ToolFuse: Provenance-Aware In-Flight Coalescing of Redundant Agent Tool Calls

Izzet Yildirim Xian-He Sun Anthony Kougkas
iyildirim@hawk.illinoistech.edu, sun@illinoistech.edu, akougkas@illinoistech.edu
Illinois Institute of Technology, Chicago, IL, USA

Gnosis Research Center @ ILLINOIS TECH

Overview: motivation, approach, and outcome

Background and motivation. Scientific LLM agents have isolated workspaces but can repeat expensive reads of immutable data. A completed-result cache misses the first concurrent wave.

Objective and approach. Reduce redundant execution without unsafe sharing. ToolFuse coalesces active calls only when a provenance-aware identity establishes equivalence. Completed results are not retained.

Measured outcome. Under contention, **16** equivalent requests become **one** execution, reducing CPU time by **11.79×** and makespan by **2.36×**. Cheap-call makespan overhead is at most **0.075%** in the measured control.

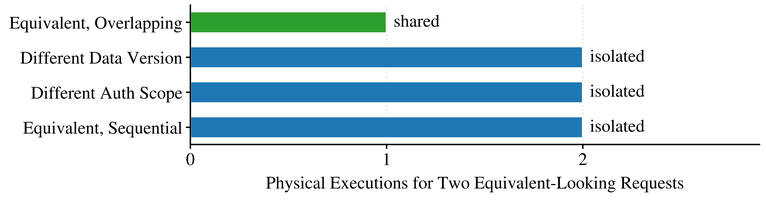
Evidence. Of **2,580** read-like calls across **625** sessions [6], **2,485** repeat. A second corpus of **127** agents shows **21%** redundant tool time and a **one-second** median gap between equivalent cross-agent calls.

Prior art and the equivalence boundary

- **Go singleflight** [5] suppresses active duplicates using a caller-supplied key
- **Parsl** [2], **Pegasus** [3] memoize completed work, and Parsl documents that concurrent identical cold calls launch multiple instances [4]
- **ToolCacheAgent** [1] reuses completed tool results

Completed-result reuse misses the first concurrent wave. Singleflight addresses that wave, but leaves safe equivalence to the caller. **ToolFuse supplies a provenance-aware identity at the scientific tool boundary. In-flight suppression itself is standard.**

Execution identity, and what it refuses



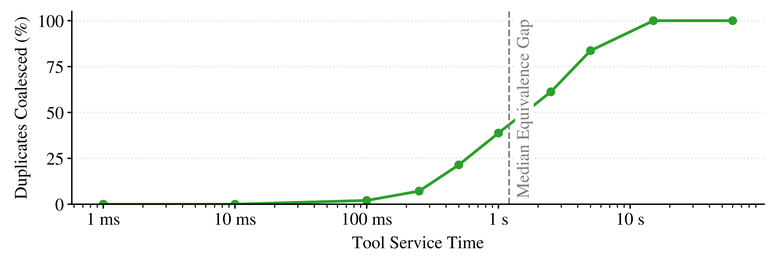
Only read-only, idempotent, deterministic, closed-world calls are eligible. The execution key hashes canonical arguments with versioned data dependencies, tool contract, implementation, environment, authorization scope, freshness epoch, and relevant hidden state.

The live safety matrix uses a real handler. Different data versions and authorization scopes execute separately. Equivalent overlapping calls share, while sequential equal calls execute twice.

One producer, N waiters, nothing retained

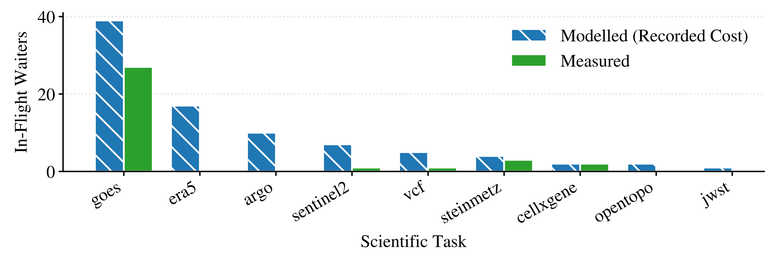
ToolFuse sits between agents and tool handlers. The first request for an active key becomes the producer. Equivalent concurrent requests await its shielded task and receive the same immutable result. Cancelling a waiter does not cancel the producer. The key is deleted on success or failure, so later calls execute again. **No completed result is retained.** The gateway is agnostic to the agent runtime.

