

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Wenbo: Scaling Agent Reasoning via Dynamic DAG Scheduling

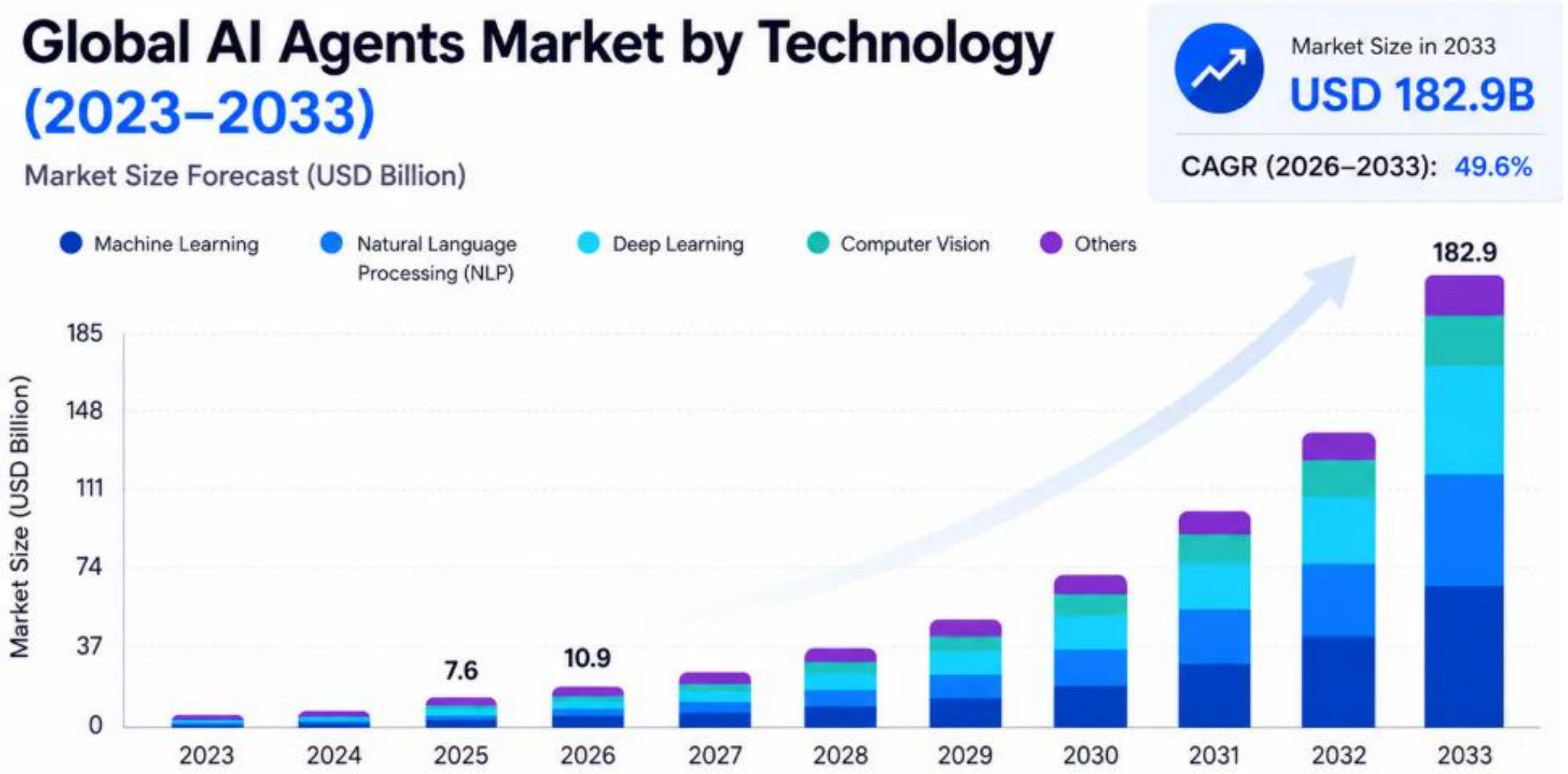
Feiyang Chen · Junqing Yang · Wentao Zhang · Zheng Wang* · Li Pan · Jianhua Li

Shanghai Jiao Tong University · New York University

Project: www.github.com/wenbo-project/wenbo

Preprints 2026 · DOI: 10.20944/preprints202611.0942.v1

The Trends of Agentic AI



Source: https://www.grandviewresearch.com/industry-analysis/enterprise-agentic-ai-market-report

Overview of Performance: Wenbo vs Flash-Searcher

1 REAL-WORLD DATASETS · ARGB-R

Four task domains: StrategyQA · MuSiQue · ToolBench · ALFWorld

1.4–3.8x
higher throughput
Across four real-world workloads

1.36–2.71x
GPU utilization gain
With 2 GPUs; CPU count varied

2.3–26.0x
lower P99 first-service wait
Mixed-priority foreground requests

1.55x
P99 response-time speedup
Average across four workloads

Takeaway: node-level sharing keeps heterogeneous resources busy and improves both first-service latency and end-to-end response.

2 SYNTHETIC DATASETS · ARGB-S

Controlled DAG topology, resource mix, and task concurrency

CPU-HEAVY 9x → 99x → + ∞ (upper bound)

Throughput speedup as the number of concurrent DAGs increases.

