Geospatial AI on HPC

Infrastructure contracts for reproducible containerized workflows

Parmanand Sinha
Computational Scientist, Research Computing Center · University of Chicago
2026 Better Scientific Software Fellow

I-GUIDE Forum 2026, co-located with the NSF HDR Ecosystem Conference

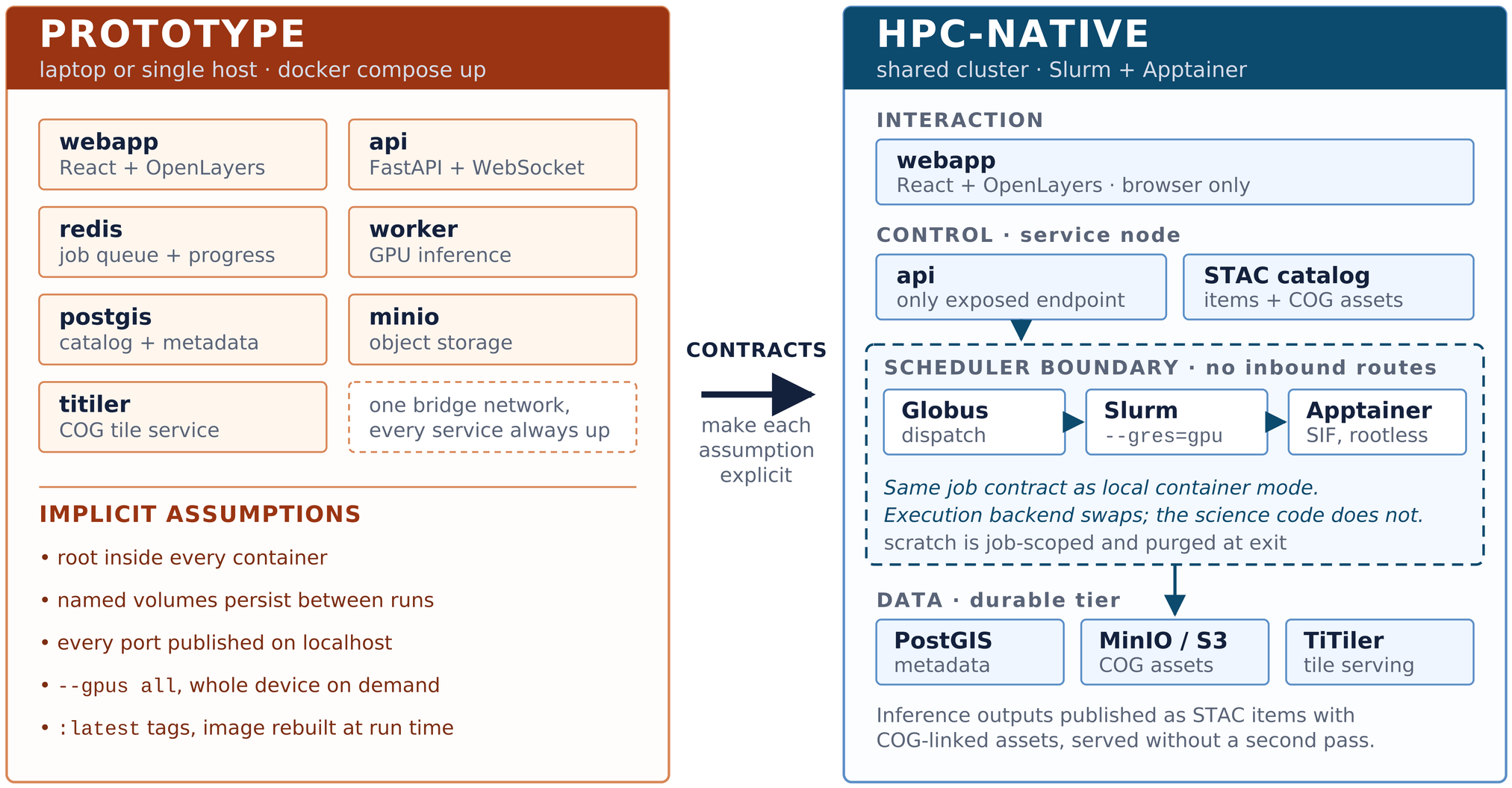

Figure 1. Moving a containerized geospatial AI stack from a single-host prototype to a shared HPC cluster. The implicit assumptions on the left are what the runtime, data, and execution contracts make explicit.

