Acceleration in Acoustic Wave Propagation Modelling using OpenACC/OpenMP and its hybrid for the Global Monitoring System

Noriyuki Kushida[*1], Ying-Tsong Lin[*2], Peter Nielsen[*1], and Ronan Le Bras[*1]

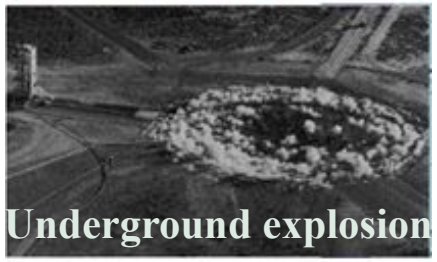
[*1] CTBTO
Preparatory Commission
for the Comprehensive Nuclear-Test-Ban Treaty Organization
Provisional Technical Secretariat

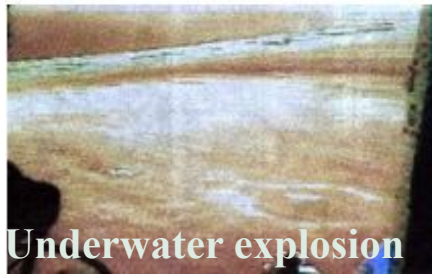
[*2] Woods Hole Oceanographic Institution, USA

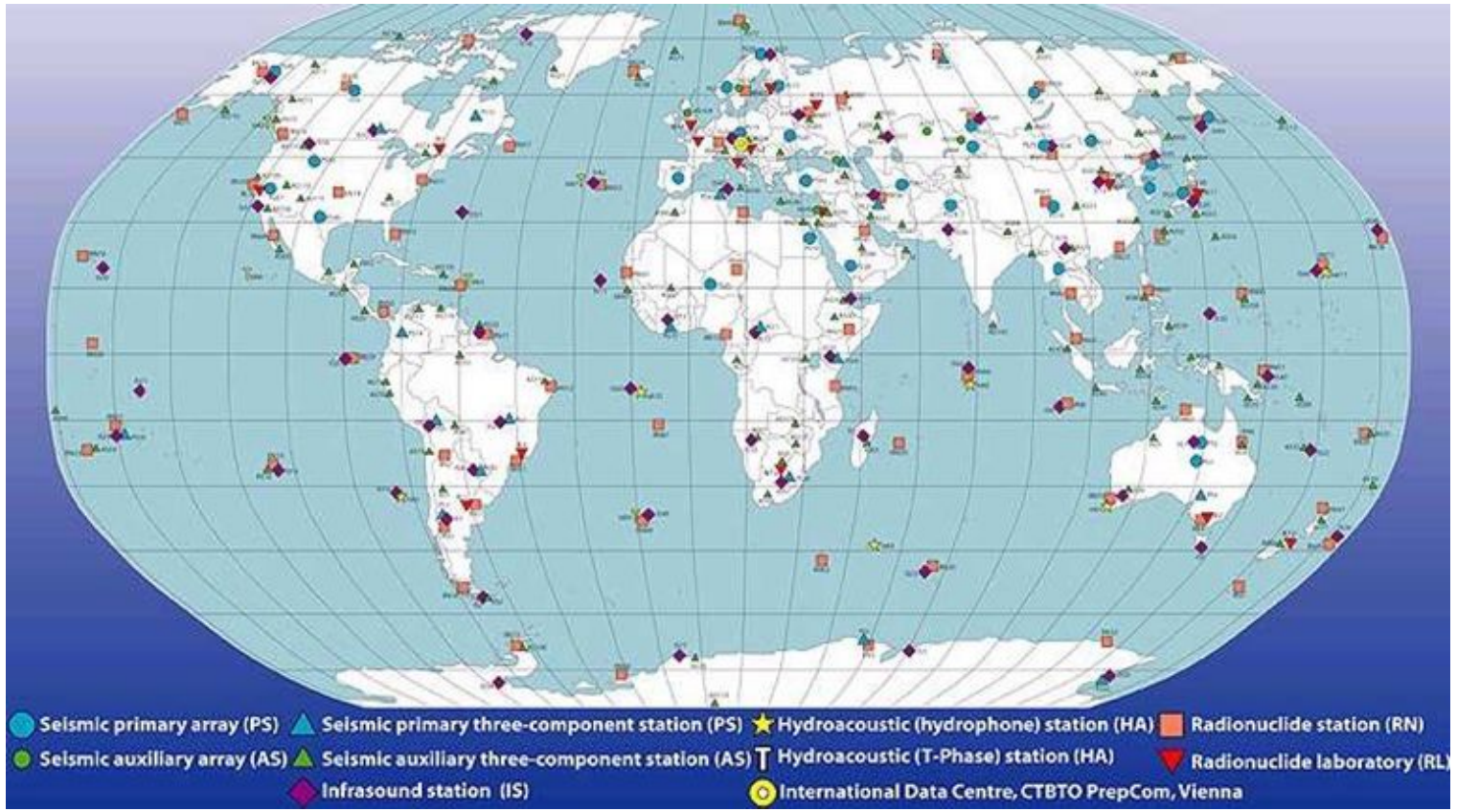
Background (Who we are)

The CTBT (**Comprehensive Nuclear-Test-Ban Treaty**) bans all types of nuclear explosions. **CTBTO** operates a worldwide monitoring system to catch the signs of nuclear explosions with four technologies:

Seismic Infrasound Hydroacoustic Radionuclide



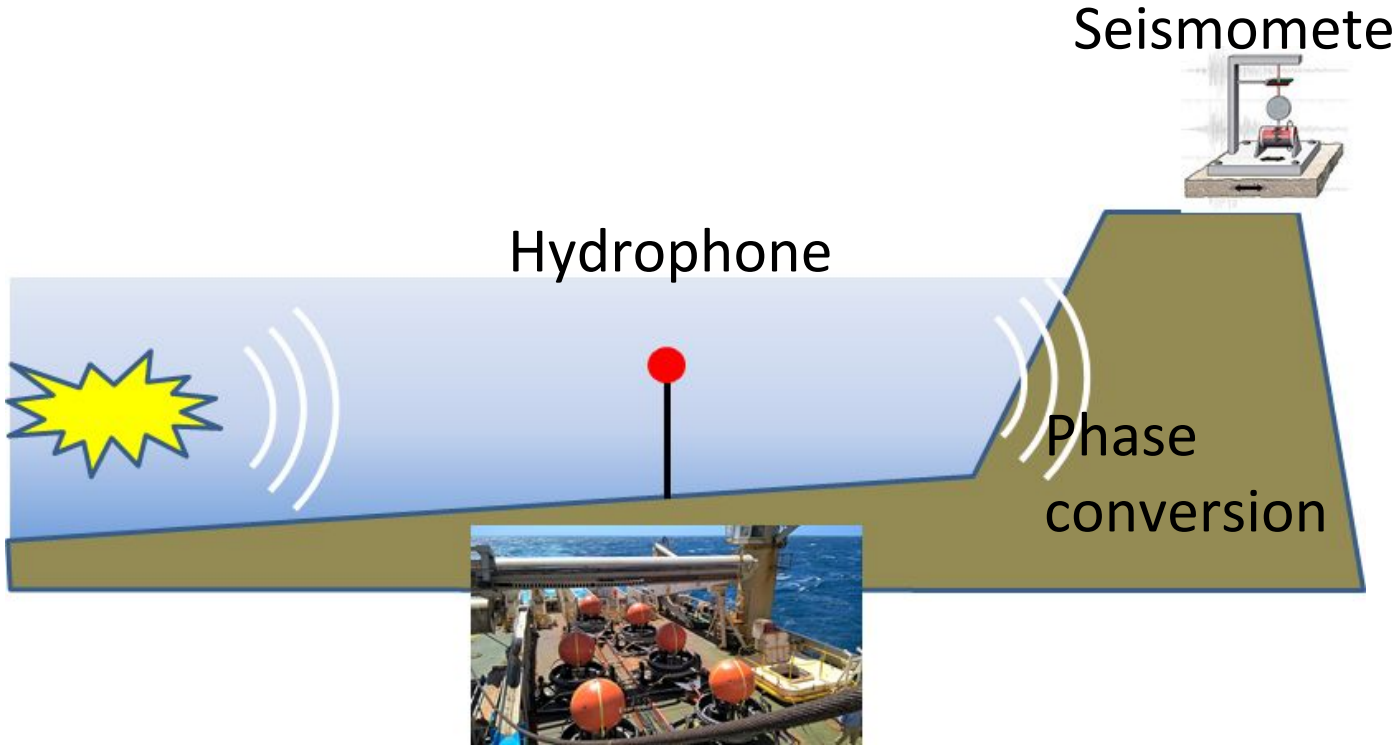




Background (Underwater event and Hydroacoustic observation)

- Hydroacoustic: acoustic waves through the Ocean

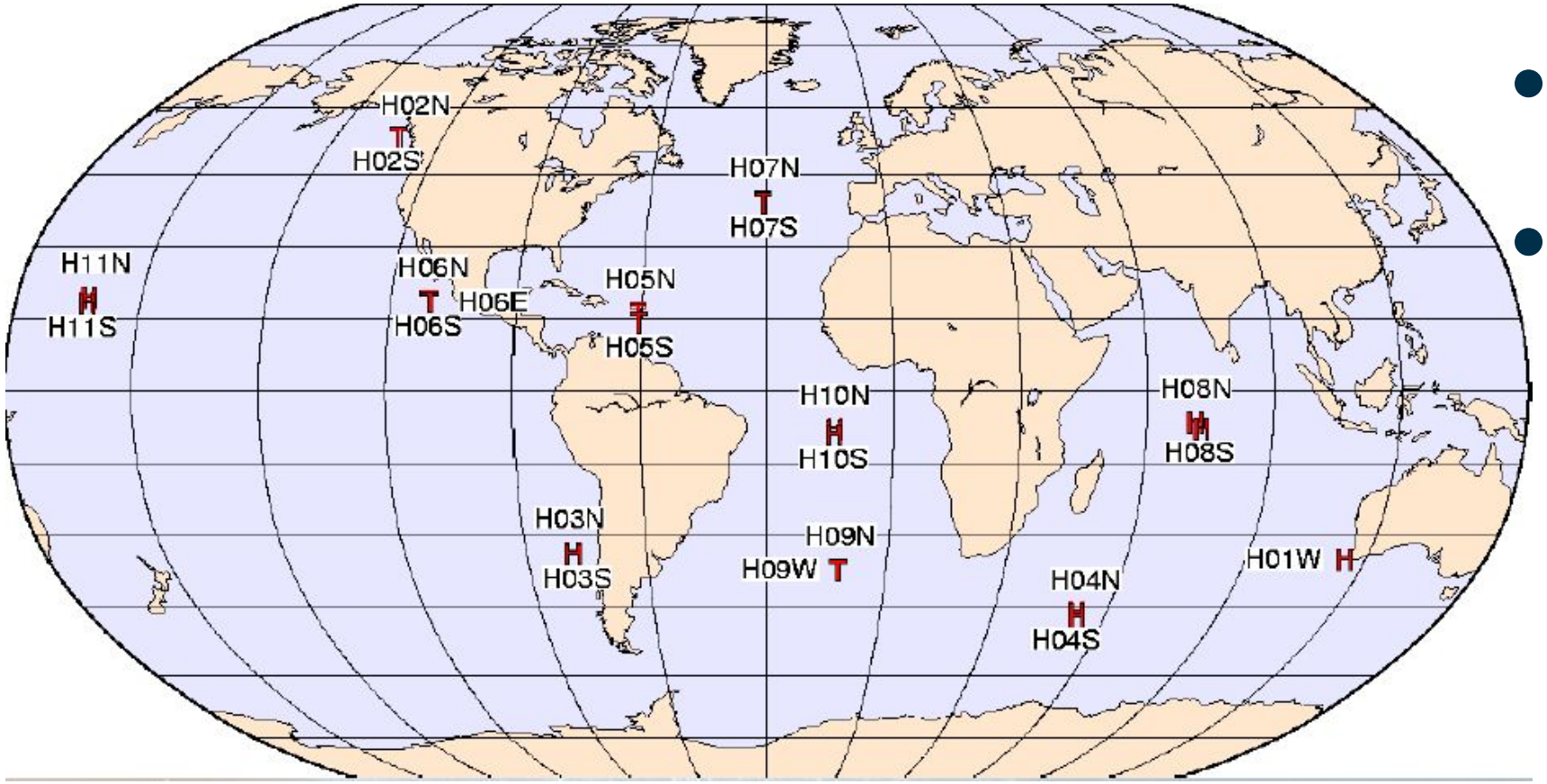
Underwater explosion



- A big explosion in the Ocean can generate a sound wave
- We can catch such sounds with
  - microphones in the Ocean (hydrophone)
  - seismometers on the coast

