Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
ICSC Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

ICSC
Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

Boosting unmodeled searches of gravitational wave transients

Giacomo Principe on behalf of the Uni. Trieste GW group: Edoardo Milotti, Agata Trovato, Giacomo Principe, Andrea Virtuoso, Riccardo Felicetti, Giuseppe Troian; collaborating with Francesco Salemi and Marco Drago (Uni. Roma 1)

ICSC Spoke 2 Annual Meeting - 20 December 2023 Casalecchio di Reno (BO)

Unmodelled GW events

GW universe is not only Binary systems

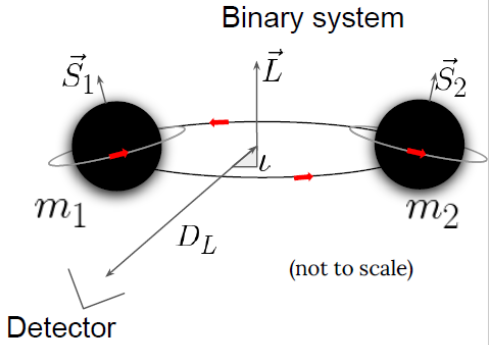
(only Compact Binary Coalescence detected so far)

- GW burst expected also for other sources:

  (Supernovae, GRB/FRB, Neutron Star/Magnetar flares,?)
- We do not have models for any possible mechanism
- Must be ready for the unexpected
- Search for unknown transients / **Burst**

GW Burst search Challenges:

1. Detection
   - Lack of models, open to all possible signals
   - Distinguish the real signal from detector noise
2. Source identification
   - Waveform shape depends on generating process
3. Sky localization
   - Search of counterparts



All sky unmodelled search | **model–informed search** | **matched-filter search**

assumptions on the GW waveform

cWB: Search for unmodelled GW events

cWB is an open source software for gravitational-wave data analysis

cWB is weakly-modelled algorithm used to search for GW bursts and compact object coalescences, and to reconstruct GW waveforms with minimal assumptions.

cWB detects the excess of signal power that is coherent in the network of detectors.

https://gwburst.gitlab.io

Abbott, R., et al. "All-sky search for short gravitational-wave bursts in the third Advanced LIGO and Advanced Virgo run." (2021)]
Salemi, F., et al. "Wider look at the gravitational-wave transients from GWTC-1 using an unmodeled reconstruction method." (2019).

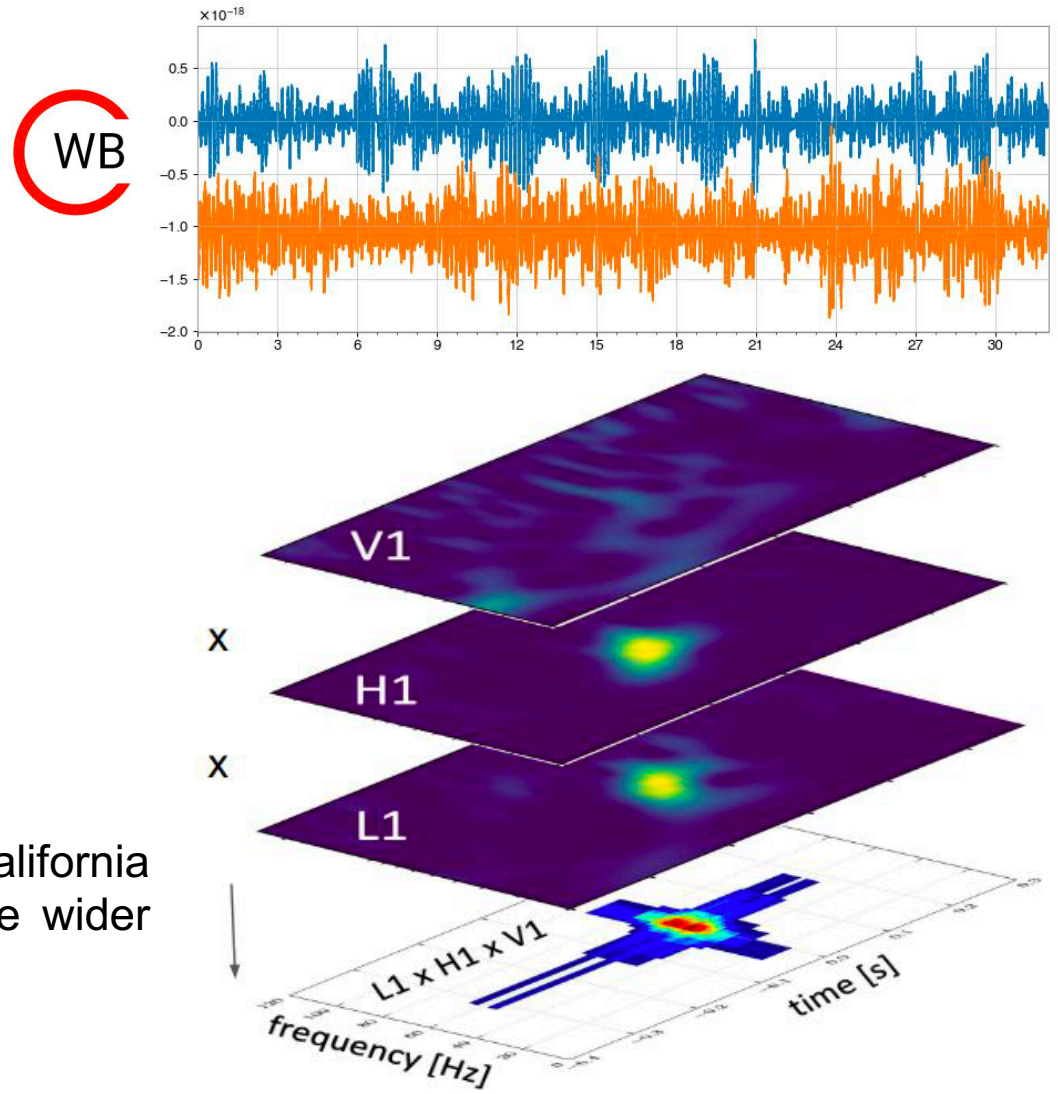
PycWB: a user-friendly, modularized Python package for GW burst search based on the core function of cWB.
https://arxiv.org/abs/2308.08639 - https://pypi.org/project/PycWB/0.18.2/

