On the efficient use of HPC ressources: Disaggregation of the full waveform inversion algorithm

Dominik Schuster, Ludovic Métivier, Romain Brossier, Alizia Tarayoun

On the efficient use of HPC ressources: Disaggregation of the full waveform inversion algorithm

D. Schuster[1], L. Métivier[1,2], R. Brossier[1], A. Tarayoun[1]

[1]Univ. Grenoble Alpes, ISTerre, F-38058 Grenoble, France
[2]CNRS, Univ. Grenoble Alpes, LJK, F-38058 Grenoble, France

January 15, 2024

Main objectives

In this abstract we present an approach to disaggregate the full waveform inversion (FWI) algorithm and manage its workflow efficiently on a computing cluster. Controlling the workflow provides flexibility in terms of problem size with respect to computational resources, improves fault tolerance, allows the batch scheduler to increase the system's overall workload and lowers wait times for the user. This work is meant to be understood in the context of computationally heavy configurations with data from multiple sources.

New aspects covered

1. Disaggregation of the FWI scheme: the algorithm will be separated into three stages: computation of partial gradients, summation and optimization. The granularity can be adjusted by the user by specifying the number of partial gradients to be computed per job submitted to the batch scheduler.

2. Implementation of a workflow manager in an HPC environment, which will pilot the submission of jobs, the verification of correct program execution and data organization.

3. A cost to benefit analysis, which takes the computational overhead and the additional amount of data transferred to and from disk into account and brings them into relation with potential benefits in terms of improving cluster workload and fault tolerance.

On the efficient use of HPC ressources: Disaggregation of the full waveform inversion algorithm

Introduction

A common problem in modern HPC infrastructure is that the total system utilization, i.e. the number of core-hours used divided by the number of core-hours available during a certain time period, is only around 90% (You and Zhang, 2013; Rodrigo et al., 2018; Leonenkov and Zhumatiy, 2019). This can be due to jobs that are being canceled by the user, have requested inaccurate walltime or terminate early. Likewise, the submission of numerous large jobs hampers the batch scheduler's capability to efficiently fill up the queue. System underutilization means that processors are idling, which costs time, energy and money. Our approach to the FWI scheme alleviates these problems by disaggregating the computation of gradients from different sources, the summation of said gradients and the optimization step. This allows dividing one large job into several small jobs, which can be more easily backfilled by the batch scheduler. Simultaneously, tolerance to faulty nodes is increased. Instead of requiring a recomputation of the entire gradient, we only lose one partial gradient, which can be detected by the workflow manager and automatically resubmitted to the system without intervention by the user. Similar concepts have been proposed by other imaging software frameworks, such as SeisFlows (Modrak et al., 2018) or COFII (Washbourne et al., 2021).

Worfklow manager

The FWI computational scheme (Virieux et al., 2017) is based on the least-squares minimization

$$\min_m \frac{1}{2} \sum_{s=1}^{N_s} \|d_{\mathrm{cal,s}}[m] - d_{\mathrm{obs,s}}\|^2, \ \text{ st. } \ d_{cal,s}[m] = Ru_s[m], \ A(m)u_s = b_s, \tag{1}$$

with the subsurface parameters m, the number of shots N_s and the observed, calculated data $d_{obs,s}$, respectively $d_{cal,s}$ of the corresponding shot s. The restriction operator R is used to extract data from the wavefield $u_s[m]$, which is the solution of the wave equation denoted by the operator $A(m)$ and associated with the shot s. A model update can be expressed as iteration

$$m^{k+1} = m^k + \alpha^k \Delta m^k, \ \ \Delta m^k = -Q^k \nabla f(m^k), \tag{2}$$

where α^k is the linesearch step length. Starting from an initial guess m^0, the descent direction Δm_k is obtained by multiplying the gradient of the misfit function $\nabla f(m^k)$ with an approximation (e.g. limited-memory BFGS or truncated Newton method) of the inverse Hessian operator Q^k.

Typically, the main parallelization strategy applied is the distribution of gradient computation of individual shots via MPI. Gathering and summing up the contributions from all shots yields the total gradient, which is subsequently used for the optimization step. While this approach scales well up to a certain point, it has limitations in the exascale range, where fault tolerance starts playing an important role. Furthermore, the necessary number of processes is always a multiple of the number of shots, which can lead to massively parallel computation and represents a certain rigidity. These constraints are addressed by the newly introduced workflow manager. It is another layer on top of the FWI code, which pilots the submission of jobs to the cluster. The goal is to distribute the computation of partial gradients from shots into different jobs and thus rendering the computational scheme more flexible.

