

GPUs in gravitational wave research

Gergely Debreczeni
(Debreczeni.Gergely@wigner.mta.hu)

*Head of
Gravitationalphysics Research Group,
& GPU Laboratory
Wigner RCP*

*Computing Coordinator
Virgo Experiment*

Content

- Wigner GPU laboratory
- The Virgo experiment
- The way we work with GPUs
 - The Compute Wrapper , CUDA/OpenCL compute backends
- Analysis examples
 - Coalescing Compact Binaries
 - The matched-filter
 - Continuous Waves
 - The frequency Hough algorithm
- Plans with Tegra K1
- Conclusion

Content

- Wigner GPU laboratory
- The Virgo experiment
- The way we work with GPUs
 - The Compute Wrapper , CUDA/OpenCL compute backends
- Analysis examples
 - Coalescing Compact Binaries
 - The matched-filter
 - Continuous Waves
 - The frequency Hough algorithm
- Plans with Tegra K1
- Conclusion

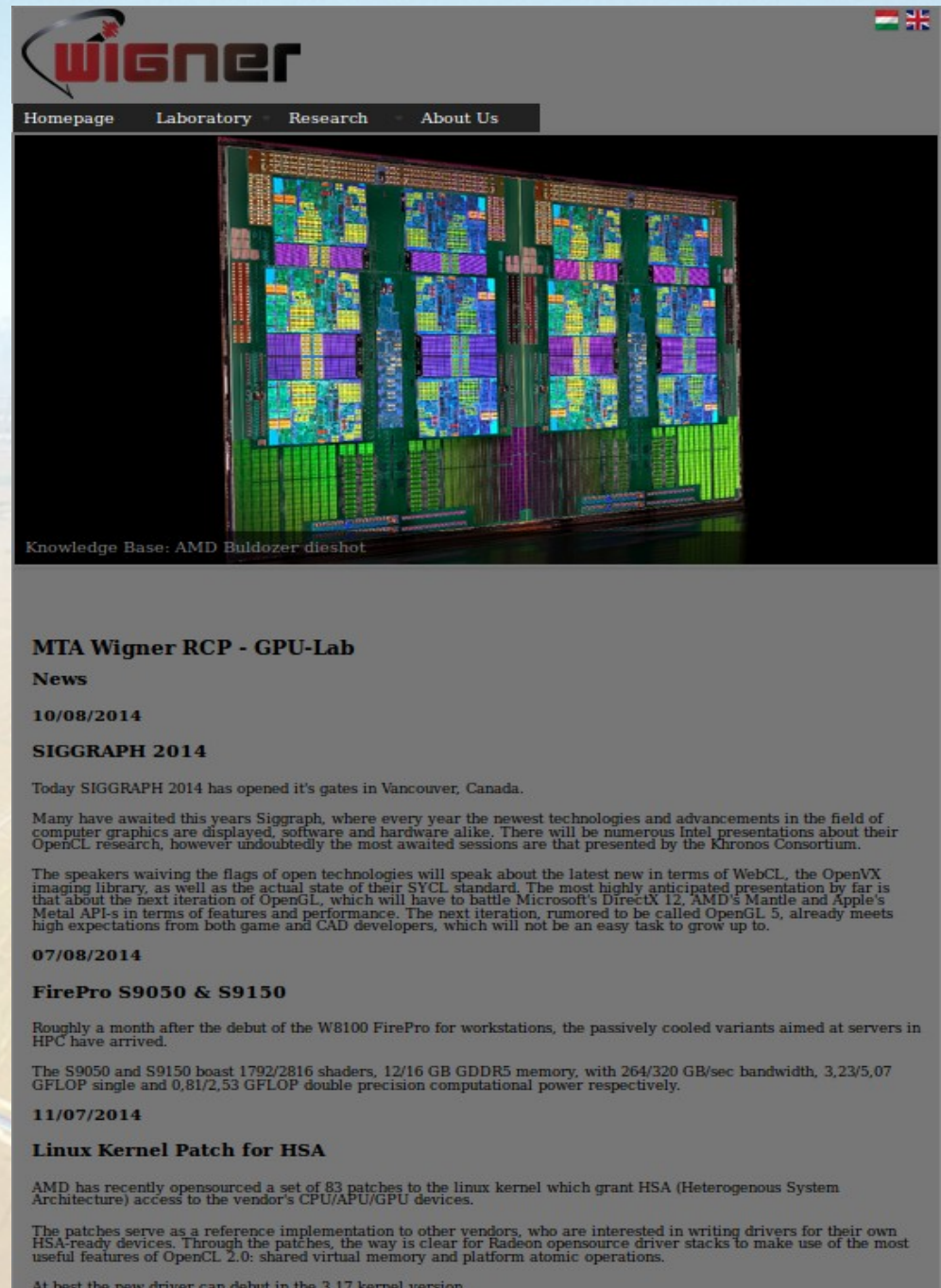
The Wigner GPU laboratory

<http://gpu.wigner.mta.hu>

- Small scale heterogenous cluster
- Serves as a playing ground for researchers in the Institute
- Support by experts:

Nagy-Egri Mate Ferenc

- Various cards: Nvidias, AMDs, Phis



wigner

Homepage Laboratory Research About Us

Knowledge Base: AMD Bulldozer dieshot

MTA Wigner RCP - GPU-Lab News

10/08/2014

SIGGRAPH 2014

Today SIGGRAPH 2014 has opened it's gates in Vancouver, Canada.

Many have awaited this years Siggraph, where every year the newest technologies and advancements in the field of computer graphics are displayed, software and hardware alike. There will be numerous Intel presentations about their OpenCL research, however undoubtedly the most awaited sessions are that presented by the Khronos Consortium.

The speakers waiving the flags of open technologies will speak about the latest new in terms of WebCL, the OpenVX imaging library, as well as the actual state of their SYCL standard. The most highly anticipated presentation by far is that about the next iteration of OpenGL, which will have to battle Microsoft's DirectX 12, AMD's Mantle and Apple's Metal API-s in terms of features and performance. The next iteration, rumored to be called OpenGL 5, already meets high expectations from both game and CAD developers, which will not be an easy task to grow up to.

07/08/2014

FirePro S9050 & S9150

Roughly a month after the debut of the W8100 FirePro for workstations, the passively cooled variants aimed at servers in HPC have arrived.

The S9050 and S9150 boast 1792/2816 shaders, 12/16 GB GDDR5 memory, with 264/320 GB/sec bandwidth, 3,23/5,07 GFLOP single and 0,81/2,53 GFLOP double precision computational power respectively.

11/07/2014

Linux Kernel Patch for HSA

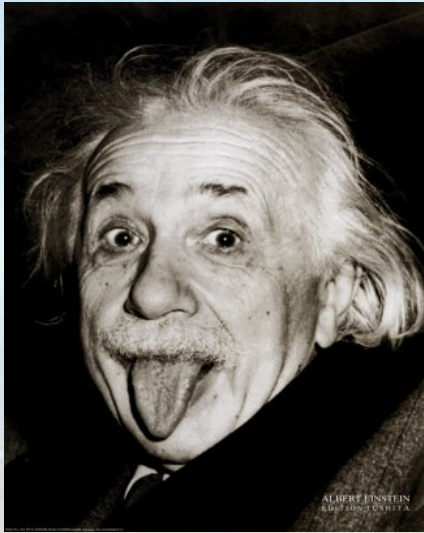
AMD has recently opensourced a set of 83 patches to the linux kernel which grant HSA (Heterogenous System Architecture) access to the vendor's CPU/APU/GPU devices.

The patches serve as a reference implementation to other vendors, who are interested in writing drivers for their own HSA-ready devices. Through the patches, the way is clear for Radeon opensource driver stacks to make use of the most useful features of OpenCL 2.0: shared virtual memory and platform atomic operations.

At best the new driver can debut in the 3.17 kernel version.

