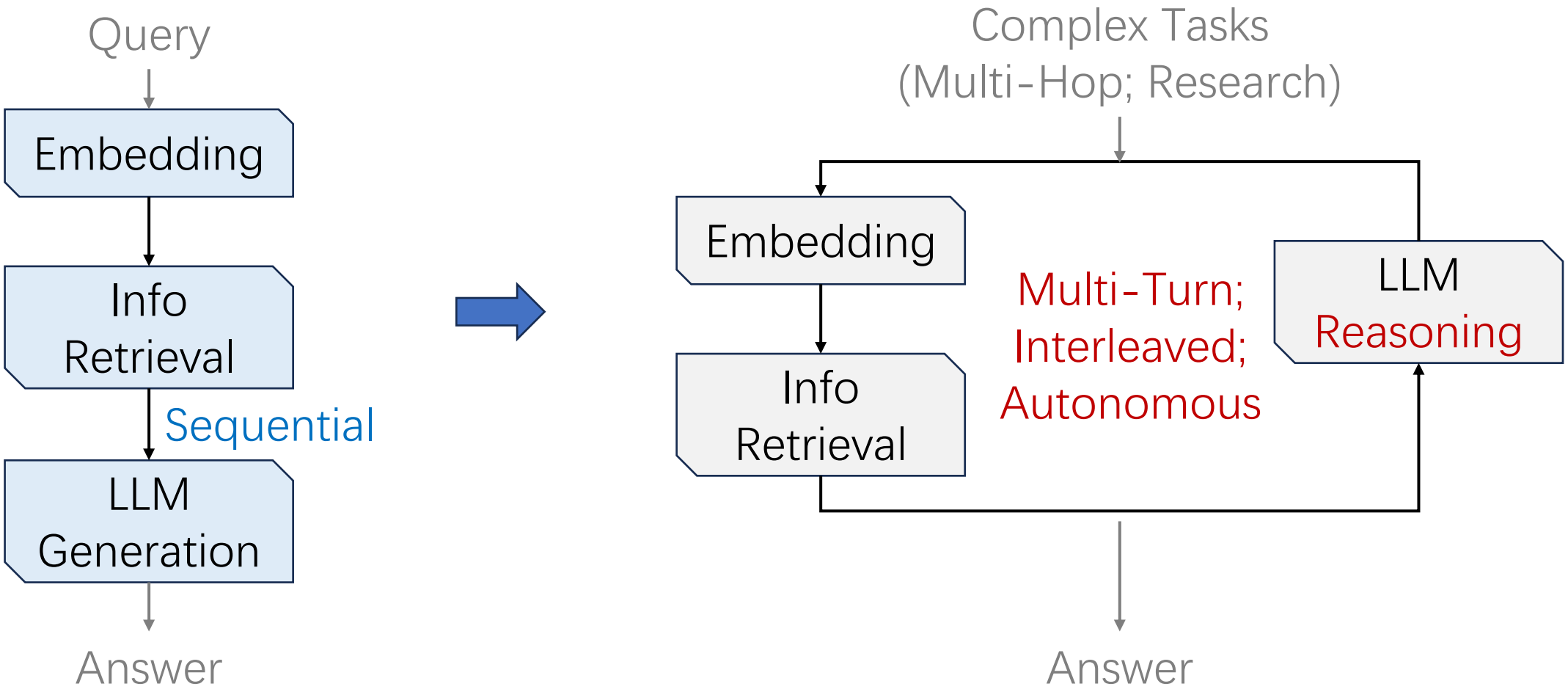
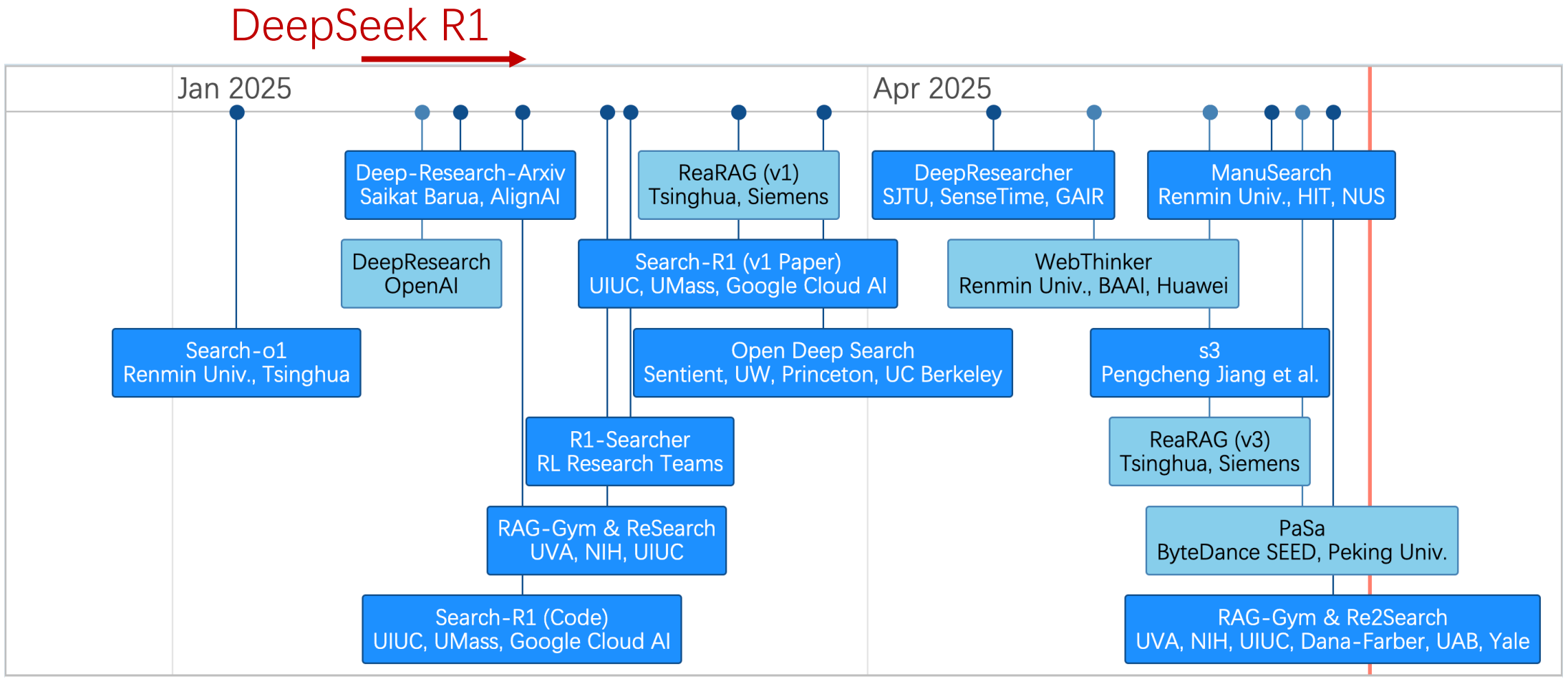
# Towards Efficient LLM Search Agents

Tiannuo Yang
Nankai University, University of Southern California

# From RAG to Search Agents



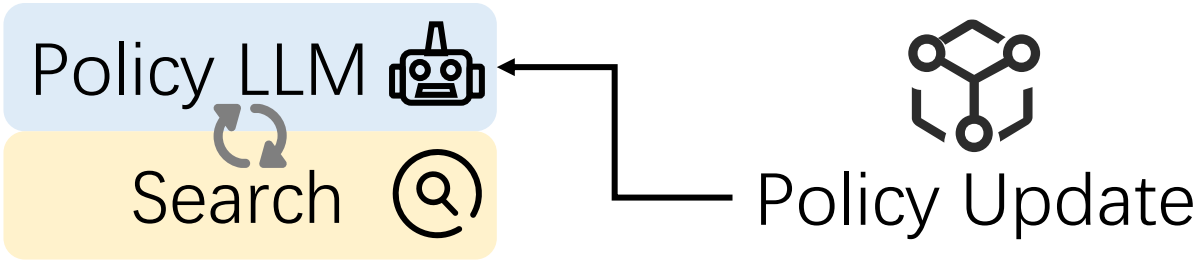
# Search Agents: Deep interactive retrieval and LLM reasoning make search intelligence!



# But it comes at severe efficiency cost

Practical Deployments:

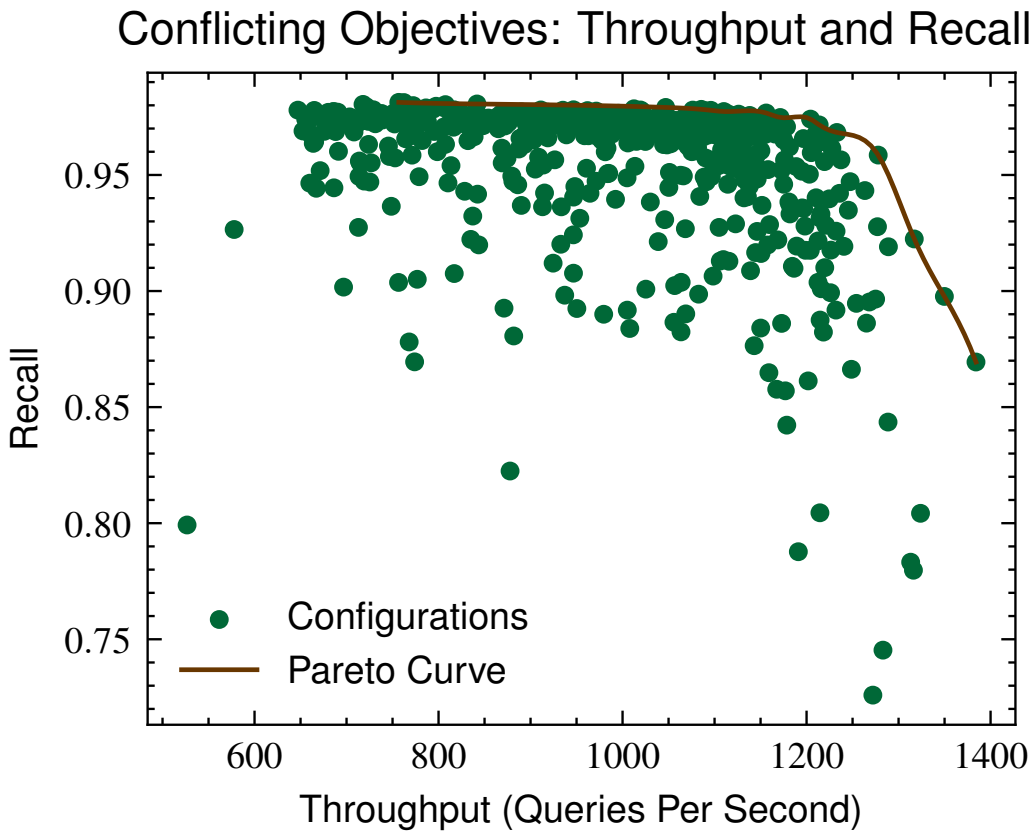
- **Post-Training** (via RL): time-consuming, multi-turn LLM rollouts
- **Serving**: expecting low request latency, high throughput agents




*What's the **bottlenecks** of **system efficiency**?*
*From what **aspects** to optimize?*