cWB workflow

cWB pipeline workflow:

1. *Time-frequency representation*
2. *Whitening*
3. *Selection of most energetic pixels*
4. *Coherent analysis (maximum likelihood)*
5. *Compute ranking statistic to each trigger*

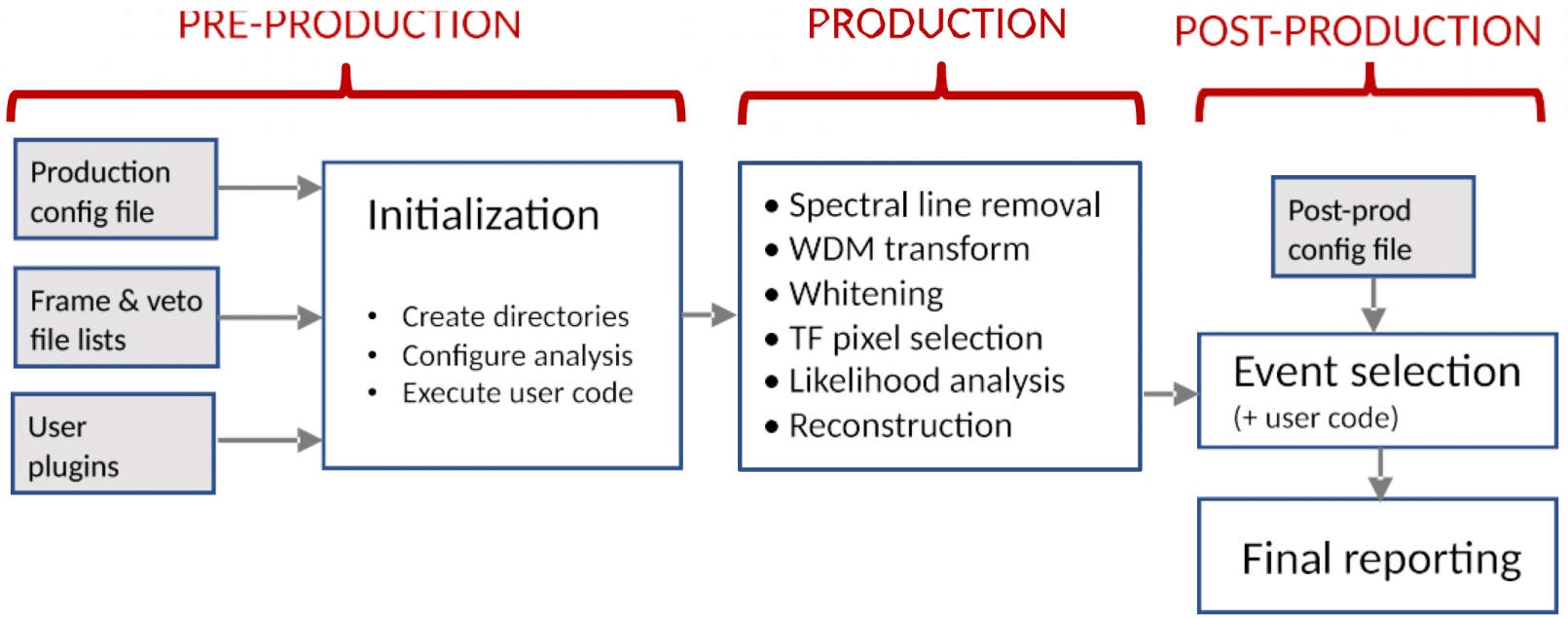
cWB almost exclusively runs on CIT (HPC center at the California Institute of Technology) which is often overburdened by the wider GW community.



