EGU Galileo Conferences

BackTrackBB workflow for seismic source detection and location with PyCOMPSs parallel computational framework

Natalia Poiata[1,2], Javier Conejero[3], Rosa M. Badia[3] and Jean-Pierre Vilotte[4]

1) National Institute for Earth Physics, Romania

2) International Seismological Centre, UK

3) Department of Computer Sciences, Barcelona Supercomputing Center (BSC-CNS), Spain

4) Université Paris Cité, Institut de Physique du Globe de Paris, France.

INFP

BSC Barcelona Supercomputing Center Centro Nacional de Supercomputación

IPGP | Université de Paris

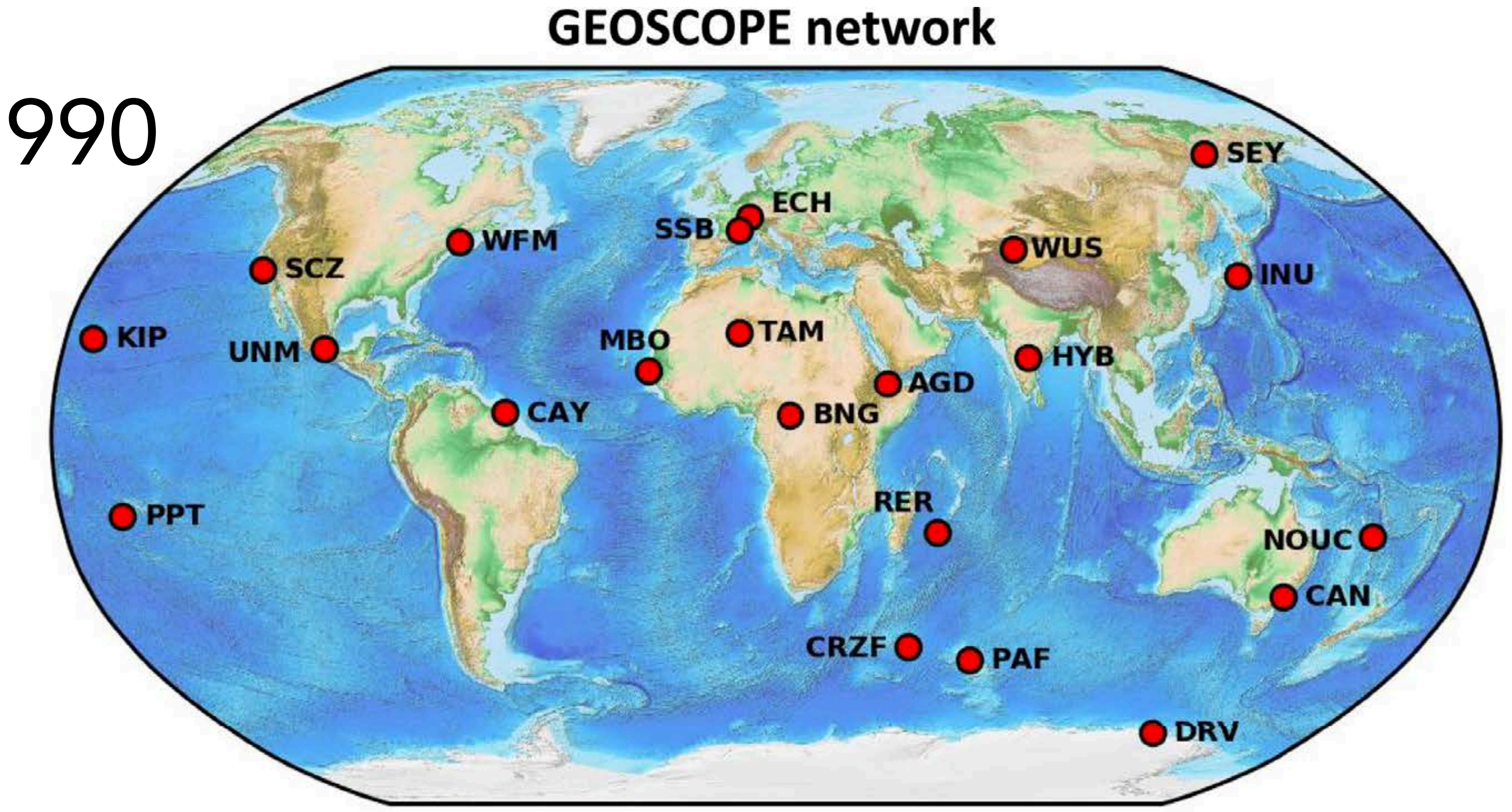
May 24, 2023

1. Introduction: Big Data in Seismology

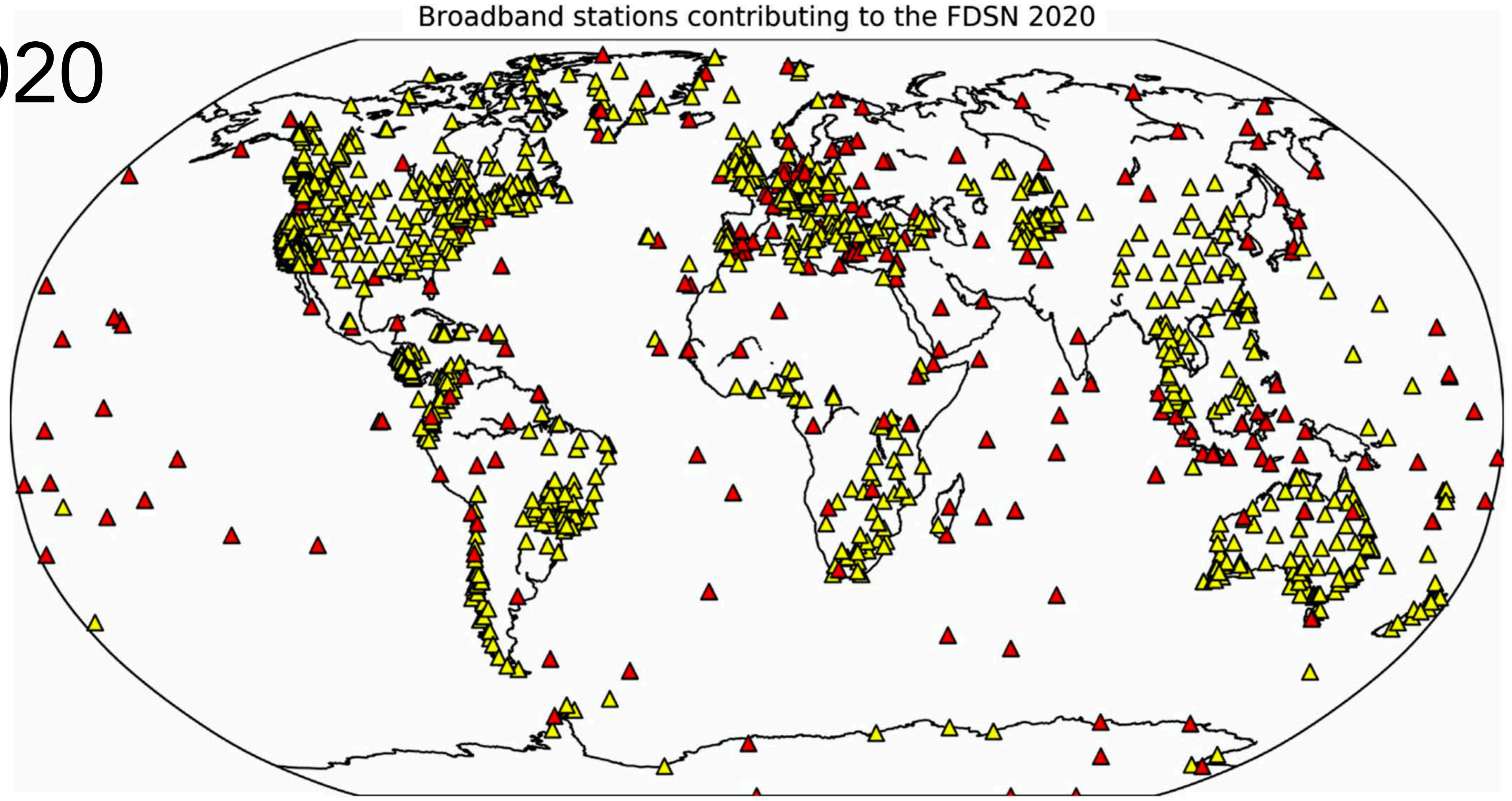
Advances in Seismological Observations: increasing density of seismic networks