# Challenge #1: Retrieval faces inherent, conflicting accuracy-speed tradeoffs

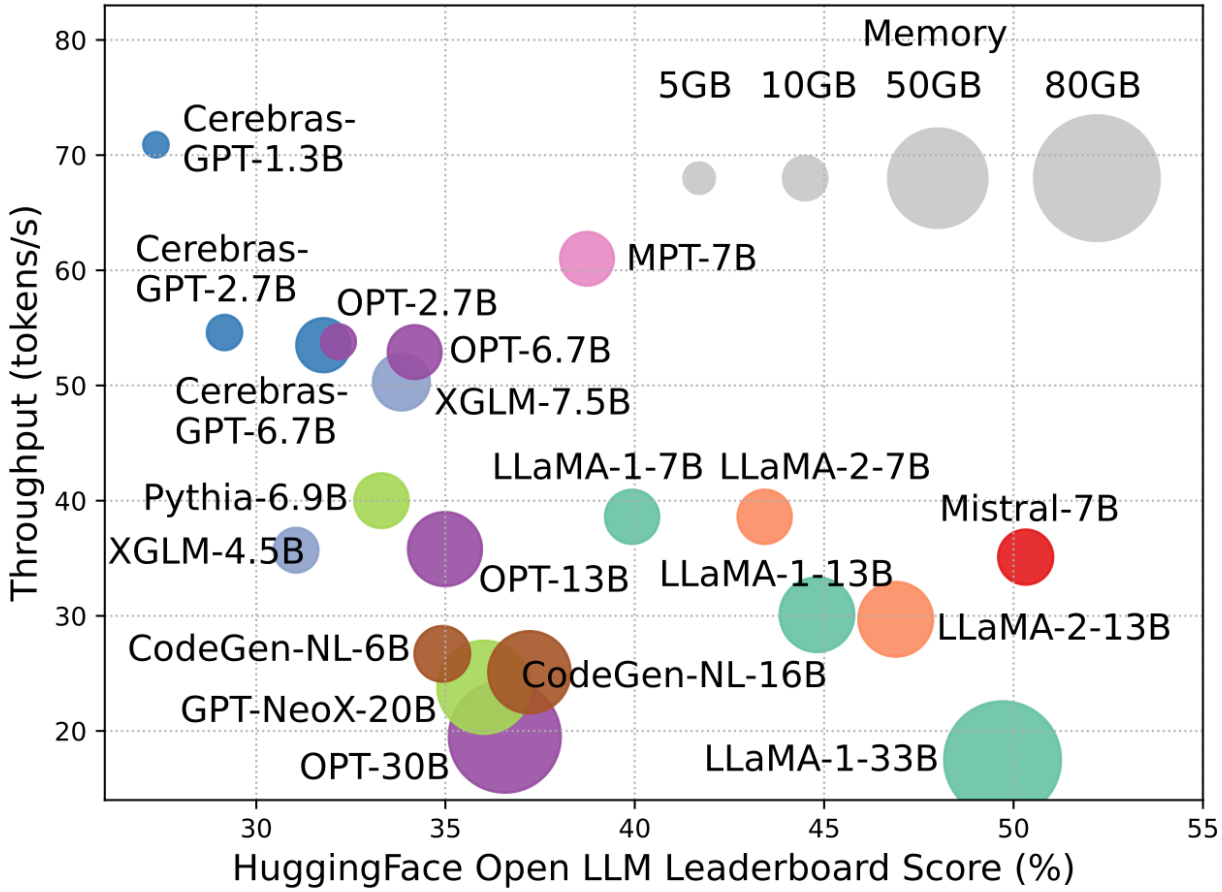
- Keyword Search
  - \+ Fast
  - – No semantics
- Exact Nearest Neighbor (ENN)
  - \+ Semantics; Full Recall
  - – Slow
- Approximate Nearest Neighbor (ANN)
  - \+ High Recall; Fast
  - – Accuracy-Speed Tradeoffs



# Challenge #2: LLM inference speed is critical to cost-effectiveness & user experiences

- Better inference quality?
  - More Resources
  - Slower Inference

- Fig from "Efficient Large Language Models: A Survey" [TMLR'2024]



# Challenge #3: End-to-end search agent systems are even more complex

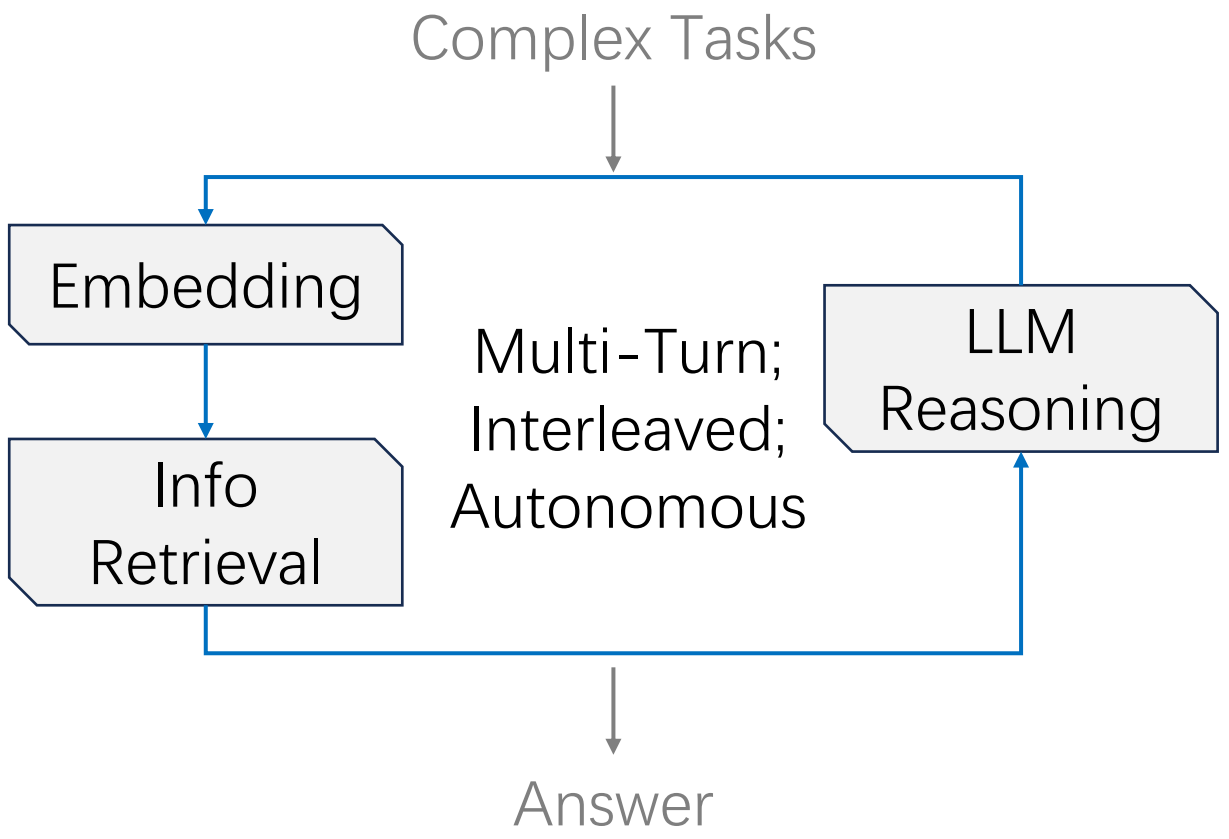
- Longer reasoning steps
- Interleaved, dynamic retrievals

Compared with naïve RAG:

→Complex request status

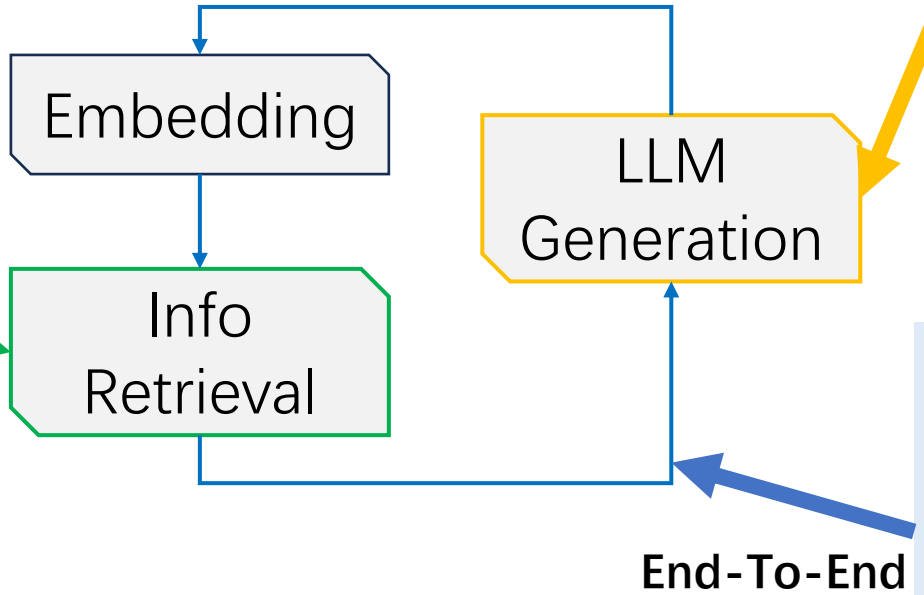
→Unclear retrieval impacts

**Need to dive deep!**



# This Talk: System Efficiency for Search Agents

**Auto-Tuning Vector Database**
For Conflicting Obj. & Multiple Indexes
[ICDE 2024, Deployed on CodeFuse]




**Auto-Tuning LLM Inference Engine**
SLO-Oriented
[WWW 2025, Oral]

★ Key Focus

**Demystifying and Enhancing Search Agent Efficiency**
Targeting Interleaved Search and LLM Reasoning
[ICLR 2026, NeurIPS ER Workshop Spotlight 2025]

# This Talk: System Efficiency for Search Agents

**Auto-Tuning Vector Database**
For Conflicting Obj. & Multiple Indexes
[ICDE 2024, CodeFuse]


# VDTuner: Automated Performance Tuning for Vector Data Management Systems

With Wen Hu, Wangqi Peng, Yusen Li, Jianguo Li, Gang Wang, Xiaoguang Liu



40th IEEE International Conference on
Data Engineering
Utrecht, Netherlands | 13th - 17th May

# Vector database is essential in search agents

Popular practice of info retrieval: Vector database (VDB)

- Ensemble of multiple ANNs
- Storage and real-time retrieval of millions to billions of vectors. E.g., Milvus, Qdrant

***Our Problem: How to optimize (auto-tune) VDB performance?***

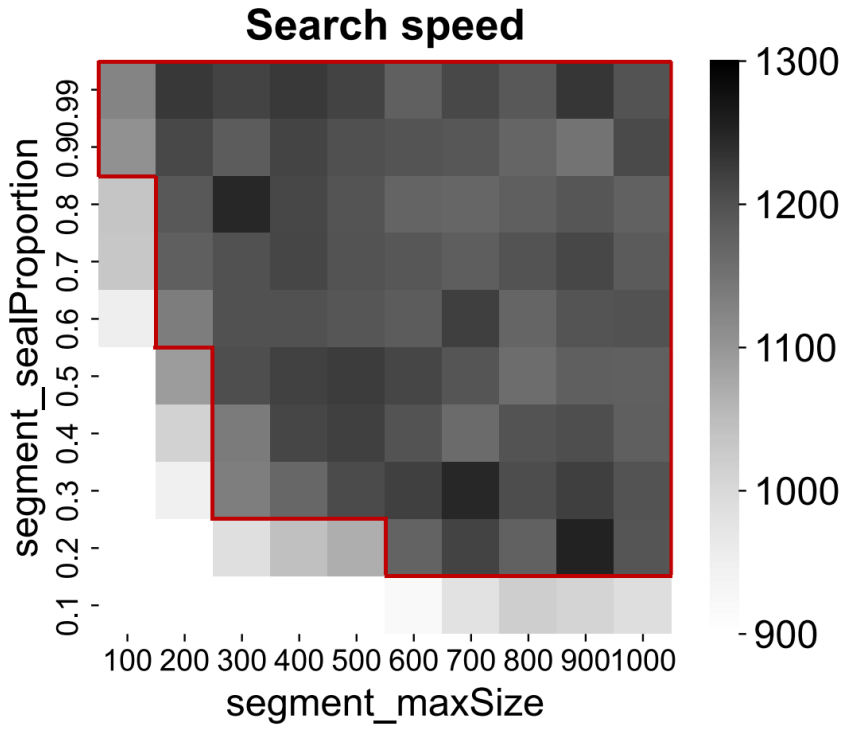
Auto-tuning: Finding the best parameter configuration through autonomous, iterative "trial and error".

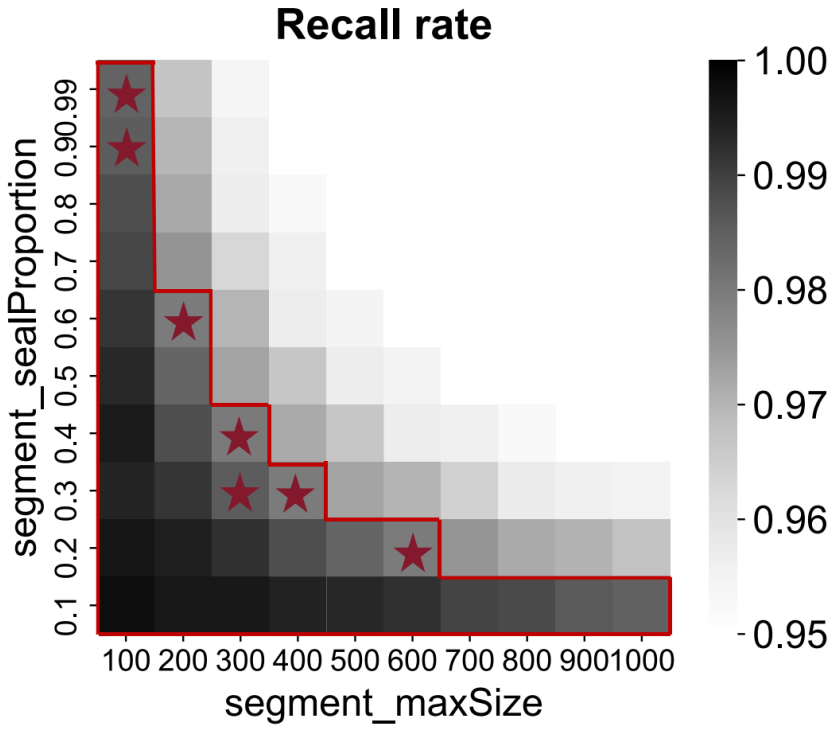
# Autotuning VDB is beneficial but challenging

- What makes it different from traditional DBs/Systems?
  - Index & System Knobs: Inter-dependent, multi-dimensional knobs
- Approximate Search
  - Two metrics: search speed & recall rate
  - They are CONFLICTING!
- Multiple Index Types
  - Cluster-based; Graph-based; Quantization…
  - Different Index has partially DIFFERENT KNOBS!

