Supporting documentation for presentation EGU21-1514

Data-streaming workflow for seismic source location with PyCOMPSs parallel computational framework

N. Poiata[1,2], J. Conejero[3], R.M. Badia[3] , J.-P. Vilotte[1]

(1) Université de Paris, Institut de physique du globe de Paris, CNRS, F-75005 Paris, France.

(2) National Institute for Earth Physics, 12 Călugăreni, Măgurele, 077125 Ilfov, Romania.

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

1. The main goal

Developing a scalable parallelization of an automatic full waveform, network-based coherency method for seismic source detection and location that would allow an efficient and easily reproducible analysis of continuous seismic records from distributed seismic networks and benefit from efficient use of computational resources. The application will further lead to a common analysis framework for systematic exploration of the increasing seismic observation data volumes, important for improving seismic monitoring systems and extracting detailed and accurately resolved seismic source catalogues.

Our development makes use of PyCOMPSs (Tejedor et al., 2017) task-based programming model for distributed computing and a python-based scheme for automatic detection and location of seismic sources – BackTrackBB (Poiata et al., 2016, 2018).

2. The backbone methods:

2.1. BackTrackBB seismic source detection and location scheme

Short overview

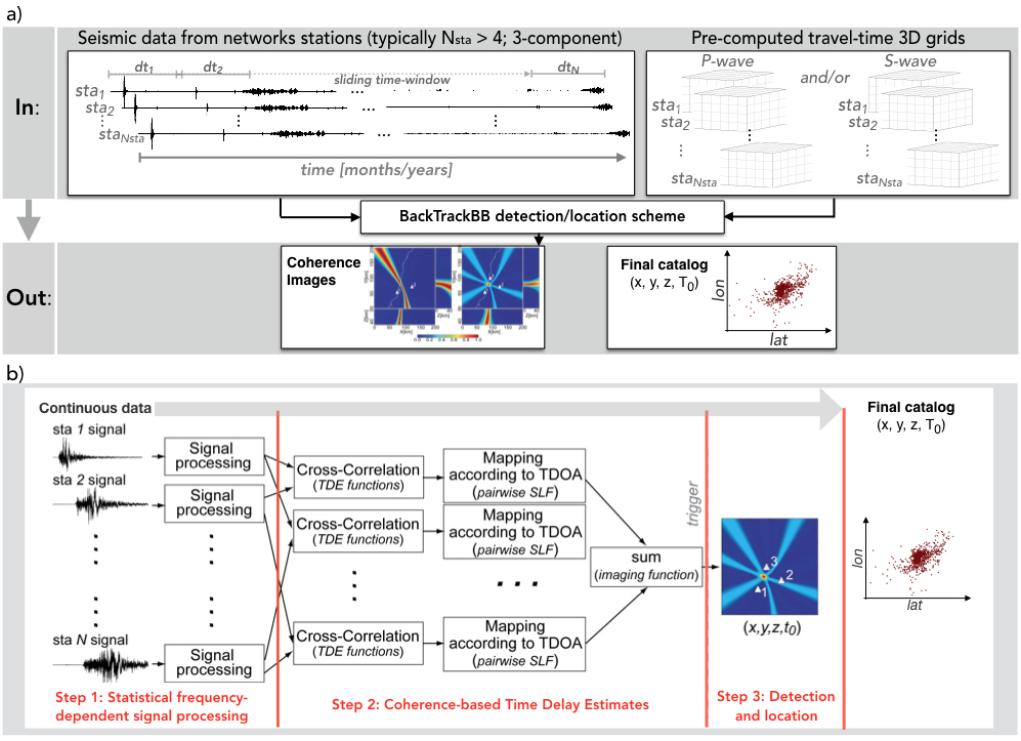


Figure 1. Schematic presentation of the BackTrackBB seismic source detection and location scheme. a) Illustration of the input - output workflow. b) Presentation of the methods' steps following Poiata et al. (2016).

