DeLIA: Dependability Library for Iterative Applications

Carla dos Santos Santana[1], Idalmis Milian Sardina[1],
Hervé Chauris[2], Claude Tadonki[2], Samuel X de Souza[1]

Motivation

- One of the primary purposes of an HPC implementation is scalability. However, with more nodes, the probability of failure is higher. Individual components (such as processors, disks, memories, and networks) have a high mean time before failure, but a system with numerous components may fail frequently. In a system with 10000 nodes, each with a mean time to failure 10^6 hours, the system will have only a mean time to failure 100 hours [4].
- **Therefore, dealing with faults is crucial for a scalable HPC application** [2].
- One of the leading research challenges in this area is developing fault tolerance techniques that **do not cause significant overheads** in normal execution conditions and do not hinder the basic HPC optimizations applied [2].
- We propose a **Dependability Library for Iterative Applications (DeLIA)** in C++ for HPC environments.
  - An fault tolerance library to HPC applications which have global synchronization between all processes, obtaining a globally consistent state after each iteration (see Figure 1).
  - The focus is to deal with failures which stops all execution and preemptive environments.

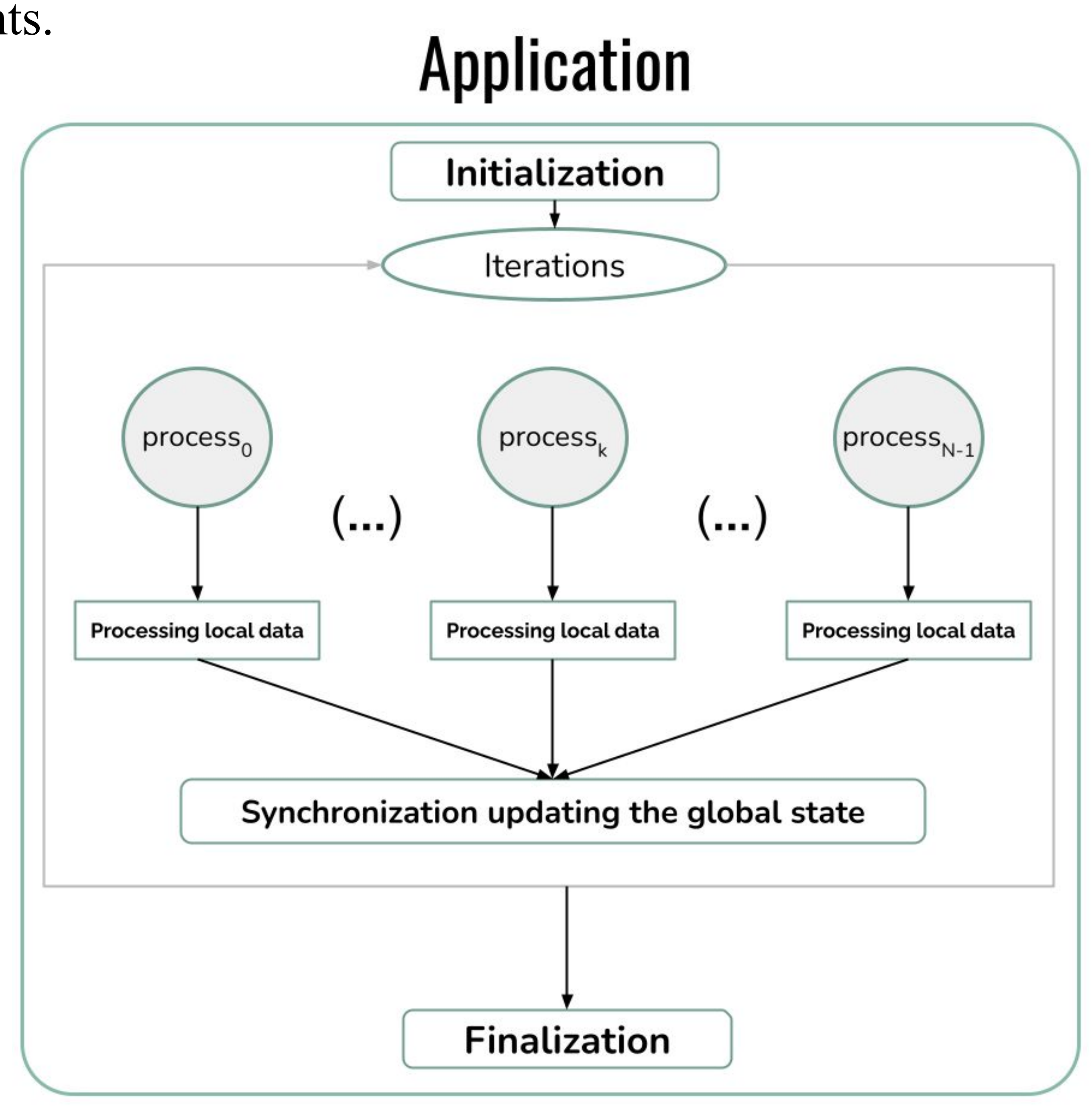

Figure 1: Application behavior

DeLIA features

- **Interruption Detection**
  - DeLIA interruption detection module includes *heartbeat monitoring* and *detection of termination signals*. If there is a possible failure, a trigger is sent to the nodes, and each one saves its local data.
    - **Heartbeat monitoring**: The observed nodes periodically send a message to the leader node. If the leader does not receive any heartbeat message in a specific time interval, it deduces that the observed node has failed [1].
    - **Detection of termination signals:** These signals notify a process to terminate at some moment. Some supercomputers and cloud systems use them to notify a job that it will be interrupted for some reason. The causes of the interruption can be, for example, that the application execution time exceeded the limit or because the environment is preemptive and reclaiming resources.
- **Checkpointing and rollback**
  - Checkpointing is the technique to save the data in a determined state. This can be used to resume the process later. The process of back-tracking the execution to such a saved state is named rollback [3].
  - DeLIA can save the global and local data of the application.
    - **Global Data**: data corresponding to the state of the whole application.
    - **Local Data**: data corresponding to the specific state of a single process