# Challenge #1: Complex Search Space; Conflicting Objectives

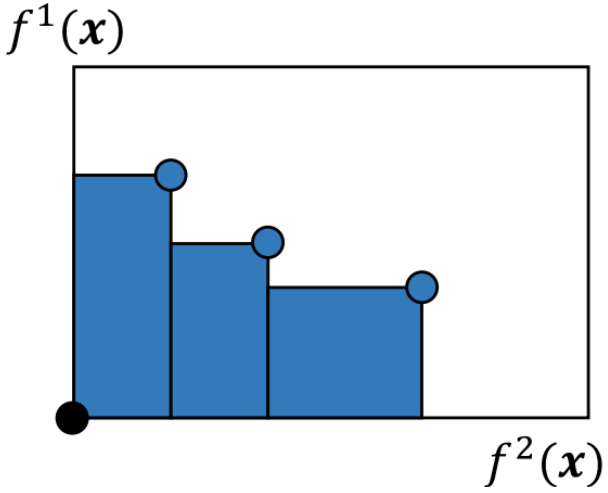
- Must tunes all knobs together
- Hard to balance speed and accuracy



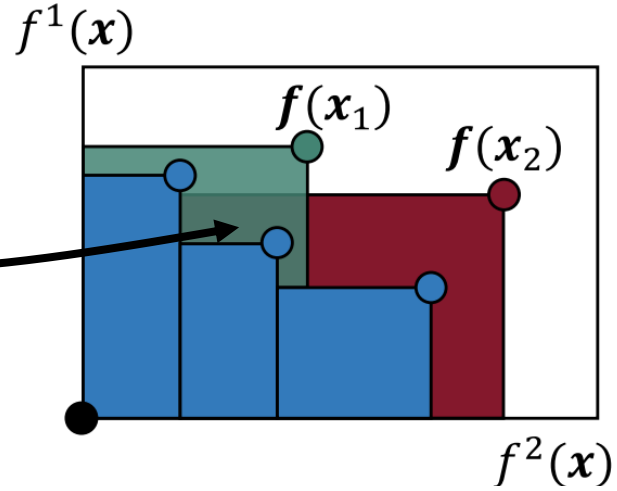



# VDTuner Solution: Multi-Objective Bayesian Optimization

- Bayesian Optimization
  - Surrogate model for cheap approximation
  - Acquisition function to choose iterative samples




- Multi-Objective Bayesian Optimization
  - Expected improvement on hypervolume

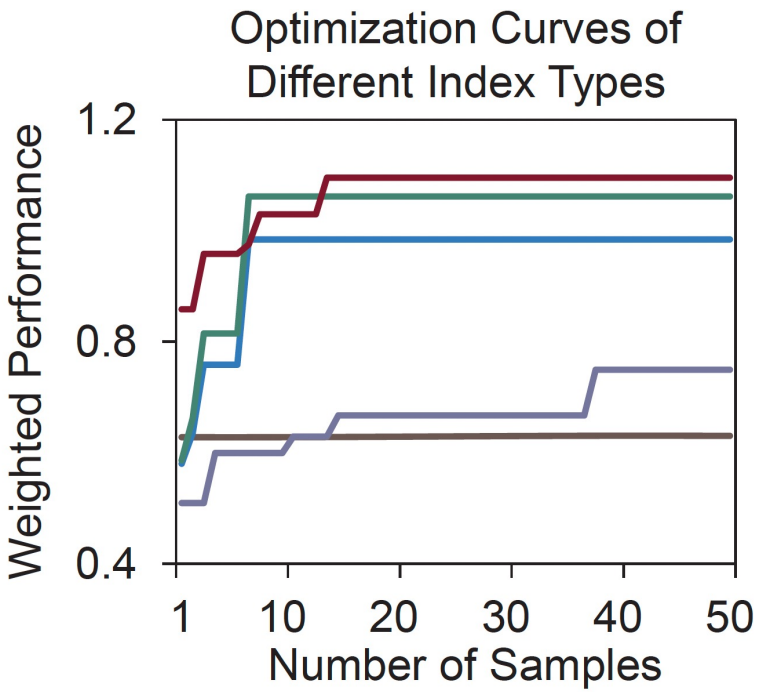



- Our context: Max. {Speed, Recall}

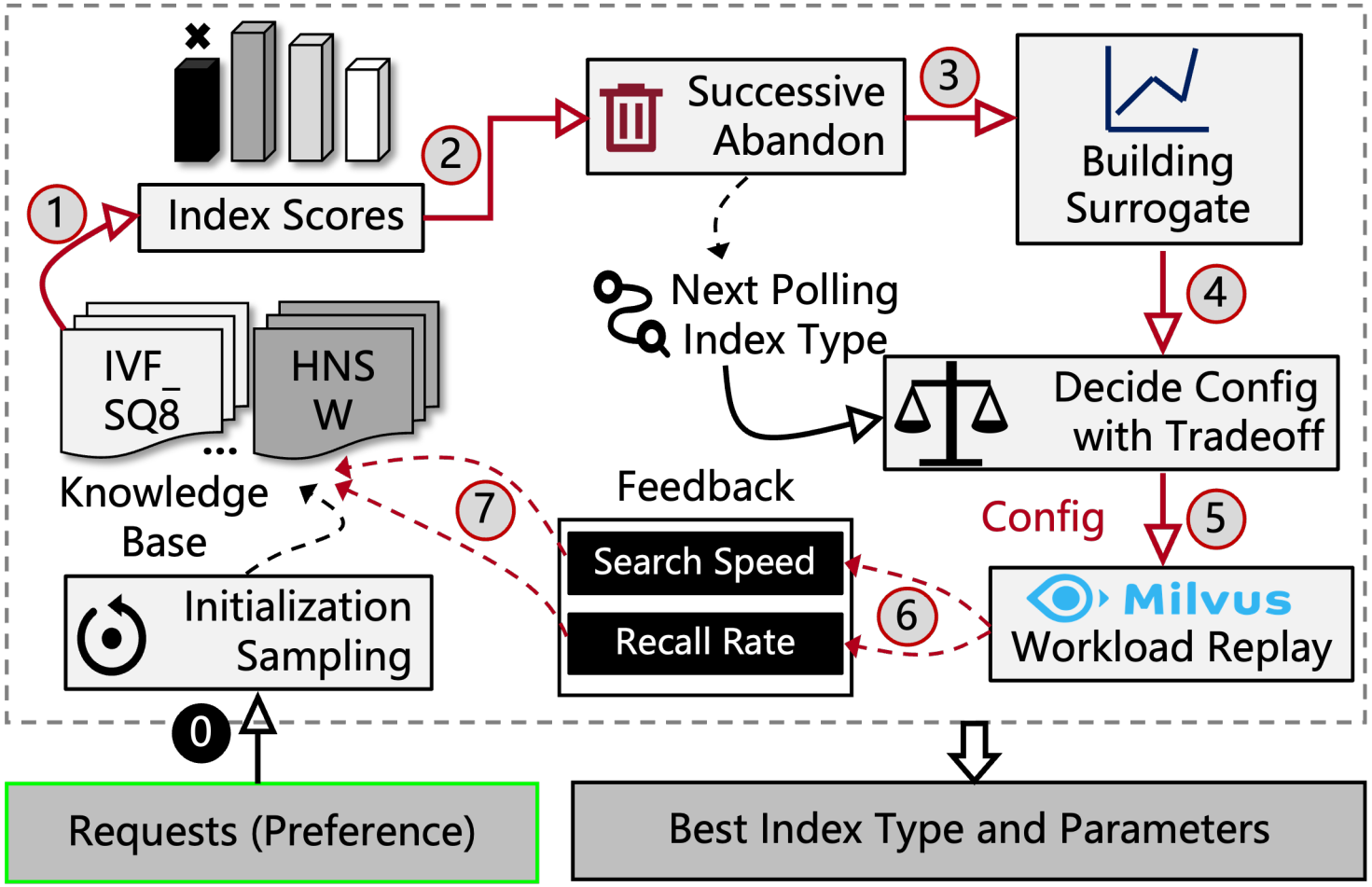
# Challenge #2: Multiple Index Types

- Tunable knobs are different for different index types;
- Identifying the best index within limited budget is challenging!

| Index Type | Building Knob | Search Knob |
|---|---|---|
| FLAT | N/A | N/A |
| IVF_FLAT | nlist | nprobe |
| IVF_SQ8 | nlist | nprobe |
| IVF_PQ | nlist, m, nbits | nprobe |
| HNSW | M, efConstruction | ef |
| SCANN | nlist | nprobe, reorder_k |



# VDTuner Solution: Holistic Learning + Index Polling




→: A complete iteration

③: Building a holistic BO model for sharing tuning info between indexes

① ② ④: Polling one index once, gradually discarding bad indexes!

# Experiment Setup

- **Platform**: Milvus (version 2.3.1), 8 index knobs & 7 system knobs
- **Evaluated Datasets**: Glove; Keyword-Match; Geo-Radius (2048dim)

| System Knobs | Type |
|---|---|
| dataCoord.segment.maxSize | int |
| dataCoord.segment.sealProportion | int |
| queryCoord.autoHandoff | bool |
| queryCoord.autoBalance | bool |
| common.gracefulTime | int |
| dataNode.segment.insertBufSize | int |
| rootCoord.minSegmentSizeToEnableIndex | int |

| Index Type | Building Knob | Search Knob |
|---|---|---|
| FLAT | N/A | N/A |
| IVF_FLAT | nlist | nprobe |
| IVF_SQ8 | nlist | nprobe |
| IVF_PQ | nlist, m, nbits | nprobe |
| HNSW | M, efConstruction | ef |
| SCANN | nlist | nprobe, reorder_k |

# Experiment Results: Tuning or Not?

- The default VDB configuration may not be optimal, and VDTuner significantly improves VDB performance (e.g., **186%** recall improv.).