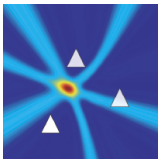
BackTrackBB is an automatic network-based coherence approach making use of frequency-dependent (statistical) characteristic functions built from the continuous seismic signal, such as higher order statistics (HOS) or energy envelopes, in order to detect and localize in space and time the seismic sources generating coherent transient signals recorded at network's stations. It proved to be efficient in extracting narrow-band small-magnitude events from the noisy records such as those corresponding to the small-magnitude earthquakes or low-frequency earthquakes that are often buried in complex, noisy signals (e.g., Poiata et al., 2018). The

workflow of the BackTrackBB method consists of signal processing, coherent imaging corresponding and detection and location steps to the detection and location. A schematic presentation of the processing steps is provided in Fig. 1.
The corresponding BackTrackBB program is implemented in Python with some of the modules written in C. It supports any seismological data formats that are included in the ObsPy framework. The continuous data processing is performed in a sliding window (of the size depending on the problem setup) and detection and location of the source is provided each time network coherence images exceed a pre-defined threshold.

References and documentation links:

- For details about the method and implementations please check the references below:
  - Poiata, N., Satriano, C., Vilotte, J.P., Bernard, P. and Obara, K., 2016. Multiband array detection and location of seismic sources recorded by dense seismic networks. *Geophys. J. Int.*, *205*(3):1548-1573. https://doi.org/10.1093/gji/ggw071.
  - Poiata, N., Vilotte, J.P., Bernard, P., Satriano, C. and Obara, K., 2018. Imaging different components of a tectonic tremor sequence in southwestern Japan using an automatic statistical detection and location method. *Geophys. J. Int*, *213*(3):2193-2213. https://doi.org/10.1093/gji/ggy070.

- BackTrackBB is an open source code available from:



  - Code - https://github.com/BackTrackBB/backtrackbb
  - Documentation - https://backtrackbb.readthedocs.io/en/latest/
  - ***Contacts***- poiata@ipgp.fr or natalia@infp.ro

2.2. PyCOMPSs framework

Short overview

PyCOMPSs (Tejedor et al., 2017) is a task-based programming model for python applications that relies in a powerful runtime able to extract dynamically the parallelism among tasks and executing them in distributed environments (e.g. HPC Clusters, Cloud infrastructures, etc.) transparently to the users.
A PyCOMPSs application is composed of tasks (methods annotated with decorators). The decorator identifies the tasks and includes the directionality of their parameters. The runtime builds at execution time a task-graph taking into account the task data-dependencies, and schedules and executes the tasks in the distributed infrastructure (HPCs, Clusters, Clouds and container managed infrastructures), taking care of the required data transfers.

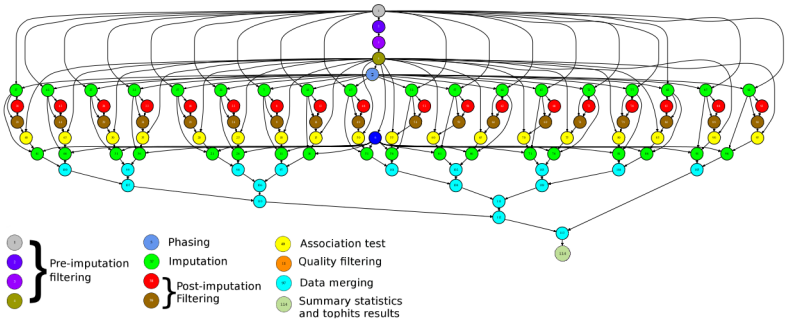


Figure 2. Task dependency graph example.

PyCOMPSs/COMPSs comes with a set of tools to improve the user experience, such as:

- Instrumentation with the Extrae library (also from BSC) to generate post-mortem tracefiles of the applications that can be visualized with the Paraver browser, offering a very powerful environment to analyze the performance of the applications.

- The COMPSs monitor: a web-based application that enables to monitor the execution of the applications. It visualizes the task-graph under execution, the available resources, the workload, etc.
- Jupyter notebooks are also supported with PyCOMPSs, enabling to run and parallelize Python notebooks (also available to run in HPCs).

References and documentation links:

- For details, including the technical description of PyCOMPSs, check the reference below:

Tejedor, E., Becerra, Y., Alomar, G., Queralt, A., Badia, R.M., Torres, J., Cortes, T. and Labarta, J., (2017). PyCOMPSs: Parallel computational workflows in Python. *Int. J. High Perform. Comput. Appl.*, *31*(1): 66-82. doi:10.1177/1094342015594678.