Photo/Video from CTBTO Website

Background (Hydroacoustic stations in CTBTO)

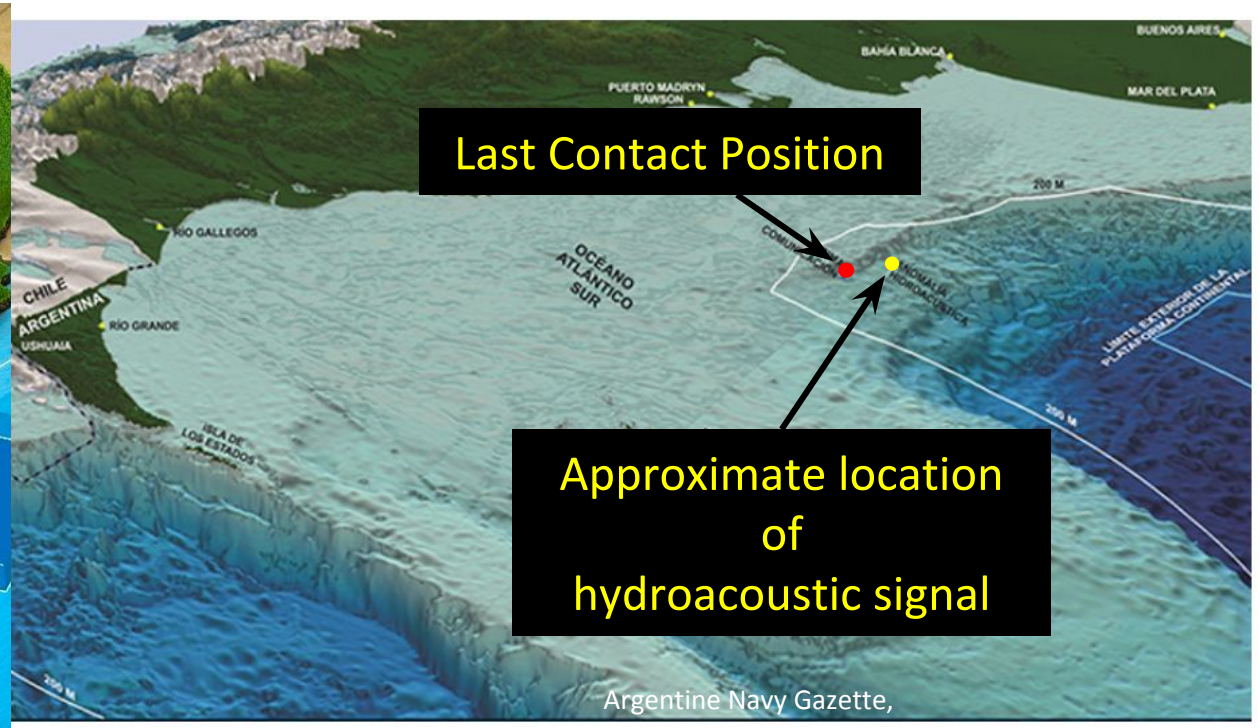


- "**H**" icone: Hydrophone (H-phase). 6 stations
- "**T**" icone: Seismometer (T-phase). 4 stations

- Hydroacoustic waves can travel very long distances
  - **Complex phenomena can be involved**
- Seismic stations; 170 (50 primary + 120 auxiliary) to cover the globe

An example: Argentinian submarine ARA San Juan 1/3

https://www.diariopopular.com.ar/general



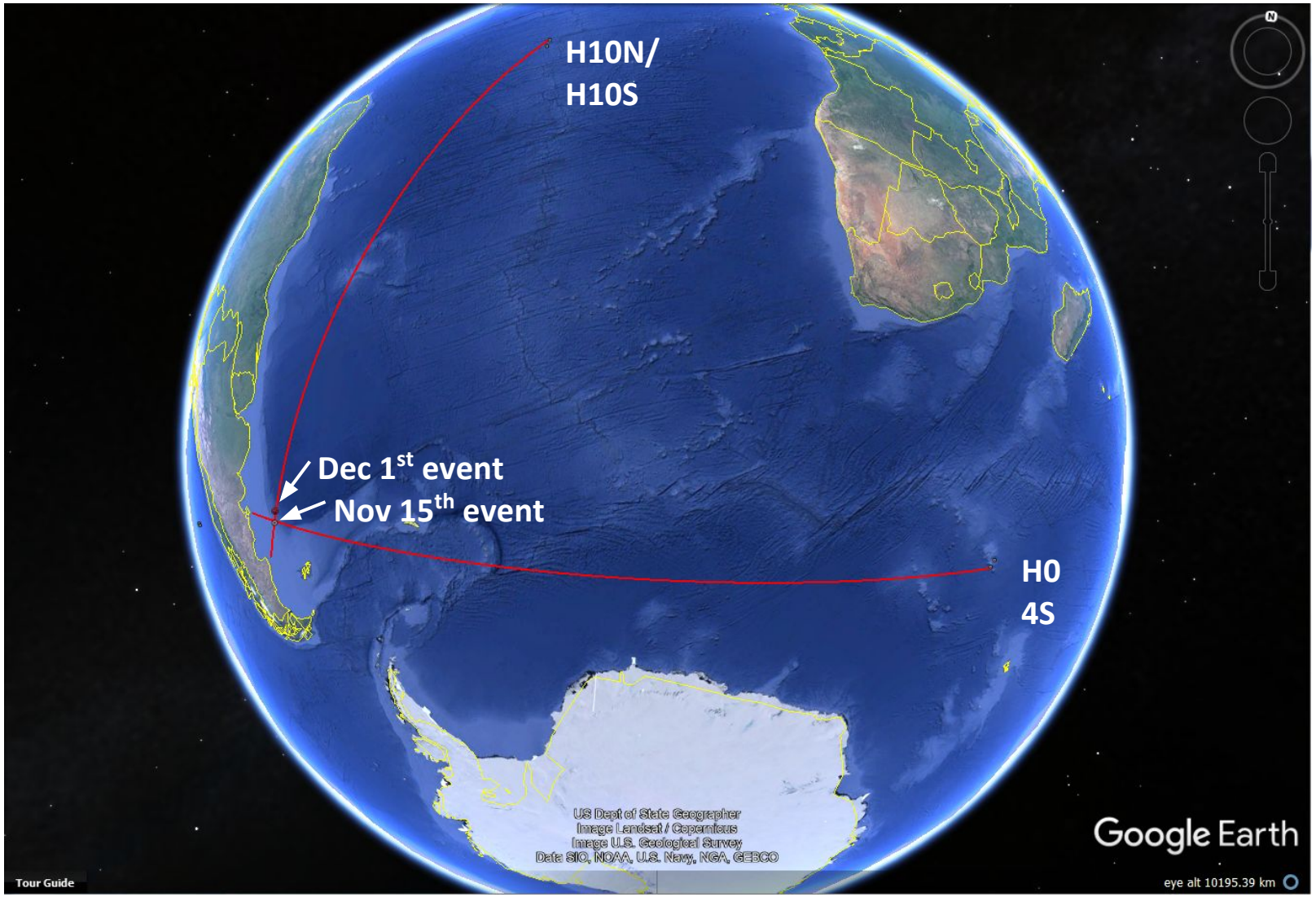
Argentine Navy Gazette,

- In November 2017, CTBTO's hydrophone stations H10 and H04 recorded a hydroacoustic anomaly located in the vicinity of the last know position of the missing Argentine Submarine ARA San Juan

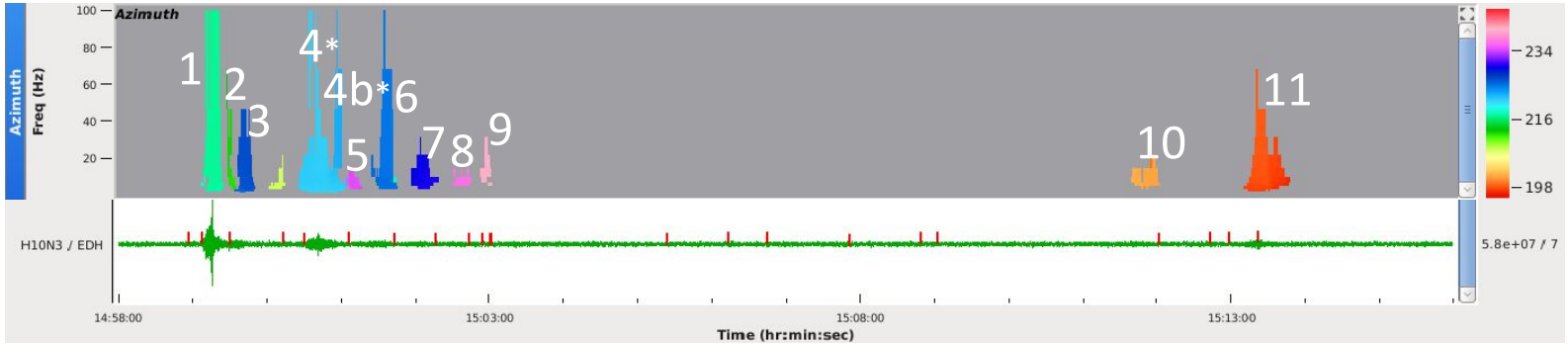
- The full presentation can be found
  - https://presentations.copernicus.org/EGU2018-18559_presentation.pdf

