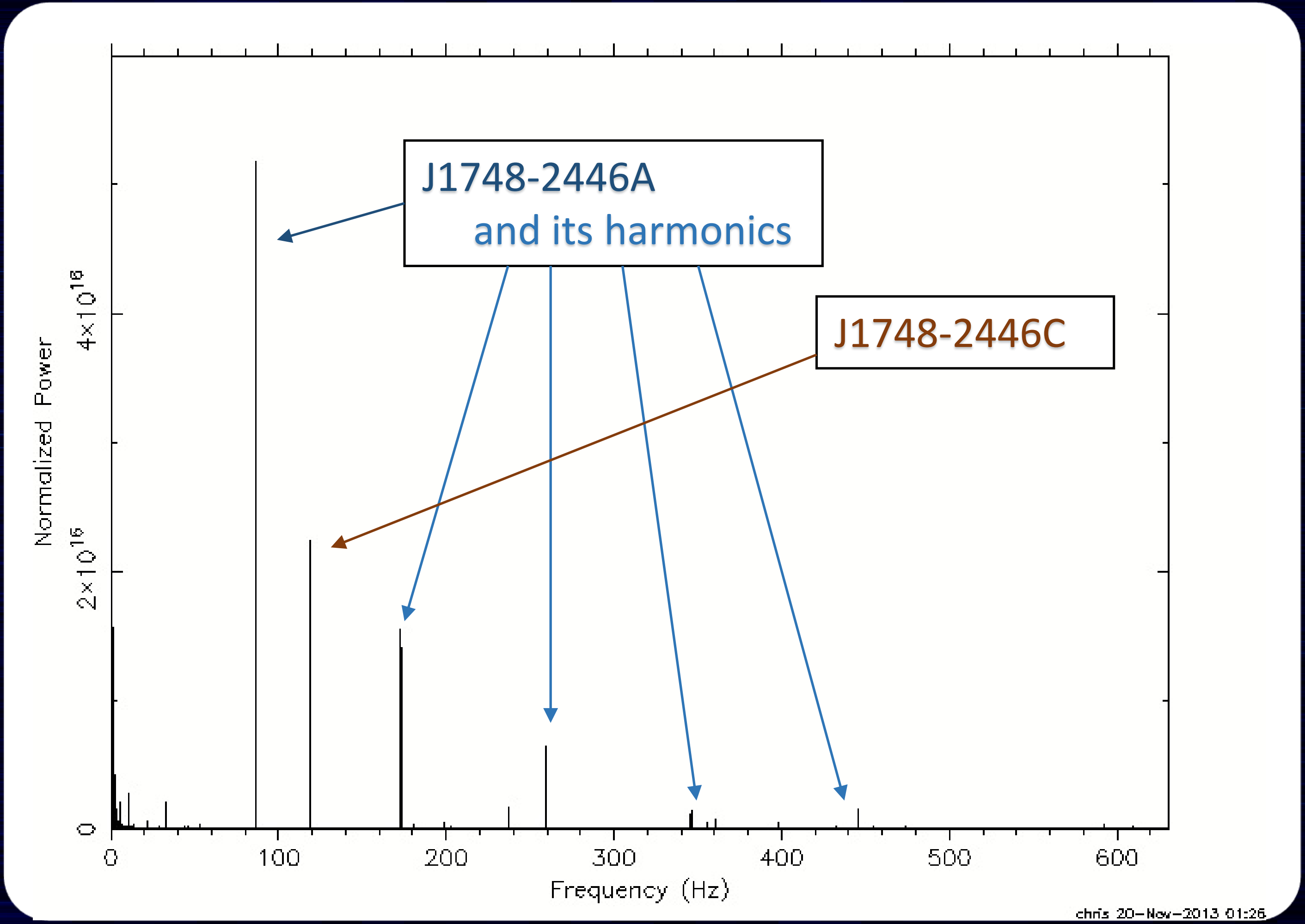
Frequency Search – Power Spectra

Lets examine a 7.3 hour observation of Terzan 5 taken on the 05/05/05 with the GBT.



