StageOnce: Speculative Cross-Job Data Staging for LLM Agent Fleets

Izzet Yildirim
Illinois Institute of Technology
Chicago, IL, USA
iyildirim@hawk.illinoistech.edu

Xian-He Sun
Illinois Institute of Technology
Chicago, IL, USA
sun@illinoistech.edu

Anthony Kougkas
Illinois Institute of Technology
Chicago, IL, USA
akougkas@illinoistech.edu

***Abstract*—Scientific workflows increasingly run as fleets of LLM agents analysing the same large immutable datasets. On a shared compute node, independent staging repeats source reads and creates storage contention, while each job discovers its inputs during reasoning. StageOnce coordinates these speculative path predictions through a node-local daemon, coalescing concurrent misses into one immutable copy while preserving dataset versions and private outputs. Tools read staged inputs directly from the filesystem, without per-read coordination. Recorded-session replays in genomics and astronomy on SSD and HDD backends reduce fleet makespan by 31–51% at 12 concurrent jobs against independent staging. Logical staging remains one copy across every tested fleet size. An ablation isolates speculation, which reduces makespan by 8–49% against reactive shared caching.**

I. Introduction

Scientific agents increasingly run as fleets: many independent jobs, on one node, analysing the same immutable dataset. Each stages its own inputs, so the fleet issues the same reads many times over.

For a frozen corpus of 72 agent sessions across six workloads and four reasoning models, independent per-job staging would move 573.0 GiB versus a retained union of 47.8 GiB, a 12.0× amplification, with 91.7% of predicted bytes avoidable. Genomics tracing shows 33% file-count precision and 99.9% byte precision.

Burst-buffer filesystems and node-local staging put data close to compute [1]. Informed prefetching acts on hints the program supplies [2]. Scheduler-guided preloading needs inputs declared at submission [3]. Cross-task caching reuses data across tasks [4]. Speculative agent execution acts on mid-run predictions [5], but within one agent. These systems use inputs already known or predictions retained within one job. An agent fleet's working set emerges during the model's thinking phase, before tool I/O begins. Each job discovers its inputs separately, so their predictions must be coordinated without breaking output or version isolation.

StageOnce closes that gap by coordinating predictions across jobs while preserving version and output isolation. We separate the benefits of speculation and sharing, then identify the regimes where their combination pays.

II. Method

StageOnce is a node-local control plane keyed by `(project, dataset, version)`. It coalesces concurrent misses so each file is fetched once, publishes one immutable copy atomically, and returns a direct filesystem root so reads are served locally with no per-read RPC. Outputs remain private and version boundaries are enforced. StageOnce accepts path predictions from any producer. Ours come from AgentStage [6], which extracts file paths from the model's token stream and redirects reads through a preload shim, but keeps its staged state private to one session.

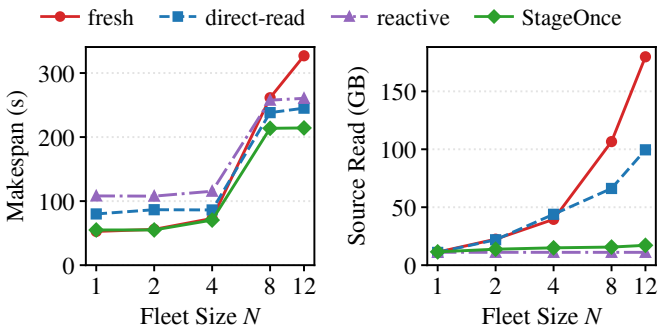


Fig. 1. Makespan and source read bytes vs. fleet size (genomics, SSD). StageOnce makespan levels off at larger fleets. Source traffic grows far more slowly than independent staging.

We replay those recorded trajectories with the models removed, on a colocated OrangeFS deployment on Ares over SSD and HDD backends, with the cache on node-local NVMe. Fleet size varies over $N = 1, 2, 4, 8, 12$ with synchronized starts and 2–3 repetitions. Alongside independent staging (*fresh* in the figures) and a direct-read baseline we add two controls: *reactive*, an identical shared one-copy cache that stages at point-of-use instead of speculatively, and *prestage*, a prologue that bulk copies the dataset once. Storage traffic uses a validated server-side counter that measures source read bytes. Sessions are paired within a repetition but redrawn between repetitions, so effects are estimated per repetition.

III. Results

Against independent staging at $N = 12$, fleet makespan falls by 31–51% across both domains and both storage tiers (Fig. 2), sign-consistent in every repetition. Logical staging remains one copy. Measured source traffic grows from 11.5

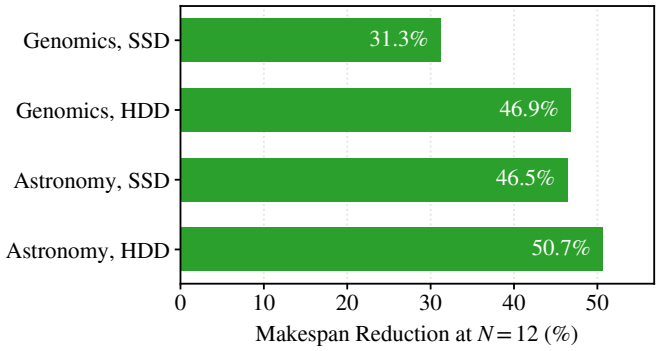


Fig. 2. Makespan reduction against independent staging at $N = 12$, for genomics and astronomy on SSD and HDD.