BALANCED 1.1x →1.75x → 2x (upper bound)

Smaller but consistent gains when workload demand better matches device capacity.

Takeaway: task complexity impact is the same as the above results

[1] Qin, Tianrui, et al. "Flash-searcher: Fast and effective web agents via dag-based parallel execution." ICLR 2026.

Intuition: Different Agent Applications Share One Graph Abstraction

RESEARCH / QA

Retrieve — Reason — Answer

TOOL-USING AGENT

Plan — API call — Verify

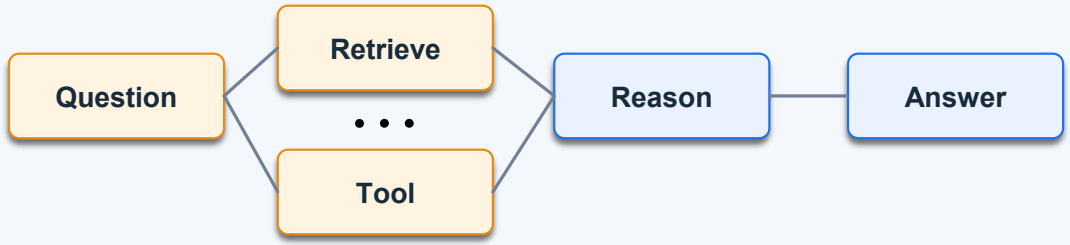
EMBODIED AGENT

Observe — Act — Check

COMMON REASONING GRAPH

Nodes = atomic computation steps

Edges = prerequisite dependencies



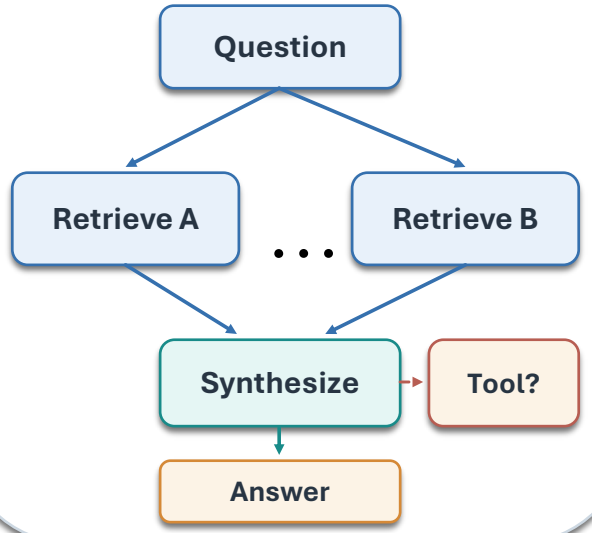
One graph model spans QA, tools, and embodied interaction.

Idea: Turn Dynamic Reasoning into Global DAG Execution

1 DYNAMIC REASONING GRAPH

One reasoning graph

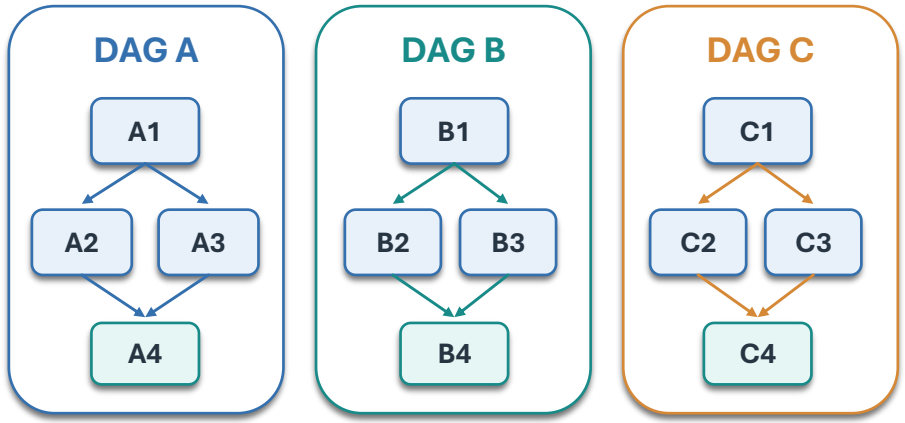
Future steps may depend on intermediate results



2 DIRECTOR: INCREMENTAL MATERIALIZATION

Materialize only determined executable flows

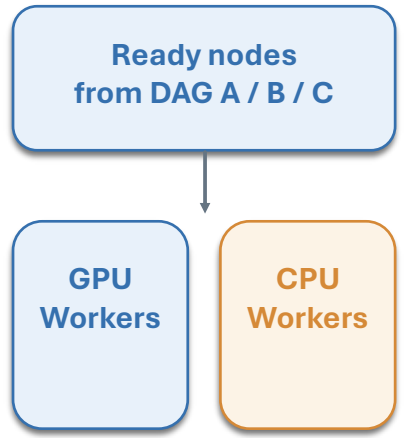
Dependency structure is explicit within each DAG



