

# Vector-Centric Machine Learning Systems: A Cross-Stack Approach

Wenqi Jiang

June 2025

# Computing infrastructure drives AI advancement

2012

## ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky  
University of Toronto  
kriz@cs.utoronto.ca

Ilya Sutskever  
University of Toronto  
ilya@cs.utoronto.ca

Geoffrey E. Hinton  
University of Toronto  
hinton@cs.utoronto.ca



2 x NVIDIA GTX 580 **gaming GPUs**

Each GPU: 3 GB memory, 1.5 TFLOPs

# Computing infrastructure drives AI advancement

2012

2015

2018

2023



**2 x GTX 580**

1.5 TOPs / chip

60M parameters

10 years

**25,000 x NVIDIA A100**

1248 TOPs / chip

> 1T parameters

**$10^3$  per-chip performance x  $10^4$  chips =  $10^7$  improvement**

# Tremendous investments on ML infrastructure

Compute

Announcing Trillium, the sixth generation of

**AI spending spree continues as Microsoft commits \$80B for 2025**

**Machine learning system efficiency matters!**

According to figures from Taiwan-based market watcher TrendForce, AI servers are projected to make up more than 70 percent of the total value of the server industry in 2025, adding up to about \$298 billion.

Sources: <https://blogs.microsoft.com/on-the-issues/2025/01/03/the-golden-opportunity-for-american-ai/>  
<https://www.trendforce.com/presscenter/news/20250106-12433.html>

# Presentation Outline

**Overview: ML system efficiency is beyond model acceleration**

My research: cross-stack, vector-centric ML systems

**RAGO: 1<sup>st</sup> systematic performance optimization for RAG**

Efficiently serving diverse and evolving RAG algorithms

**Chameleon: 1<sup>st</sup> heterogeneous accelerator system for RAG**

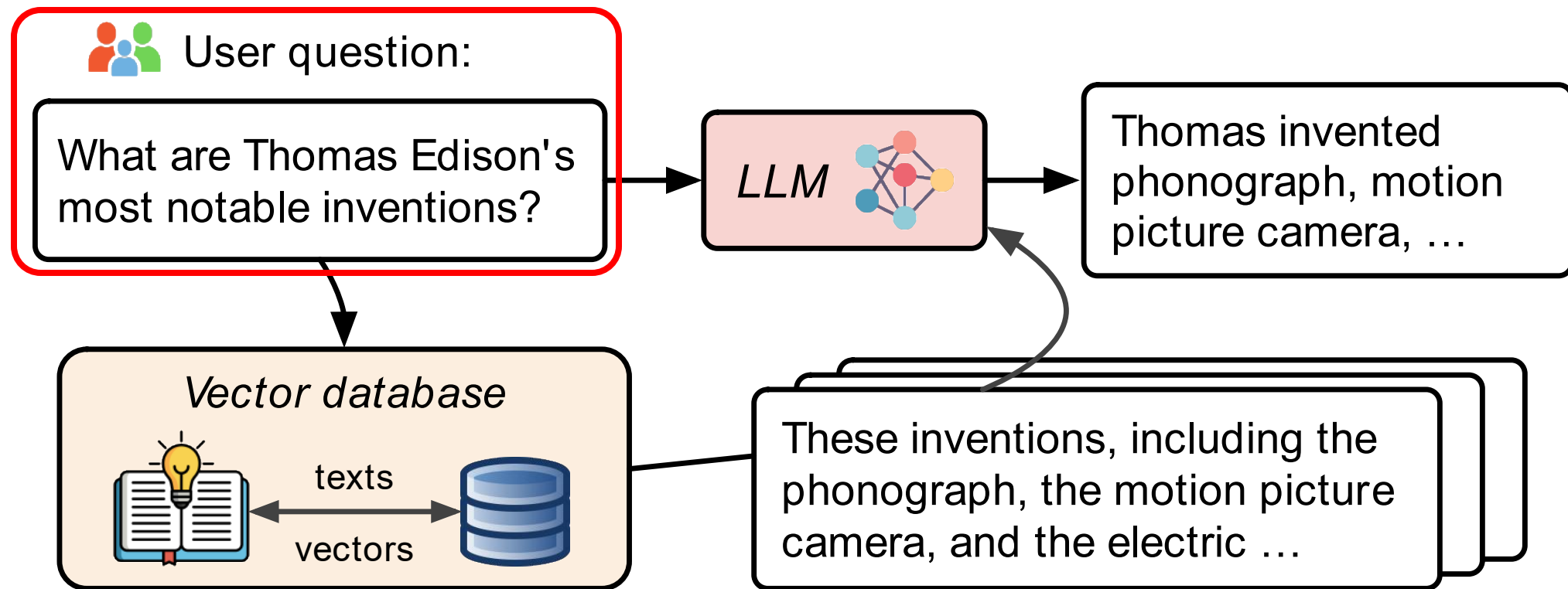
Explore hardware specialization for vector search