cWB and computations

Because of its non-stationarity and non-Gaussianity, the *background noise must be empirically estimated* (i.e. repeating the search using *time slides*). The computational burden of this process scales linearly with the level of accuracy needed at a given *false alarm rate*. The calculation of the likelihood is the most time-consuming part of the code, and its efficiency is currently boosted by:

1. computing an approximate version of the likelihood early on in the process,
2. using the approximate likelihood to select only promising candidate events and proceed to further refining steps (hierarchical approach),
3. low-level manual optimization of the code targeting single-CPU x64 architectures.



Porting cWB-2G on CINECA cluster (and machine learning features)

We plan to supersede these computing burden limitations (which are particularly relevant for the likelihood analysis) by porting and adapting the pipeline code to the GPUs available at the Centro Nazionale HPC.

cWB is coded in C/C++ for efficiency, with vectorizations wherever possible.

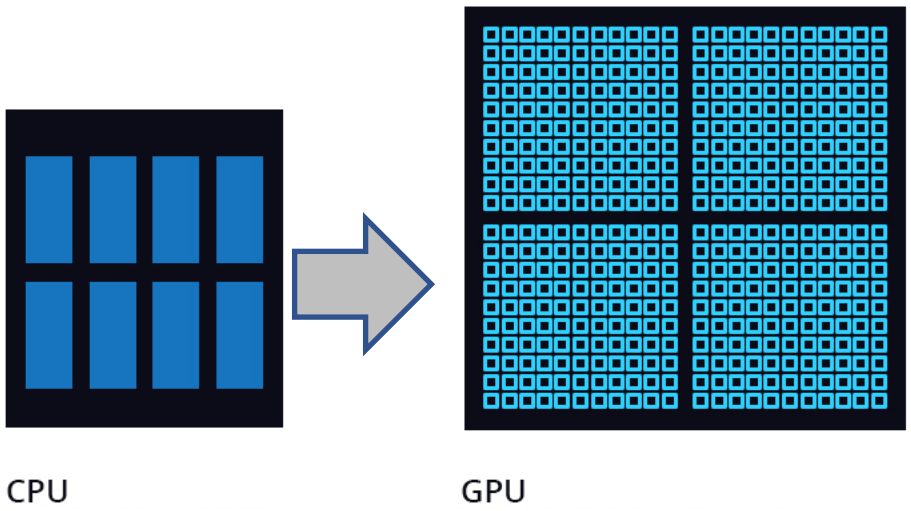
- parallelization, partial rewriting using CUDA for *GPU exploitation* and (possibly) MPI to distribute load over several nodes.

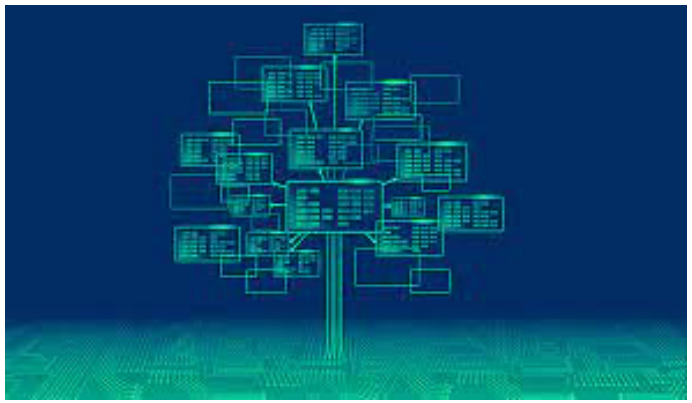
The pre- and postprocessing scripts are coded in ROOT (cWB-2G) and Python (PycWB).