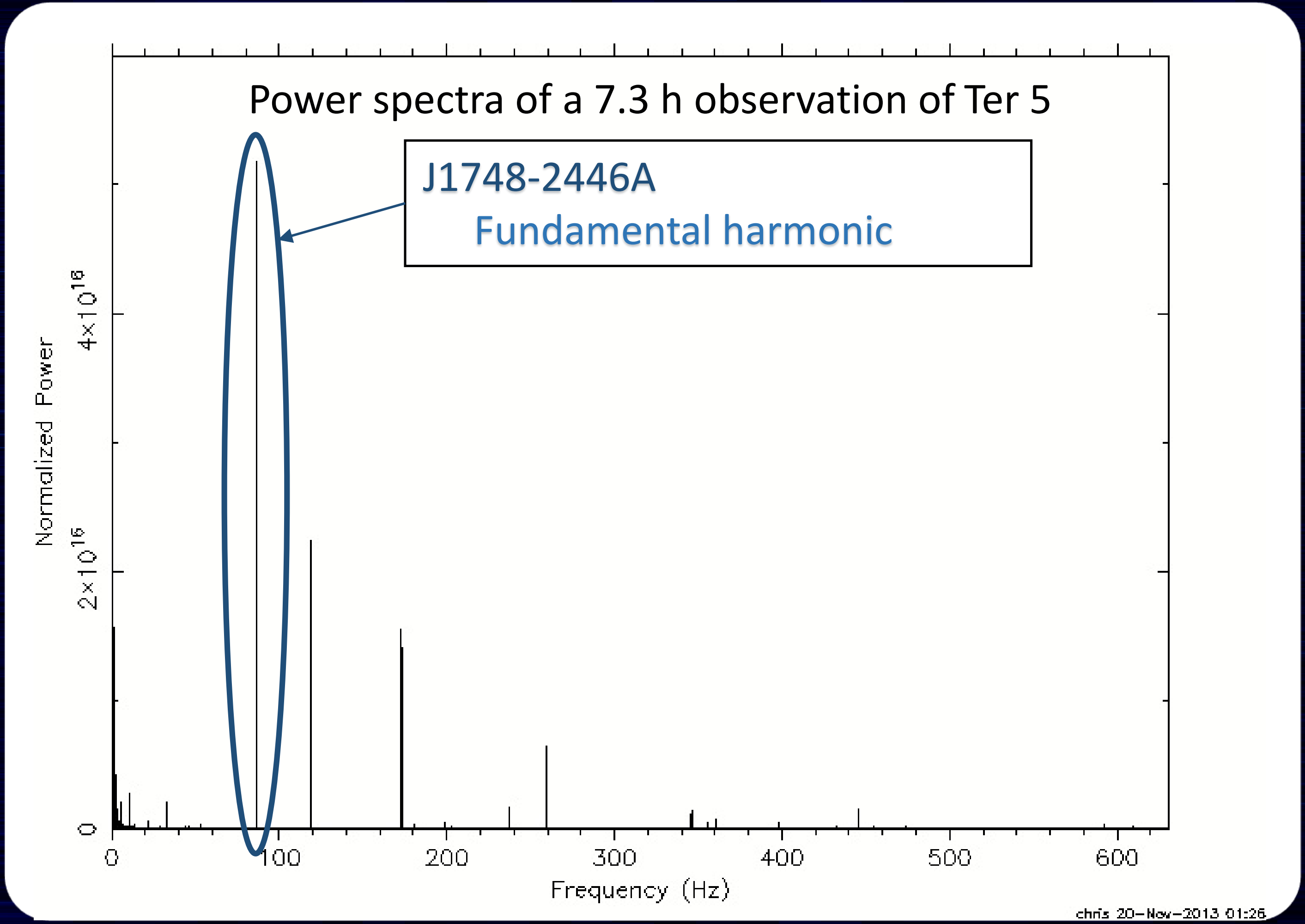
Frequency Search – Power Spectra

Lets examine a 7.3 hour observation of Terzan 5 taken on the 05/05/05 with the GBT.