Director updates flows as intermediate results arrive

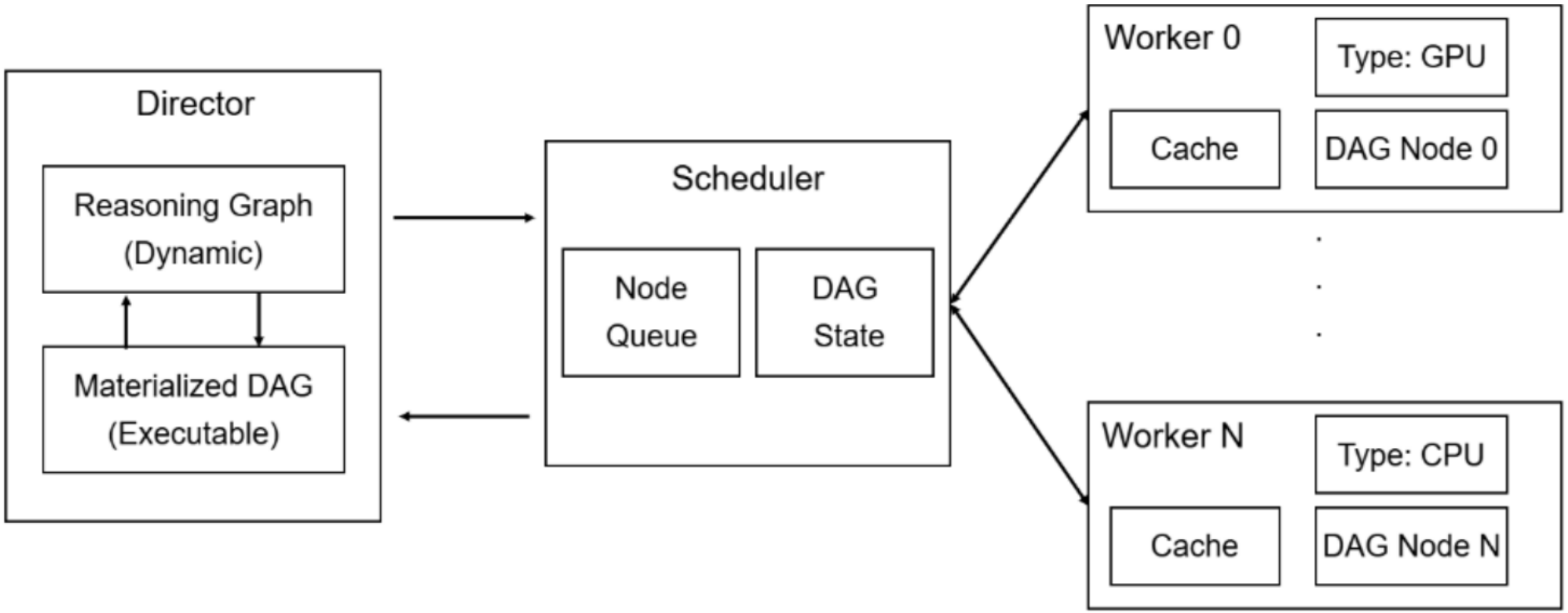
3 GLOBAL SCHEDULER

Dependency-aware queue



Dispatch by dependency
+ resource requirement

Wenbo System Overview



- Director evolves reasoning graphs and emits executable computation plans;
- Scheduler globally schedules dependency-aware nodes across heterogeneous workers;
- Workers execute individual nodes and return runtime feedback.

Project page: https://github.com/wenbo-project/wenbo

Benchmark for Large-scale Agent Reasoning

- Agent Reasoning Graph Benchmark (ARGB)
  - ARGB-R, derived from public agent task datasets.
  - ARGB-S, generated using a parameterized synthetic workload generator.

Dataset	Task Type	Graphs	Nodes	Edges	Avg. Depth	GPU/CPU Ratio
StrategyQA	Factual reasoning and Boolean QA	2,061	7,642	6,090	2.67	73.6% / 26.4%
MuSiQue	Multi-hop evidence reasoning	19,938	95,044	85,668	4.25	57.3% / 42.7%
ToolBench	Tool selection and invocation	26,999	124,408	131,936	3.20	41.1% / 58.9%
ALFWorld	Embodied planning and interaction	6,374	76,455	79,065	10.69	44.4% / 55.6%
Total	Four public task sources	55,372	303,549	302,759	4.42	47.8% / 52.2%