Global broad-band seismic stations

1990

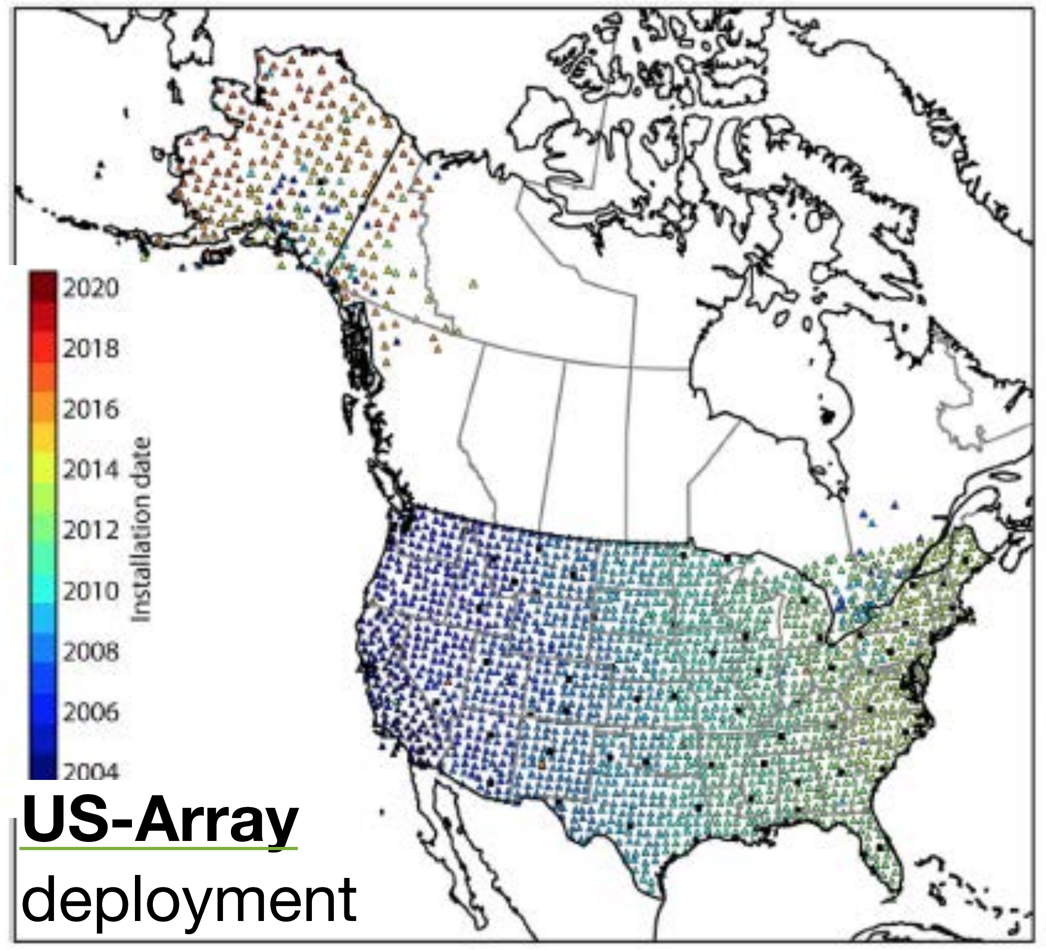


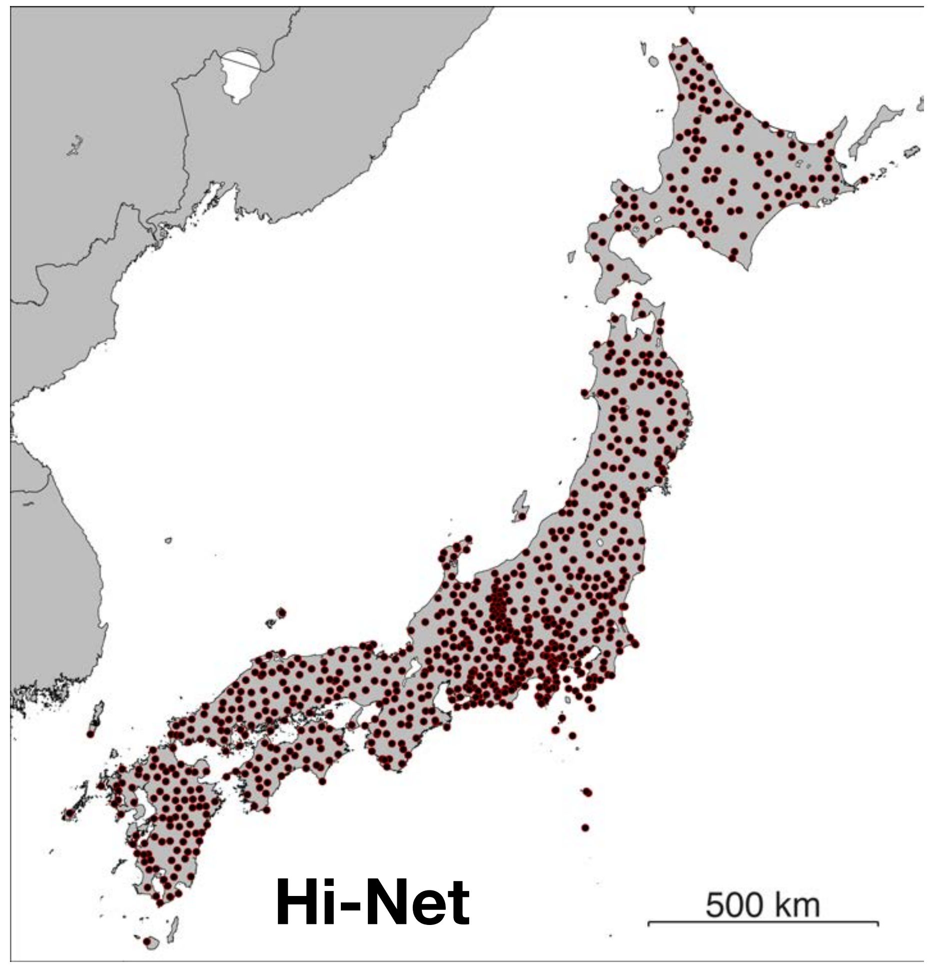
2020

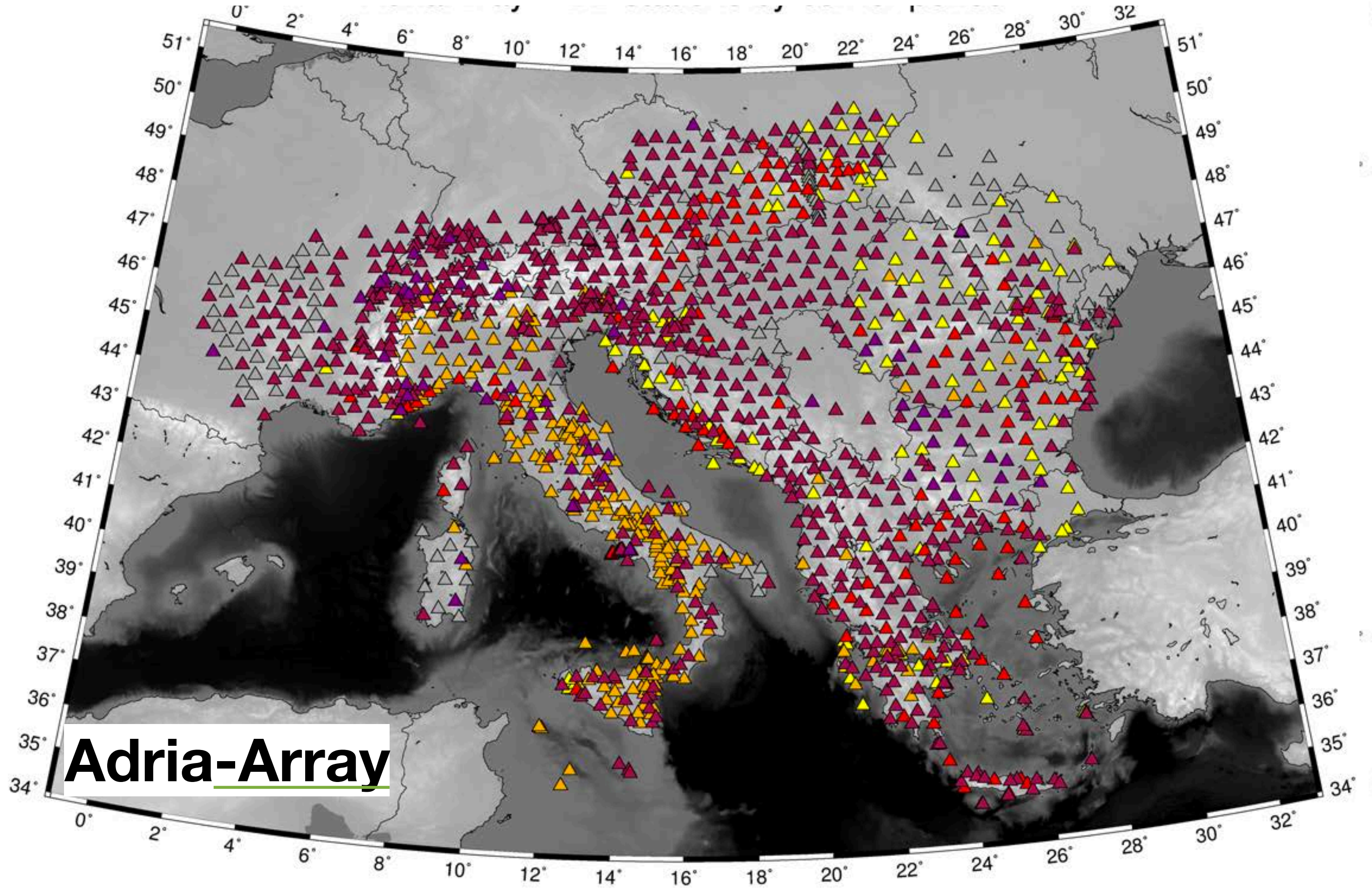


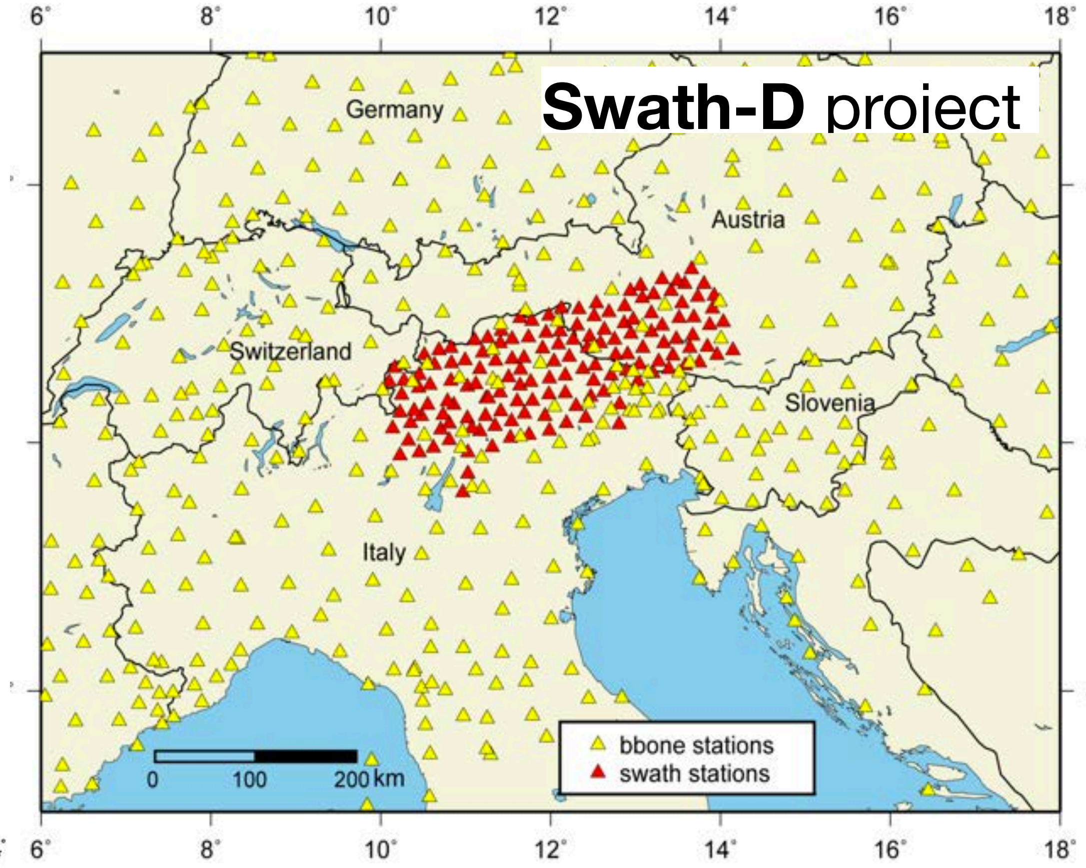
from Arrowsmith et al. (2022)

Regional and local deployments







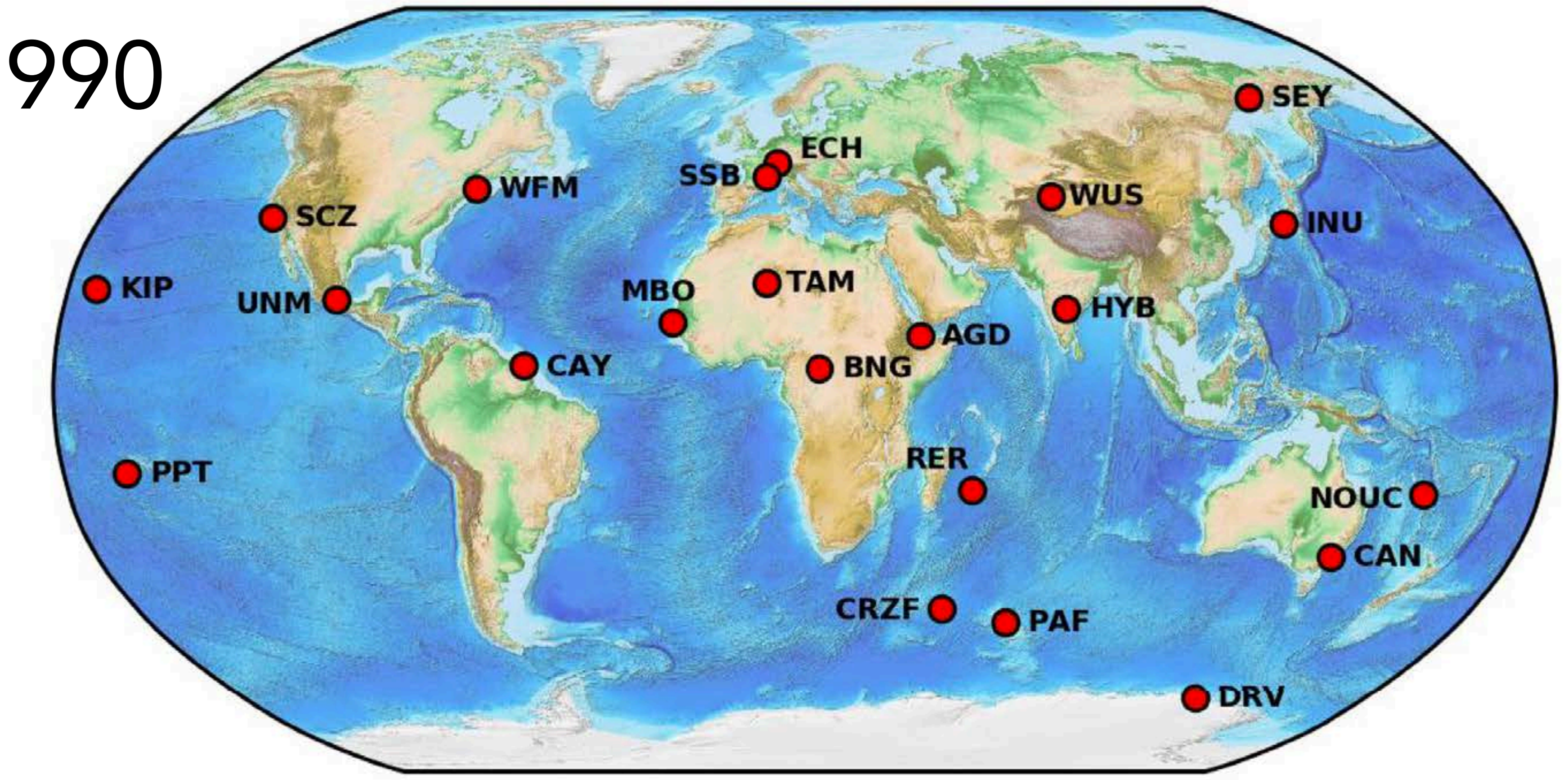


1. Introduction: Big Data in Seismology

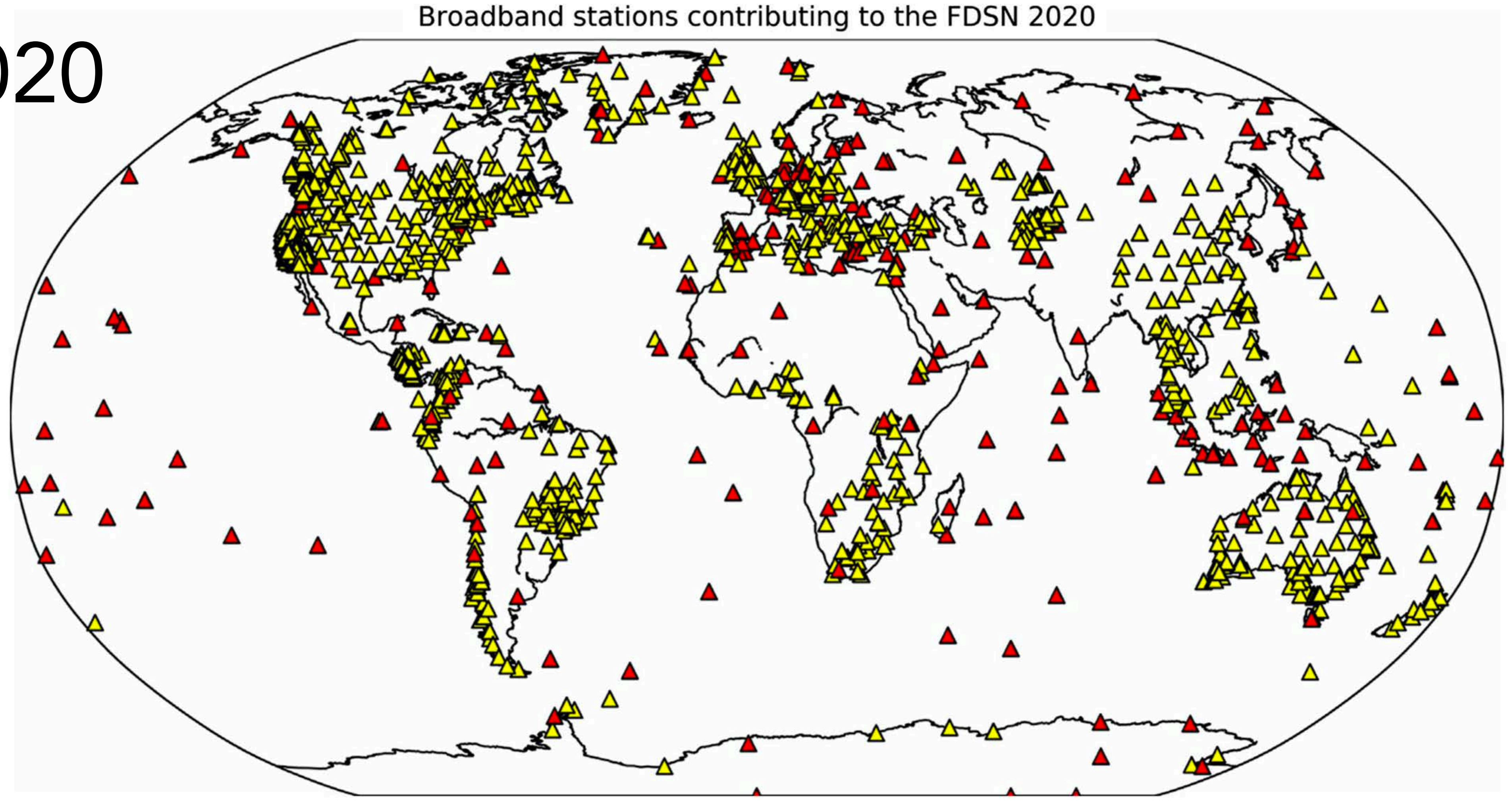
Increased Volumes of Archived Seismic Data

Global broad-band seismic stations

1990

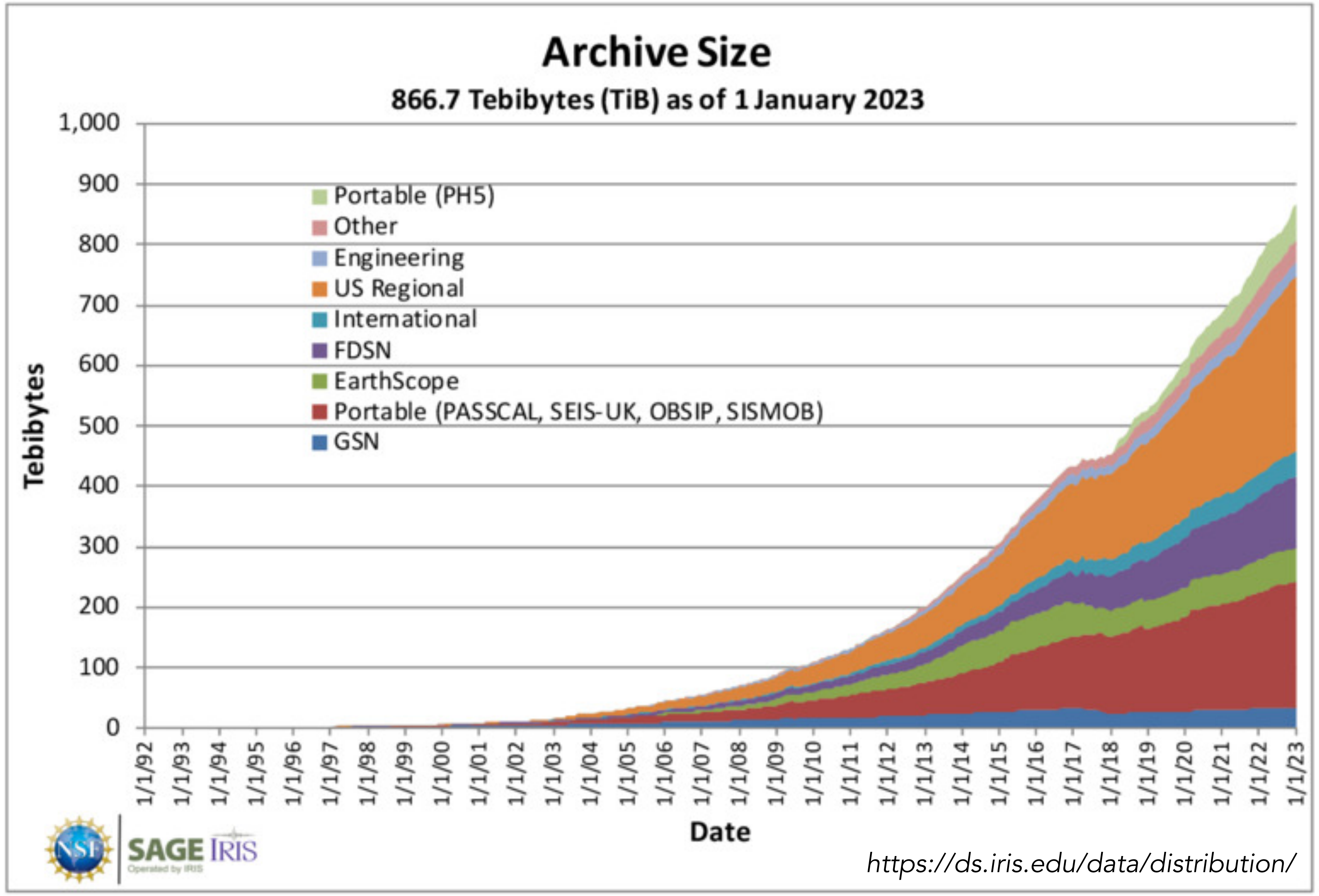


2020



from Arrowsmith et al. (2022)