The script is written with Python and makes use of Cron, a tool for Unix-like systems to schedule periodic execution of commands at specified times or intervals. An overview of the workflow manager functionality is depicted in Figure 1. On the cluster frontend node, the user calls the Cron manager by specifying a case directory. It supervises active Cron tabs, which define the periodic invocation of the workflow manager. At the start of every execution, a file is read to retrieve information about the previous run, such as a list of queued jobs. If the computation has converged or failed (this can be communicated from the FWI code via a flag file) the Cron tab will be deleted and the script terminated. Otherwise the job scheduler is queried to determine the current state of jobs in order to decide the next action to take (see Algorithm 1). At the end of execution the current state of queued jobs will be written to file before exiting.

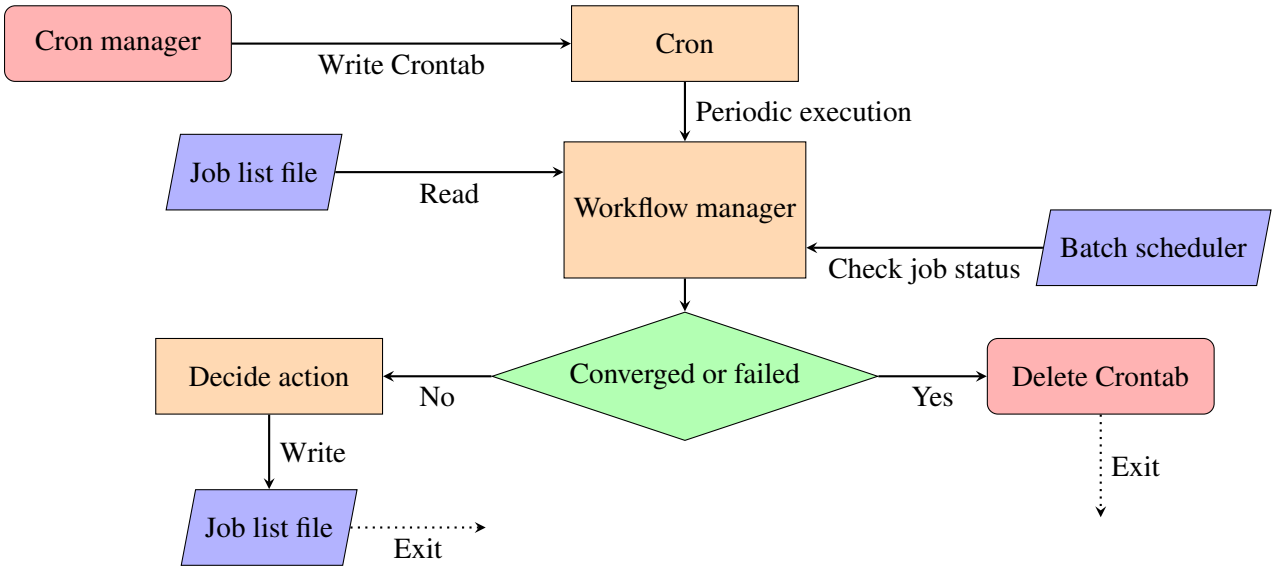

Figure 1 *Workflow manager flow chart*

An important aspect of this approach is that we assume no computation or input/ouput (IO) can be performed on the cluster frontend nodes and any such operation must be send to backend nodes via job scheduling. We tested our script on a cluster employing the OAR scheduler (https://oar.imag.fr/), but the concept is transferable to any other batch schedulers, such as SLURM. If Cron is not permitted on a computing cluster, another possibility would be to run the workflow manager on a backend node.

Algorithm 1 shows the process of deciding which action is to be taken by the workflow manager. At first execution or when an optimization job has terminated, the computation of gradients is submitted. If at least one gradient job has finished and no other summation job is currently queued, a summation job is submitted. Summation of partial gradients is performed continuously to minimize disk space usage. However, summations are prohibited from running simultaneously to avoid concurrent disk access. If all partial gradients have been computed and summed and no other optimization is currently queued, a new optimization job is submitted. In any other case, no action is required. Additionally, the workflow manager is verifying the successful job termination by reading job output files and confirm the presence of generated files.