**Future work: next-generation machine learning systems**

Spanning algorithms, databases, systems, and hardware

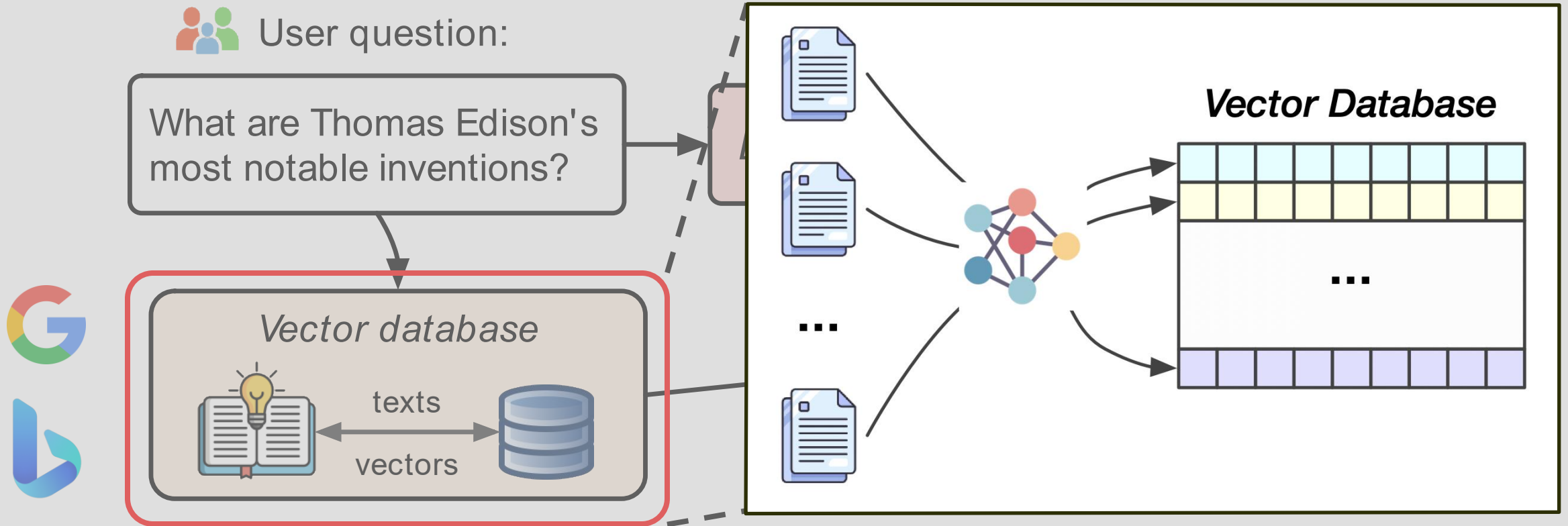
# Retrieval-augmented generation (RAG)

Key idea: pair LLMs with **retrievals from external databases**



# Retrieval-augmented generation (RAG)

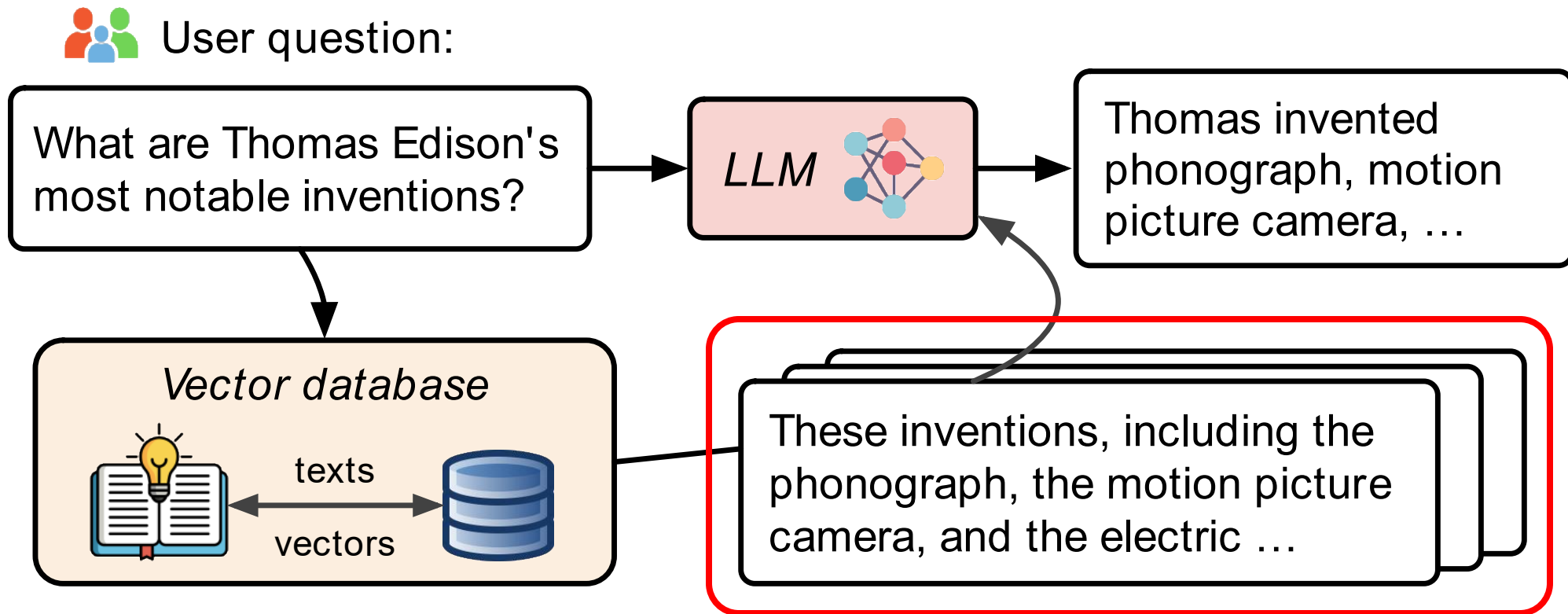
Key idea: pair LLMs with **retrievals from external databases**