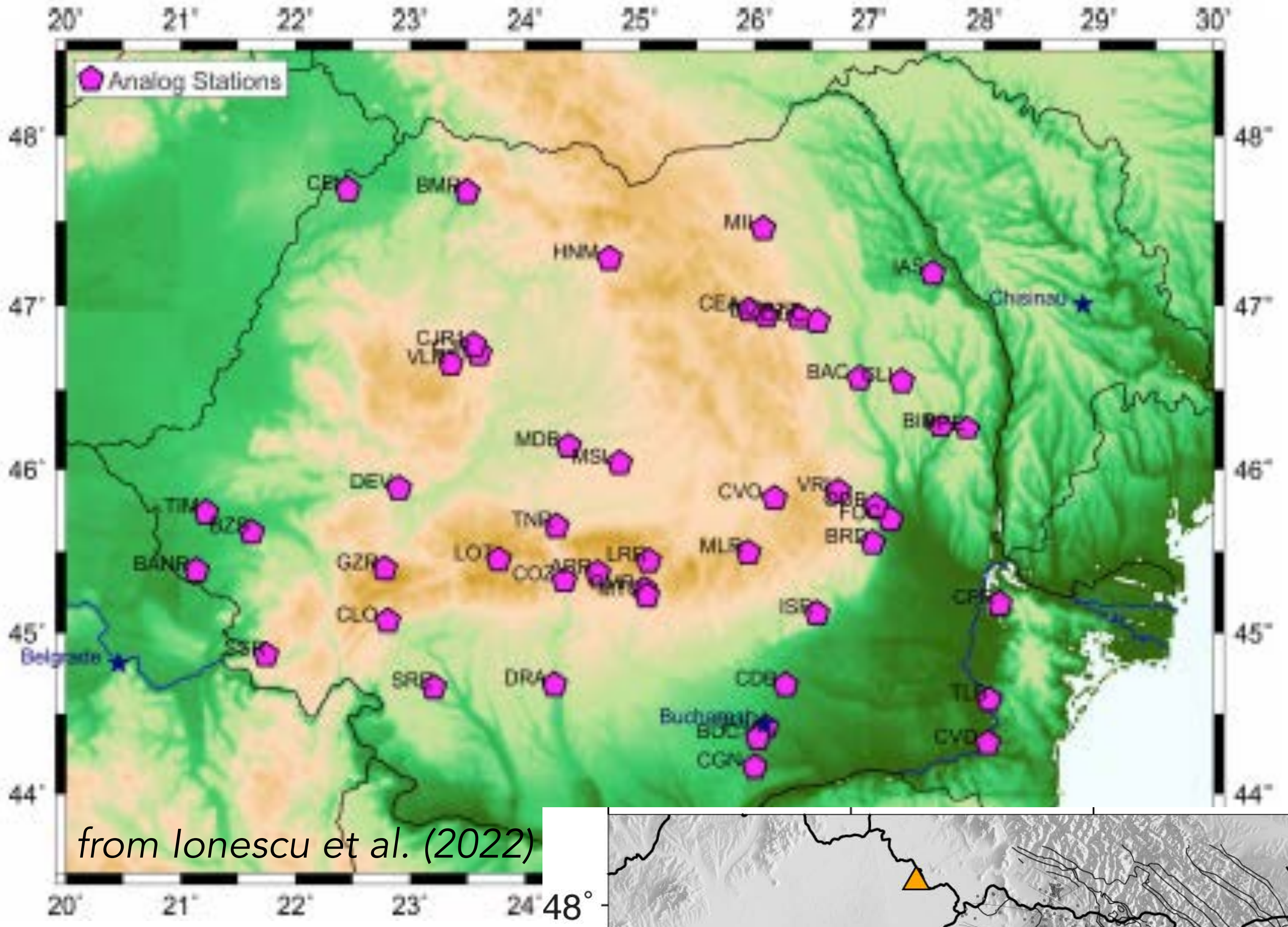
Globally-archived seismological data

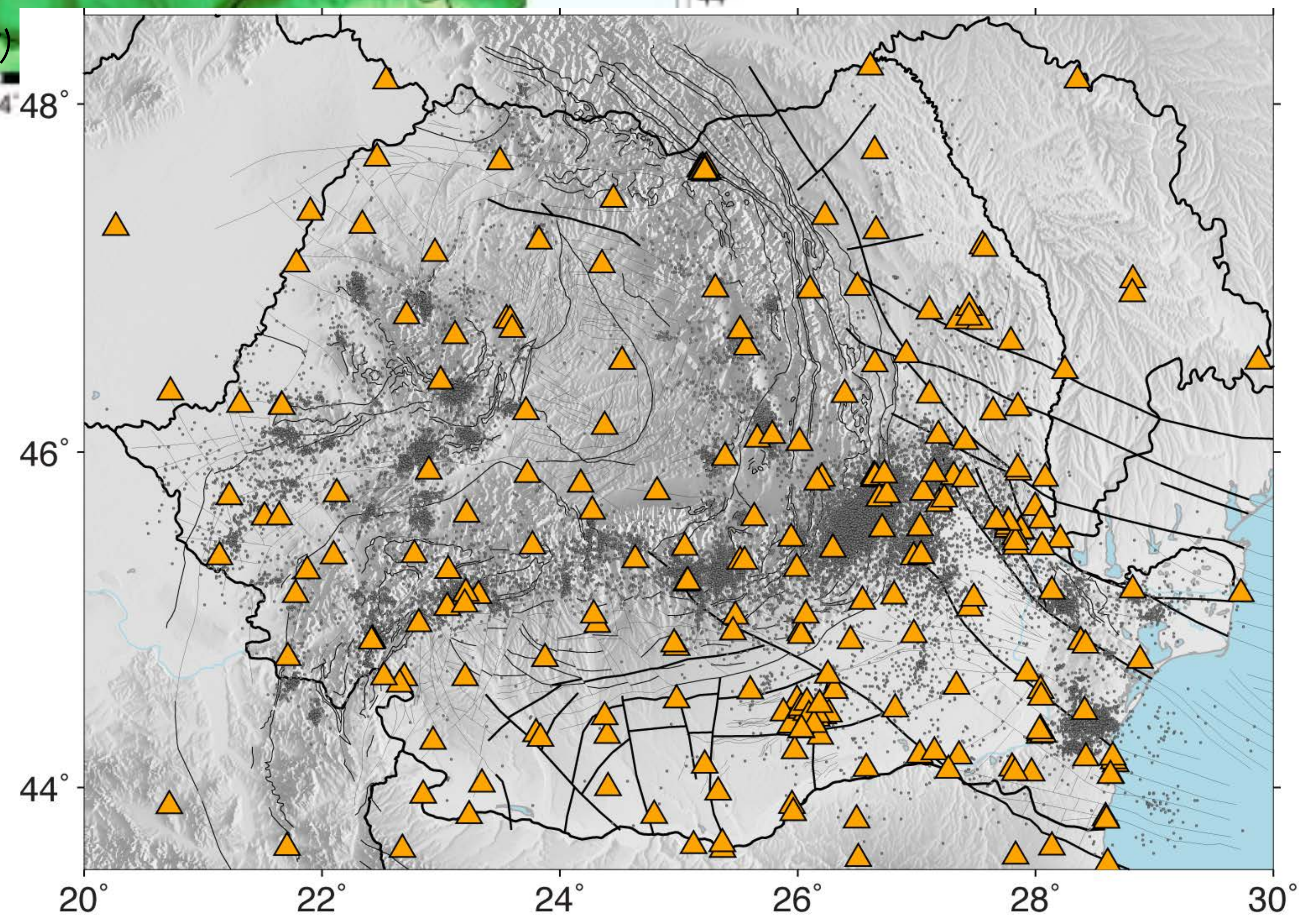


1. Introduction: Big Data in Seismology

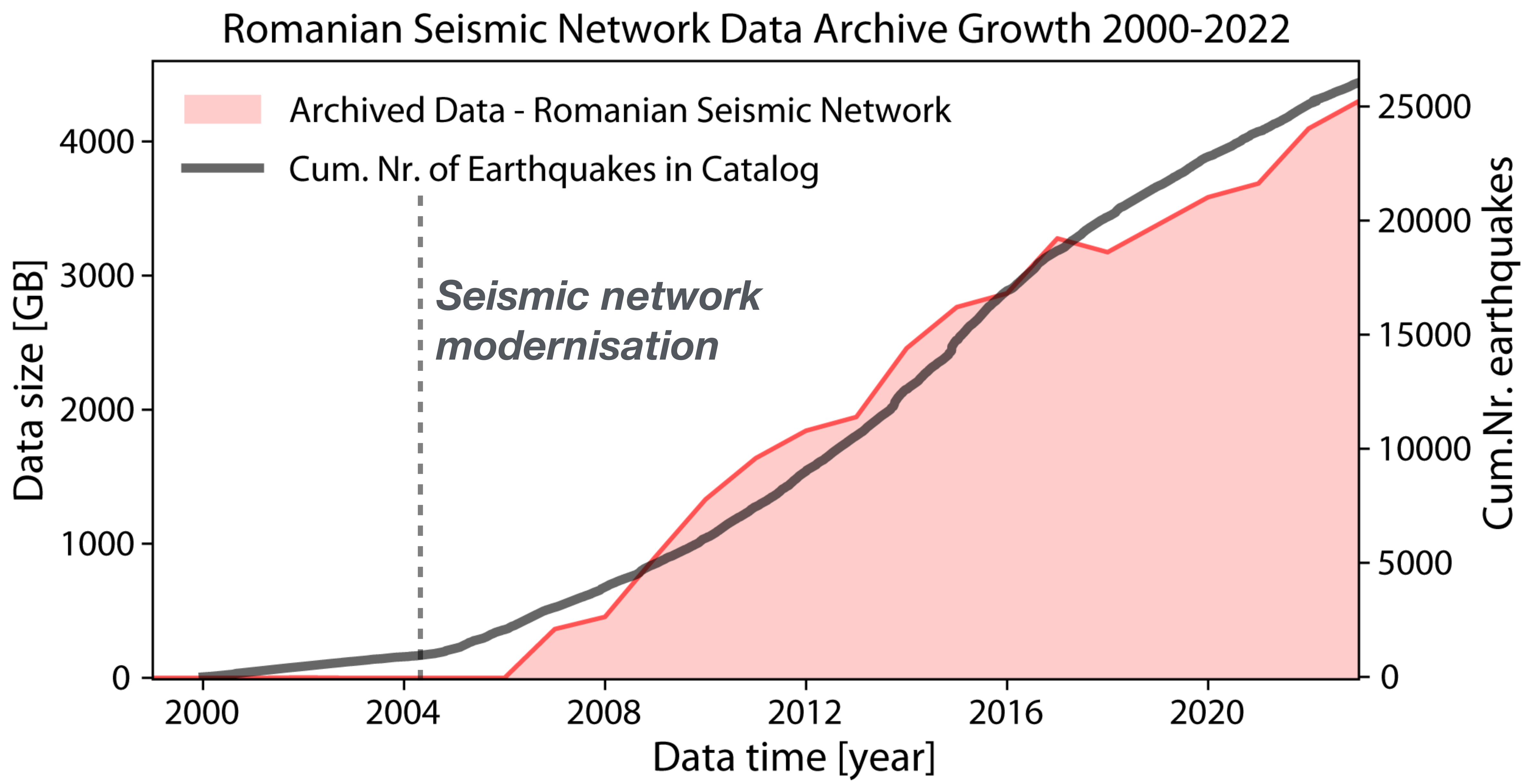
Evolution on a scale of a single data-center - Romanian Seismic Network

Current state of the Romanian Seismic Network





Locally-archived seismological data



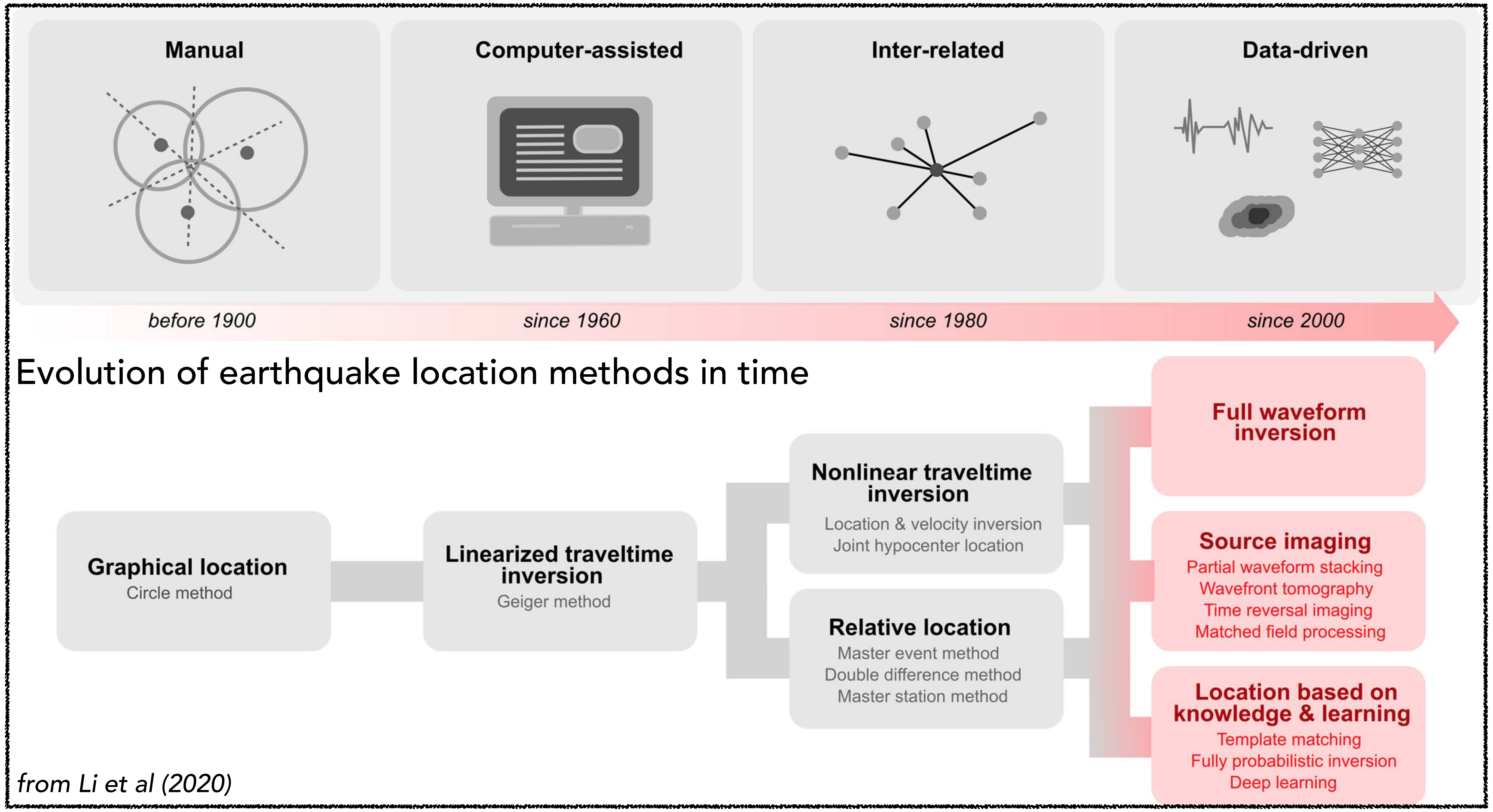
1. Introduction: Methodological Advances

Transformative evolution of earthquake detection and location approaches

Rise of new data-driven methods operating on continuous seismic data

Targeting wide range of seismic source types and different environments (tectonic, volcanic, anthropogenic, …)