Table: Summary of the ARGB-R Benchmark

ARGB-R: Experiment on Throughput

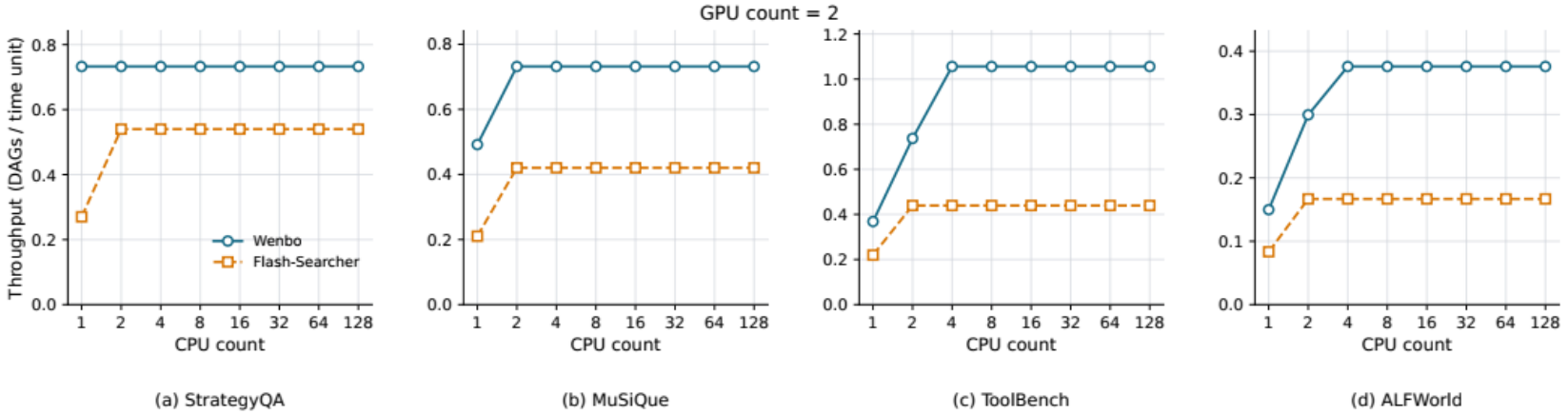


Figure 3: Throughput scaling with CPU resources on ARGB-R, with 2 GPUs fixed.

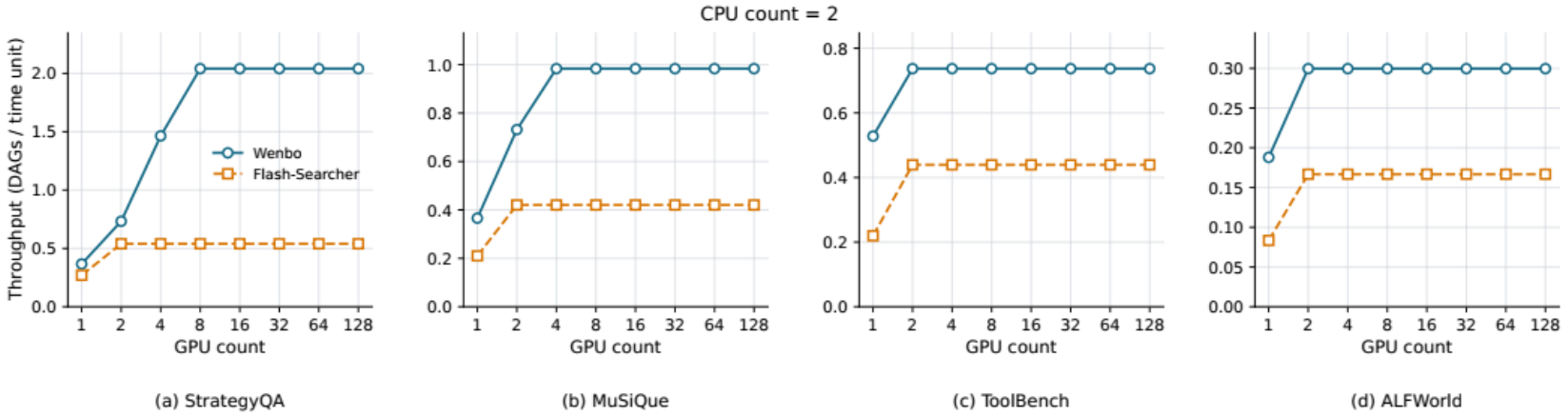


Figure 4: Throughput scaling with GPU resources on ARGB-R, with 2 CPUs fixed.

1.4–3.8×

higher throughput

Why it works

Ready nodes from different DAGs can use whichever device is idle, instead of waiting behind a task's reserved pair.

The largest gains appear when the device pool and the workload composition are both skewed in the same direction.

ARGB-R: Experiment on Device Utilization

Table 2: GPU utilization (%) on ARGB-R with the number of GPUs fixed at 2.

#CPUs	StrategyQA			MuSiQue			ToolBench			ALFWorld		
	Flash-Searcher	Wenbo	Gain↑	Flash-Searcher	Wenbo	Gain↑	Flash-Searcher	Wenbo	Gain↑	Flash-Searcher	Wenbo	Gain↑
1	36.833	99.940	2.71×	28.724	67.200	2.34×	20.803	34.894	1.68×	22.180	39.847	1.80×
2	73.650	99.940	1.36×	57.447	99.994	1.74×	41.604	69.785	1.68×	44.356	79.684	1.80×
4	73.650	99.940	1.36×	57.447	99.994	1.74×	41.604	99.994	2.40×	44.356	99.958	2.25×
8	73.650	99.940	1.36×	57.447	99.994	1.74×	41.604	99.994	2.40×	44.356	99.959	2.25×
16	73.650	99.940	1.36×	57.447	99.994	1.74×	41.604	99.994	2.40×	44.356	99.959	2.25×
32	73.650	99.940	1.36×	57.447	99.994	1.74×	41.604	99.994	2.40×	44.356	99.959	2.25×
64	73.650	99.940	1.36×	57.447	99.994	1.74×	41.604	99.994	2.40×	44.356	99.959	2.25×
128	73.650	99.940	1.36×	57.447	99.994	1.74×	41.604	99.994	2.40×	44.356	99.959	2.25×

GPU Utilization improvement ratios ranging from 1.36x to 2.71x.