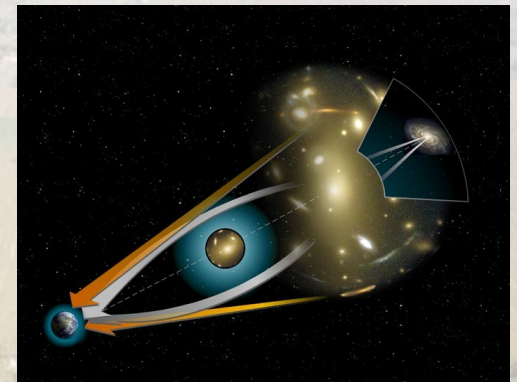
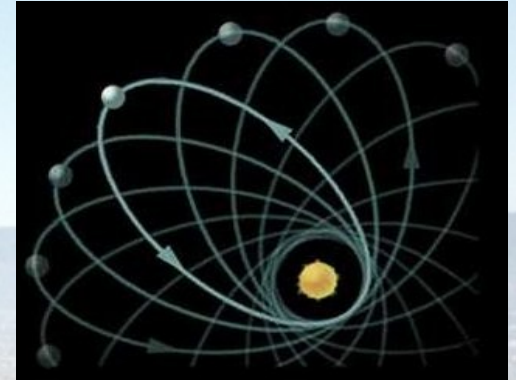
General Relativity

Observables



Experimentally proved:

1. Perihelium shift of planets
2. Gravitational lensing
3. Time dilatation/contraction in strong/weak gravitational fields



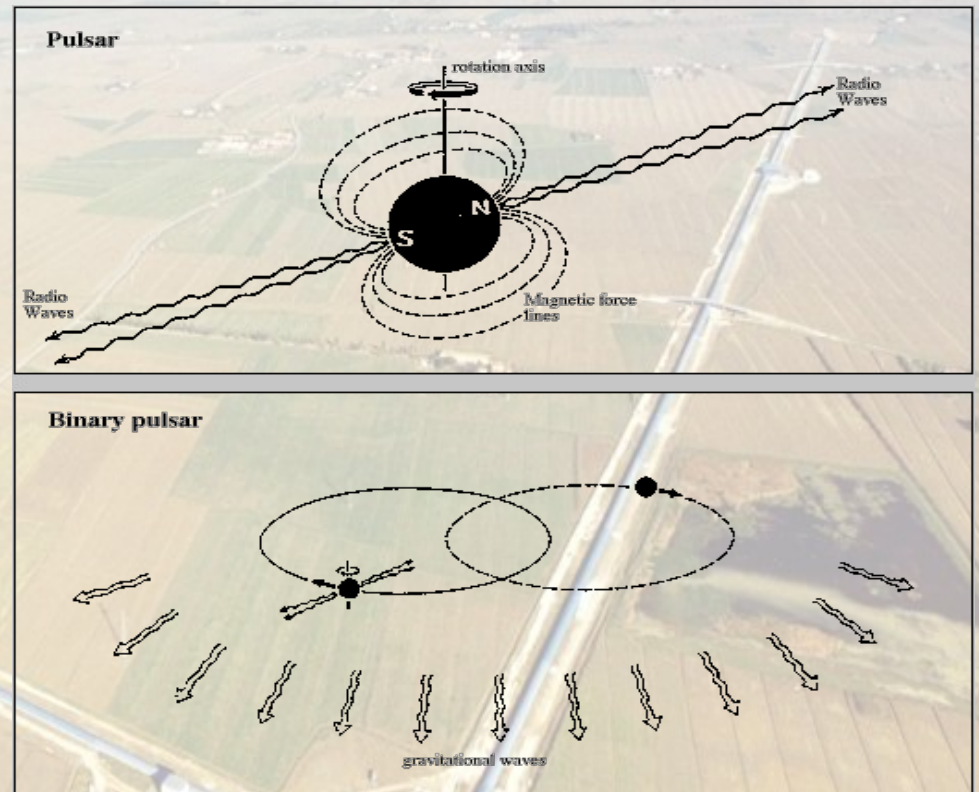
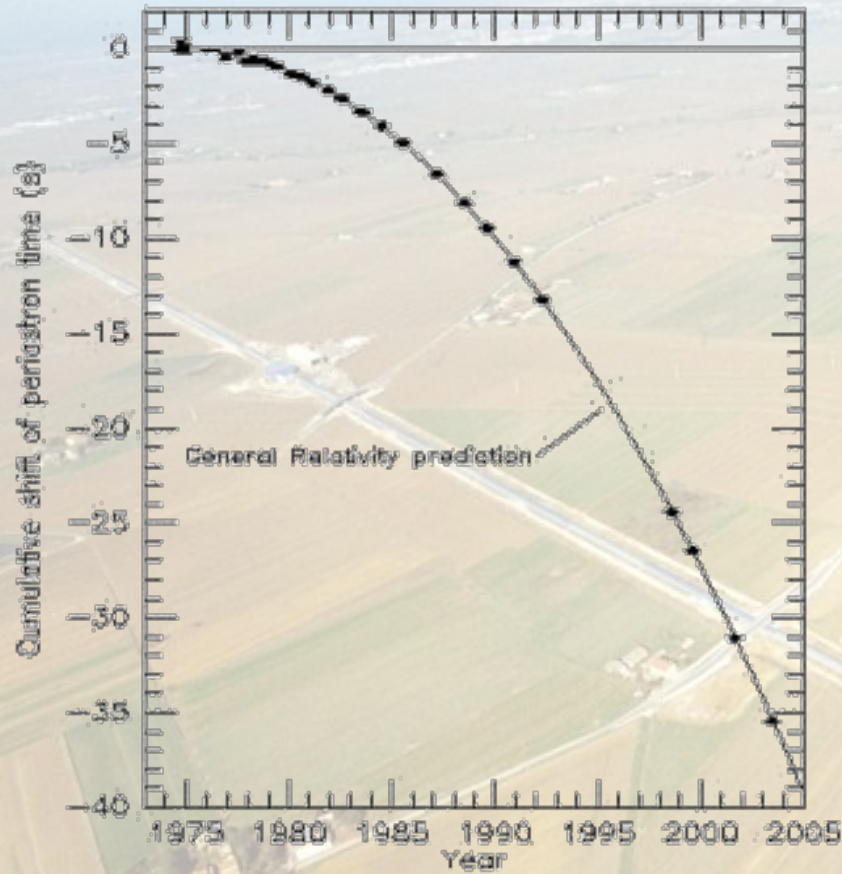
4.... and the existence of gravitational waves...

- Gravitational waves are ripples/waves of space-time itself created by certain energetic acceleration of huge masses.
- As of today, we have only indirect – but quite convincing – proof of their existence



A Hulse-Taylor pulsar

- The „Holy Grail” of gravitational wave researchers
- Discovered in 1974 (Russel Hulse, Joseph Taylor)
- 1993 Shared Nobel-prize
- Probably two neutron star
- 3,1 mm shrink for every orbit cycle
- 59 ms-os pulsating period
- 7,75 hour orbital period

