Diamond Agent: Agentic Control of Federated HPC Resources as a Service

Haotian Xie*, Junlin Chen*, Mingkai Zheng*, Zhao Zhang*
*Rutgers University. Email: {haotian.xie, junlin.chen110, mingkai.zheng, zhao.zhang}@rutgers.edu

***Abstract*—Scientific coding agents can plan high-performance computing (HPC) workflows, but resources are fragmented across independently administered sites with different schedulers, queues, architectures, and data locations. Diamond Agent provides a user-scoped control plane through which one agent can inspect, submit, and monitor work across multiple supercomputers. A ResourceEnvelope specifies minimum viable and maximum useful resources, while deterministic checks enforce authorization and compatibility before placement. Durable task state and event-driven continuation let the agent yield while batch jobs run. In 19 matched placement rounds comprising 83 jobs on four production systems, the median additional completion time relative to the fastest observed site fell from 42 s for a fixed-site policy to 4 s for best-feasible placement.**

***Index Terms*—agentic AI, high-performance computing, event-driven agents, federated resource service, Globus Compute, resource management, AI for Science**

I. Introduction

An agent confined to one supercomputer sees only that system's queue, storage, and execution environment. Moving a workflow to another facility can require another agent session, new job scripts, data transfer, and manual reconstruction of task state. Long queue waits also make repeated model-side status checks wasteful. Diamond Agent extends the Diamond execution service [1] with a persistent agent-facing workspace. Typed operations expose federation inspection, resource specification, task and task-group submission, data movement, and result retrieval. One agent keeps a unified task history while each center continues to control admission, priorities, accounting, and execution.

II. System Design

Diamond Agent maintains a Global Resource View that records each authorized site's architecture, accelerators, partitions, queue state, allocation, and endpoint health. For each workload, a ResourceEnvelope defines its minimum viable resources, maximum useful resources, and compatibility constraints. A user can supply the envelope or request an AI draft; the draft remains auditable. Deterministic rules check access, account credit, hardware compatibility, capacity, wall time, and site policy. The best-feasible policy then minimizes estimated queue wait, runtime, and environment setup over the feasible candidates. This estimate does not guarantee the fastest actual completion.

A persistent Task Orchestrator owns durable handles, execution attempts, results, and events after the agent yields. Site adapters dispatch work through Globus Compute [2]; Globus Transfer [3] handles large cross-site data movement. The orchestrator monitors jobs, records provenance, and performs bounded re-placement when policy permits. A result or decision-relevant event reactivates the agent with compact state and output references, so the agent need not stay active throughout a batch job.

III. Evaluation

The evaluation spans Delta, DeltaAI, Anvil, and Vista. During 27 hours of monitoring, the collector produced 543 federation snapshots with mean data freshness of 183 s. In 19 matched submission rounds, 83 fixed-shape proxy jobs were sent to candidate sites within a narrow time window. Relative to each round's fastest observed job, median additional completion time was 42 s for a fixed Delta policy and 4 s for best-feasible placement. Best-feasible's median actual completion was 196 s, close to the 194 s post hoc fastest median. However, it picked the actual fastest site in only 47% of rounds, and its 95th-percentile additional completion was 3,000 s. Queue-estimation error remains significant.

Three AI-drafted ResourceEnvelopes were tested in 33 sizing jobs. They captured CPU preparation, single-GPU inference, and multi-GPU training, but useful parallel speedup did not always minimize completion under queue delay. An infeasible high-memory request was rejected at every site with explicit reasons.

The poster's cross-system application demonstration uses DeltaAI GPUs to serve a language model and Frontera CPUs to run Terminal-Bench sandboxes. In another experiment, four GPU workers were initially distributed across Vista, Delta, DeltaAI, and Anvil. When the Anvil worker remained queued, Diamond Agent re-placed it on Vista; the group finished in about 32 min. After a separate endpoint loss, a replacement was submitted 34 s after detection. Two end-to-end coding-agent workflows resumed after 37- and 84-min remote jobs without model-side status calls.

IV. Conclusion

Diamond Agent shows that one persistent agent interface can coordinate heterogeneous HPC allocations while preserving site authority. The measured placement benefit is typical-case rather than a worst-case guarantee: the matched study used short proxy jobs, four sites, and showed large tail delays. Future work should jointly optimize resource size, site placement, and data movement, and improve queue prediction from observed outcomes.

REFERENCES

[1] H. Xie et al., "Diamond: Harnessing GPU Resources for Scientific Deep Learning," in Proc. IEEE Int. Conf. eScience, 2025, pp. 196-204, doi: 10.1109/eScience65000.2025.00031.

[2] R. Chard et al., "funcX: A Federated Function Serving Fabric for Science," in Proc. 29th Int. Symp. High-Performance Parallel and Distributed Computing, 2020, pp. 65-76, doi: 10.1145/3369583.3392683.

[3] K. Chard, I. Foster, and S. Tuecke, "Globus: Research Data Management as Service and Platform," in Proc. Practice and Experience in Advanced Research Computing, 2017, pp. 1-5, doi: 10.1145/3093338.3093367.