**Retrieval = Vector Search**

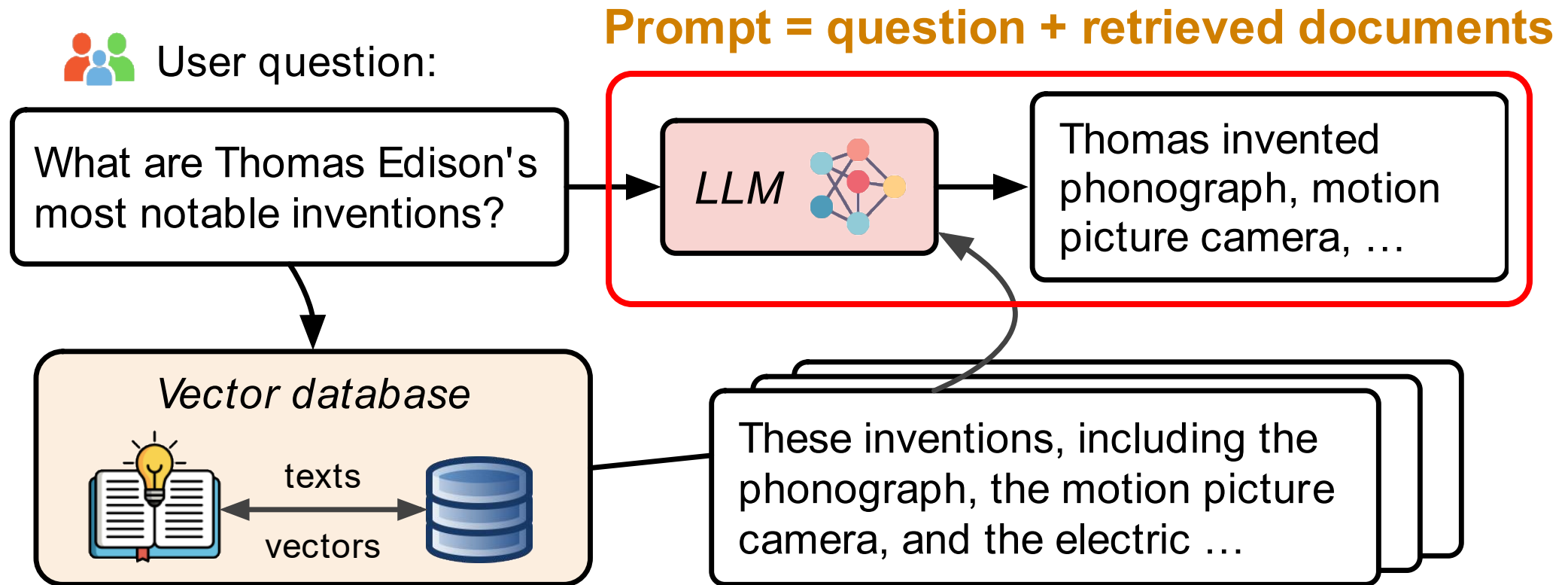
# Retrieval-augmented generation (RAG)