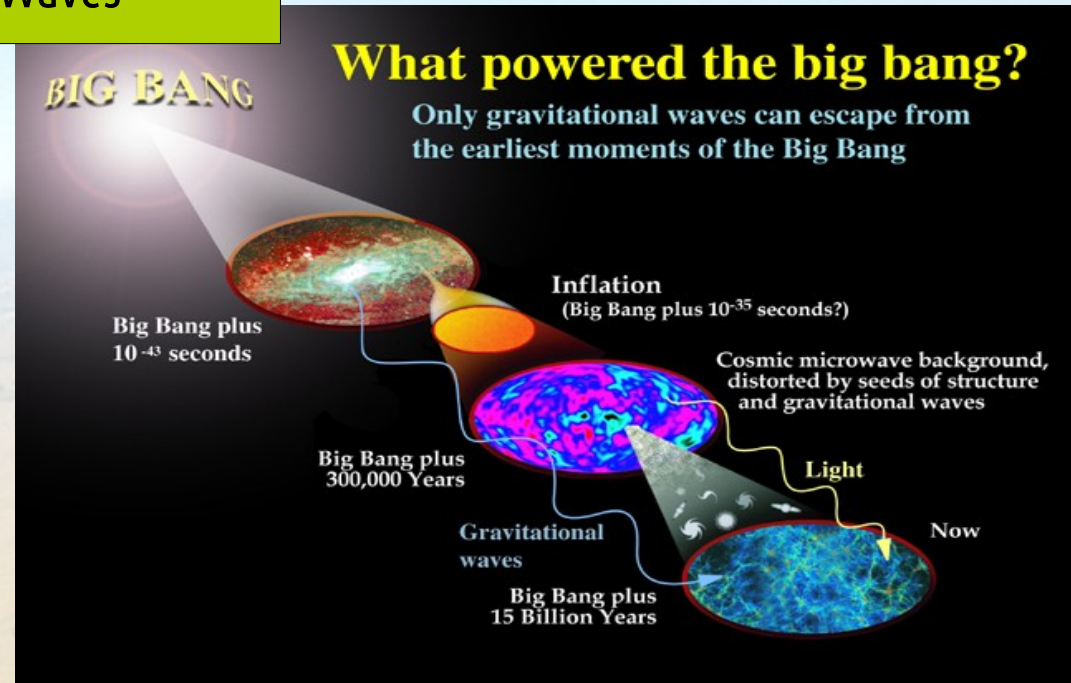


Possible GW sources

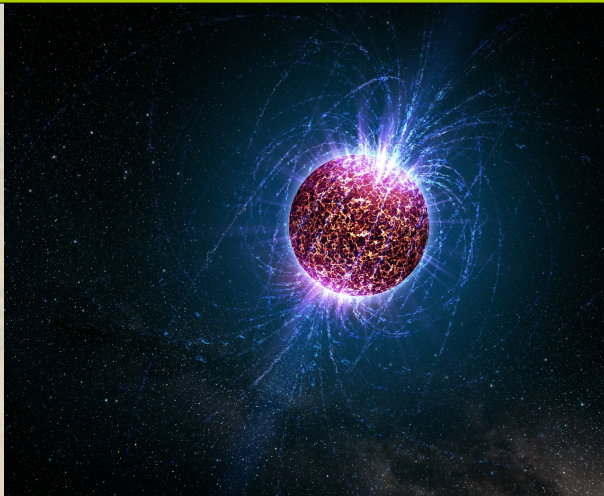
Compact Coalescing Binaries



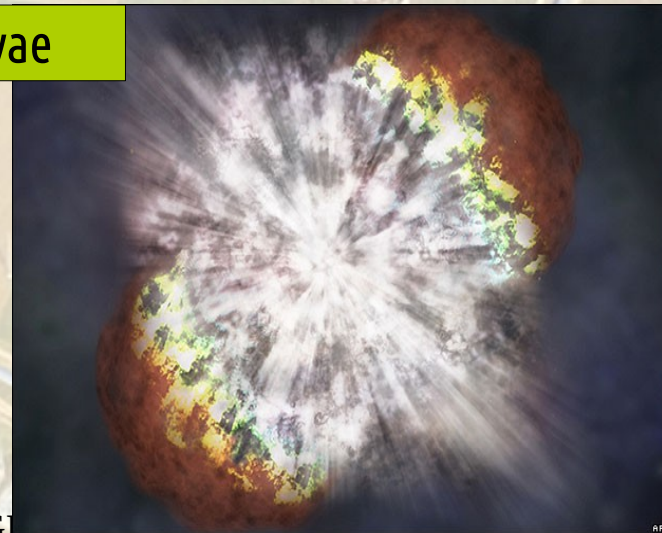
Stochastic Waves



Pulsars

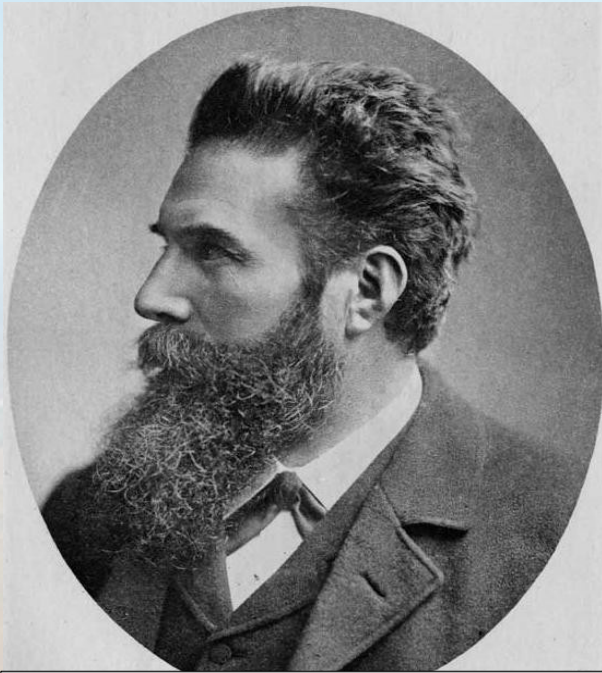


Supernovae

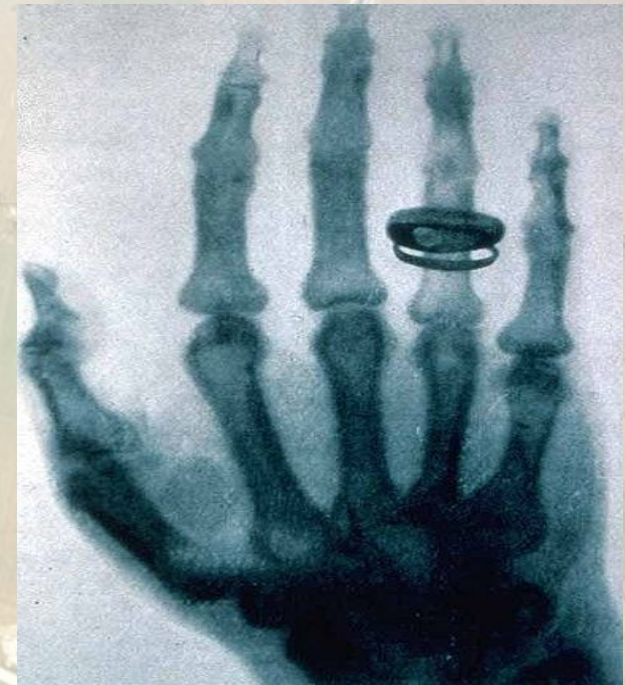


Why interesting ?

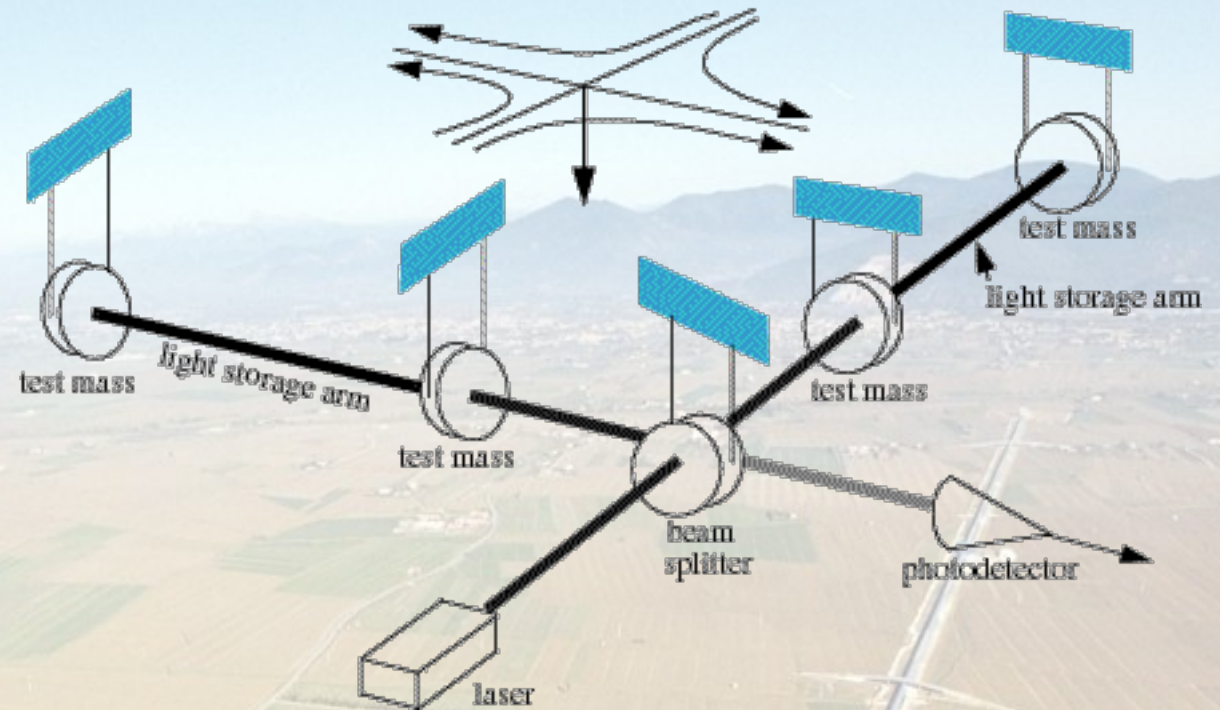
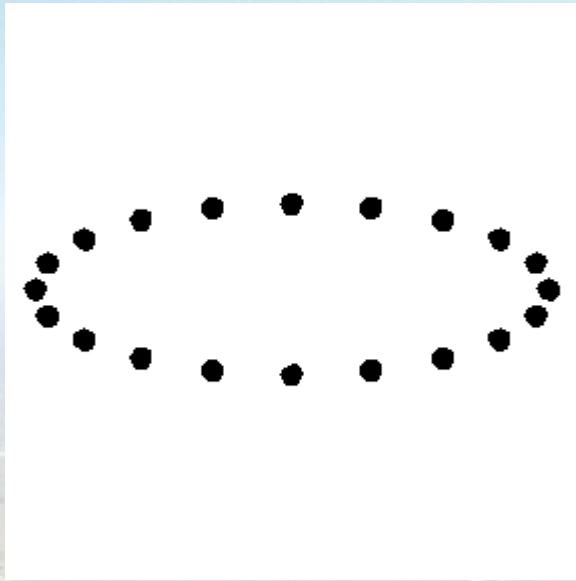
- Completely new face of astrophysical objects, of Universe
- One can observe so far hidden phenomenae
- Would be the new „sense-organ” of Humanity
- Testing theories