ARGB-R: Priority Scheduling under Mixed Workloads (I)

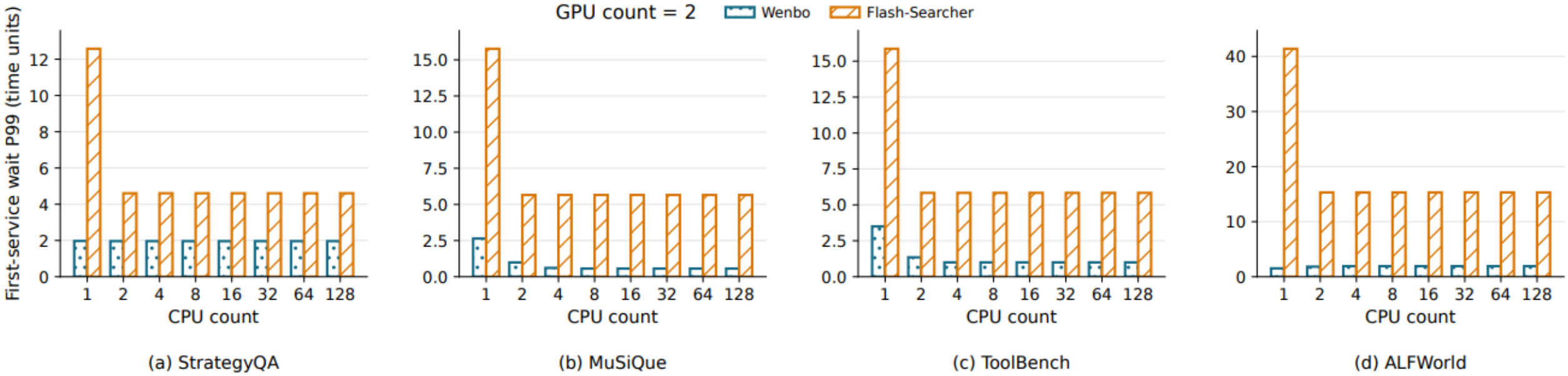


Figure 5: P99 first-service waiting time on ARGB-R as the number of CPUs varies, with the number of GPUs fixed at 2.

ARGB-R: Priority Scheduling under Mixed Workloads (II)

Table 3: P99 foreground response time (units) in the same-source mixed-workload experiment on ARGB-R, with 2 GPUs, 8 CPUs, and 16 outstanding background requests.

Dataset	Foreground P99 Response Time		Speedup↑
	Flash-Searcher	Wenbo	
StrategyQA	8.892	5.911	1.50×
MuSiQue	10.881	7.035	1.55×
ToolBench	11.458	7.251	1.58×
ALFWorld	29.631	18.837	1.57×

