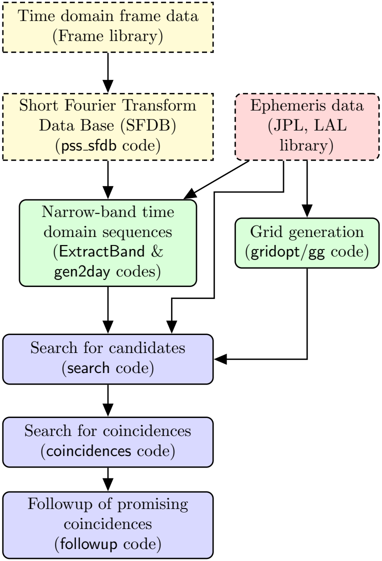
GPU version of the Polgraw all-sky time-domain F-statistic pipeline

Michał Bejger (Polgraw/Copernicus Center)

VDAS, 26.04.16

All-sky pipeline `https://github.com/mbejger/polgraw-allsky.git`



- ⋆ Input data generation (Raw time domain data $\sim$ *PB*)
- ⋆ Pre-processing $\rightarrow \sim$ *TB* (input time series, detector ephemerids and grid of parameters),
- ⋆ Stage 1: F-statistic search for candidate GW signals (the most time-consuming part of the pipeline)

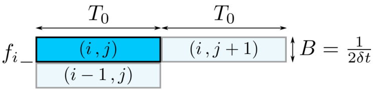
$\rightarrow$ 10^{10} candidates/detector, 100 TB of output.

- ⋆ Stage 2: Coincidences among candidate signals from different time segments,
- ⋆ Stage 3: Followup of interesting coincidences - evaluation of F-statistic along the whole data span.

Methods of data analysis

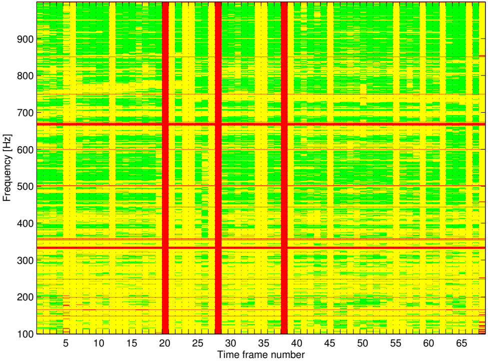
Computing power $\propto T_0^5 \log(T_0)$. Coherent search of $T_0 \simeq 1$ *yr* of data would require zettaFLOPS (10^{21} FLOPS) $\rightarrow$ currently impossible $\ddot{\frown}$

Solution: divide data into shorter length time frames ($T_0 \simeq 2$ days)



- ⋆ narrow frequency bands - sampling time $\delta t = 1/2B$, number of data points $N = T_0/\delta t \rightarrow N = 2T_0 B$

$\rightarrow$ feasible on a "normal" supercomputer.



Example search space (Virgo Science Run 1). Red: no data, yellow: bad data, green: good data.

Calculation of the F-statistic

To estimate how well the model matches with the data $x(t)$, we calculate $\mathcal{F}$,

$$\mathcal{F} = \frac{2}{S_0 T_0} \left(\frac{|F_a|^2}{\langle a^2 \rangle} + \frac{|F_b|^2}{\langle b^2 \rangle} \right)$$

where S_0 is the spectral density, T_0 is the observation time, and

$$F_a = \int_0^{T_0} x(t) a(t) \exp(-i\phi(t)) dt, F_b = \ldots$$

and $a(t)$, $b(t)$ are amplitude modulation functions (depend on the detector location and sky position of the source),

$$h_1(t) = a(t) \cos\phi(t), \quad h_2(t) = b(t) \cos\phi(t),$$

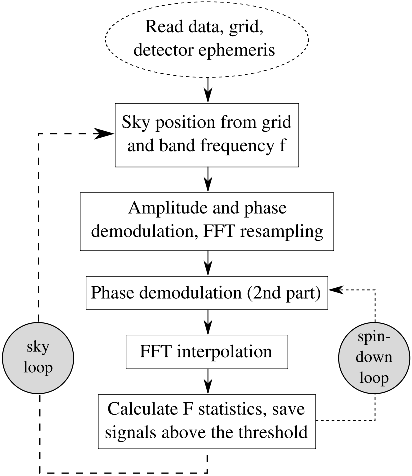
$$h_3(t) = a(t) \sin\phi(t), \quad h_4(t) = b(t) \sin\phi(t),$$

related to the model of the signal (h_i, $i = 1, \ldots, 4$)

$$h(t) = \sum_{i=1}^{4} A_i \, h_i(t).$$

For triaxial ellipsoid model: dependence on extrinsic $(h_0, \psi, \iota, \phi_0)$ and intrinsic $(f, \dot{f}, \alpha, \delta)$ parameters.