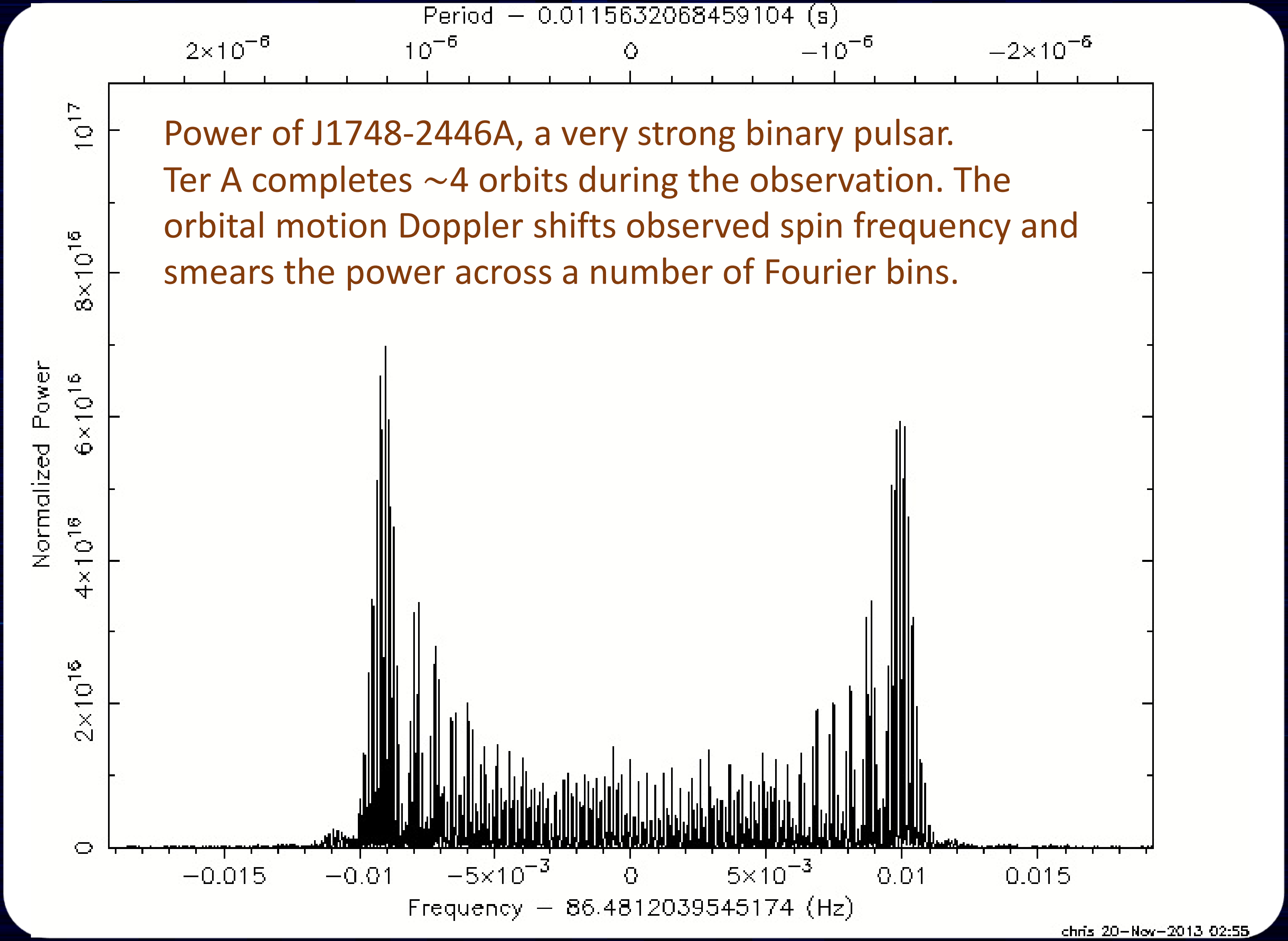


Frequency Search – Power Spectra

Lets examine a 7.3 hour observation of Terzan 5 taken on the 05/05/05 with the GBT.