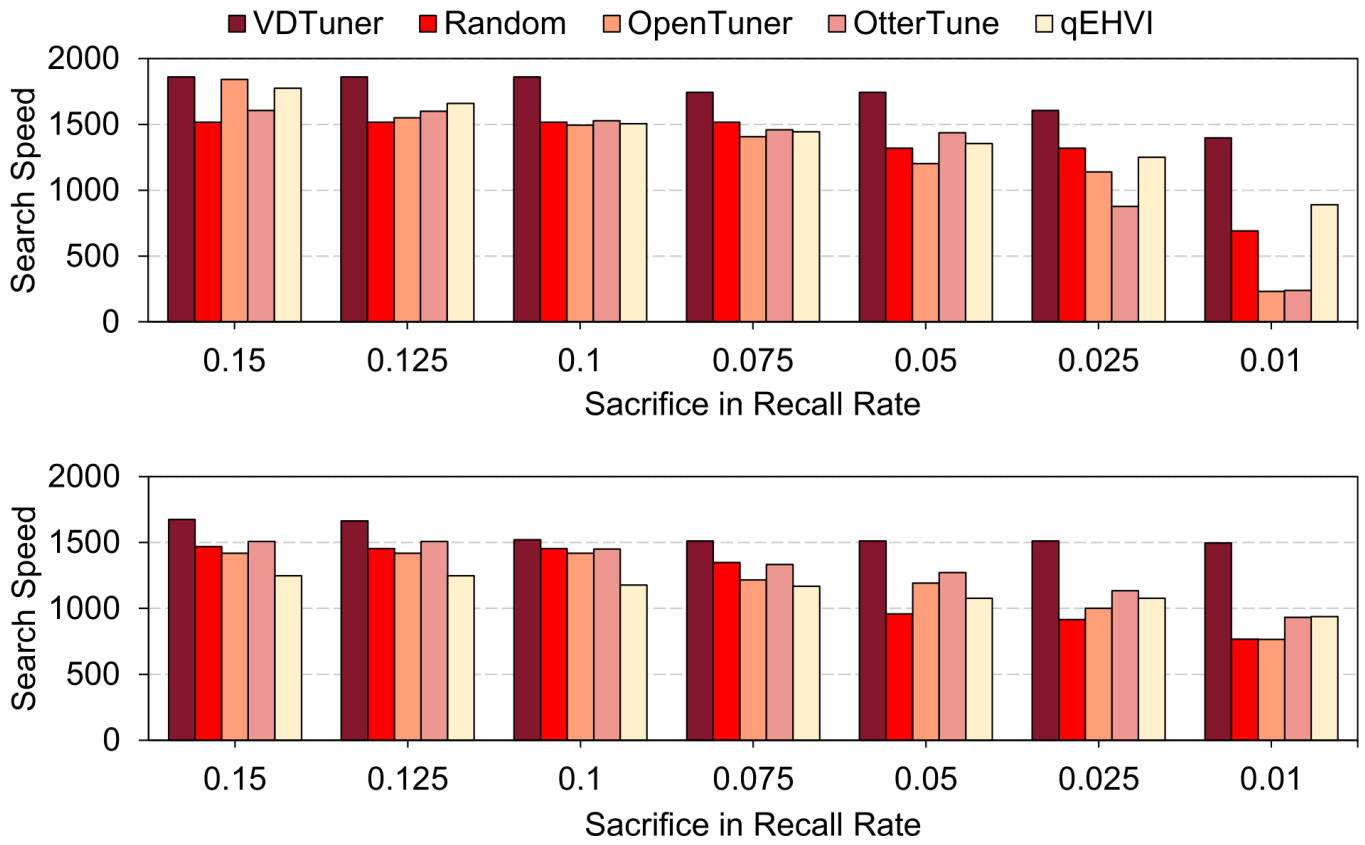
PERFORMANCE IMPROVEMENT BY AUTO-CONFIGURATION.

| Metric | Datasets | | |
|---|---|---|---|
| | GloVe | Keyword-match | Geo-radius |
| Speed Improvement | 10.46% | 11.17% | 14.12% |
| Recall Improvement | 17.16% | 62.61% | 186.38% |

*Note: Perf improv. is defined as the maximum enhancement in search speed (or recall rate) without sacrificing recall rate (or search speed) relative to default performance.*

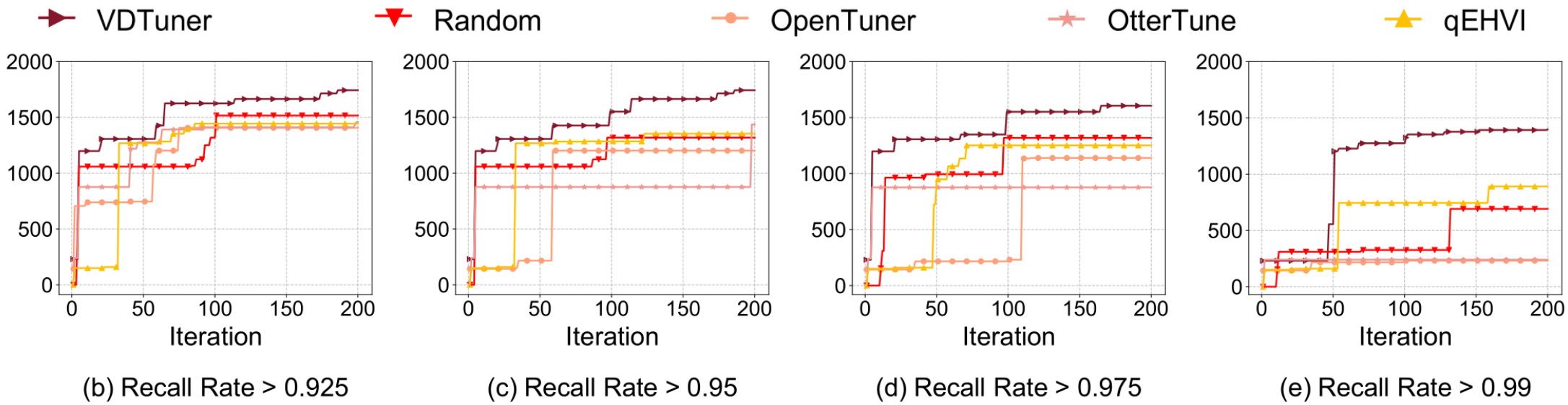
# Experiment Results: VDTuner or Other Tuners?

- VDTuner strikes a good balance between goals, surpassing baseline' perf (**59.54%** better speed).
- Stricter recall restrictions, greater advantages! (11.15% → 59.54%)



# Experiment Results: Tuning Speed

- VDTuner identifies better configurations markedly faster compared with competing baselines (up to **3.57x** faster).




(b) Recall Rate > 0.925 (c) Recall Rate > 0.95 (d) Recall Rate > 0.975 (e) Recall Rate > 0.99

# VDTuner Summary

- Automated Performance Tuning: VDTuner optimizes vector database index and system knobs via:
  - Multi-objective Bayesian optimization (MOBO);
  - A polling structure with a specialized surrogate model
- Significant Performance Gains & Scalability:
  - Substantially improves VDB search speed & recall rate;
  - Adapts to fluctuating user preferences and cost-aware objectives
- Deployed in Ant Group's CodeFuse: Enhancing RAG, LLM Caching

# This Talk: System Efficiency for Search Agents

**Auto-Tuning Vector Database**
For Conflicting Obj. & Multiple Indexes
[ICDE 2024, CodeFuse]

Info Retrieval

LLM Generation

**Auto-Tuning LLM Inference Engine**
SLO-Oriented
[WWW 2025, Oral]

# SCOOT: SLO-Oriented Performance Tuning for LLM Inference Engines

With Ke Cheng, Zhi Wang, Wen Hu, Jianguo Li, Sheng Zhang

# Search agents necessitate LLM inference engines for text generation

- LLM inference engines, e.g., vLLM and Tensorrt-LLM, are equipped with cutting-edge techniques designed for serving LLMs.
- Engine knobs: adjusting specific techniques, e.g., the scheduler

vLLM

| Configuration Parameter | Type | Range |
|---|---|---|
| tensor-parallel | Integer | [1, #GPUs] |
| max-num-seqs | Integer | [64, 8192] |
| max-num-batched-tokens | Integer | [64, 8192] |
| block-size | Enumeration | {8, 16, 32} |
| scheduler-delay-factor | Float | [0, 2] |
| enable-chunked-prefill | Boolean | {True, False} |
| enable-prefix-caching | Boolean | {True, False} |
| disable-custom-all-reduce | Boolean | {True, False} |
| use-v2-block-manager | Boolean | {True, False} |

TensorRT-LLM

| Configuration Parameter | Type | Range |
|---|---|---|
| tp-size | Int | [1, #GPUs] |
| max-batch-size | Int | [64, 8192] |
| max-num-tokens | Int | [64, 8192] |
| tokens-per-block | Enum | {4, 8, 16, 32, 64, 128} |
| use-paged-context-fmha | Bool | {True, False} |
| enable-chunked-context | Bool | {True, False} |
| enable-block-reuse | Bool | {True, False} |
| capacity-scheduler-policy | Enum | {FCFS, EQUAL} |
| context-chunking-policy | Enum | {MAX_UTI, NO_EVICT} |

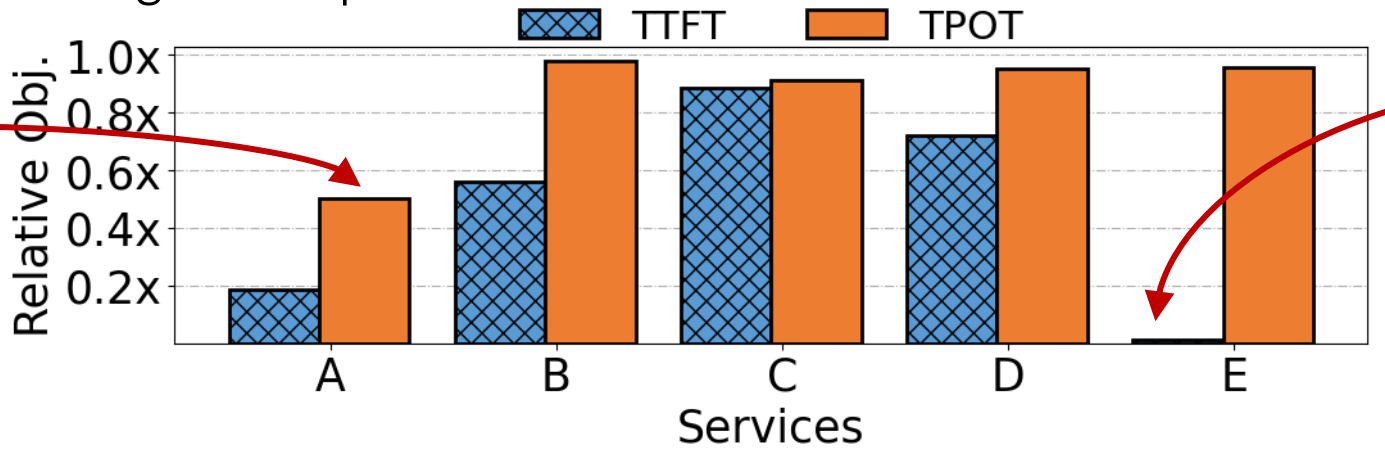
# Tuning engine knobs enhances performance!

Table: VARIOUS SERVICE CHARACTERISTICS

| Service | Application | GPU Type | GPU Number | LLM |
|---|---|---|---|---|
| A | SQL | A100 | 2 | LLAMA2-13B |
| B | **BOT** | A100 | 2 | LLAMA2-13B |
| C | BOT | **A10** | 2 | LLAMA2-13B |
| D | BOT | A10 | **4** | LLAMA2-13B |
| E | BOT | A10 | 4 | **LLAMA2-7B** |

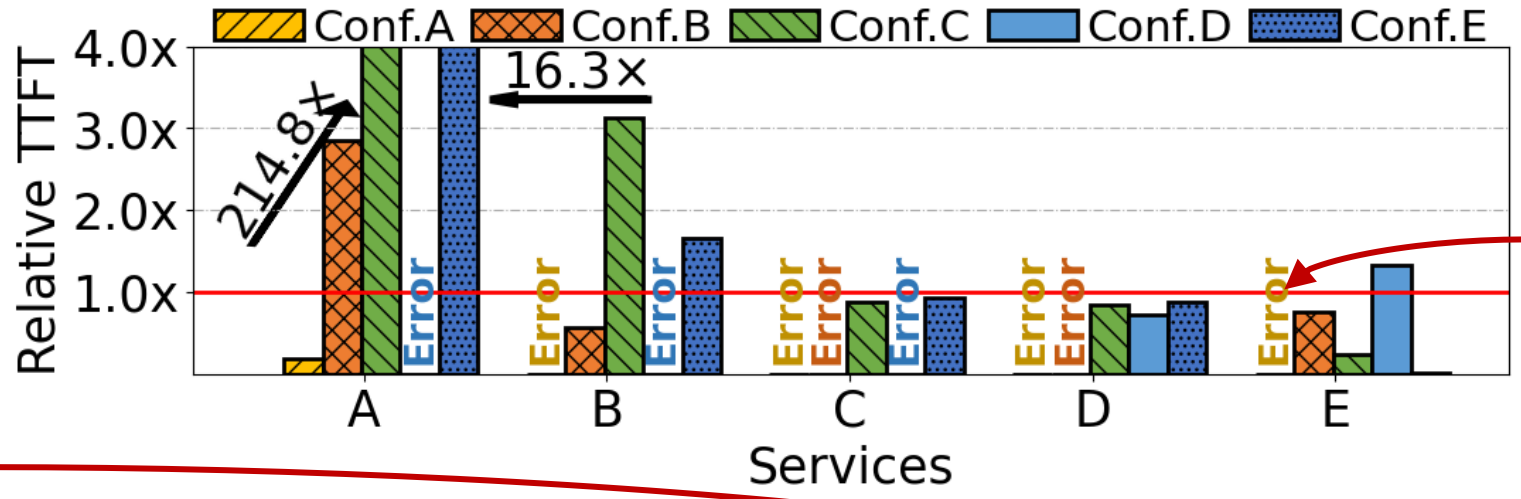
Figure: Optimal TTFT and TPOT for various services.