An example: Argentinian submarine ARA San Juan 2/3

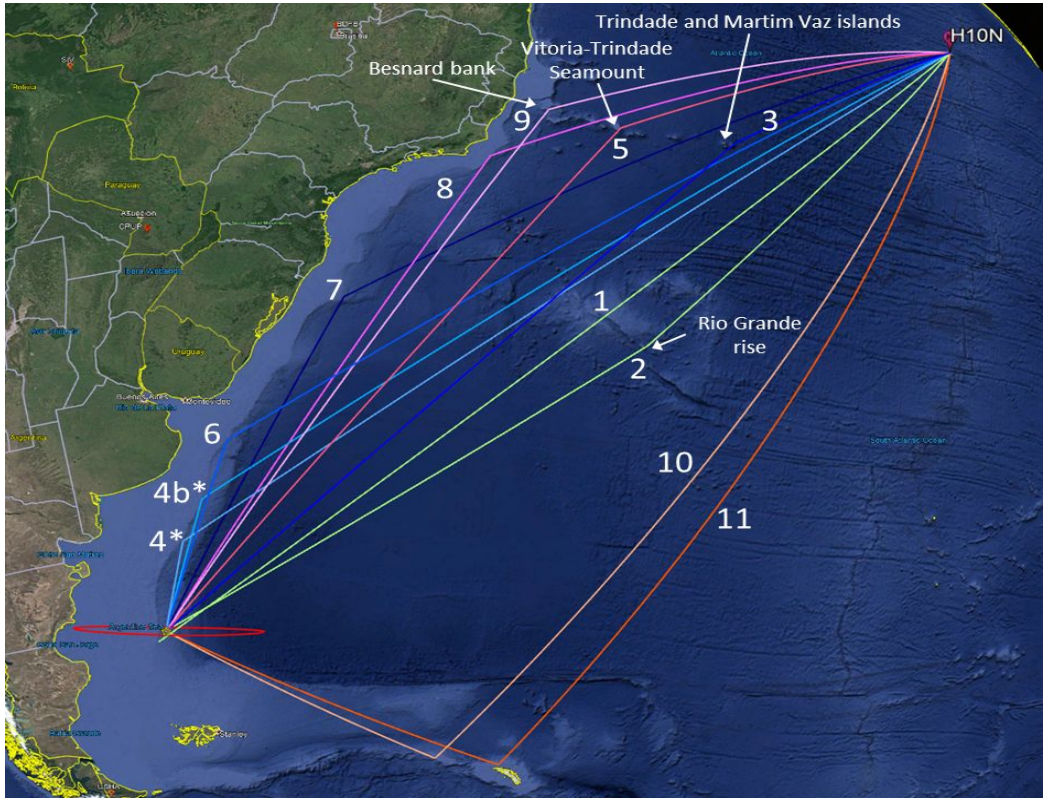
- Signal of unknown origin on November 15th 2017.
- Controlled explosion test conducted by Argentine Navy on December 1st 2017, with source position and time information.
- The test source was detected on CTBT IMS hydrophone stations HA10 and HA04.
- The location of the test source (centre of the 90% confidence ellipse deter-mined based on HA10 and HA04 signals) is 37 km East relative to source location declared by the Argentine authorities.



An example: Argentinian submarine ARA San Juan 3/3



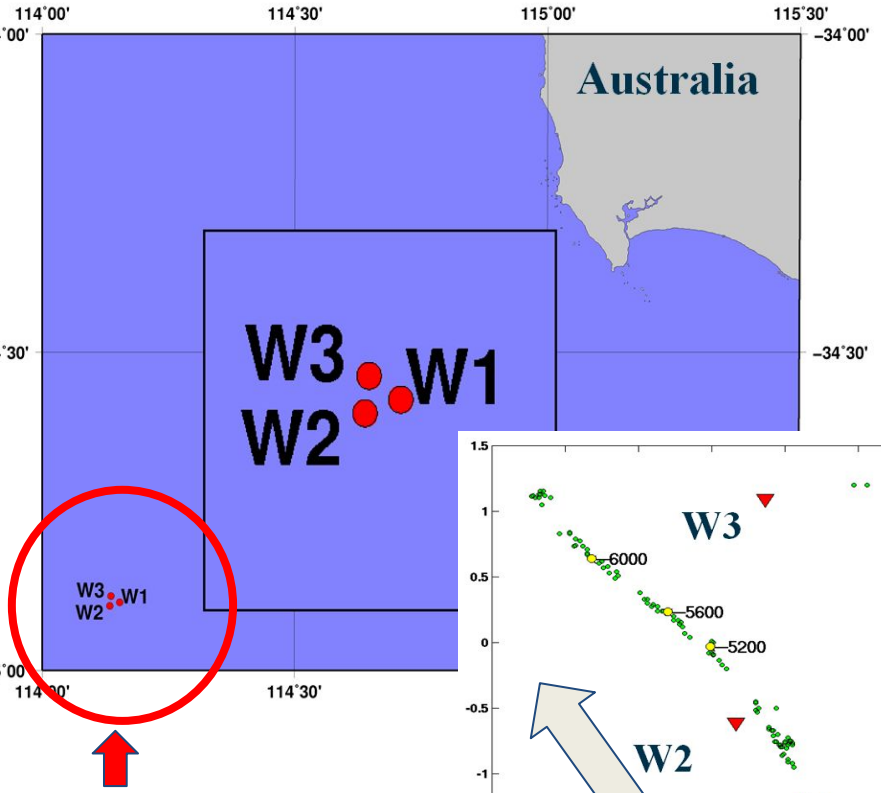
- November 15th signal received on H10N and analyzed using a Progressive Multi Channel Cross-correlation (PMCC) processing algorithm.
- A sequence of 10 late arrivals following the direct main arrival (path number 1) is identified by analyzing a 15 min time window after the main arrival.
- Late arrivals are attributed to reflections off underwater bathymetric features.



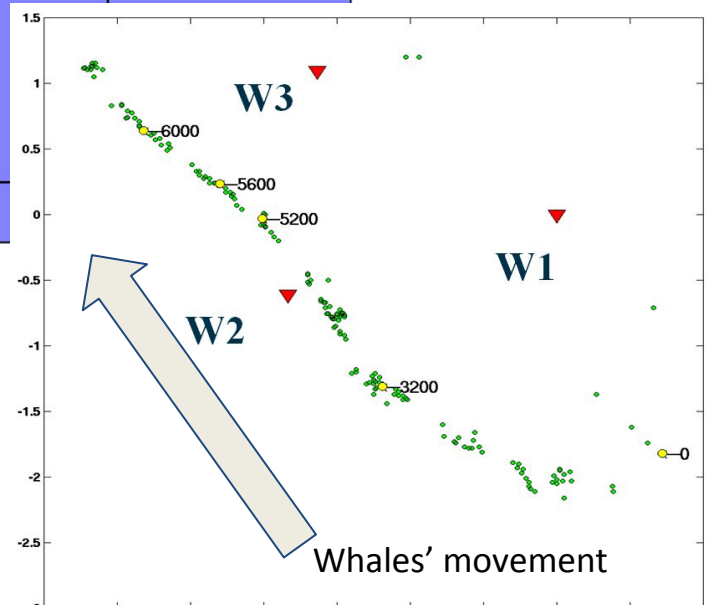
Many reflections are observed, and we are building capacity of utilizing those.

Background:other scientific applications in hydroacoustic (Whales)

- Songs of whales are sometimes observed at a hydrophone station
  - In 2017, one whale passed by the station, near Australia

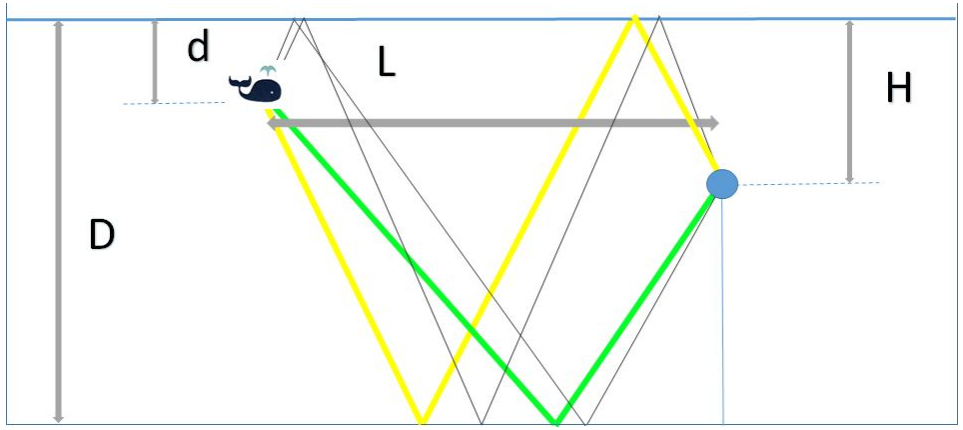


Hydrophone station array



- In the future, we may better know about marine mammals
  - Migration pattern, abundance
  - Significance of the calls – Echolocation – Communication - Social functions

- Desire for a more accurate modelling.
- There seems to be 3D scattering effect
- In a long range, we assume 2D



By courtesy of Dr Le Bras

Objectives in general

We are

- **not** a research institute
  - Computer resource is **limited**: No supercomputer
  - Human resource is **limited**: No time for code tuning

But, we know

- accurate modelling helps our analysis
  - Horizontal reflection, diffraction, 3D effects, attenuation, dispersion, etc.
- many hypothetical cases help **decision makers**
  - Each state signatory decides

So, we have just started exploring HPC technologies

- GPU with NVIDIA DGX-Station

NVIDIA DGX Station

- NVIDIA V100 GPU * 4
  - 7.5 TFLOPS / GPU (double precision)
  - 32GB / GPU
  - 900 GB/sec
- Intel Xeon E5-2698 v4 20 cores
  - 0.576 TFLOPS / CPU: 0.0288 TFLOPS / core
  - 256 GB
  - 71.53 GB/sec

image @NVIDIA web site

Target codes

- 3D-SSFPE
  - Solves **P**arabolic **E**quation (PE) using **S**plit-**S**tep **F**ourier (SSF)
    - Frequency domain based
  - *Vertical* as well as horizontal reflections are taken into account
  - Originally developed in Matlab/Octave

- FDTD with Yin-Yang grid
  - Solves the Wave Equation using **F**inite **D**ifference **T**ime **D**omain (FDTD)
    - Real-space, time-developing
    - Yin-Yang grid is employed for the global modelling
  - Horizontal reflections are mainly considered
  - Developed in Fortran 90

Background and Objective (3D-SSFPE)

- 3D-SSFPE is designed as a long-range hydroacoustic PE solver (Lin, Duda and Newhall, JCA, 2012)
  - Cartesian coordinate (3D-SSFPE) v.s. Polar coordinate (conventional)

Helmholtz wave equation

$$\nabla^2 p + k_0^2 n^2 p = 0$$

p: sound pressure

k_0 : reference media wavenumber

n: index of refraction

One-way wave equation. Coordinate split (η and $\perp$)

$$\frac{\partial}{\partial \eta} p(\vec{x}_\perp, \eta) = ik_0 \sqrt{k_0^{-2}\nabla_\perp^2 + n^2(\vec{x}_\perp, \eta)}\, p(\vec{x}_\perp, \eta)$$