Applicable for continuous (real-time) monitoring of seismic activity



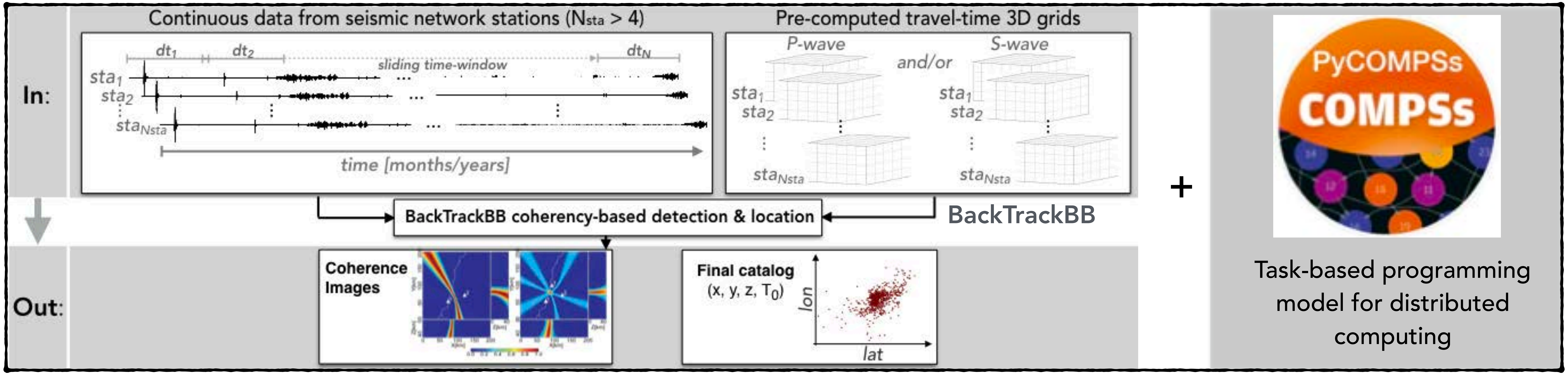
from Li et al (2020)

2. BackTrackBB with PyCOMPSs

Optimising earthquake detection and location for HPC resources

Scalable parallelisation of automatic full waveform, coherency method for seismic source detection and location
Framework for efficient and easily reproducible analysis of continuous seismic records from distributed networks

BackTrackBB coherency-based detection and location method *PyCOMPSs framework*

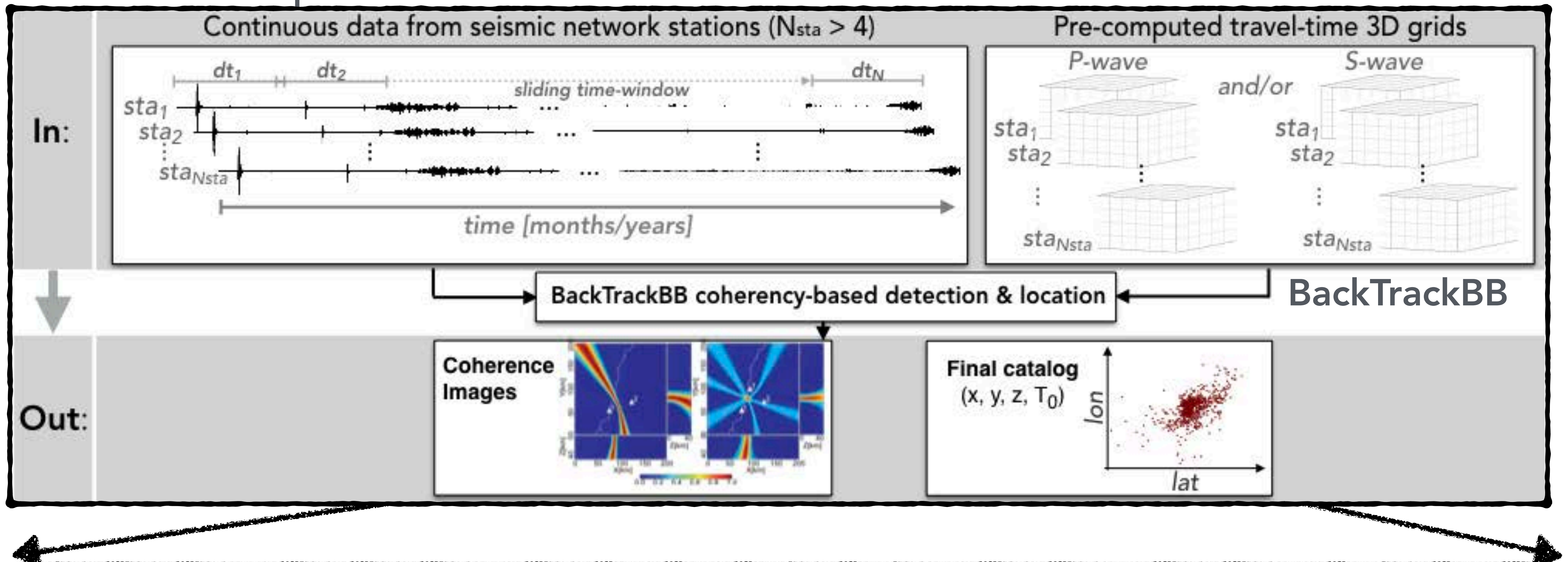


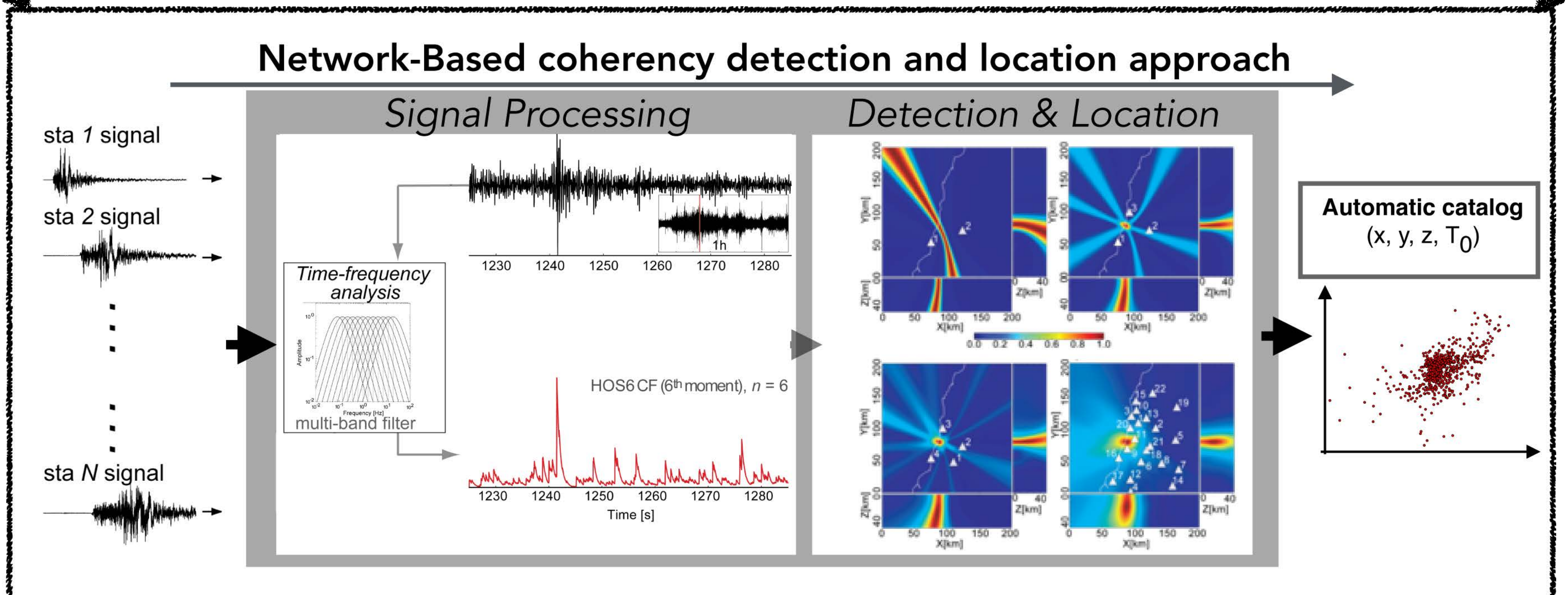
BackTrackBB with PyCOMPSs
framework for efficient and reproducible earthquake detection and location using continuous data

2.1. BackTrackBB with PyCOMPSs - Backbone methods

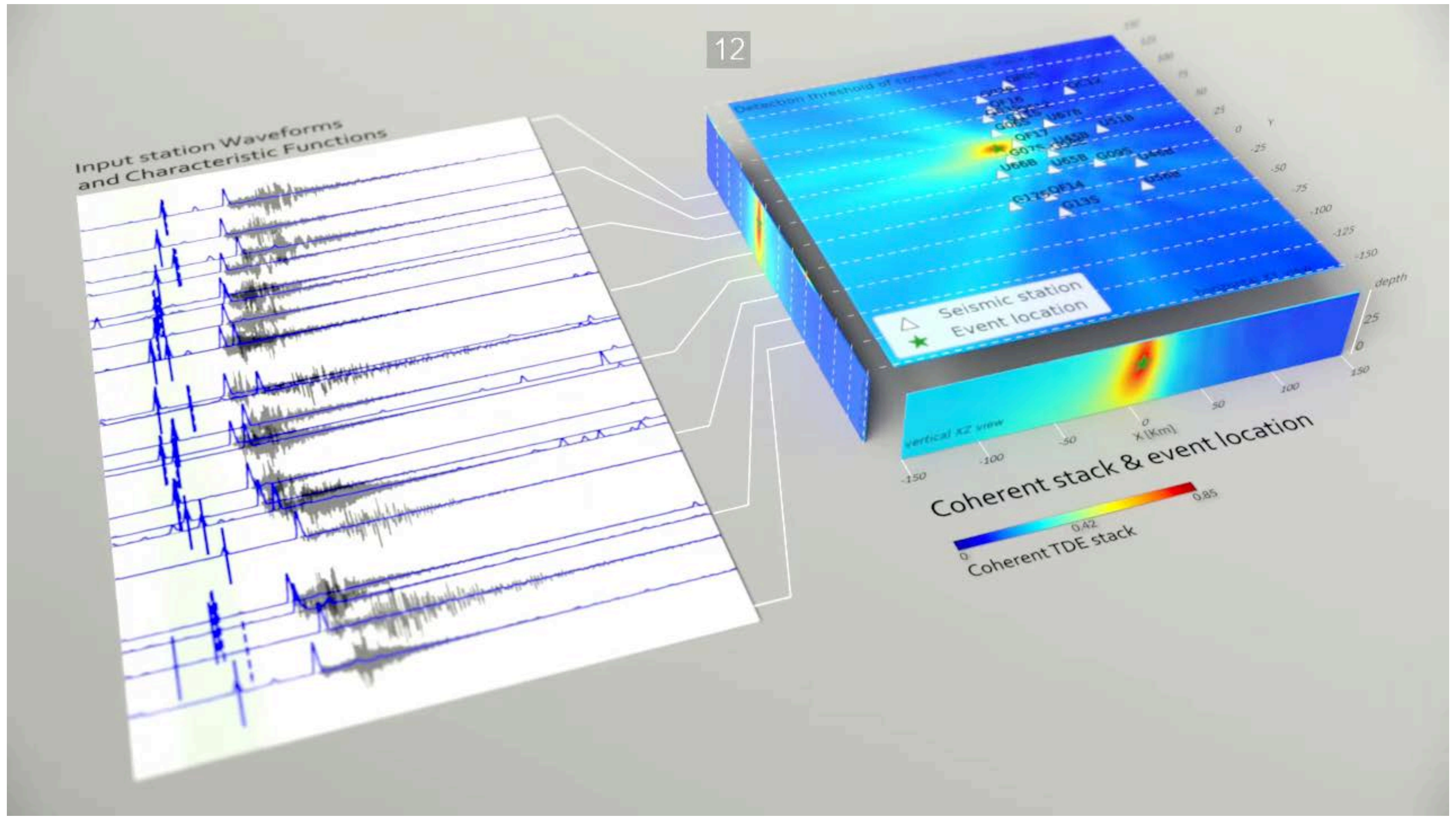
BackTrackBB - earthquake detection and location

Schematic presentation of the method





BackTrackBB detection and location method (Poiata et al., 2016)



Automatic coherency-based detection and location method making use of continuous seismic data

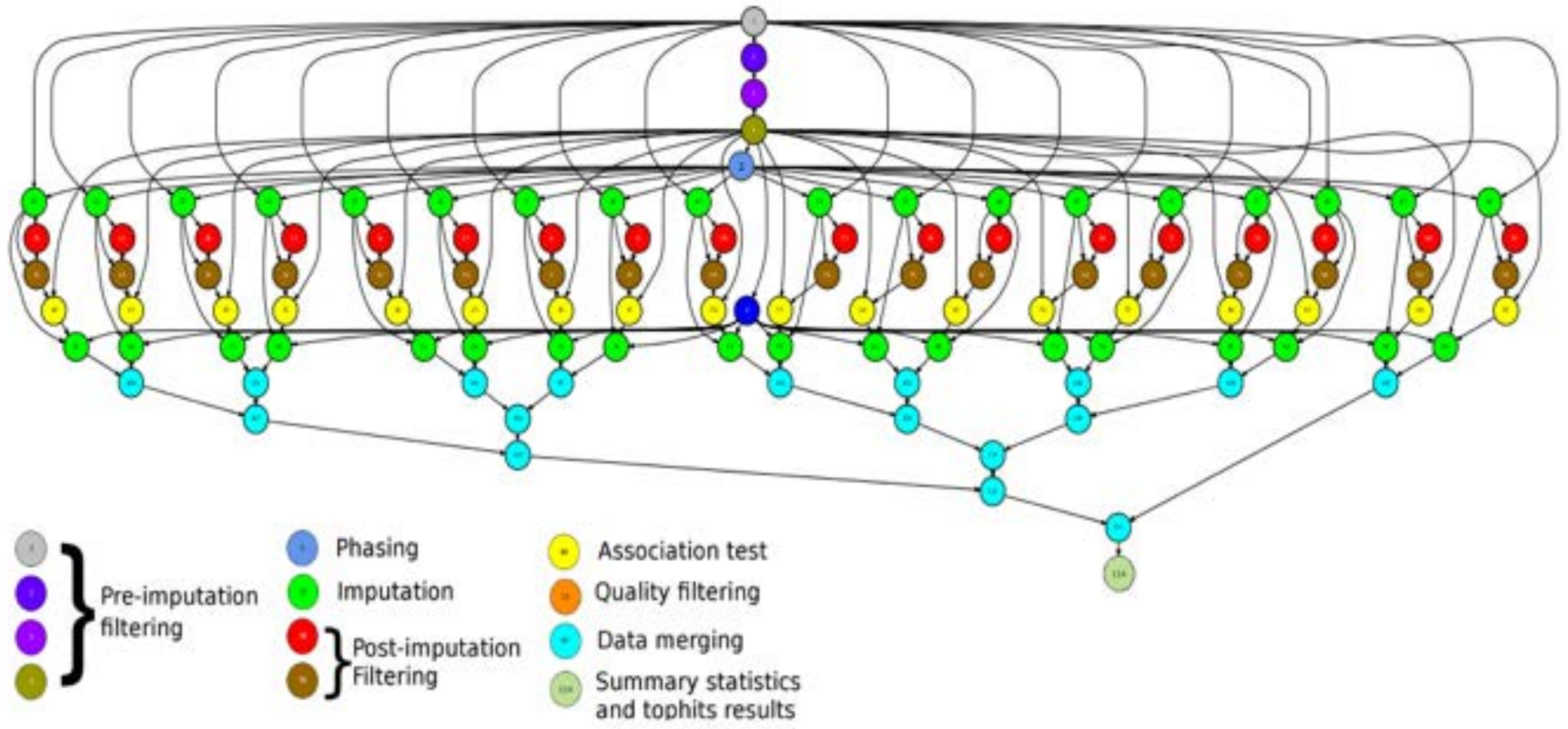
Efficient in detecting events from low signal-to-noise ratio records

Capable to significantly improve earthquake's catalogues

Data streaming analysis capabilities

2.1. BackTrackBB with PyCOMPSs - Backbone methods

PyCOMPSs - task-based programming model for Python application



PyCOMPSs task-dependency graph example

- PyCOMPSs (BSC) task-based programming model for Python applications (Tejedor et al., 2017)
- Framework facilitating development of parallel computational workflow in Python (for sequential codes)
- Relies on runtime for dynamically extracting parallelism among tasks and executing them in distributed environments (HPC clusters, cloud infrastructures, …)
- Runtime system in charge of exploiting inherent concurrency of the script, detecting data dependency among tasks and sending them to available resources
- Transparent to user
- Comes with performance analysis and monitoring tools