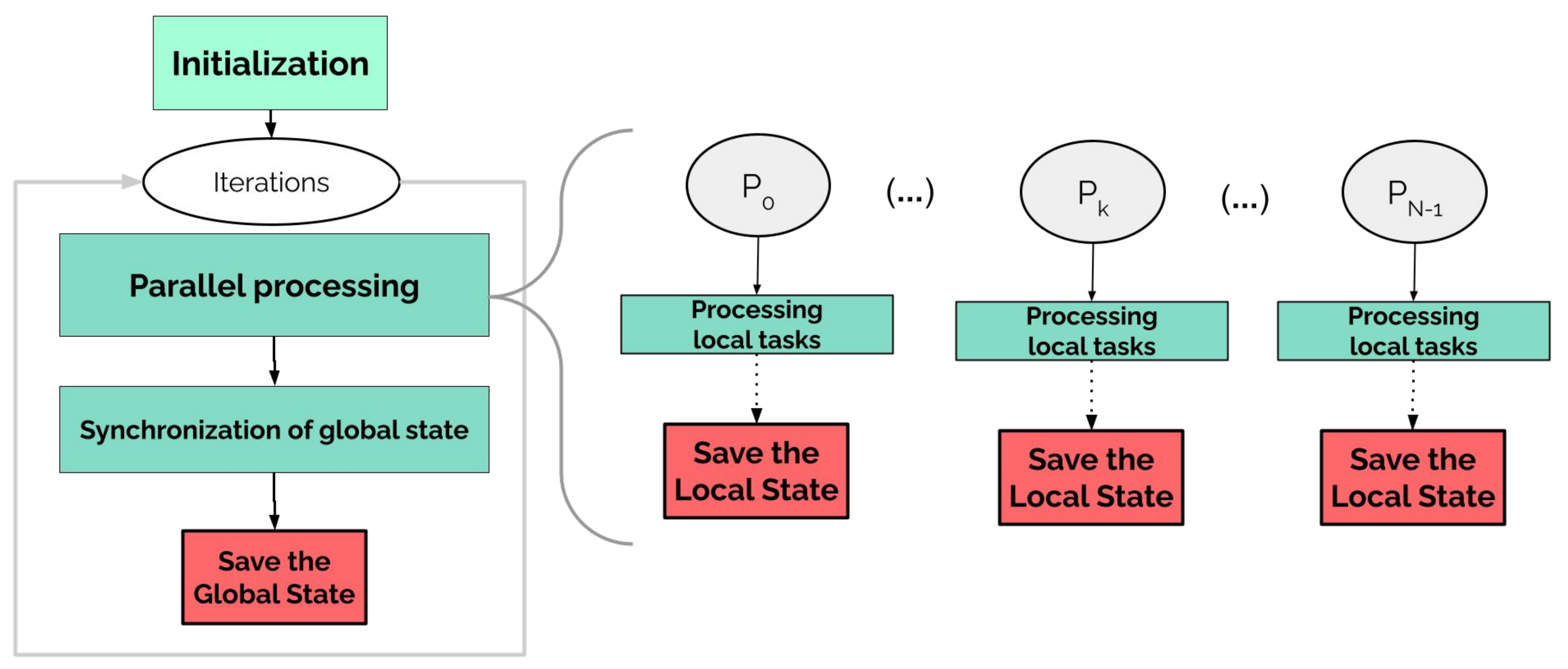

Figure 2: Checkpointing in the Application

- **Data Replication**
  - The data replication is between the process; each one sends its state data to the neighbor process; when the process makes the checkpointing, it will save its data and the neighbor's data.

Case Study: 3D Full Waveform Inversion

- The case study to be analyzed is the integration of DeLIA with the Full Waveform Inversion, an algorithm standard in geophysics.
- We tested with the interruption detection and checkpointing.
- This application is an ideal example to validate the library because it generates significant data, has built-in global data synchronization, and is used frequently in academia and the oil & gas industry.
- We executed FWI without and with DeLIA and investigated the overhead generated by DeLIA.