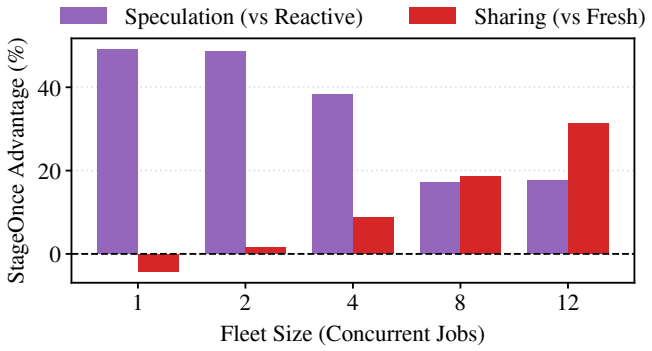


Fig. 3. Two separable effects: speculation, isolated against the reactive control, and sharing, against independent staging.

TABLE I
WHERE SOMETHING SIMPLER WINS.

Regime	**What wins**	**Margin**
506 MB per job	direct read	StageOnce +4.5% at $N = 12$
Working set ≈ dataset	prologue bulk copy	3–7%
$N = 1$ and most $N = 2$ cases	independent staging	direction only

to 17.1 GB, versus 11.3 to 179.8 GB for independent staging (genomics/SSD, Fig. 1).

A shared cache alone does not explain the result. The reactive control differs only in *when* staging happens. Speculation yields 8–49% reductions, with consistent direction in 18 of 20 cells across four workload/tier groups. Figure 3 shows genomics/SSD. At $N = 1$, there is no cross-job reuse, yet reactive takes twice as long as StageOnce. The controls stage the same logical bytes. Reactive has 5–10× bandwidth headroom. This control isolates the timing benefit: blocking stage-in exposes latency that speculation hides.

IV. Conclusion

Three regimes bound the result (Table I). StageOnce beats independent staging when fleet demand exceeds what reasoning windows can hide and reuse amortises coordination overhead. Smaller working sets reduce its copy cost relative to bulk prestaging. Reuse is node-local, so jobs must be co-scheduled.

The prologue comparison depends on how many fleets share setup. With a 26.9 GB root and a 10.7 GB working set, a prologue is slightly faster per fleet but incurs a 123 s setup cost that StageOnce avoids. At $N = 12$, it needs 13 to 21 fleets to amortise, with greater uncertainty at smaller fleets. The threshold grows with the dataset-to-working-set ratio: a prologue's cost scales with the dataset, ours with the working set.

Coordinating speculative path predictions across a fleet therefore pays under those conditions. Scheduler integration, capacity, and eviction remain future work.

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] M. J. Brim, A. T. Moody, S.-H. Lim, R. Miller, S. Boehm, C. Stanavige, K. M. Mohror, and S. Oral, "UnifyFS: A user-level shared file system for unified access to distributed local storage," in *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2023, pp. 290–300. [Online]. Available: https://doi.org/10.1109/IPDPS54959.2023.00037

[2] R. H. Patterson, G. A. Gibson, E. Ginting, D. Stodolsky, and J. Zelenka, "Informed prefetching and caching," in *Proceedings of the Fifteenth ACM Symposium on Operating Systems Principles (SOSP)*. ACM Press, 1995, pp. 79–95. [Online]. Available: https://doi.org/10.1145/224056.224064

[3] A. Miranda, R. Nou, and T. Cortes, "echofs: A scheduler-guided temporary filesystem to leverage node-local NVMs," in *2018 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*. IEEE, 2018, pp. 225–228. [Online]. Available: https://doi.org/10.1109/CAHPC.2018.8645894

[4] I. Raicu, Y. Zhao, I. T. Foster, and A. Szalay, "Accelerating large-scale data exploration through data diffusion," in *Proceedings of the 2008 International Workshop on Data-aware Distributed Computing (DADC)*. ACM, 2008, pp. 9–18. [Online]. Available: https://doi.org/10.1145/1383519.1383521

[5] Y. Sui, H. Zhao, R. Ma, Z. He, H. Wang, J. Li, K. Xu, K. Chen, and Y. Yang, "Parallelizing tool execution and LLM generation for low-latency agent serving," arXiv preprint arXiv:2603.18897, 2026. [Online]. Available: https://arxiv.org/abs/2603.18897

[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, main track. [Online]. Available: https://www.escience-conference.org/2026/papers