2.2. BackTrackBB with PyCOMPSs - Implementation

BackTrackBB code - main specifications

- Python-based with C modules and Obspy use for basic signal processing
- Embarrassingly-parallel computations
- Parallel (CPU) capability using Python's *multiprocessing* module
- Input - configuration parameters and data (waveforms & 3D travel-time grids)
- Output - detected and located events ASCII file(s) & plots

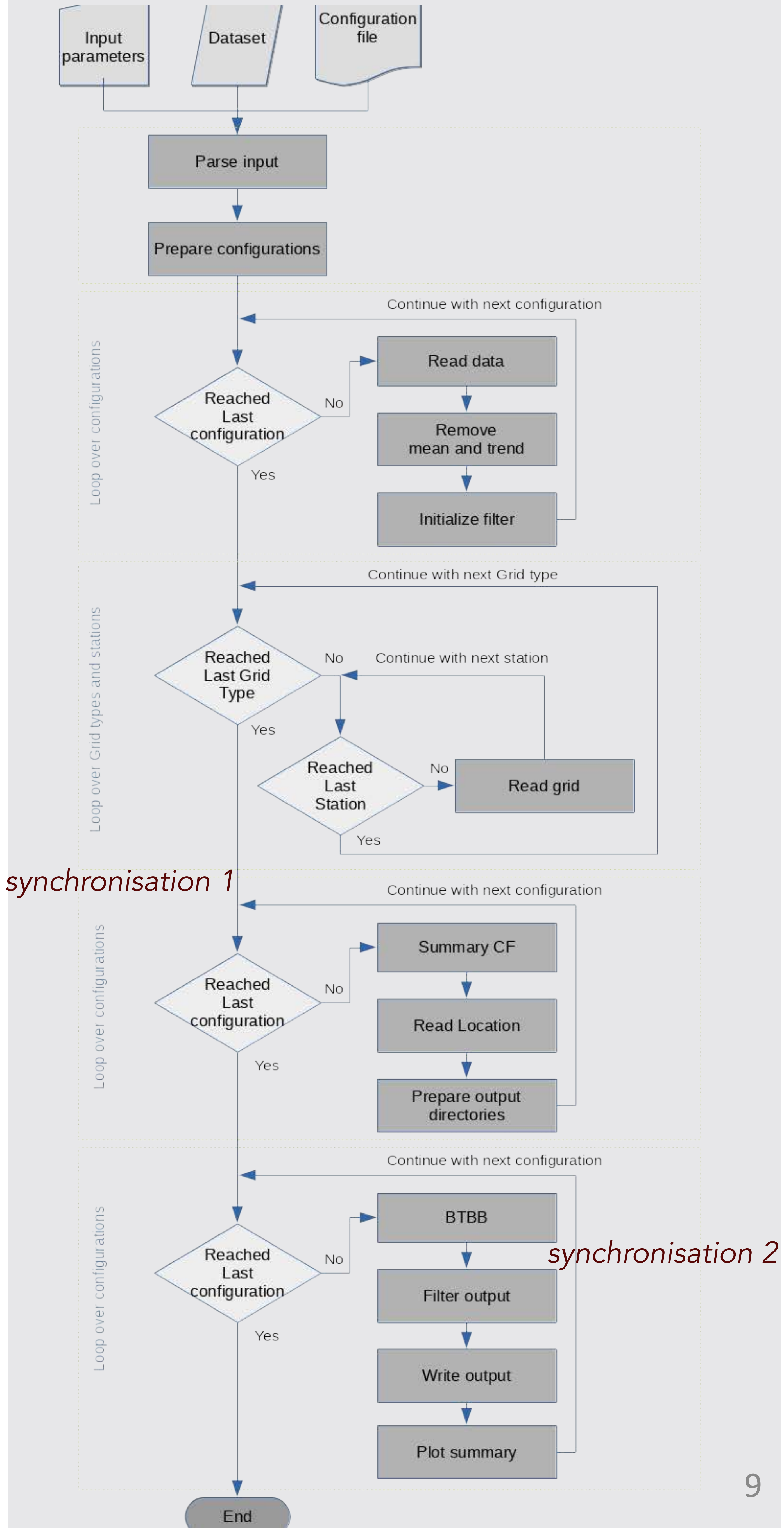
Parallelization of BackTrackBB with PyCOMPSs

- Determine parallelization opportunity using operation flowchart
- (considering number of loops and iterations per loop)
- Identify functions as candidate for PyCOMPSs ***tasks*** (decorators recognised by runtime)
- Include PyCOMPSs task decorators into the BackTrackBB code
- Identify required synchronisations during execution
- Identify required developments - support for dictionaries containing future objects (included)
- Analyse extracted parallelism and test the implementation

Example BackTrackBB task annotation of read_grid function

```python
@task(returns=2, grid_bname=FILE_IN)
def read_grid(grid_bname):
    grid = NLLGrid(grid_bname)
    return grid, (grid.sta_x, grid.sta_y)
```

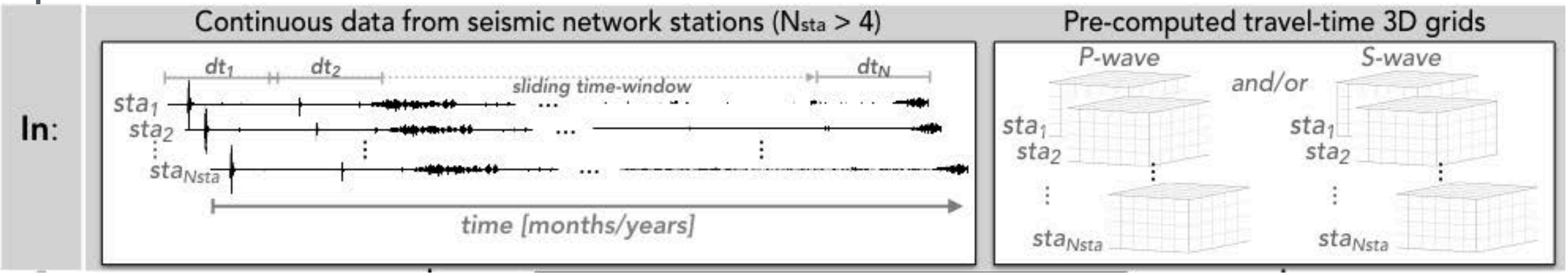
BackTrackBB flowchart



2.2. BackTrackBB with PyCOMPSs - Implementation

Illustration of the extracted parallelism

Input data scheme of BackTrackBB



Data are analysed in 1 hour time-period

Detection and location carried in sliding time-window within each 1hour time-period

Each time-periods is split in blocks of sliding time-windows - more efficient implementation

Task dependency graphs

Confirms good parallelism, able to produce an avalanche of independent tasks

Efficiently uses available resources to its best

2.2. BackTrackBB with PyCOMPSs - Performance testing

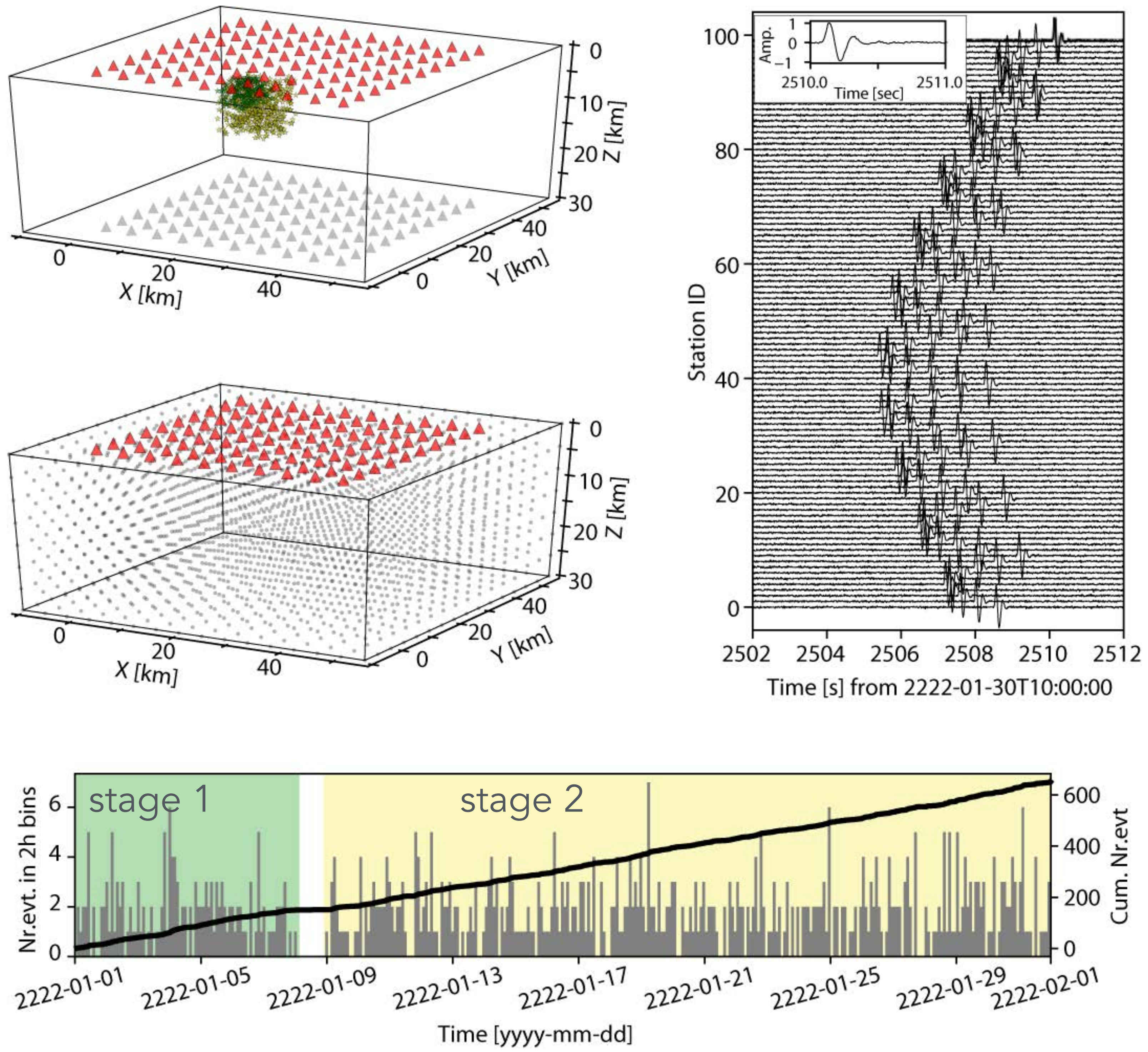
Test platform description

MareNostrum IV supercomputer, Barcelona Supercomputing Center (BSC)
3456 nodes Intel Xeon Platinum 8120 CPU's (24 cores, 2.1Gbit/s, 32 MB cache)
Main memory - 96 GB (216 nodes with 380 GB)
100 Gbit/s Intel Infiniband and 10 Gigabit Ethernet network interconnections

Tests ran with max 37 compute nodes (1 master - N worker node configuration)

Test datasets setup

Synthetic dataset: 100 stations & 1 month data

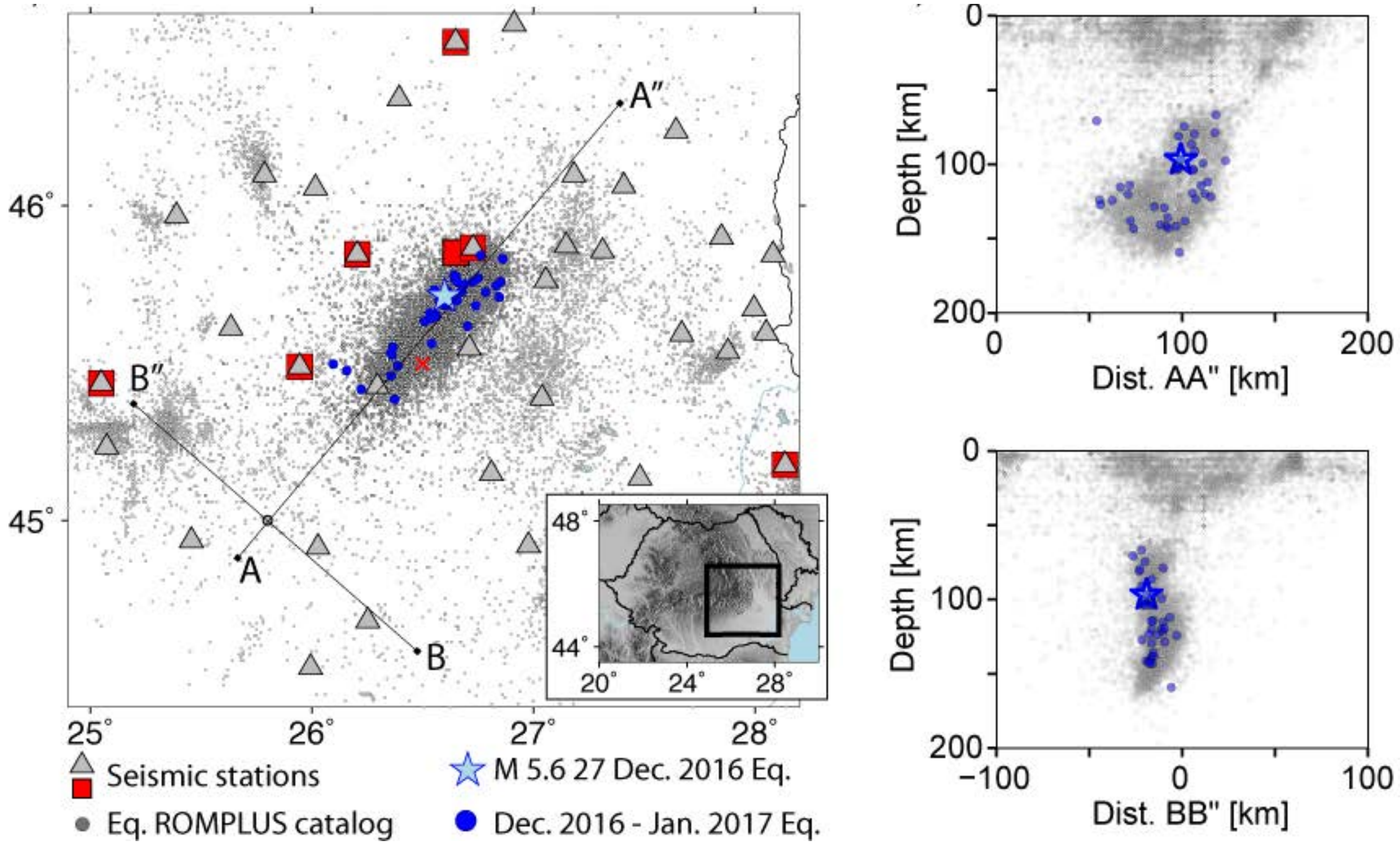


Stress-testing
Controlled environment
Allows multiple parameter testing

Observed dataset
Dealing with data quality issues
(*gaps in recorded data*)

Real-case dataset: Vrancea (Romania)