*49.9% TPOT Reduction*



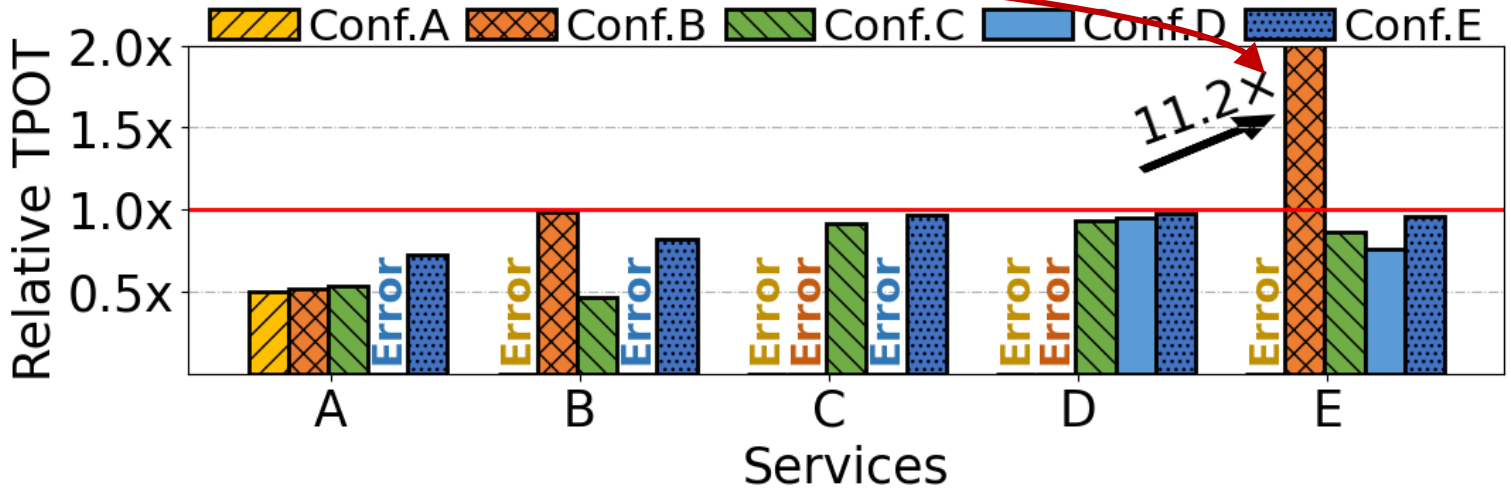

*98.9% TTFT Reduction*

# No Universal Best-Practice Knob Config




*Crash Caused by Errors*

*Performance Degradation*



# Challenges of Auto-Tuning LLM Engines

Diverse Optimization Objectives:

Users need tailored optimizations (e.g., throughput, TTFT/TPOT).

→ Yes, MOBO again! But what's new?

Complex Constraints:

- Known parameter dependencies
  - E.g., *max_num_batched_tokens* > *max_num_sequences*
- Unknown (Hidden) crashes due to specific parameter combinations
  - In vLLM 4.0, crashes happen with too large *scheduler_delay_factor*

# SCOOT Problem Definition

Multiple Objectives:

- Throughput (offline scenarios, e.g., recommendation);
- Tail latency; TTFT; TPOT (online scenarios, e.g., chatbot)

$$\mathbb{P}: \max_{\boldsymbol{x} \in \boldsymbol{\Lambda}} \quad \lambda_t \cdot T(\boldsymbol{x}), \lambda_l \cdot L(\boldsymbol{x}), \lambda_\phi \cdot \Phi(\boldsymbol{x}), \lambda_\theta \cdot \Theta(\boldsymbol{x}) \qquad (1)$$

**s. t.**

Known Constraints $\quad c_i, \forall c_i \in \mathcal{C}, \qquad (2)$

Hidden Constraints $\quad POF(\boldsymbol{x}) \geq \Delta, \qquad (3)$

Prob. Of Feasibility

# SCOOT's Way of Avoiding "Crashing Configs"

A lightweight ML model to predict $POF(x)$

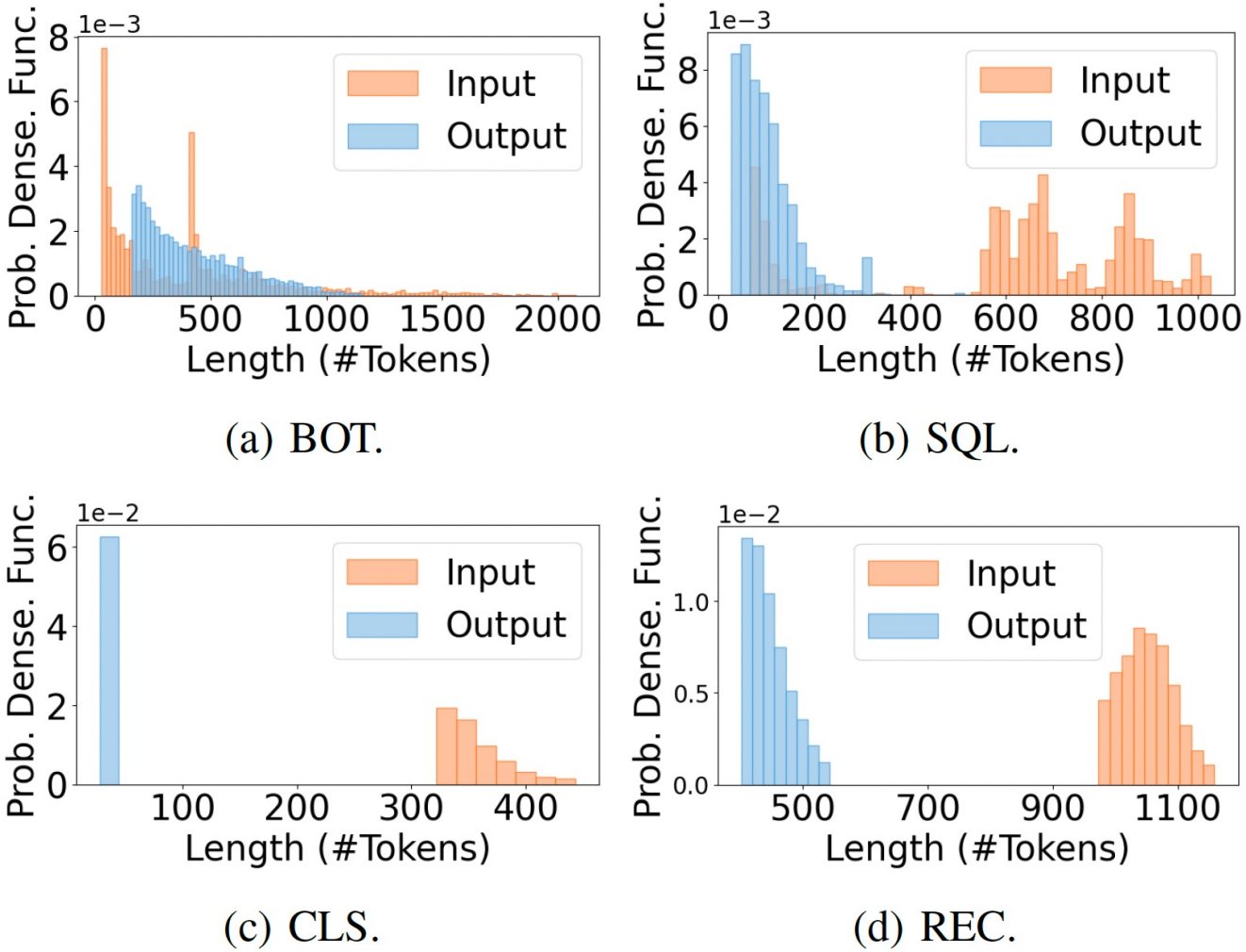
- E.g., Random Forest
- Refining the model with more observed configs

Adaptively adjusts threshold $\Delta$ to balance "exploration-exploitation"

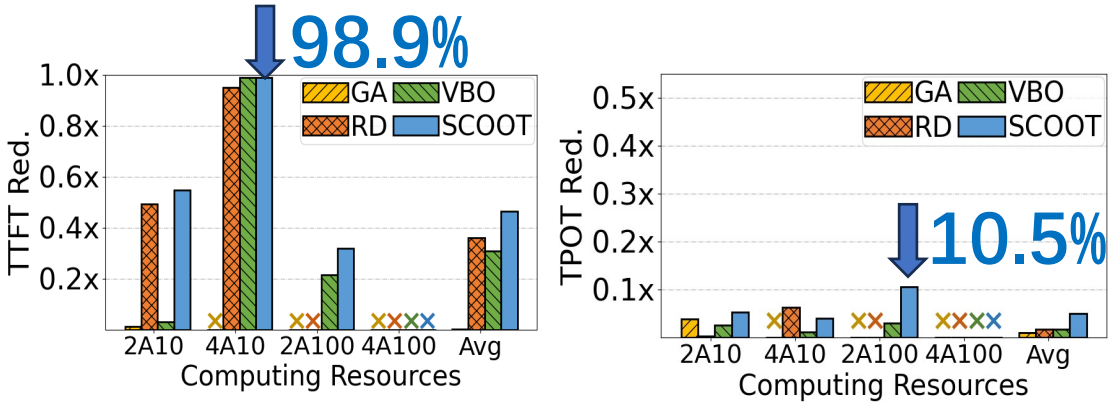
- Infeasible config found: $\Delta\uparrow$, be more cautious;
- Successive feasible configs: $\Delta\downarrow$, be more ambitious!

# Experiment Setup

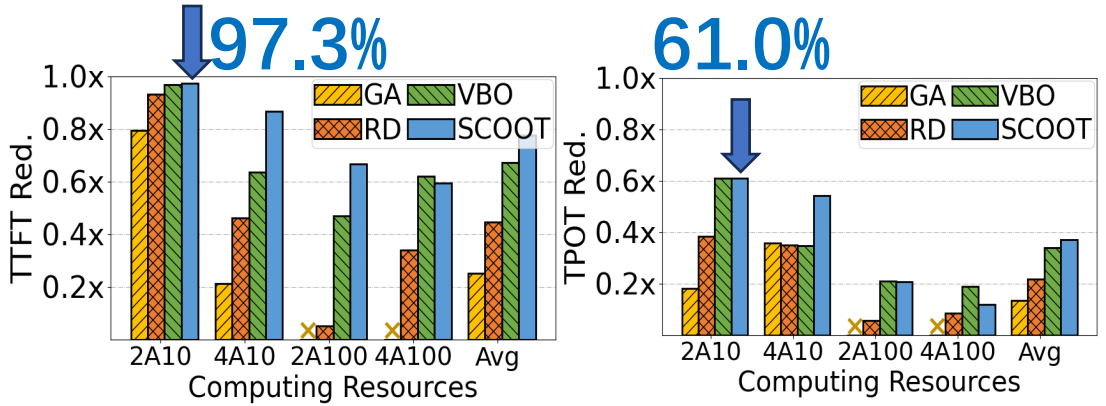
- Models: LLAMA2-7B, LLAMA2-13B
- Testbed:
  - 2× & 4× NVIDIA 24GB A10 GPUs
  - 2× & 4×NVIIDA 80GB A100 GPUs
- Request Traces Collected from Ant Group (Right Figs)
- Baselines
  - Uniformly Random Sampling (RD)
  - Genetic Algorithm (GA)
  - Vanilla Bayesian Optimization (VBO)



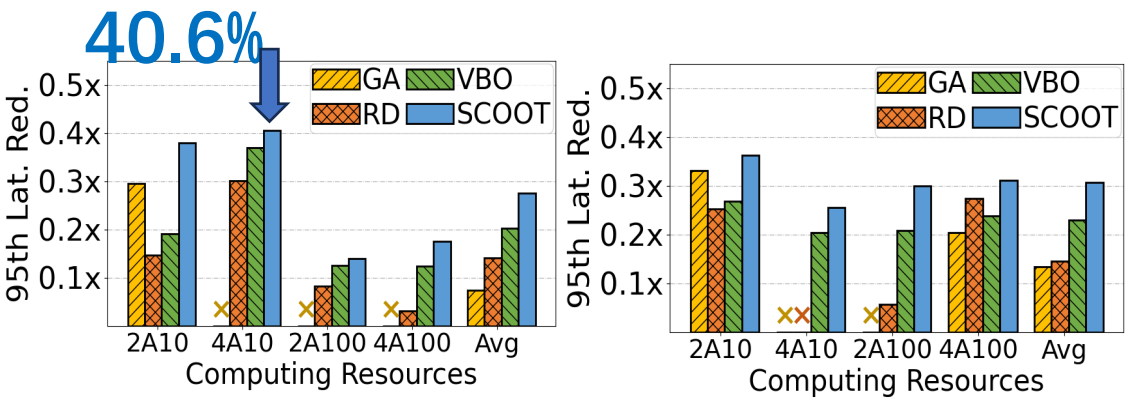
# Experiment Results: Overall Performance



