The Journal of Open Source Software

distribute: Easy to use Resource Manager for Distributed Computing without Assumptions

Brooks Karlik [1*¶] **and Aditya Nair** [1*]

1 University of Nevada, Reno ¶ Corresponding author * These authors contributed equally.

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Ana Trisovic

Reviewers:

- @HaoZeke
- @jeremylt

Submitted: 07 March 2023
Published: unpublished

Summary

Distributed computing is an essential element for advancing computational sciences. Workload managers used by computing clusters help to meet the need for running large-scale simulations and efficiently sharing computational resources among multiple users (Reuther et al., 2016). In this context, we introduce a tool designed to allocate computational resources effectively for small clusters of computers commonly found in research groups lacking specialized hardware such as computing fabric or shared file-systems. Additionally, compute nodes within the cluster can also serve as standard workstations for researchers, and any ongoing job on a machine can be temporarily suspended to enable normal research activities to proceed.

Statement of need

Many research groups face challenges with inefficient utilization or over-scheduling of their computational resources. While some computers work for days to complete a queue of simulations, others remain unused. Dividing simulations manually among computers is time-consuming and often ineffective due to varying hardware configurations. Additionally, multiple individuals running jobs on a single computer can slow simulatiaoni progress for everyone involved.

To meet the growing challenges of managing compute resources between researchers, workload managers such as Slurm (Yoo et al., 2003) and TORQUE (Staples, 2006) quickly rose in popularity. These workload managers enforce strict upper bounds on memory, CPU time, and CPU cores on each job submitted. Leveraging known runtime and hardware constraints, these traditional workload managers optimize the scheduling of hundreds of jobs concurrently across thousands of CPU cores and terrabytes of memory.

While Slurm and TORQUE provide elegant and powerful solutions to scientific computing, they are incompatible with cheap off-the-shelf hardware present in many research labs. In order to schedule a compute task across hundreds of cores and dozens of individual computers concurrently, traditional workload managers require specialized memory and filesystems to enable communication between multiple computers. Although this hardware is standard in conventional supercomputer clusters, it leaves collections of common desktop computers present in research labs underserved.

Users with access to scientific computing clusters are usually given minimal priviledges: only enough to start jobs and download the results. Since users do not have access to individual compute nodes, Slurm and TORQUE generally assume they are used for purely compute reasons and provide no method to temporarily halt a job on a single node. However, in a lab environment, desktops used for compute purposes often double as workstations for researchers, and the execution of jobs in the background may slow down day-to-day work.

The Journal of Open Source Software

distribute Overview

Our tool, *distribute*, eliminates the need for specialized hardware such as memory fabric or shared filesystems, making almost no assumptions about the architecture of the computers. It schedules multiple jobs concurrently across several computers, as shown in Figure 1, avoiding the simultaneous execution of a single job across multiple computers. Since the hardware that *distribute* targets also doubles as research workstations, we offer a simple interface to pausing the background execution of jobs for normal research work to continue.

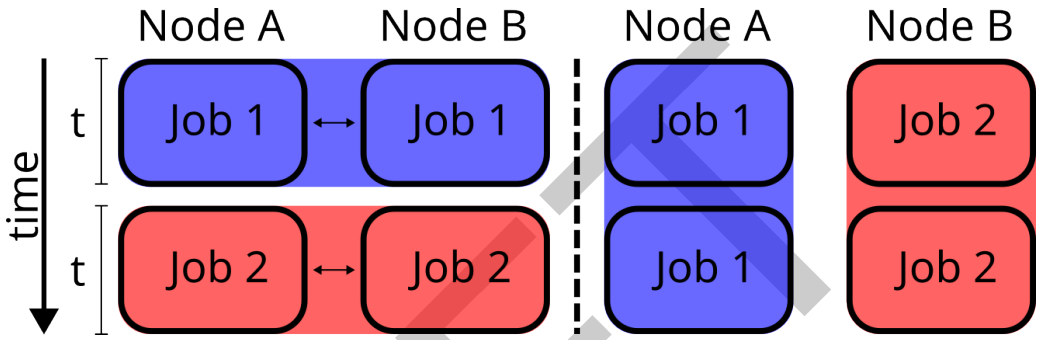

Figure 1: Left: traditional horizontally scaling compute framework. Right: proposed compute framework.

