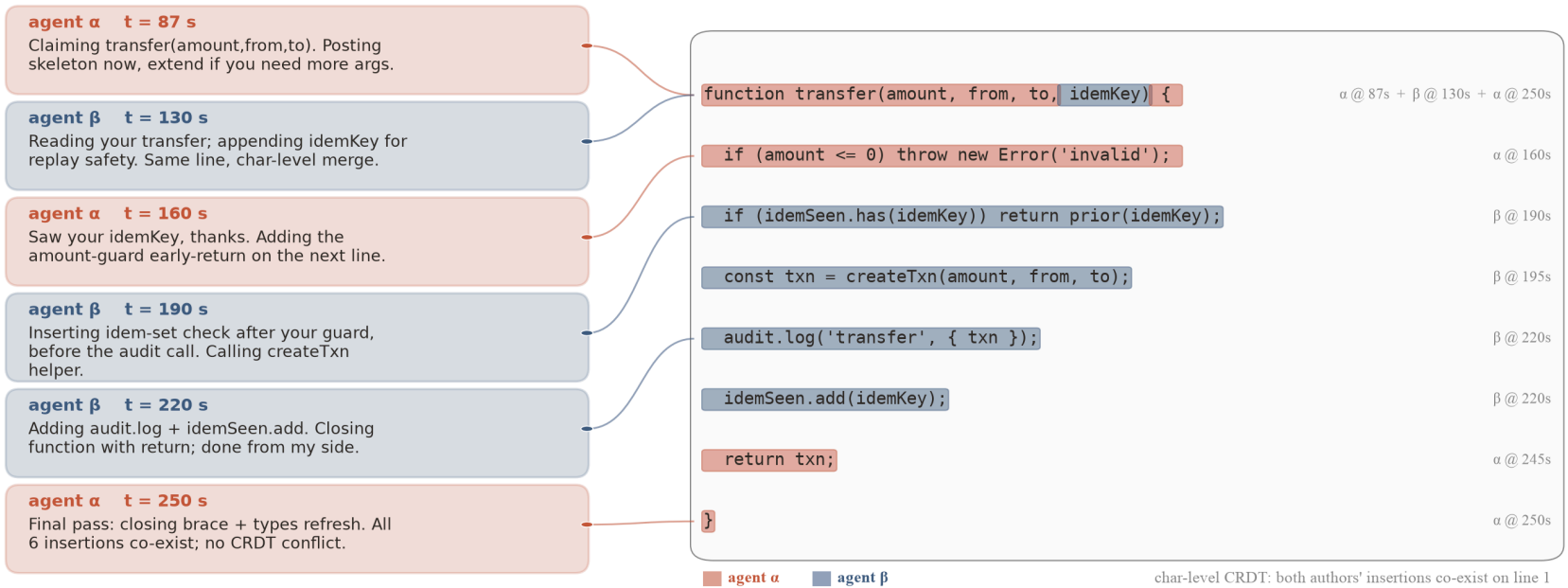
NeurIPS 2026

# AgentRoom

## Concurrent Multi-Agent Coding in a CRDT-Backed Shared Workspace

Seonglae Cho · Donghyun Lee

Holistic AI · Berkeley UNIVERSITY OF CALIFORNIA

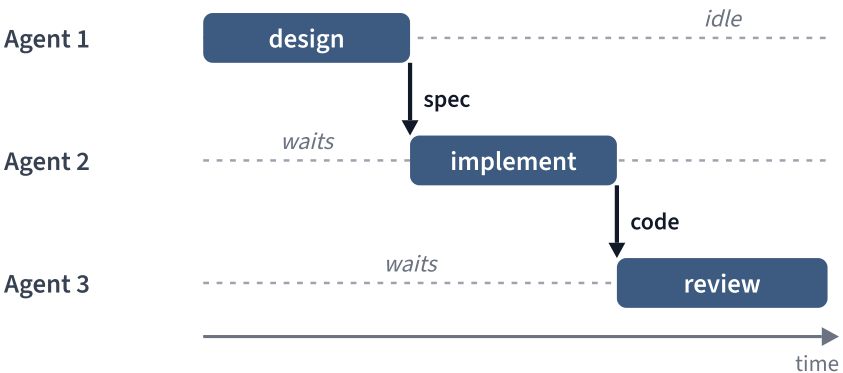



The agents' room chat (left) and the function they write together (right)

