SCALE WORKSHOP · ICML 2026

The Orchestrator Bottleneck: Formal Coordination Strategies for Cost-Optimal Multi-Agent Enterprise Workflows

Rudrendu Kumar Paul, Boston University

Sourav Nandy, University of Texas at Austin

THE PROBLEM

Multi-agent orchestration strategy is selected once at design time — with no formal analysis of cost-optimality

CURRENT PRACTICE

Production teams select an orchestration strategy once at system design time — sequential, parallel, or hierarchical — and never revisit it. This is treated as an infrastructure configuration choice; cost-optimality analysis plays no role in the selection.

> Given agents with known cost profiles and a task with a dependency structure, which coordination strategy minimizes weighted cost and latency? This is a constrained planning problem — one that formal optimization methods can solve.

WHY THIS MATTERS

Strategy selection is irreversible at inference time. A system locked into sequential delegation on a parallelizable task wastes latency budget. A parallel fan-out on a dependent task wastes token budget. Both failures require full redesign to correct.

The gap: closed-form conditions telling practitioners when each strategy is optimal — as a function of task dependency structure — remain absent from the literature.

FORMAL MODEL

Orchestrator as a constrained optimization problem over a typed agent registry

ICML International Conference On Machine Learning

SYSTEM DEFINITION

The orchestrator is formalized over a **typed agent registry** — each agent carries capability declarations, cost profiles, and trust boundaries. Strategy selection is a first-class planning decision that must occur before any subtask executes.

> **Result:** Closed-form threshold conditions derived for when each strategy is optimal, expressed as a function of task dependency depth, agent cost ratios, and parallelism degree.

THREE COORDINATION STRATEGIES

STRATEGY 1

Sequential Delegation

Tasks execute in dependency order, one at a time. Optimal when subtask outputs are strongly coupled and intermediate results gate downstream context.

STRATEGY 2

Parallel Fan-Out

Independent subtasks execute simultaneously. Optimal when the dependency graph is shallow and subtask parallelism is high.

STRATEGY 3

Hierarchical Decomposition

Sub-orchestrators manage clusters of dependent subtasks in parallel. Optimal when the task graph has separable, internally-dependent subgraphs.

SYSTEM

OrchestRAte: runtime strategy selection without LLM overhead — 28–42% cost reduction

ICML International Conference On Machine Learning

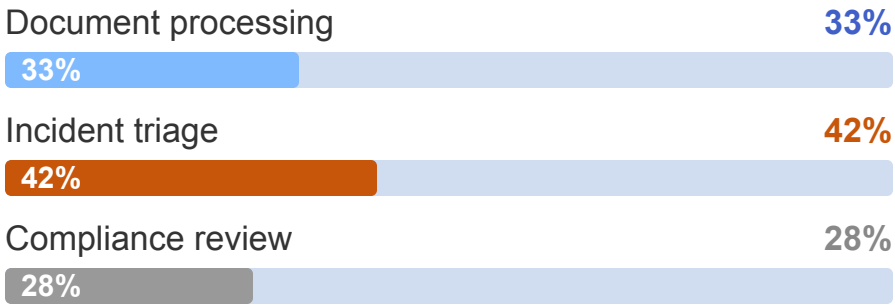
HOW ORCHESTRATE WORKS

OrchestRAte selects coordination mode dynamically using lightweight dependency graph analysis at the start of each inference request. It applies the closed-form threshold conditions from the formal model; the full token budget goes to task execution, with planning handled entirely by the graph analysis step.

BENCHMARK RESULTS

Workflow	Task type	Cost reduction	Quality delta
Document processing	Parallel-heavy	33%	−0.6 pp
Compliance review	Sequential-heavy	28%	−0.3 pp
Incident triage	Highly parallelizable	42%	−0.1 pp

COST REDUCTION VS. STATIC SEQUENTIAL BASELINE



Key result: 28–42% cost reduction vs. static sequential baselines, with at most **0.6 percentage points quality loss** across all three benchmarks.

Baseline: static sequential delegation. Quality measured by task-specific eval rubric (F1 for extraction, correctness rate for compliance, resolution accuracy for triage).

FINDINGS

Three empirical findings with direct design implications for production multi-agent systems

15%
orchestrator overhead threshold — beyond this, coordination cost erases planning advantage

<1%
request latency cost of dependency graph analysis at runtime

42%
peak cost reduction on highly-parallelizable workflows (incident triage)

1 **Orchestrator overhead threshold at ~15% of total token budget.** Beyond this point, coordination cost erases the planning advantage. Systems that exceed this threshold in static-sequential mode pay a coordination tax on every request while quality holds flat.

2 **On highly parallelizable tasks, static parallel baseline still achieves lowest raw latency.** Dynamic strategy selection adds a small overhead; for latency-critical applications with fully independent subtasks, parallel fan-out without selection logic is the correct default.

3 **The right strategy depends on task dependency structure — and that structure can be analyzed at runtime.** Dependency graph analysis at inference time costs less than 1% of total request latency across all three benchmarks, making dynamic selection practical at production scale.

CONTRIBUTIONS & TAKEAWAY

Formal planning conditions for strategy selection: the orchestrator bottleneck is solvable at runtime

ICML International Conference On Machine Learning

CONTRIBUTIONS

- Formal model of orchestrator as a **constrained planner** over typed agent registries with capability declarations, cost profiles, and trust boundaries
- Closed-form optimality conditions for all three coordination strategies as a function of task dependency structure
- **OrchestRAte system**: 28–42% cost reduction vs. static sequential baselines across three enterprise workflow benchmarks
- Orchestrator overhead threshold at **15% of total token budget** — a practical design rule for production systems

CENTRAL ARGUMENT

Multi-agent coordination is a **planning problem with formal solutions**. The choice between sequential, parallel, and hierarchical orchestration is a function of task dependency structure that can be computed at runtime.

TAKEAWAY

Treating strategy selection as a fixed design-time decision carries measurable cost consequences. Match coordination mode to task dependency structure — the formal conditions make this computable.