DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING, THE OHIO STATE UNIVERSITY

AI-Driven Resource Optimization for High-Performance Computing: A Comprehensive Framework

Manikya Swathi Vallabhajosyula, Rajiv Ramnath

Personal Webpage

Publications

Motivation:

Shared HPC centers waste cycles because users must **guess** walltime, memory, and accelerators across heterogeneous policies—driving requeues, poor backfill, and long turnaround. The problem is structural: steep learning curves and policy complexity push "guess-and-submit" behavior. A **center-first, estimation-driven** stack addresses this by right-sizing submissions, composing **policy-compliant** plans, and closing the loop with **event-driven orchestration**, yielding fewer failures/resubmissions, higher utilization, and shorter time-to-solution without disrupting user workflows. To make this concrete, the approach is realized through four tightly coupled components at the center:

(A) Estimators (black-box + white-box): Fast, CI-aware predictions of runtime and memory from hardware and configuration, calibrated for accurate feasibility and cost/time forecasts.
(B) HPC Application Resource Prediction (HARP) framework: A standardized pipeline that selects the best estimator under measured error and site policy, yielding policy-compliant resource plans.
(C) Estimator with Scheduler integration: Turns estimates into valid submissions, chooses partitions/QoS, and balances time versus cost with resubmission and what-if planning.
(D) Rule-engine based orchestration: Kafka-driven events feed the Intelligence Plane to trigger scheduler and services, with history/policy feedback for continuous improvement.

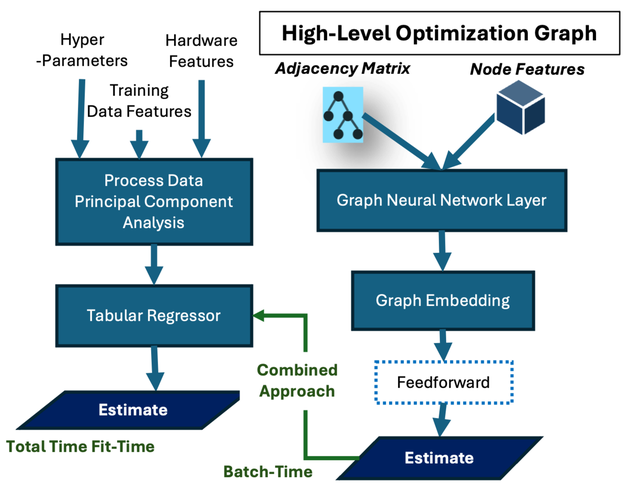

Figure 1: Architecture of Estimation Model Pipeline

(A) Estimators (black-box + white-box):

Need: Accurate, fast **submission-time** estimates to end "guess-and-submit" (OOM/timeouts, requeues, poor backfill).
Two Types of Estimators:

- **Black-box (history-driven):** Learns from prior runs; captures center effects (queues, libs, contention); **accurate in-distribution**, low latency; weak extrapolation.
- **White-box (model-aware):** Uses operator/graph features (From High-Level Optimization (HLO) Graphs) + scaling laws; **generalizes** to new hardware/configs; requires introspection/feature extraction,
- **Hybrid:** Combine BB (fast fit to observed behavior) + WB (structure & generalization). Joint calibration reaches **comparable error** with fewer profiles—less data and cost—while still enabling submission-time runtime/memory estimates.

Model	Type	Target Models	Data	MAPE (%)
Tabular Model 1	Blackbox	Seen	Full Training Data (10K)	3.7
Tabular Model 2	Blackbox	Seen		**1.63**
Graph Network	Whitebox	Seen		11.16
Tabular Model 1	Blackbox	Unseen		87.96
Tabular Model 2	Blackbox	Unseen		**10**
Graph Network	Whitebox	Unseen		13.54
Tabular Model 2	Blackbox	Unseen	20% Training Data (2K)	77.29
Graph Network	Whitebox	Unseen		75.48
Tabular Model 2 + Graph Network	BOTH	Unseen		**10.55**

Table 1: Comparison of Estimation Models – Blackbox (Tabular Model 1 & 2: Random Forest & XGBoost) using tabular features vs. Whitebox (Graph Neural Network) using HLO graph features

(B) HARP framework: Generate → Build → Predict

Problem: Centers lack application-and execution-aware training data; coarse logs and ad-hoc traces don't capture model, input, and hardware interactions, making predictors brittle and non-transferable.
Goal: A repeatable pipeline that creates the right data, trains the right models, and delivers submission-time predictions of runtime and memory for diverse workloads at the center.