Key idea: pair LLMs with **retrievals from external databases**

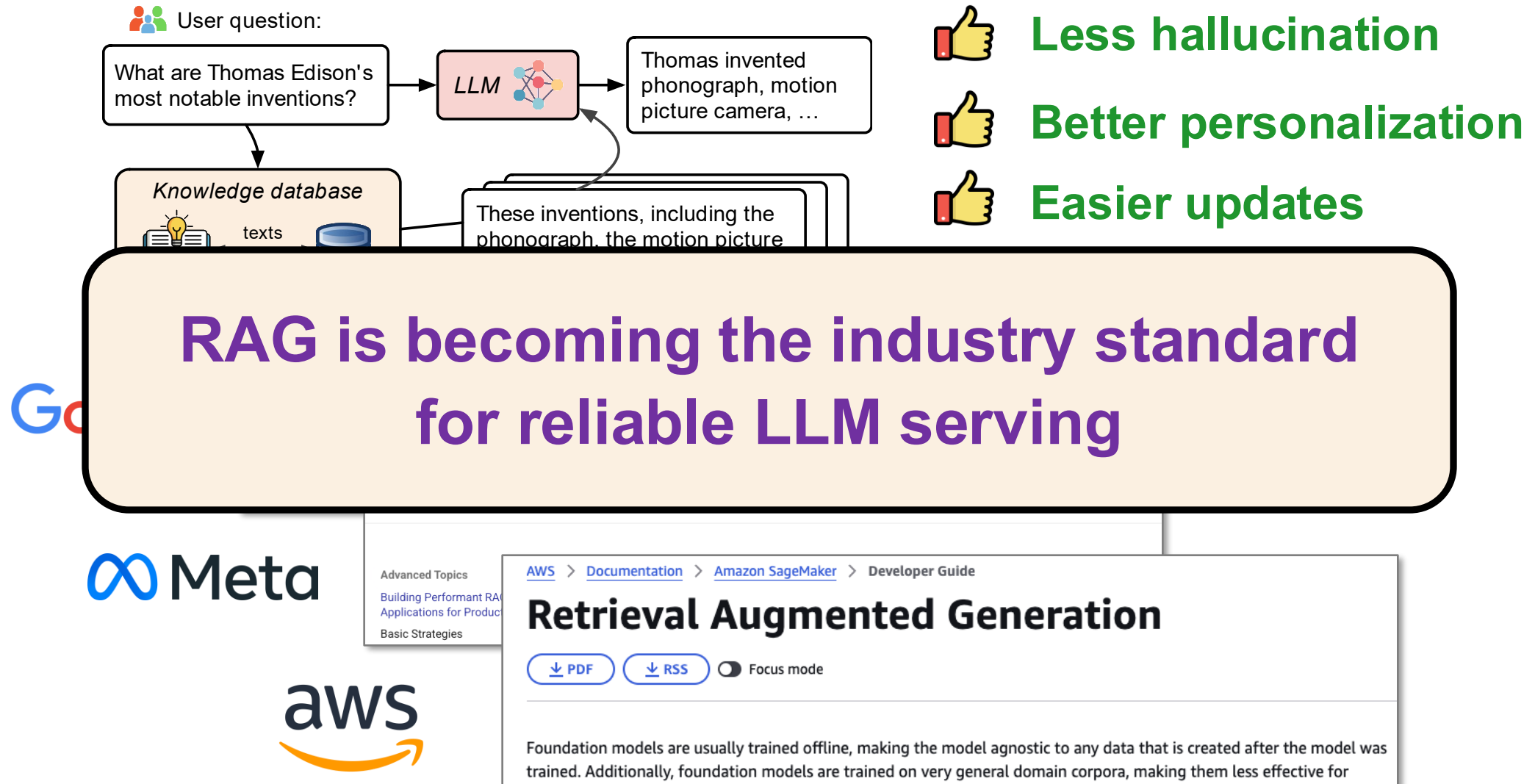


# Retrieval-augmented generation (RAG)

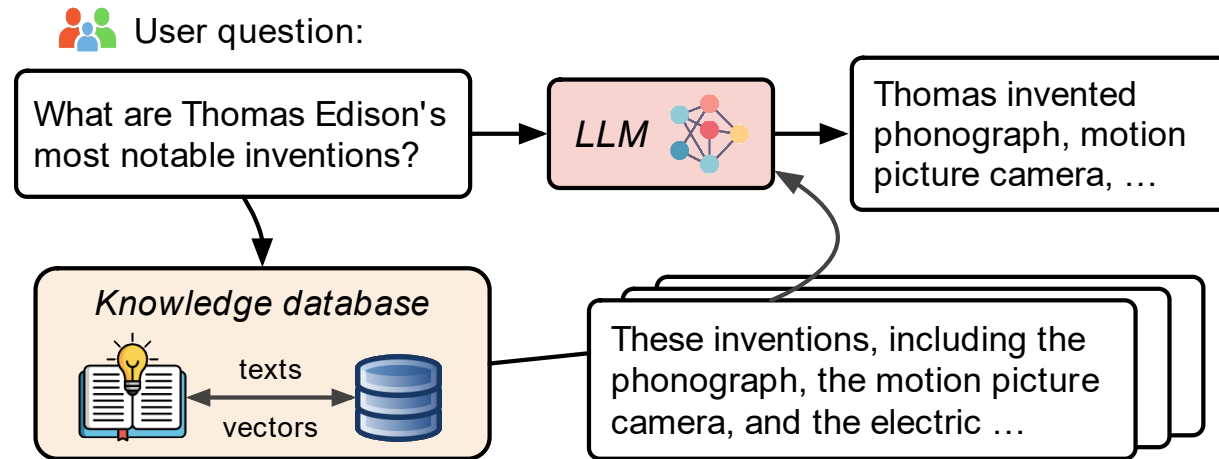
Key idea: pair LLMs with **retrievals from external databases**