- we will boost the efficiency of the Python wrapper by moving to GPU-enhanced versions of the Python libraries.

Inclusion of new features using *machine learning* :

- development of *decision-tree classification algorithm (XGBoost)* to separate GW signals from noise





Future implementations: Wavelet and Qp transform

Implementing on cWB (or a future pipeline) the new results obtained by Virtuoso et al. (in preparation)

1. Implement a wavelet version of the Q-transform which does have a non-standard inversion formula
   - effective and computationally fast denoising formula

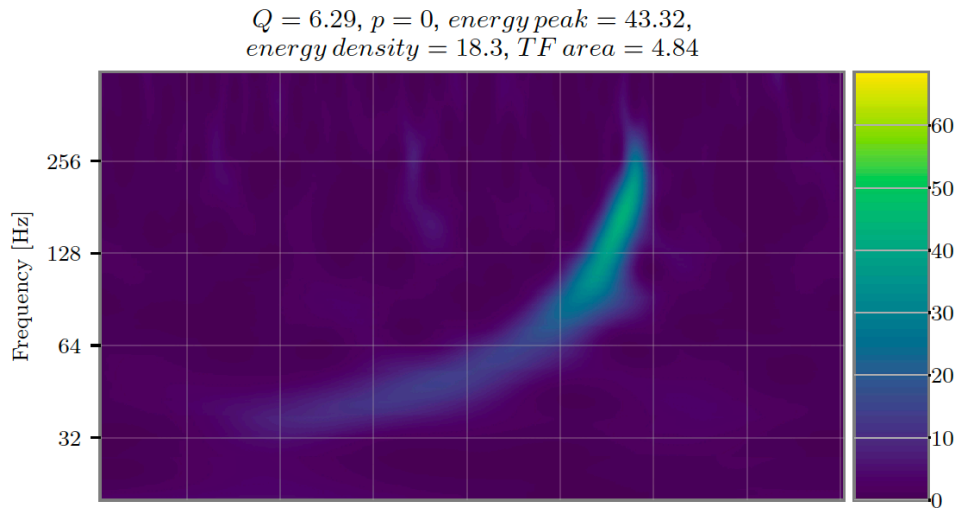
$$s(\tau) = \frac{2}{\left(\frac{2}{\pi Q^2}\right)^{1/4} \operatorname{erf}\left(\frac{Q}{2}\right)} \operatorname{Re}\left[\int_0^{+\infty} d\phi \, \frac{1}{i\sqrt{2\pi\phi}2\pi\phi} \frac{\partial}{\partial\tau} T(\tau, \phi, Q)\right]$$

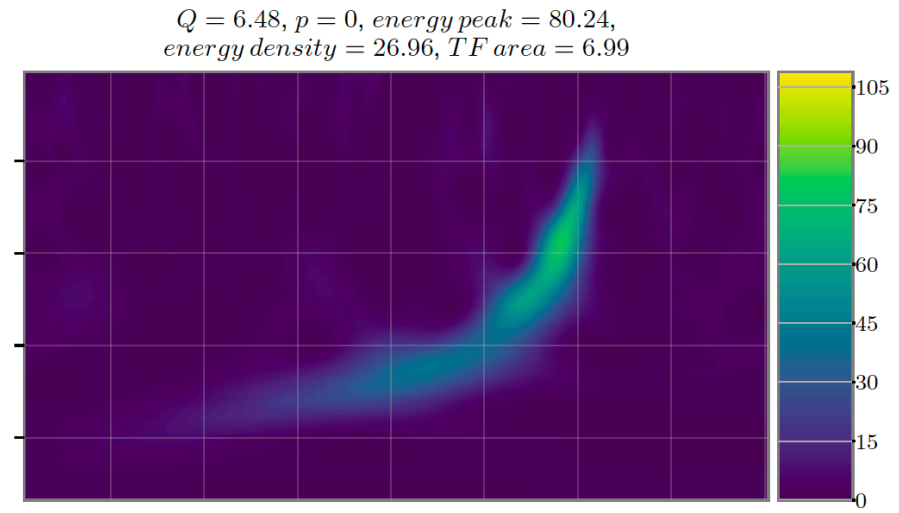
2. Extend the Q-transform (Qp) to better follow frequency-evolving / chirp-like GW signals
   - very effective noise filtering
   - more compact representation of chirps, improving likelihood, SNR and waveform reconstruction