η: solution marching direction

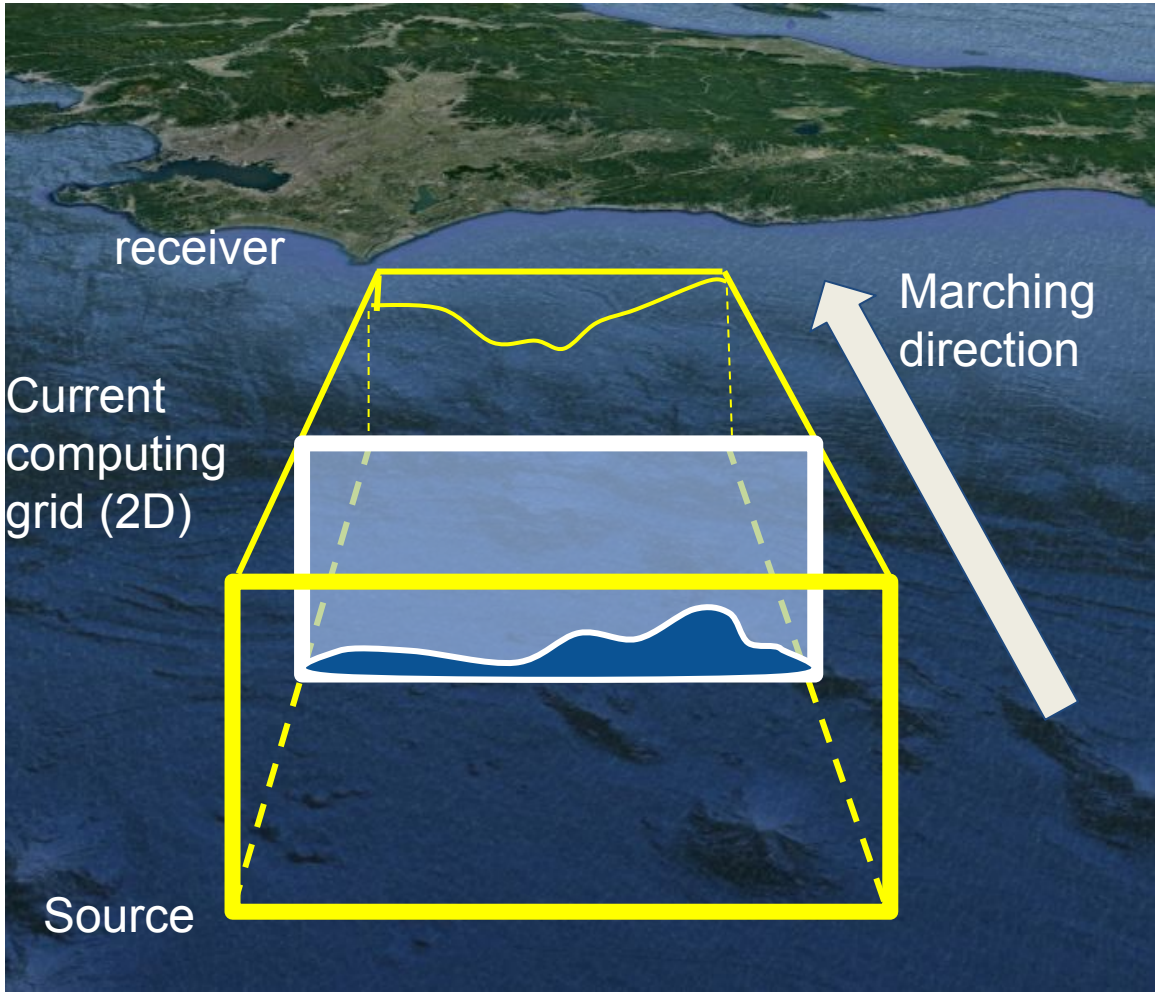
$\perp$: transverse directions

Marching to the next grid

$$p(x_\perp, \eta + \Delta\eta) = \exp\{ik_0 \Delta\eta(n-1)\}\mathcal{F}^{-1}\left(\exp\left\{i\Delta\eta\left(-k_0 + \sqrt{k_0^2 - |k_\perp|^2}\right\}\mathcal{F}\left(p(x_\perp, \eta)\right)\right)\right.$$

iFFT & FFT

- 2D Computational grid marches from source to receiver
  - **Not** time-marching
- **FFT & iFFT** are repetitively used
  - Material properties (incl. bathymetry) are updated accordingly



Picture from Google earth

“Boundary conditions”

- Pre/post processes are also written in Matlab
  - Input/Output data are stored in the Matlab matrix file format
  - **Octave** is the only FOSS that can read/write those files

- 3D-SSFPE is an embarrassingly parallel application
  - **Completely independent** along with frequencies
  - Many computers can be used in parallel

- Matlab’s GPU functions did not improve the entire performance
  - Expectation: FFT functions are dominant
  - Lessons learned: Remaining parts of the kernel should be on GPU

Strategy

- Octave: open source clone of Matlab

- Octfile: functionality to build a user function in C++
  - An object (executable) binary on Linux

- OpenACC & C++
  - First, port to C++
  - Second, port with **OpenMP** to know if parallelizable
  - Then, switch to **OpenACC** by adding data transfer
  - FFT: cuFFT library

Looked perfect at the beginning,,,,,, :(

Problems encountered

- Compilers (PGI, GCC) do not recognize/handle GPUs properly

- PGI 19.5
  - PGI is fixing this but;
  - ```
    Failing in Thread:1
    call to cuMemcpyDtoHAsync returned error 700: Illegal address during kernel execution
    Failing in Thread:1
    call to cuMemFreeHost returned error 700: Illegal address during kernel execution
    ```
  - printf("Num devices: %d\n", acc_get_num_devices(acc_device_nvidia)); => 0 # No GPU is found

  - releases upto 19.10 (latest in November 2019) do not include a fix on this issue

- GCC-9
  - worked fine with some workarounds (brief in the next slide)
    ```

Error in GCC and Workaround

- OpenACC directives do not allocate memory on GPU
  - such as **#pragma acc data copy**

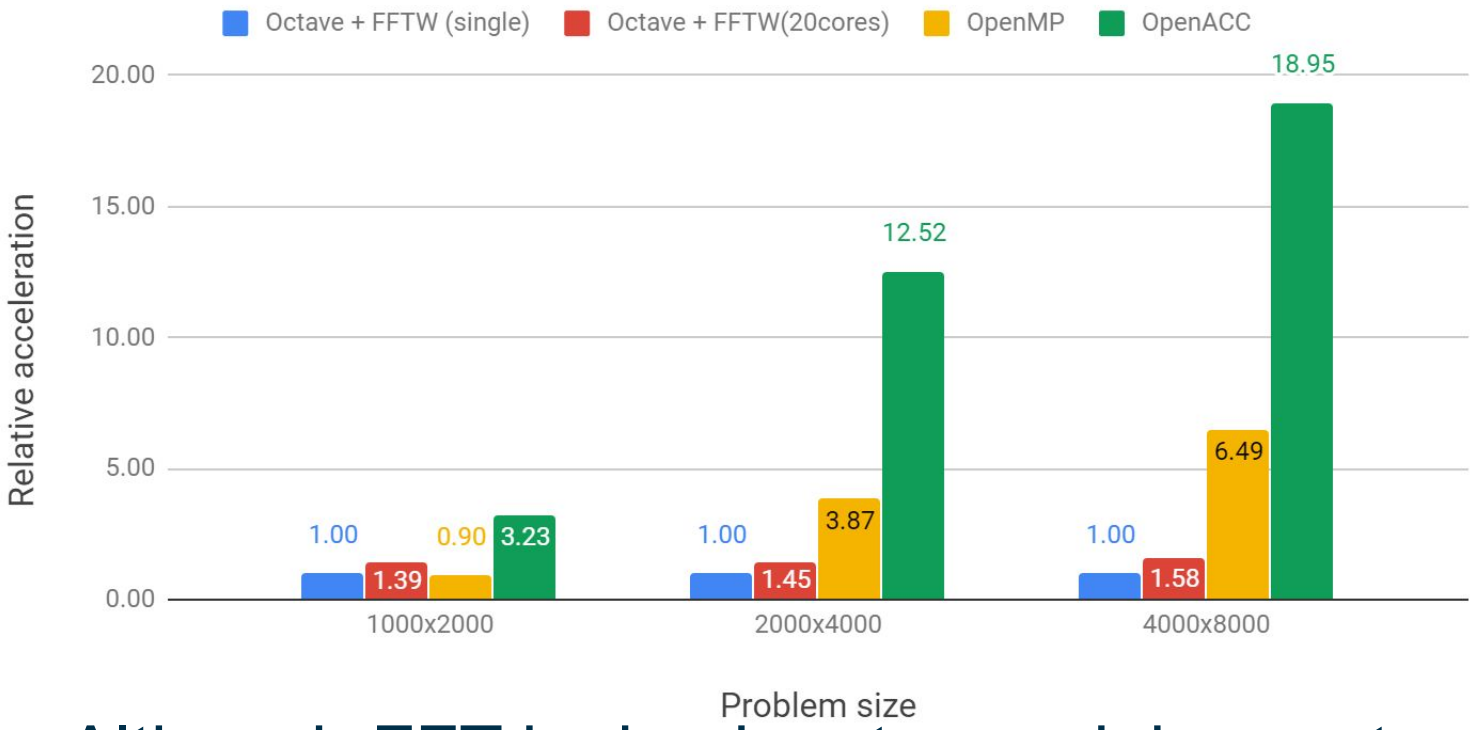
- Fortunately, following functions can be used alternatively