- PyCOMPSs is an open source project that is available from:
  - Code - https://pypi.org/project/pycompss/
  - Poster - https://pycompss-poster.readthedocs.io/en/latest/
  - Documentation - https://pycompss.readthedocs.io/en/stable/
  - Video advert - https://youtu.be/zoXUvIO001Q
  - Tutorial - https://www.youtube.com/watch?v=T9wHIcS7HAQ&list=PLeY7VuKJA5aVPYO50zfdc5FNrsu-KaIq-
  - ***Contacts*** - Javier Conajero - javier.conejero@bsc.es
    Rosa M. Badia – rosa.m.badia@bsc.es

More about the Workflow and Distributed Computing Group of Barcelona Supercomputing Center (BSC) on:
https://www.bsc.es/discover-bsc/organisation/scientific-structure/workflows-and-distributed-computing

- Examples of scientific applications benefiting from use of PyCOMPSs can be found in the following articles:
  - J. Conejero, C. Ramon-Cortes, K. Serradell and R. M. Badia, "Boosting Atmospheric Dust Forecast with PyCOMPSs," *2018 IEEE 14th International Conference on e-Science (e-Science)*, Amsterdam, 2018, pp. 464-474, doi: http://doi.org/10.1109/eScience.2018.00135.
  - Amela, Ramon and Ramon-Cortes, Cristian and Ejarque, Jorge and Conejero, Javier and Badia, Rosa M., "Enabling Python to Execute Efficiently in Heterogeneous Distributed Infrastructures with PyCOMPSs", Proceedings of the 7th Workshop on Python for High-Performance and Scientific Computing (PyHPC'17), Denver, CO, USA, 2017. https://doi.org/10.1145/3149869.3149870.
  - Javier Álvarez Cid-Fuentes, Pol Álvarez, Ramon Amela, Kuninori Ishii, Rafael K. Morizawa, Rosa M. Badia, "Efficient development of high performance data analytics in Python", Future Generation Computer Systems, Volume 111, 2020, Pages 570-581, ISSN 0167-739X, https://doi.org/10.1016/j.future.2019.09.051.
  - Andrio, P., Hospital, A., Conejero, J. et al. BioExcel Building Blocks, a software library for interoperable biomolecular simulation workflows. Sci Data 6, 169 (2019). https://doi.org/10.1038/s41597-019-0177-4.
  - Kahira, Albert Njoroge and Gomez, Leonardo Bautista and Conejero, Javier and Badia, Rosa M., "Accelerating Hyperparameter Optimisation with PyCOMPSs", ICPP 2019: Proceedings of the 48th International Conference on Parallel Processing: Workshops, Kyoto, Japan 2019 https://doi.org/10.1145/3339186.3339200

3. The outcome – BackTrackBB with PyCOMPs computational framework for seismic source locations

The main implementations for BackTrackBB with PyCOMPs version consisted of including the PyCOPSs task decorators to identified main functions of BackTrackBB code, such as those reading the input data, performing signal processing and detection/location steps. The original version proved to be already modular enough, thus, PyCOMPSs implementation did not require major code rearrangements. The above mentioned modifications led to a version that allows for a more efficient parallel task distribution, able to exploit multiple cluster nodes.

Task dependency graph in Fig. 3 illustrates this PyCOMPSs-based BackTrackBB implementation The main gain of the implementation is that it allows to achieve an efficient task-based parallel execution and monitoring of the seismic source detection and location from continuous data on HPC infrastructures. It also provides an easier deployment of BackTrackBB on different HPC environments.

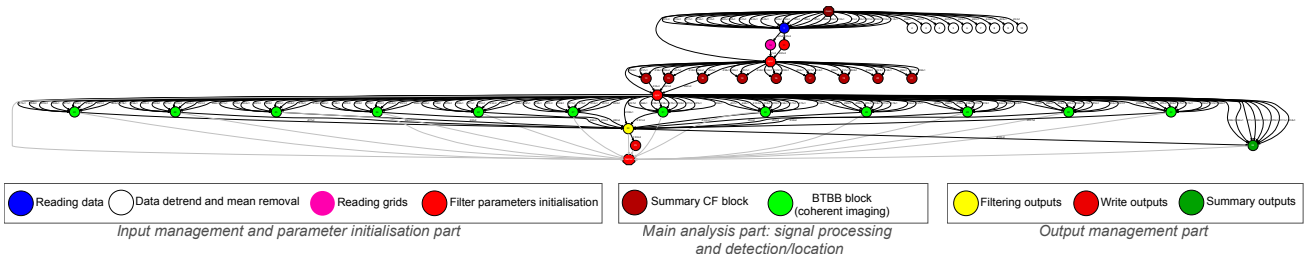


Figure 3. PyCOMPSs-based BackTrackBB task dependency graph illustrating different processing steps.