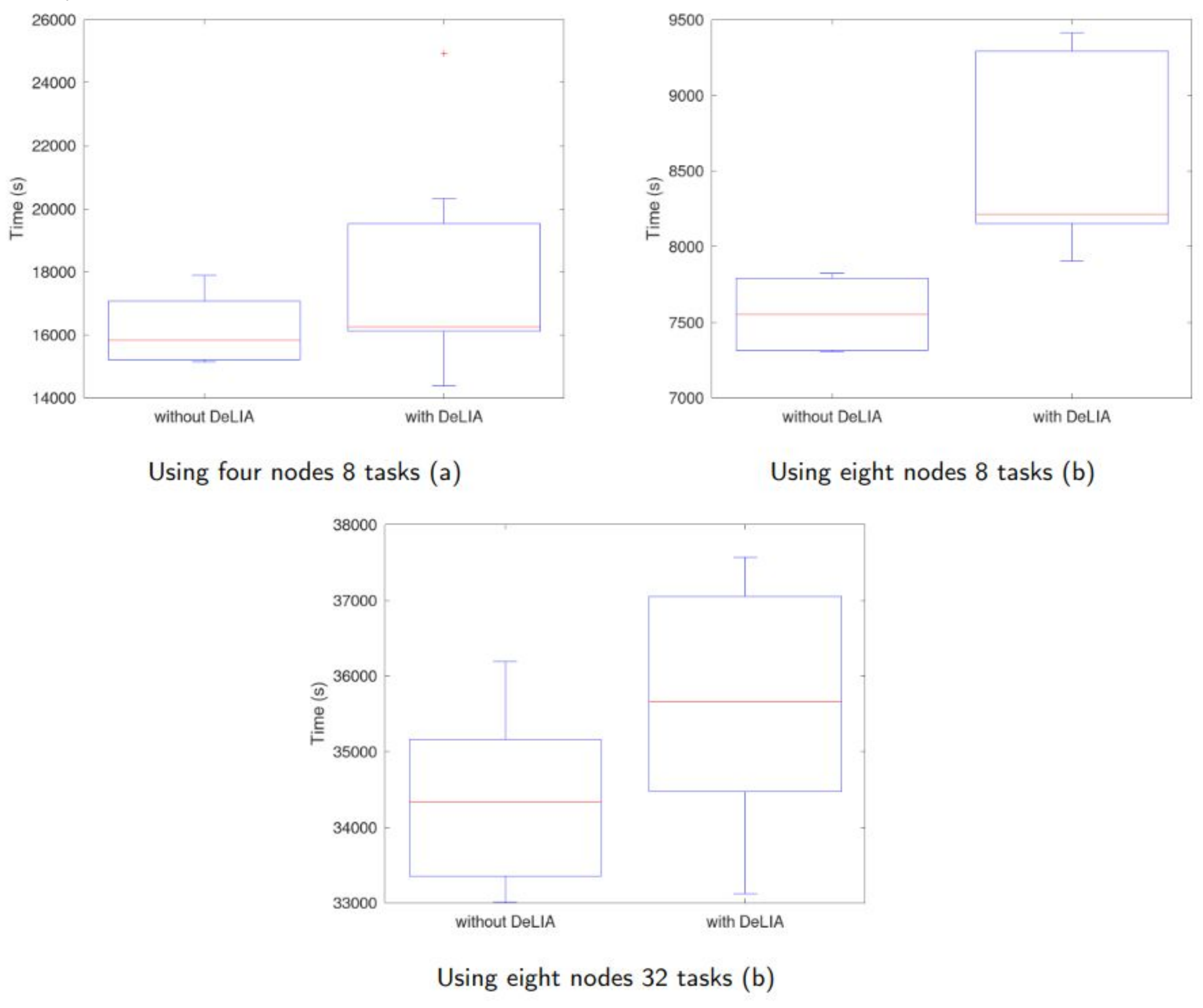

Figure 3: FWI without and with DeLIA

	4 nodes 8 tasks	8 nodes 8 tasks	8 nodes 32 tasks
DeLIA overhead	2.54%	8.8%	3.84%
Relative standard deviation without DeLIA	6.88%	3.37%	3.78%
Relative standard deviation with DeLIA	17.40%	7.20%	5.14%

Table 1: Analysis of DeLIA in FWI

DeLIA Usability

- DeLIA **provides an API** to programmers to use the features in their software; this allows them to call DeLIA functions, abstracting the library implementation.
- The main parameters for DeLIA are defined by the developer in a JSON file.
- The library and its documentation are available at https://lappsufrn.gitlab.io/delia.

Next steps

- Checkpoint with compressed data.
- The process shares encoded pieces of local data with more than one process. If one piece is lost, the other can reconstruct all data using the Reed-Solomon technique [5].
- Portability to other languages (e.g., python).

Acknowledgements