```
acc_malloc      # allocate memory on GPU
acc_map_data    # link CPU and GPU pointers
```

- After those functions, OpenACC directives work as expected
  - If you are struggling with errors relevant to device pointers, this can be a help

- There are many other workarounds. Please refer to the article.
  - And also, a simplified working example is uploaded onto Zenodo

Result (3D-SSFPE)



- Octave + FFTW(single)
  Fully single thread
- Octave + FFTW (20 cores)
  FFTW is threaded
- OpenMP
  Entire code is threaded with OpenMP + FFTW (15 cores)
- OpenACC
  Entire code is on GPU

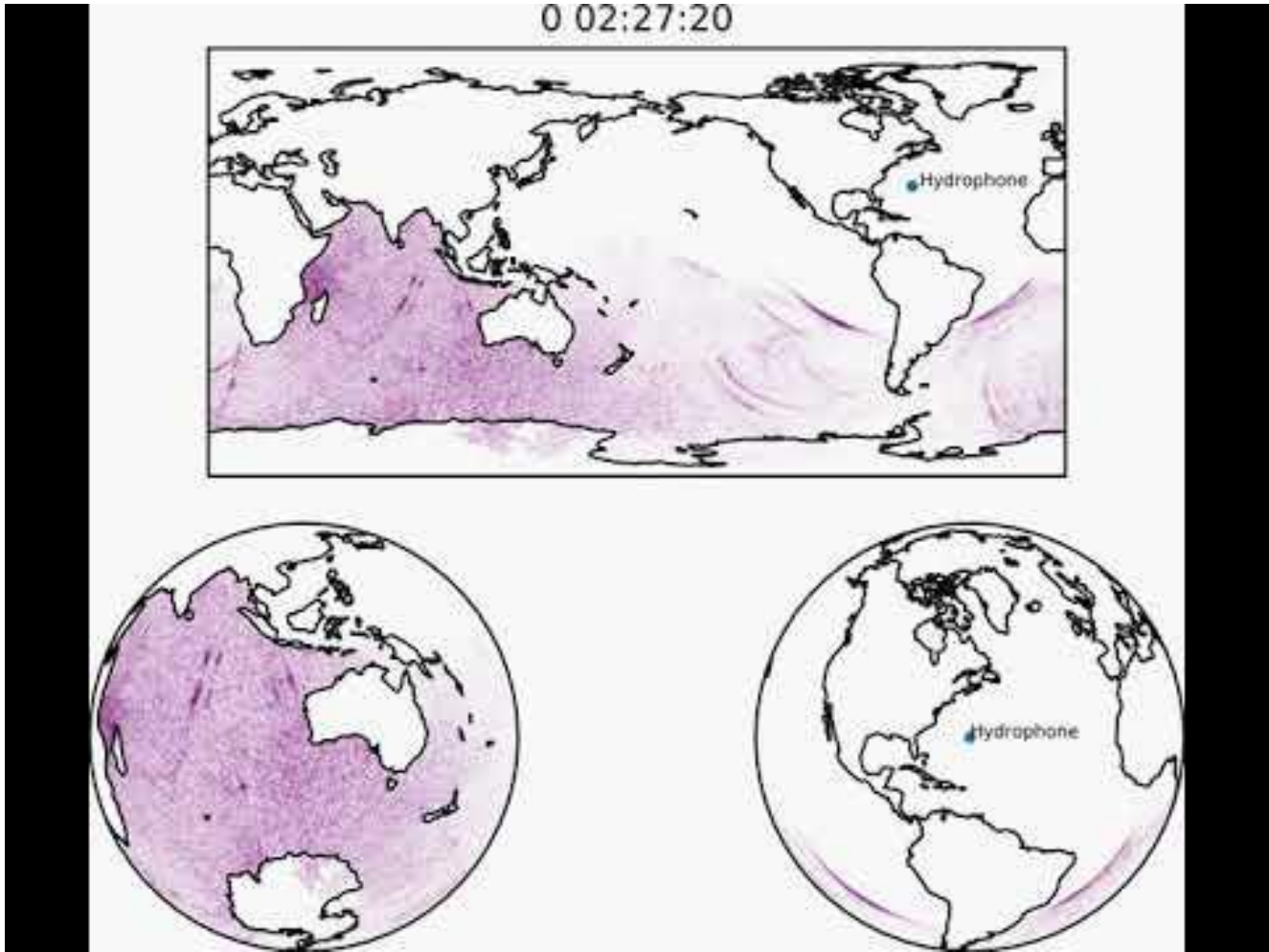
- Although FFT is dominant, remaining parts are not negligible
  - Improvement with the threaded FFTW is limited
  - OpenMP improves the entire performance significantly

- Porting the entire of kernel is important for high performance

Background (FDTD)

- Real-space time-domain in global hydroacoustic modelling has not been popular
  - High demand in computer resource, especially at a high frequency
  - Instability for long-term and long-range computation

- Advantages
  - Explicit waveform input
  - Easy to handle inhomogeneous media
    - even dynamic change
  - possibility for multi-physics



Objective (FDTD)

- On a global scale, horizontal propagation is dominant
  - 2D spherical coordinate
  - Hydroacoustic propagates through SOFAR channel

- **SO**und **F**ixing **A**nd **R**anging channel
  - works like the optical-fiber cable
  - traps and conducts hydroacoustics efficiently
  - is a layer around 1000m depth

- Demand for HPC
  - larger number of hypothetical events
  - higher frequency
    - *2 frequency => *0.5 grid size
    - *2^2 grids *2 time steps: Time = $O(n^3)$, RAM = $O(n^2)$

Governing equation

Background Pressure Velocity

$$\left(\frac{\partial}{\partial t}+\mathbf{v}\cdot\nabla\right)p+\rho c^2\nabla\cdot\mathbf{w}=\rho c^2 Q$$

Pressure distribution caused by wave velocity

$$\left(\frac{\partial}{\partial t}+\mathbf{v}\cdot\nabla\right)\mathbf{w}+(\mathbf{w}\cdot\nabla)\,\mathbf{v}+\frac{\nabla p}{\rho}=\frac{\mathbf{F}}{\rho}$$

Velocity driven by pressure on **background flow**

- Introduced by **Ostashev** *et al*. (2005) to analyze **infrasound propagation** in the atmosphere
  - They revealed the explicit form in the cartesian coordinates

- Time marching **wave equation** (split form) on a **background flow**
  - **Inhomogeneity of material properties** can be taken into consideration

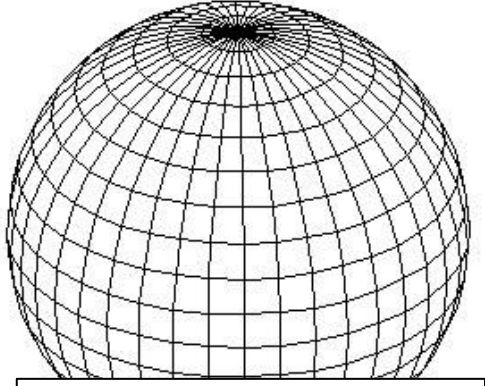
Transform to the spherical coordinates, and drop the radial direction components
(because we are interested in the long-range propagation, vertical components are negligible)

$$\frac{\partial p}{\partial t}=-\left(\frac{v_\theta}{r}\frac{\partial p}{\partial\theta}+\frac{v_\phi}{r\sin\theta}\frac{\partial p}{\partial\phi}\right)-\kappa\left(\frac{1}{r\sin\theta}\frac{\partial}{\partial\theta}(\sin\theta w_\theta)+\frac{1}{r\sin\theta}\frac{\partial w_\phi}{\partial\phi}\right)+\kappa Q$$