- **(Step 1) Generate:** Profile workloads (downsized + full runs) across hardware/configs, collecting runtime, utilization, memory, I/O, and metadata [+ optional HLO/graph].
- **(Step 2) Build:** Process features (hardware, hyperparameters, input) and, if available, graph/HLO. Train two estimators: black-box (Random Forest/XGBoost) for quick fit and white-box (graph/operator-aware) for generalization.
- **(Step 3) Predict:** Given hardware + config + metadata [± HLO], output runtime/memory with uncertainty. Provide APIs for submission-time queries and "what-if" config comparisons.

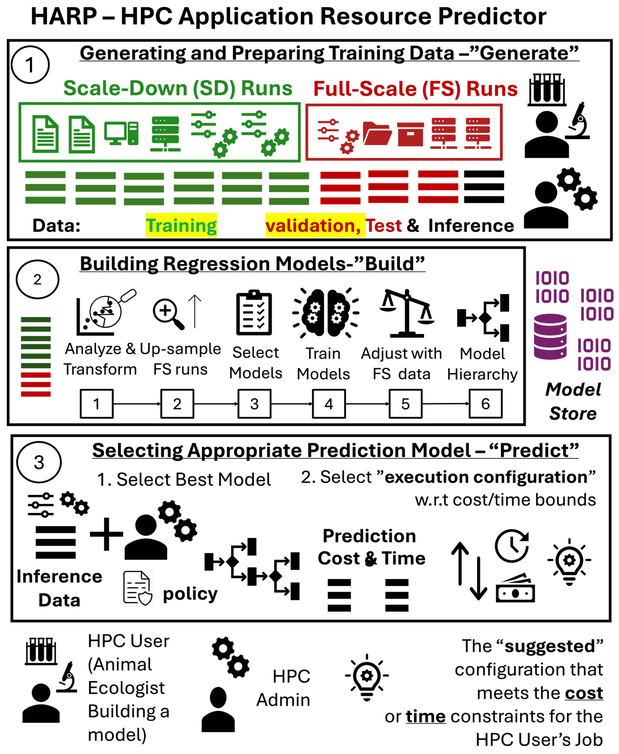

Figure 2: End-to-end framework for workload profiling, estimator training, and runtime/memory prediction.

(C) Estimator with Scheduler integration:

Predictions only create value when they become **policy-compliant, queue-aware submissions**. (B) **HARP** supplies submission-time estimates; the scheduler turns them into executable plans that cut failures, shorten waits, and control cost—without adding user burden.
Components:

- **What-If Analyzer.** Enumerate HARP-suggested configs; present fastest/cheapest **policy-valid** options.
- **Plan Builder:** Fuse HARP outputs with the **Cyber-Infrastructure (CI) DB** to choose partition/QoS, walltime, memory, devices; enforce policy/billing limits; emit a validated **plan spec**.
- **Queue-Aware Selector.** Compare candidate plans against **current availability** (idle nodes, queue depth); select by **time–cost** targets.
- **Scheduler Adapter.** Register apps/containers, submit jobs via TAPIS APIs (Internally SLURM), manage tenancy/tokens, and track versions.
- **Submission & Monitoring.** Launch with callbacks; stream status/telemetry to **Kafka** for progress and exceptions.
- **Re-scheduler.** Checkpoint/restart, split near walltime limits, or migrate partitions when estimates vs. actuals diverge.