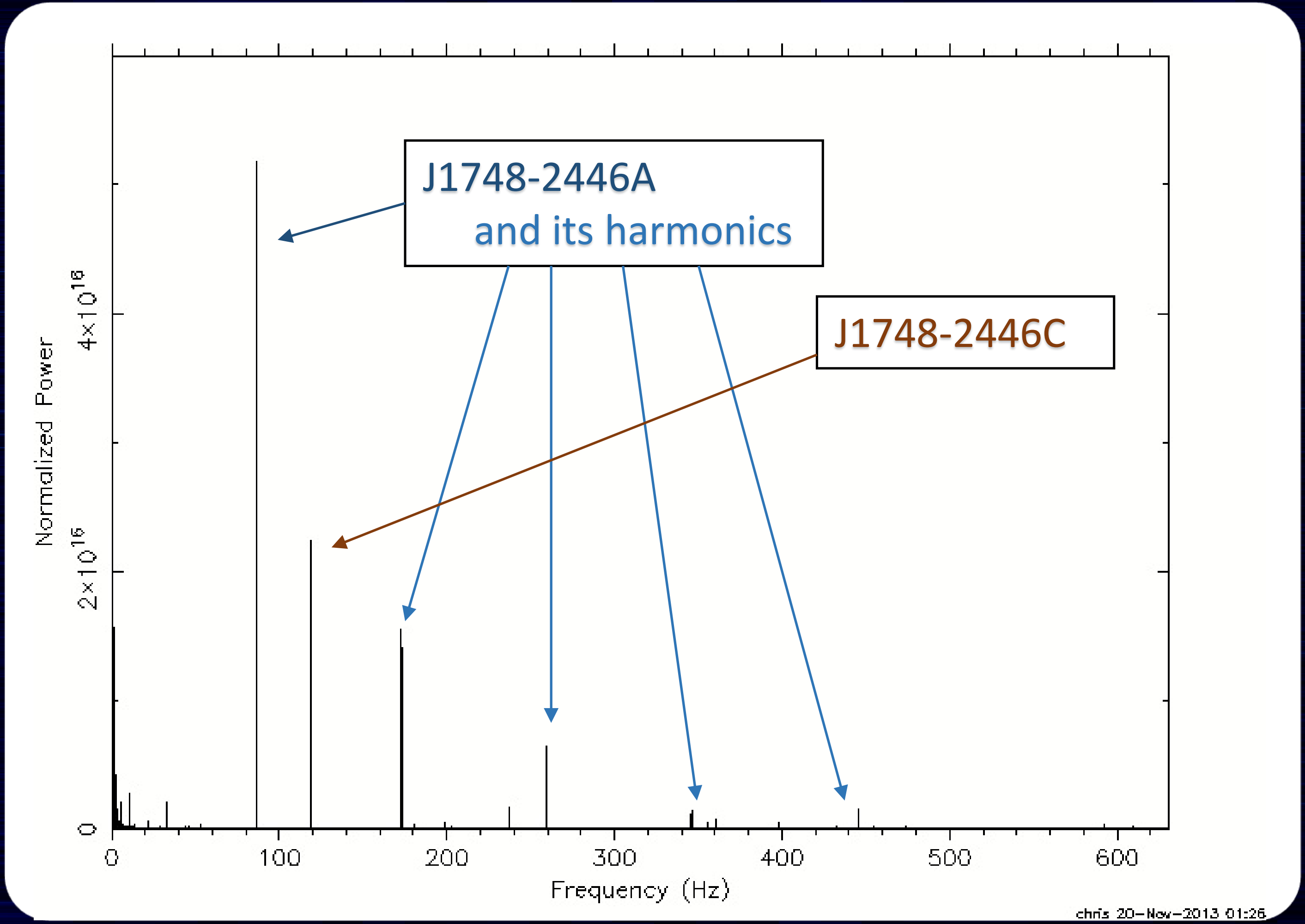


Frequency Search – Power Spectra



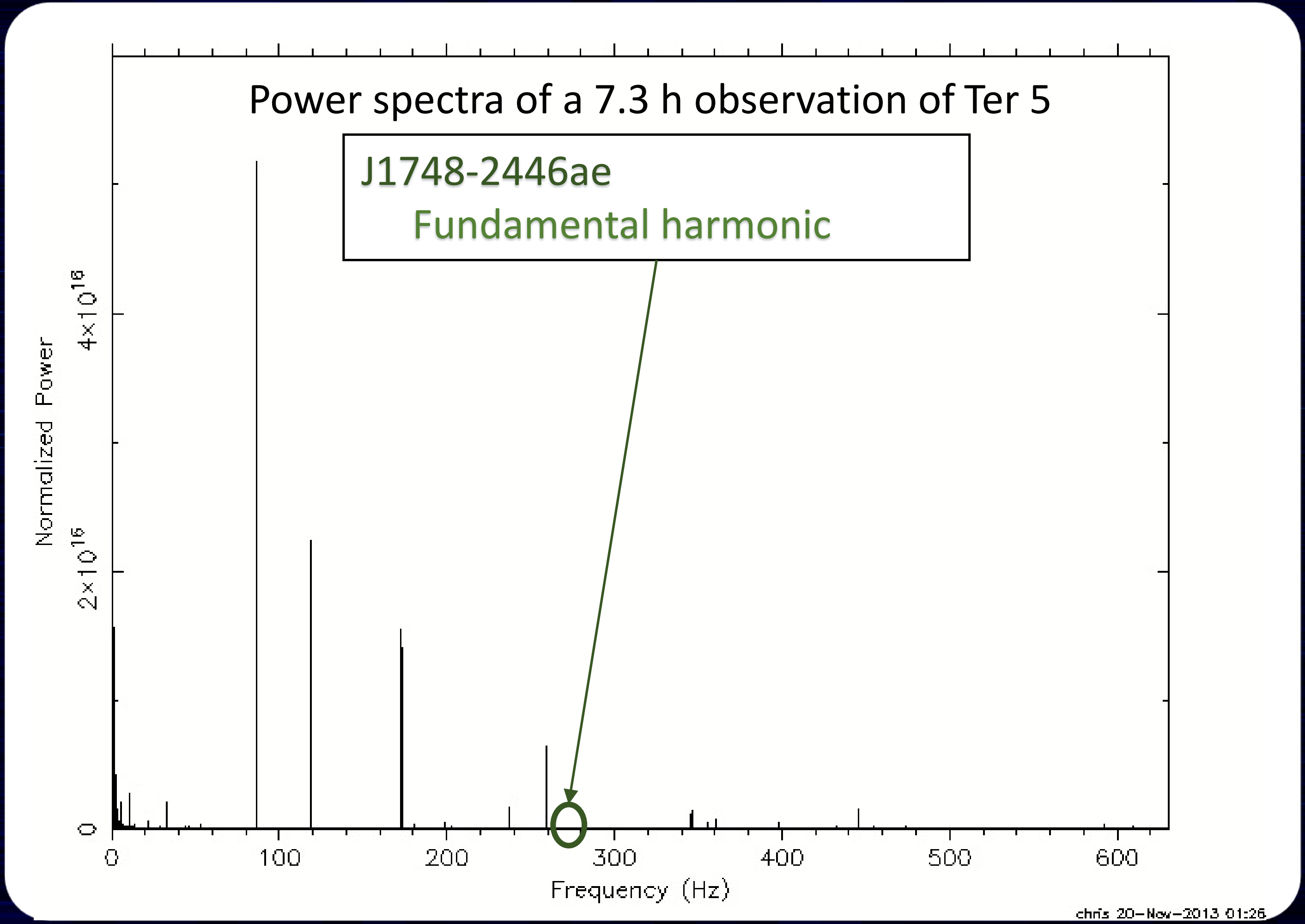
Frequency Search – Power Spectra

Lets examine a 7.3 hour observation of Terzan 5 taken on the 05/05/05 with the GBT.