$$\frac{\partial w_\theta}{\partial t}=-\left(\frac{v_\theta}{r}\frac{\partial w_\theta}{\partial\theta}+\frac{v_\phi}{r\sin\theta}\frac{\partial w_\theta}{\partial\phi}-\frac{v_\phi w_\phi\cot\theta}{r}\right)-\left(\frac{w_\theta}{r}\frac{\partial v_\theta}{\partial\theta}+\frac{w_\phi}{r\sin\theta}\frac{\partial v_\theta}{\partial\phi}-\frac{w_\phi v_\phi\cot\theta}{r}\right)-b\frac{1}{r}\frac{\partial p}{\partial\theta}+bF_\theta$$

$$\frac{\partial w_\phi}{\partial t}=-\left(\frac{v_\theta}{r}\frac{\partial w_\phi}{\partial\theta}+\frac{v_\phi}{r\sin\theta}\frac{\partial w_\phi}{\partial\phi}+\frac{v_\phi w_\theta\cot\theta}{r}\right)-\left(\frac{w_\theta}{r}\frac{\partial v_\phi}{\partial\theta}+\frac{w_\phi}{r\sin\theta}\frac{\partial v_\phi}{\partial\phi}+\frac{w_\phi v_\theta\cot\theta}{r}\right)-\frac{b}{r\sin\theta}\frac{\partial p}{\partial\phi}+bF_\phi$$

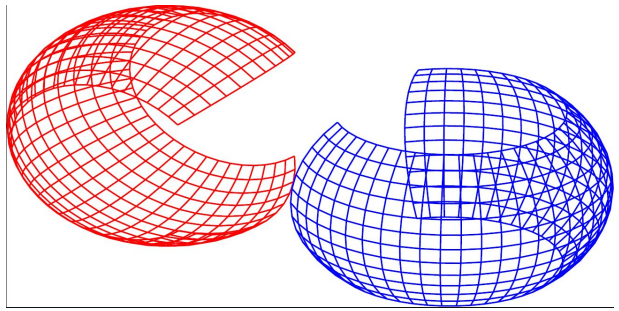
Yin-Yang grid

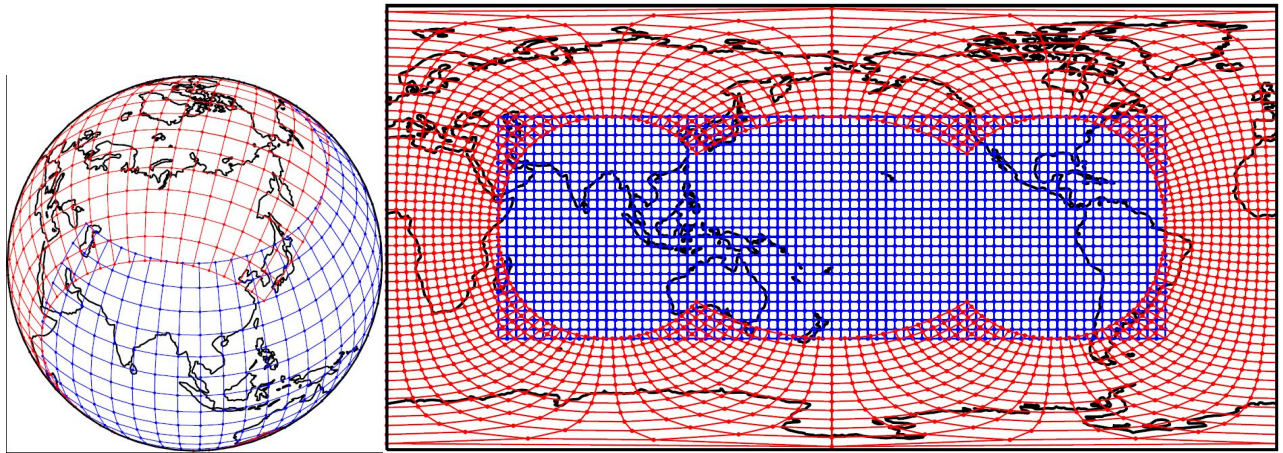


lat-lon grid @ mitgcm.org/

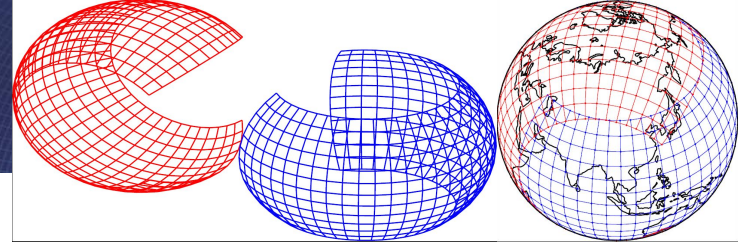
- Developed by Kageyama (2004) for FDM with the spherical coordinates
- Combine two identical grids which cover the entire globe

- Pros
  - No “pole problem” (singularity at the poles)
  - Quasi-uniform grid: In the standard spherical coordinate, the grid size is quite small in high-altitude areas. This causes instability and requires more computational effort
- Cons
  - Complexity is introduced in programming, especially in parallel environments
  - Interaction between two grids can cause instability: high frequency error was observed in prior studies

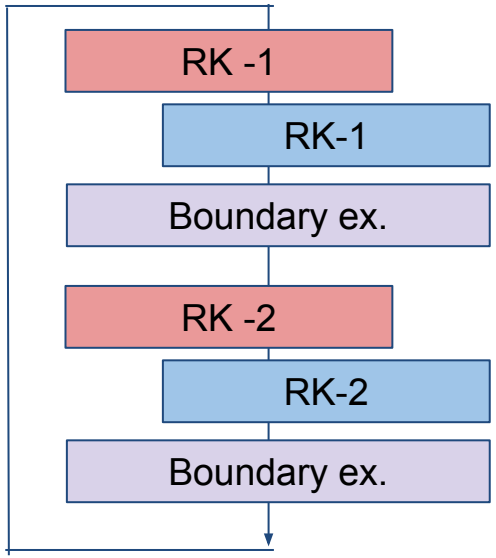




OpenACC and multiple GPUs

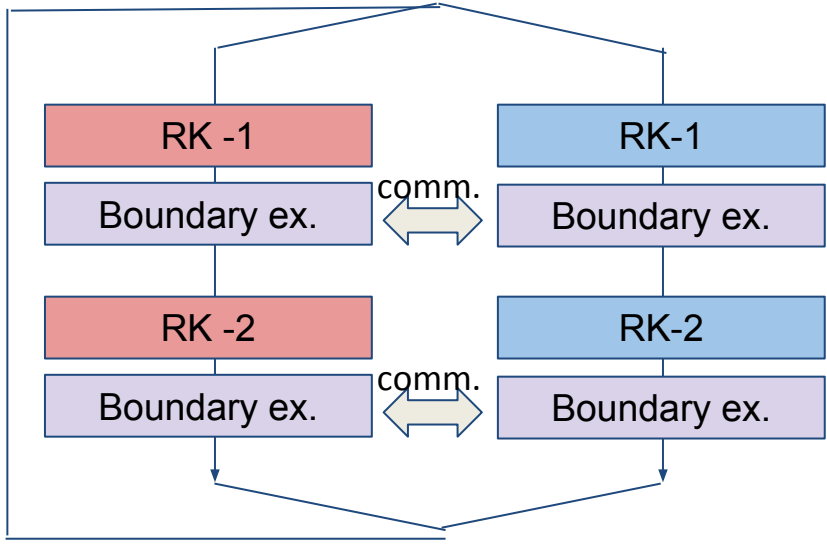


- Yin-Yang grid consists of two **identical** grids
  - two grids can be evaluated in parallel
  - We can use two GPUs

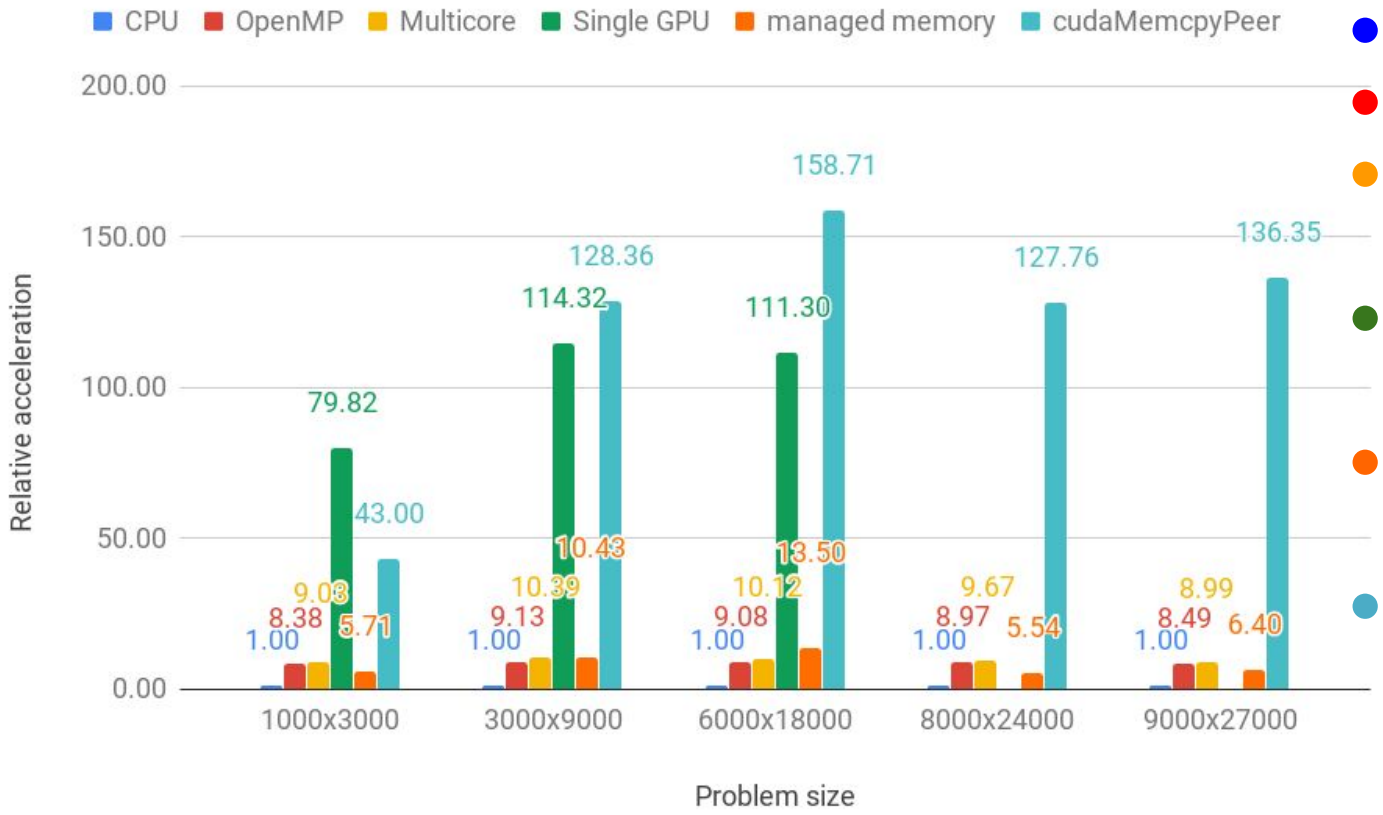


- Hybridization of OpenACC and OpenMP
  - OpenMP: to launch threads that control assigned GPUs
  - OpenACC: `acc_set_device_num` to assign a GPU to a thread

- Exchanging data between Yin and Yang
  - "managed memory" in PGI
    - "unified memory" in CUDA context
    - Handles data communication automatically
  - cudaMemcpyPeer
    - users need to control data transfer
    - enables direct communication among GPUs



Results of speed-up



- CPU: Fully single thread
- OpenMP: 20 cores
- Multicore: OpenACC @ -ta=multicore (20 cores multithread)
- Single GPU: OpenACC @ 1 GPU w. managed mem.
- Managed memory: OpenACC @ 2 GPUs.
- cudaMemcpyPeer: OpenACC @ 2 GPUs w. cudaMemcpyPeer

- multithreading codes provide **x10** acceleration to 1 core
  - OpenACC is slightly better
- Single GPU provides **x100** acceleration
- double GPU w. cudaMemcpyPeer **x150** acceleration