# Retrieval-augmented generation (RAG)



# Retrieval-augmented generation (RAG)

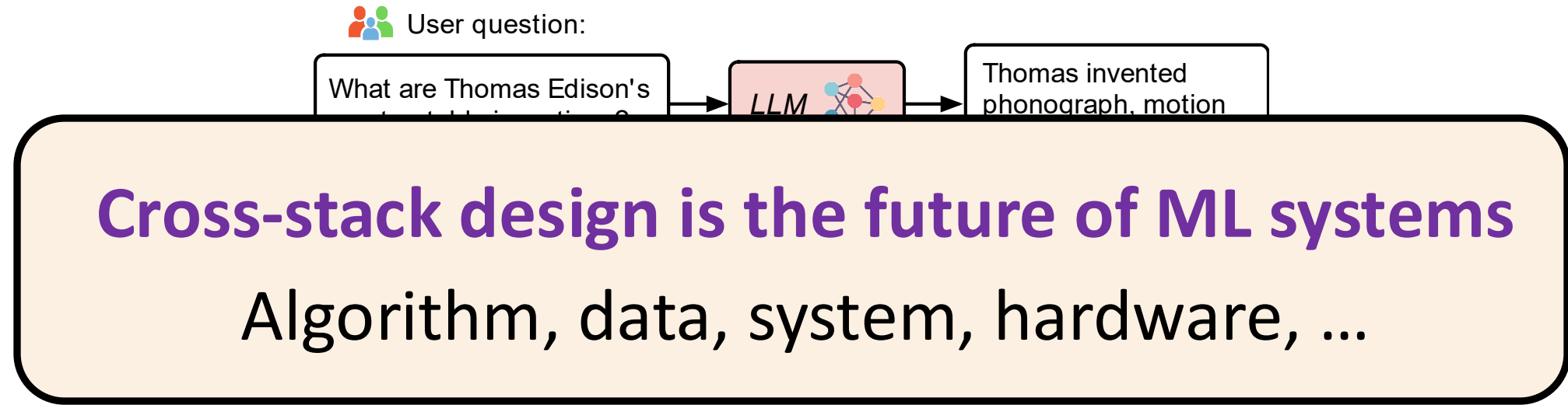


**Vector database and retrieval** play a key role in the pipeline

**Various RAG algorithms** of drastically different workload

**Multiple system components** on **heterogeneous hardware**

# Retrieval-augmented generation (RAG)

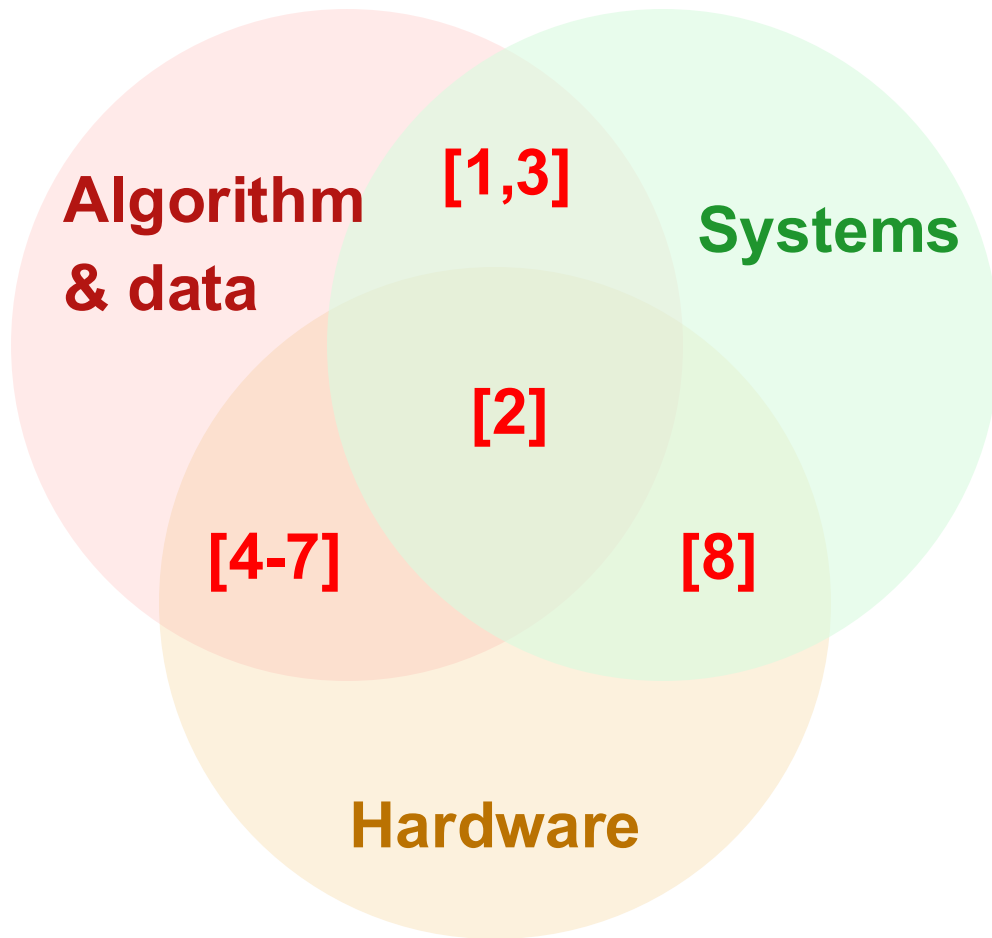


Vector **database** and retrieval play a key role in the pipeline

Various RAG **algorithms** of drastically different workload

Multiple **system** components on heterogeneous **hardware**

# My research: cross-stack, vector-centric ML systems



**Cross-stack design is the future:**  
Algorithm, data, system, hardware, ...

[1] RAGO [ISCA'25]

[2] Chameleon [VLDB'25]

[3] PipeRAG [KDD'25]

[4] FANNS [SC'23]

[5] Falcon [VLDB'25 (revision)]