First serve waiting time

2.3× → 30×

P99 foreground response time

~1.55×

ARGB-S: Throughput as DAG Number Grows

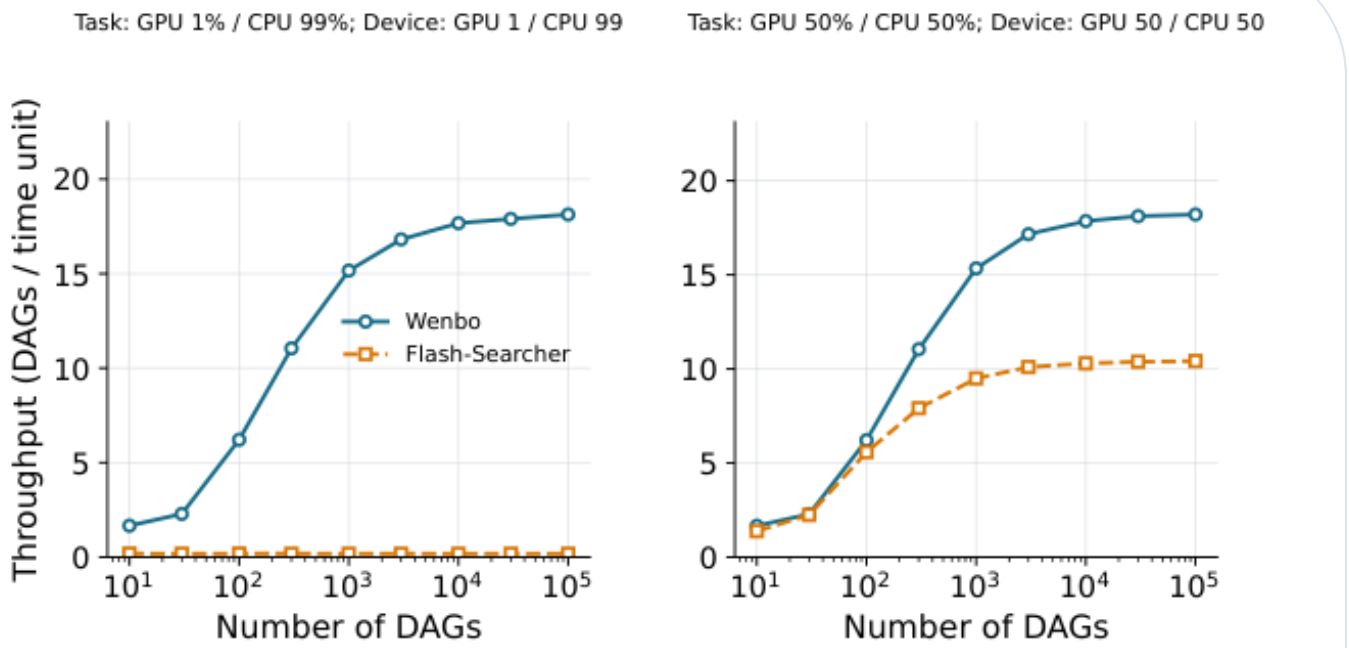


Figure 6: Throughput as the DAG number varies on ARGB-S.

CPU-HEAVY

9× → 99×

Throughput speedup as DAG count grows

BALANCED

1.1× → 1.75×

Smaller but consistent gains when task demand better matches device supply.

ARGB-S varies the number of independent DAGs while keeping device and workload ratios controlled.

ARGB-S: Throughput as DAG Size/Complexity Varies

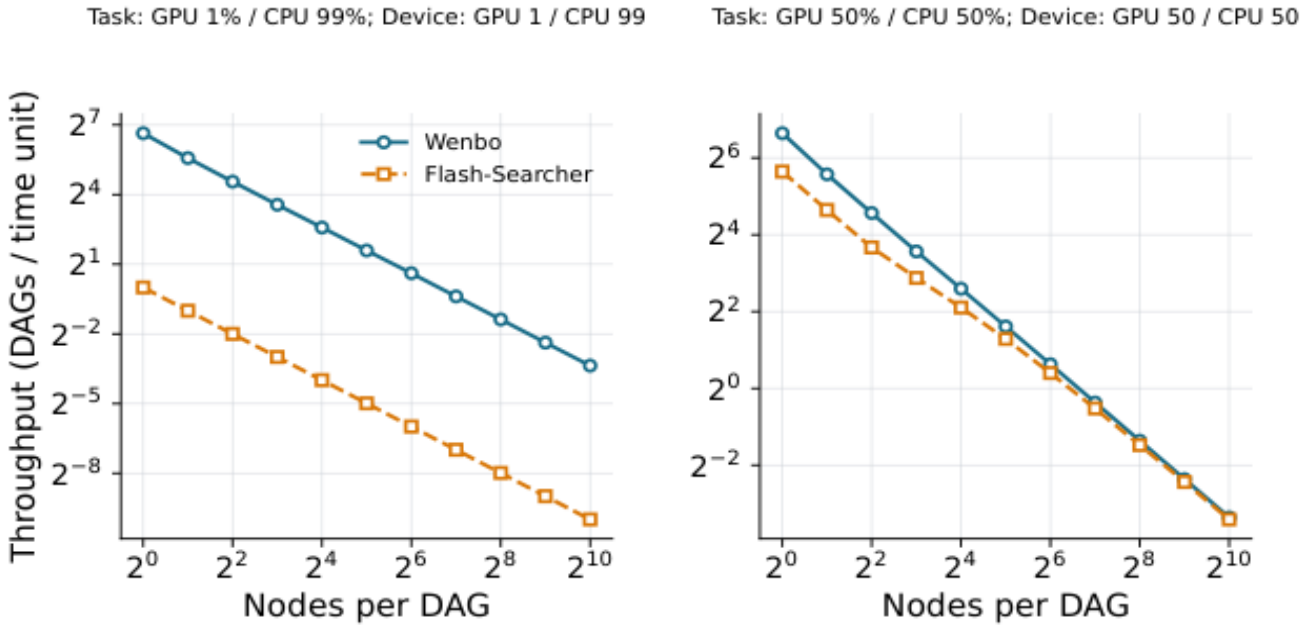


Figure 7: Throughput as the DAG size varies on ARGB-S.

The paper fixes 1,000 inference tasks and varies nodes per DAG.

CPU-HEAVY

~99×

Wenbo maintains an approximately 99× throughput advantage as DAG size increases.

BALANCED

Gap narrows ~1.5x

Larger DAGs expose more executable nodes, allowing Flash-Searcher to saturate the devices and reducing Wenbo's marginal benefit.

THEORETICAL ANALYSIS

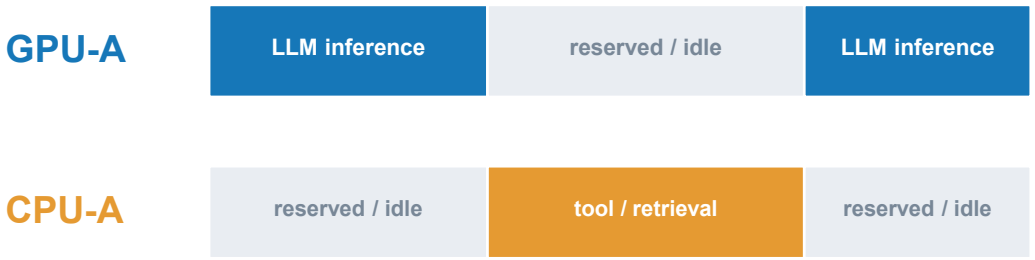


The Bottleneck is Resource Ownership at Task Granularity

TASK-LEVEL RESERVATION

One task owns a GPU + CPU pair

Any phase that is not using a device still holds it.



