Nnodes: a workflow manager for full waveform inversion in large scale HPC clusters

Congyue Cui[1], Lucas Sawade[1], Jeroen Tromp[1]

[1] *Princeton University*

Abstract

We present a workflow manager that addresses many of the pain points we have experienced when running full-waveform inversion (FWI) on large-scale HPC clusters. FWI can be computationally expensive and complicated in structure, depending on the application. Our workflow manager makes it easy to control the progress and reduces repetitive work. The design of our workflow manager focuses on three key points. First, the workflow definition is flexible to make sure that users can easily make adjustments and experiment with new ideas. Second, the progress management is adaptable so that an interrupted workflow can be stopped and resumed at any point, and it is possible to rewind to a previous state if any parameter does not turn out to perform well; running a partial workflow or merging multiple workflows are also supported. Third, parallel execution of multiple MPI functions is naturally supported. An MPI operation from any part of the workflow will be managed by an MPI executor and users do not need to manually adjust the workflow to make the most of the cluster resources. We have successfully applied the workflow manager to both regional structural inversion and global CMT inversion. Migrating existing tools to this workflow manager was seamless in our case. We hope this is a useful tool for FWI researchers as well as the general HPC cluster users.

Design

Nnodes is a workflow manager that makes your life easier when defining or running complicated jobs either in your local computer or in a large-scale cluster. The basic execution unit of nnodes is called a "node", which consists of 4 elements: task, directory, properties and child nodes.

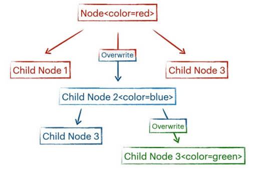
- **Task:** can be a Shell command or Python function. If it is a function, the "node" object will be passed to it during execution so that it can read its properties.
- **Directory**: working directory of the node, task output will by default be saved to its directory. The node object also come with various file system utilities like copy, move, read, write, etc.
- **Properties**: additional information that can be read by the task. The properties of a node will be propagated to all its child nodes, unless overwritten.
- **Child nodes**: nodes that will be executed after the task of current node is complete. A node can be configured to execute its child nodes either sequentially or concurrently.

While designed for the use case of full waveform inversion, nnodes is a general-purpose workflow manager with no pre-defined workflow. Some key advantages of nnodes include

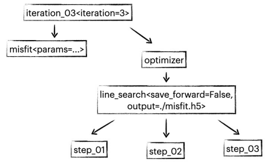
- **Flexibility**: compared to some all-in-one FWI packages, there is no restriction on where you can put extra steps, and everything is automatically backed up.
- **Simplicity**: unlike most professional workflow managers, which have very steep learning curves, nnodes provides a unified interface for all tasks. Migrating existing workflows to nnodes is seamless in most cases.
- **Scalability**: nnodes introduces a concurrent structure that behaves consistently in all computing platforms. The same workflow can either run on a local computer or in a large cluster and nnodes will automatically scale for best performance.

Property / directory management

Nnodes introduces a hierarchical structure to manage properties and directories. This is inspired by the styling of HTML document, in which properties will by default propagate to the child nodes and can be overwritten if necessary.



FWI usually involves many parameters, like iteration number, simulation mode, current model, etc. By introducing a hierarchical structure like this, we no longer need to pass the same parameter repeatedly. Below is an example a partial FWI workflow


iteration_03<iteration=3>
misfit<params=...>
optimizer
line_search<save_forward=False, output=./misfit.h5>
step_01
step_02
step_03

The code to define the structure above is appended below. The same mechanism applies to directories as well, the function step_03, for instance, is run under directory iter_03/step_03 because cwd parameter is passed when defining the iteration and search step.

```
def main(node):
    node.add(iterate, cwd='iter_03', iteration=3)

def iterate(node):
    node.add(solver, cwd='solver')
    node.add(optimizer)

def solver(node):
    node.add(specfem)
    node.add(misfit)

    if not node.misfit_only:
        node.add(adjoint)
```

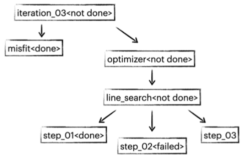
```
def optimizer(node):
    node.add(line_search)

def line_search(node):
    node.add(step, step=0, cwd='step_00')
    node.add(step, step=1, cwd='step_01')
    node.add(step, step=2, cwd='step_02')

def step(node):
    node.add(solver, misfit_only=True)
```

Progress control

The ability to backup and restore progress is one of the most important requirements for FWI as well as many other applications. In nnodes, a node has four possible states: pending, running, done, terminated, or failed. The following example shows the behavior of nnodes when a node fails.