Algorithm 1 Decide action

```
Bool A ← Gradient job queued?
Bool B ← Summation job queued?
Bool C ← Optimization job queued?
Bool D ← At least one gradient job has terminated?
Bool E ← Optimization job has terminated?
if (A & not B & not C & not D & E) then
    Action ← Submit gradient jobs
else if (not B & not C and D & not E) then
    Action ← Submit summation job
else if (not A & not B & not C & not E) then
    Action ← Submit optimization job
else
    Action ← No action required
end if
```

Cost to benefit analysis

While introducing a workflow manager potentially improves fault tolerance, reduces waiting times for the user and increases the average system workload, there are also drawbacks. The main disadvantage is

the increase in disk IO, since we need to write partial gradients (plus preconditioner or pseudo-hessians depending on the optimization method) and read them again during summation. We can define the granularity g of our disaggregated computation as the number of sources computed per job submitted to the scheduler. In principle this value can be in the range of computing one single source per job to all sources in one single job. However, it is good practice to fill up computing nodes, so reasonable values of granularity lie within one to few filled nodes. Using the l-BFGS optimizer, the disk IO needed per gradient computation in 3D and for single precision is

$$IO = \left((2l+3) + 2\frac{N_s}{g} + N_{sum} \right) \cdot N^3 \cdot N_p \cdot 4\,bytes, \tag{3}$$

where N is the average number of points per direction in the domain, N_p is the number of subsurface parameters stored in m, N_{sum} is the number of summations performed ($N_{sum} < N_s/g$) and l is the memory size of l-BFGS. For comparison, we write one gradient of size $N^3 \cdot N_p \cdot 4\,bytes$ to disk when not using the workflow manager. Figure 2 shows, that the amount of disk read and write can increase significantly for small values of g. On the other hand, high values of granularity (close to the number of shots) should also be avoided, since they offer little benefit but cause a considerable amount of IO overhead. This estimation does not take into account the complete disk IO of the FWI algorithm but only the part which is related to the computation of the gradient. The amount of excess core hours used will be negligible for large sized configurations.

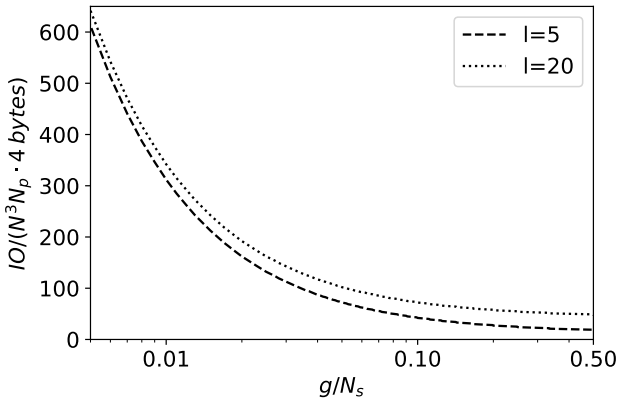

Figure 2 *Increase of disk IO related to the gradient computation of the disaggregated workflow w.r.t. the standard algorithm as function of granularity for $l = 5, 20$.*

It is difficult to quantify the benefit in the reduction of user wait time and increase of average system workload because they depend on the behavior of the users on a specific cluster, which can vary over time. Figure 3 illustrates an example how the disaggregated workflow contributes to filling up the job queue and minimizing the idle time of computational nodes. Without workflow manager a faulty node means loosing the whole gradient computation, wasting core hours since the other nodes continue running and requires manual restart. Following our approach, this loss can be reduced by a factor of N_s/G and the unsuccessful computation can be detected and restarted without user intervention.

Conclusions

Our work introduces a novel approach to enhance the efficiency of the FWI algorithm in HPC clusters. By disaggregating the FWI scheme into three distinct stages and implementing a workflow manager, we address key challenges associated with large-scale computations. The user-adjustable granularity allows flexible task distribution across the cluster, making it easier to subdivide large jobs and streamline batch scheduler backfilling. Our workflow manager ensures systematic control, overseeing job submission, verifying execution, and organizing data.

While acknowledging increased disk input/output (IO) as a trade-off, our approach offers benefits in fault tolerance and increased system workload. By optimizing HPC infrastructure, we contribute to more efficient seismic data processing and hold promise for broader applications in computational geophysics, particularly in complex subsurface imaging.