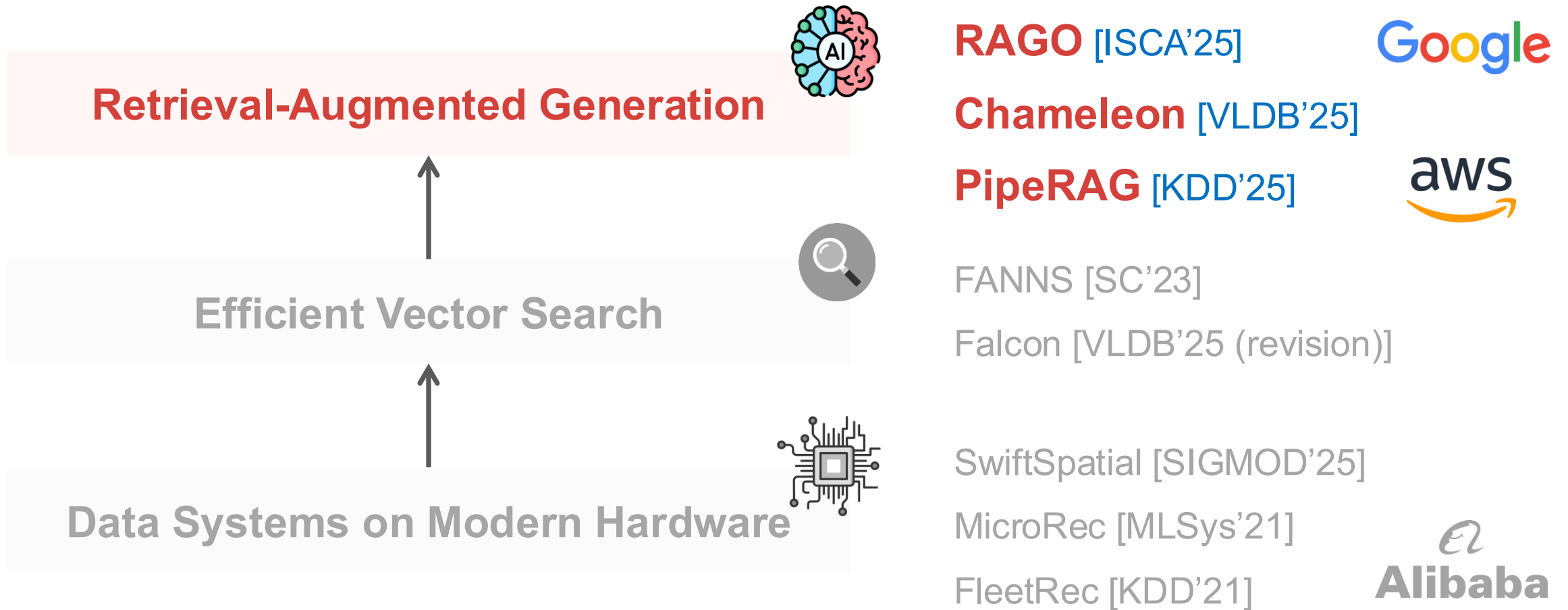
[6] SwiftSpatial [SIGMOD'25]

[7] MicroRec [MLSys'21]

[8] FleetRec [KDD'21]

Only first-author papers are listed

# My research: cross-stack, vector-centric ML systems

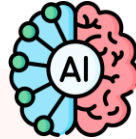


Only first-author papers are listed

# My research: cross-stack vector-centric ML systems

System for algorithms

Retrieval-Augmented Generation



**RAGO** [ISCA'25]

Google

Chameleon [VLDB'25]

PipeRAG [KDD'25]

aws

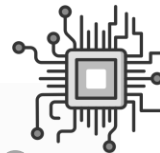
Efficient Vector Search



FANNS [SC'23]

Falcon [VLDB'25 (revision)]

Data Systems on Modern Hardware



SwiftSpatial [SIGMOD'25]

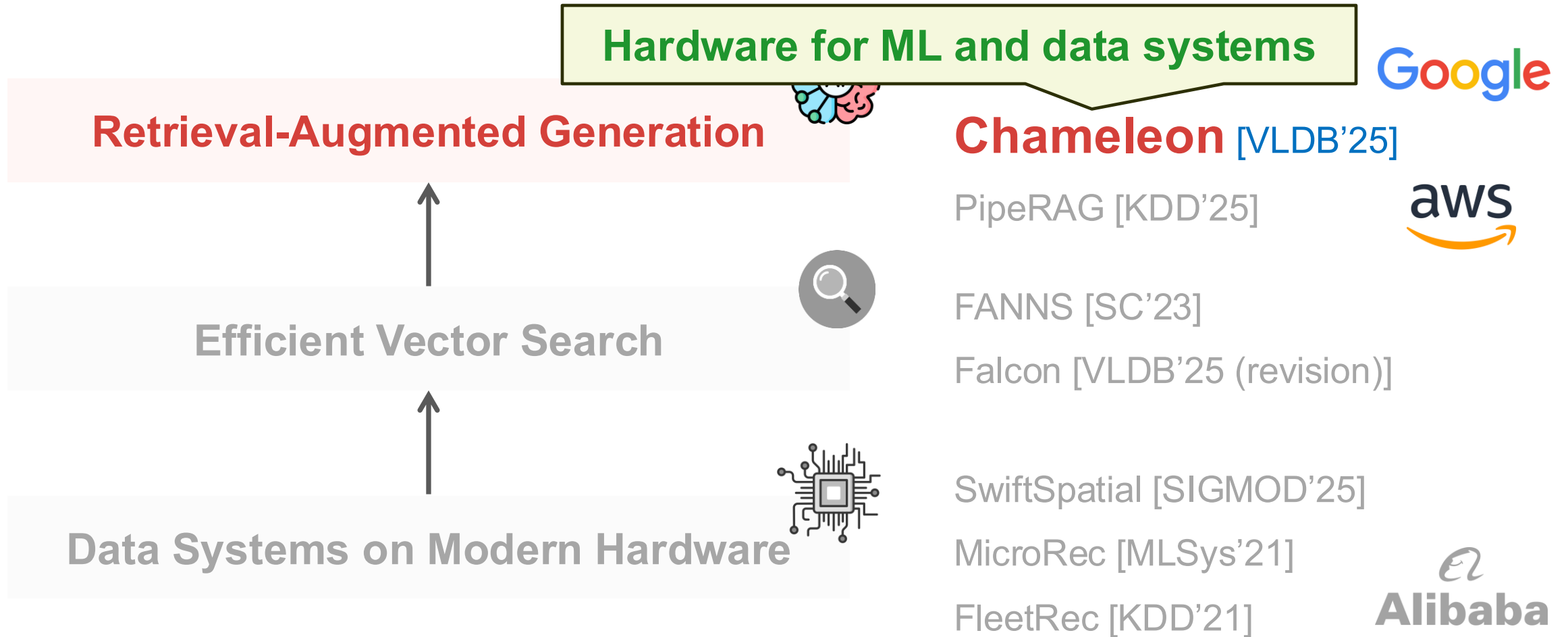
MicroRec [MLSys'21]