Sharing grows with call duration



The model uses first-corpus discovery traces with fixed arrivals. Sharing grows near the **1.2-second** median equivalence gap. Real replay of sub-millisecond calls gave **zero** sharing and at most **0.075%** makespan overhead. A cache removed 44 to 75% of executions at 4 to 8 agents.

Measured on natural agent traces

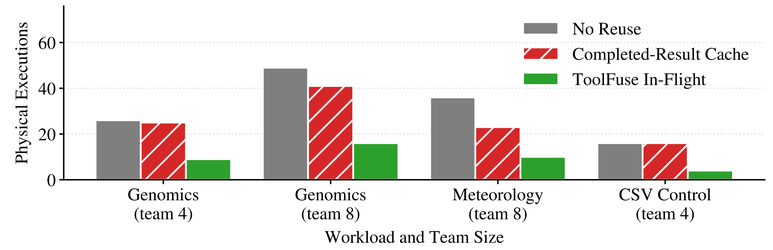


Modeling the second corpus at its recorded arrival offsets and service times predicts **87 of 919** calls overlapping in flight. Real execution of five tasks confirms sharing, with counts affected by changed service times.

Limitations

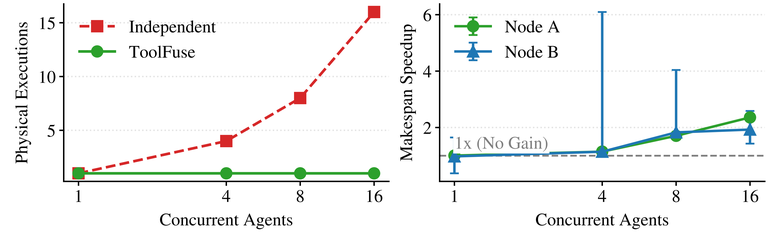
Recorded sessions ran separately, so trace concurrency is counterfactual. Natural replay shows unchanged makespan with fixed arrivals and 97 to 99% idle nodes. Earlier returns cannot advance later requests in this replay.

An empty cache misses overlapping calls



With **5 s injected delay**, discovery-call duplicates miss an empty cache. In eight-agent genomics replay, cache executions rise from **14 to 41**, versus **16** for ToolFuse at that delay.

Under contention, work and time both fall



Ten paired repetitions use a real Parquet handler on an exclusive 40-CPU Ares node. No model is timed. Digests match, and peak memory growth falls from **6.44** to **0.53 GiB**. Error bars show 95% CIs.

ToolFuse brings provenance-aware execution identity to in-flight sharing of scientific tool calls. **Under contention, 16 equivalent requests become one execution, reducing CPU time by 11.79× and makespan by 2.36×, with negligible overhead in the cheap-call control.**

Acknowledgment

This material is based upon work supported in part by the National Science Foundation (NSF) under Awards OAC 2411318 and OAC 2313154.

References

[1] Anonymous. Tool Cache Agent: Accelerating LLM agent through intelligent tool call caching. Anonymous ICLR 2026 submission, OpenReview, 2026.

[2] Y. Babuji, A. Woodard, Z. Li, D. S. Katz, B. Clifford, R. Kumar, L. Lacinski, R. Chard, J. M. Wozniak, I. Foster, M. Wilde, and K. Chard. Parsl: Pervasive parallel programming in Python. In *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing (HPDC)*, pages 25–36. ACM, 2019.

[3] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. Ferreira da Silva, M. Livny, and K. Wenger. Pegasus, a workflow management system for science automation. *Future Generation Computer Systems*, 46:17–35, 2015.

[4] Parsl Project. Memoization and checkpointing. Parsl 1.2.0 user guide. Accessed: September 30, 2026.

[5] The Go Authors. Package singleflight. Go package documentation, golang.org/x/sync v0.23.0. Accessed: September 30, 2026.

[6] I. Yildirim, X.-H. Sun, and A. Kougkas. AgentStage: Exploiting LLM thinking for data staging in scientific agents. In *22nd IEEE International Conference on eScience (eScience)*, 2026. Accepted.