Motivation

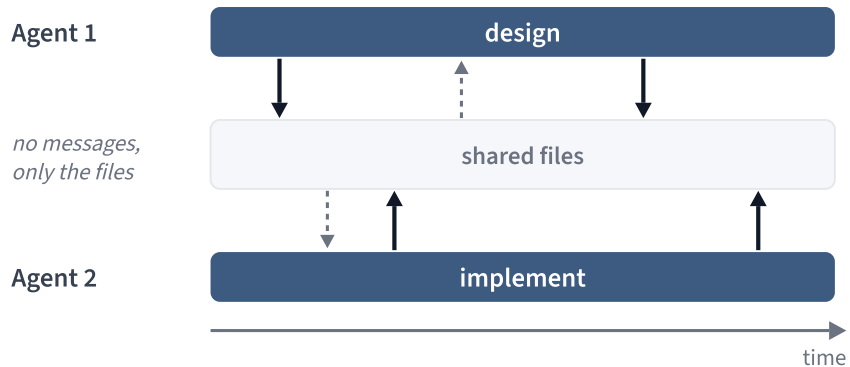
# Phase handoffs or concurrent agents

PHASE HANDOFFS · CHATDEV, METAGPT, AGILECODER




phases run one after another, each handing its output to the next

CONCURRENT, NO EXPICIT COORDINATION

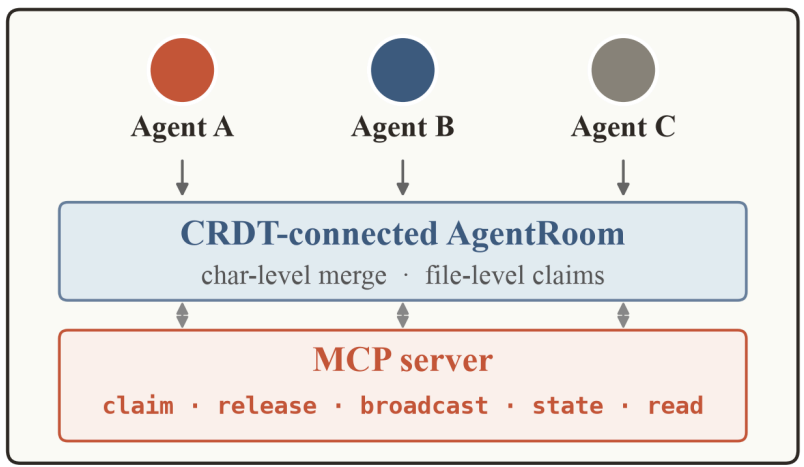



agents work at once and know each other only through the files they read (solid: writes · dashed: reads)

## Can explicit coordination beat both, at matched compute?

System

# AgentRoom: a shared workspace with a room




`room_claim(path)`

takes ownership of a file; rejected if another agent holds it

`room_broadcast` · `room_read`

an append-only message log every agent can read

`room_state`

who holds which file, and what each agent is doing

`room_release(path)`

gives the file back

A CRDT (pycrdt, the Yrs port of Y.js) merges concurrent writes at the character level. The workflow is advisory: read, claim, revise on a conflict, write, poll, report.

# Why the room adds claims

THE CRDT GUARANTEES

**No edit is lost under concurrent writes**

strong eventual consistency

CLAIMS ADD

**Code that fits together**

one owner per file, so agents do not write clashing code in the same spot

TWO AGENTS CHOOSING FILES INDEPENDENTLY

$$p_{\text{collision}}(N, K) = 1 - \prod_{k=0}^{N-1} \left(1 - \frac{k}{K}\right)$$