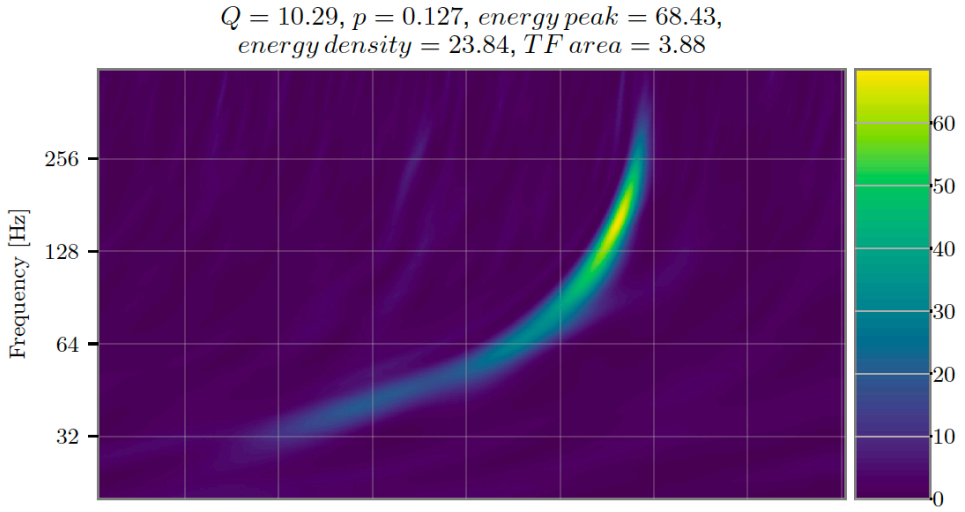
Future implementations: Wavelet and Qp transform

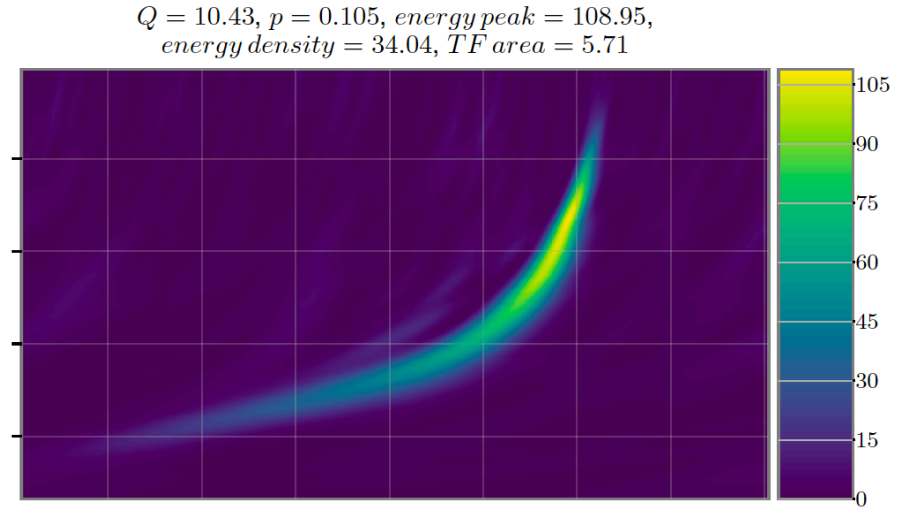
Q-transform

Qp-transform









Workplan

Plan for adapting the cWB pipeline code to the GPUs available at the Centro Nazionale HPC and for including new features to improve its performance and optimise its computing time.

While we wait for the HPC resources we are making some GPU tests 'revamping' an old machine
Mac Pro 5.2 (2012) CPU: 2 x 2.4GHz 6-Core Intel Xeon E5645 GPU: NVIDIA TESLA K20

	2024				2025			
TASK TITLE	**Months 1-3**	**Months 4-6**	**Months 7-9**	**Months 10-12**	**Months 1-3**	**Months 4-6**	**Months 7-9**	**Months 10-12**
Setup computing environment	■	■						
Import O3 LVK public data		■						
Benchmark existing cWB on LEONARDO		■	■					
Accurate timing to find time-consuming code			■	■				
Timing tests on PycWB				■	■			
cWB and PycWB optimization				■	■	■	■	
Algorithmic mods in cWB and PycWB			■	■	■	■		
cWB PycWB + mods: tests & optimization				■	■	■	■	
communication of results at conferences				■	■	■	■	■
paper writing					■	■	■	■

Thanks for your attention