Wilhelm Conrad Roentgen
1845 - 1923



Interferometers

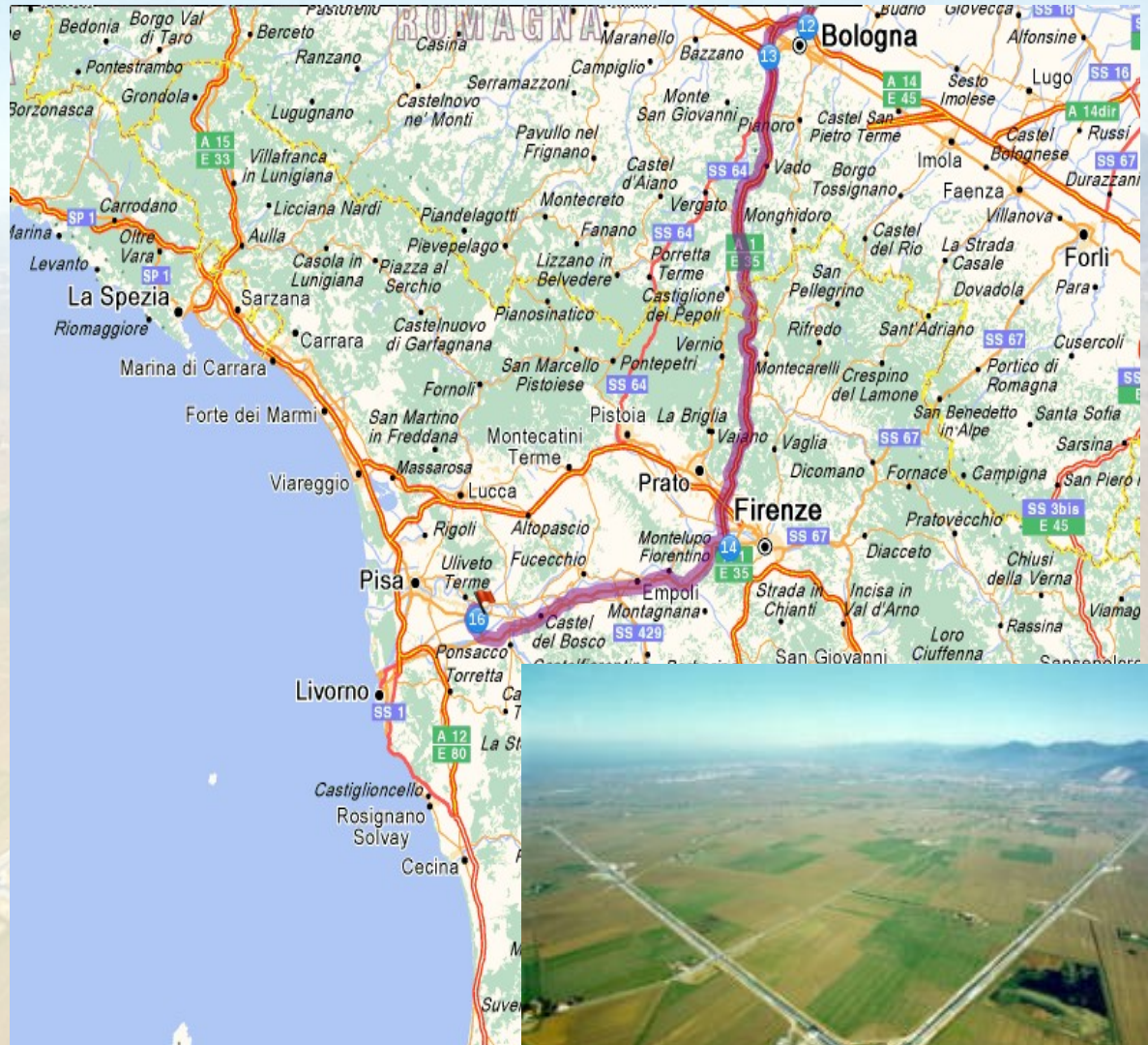


- Incident GW changes the arm length
- Light intensity changes in the interference point
- The outcome of the measurement is the difference of arm lengths
- The world's most precise relative measurement with a precision of 10^{-18} .
- Huge collection of environmental noise have to be cancelled
- Isolated detectors are needed for coincidence detection of same event (see LIGO).

The Virgo experiment



- A Virgo detektor az EGO (European Gravitational Observatory) telephelyén az Arno folyó síkságán Pisa melletti Cascina-ban található
- Az építkezés 2003-ban fejeződött be
- A Wigner FK Rácz István vezetésével 2008-ban csatlakozott a kísérlethez
- A francia - olasz együttműködésként induló kollaborációnak ma már magyar és lengyel tagjai is vannak.



The GWTools project

The GWtools project

<http://gwtools.org>

GWTools - the C++/OpenCL [1] based Gravitational Wave data analysis Toolkit - is an algorithm library aimed to bring the immense computing power of emerging many-core architectures - such as GPUs, APUs and many-core CPUs - to the service of gravitational wave research. GWTools is a general algorithm library intended to provide modular building blocks for various application targeting the computationally challenging components of GW data analysis pipelines.



wigner Welcome to the GWTools web site
The Gravitational Wave Data Analysis Toolkit

Max-Planck-Institut für Gravitationsphysik (Albert-Einstein-Institut)

Main page | Apps | Docs | Downloads | Links | News | Career | Devel | Contacts

GWtools is a **Gravitational Wave** data analysis **Toolkit** which seamlessly brings the immense computing power provided by the emerging many-core architectures - such as GPUs, APUs, MICs - to the service of scientists of the era of advanced detectors...

[...more](#)

VIRGO

The GWtools project
The GWTools - Gravitational Wave data analysis Toolkit - is jointly developed by the Virgo group of the Wigner Research Centre for Physics of the Hungarian Academy of Sciences, Budapest and the LIGO Group of Max Planck Institute for Gravitational Physics, Hannover. The goal of the project...

[...more](#)

LIGO

Technical description
GWTools is an implementation of many data analysis algorithms widely used within the LIGO-Virgo gravitational wave research community. Based on C++ and OpenCL, it is modular, portable across operating systems and hardware platforms and compatible with all major batch systems. The OpenCL implementation inherently allows the automated and invisible parallelisation of the various routines on CPUs as well as on GPUs.

People
We have already received many contributions from people with expertise in various fields, such as software engineers, data analysts, theoretical and experimental physicist together with new requests for various analysis groups for applications. Current team members are ...

[...more](#)

The Compute Wrapper - features

- Forever ongoing discussion about cards and programming languages
- We created a Compute Wrapper (CW) and using multiple backend (CUDA / OpenCL / CPU / etc..)
- Development and testing happens in OpenCL, than the code ported to the appropriate backend.
- OpenCL → CUDA kernel translation is trivial (almost semi-automatic)
- CUDA → OpenCL can be problematic in some cases and requires special attention
- Built-in GL interop
- The Compute Wrapper also provides a
 - thread pool
 - a host side scheduler with
 - relational workflow handling
 - for CPU side optimisation for optimal load balancing.

