

From Laptop to Cluster: HPC Translation Patterns for AI Stacks

Parmanand Sinha, Ph.D. pnsinha@uchicago.edu

Research Computing Center, University of Chicago, 0000-0001-6992-6555

Keywords: high-performance computing, research software engineering, workflow migration, containers, orchestration

Research software teams increasingly inherit AI workflows that were built for laptops or small servers and are then expected to run them on shared HPC systems. The gap is not just one of scale. It reflects a mismatch in operating model. Docker Compose assumes long-lived services, open ports, and local control over processes and storage. HPC centers instead enforce scheduler-mediated execution, restricted container runtimes, shared filesystems, and tighter network boundaries. Mature tooling exists for translating Compose to Kubernetes, but no equivalent path exists from Compose to Slurm. Moving these stacks is therefore not just a packaging task but a systems translation problem that recurs across research computing support work.

This talk presents work in progress from a Better Scientific Software Fellowship project that is building a practical pattern library for translating multi-service AI stacks into HPC-native workflows. The focus is on recurring technical decisions: treating the scheduler allocation as the orchestration boundary, mapping service graphs to Slurm or PBS launch structures, replacing Docker-centric assumptions with Apptainer and Charliecloud runtime patterns, handling state through job-scoped storage, and restoring interactivity through controlled access paths such as tunneling and brokered services. The project also examines where distributed execution tools such as Globus Compute fit naturally into these designs and where portability limits appear across centers. Outputs are published openly at laptop-to-cluster.org.

The material is informed by active deployment work at the University of Chicago Research Computing Center, including A3RD (AI-Assisted Archaeological Remains Detection), where prototype AI services had to be re-expressed in terms the cluster can support. One migration runs end to end in the talk. A nine-service Compose stack becomes nine rootless container invocations under a single allocation: readiness polling replaces `depends_on`, named volumes become binds onto job-scoped scratch, and GPU access requires explicit CUDA library binding rather than a runtime flag. The failure modes are as instructive as the successes. Relative-path resolution in COPY differs from Docker, and SquashFS conversion was unavailable on the target system, forcing a directory-format fallback.

The talk distills implementation patterns that RSEs can reuse when moving a working multi-container prototype onto a batch-scheduled system. Attendees will see how these patterns reduce rework, make portability boundaries more visible, and provide a more precise way to reason about workflow migration across centers. Patterns are labelled by validation status. The Slurm and Apptainer path is validated on NSF ACCESS Anvil and exercised on a second GPFS-backed cluster, with Charliecloud runtime independence confirmed; PBS-family templates and non-x86_64 targets are identified but not yet validated.

Acknowledgment. This work was supported by the Better Scientific Software Fellowship Program, a collaborative effort of the U.S. Department of Energy (DOE), Office of Advanced Scientific Research via ANL under Contract DE-AC02-06CH11357 and the National Nuclear Security Administration Advanced Simulation and Computing Program via LLNL under Contract DE-AC52-07NA27344; and by the National Science Foundation (NSF) via SHI under Grant No. 2435328.

Connection to Mission, Goals, & Interests of US-RSE Community:

Translating a working multi-service prototype onto a batch-scheduled cluster is work RSEs do constantly. This submission turns one such translation into shared, reusable material, which connects directly to US-

RSE's aims of advocating for the research software engineer role, building a professional community, and providing resources its members can use.

This is characteristic RSE labor, solved independently at center after center because it seldom yields a publication. Existing documentation stops short: center guides cover single containers under a scheduler, not multi-service stacks whose components expect a network the scheduler does not provide.

The talk makes that work visible and transferable. Outputs are published openly at laptop-to-cluster.org under a permissive license: learning modules, scheduler launch templates, and per-site notes recording what varies between centers. They apply to any multi-service research stack, not only AI workloads, and are structured to feed training on the HPC-specific skills involved. The per-site notes invite contribution, since portability boundaries can only be established by testing at more than one center.