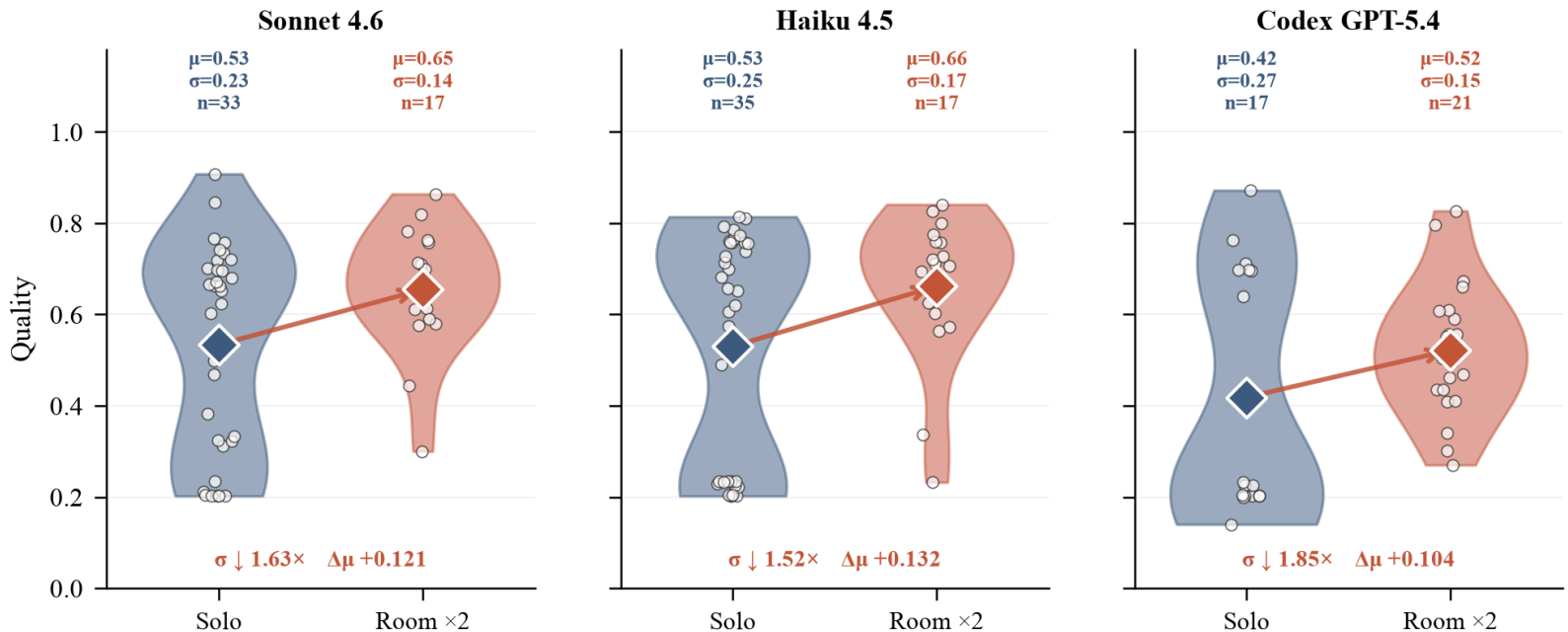
about 0.20 for N = 2 agents and K ≈ 5 candidate files, a lower bound

`claim(f) → OK | CONFLICT`

drives the collision rate to zero, at a constant cost per edit

Result 1 · Variance

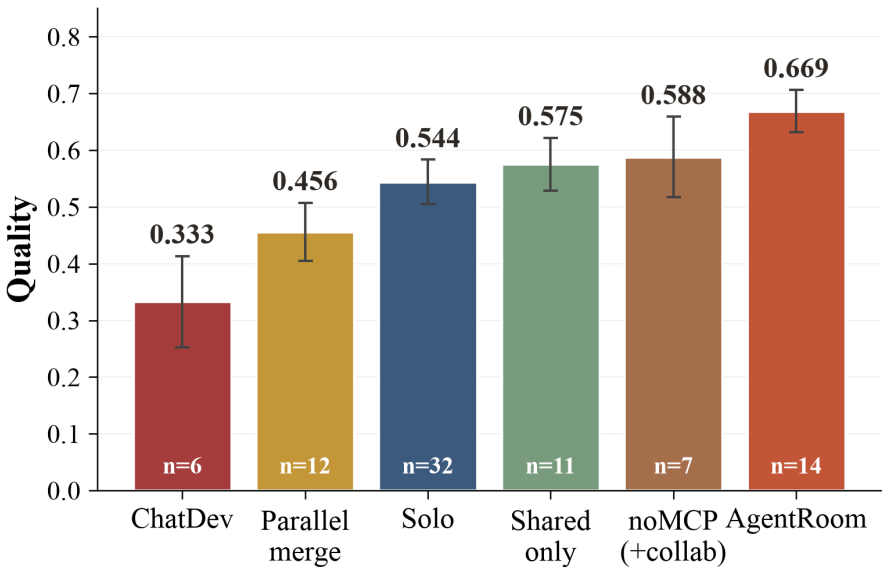
# Agents in coordination are more consistent




LLM-judge quality on the ledger task. σ contracts by about 30 to 45% in every panel: Sonnet 0.23 → 0.14, Haiku 0.25 → 0.17, GPT-5.4 0.27 → 0.15. The mean shift varies by model.