3.1. Performance tests and application examples

The scalability tests and memory usage analysis were performed, in the context of the European Centre Of Excellence (CoE) ChEESE for Exascale computing in solid earth sciences, on the MareNostrum4 (MN4) High-Performance computer of the Barcelona Supercomputing Centre, using a (large-scale) dataset of synthetic data. The executions have a master-worker architecture (sequential part of the code runs on a master node) and use a different number of nodes: 1, 2, 4, 8, 16 and 32 (48 CPU /node) for the test data set of 7 days long record and 100 stations. Results of the tests are illustrated in Fig. 4, showing comparison of the trace images for the runs. The run time for the corresponding jobs is depicted in Fig. 5.

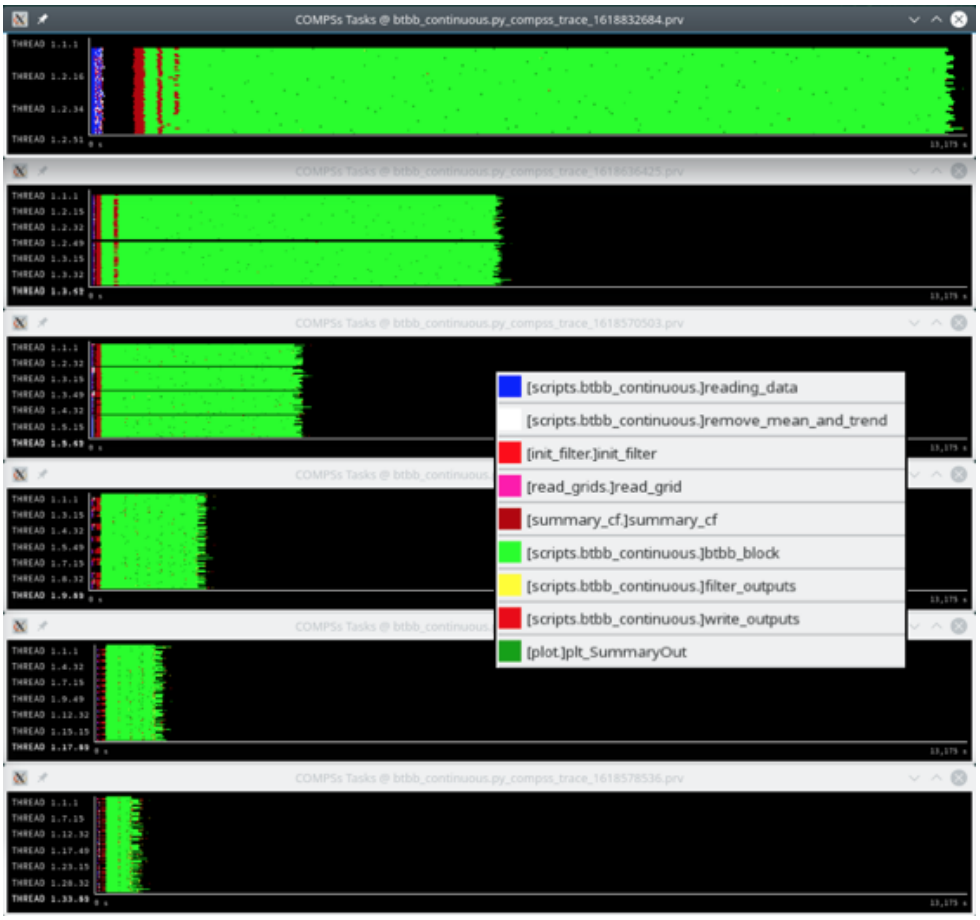


Figure 4. Trace image of BackTrackBB with PyCOMPSs run for 100 stations 7-days synthetic data records on MN4 using different number of worker nodes. Panels from top to bottom: 1, 2, 4, 8, 16 and 32 worker nodes. Horizontal extent shows time (13175 seconds, which is ~219 minutes).

