hoist: Read Ahead of Your Agent

An MCP proxy that learns which reads an agent makes next and makes them for it, in the same tool result, so the agent needs fewer LLM turns.

september 27, 2026 · Projects

---

watch an agent work a support ticket and a lot of its turns are not really decisions. it looks up the order, and because the order has a customer id in it, it looks up the customer. then the product, because the order has a product id. each of those is a full llm turn: the whole context goes back to the model so it can ask for the thing anyone reading the last result could have guessed.

hoist takes those turns. it sits between the agent and an mcp server as a proxy, records the sessions, and learns from them which reads follow which calls and where each argument comes from: a field in an earlier result, or a constant. that is a flow. when it serves the flow, it makes those reads after each call and adds their results to the tool result the agent already asked for. the agent has what it would have asked for next, and skips the turn.

it only reads. a flow calls only the tools the server does not mark as writes, so the worst it can do is a detour: a lookup the agent did not use, which costs tokens and changes nothing. that is the whole safety argument, and it is why it can afford to guess at all.

the part i like is the question it asks. not "what will the agent call next", but "will the agent use this read before its next write". reads commute with each other until something writes, so that is the event worth counting. the reach decider makes a lookup when that probability clears a threshold, and the threshold is just the ratio of what a detour costs to what a saved turn saves. there is no model and no key. it counts, and a decision takes under a millisecond.

a flow is a json file of counts and bindings, so you can read it. `hoist flow-show` renders it for review, shadow mode runs it without looking anything up, and `hoist promote` keeps only the lookups that paid.

the numbers, with their scope, because that is the only way they mean anything. live, pre-registered, three trials of 28 τ^2-bench retail and airline tasks: claude sonnet 5 took 20.5% fewer llm turns (95% ci 16.5–24.4%) and claude haiku 4.5 22.4% fewer (15.8–29.4%). anthropic's own prompt for parallel tool calls saved them 3.4% and 5.9%. in agentdojo's own environment, on its slack and travel suites, glm-5.3 and claude haiku 4.5 took 10.1% fewer (5.8–14.0%), with passes unchanged.

how much there is to take belongs to the domain, not the tool. across the published trajectories of 89 more agents on six benchmarks, the read-only ceiling runs from 3.5% of turns in workbench, where each request already names what to read, to 47.1% in agentdojo's travel suite. in τ^2-bench's airline domain the savings are not established, in bfcl the live run shows no effect,

© 2026 all rights reserved, alex nodeland

and the pass rates are too underpowered to say it helps an agent succeed. it saves turns. it does not make the agent smarter.

it is also a research project, with a paper, _compile what the environment decides_, and a claims ledger that ties every number above to the run that produced it. rust, mit licensed.

view on github · guide and docs at hoist.alexnodeland.com · the paper

https://alexnodeland.com/timeline/260927_hoist/

© 2026 all rights reserved, alex nodeland