7 - 22 stations & 2 month data

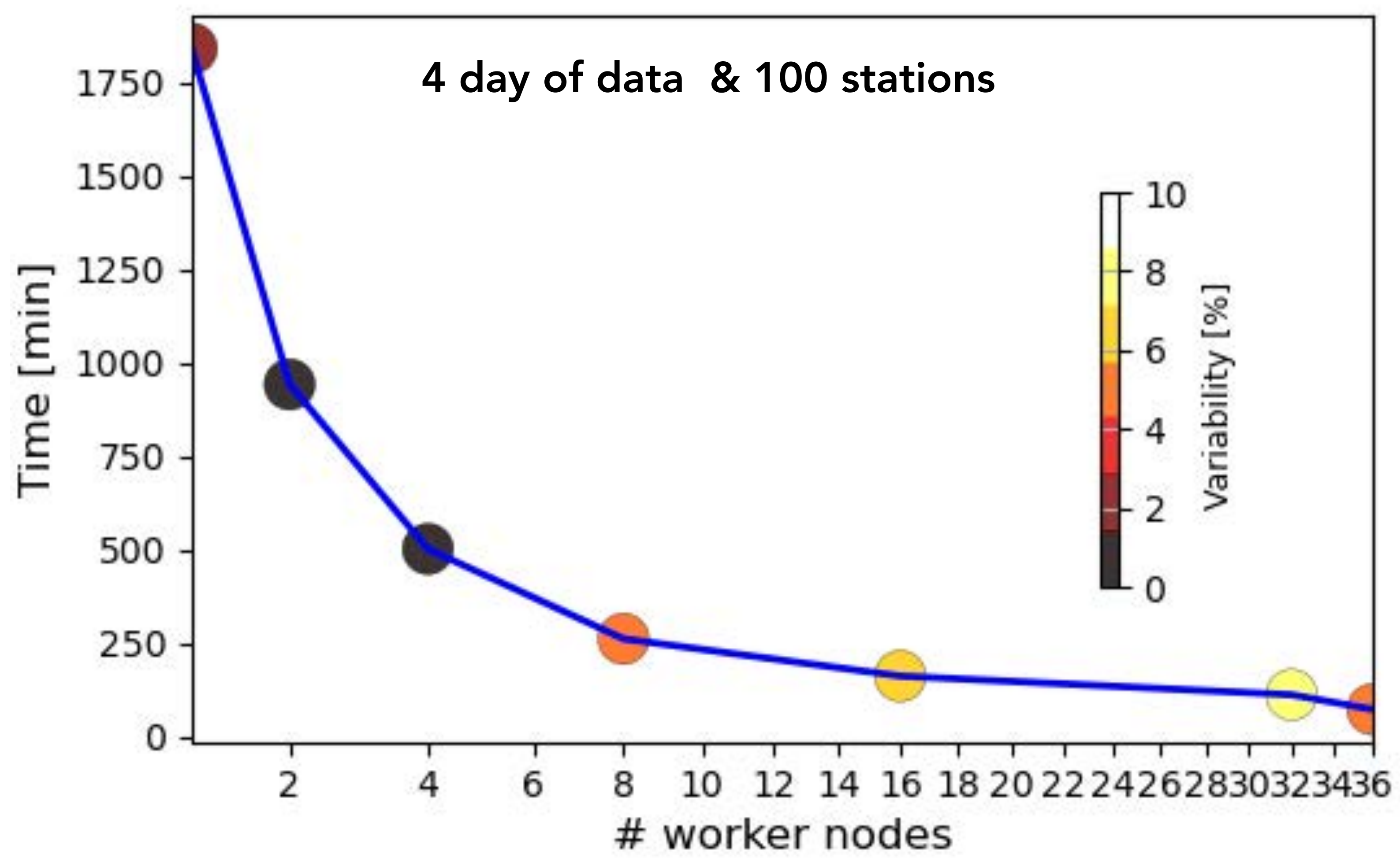


2.2. BackTrackBB with PyCOMPSs - Performance testing results

Performance and parallelisation behaviour

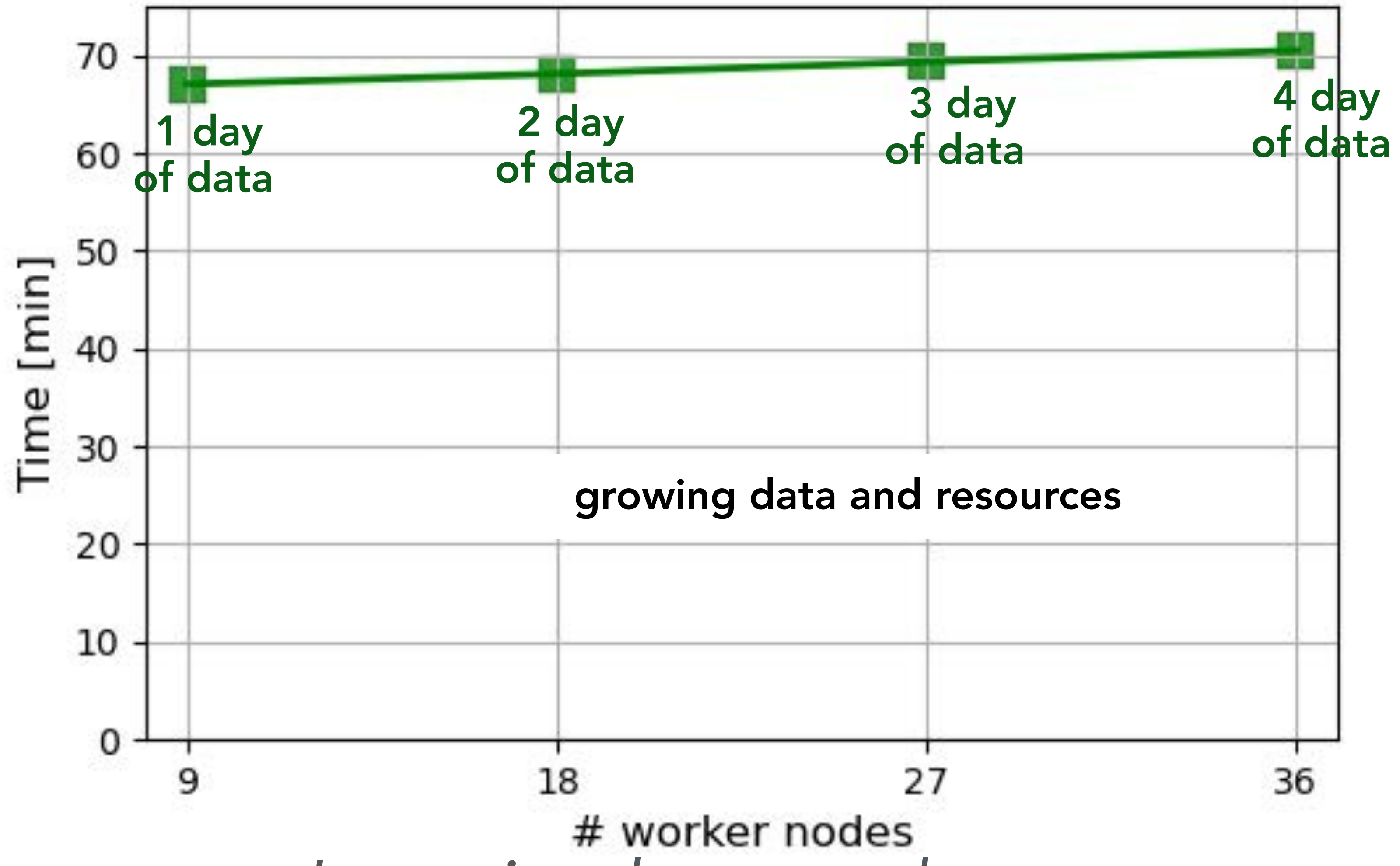
Scalability analysis - performance for synthetic dataset

Strong scaling testing case



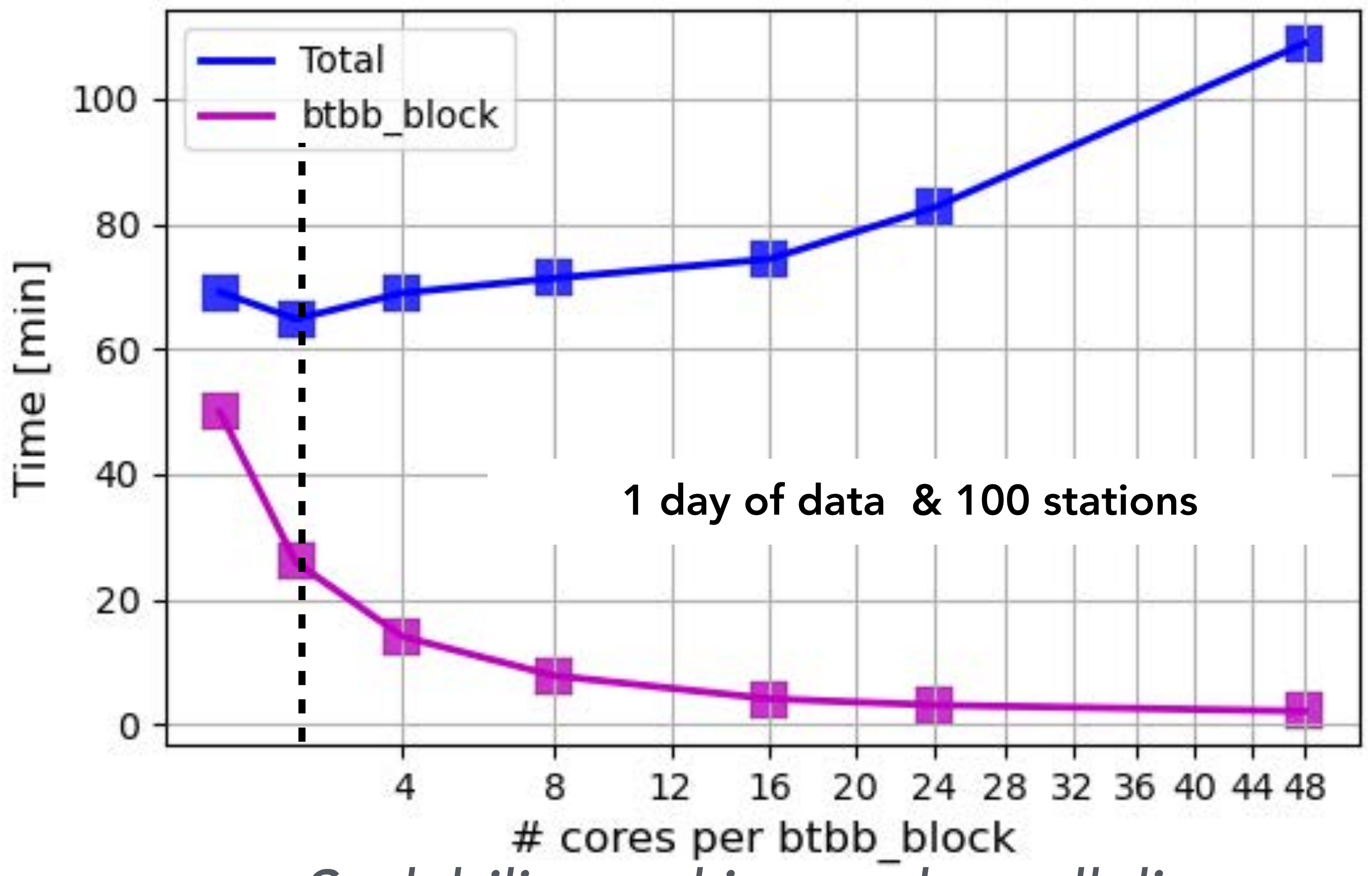
Same dataset and increasing resources

Weak scaling testing case



Increasing dataset and resources

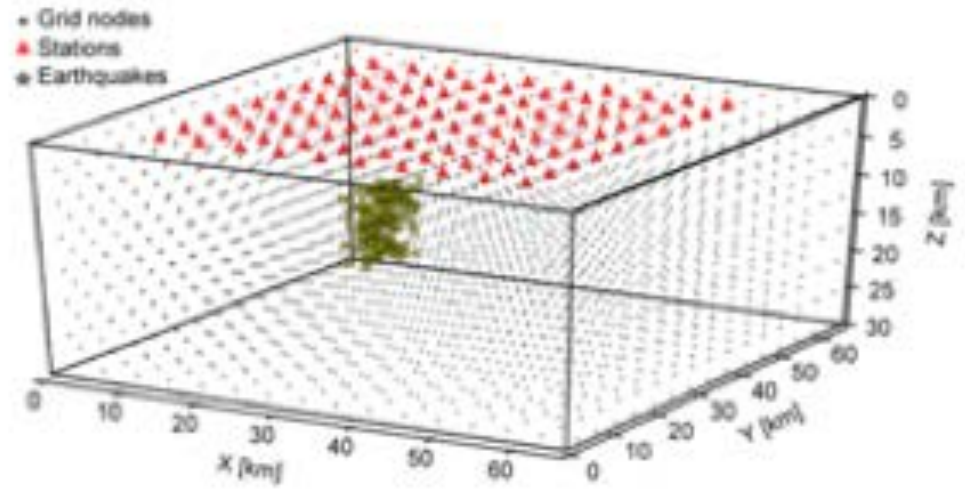
Performance with internal parallelism



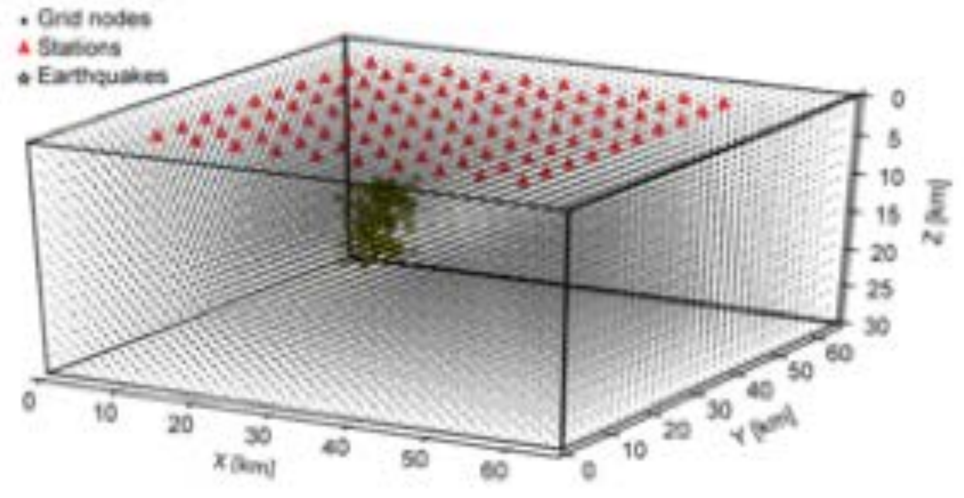
Scalability and internal parallelism

Memory performance testing

1.0 km grid discretisation - 126,750 nodes



0.5 km grid discretisation - 11,014,000 nodes



Using different : 3D model-grid discretisation schemes
Number of analysed stations

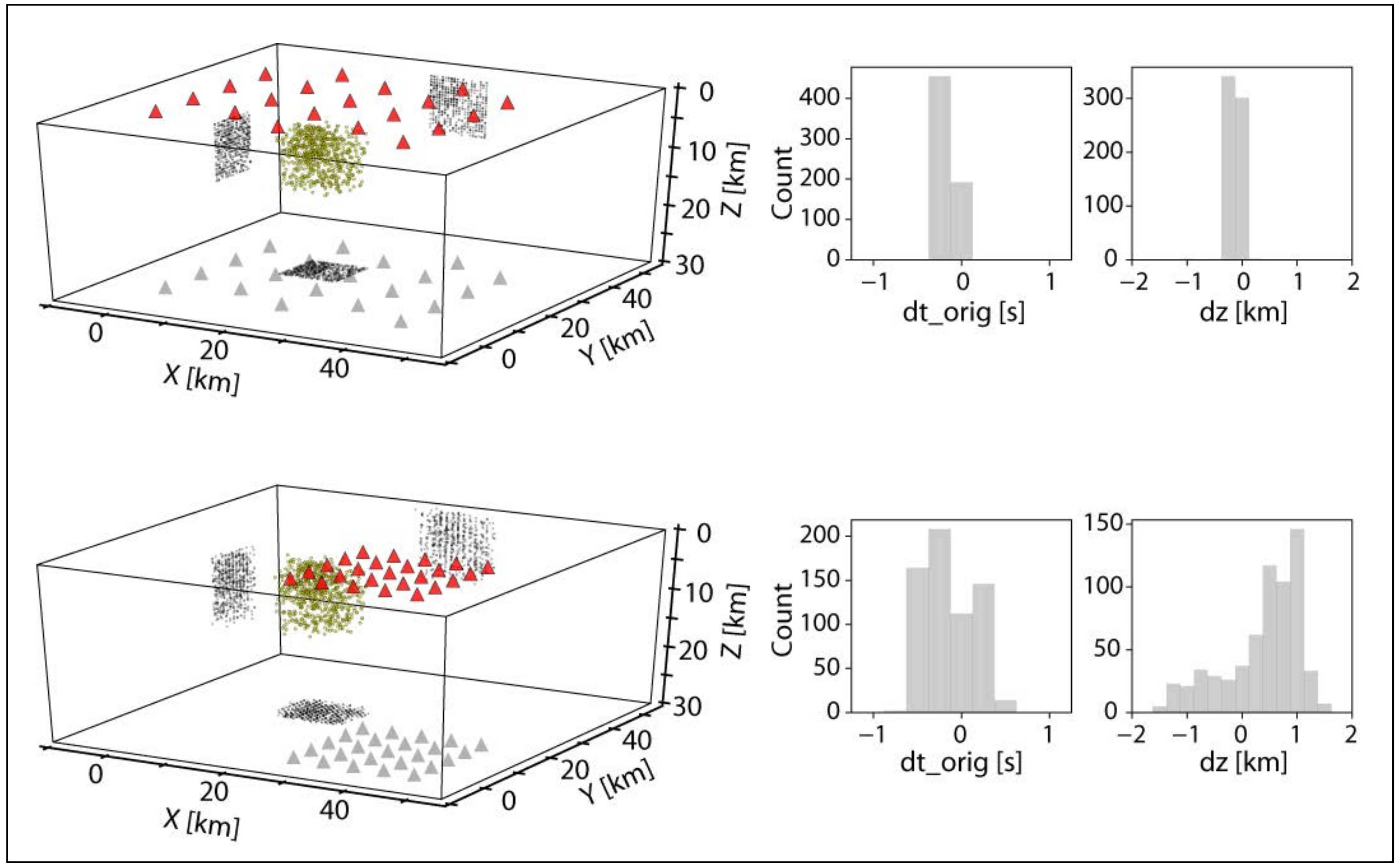
A memory intensive problem - reaches limitations of setup size (e.g., number of stations and analysed region)
main limitation