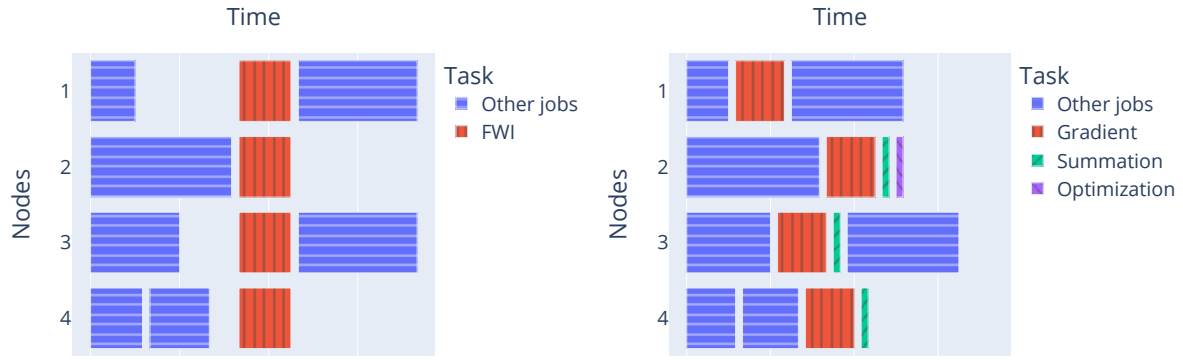

Figure 3 *Job scheduling with standard (left) and disaggregated workflow (right).*

The next step in our project is the actual implementation of this approach on a 3D field data FWI experiment on a national cluster. Another perspective is also the integration in this job scheduling task strategy of the uncertainty estimation through Ensemble Kalman Transform Filters developed in Thurin et al. (2019) and Hoffmann et al. (2023). In the latter framework, an ensemble of FWI have to be ran in parallel, with simple linear algebra communication tasks between FWI runs, which simply represent an additional layer in the current workflow manager architecture.

Acknowledgments

This study was partially funded by the SEISCOPE consortium (`http://seiscope2.osug.fr`), sponsored by AKERBP, CGG, DUG, EXXONMOBIL, GEOLINKS, JGI, PETROBRAS, SHELL, SINOPEC and TOTALENERGIES. This study was granted access to the HPC resources provided by the GRICAD infrastructure (https://gricad.univ-grenoble-alpes.fr), Cray Marketing Partner Network (`https://partners.cray.com`) and IDRIS/TGCC/CINES under the allocation 046091 made by GENCI.

References

Hoffmann, A., Brossier, R., Métivier, L. and Tarayoun, A. [2023] Uncertainty quantification for 3D time-domain full waveform inversion with ensemble Kalman filters: application to the Valhall OBC dataset. *Geophysical Journal International*, **in revision**.

Leonenkov, S. and Zhumatiy, S. [2019] Supercomputer Efficiency: Complex Approach Inspired by Lomonosov-2 History Evaluation. In: Voevodin, V. and Sobolev, S. (Eds.) *Supercomputing*. Springer International Publishing, Cham, 631–640.

Modrak, R.T., Borisov, D., Lefebvre, M. and Tromp, J. [2018] SeisFlows—Flexible waveform inversion software. *Computers & Geosciences*, **115**, 88–95.

Rodrigo, G.P., Östberg, P.O., Elmroth, E., Antypas, K., Gerber, R. and Ramakrishnan, L. [2018] Towards understanding HPC users and systems: A NERSC case study. *Journal of Parallel and Distributed Computing*, **111**, 206–221.

Thurin, J., Brossier, R. and Métivier, L. [2019] Ensemble-based uncertainty estimation in Full Waveform Inversion. *Geophysical Journal International*, **219**(3), 1613–1635.

Virieux, J., Asnaashari, A., Brossier, R., Métivier, L., Ribodetti, A. and Zhou, W. [2017] *6. An introduction to full waveform inversion*. R1–1–R1–40.

Washbourne, J., Kaplan, S., Merino, M., Albertin, U., Sekar, A., Manuel, C., Mishra, S., Chenette, M. and Loddoch, A. [2021] *Chevron optimization framework for imaging and inversion (COFII) – An open source and cloud friendly Julia language framework for seismic modeling and inversion*. 792–796.

You, H. and Zhang, H. [2013] Comprehensive Workload Analysis and Modeling of a Petascale Supercomputer. In: Cirne, W., Desai, N., Frachtenberg, E. and Schwiegelshohn, U. (Eds.) *Job Scheduling Strategies for Parallel Processing*. Springer Berlin Heidelberg, Berlin, Heidelberg, 253–271.