- The authors gratefully acknowledge support from Shell Brasil through the "Inversão Sísmica 4D" project at the Universidade Federal do Rio Grande do Norte (UFRN) and the strategic importance of the support given by ANP through the R&D levy regulation. The authors are grateful for the partnership with UFRN and Ecole Mines Paris- PSL. The authors also thank High-Performance Computing Center at UFRN (NPAD/UFRN).

1. Universidade Federal do Rio Grande do Norte, Brazil.
2. Mines Paris- PSL, France.

Contact: carla.ecomp@gmail.com

References

[1] Chetan S, A. Ranganathan, and R. Campbell. Towards fault tolerance pervasive computing. IEEE Technology and Society Magazine, 24(1):38–44, 2005.

[2] Herault T. and Robert Y. Fault-tolerance techniques for high-performance computing. Springer, 2015.

[3] Kalaiselvi S. and Rajaraman V.. A survey of checkpointing algorithms for parallel and distributed computers. Sadhana, 25(5):489–510, 2000.

[4] Reed, Daniel A., and Celso L. Mendes. "Reliability challenges in large systems." Future Generation Computer Systems 22.3 (2006): 293-302.

[5] Stephen B. Wicker and Vijay K. Bhargava. Reed-Solomon codes and their applications. John Wiley & Sons, 1999.