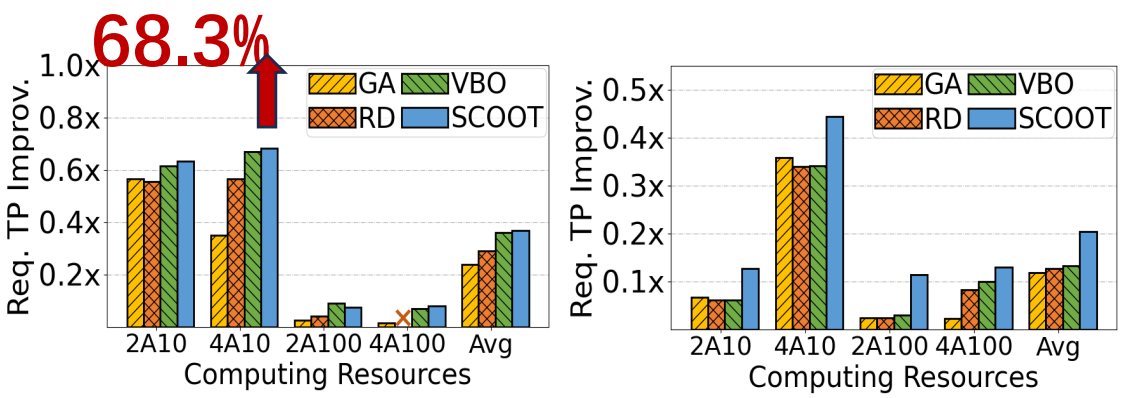

BOT, LLAMA2-7B, TTFT (left) & TPOT (right)




SQL, LLAMA2-7B, TTFT (left) & TPOT (right)




CLS, 95th Lat., LLAMA2-7B (left) & 13B (right)

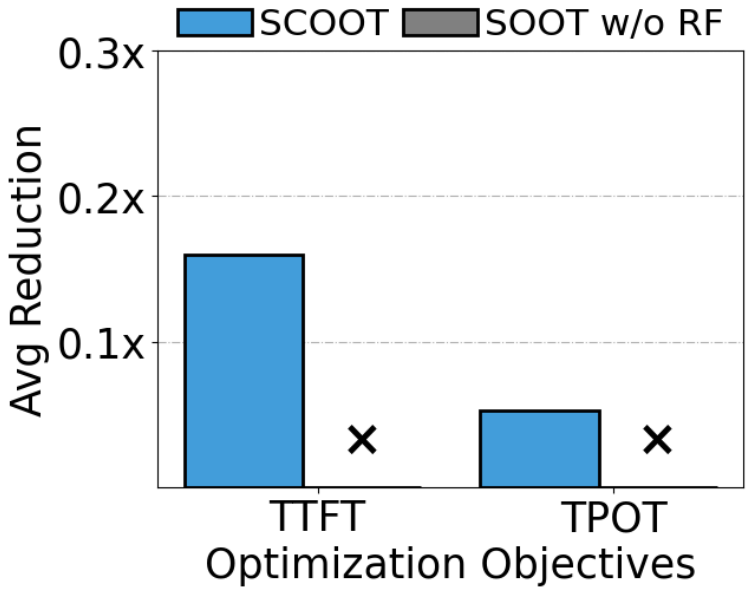



REC, Throughput, LLAMA2-7B (left) & 13B (tight)

# Experiment Results: Avoiding Crashing Configs

- SCOOT's feasibility learning greatly enhances tuning effectiveness!


95th Lat Reduction, CLS, LLAMA2-7B & 13B




TTFT & TPOT Reduction, BOT, LLAMA2-7B

# Lessons From Two Auto Tuning Research

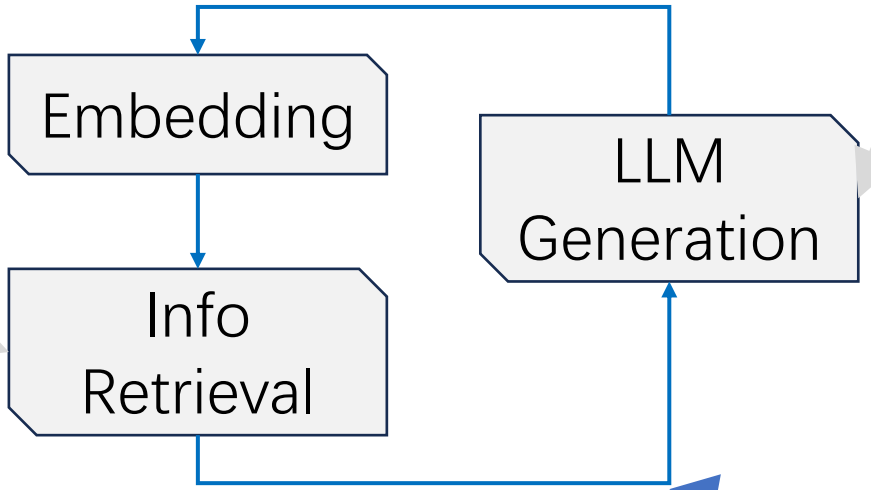
- Note! Your default configuration may be costing you profits!
- Worse, no universal best-practice configuration.

- Black-box optimization goes first — for its generality; **BUT** needs further accelerations, including:
  - Conflicting goals; Hierarchical knob dependencies; Config crashes,..

- While effective, it's still time-consuming, e.g., 2-3 hours for SCOOT.
  - Parallel sampling
  - Offline-online-separated tuning

# This Talk: System Efficiency for Search Agents

**Auto-Tuning Vector Database**
For Conflicting Obj. & Multiple Indexes
[ICDE 2024, CodeFuse]




**Auto-Tuning LLM Inference Engine**
SLO-Oriented
[WWW 2025, Oral]

★ Key Focus

**Demystifying and Enhancing Search Agent Efficiency**
Targeting Interleaved Search and LLM Reasoning
[ArXiv 2025]

SEARCHAGENT

# Demystifying and Enhancing the Efficiency of Large Language Model Based Search Agents

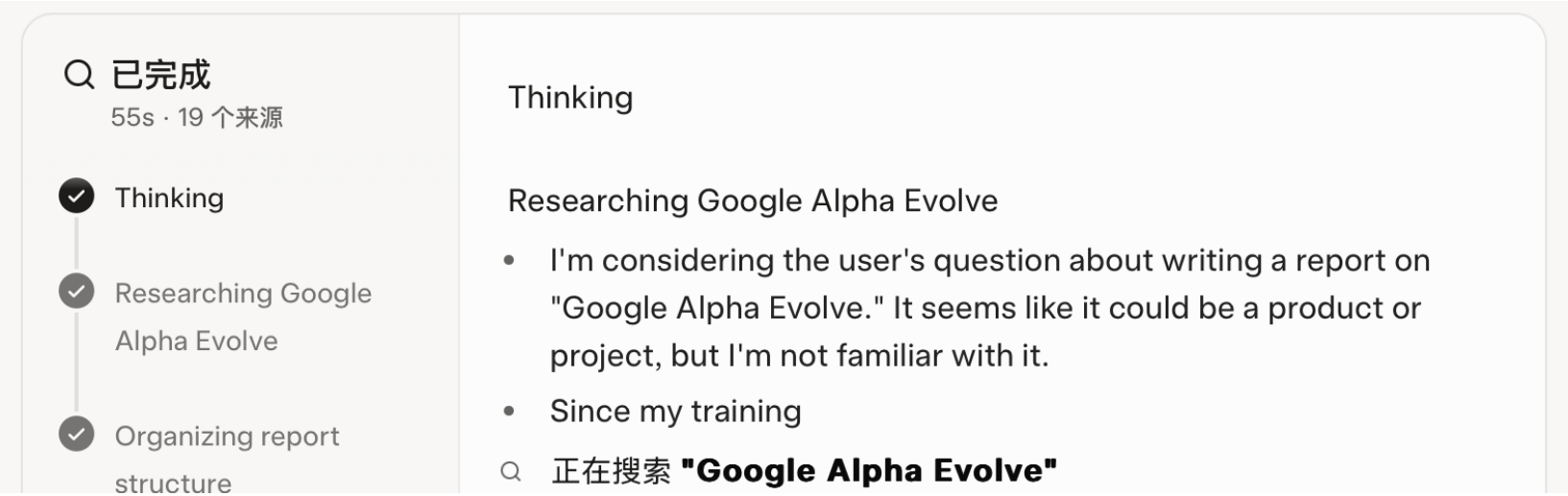
With Zebin Yao, Bowen Jin, Lixiao Cui, Yusen Li, Gang Wang, Xiaoguang Liu


南開大學
Nankai University

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

# Now, agents know what/when to search!

- RL and Reasoning supercharge RAG. LLM models now learn to autonomously call “retrievals” during self-reasoning!
- E.g., DeepResearch, DeepSearch, Search-R1, ReCall, R1-Searcher..




Grok DeepSearch

<think> </think>
<search> </search>
<information> </information>

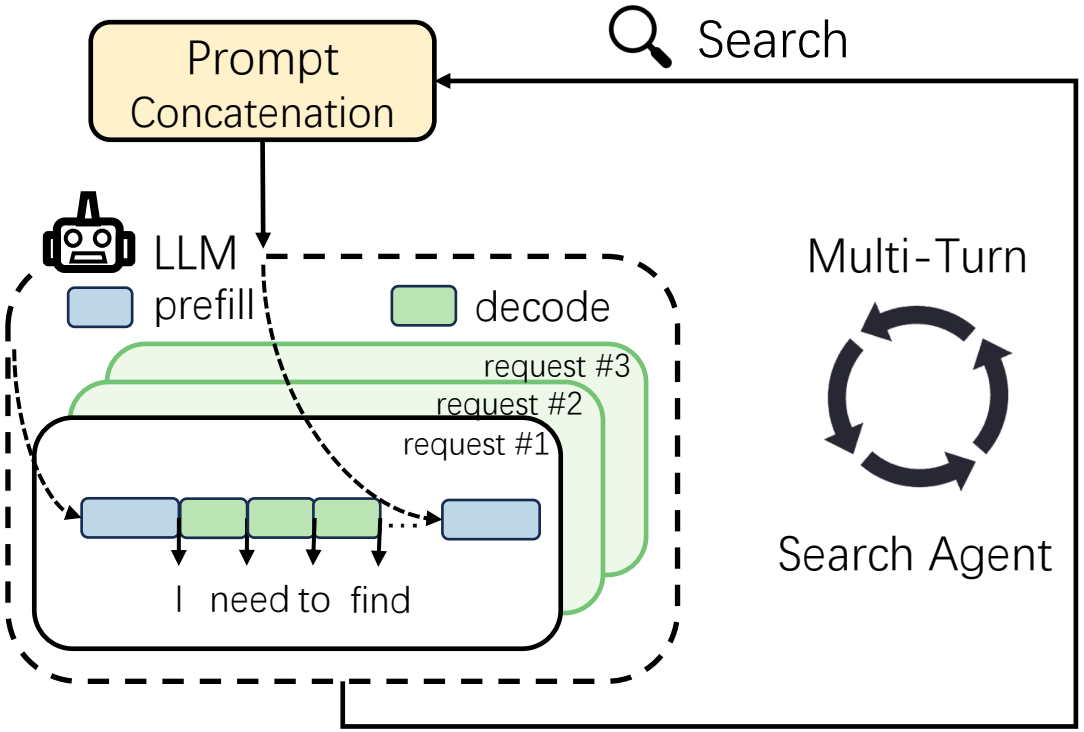
<think> </think>
<search> </search>
<information> </information>
…
<answer> </answer>

Search-R1 Workflow

# But, enhanced quality comes at efficiency cost!

Current Search Agent Systems

- Multi-Turn Reasoning Support:
  - Prompt concatenation
  - Prefix cache
- Sequence Scheduling:
  - Request-level (FastTransformer)
  - Iteration-level (vLLM, SGLang) ★