2.3. BackTrackBB with PyCOMPSs - Performance testing results

Main implications for the seismological data analysis detection and location problem

Synthetic dataset

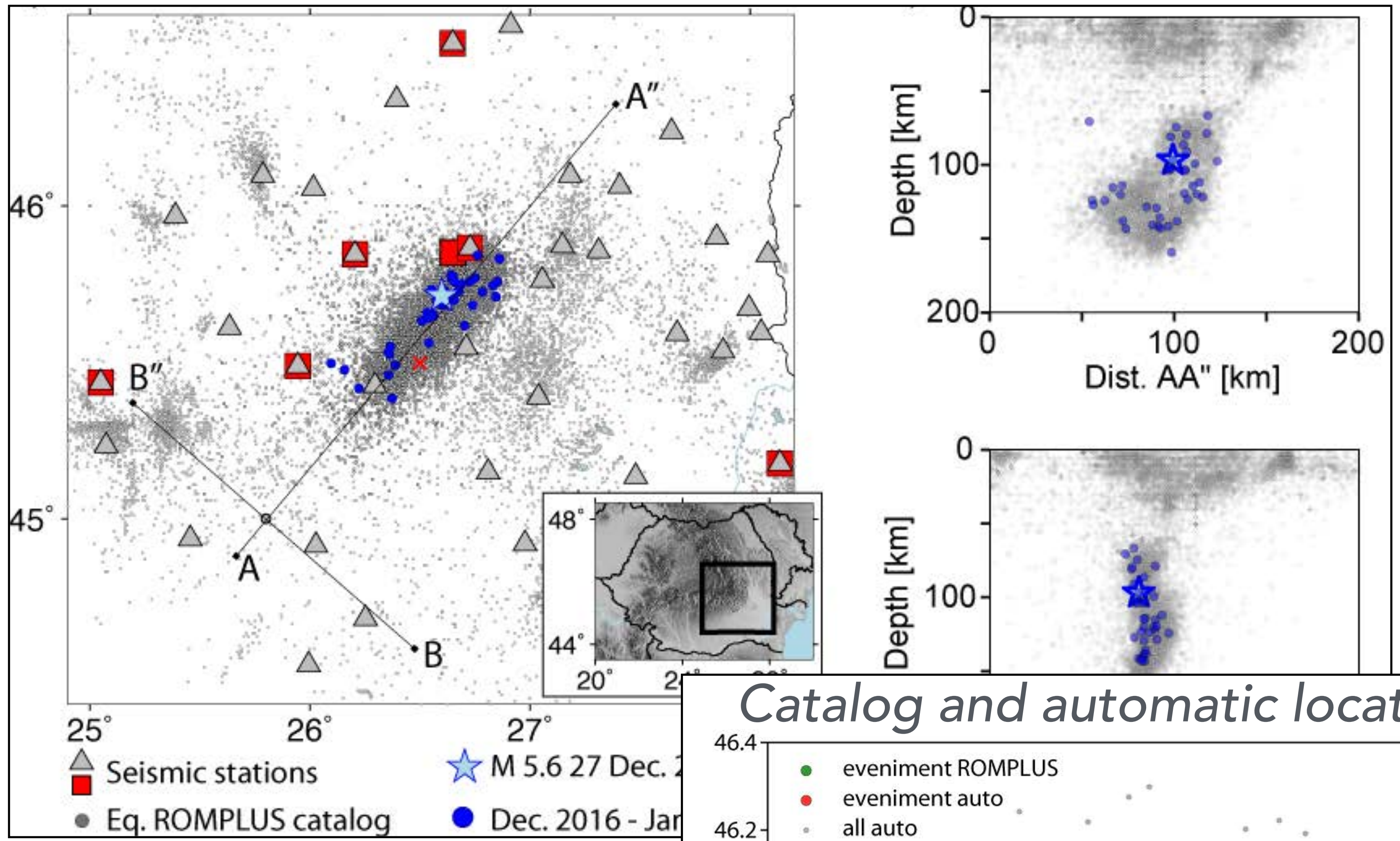
Demonstration of scalability and performance
Potential to create and analyse different parameter setup



100 station test - used as memory stress-test
Only finalised on large memory nodes
Provided as example test with program for deployment

Real-case dataset

Case of hazardous seismic activity in Europe
Testing performance and elapsed times

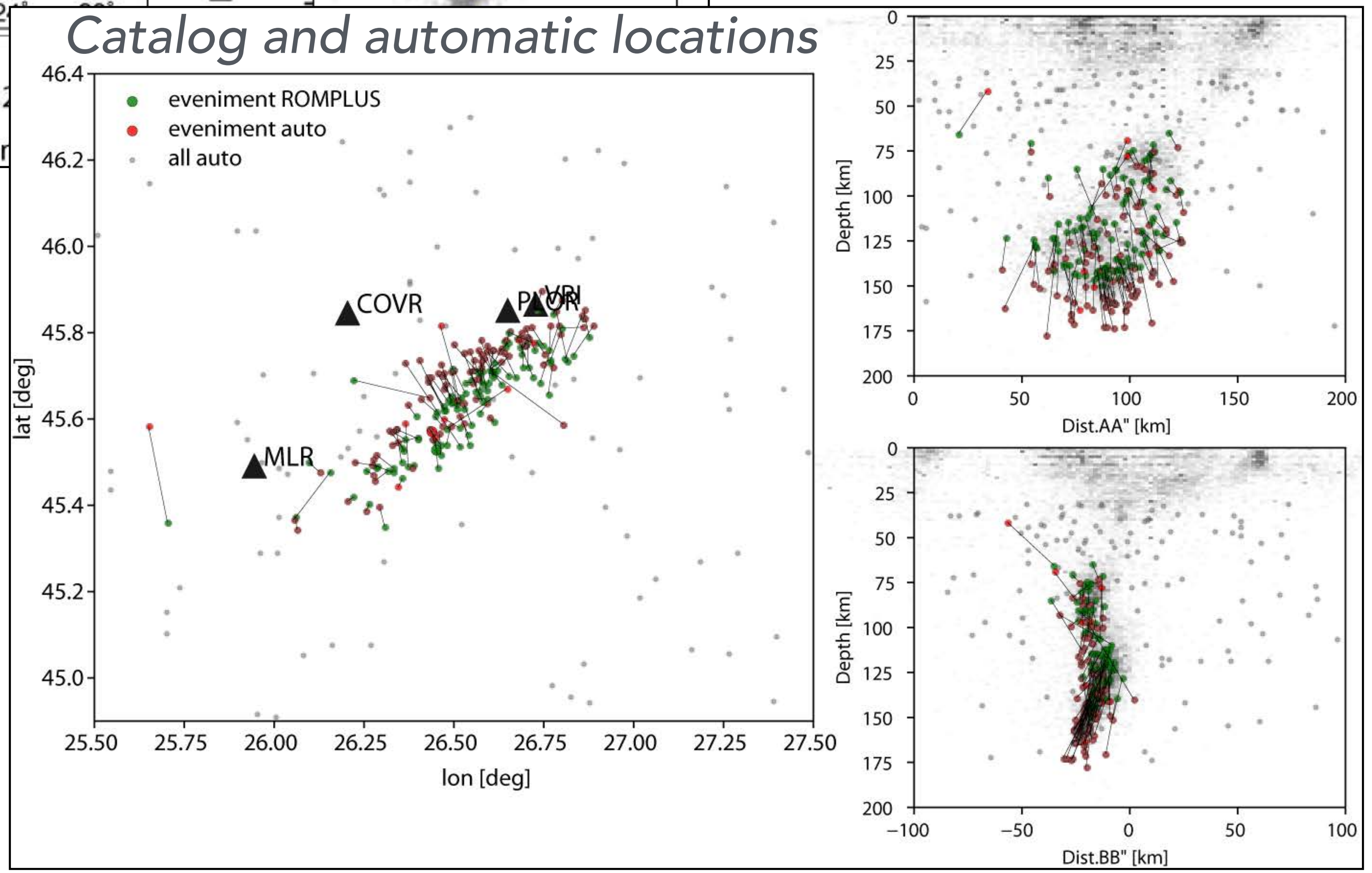


7.5 h for 1 month data & 22 stations - 50 nodes

Low increase in event detection numbers - methodological adjustments required

Data reprocessing is achievable

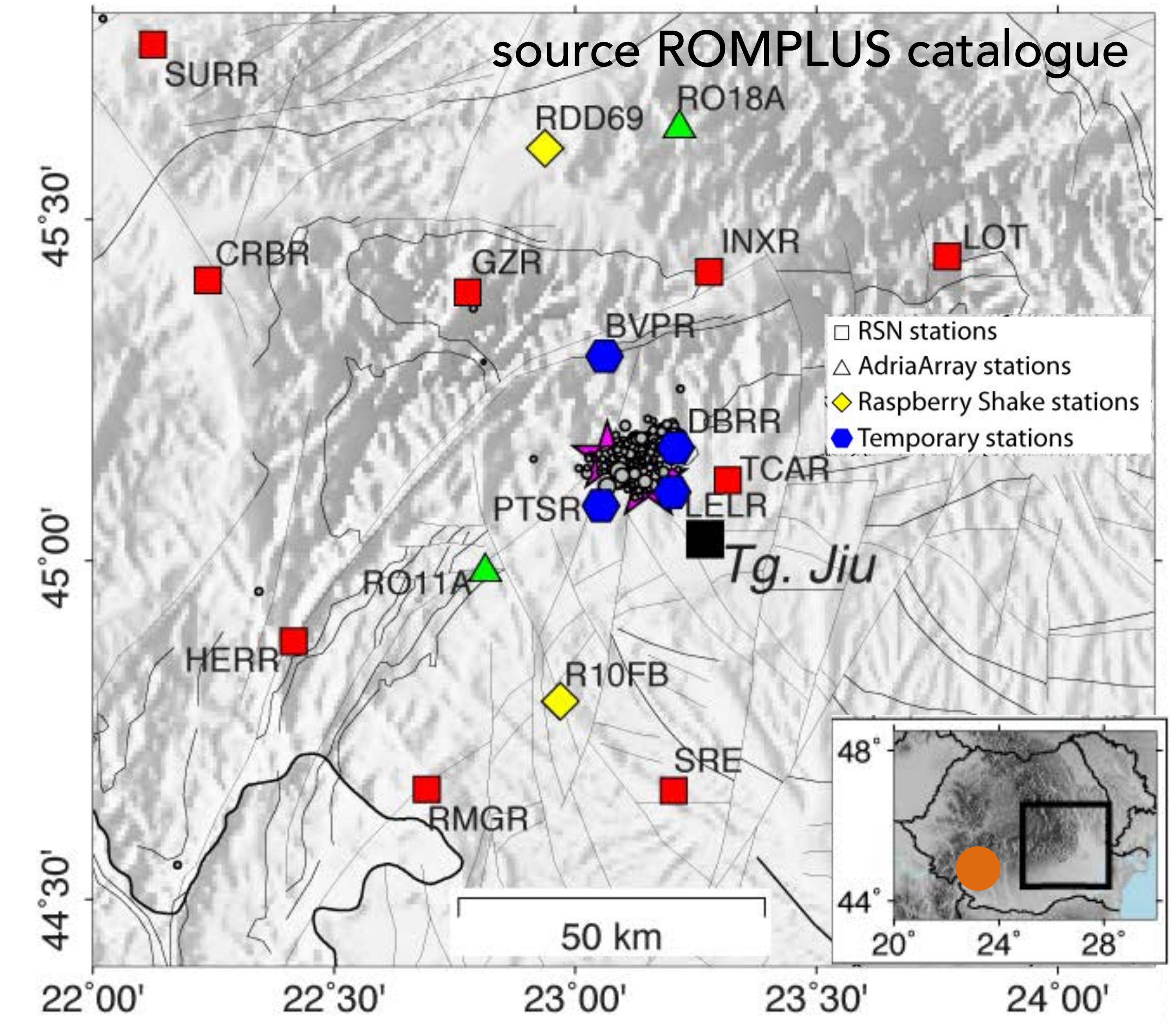
Possibility for multiple parameter testing
(e.g., velocity models, signal processing, …)

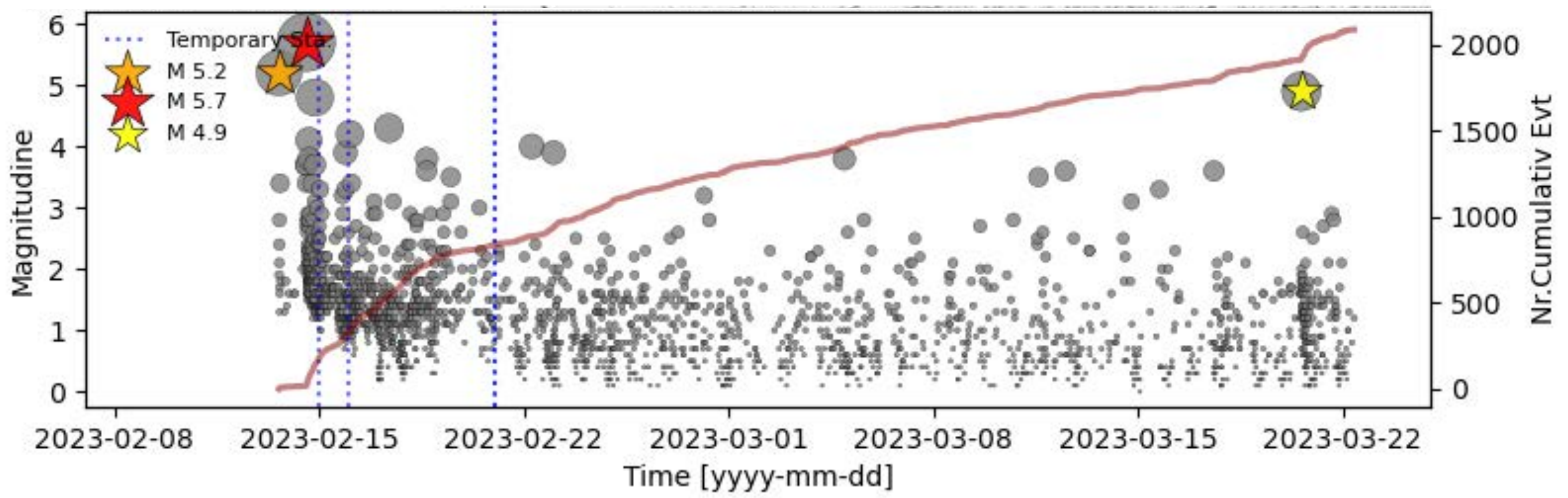


2.3. BackTrackBB with PyCOMPSs - Implications for earthquake monitoring

Rapid reply and emergency analysis in crises situation

Example of earthquake sequence in Romania (February - March 2023)





Gorj seismic sequence according to official data
Partial, manually revised results (work in progress)
~ 1500 events in > 1 month