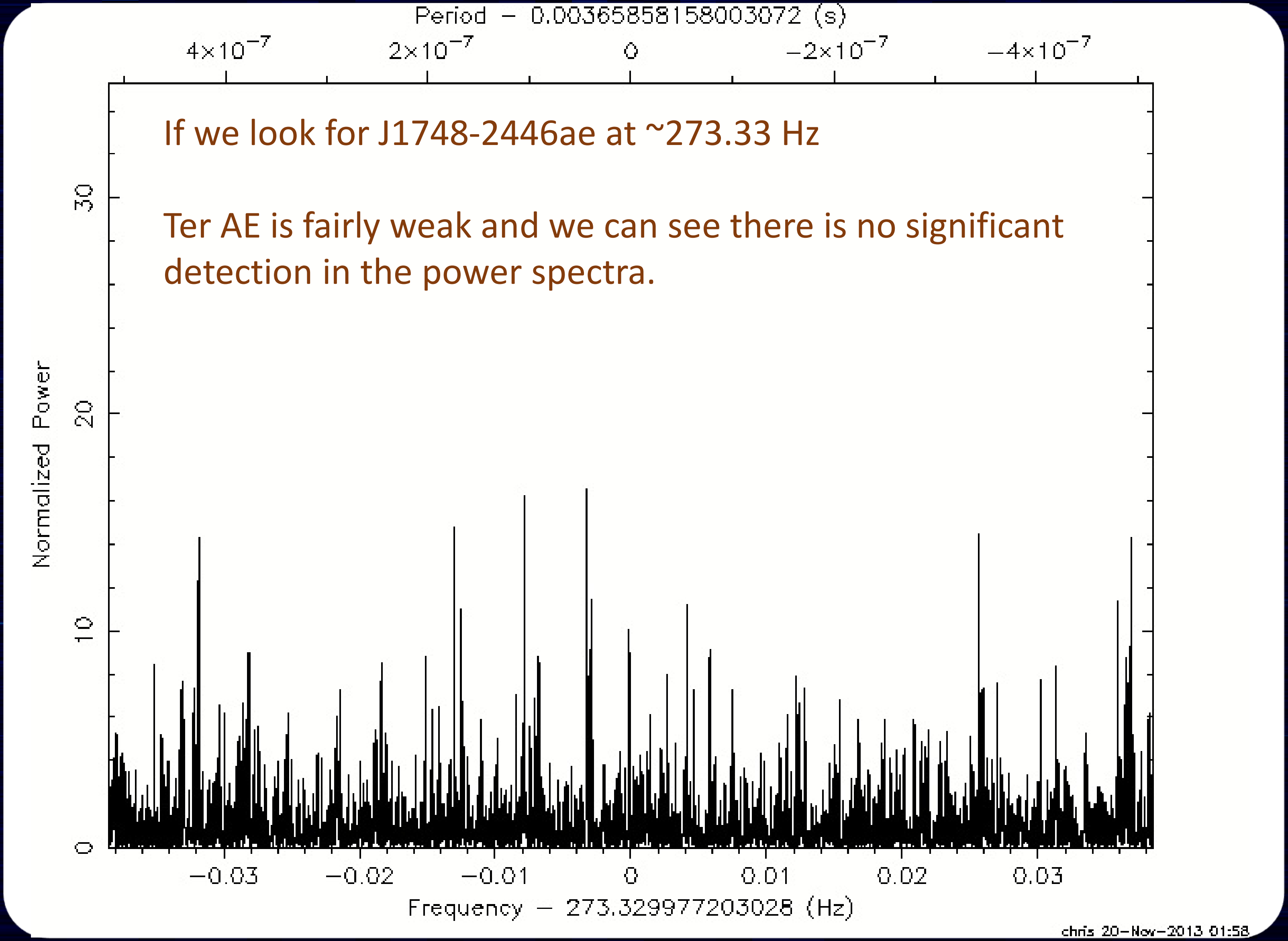


Frequency Search – Power Spectra

Lets examine a 7.3 hour observation of Terzan 5 taken on the 05/05/05 with the GBT.