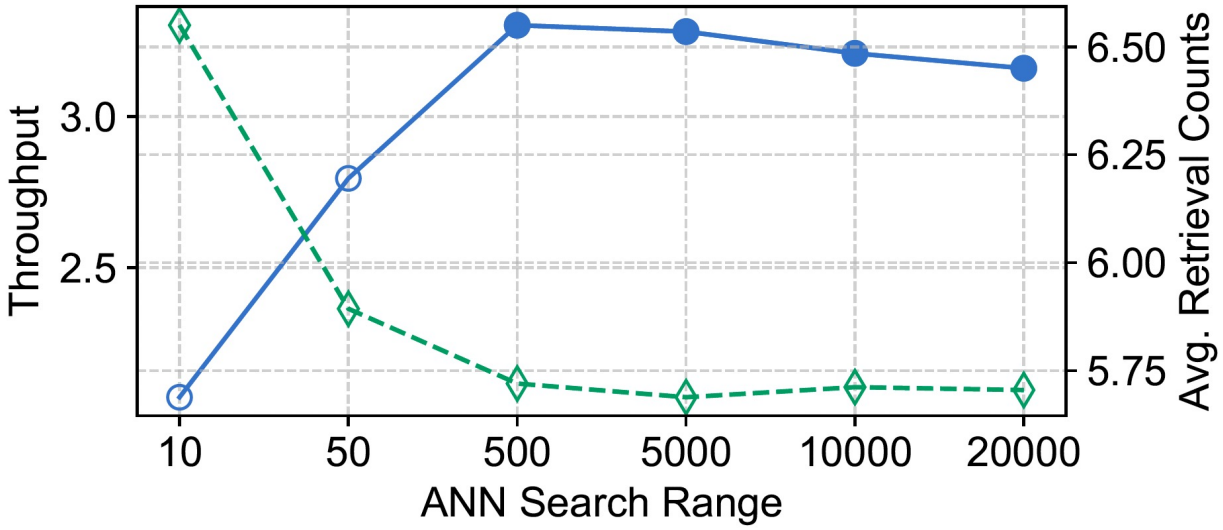



*Are current solutions **sufficient**?*

*What are the **key factors** impacting system efficiency?*

# Key Factor #1: Retrieval Accuracy

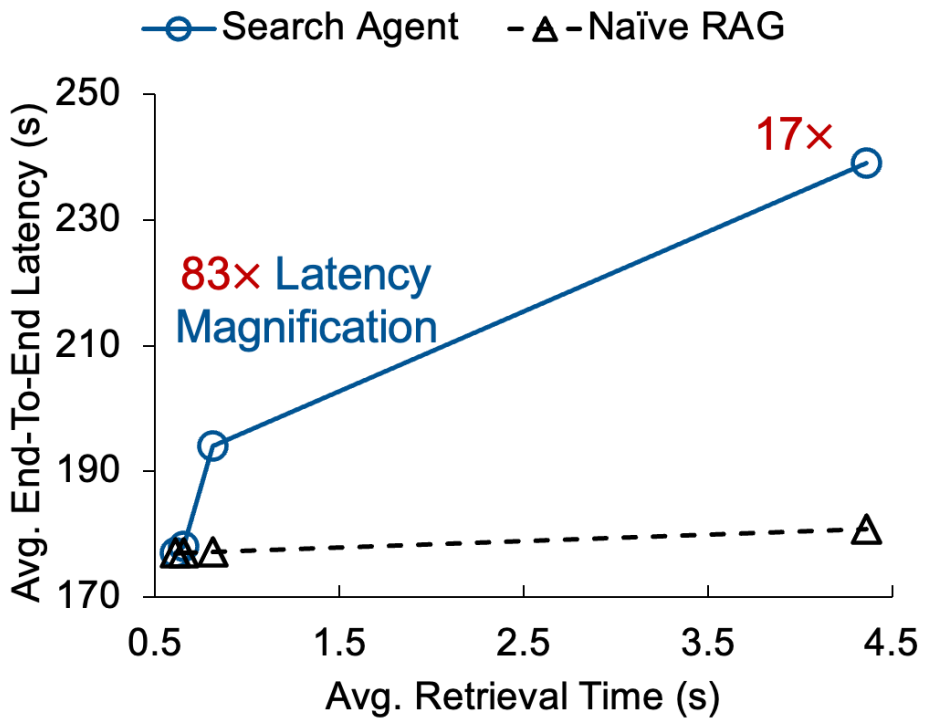
- Intuitively, the higher the retrieval quality, the better the efficiency
- NOT EXACTLY in Search Agent Systems!
  - Too low accuracy: More retrieval and reasoning steps to compensate;
  - Too high accuracy: Retrieval itself cost much!



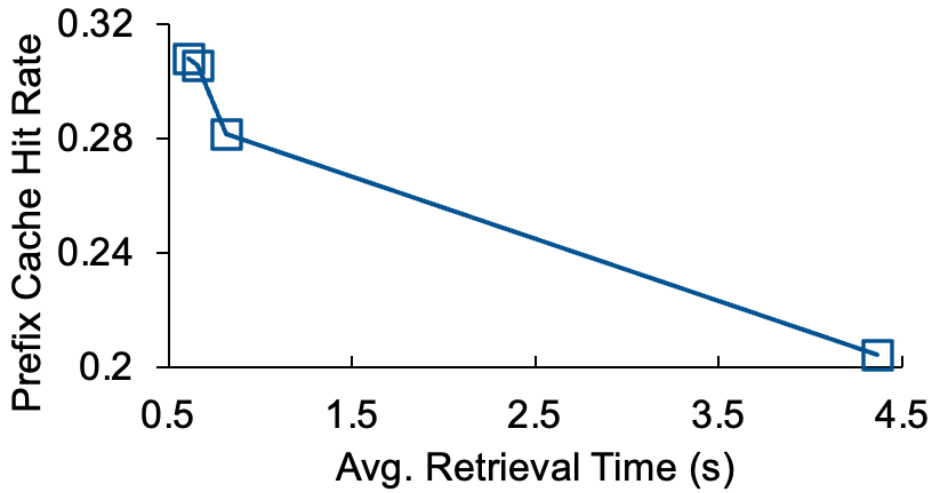

***High Recall ANN is All You Need!***

# Key Factor #2: Retrieval Latency

- Amplification Effect: Minor retrieval delays lead to substantial system inefficiency! Highly related to prefix cache hit rate!

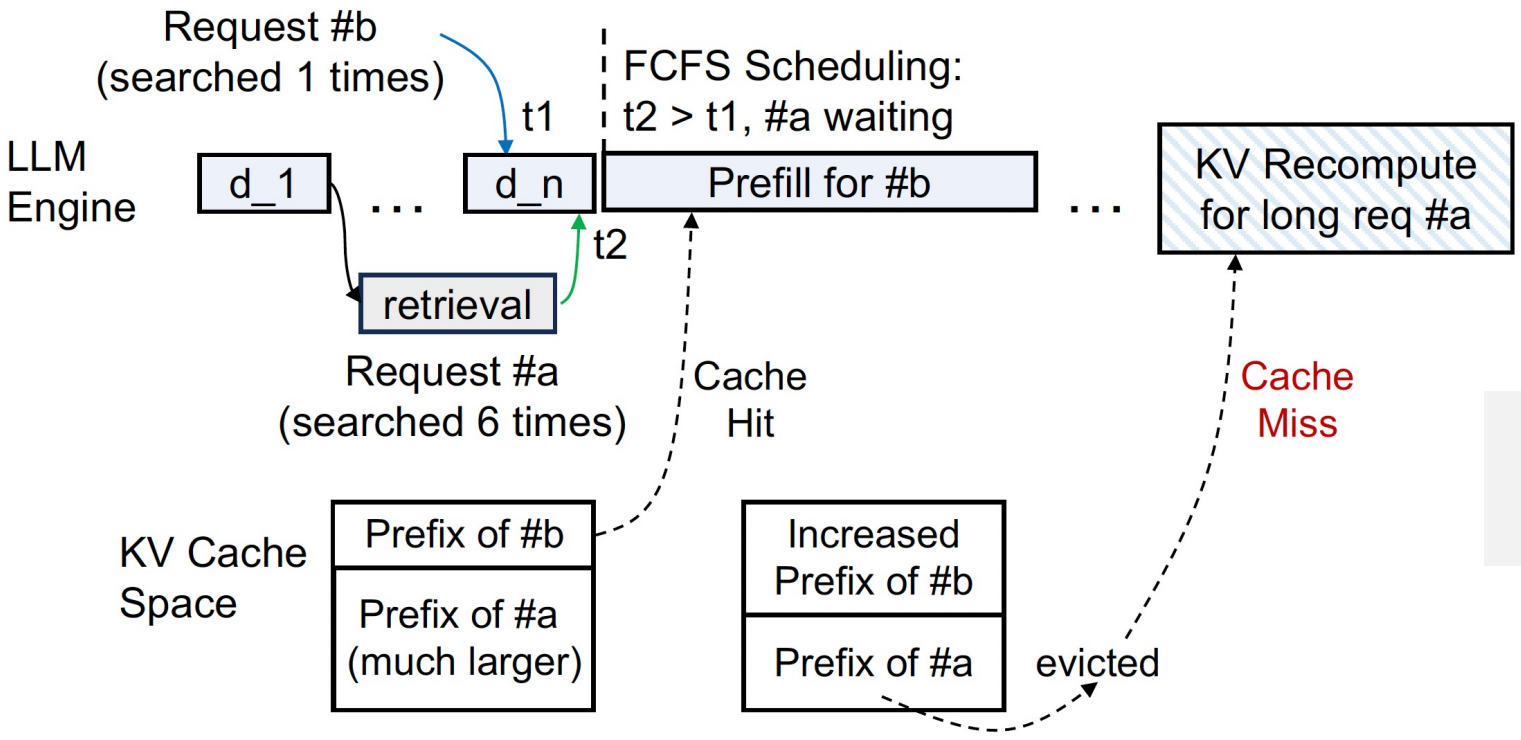



Caused By



# Root Causes of "Amplification Effect"

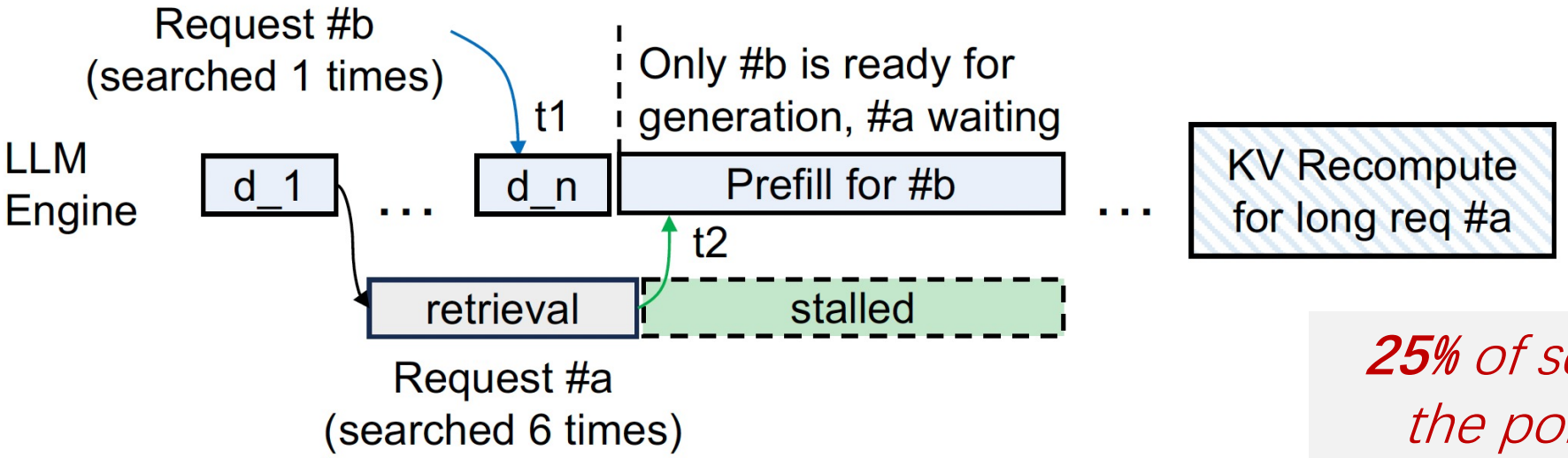
- Improper Scheduling: Standard "first-come, first-served" may allow short tasks to preempt the computing resources of long tasks




***55.9%** of the tokens recomputed unnecessarily!*

# Root Causes of "Amplification Effect"

- Retrieval Stalls: long requests missing the scheduling point must wait for next iteration, risking preemption by short requests!




***25%** of sequences missing the point due to stalls!*

# Summary of Insights

Current search agent systems suffer from low efficiency!

- Insight #1: Both overly high and overly low retrieval recall degrade end-to-end efficiency.