FleetRec [KDD'21]

Alibaba

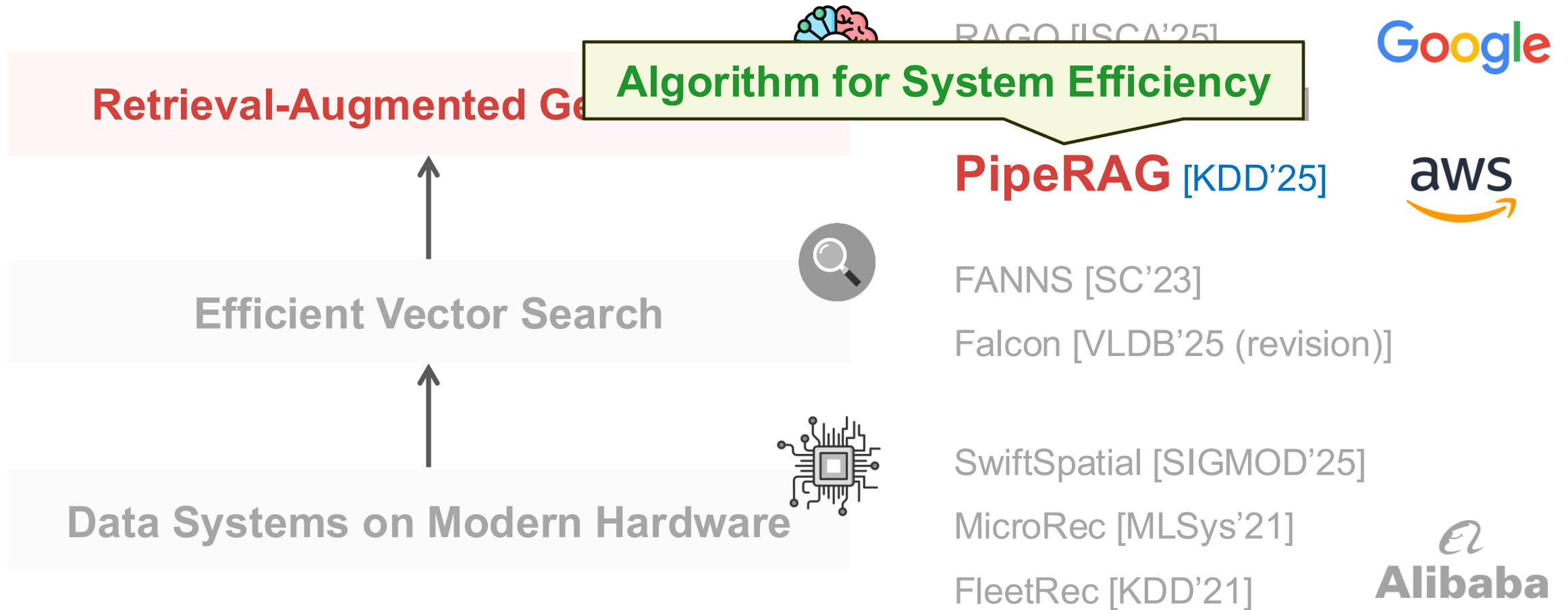
Only first-author papers are listed

# My research: cross-stack, vector-centric ML systems



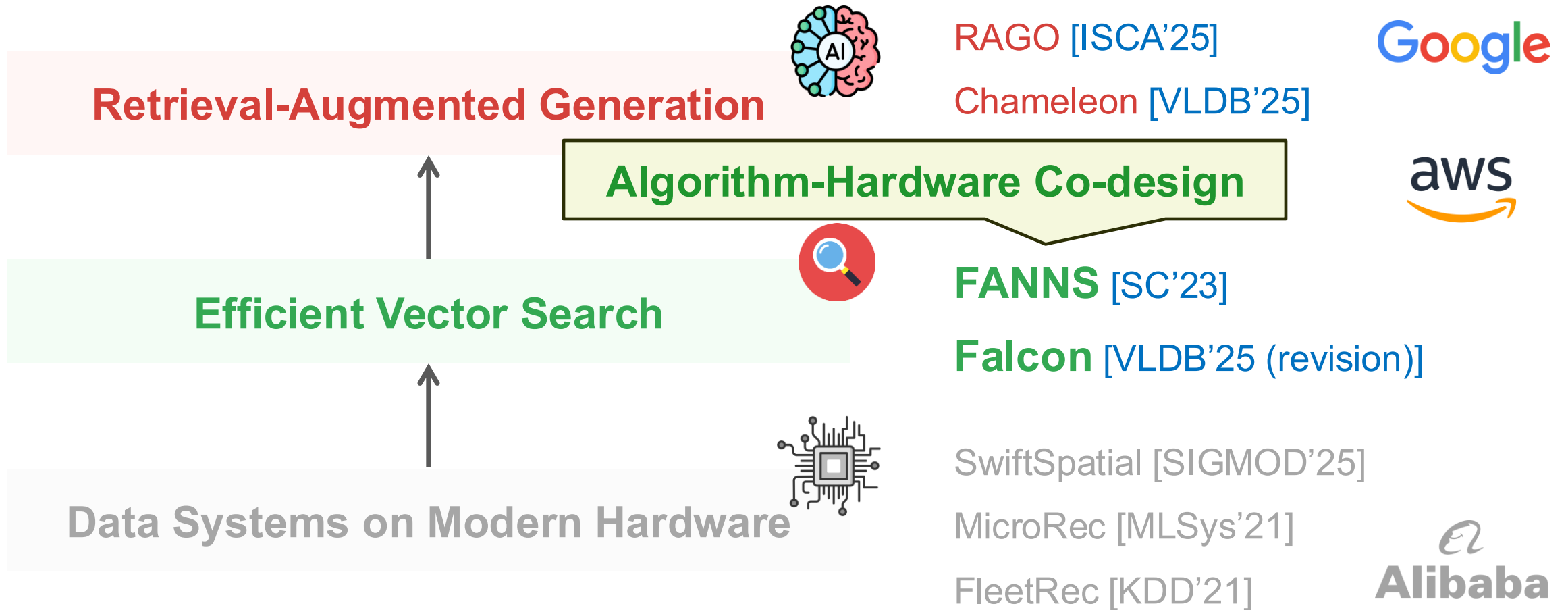