Frequency Search – Power Spectra

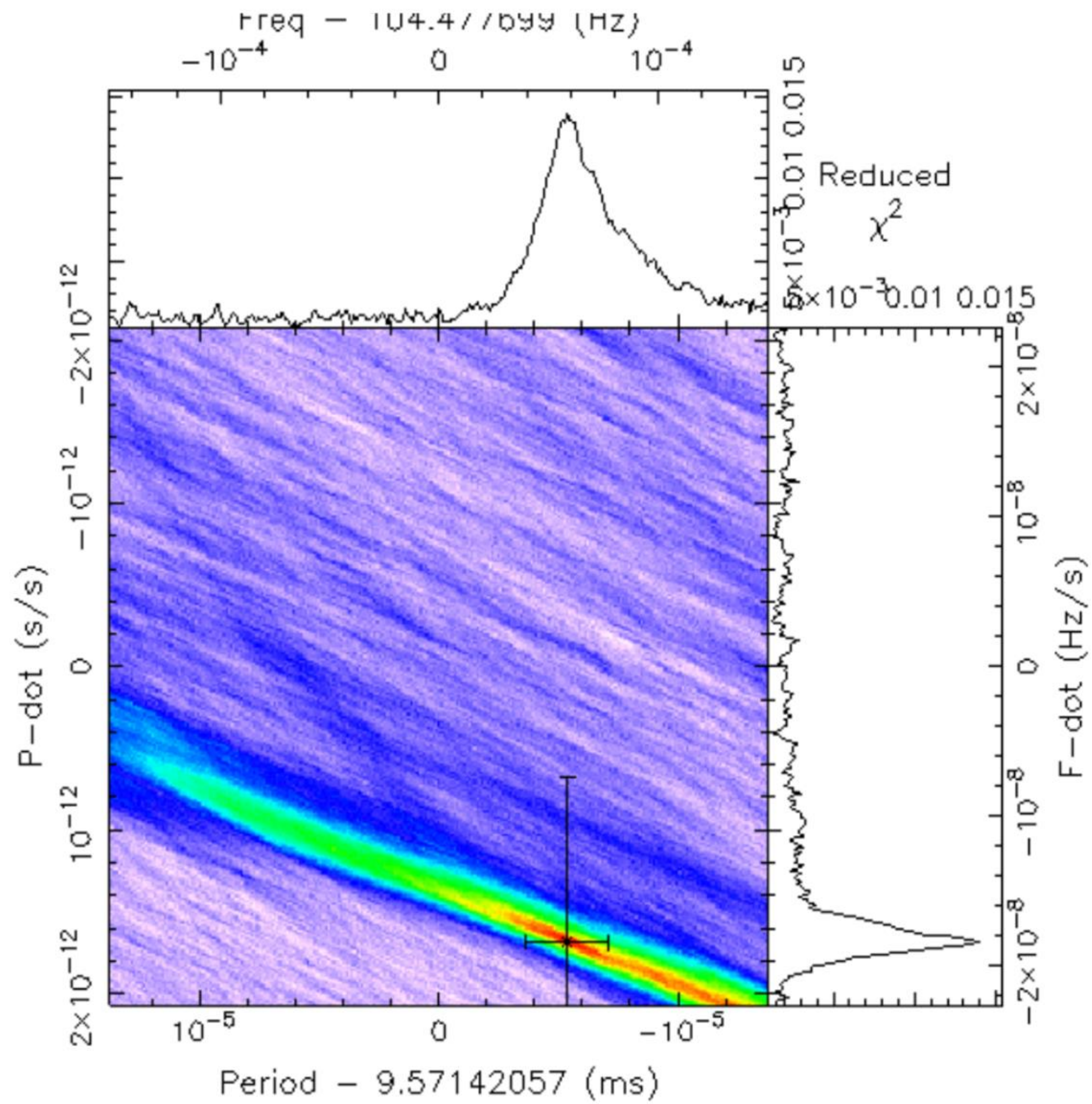


Finding a new binary pulsars?