The Compute Wrapper – code example

```
#include <ComputeWrapper/include/ProgramManager.h>
#include <ComputeWrapper/include/Factory.h>
#include <ComputeWrapper/include/IBuffer.h>
#include <System/include/ModulParameterFactory.h>
#include <Utils/include/ImageTypes.h>

int main() {

    // Host side objects and problem size
    math::Size m_size(1024,2048);
    float * h_buffer = new float[m_size.getCount()];

    // Create device objects
    cw::IBuffer * m_buffer = cw::Factory::createImage2D(m_size, IMAGE_UCHAR_C4, MEM_READ_WRITE);
    cw::IImage * m_image = cw::Factory::createBuffer(m_size.getCount() * sizeof(float), MEM_READ_ONLY | MEM_COPY_HOST_PTR, h_buffer);

    // Create a program
    m_program_id = FIND_PROGRAM_ID("hough_transform");

    // Kernel arguments
    cw::cwInt arg1(10);
    cw::cwInt arg2(20);

    // Call the kernel
    PROGRAM_BY_ID(m_program_id)("integral_map", cw::Range(global_size_x, global_size_y), cw::Range(32,23), 4, m_buffer, m_image, &arg1, &arg2);

    // Finish every kernel execution
    cw::ContextManager::FinishQueue();

    // Read back the results
    m_buffer->readBuffer(h_buffer);

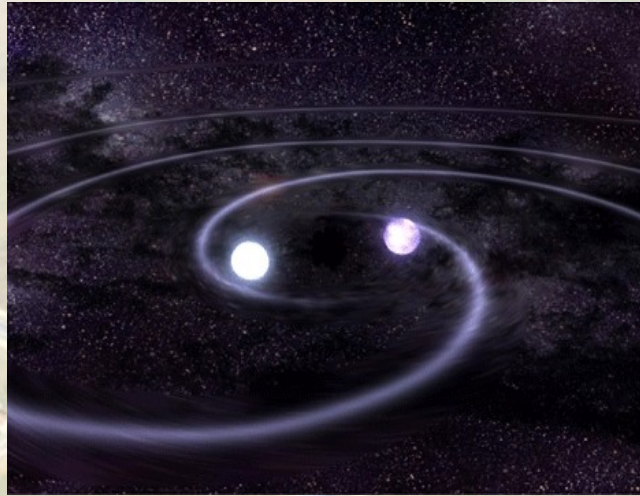
    // Exiting
    return 0;
}
```

About gravitational wave data

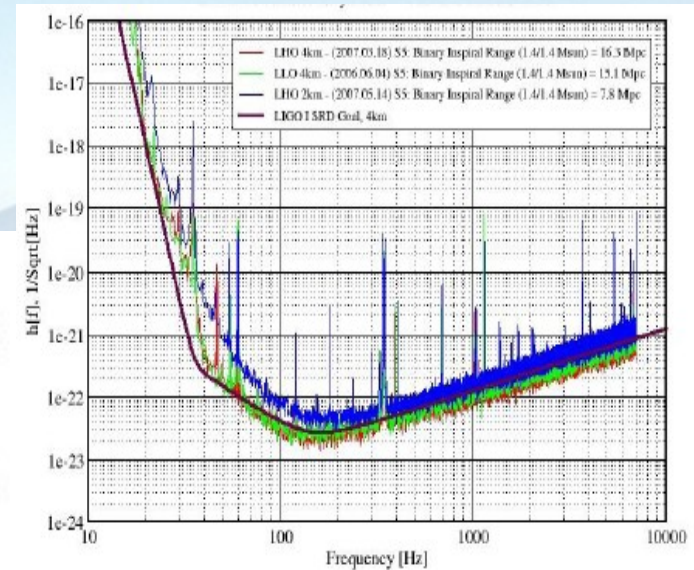
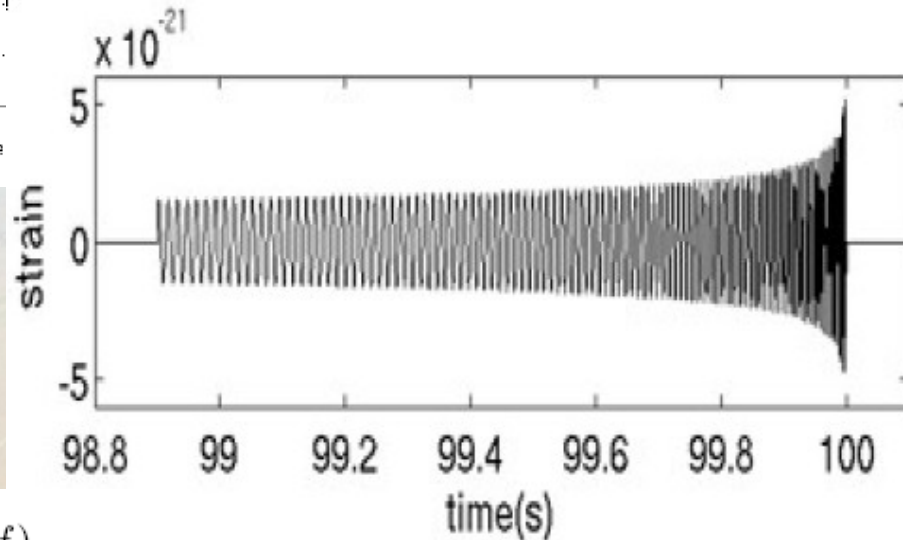
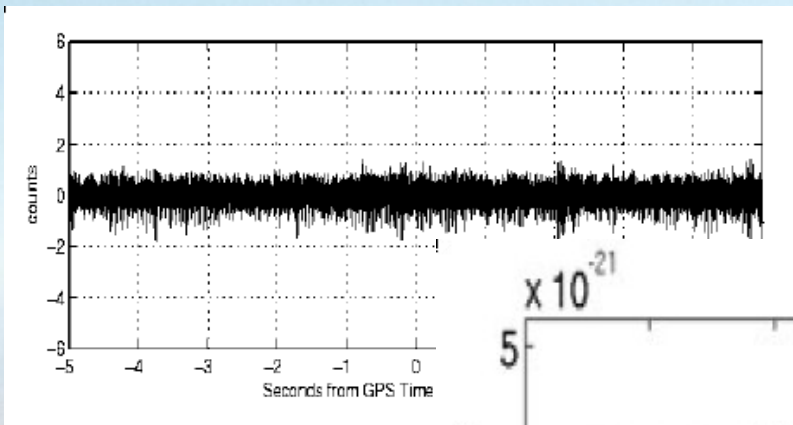
- GW data analysis is compute intensive not data intensive
- By quantity it is much less than LHC data: $O(100-300 \text{ TB} / \text{detector} / \text{year})$
- However, it's arithmetic density is much higher
- Available computing power directly translates to detector sensitivity !
 - For CW searches sensitivity goes as $\sim 1.0 / \sqrt{T}$, while necessary computing power is $\sim \exp(T)$
 - For CBC \sim number of background estimation and lowering clustering thresholds $\sim N$
- In contrast to HEP experiments, events cannot be regenerated
- Quick, low latency analysis are necessary for externally triggered searches
- Order of CPU cores $\sim 20K$
- Order of GPUs ~ 300

Analysis example I:

Compact Binary Coalescence



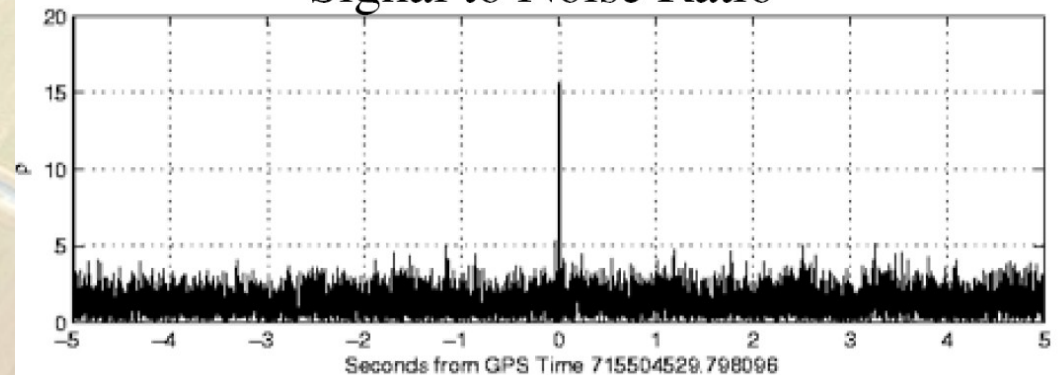
Compact Binary Coalescence



$$(A, B) = \int \frac{A^*(f)B(f)}{S_r(f)} df,$$

$$\text{SNR} = \frac{z}{\sqrt{\langle(\delta z)^2\rangle}} = \frac{(\tilde{Q}, \tilde{s})}{\sqrt{(\tilde{Q}, \tilde{Q})}},$$