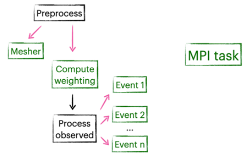


When any node is terminated or fails, the workflow will exit. In this example, the line search step_02 fails and the workflow will exit. By default, if this is the first time that step_02 fails, the current job will automatically re-submit. You can also choose to re-submit manually. When the workflow runs again, it will directly go to step_02 because all previous steps have been marked as done.

It is also possible to manually tweak a node state. If, for example, you wish to run misfit calculation again, you can simply set the state of misfit to pending, and when running the workflow again, the workflow will redo the misfit calculation.

MPI execution

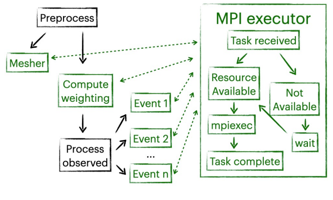
The following example shows a use case for the parallel execution of multiple MPI tasks. In the preprocessing step, the mesher is called to generate the mesh for the current model. Meanwhile, there are still some remaining processors to process the data. After weightings are computed, we begin processing observed data for n events. There are a total of n+2 MPI tasks, mesher can run simultaneously with all other tasks, while processing observed data needs the result of weighting computation.



This heterogenous workflow cannot be executed efficiently in most workflow managers because of the following reasons:

- Heterogeneous workflow may not be supported at all.
- We need to dynamically call new MPI tasks once an old task is complete.
- The number of events can be larger than the maximum number of processes that can be spawned.

Nnodes solves the problem with a concurrent structure, shown in the figure below. Background tasks like MPI calls or subprocesses are automatically grouped to a single executor while the main process remains single-threaded. This allows users to spawn thousands of background tasks (as long as the cluster allows) while not having to worry about compatibility or stability issues cased by multiprocessing. It also scales well, there is no need to manually adjust the workflow in order to make the most of the resources of a cluster.



Application to source inversion

We have successfully migrated our current source inversion workflow from a set of submission scripts for different job systems to nnodes with ease. We designed a(n) (iterative) workflow for a single event which can be run concurrently for many events by nnodes' design. Hence, we can also simply rerun a single event with a different set of parameters without a problem.

In the development of the source inversion workflow, the automatic backup-on-fail feature stood out in particular. In case a bug was introduced in a small piece of code somewhere in the workflow, one can fix the bug and start the workflow exactly where one left off, without the loss of "computational expense".

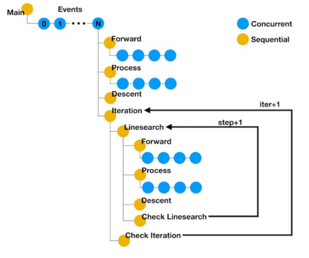
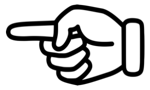


Illustration of a source inversion

Application to structural inversion

We have successfully run more than 50 iterations of regional full waveform inversion using nnodes. Parameters / directory management and MPI execution prove to be very useful to this application. The progress control has been especially helpful, because the 50 iterations are split into ~15 jobs and each job is automatically submitted once its previous job reaches the allocated walltime. We have also been able to fork from certain iterations with different parameters. More details in Poster #32.



Cluster support

Nnodes has no strong connection with any specific job system, when running MPI tasks it just serves as a wrapper for commands like mpiexec, srun, jsrun, etc. So, it is easy to add a new cluster configuration with the configuration file. Nnodes has built-in support for:

- Slurm
- LSF
- Local computer with multiprocessing

Learn more (GitHub documentation)

RESEARCH POSTER PRESENTATION DESIGN © 2019
www.PosterPresentations.com