Acceleration search

- Assumes the orbital period is significantly longer that the observation. The acceleration can be assumed to be close to constant during this observation.
- This constant acceleration can be compensated for and most of the power regained.
- This is essentially a 2D parameter search. (f and $\dot{f}$)

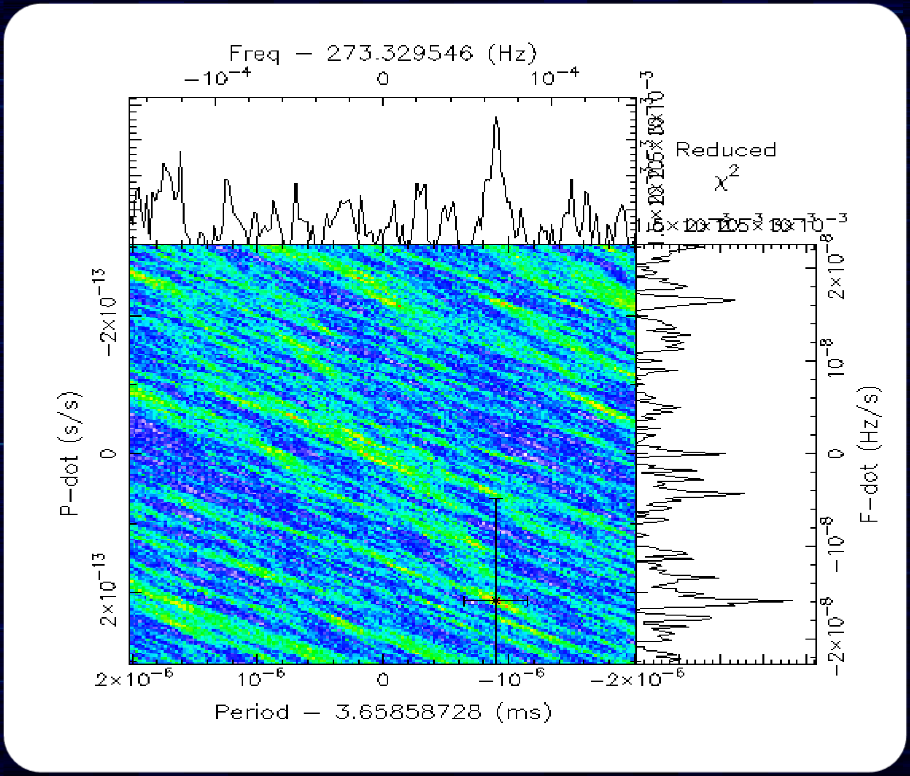
Acceleration search - f and $\dot{f}$



Searching for J1748-2446ae

- Ter AE has short orbital period (4 hours)
- Thus completes ~1.8 orbits during the 7.38 hour observation.
- It is this not detected with a acceleration search.

So what is next?



Create a f-dot plain

- Prepare kernels (make 2d array)
- Read fft
  - Prepare (1D data)
- Create f-dot plain
  - Multiply kernels with data
  - FFT
  - Powers
- Search (optional)

Preparer the kennels

- This is only don once!

- Calculate kernel columns – only dependent on width and height (Fresnel integrals)
- Place data (half and split)
- Fourier transform (y columns)

Prepare the input Data

- Read raw powers ~8K (float2)
- Calculate powers
- Calculate median
- Normalize raw powers (Using median of powers)
- Spread
- FFT

Create f-fdot

- Multiply Input (vector) by kernel column by column
- FFT data
- Chop ends
- Calculate powers
- Copy to f-fdot plain

Search f-fdot plain

- Find values above a threshold
- Compare to neibours (block 16 x16)
- If local maxima add to list

Add plains

- Scale x and y, sum “up” to highest harmonic.

8 Harmonics

- Create fundamental
  - Search fundamental
  - For stages (Powers of 2, ½, ¼, 1/8, ...)
    - For sub harmonics
      - Create
      - Sum with fundamental
    - Search

n Harmonics v1

• Make n input data sets	1 kernel
• Create n f-fdot plains	n kernels
• For stages	
add all subs	s kernels
search	1 kernel

n Harmonics v2

- Make n input data sets — 1 kernel
- Create f-fdot
  - Multiply — n kernels
  - – FFT's — ? kernels
- Sum and search — 1 kernel
  - – For stages
    - Create powers
    - Sum to shard memory
    - Search section of f-fdot plain