Intense seismic sequence in an area with low seismicity

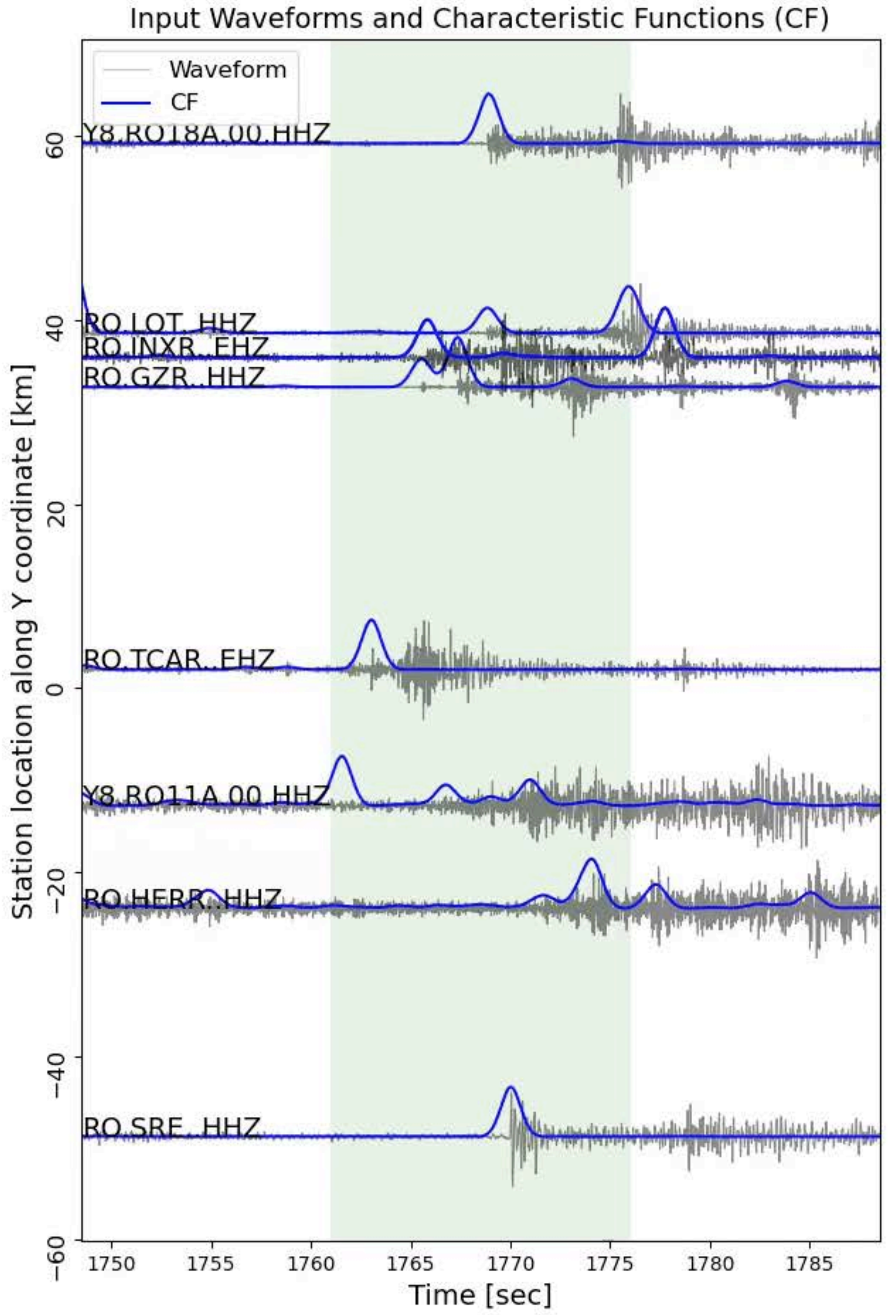
Big stress on standard technique analysis - large processing delays

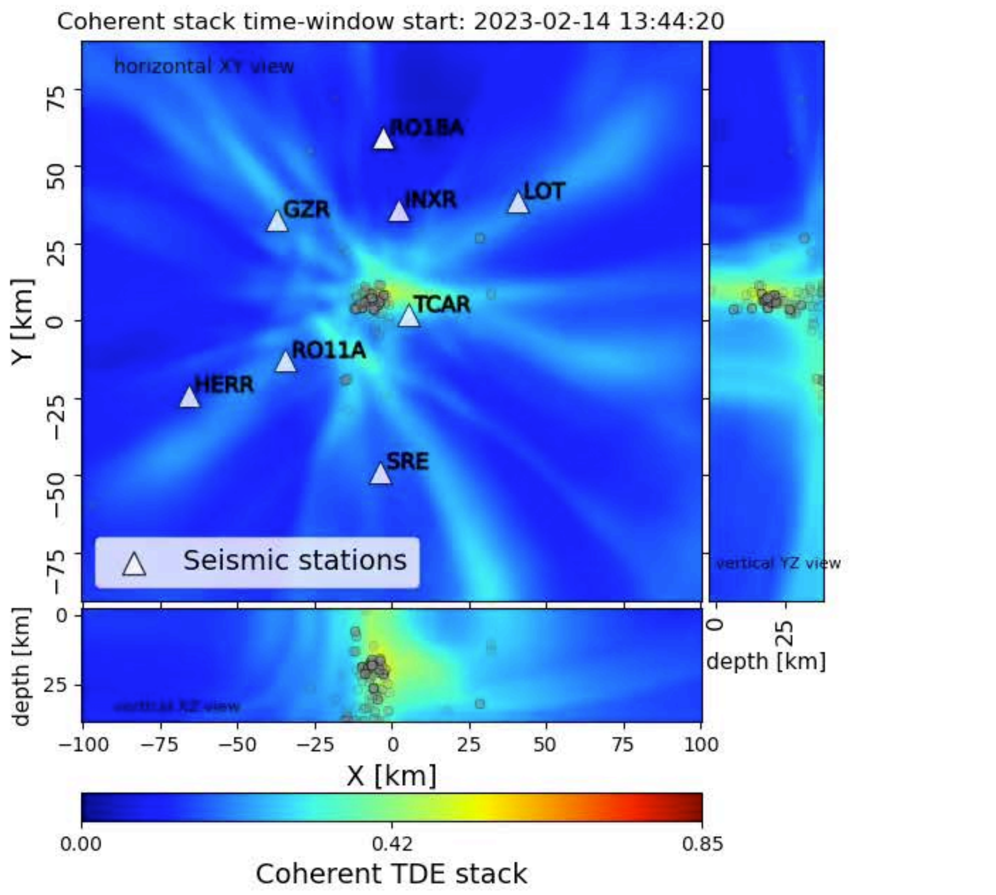
BackTrackBB with PyCOMPSs analysis of 11 February - 21 March 2023 data:
- 8 stations & 9 nodes - 4.68 h
- 12 stations & 9 nodes - 5.20 h

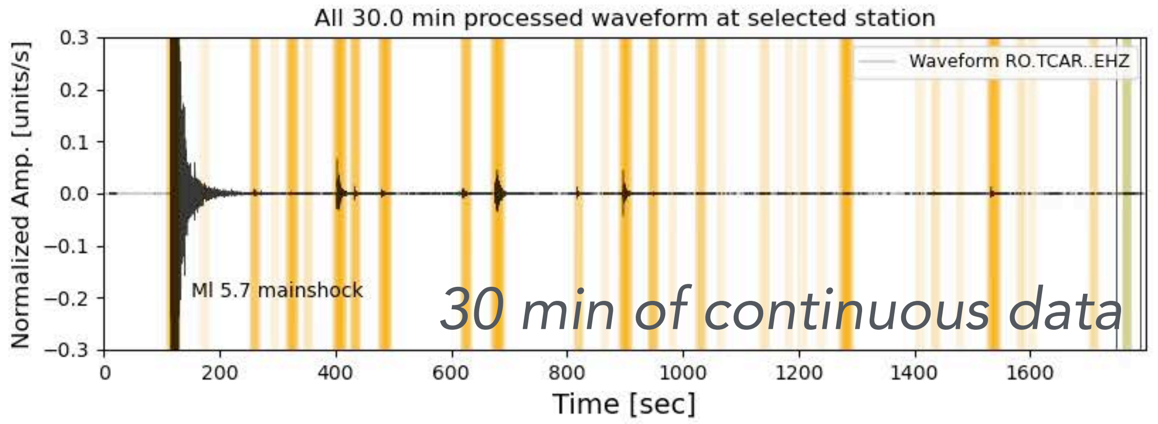
Can perform multiple tests
Provide information for emergency response actions

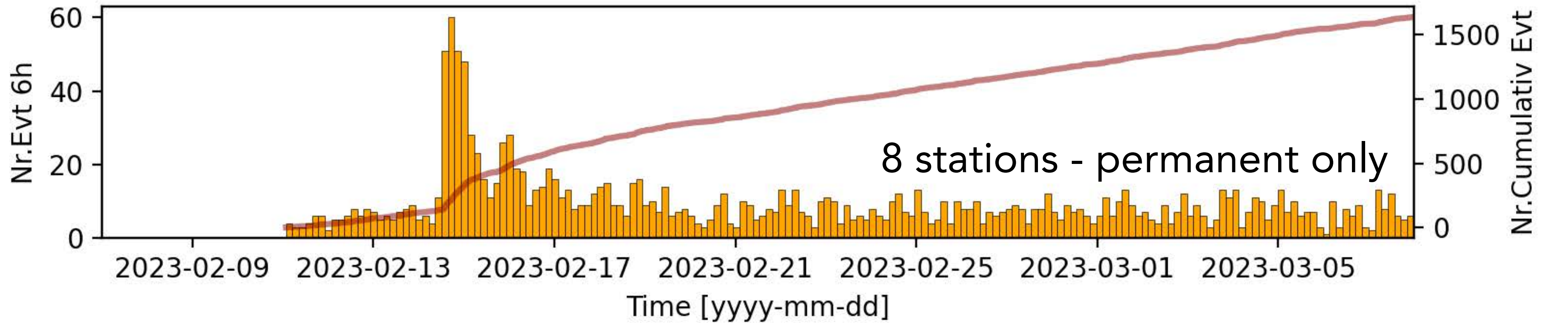
BackTrackBB with PyCOMPSs can be used for emergency monitoring

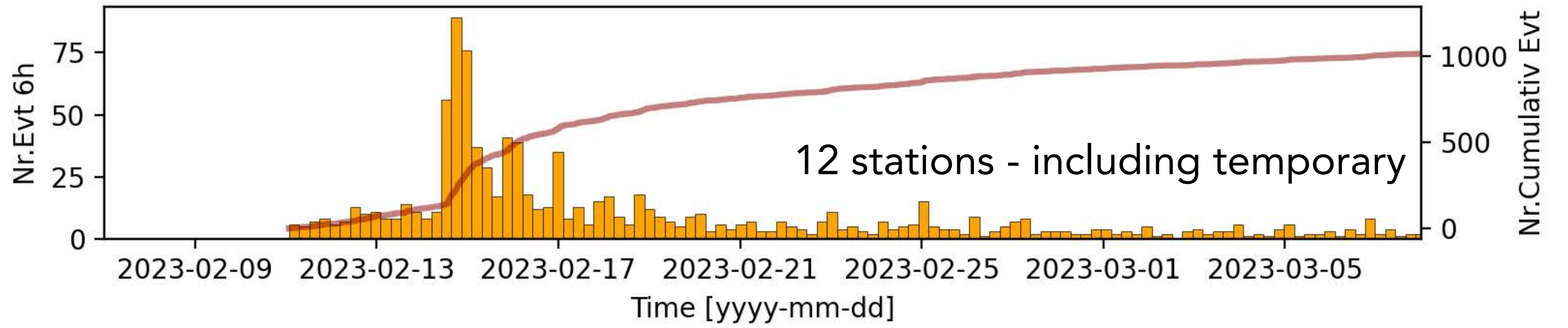
Example of BackTrackBB analysis











3. Conclusions

Big Data seismology require efficient methods to analyse and extract information from the datasets

Combining new methods for earthquakes detection and location with efficient use of available computational resources is required

BackTrackBB with PyCOMPSs - a task-based parallelisation of earthquake detection and location for efficient and reproducible analysis of continuous seismic data

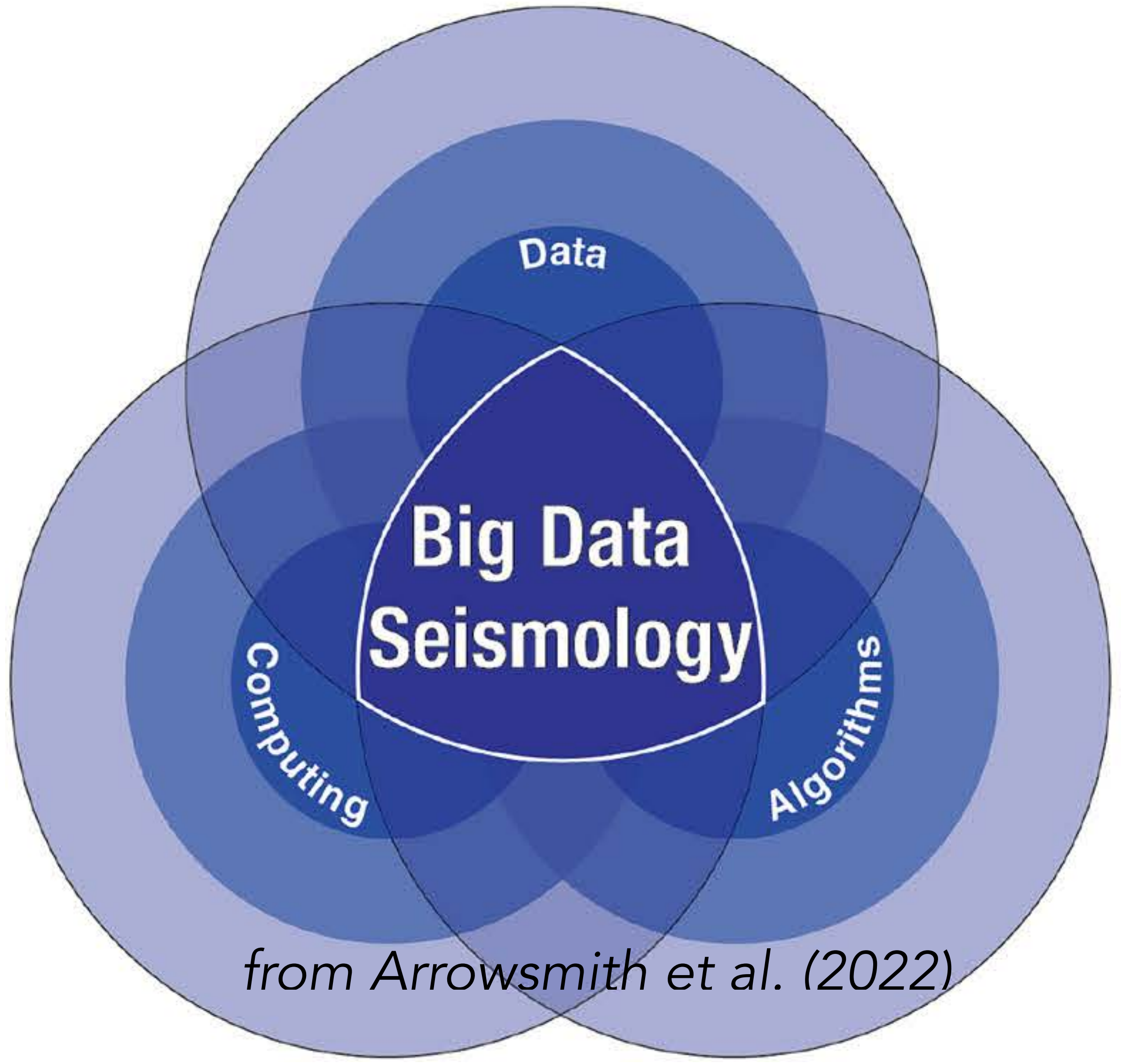
Provides perfect scalability confirmed by extensive testing

Current issue: memory-intensive problem

BackTrackBB with PyCOMPSs - can provide important input for crises situation if/when necessary computational (***on demand***) resources are available

Outstanding problem represent data access and deployment -> building connection between the seismological data centres and computational facilities

BackTrackBB with PyCOMPSs code availability - GitHub & WorkflowHub repositories



from Arrowsmith et al. (2022)