Signal to Noise Ratio



Compact Binary Coalescence

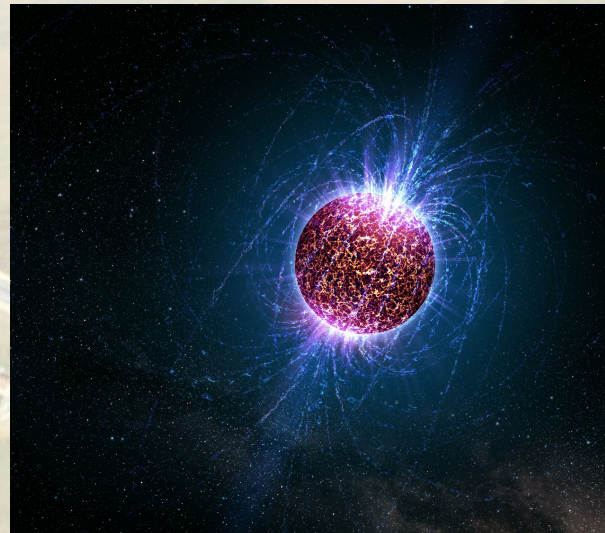
Algorithmic steps of the analysis:

- Spectral density calculation (OK)
 - Theoretical template generation (OK)
 - Template filtering
 - vector multiplication (OK)
 - Inverse FFT (OK)
 - Maximum finding (OK)
 - Clustering (can be challenging depending on thresholds)
 - Post-processing (on CPU)
- Golden application of GW searches
 - Runs on AMD and Nvidia
 - CUDA FFT is superior
 - Typical data segment length $10^{20} - 10^{24}$
 - Runs mainly on Teslas and cheap GTX cards

Almost optimal in all sense (occupancy, throughput, CPU/GPU balance, etc)!

Analysis example II:

Continuous Waves: Pulsars



CW: The Hough search I

- Searching for **neutron stars with unknown frequency** evolution on the sky
- Nor the frequency neither its time derivatives (spin down parameters) are known -> has to be scanned
- Basic steps of the method:
 - Segment the data (T_{total}) into coherent segments (T_{coh}), where the frequency can be approximated to be constant (order of 30 min, but depend on the actual frequency bin).
 - Calculate the power spectra of the entire data and normalize the Fourier Transformed segments with it. The normalized FFT domain segments are then thresholded -> converted to 0s and 1s -> the peakmap is obtained in the time observed - frequency plane
 - For each position in the sky the peakmap is doppler de-modulated to obtain the (time - real frequency) peakmap. The happens on-the-fly, no need to store them.

CW: The Hough search II

- Taking into account only the first spindown parameter d , (i.e. assuming linear relationship) one can write up the following equation:

$$f_{\text{instantaneous}} = f_0 - d * (t - t_0)$$

that is a straight line in the $f_0 - d$ plane.

- Since we have a finite frequency resolution (df) this equation becomes

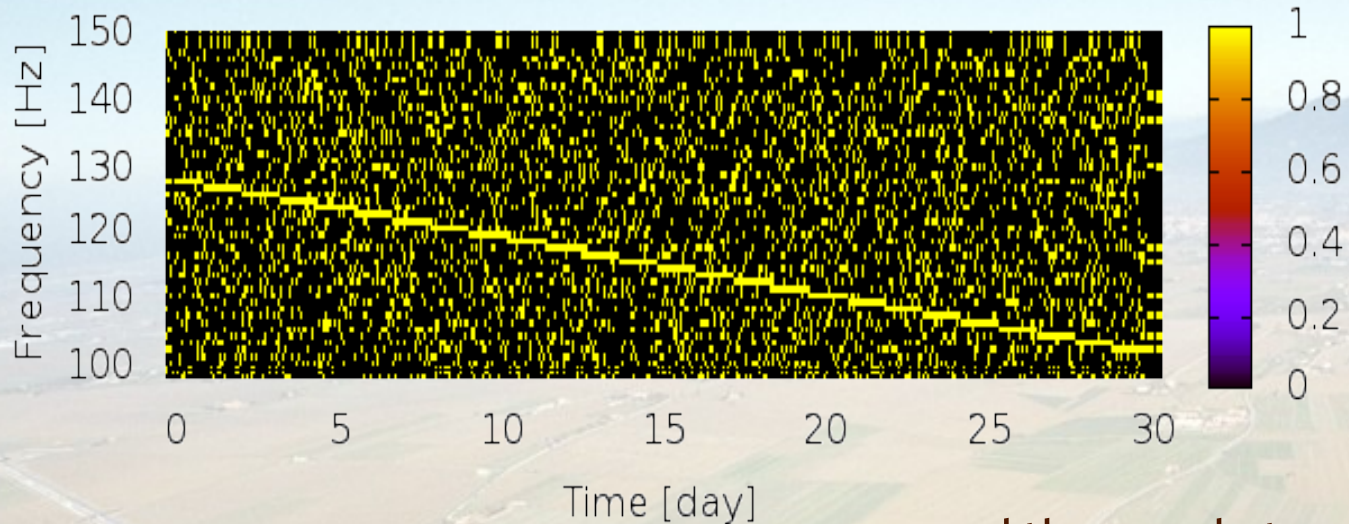
$$-f_0 / (t - t_0) + (f - df / 2) / (t - t_0) < d < -f_0 / (t - t_0) + (f + df / 2) / (t - t_0)$$

i.e, two straight line on the plane

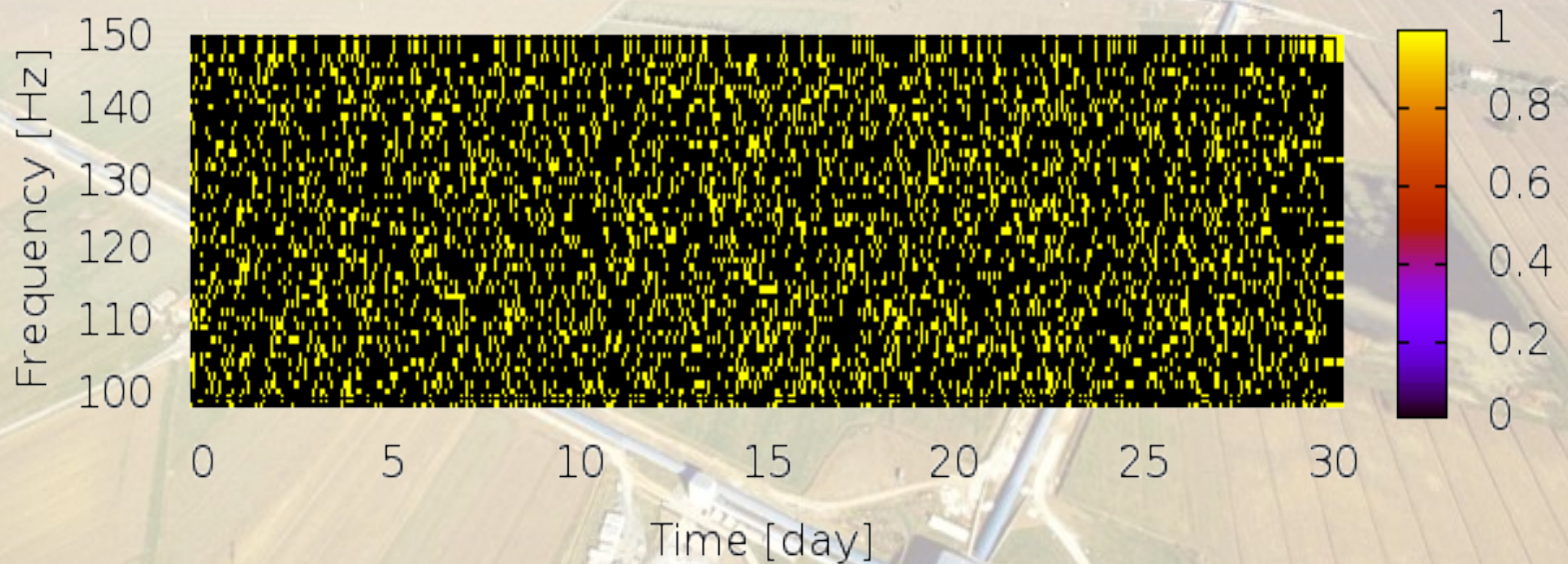
- For every point in the peak map (every t , $f_{\text{instantaneous}}$ value) draw a line in the f_0, d plane and increase the value of each bin by 1 which is intersected by the line, i.e. a 2D histogram.
- There are tricks
 - to increase the resolution one uses oversampled frequency bins
 - and to decrease computational cost one uses differential Hough maps when drawing lines with 1s and -1s and integrating only at the very end. This save significant computational cost, since it not necessary to shade / fill large areas in memory.