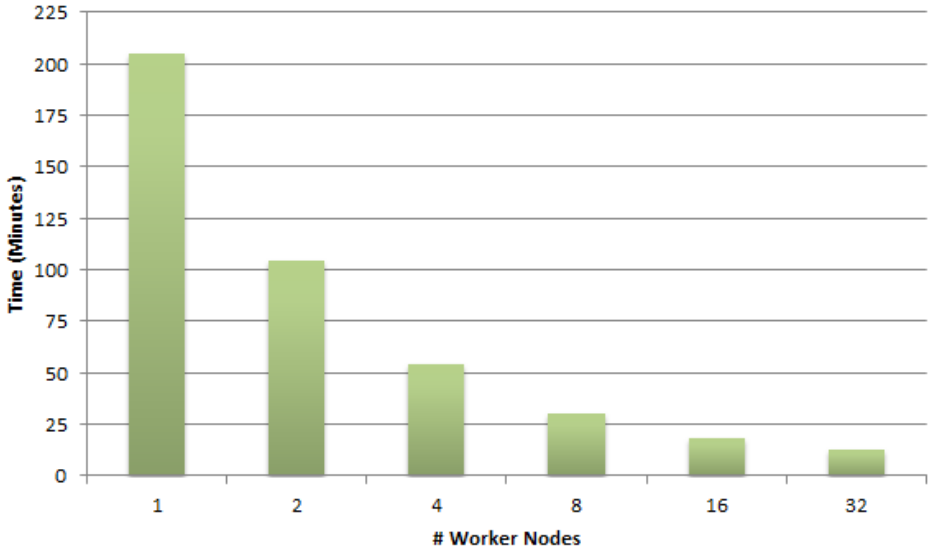


Figure 5. Performance of BackTrackBB with PyCOMPSs run for 100 stations 7-days synthetic data records on MN4 using different number of worker nodes.

- Synthetic dataset description

Synthetic toy model, used for testing the implementation, scalability and memory usage as well as the data-streaming workflow, is based on hypothetical scenario of 100 single-component seismic stations distributed on the surface over an area of 50 km x 50 km and recording 100 sps sampled seismic signal during 7-day time period. Seismic sources (generating simplified Gaussian pulse propagating at P-wave speed of 6 km/s) are located at depth range 10-20 km in a restrained volume.