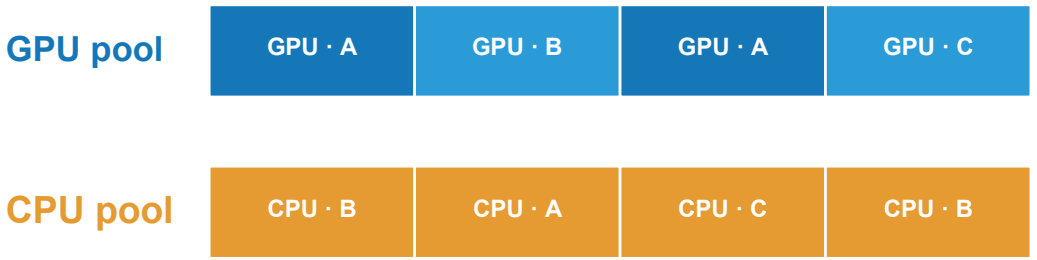
Fixed pair remains held until the DAG completes

Independent tasks cannot freely borrow the idle device.

NODE-LEVEL SHARING (Ours)

Ready nodes from many DAGs share a global pool

The Scheduler dispatches work by node type and readiness.



Idle capacity can be filled by another DAG

The unit of dispatch is a ready computation node, not a task.

Throughput Improvement Factor

THEOREM 3 (THROUGHPUT IMPROVEMENT FACTOR). *Let* $\lambda = \Theta_{node}/\Theta_{task}$ *be the throughput improvement factor of Node-level parallelism over task-level parallelism. Then*

$$\lambda_{min} \leq \lambda \leq \lambda_{max}, \tag{4}$$

where

$$\lambda_{min} = \frac{\max(\tau_{GPU}, \tau_{CPU})}{\min(N_{GPU}, N_{CPU})} \min\left(\frac{N_{GPU}}{\tau_{GPU}}, \frac{N_{CPU}}{\tau_{CPU}}\right), \tag{5}$$

$$\lambda_{max} = \frac{\tau_{GPU} + \tau_{CPU}}{\min(N_{GPU}, N_{CPU})} \min\left(\frac{N_{GPU}}{\tau_{GPU}}, \frac{N_{CPU}}{\tau_{CPU}}\right). \tag{6}$$

Throughput gains is the alignment between resource supply and workload demand: when both imbalances favor the same resource type, node-level scheduling can substantially improve the throughput.

Throughput Gain Visualization

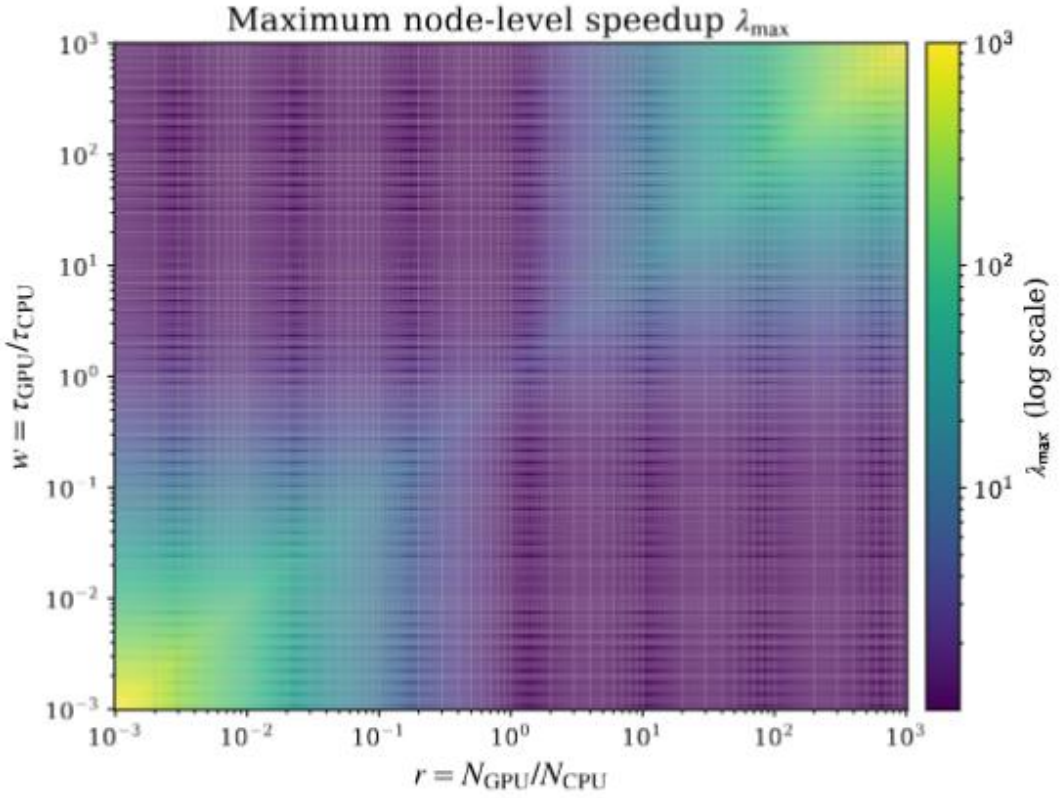


Figure 8: Maximum node-level speedup λ_{max} as a function of resource ratio r and workload ratio w. The heatmap uses a logarithmic color scale. This corresponds to fully serial intra-task phases.