F-stat all-sky search description



Main parameters in coherent search for continuous wave signals:

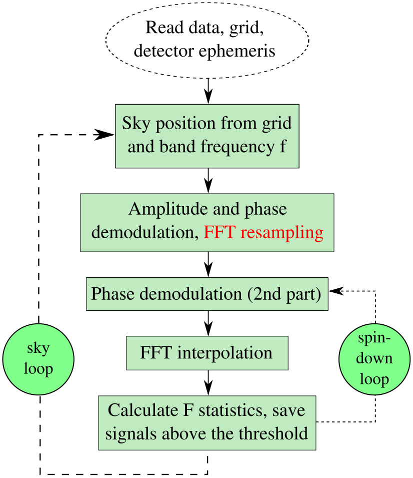
- ⋆ bandwidth 0.25 Hz
- ⋆ sampling time 2 s
- ⋆ data length N = 86164 (two sideral days)

⋆ 4D grid: α, δ, f, $\dot{f}$ - sky positions, frequency and spindown

⋆ Uses the F-statistic defined in Jaranowski, Królak & Schutz (1998), algorithm described and tested in Astone et al. (2010)

⋆ No. of F-statistic evaluations $\propto f^3$ (no. of sky positions $\propto f^2$, spindown $\propto f$)

F-stat all-sky search description



Basically the whole loop over sky (α, δ) can be computed in parallel since the sky positions are independent of each other

The majority of computing is spent on

- ⋆ calculating the phase (trigonometric functions, $\gtrapprox$ 20%)
- ⋆ FFT ($\gtrapprox$ 70%)

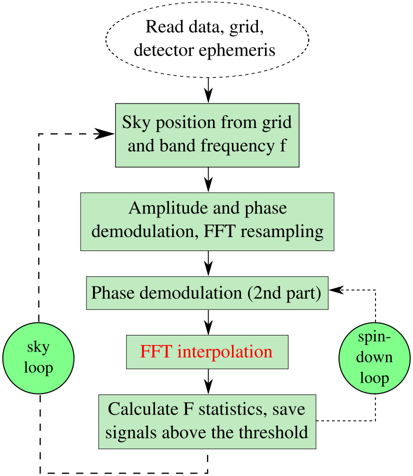
Efficient FFT requires 2^N data points ($N_{data} = 86164 < 2^{17}$) $\rightarrow$ padding with zeros to $N = 2^{17}$

FFT: resampling

- ⋆ Resampling to barycentric time - FFT and inverse:
  - ⋆ nearest-neighbour ($\simeq$ 5% error),
  - ⋆ splines ($\simeq$ 0.1% error)

The only part that has to be done in double-precision.

F-stat: parallelization strategy



- ⋆ How to do FFT with GPU:
  - ⋆ use CUDA cuFFT library:
    - ⌣ well-optimized (Cooley-Tukey, Bluestein), 1D/2D/3D double precision complex/real transforms, multiple transforms, in- and out-of-place transforms,
    - ⌢ cannot launch many instances at the same time (at least not with every card/CUDA version).
  - ⋆ write custom kernel for FFT, launch concurrently.
- ⋆ cuSPARSE (sparse matrix routines)

Results of implementation on GPUs

- $\star$ Input data loaded to device once. For each detector (V1, L1 & H1),
  - $\star$ time-series ($N \times \mathit{sizeof}(\mathit{double}) = 674$ KB)
  - $\star$ ephemerids ($3N \times \mathit{sizeof}(\mathit{double}) = 2$ MB)

  + a grid-generating matrix (388 B).
- $\star$ Sequence of kernels launched in a loop from CPU,
- $\star$ Time resampling done using double precision, everything else (main spindown loop) using single precision,
- $\star$ Asynchronous output transfer to host.

Current GPU results: $\sim \times 10$ speedup **with respect to the optimized CPU code**

Estimated time τ to match one template:

- $\star$ CPU (Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz) $\simeq 5 \times 10^{-3}$ s
- $\star$ GPU (GeForce GTX Titan) $\simeq 3 \times 10^{-4}$ s

Performance scaling - favorably for high frequencies (no. of spindowns $\propto f$).

Summary/references

We have two search codes for candidate signals for a network of detectors:

- ⋆ a well-optimized CPU code,
- ⋆ and a working GPU version that still needs some optimization (improve kernel concurency).

- P. Astone, K. M. Borkowski, P. Jaranowski, M. Piętka and A. Królak, PRD, **82**, 022005 (2010)
- `https://developer.nvidia.com/cuFFT`
- P. Jaranowski, A. Królak, and B. F. Schutz, PRD **58**, 063001 (1998).
- Polgraw-allsky github repository: `https://github.com/mbejger/polgraw-allsky.git`