While Slurm and TORQUE provide simple interfaces to schedule a single large job, *distribute* focuses on the scheduling of tens to hundreds of small to medium sized jobs. In a *distribute* configuration file, a method for compiling a computational task ("solver") is specified with a list of job names. Each job name specifies a list of files that serve as inputs to the solver. Figure 2 provides an overview of the *distribute* execution model. When each job is scheduled, *distribute* ensures that the solver is correctly compiled on each compute node and provided with access to the input files for the job. When the job is completed, the solver outputs are transported and archived on the head node.

In distribute, the user is given two methods of providing a solver. In the first method, an apptainer (Collaboration, 2022) image may be compiled and submitted with the configuration file. With this method, the user is able to verify that all dependencies are correctly supplied and the container functions exactly as expected. Although less robust, the alternative method is to supply a python script to compile your solver at runtime on each compute node.

After the completion of job batches, *distribute* provides a simple method to query the archived outputs and selectively download files from the head node. With this methodology, users can avoid manually writing scripts with *rsync* to download files stored on a compute cluster.

Running a multidimensional parameter space sweep may result in hundreds of jobs, and our tool provides a Python package to programmatically generate the configuration file. Furthermore, *distribute* can transpile its job execution configuration to the SLURM format, making it easy to run a batch of jobs previously executed on a *distribute* cluster of in-house machines on a larger cluster with hundreds of cores.

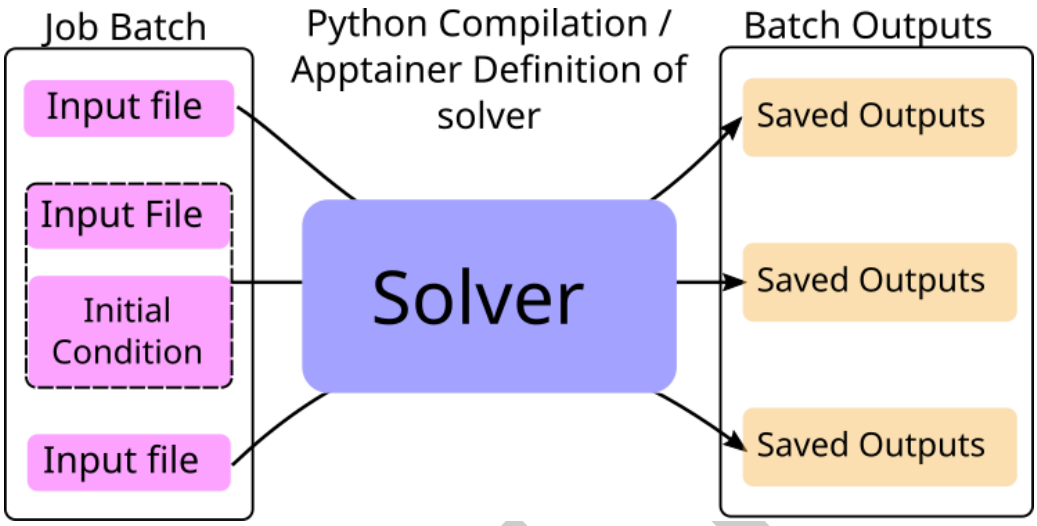

Figure 2: The input-output model for *distribute* jobs. A stateless solver executes a computational workload and produces output.

Acknowledgements

AGN acknowledges the support the National Science Foundation AI Institute in Dynamic systems (Award no: 2112085, PM: Dr. Shahab Shojaei-Zadeh).

References

Collaboration, A. (2022). Apptainer: Application containers for linux. In *GitHub repository*. GitHub. https://github.com/apptainer/apptainer

Reuther, A., Byun, C., Arcand, W., Bestor, D., Bergeron, B., Hubbell, M., Jones, M., Michaleas, P., Prout, A., Rosa, A., & others. (2016). Scheduler technologies in support of high performance data analysis. *2016 IEEE High Performance Extreme Computing Conference (HPEC)*, 1–6. https://doi.org/10.1109/hpec.2016.7761604

Staples, G. (2006). Torque resource manager. *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, 8–es.

Yoo, A. B., Jette, M. A., & Grondona, M. (2003). Slurm: Simple linux utility for resource management. *Job Scheduling Strategies for Parallel Processing: 9th International Workshop, JSSPP 2003, Seattle, WA, USA, June 24, 2003. Revised Paper 9*, 44–60. https://doi.org/10.1007/10968987_3