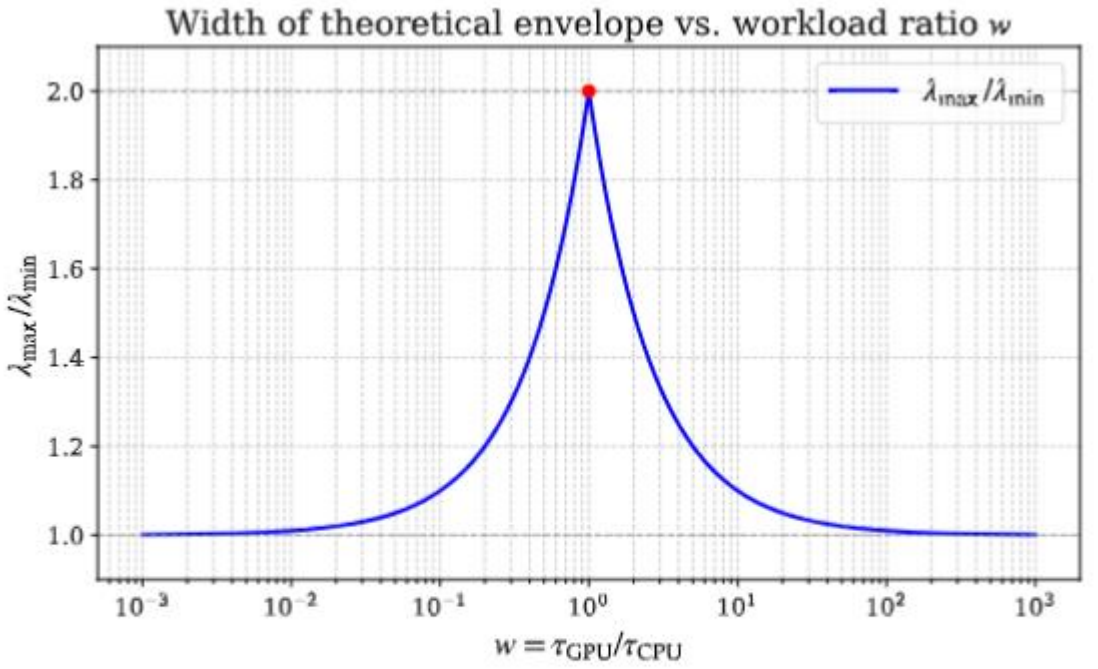


Figure 9: Ratio $\lambda_{max}/\lambda_{min}$ versus workload ratio w. The ratio lies in the narrow interval $[1, 2]$, reaching its maximum at $w = 1$. This ratio captures the maximum effect of intra-task phase overlap on speedup.

Quick Throughput Gain Estimator

- Case 1: Device Counts and Task Workloads Are Known

  Example: with 1 GPU, 1,024 CPUs, and per-task workloads of 1 GPU unit and 64 CPU units, the estimated gain is 64×.

$$\lambda \approx \begin{cases} \min\left(\frac{N_{GPU}}{N_{CPU}}, \frac{\tau_{GPU}}{\tau_{CPU}}\right), & N_{GPU} \geq N_{CPU}, \\ & \tau_{GPU} \geq \tau_{CPU}, \\ \min\left(\frac{N_{CPU}}{N_{GPU}}, \frac{\tau_{CPU}}{\tau_{GPU}}\right), & N_{GPU} \leq N_{CPU}, \\ & \tau_{GPU} \leq \tau_{CPU}, \\ 1, & \text{otherwise.} \end{cases}$$

- Case 2: Only Device Counts Are Known.

$$1 \leq \lambda \leq \frac{N_{GPU} + N_{CPU}}{\min(N_{GPU}, N_{CPU})} = 1 + \frac{\max(N_{GPU}, N_{CPU})}{\min(N_{GPU}, N_{CPU})}.$$

Example: with 1 GPU and 1,024 CPUs, the potential speedup can reach 1,024x.

- Case 3: Only Workload Composition Is Known.

$$\frac{N_{GPU}}{N_{CPU}} = \frac{\tau_{GPU}}{\tau_{CPU}} \quad \Longrightarrow \quad \lambda \approx \frac{\max(\tau_{GPU}, \tau_{CPU})}{\min(\tau_{GPU}, \tau_{CPU})}.$$

Example: If GPU and CPU workloads account for 20% and 80%, set GPU/CPU resource to 1:4, to get the max throughput gain 4x.



Thank you — questions and feedback are welcome.

- Teacher: Zheng Wang
- Contact: wzheng@sjtu.edu.cn
- Phone: 13701249357
- Affiliation: School of Computer Science, STJU