- double GPU w. Managed memory is slow, but provides large RAM

Discussion

- Alternative ways for multi-GPUs
  - MPI: *a steam-hammer to crack a nut* for an NVIDIA DGX-Station
  - **async** directive on OpenACC: switching GPUs with a single thread
    - may lead to another complication

- Pros and Cons of the hybridization
  - We were able to build the codes incrementally, without major code rewriting
    - formulation, single core, OpenMP, OpenACC, multiple-GPUs
  - Not straightforward for hierarchical parallelization
    - Yin-Yang: OK, domain decomposition of each: not easy

- Generally, OpenACC/OpenMP hybrid is suitable for “small labs”

Summary (overall)

- 3D-SSFPE obtained **x20** acceleration with OpenACC
  - this might be the first attempt of OpenACC accelerated Matlab/Octave

- FDTD with Yin-Yang obtained **x160** acceleration with hybridization
  - No major code rewrite has been required
  - suitable for small labs

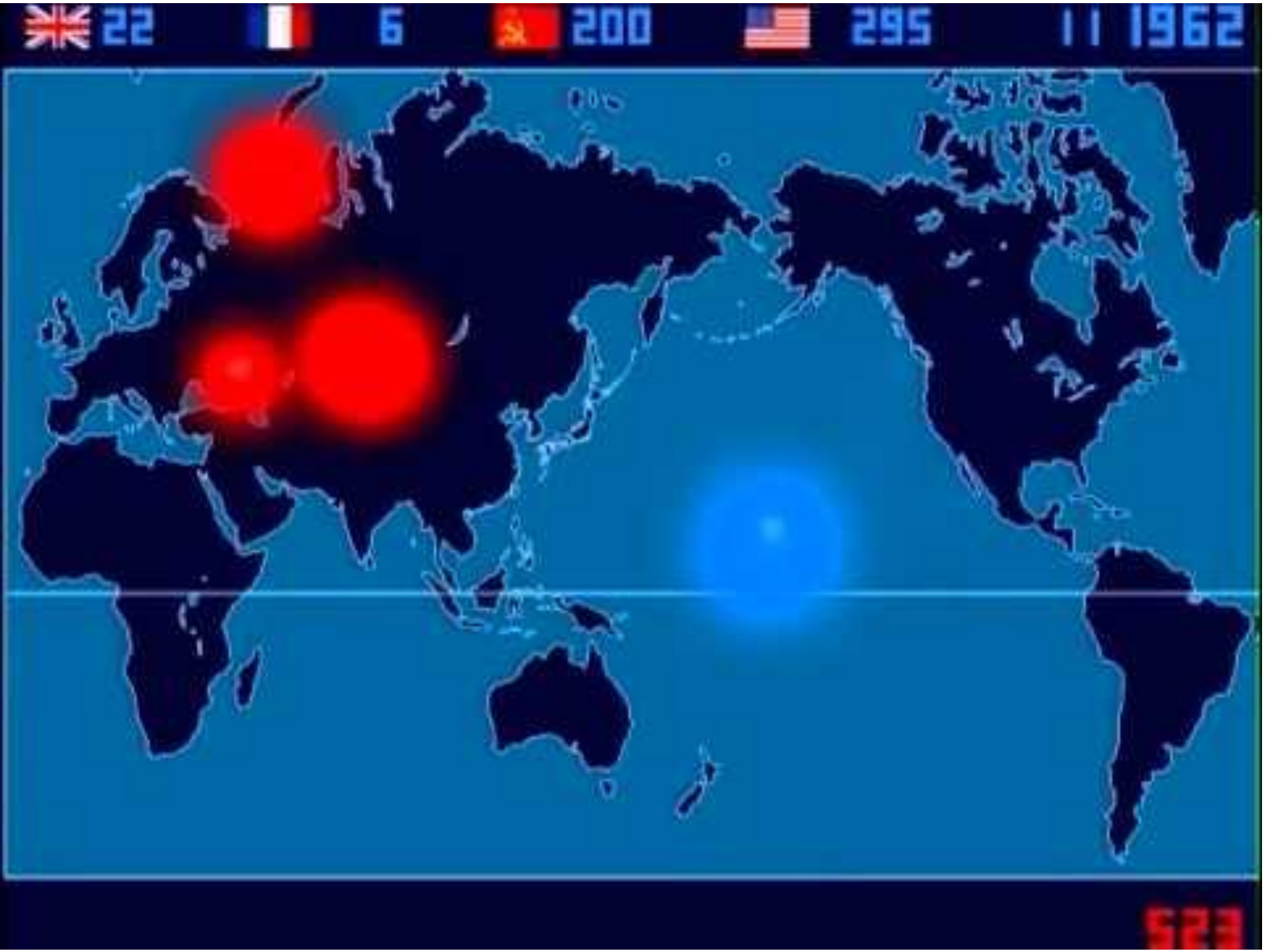
- Those achievements may encourage scientists to use GPUs
  - particularly, those who do not have access to HPC

Future works

- 3D-SSFPE
  - Multi-GPUs / instance
  - Mixed-precision computing
  - embarrassingly parallel @Massively parallel computer

- FDTD
  - Higher frequencies
  - 3D
    - Full 3D and multi-physics (atmosphere/ocean/solid earth)
    - Quasi-3D (2D Yin-Yang + vertical mode)
    - demand for MPI (distributed) parallel. Irena Skylake@CEA via PRACE
  - Other physics features
    - Attenuation, dispersion, energy loss at boundaries,

One more thing: About nuclear explosion testing:



THE ASTANA T
BRINGING KAZAKHSTAN TO THE W

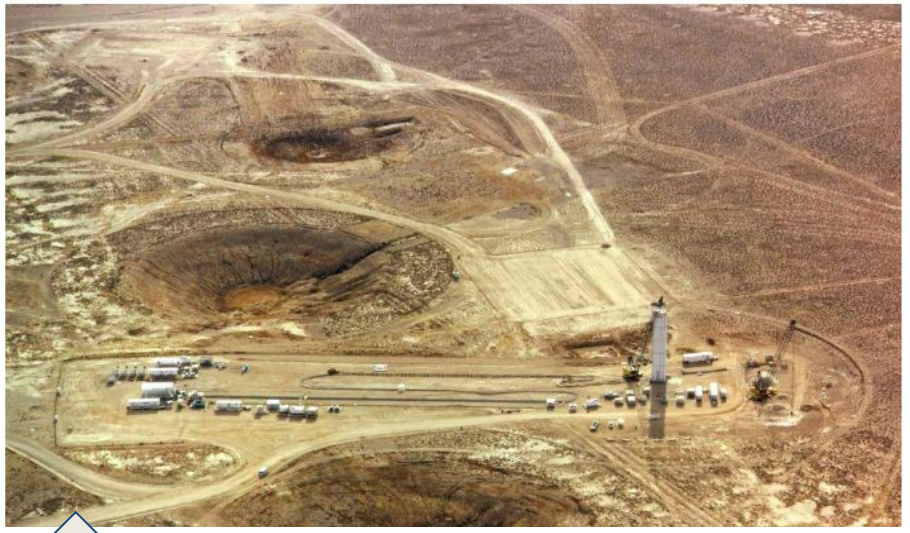
Nation | Nur-Sultan | International | Business | Tourism | Culture | Sports | People | Editoria

Top News: Shalabai breeding farm renews Akbas cow breed line, boosts local welfare

Kazakh Government provides compensation to thousands of nuclear weapons tests victims

BY STAFF REPORT *in* NATION *on* 6 SEPTEMBER 2019

NUR-SULTAN – Approximately 3,000 compensations have been paid to Kazakh nuclear test victims since the beginning of the year. They were provided by Government for Citizens, a single public service provider akin to Canada Service and Centrelink (Australia), which integrates all public service centres into one system.

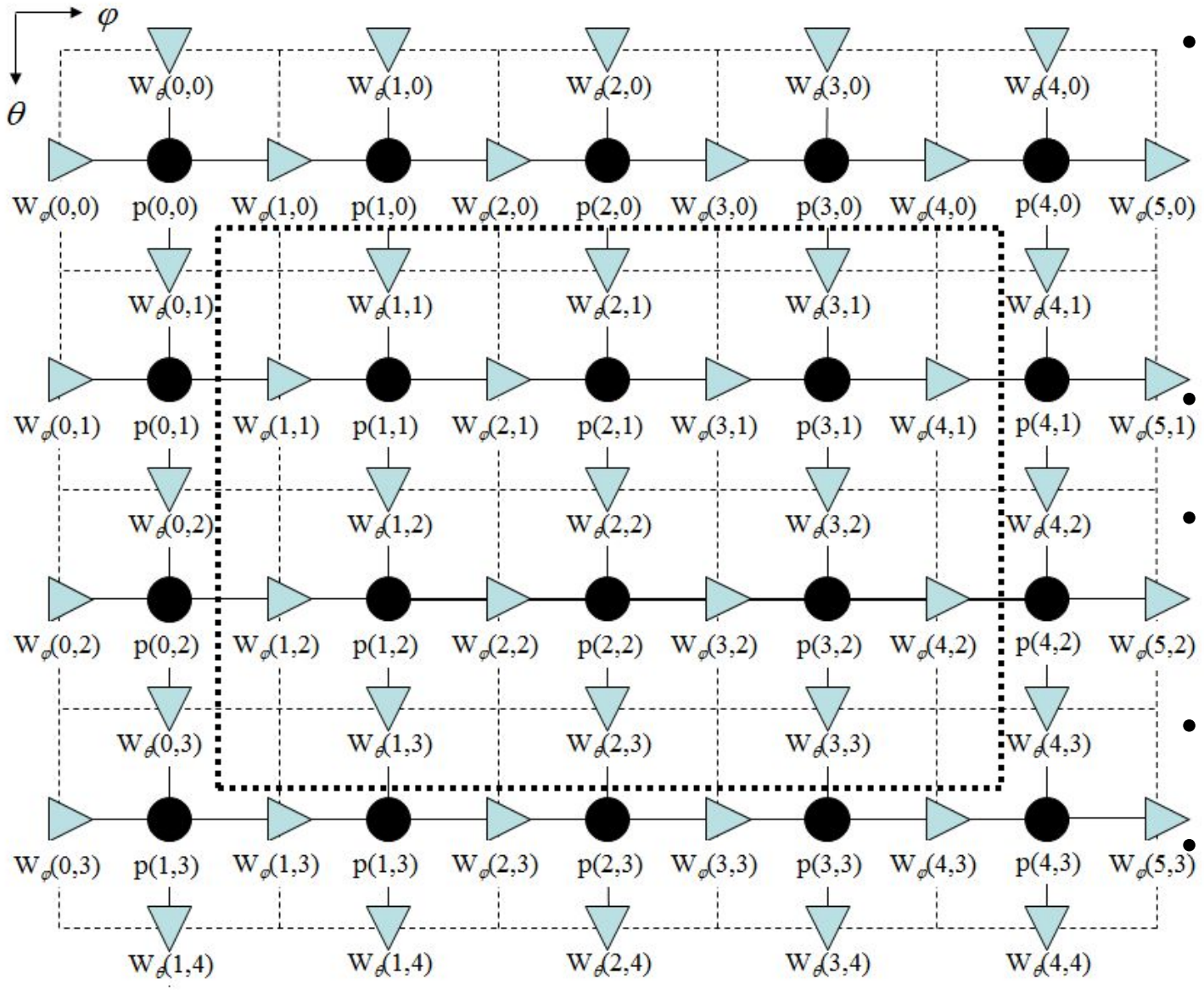


About victims of radiation
(The Astana Times, 9 Sep, 2019)

How many explosions?
https://youtu.be/cjAqR1zlCA0

Reserved slides

Staggered FDM and Yin-Yang grid



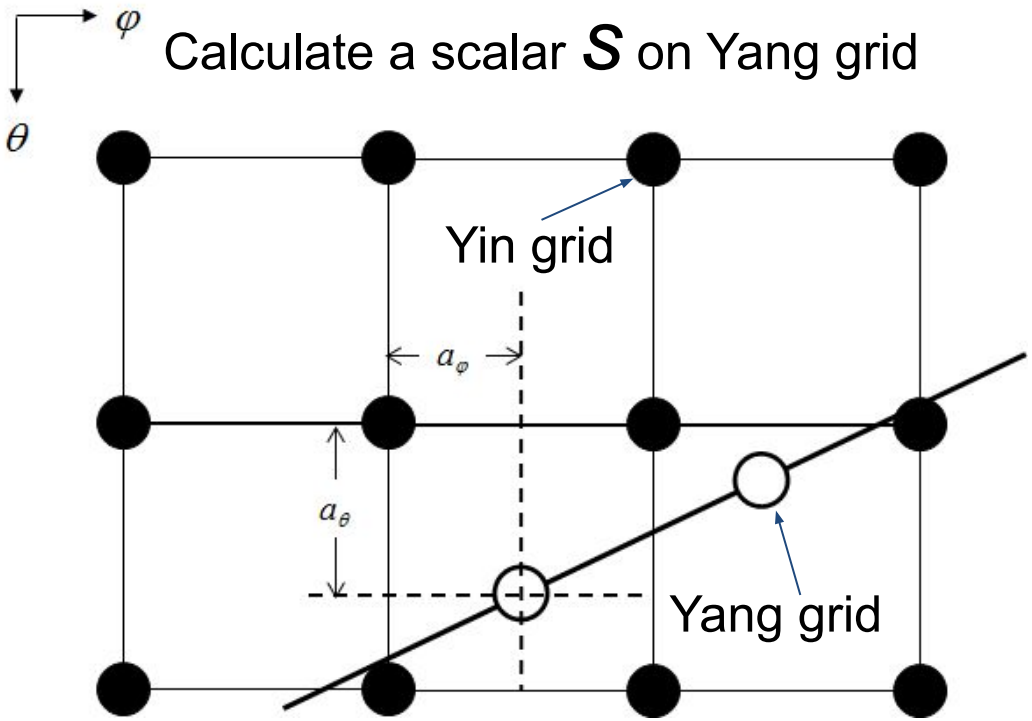
- Pressure and velocity components are defined on different grid points
  - To obtain differential in a natural way
  - Ostashev equation used in the cartesian coordinate
- Points in the dotted line rectangle are computed
- <u>External points are imported from the other grid</u>

- Other scalar values are mapped on the pressure grid