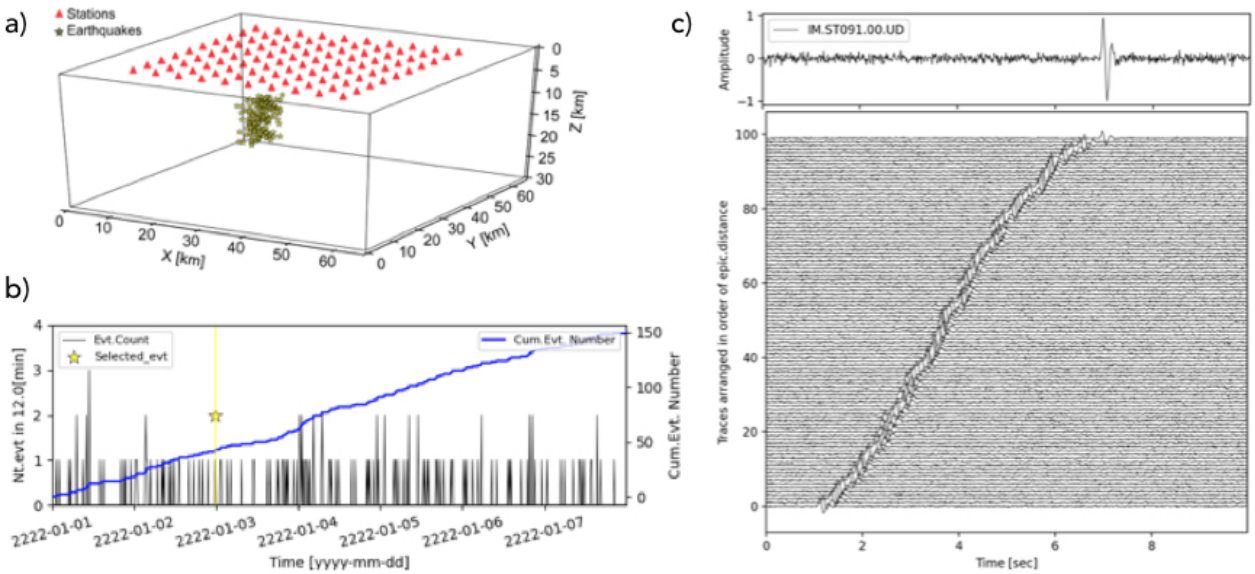


Figure 6. a) Space distribution of stations (red triangles) and earthquakes (yellow stars). b) Time distribution of earthquakes occurrence in the number of events in 12 hours (black curve) and cumulative number of events (blue curve). c) Example of the synthetic waveforms at single and 100 stations.

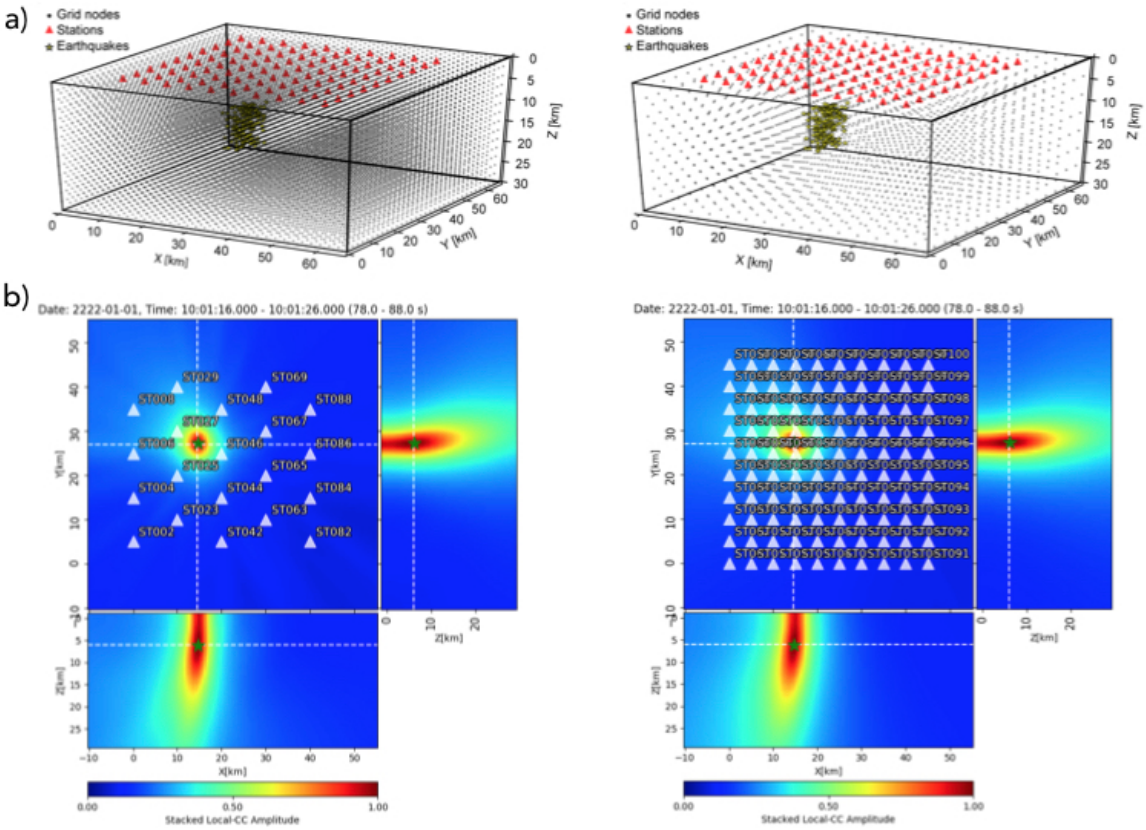


Figure 7. a) 100 station synthetic dataset setup and grids used for optimization testing: 0.5 km discretization (left) and 1km discretization (right). b) Illustration of an example source location testing 20 stations (left) and 100 stations (right) setup.