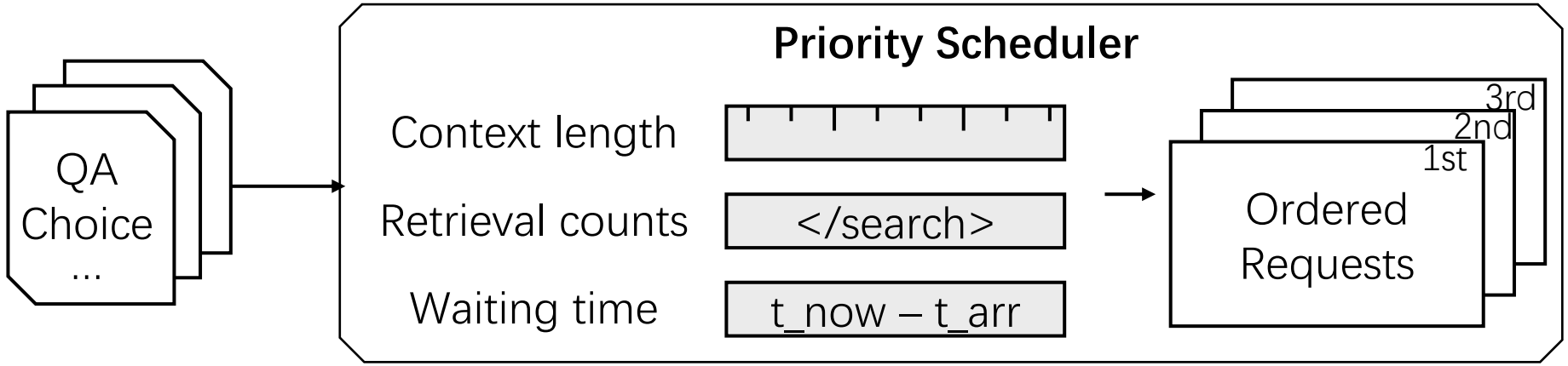
→ So, use a high-recall ANN!

- Insight #2: Search agents are very sensitive to minor retrieval delay due to ignoring inter-request priorities and retrieval stalls.

→ Address them!

# SearchAgent-X's Priority-Aware Scheduling

- Prioritize the most valuable requests to reduce unnecessary waiting and repetitive computation.
  - Retrieval count: more computational results; greater reuse reuse value
  - Context length: longer reusable caches
  - Waiting time: ensuring fairness

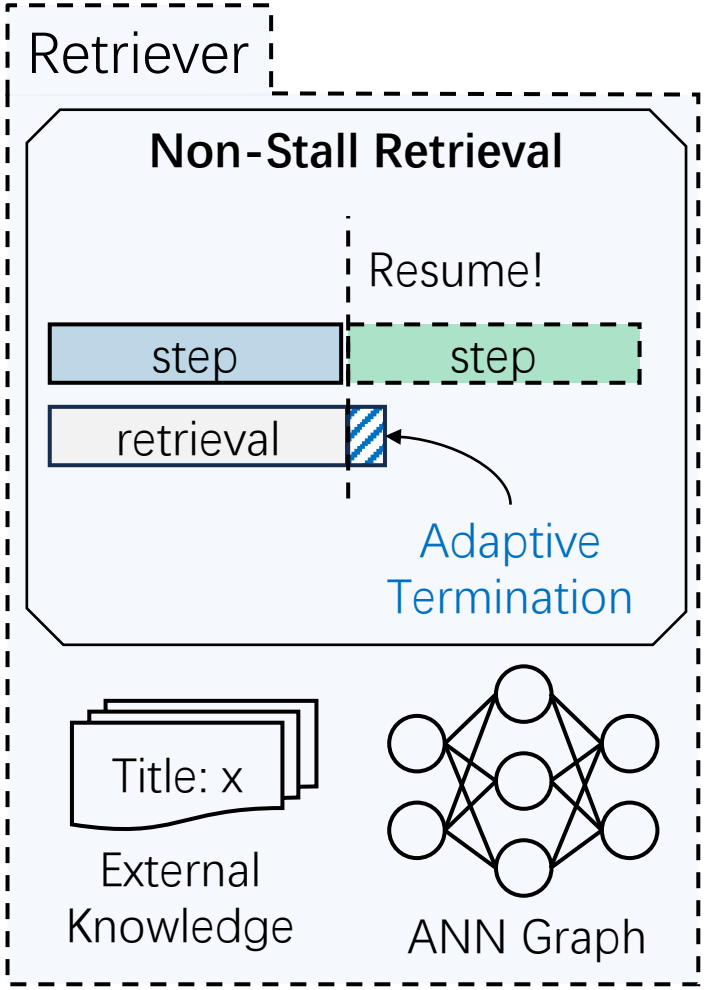


# SearchAgent-X's Non-Stall Retrieval

Adaptively judge whether to "quit while ahead", if both:

- retrieval results are "sufficiently mature"
- the LLM engine is ready

- Just-right "let go"! Ensuring information quality while preventing unnecessary waiting during the generation process



# Experiment Setup

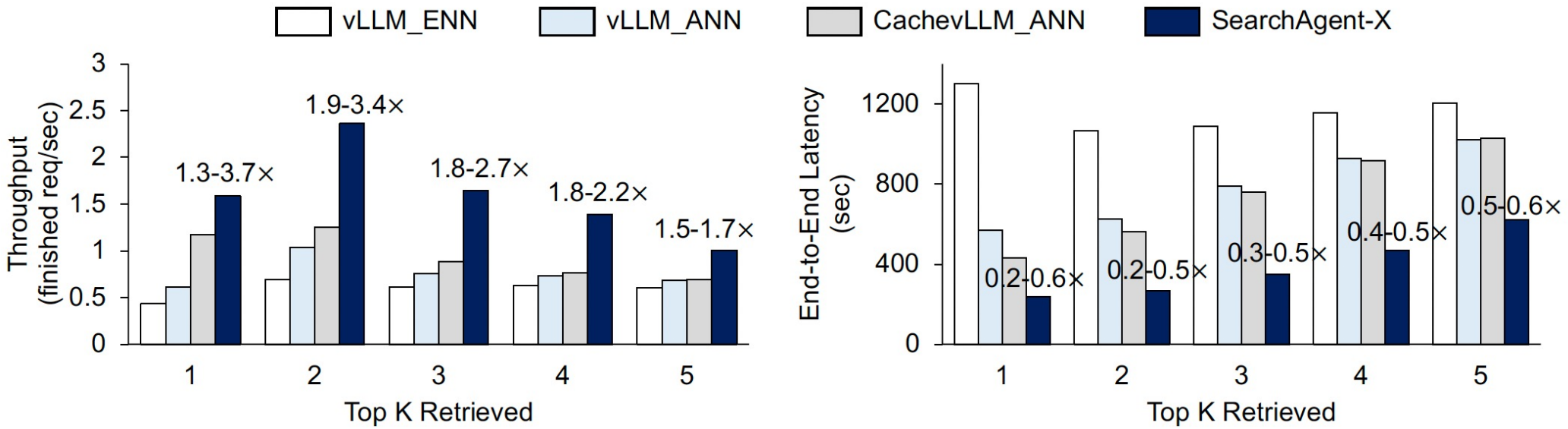
- Models: 7B & 14B search agent reasoning model from Search-R1
- Retrieval: HNSW, a popular graph-based ANN method
- Testbed:
  - 1x48G L20 for 7B model
  - 2x40G A100 for 14B model
- Baselines:
  - vLLM_ENN
  - vLLM_ANN
  - CachevLLM_ANN

*Easily **adaptable** to other **ANN methods** and **LLM reasoning models**!*

# Experiment Results: End-To-End Efficiency

Offline Inference (all requests arrive at once):

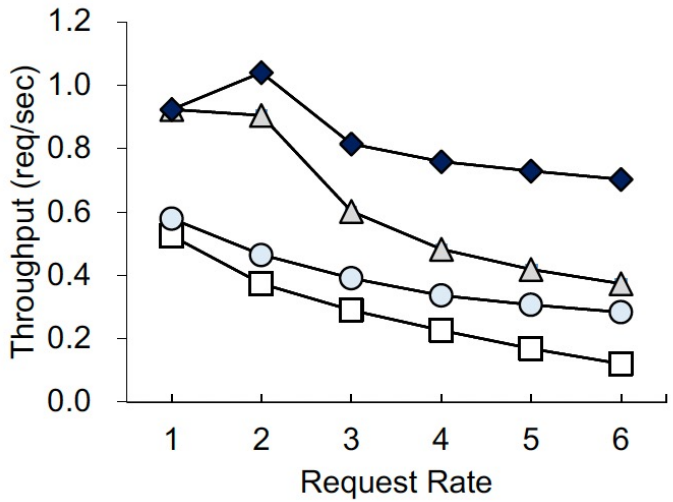
- SearchAgent-X achieves **1.3-3.4x** higher throughput than baselines, with average latency reduced to **20%-60%** of the baselines.

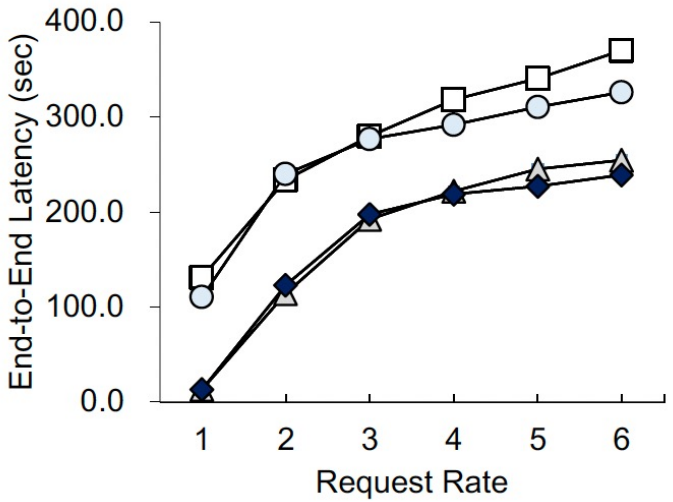


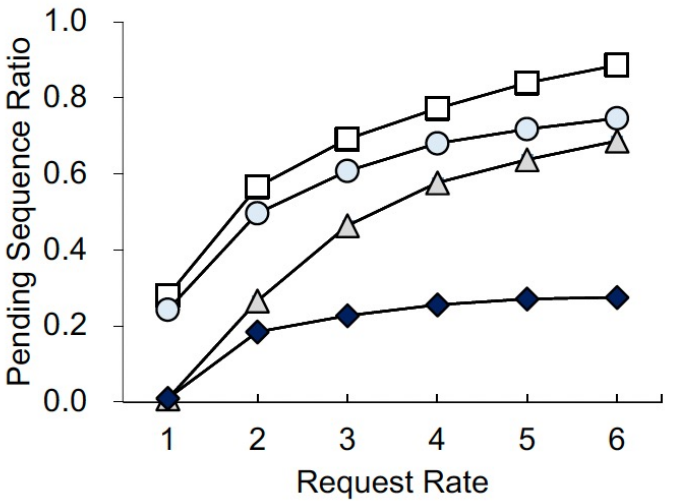
# Experiment Results: End-To-End Efficiency

Online Inference (requests arrive at a rate):

- SearchAgent-X completes **1.5-3.5x** more requests than baselines
- The higher the request rate, the more pronounced its advantage, reaching up to **5.8x** that of some baselines at its peak



# Experiment Results: Generation Quality

- No Compromises on Quality!

| Dataset | Musique | NQ | 2Wiki | HotpotQA | Bamboogle | StrategyQA | Avg. |
|---|---|---|---|---|---|---|---|
| **Generation Accuracy** | | | | | | | |
| Exact Retrieval | 0.203 | 0.316 | 0.371 | 0.429 | 0.472 | 0.788 | 0.430 |
| `SearchAgent-X` | 0.203 | 0.320 | 0.370 | 0.428 | 0.472 | 0.789 | 0.430 |
| **Retrieval Counts** | | | | | | | |
| Exact Retrieval | 3.247 | 2.288 | 3.126 | 2.702 | 2.440 | 2.496 | 2.717 |
| `SearchAgent-X` | 3.251 | 2.292 | 3.138 | 2.699 | 2.448 | 2.476 | 2.717 |
| **Output Length** | | | | | | | |
| Exact Retrieval | 8125 | 5839 | 7575 | 6840 | 6152 | 6402 | 6822 |
| `SearchAgent-X` | 8134 | 5847 | 7600 | 6839 | 6151 | 6382 | 6826 |

# Lessons of SearchAgent-X

Finding the bottleneck and explaining why > Your solution!

"System for AI": AI goes first!

- Keep up with AI developments
- Collaborate with AI people

Jensen Huang: AI is now Infrastructure.

Nvidia to invest $500b in US AI infrastructure by 2029

# Questions & Collaborations!

- Email: tnyang2000@gmail.com
- https://bytedance.larkoffice.com/docx/BMpfdhuo4oADNAxXn3dcPlD8nDe

微信

主页