- Other vector values are mapped on the velocity grid

Interpolation for staggered grid

Scalar: simple bi-linear interpolation

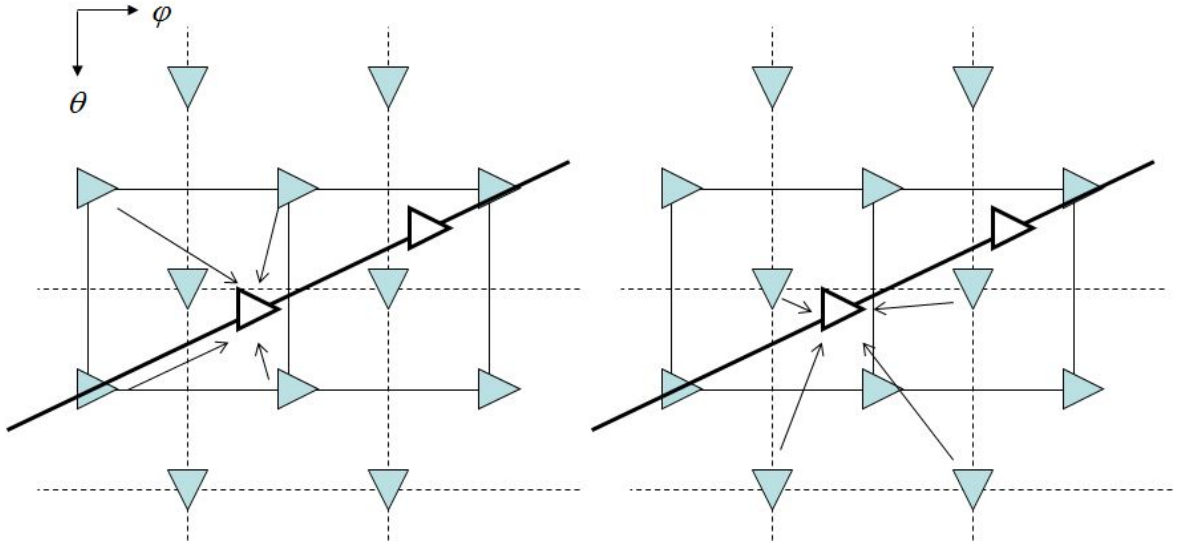
Calculate a scalar ***S*** on Yang grid



$$S^{\text{Yang}} = \square \times \square + \square \times \square + \square \times \square + \square \times \square$$

Vector: interpolate all components, and transform

Calculate φ component of a vector ***V*** on Yang grid



interpolate φ component interpolate θ component

Transform

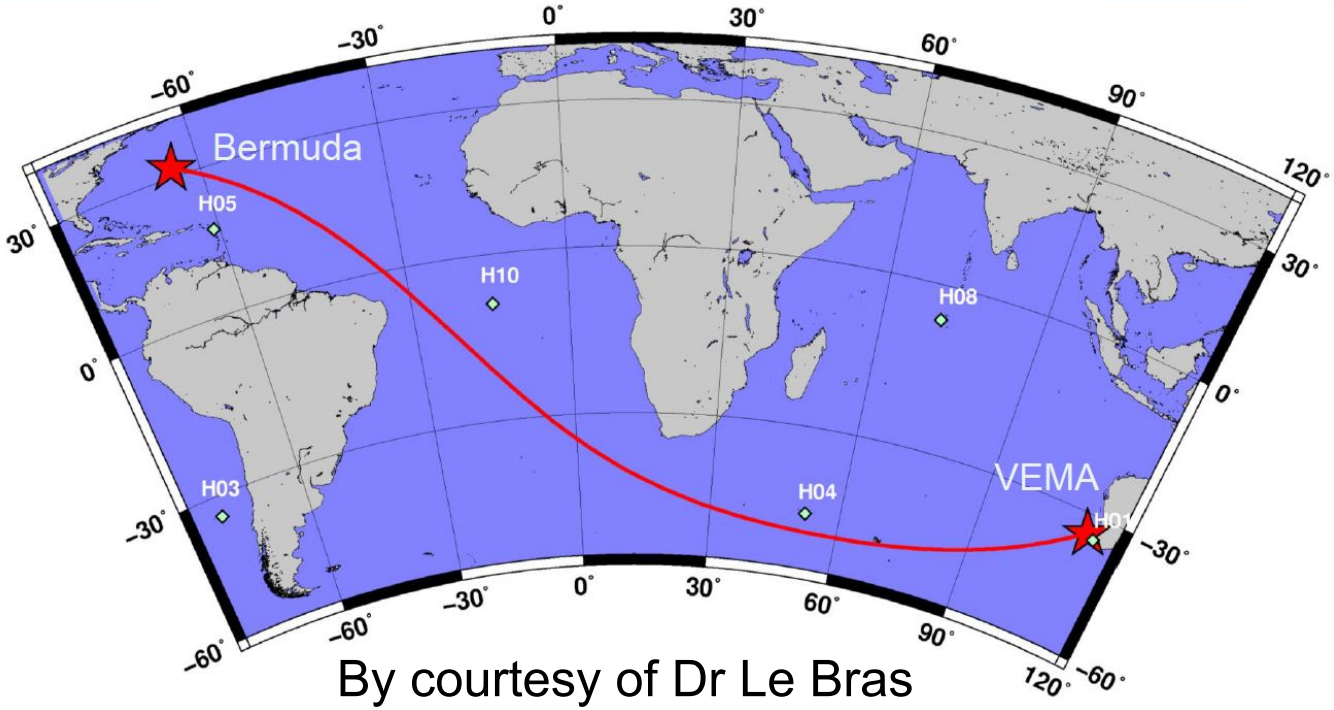
$$\begin{pmatrix} v_\theta^{Yang} \\ v_\varphi^{Yang} \end{pmatrix} = \begin{pmatrix} -\sin\theta^{Yang}\sin\phi^{Yin} & -\cos\varphi^{Yin}/\sin\theta^{Yang} \\ \cos\varphi^{Yin}/\sin\theta^{Yang} & -\sin\theta^{Yang}\sin\phi^{Yin} \end{pmatrix} \begin{pmatrix} v_\theta^{Yin} \\ v_\varphi^{Yin} \end{pmatrix}$$

φ and θ denote the location of point on each grid

Australia-Bermuda experiment (hydroacoustic)

- Conducted in 1960
- Chemical explosion (Amatol) was used as a source of hydroacoustic wave
  - Detonated near Australia
- Several research groups have tried to identify the sound speed profile
  - A mystery: The sound speed observed was faster than expected.

- In the present study, it was used as a reference
  - We did not aim to match the results, but aimed to confirm that the Implementation was sound.



By courtesy of Dr Le Bras

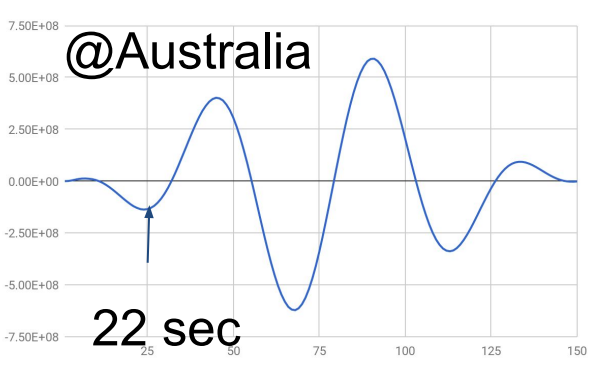
Simulation configuration

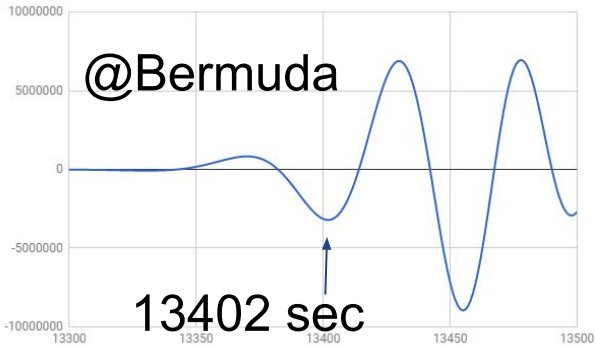
- Sound speed: 1485 m/s.
  - Observed sound speed provided that the ocean is homogeneous
- Depth: 1000m.
  - SOFAR channel level
  - Computational domain (Ocean) is determined with bathymetry contour NOAA at 1000m
- Frequency of the source: 0.02Hz.
  - Corresponds to the width of the pulse observed
- Boundary condition
  - Perfect reflection. If a grid point is on the land, the sound speed becomes 0 m/s
  - More realistic computational wave speed could be used
- No tidal current/coriolis force is considered
- First order differential for space
- Fourth order Runge-Kutta method for time

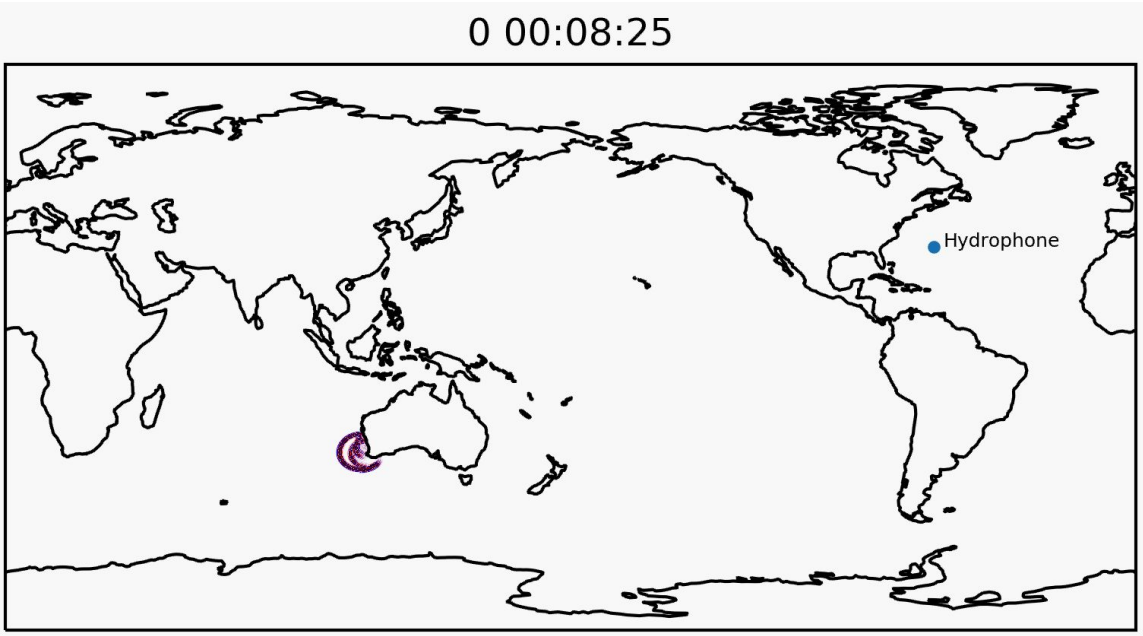
Results of modelling verification

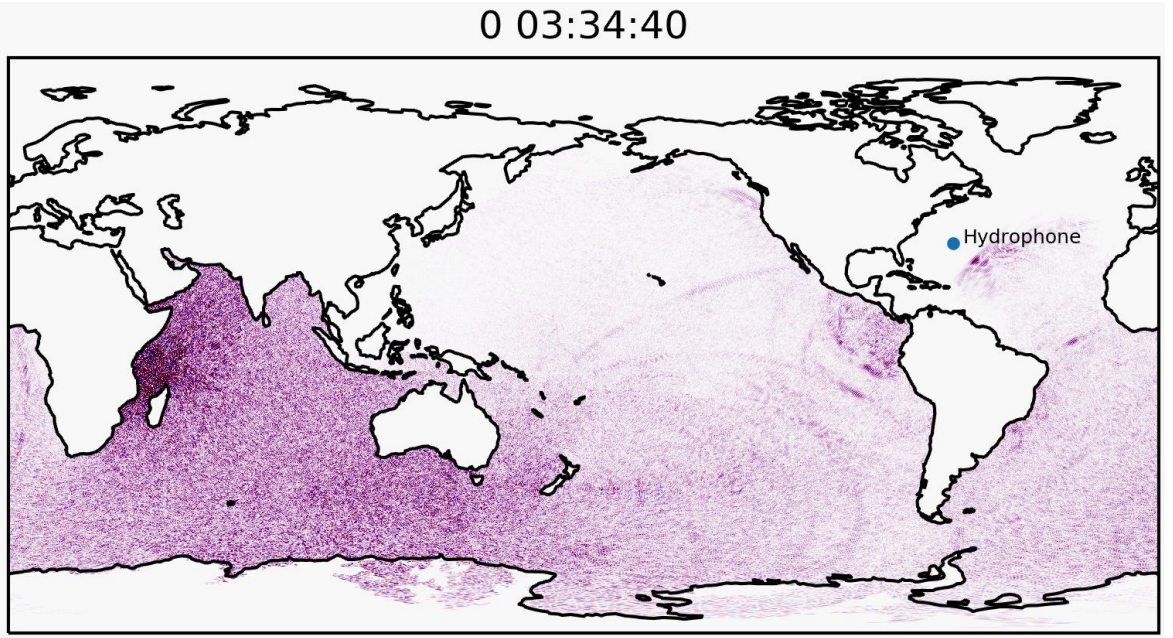
- Travel time from Detonation point (Australia) to Bermuda
  - Observation: 13278 - 13344 sec
  - Simulation: 13380 sec (probably because of the definition of Earth radius)

- Snapshots









Discussion or Impression or how much OpenACC helps us (3D-SSFPE)

- Number of lines
  - Octave: 250
  - C++: 910
  - CUDA/OpenCL: ?
    Not impossible, but we may have needed OpenACC to make a step

- Time spent
  - 3 months to port from Octave to C++ (OpenMP) with <20% effort
    - Mostly, re-forming data to a format understandable for OpenACC
  - 3 months to fight against compilers!
    - After we found the working options, just a couple of weeks