CW: The Hough algorithm: The peakmap

- A peakmap example with strong signal

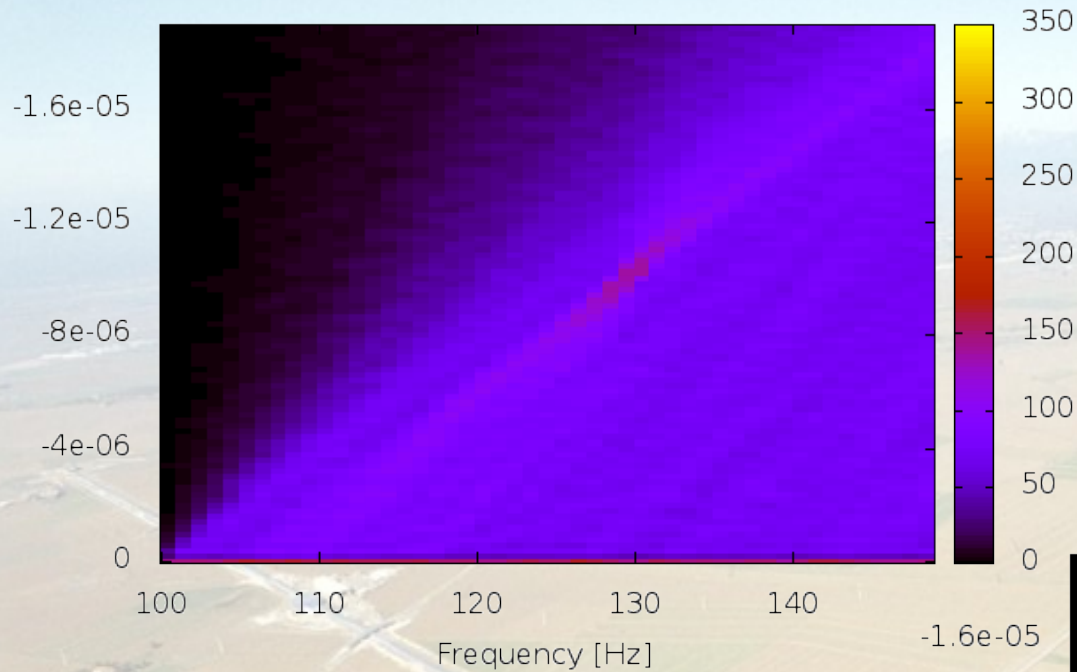


... and the same but weaker signal

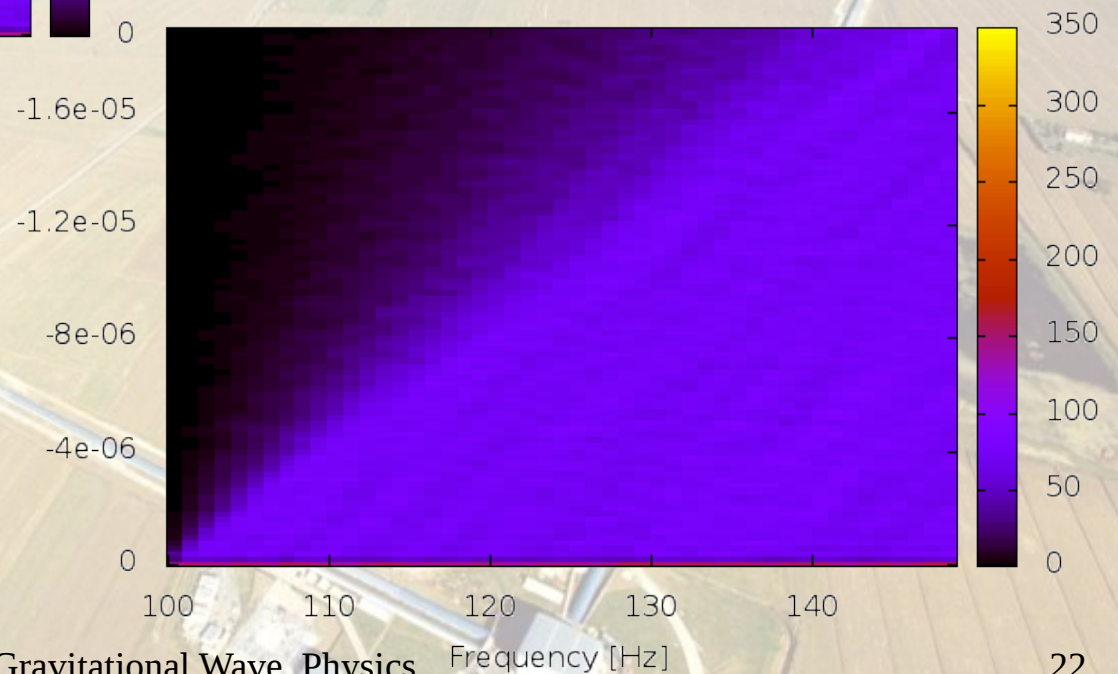


CW: The Hough algorithm: The Hough - map

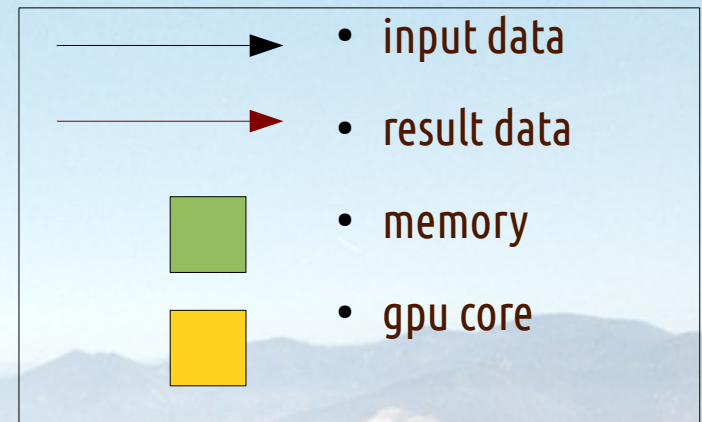
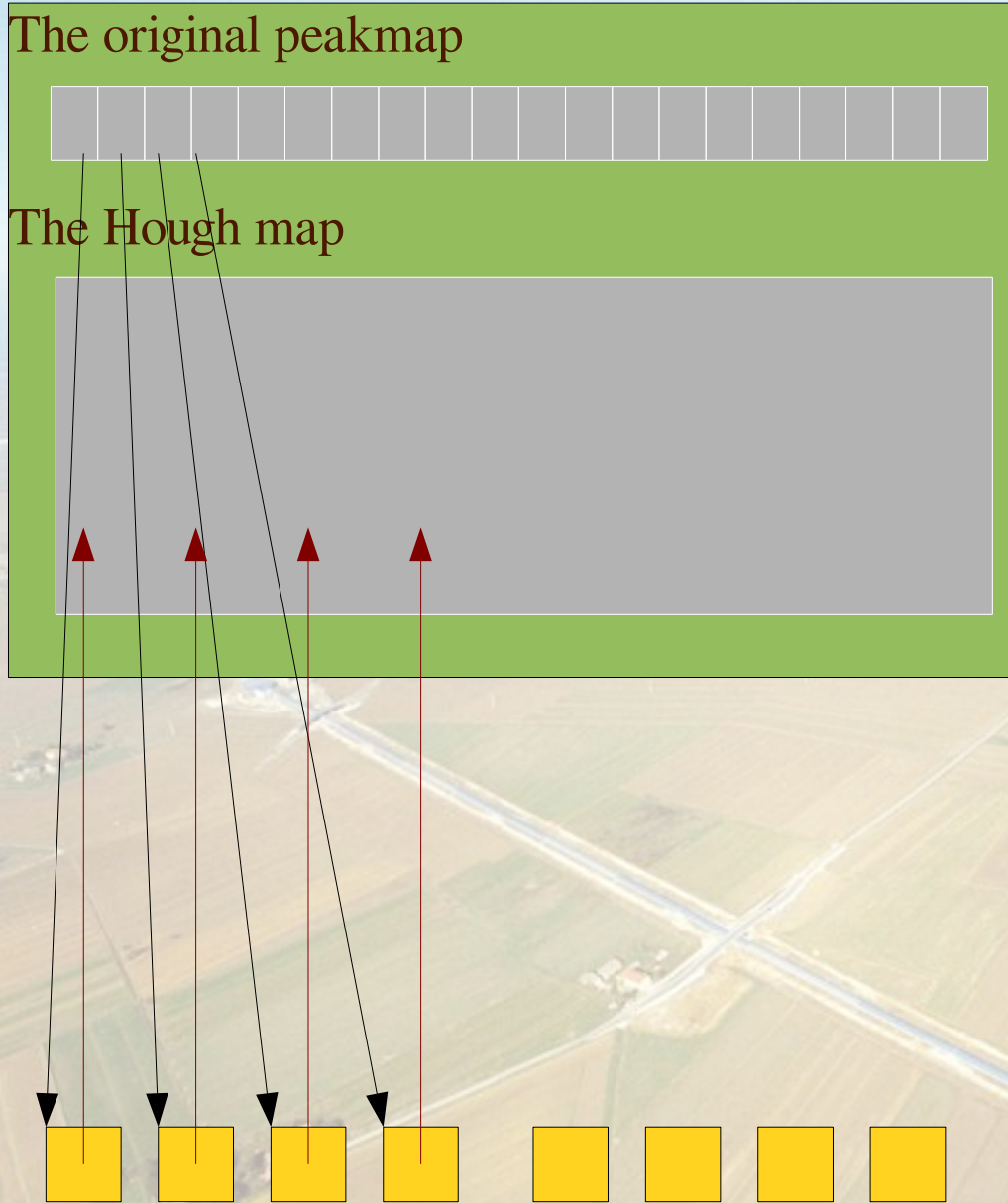
- The Hough map of the previous peakmap with with the weak signal