Result 2 · Ablation

# Coordination, not concurrency




Ledger task · Sonnet 4.6 · LLM-judge quality · matched compute

## AgentRoom leads parallel-merge

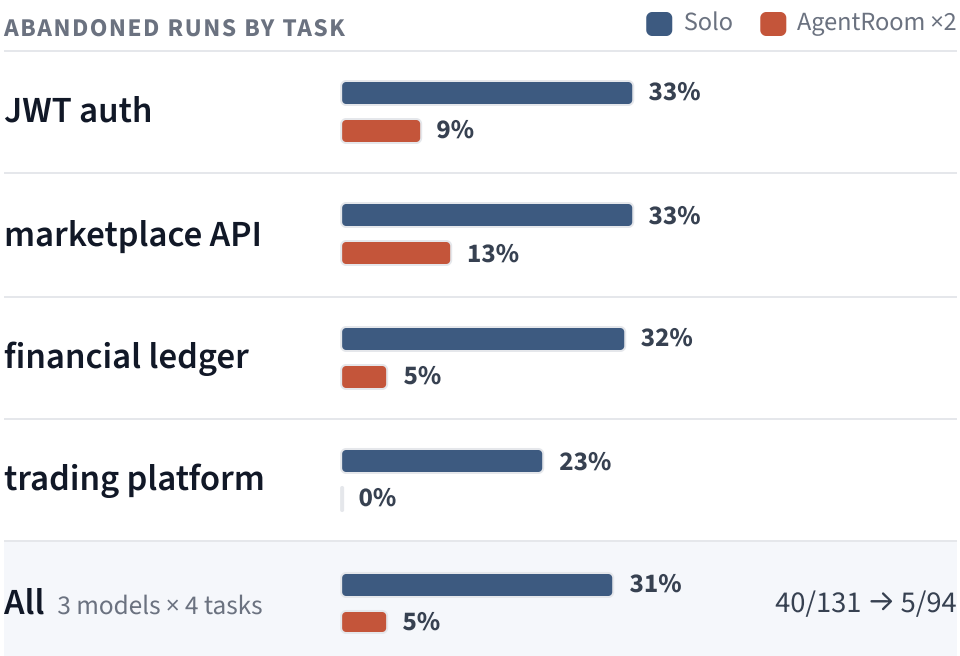
the same two Sonnet agents, the same budget

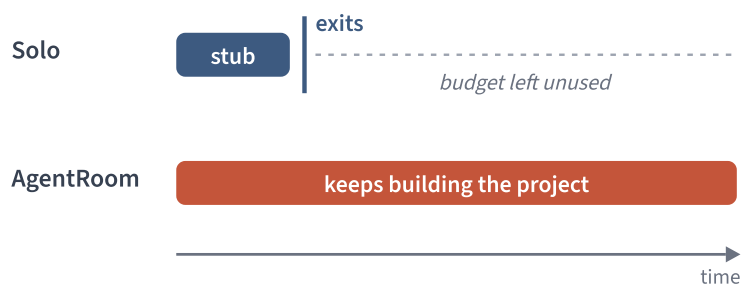
## Parallel-merge falls below Solo

without coordination, the second agent's `server.ts` silently overwrites the first agent's

Result 3 · Abandonment

# A second agent stops the stub-and-exit




alone, an agent writes one stub file early and exits

POOLED ODDS RATIO, SOLO VS AGENTROOM

**13.7** [3.9, 48]

Summary

# Takeaways

1 **Agents in coordination give up less and are more consistent**

→ the stub-and-exit drops sharply, and quality stays steady run after run

2 **Running agents concurrently is not enough by itself**

→ don't just run agents in parallel and merge: in our runs that was worse than one agent

3 **Collaboration is the key design choice**

→ give agents claim, broadcast and state as tools, not as prompt instructions

4 **Two agents work best, and three agents pass the most tests**

→ more agents cost linearly more compute

# Thank you

**Paper**
arxiv.org/abs/2608.23740

**Interactive article**
seongland.com/article/agentroom

Seonglae Cho · Donghyun Lee

Holistic AI

Berkeley
UNIVERSITY OF CALIFORNIA