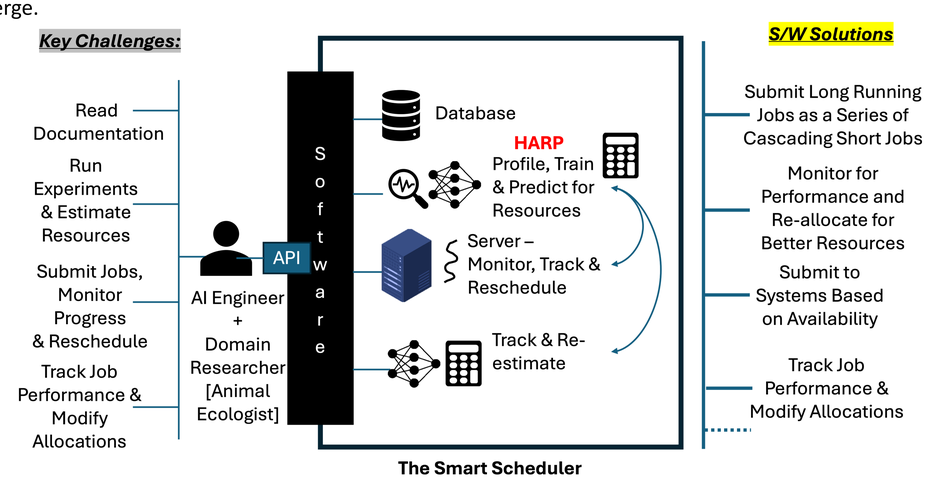

Figure 3: Challenges to Solutions using Resource Estimators with HARP and Executables with Scheduler.

(D) Closed-Loop Orchestration & Path to Agentic Scheduler:

Kafka streams job/filesystem signals into the Backend Services, which apply estimators and policies to drive the scheduler, data-gen, and orchestration.
Next: add goal/constraint inference and drift-triggered self-updates for autonomous scheduling & training—LLM-optional, MCP-ready.

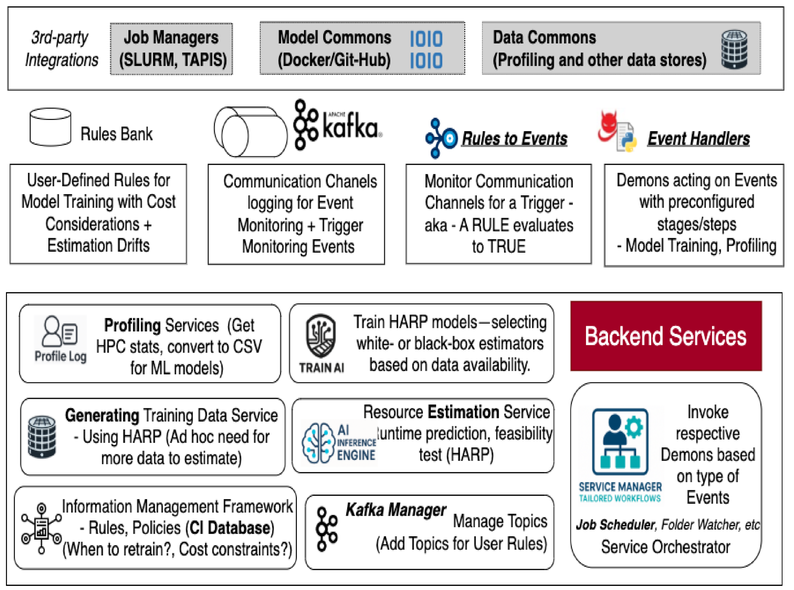

Figure 4: End-to-end framework for workload profiling, estimator training, and runtime/memory prediction.

Agents:

- Intelligence Plane (coordination)
- Resource Estimation (HARP BB/WB)
- Optimization,
- Profiling/Data-Gen,
- Scheduler/Orchestrator, and
- Model Promotion

Each:
senses → decides → acts for its subgoal.

Not agents: Kafka/Kafka Manager (event bus), SLURM/TAPIS job managers (external actuators), Data/Model Commons (storage), and policy/CI DB (knowledge base).

The Ohio State University College of Engineering
ICICLE DEMOCRATIZING AI
OAC-2112606
NSF