... and the same without signal

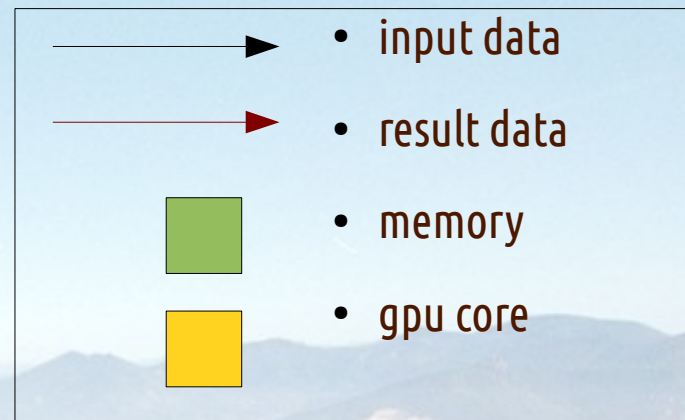
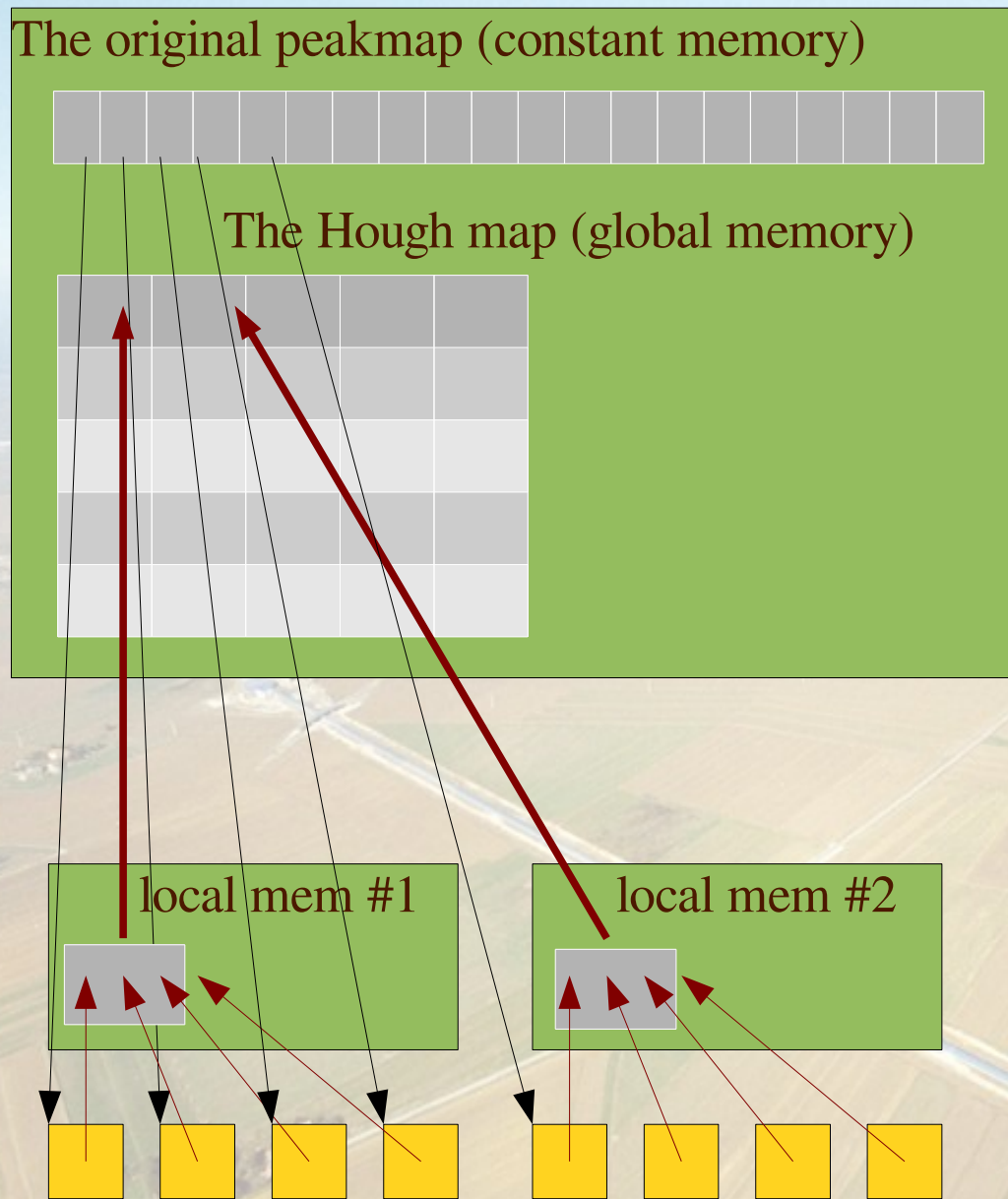


A possible implementation



- Each core reads a (some) peak
- Draws the corresponding line on the Hough map
- Problems
 - looping through on global memory is slow
 - intersection of lines requires atomic operations

A better implementation



- Hough map is partitioned
- Partitions are filled in local memory and copied back to global and the end
- All the line which intersects the given partition are drawn
 - much faster
 - no atomic operations involved

Continuous Waves: Algorithmic steps

Algorithmic steps of the analysis:

- FFT coherent segments (OK)
- Thresholding (OK)
- Differential Hough-map (OK)
- Integral Hough-map (OK)
- Maximum finding, clustering (can be tricky)

By now also very much optimized

Future (?) and testing: Tegra K1

- Good experiences with Tegra K1
 - Enables for unified memory access
 - (framework handles it)
 - 192 compute core
 - 4 ARM cores with NEON
 - Ubuntu Linux on devboard
- Low power consumption
- CUDA compiles quite OK, profiling, debugging sometime difficult
- Considering for future use



Conclusions

- GPUs are optimal for GW analysis
- With proper framework multi-language, multi-platform development does not require additional manpower
- GW analysis is kind of prepared for the challenges of forthcoming years