Reproducibility in convergence science depends on treating compute scheduling, geospatial cataloging, and access governance as one design surface.

The gap

Geospatial AI workflows often break when teams move from cloud development to HPC production. Containers package dependencies, but execution assumptions change across schedulers, GPU allocation models, and storage layouts.

Geospatial outputs add a second source of drift. They are spatial assets that have to stay discoverable, tileable, and traceable back to their inputs and run settings.

Three gaps

Runtime portability is incomplete

Many HPC sites disallow privileged containers and require Apptainer or Singularity under scheduler-managed GPU allocation. Workflows that assume Docker-only behavior lead teams to fork code paths or patch per site.

Data products are under-specified

Users cannot answer basic questions about outputs stored as files without complete metadata. Which model produced this raster? Which scene did it come from? Can a tile server stream it now?

Orchestration data is too thin for replay

A queue that records only "job succeeded" does not support operational replay. Replay needs model version, input asset references, parameter set, runtime path, timestamps, and workspace scope.

Stack

`Apptainer` `Singularity` `Slurm` `Globus Compute` `STAC` `COG` `Redis` `OTBTF` `FastAPI` `WebSocket`

Three contracts

Standardize the contracts, then let heterogeneous backends sit behind them.

Runtime contract

One job schema across local and HPC execution. Local workers run containerized inference directly. HPC workers execute equivalent jobs under scheduler allocation with Apptainer.

The API contract, input bindings, output paths, and status schema do not change. Only the runtime layer beneath them does.

Data contract

Outputs are catalog objects, not loose files. Register them as STAC items with asset metadata and store rasters in COG-compatible form so tile services can stream them directly.

Execution contract

Queue-backed GPU dispatch with explicit lifecycle state and persisted run metadata. Status transitions, errors, and artifact URIs belong to the contract, not to optional logs.

Reference implementation

A production deployment at the University of Chicago runs object detection over historical and modern satellite imagery.

Locally, a GPU worker consumes jobs from a Redis queue and spawns sibling OTBTF containers. On HPC, the same jobs dispatch via Globus Compute to Slurm-allocated endpoints running Singularity. FastAPI handles submission, workspace resolution, and role-based access, so every job carries its governance context.

Design validation

Operational evidence from deployment behavior, not a benchmark suite. A formal cross-site benchmark is future work.

1. Queue-mediated execution keeps API interactions responsive during long-running inference. Users get immediate job acknowledgment and asynchronous progress over WebSocket while GPU workers run compute-bound tasks.
2. One job contract holds across local container mode and HPC dispatch mode. The same job schema, status state machine, and artifact registration path apply to both backends.
3. Inference outputs become STAC-linked products during normal job completion rather than through post hoc curation, which reduces artifact drift and supports immediate map-based inspection.
4. Workspace-scoped access control runs inside the job and data workflows. A run that cannot be tied to tenant scope is not fully reproducible for operational use.

Limits

No cross-site replication under multiple institutional scheduler policies yet, and no formal cost-performance analysis across cloud and HPC tiers.

Full guide and companion repository
`laptop-to-cluster.org`
pnsinha@uchicago.edu

Middleware decisions that are often treated separately are tightly coupled in practice. Scheduler behavior influences data movement. Data contract design influences serving latency. Access policy influences what can be replayed and audited.

Infrastructure patterns deserve evaluation by contract quality and transferability across sites, not only by raw throughput. A system that runs slightly slower but stays replayable, auditable, and portable can be worth more to convergence science than a brittle fast path.

The pattern transfers to hazard response mapping, agriculture monitoring, and urban change analysis, as long as the data contract and scheduler interfaces stay explicit.

Supported in part by a 2026 Better Scientific Software (BSSw) Fellowship. The BSSw Fellowship Program is supported by the U.S. Department of Energy, Office of Advanced Scientific Computing 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 via SHI under Grant No. 2435328.

STAC Specification v1.0.0 (Radiant Earth, 2021). · OGC Cloud Optimized GeoTIFF Standard, OGC 21-026 (2023). · Kurtzer, Sochat and Bauer, PLOS ONE 12(5), 2017. · Wang, Annals of the AAG 100(3), 2010. · Soroush et al., SPIE 2020.

Poster #23 · Wednesday August 5, 5 to 7pm · Thompson C, Student Center West