Only first-author papers are listed

# My research: cross-stack, vector-centric ML systems



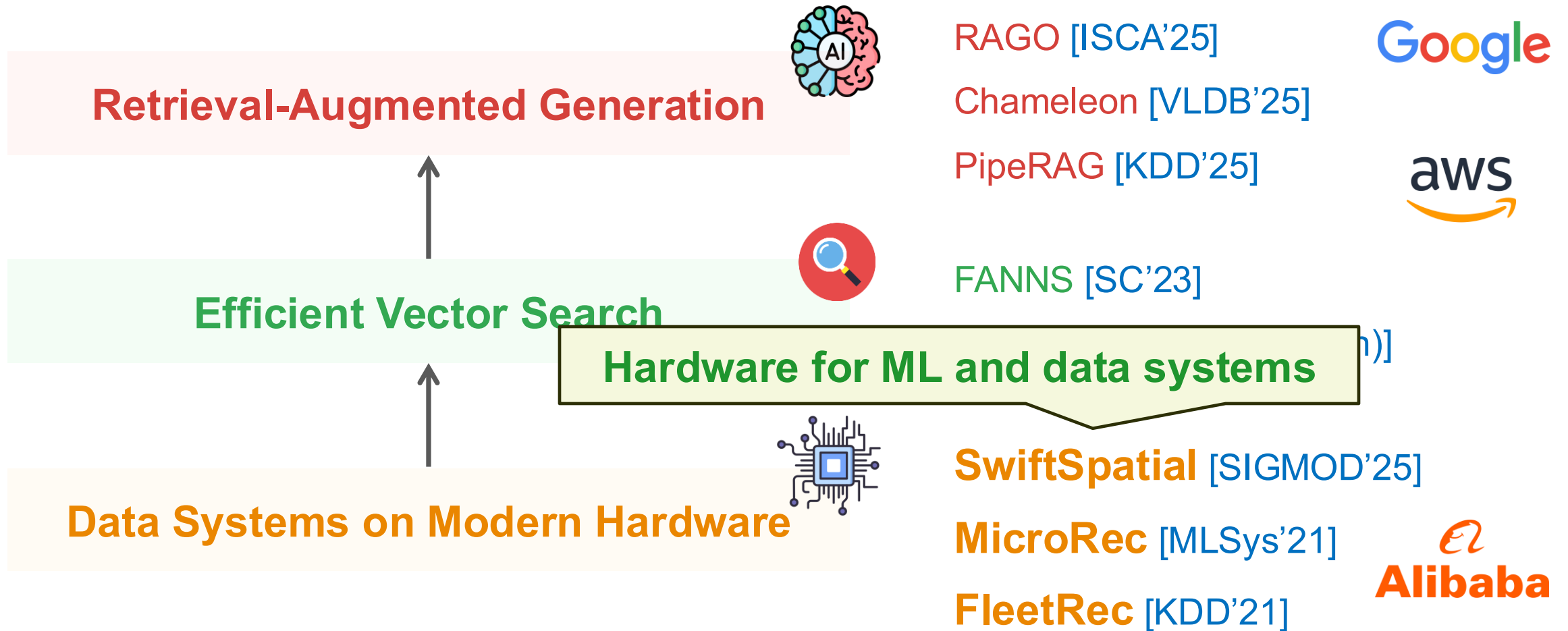
Only first-author papers are listed

# My research: cross-stack, vector-centric ML systems



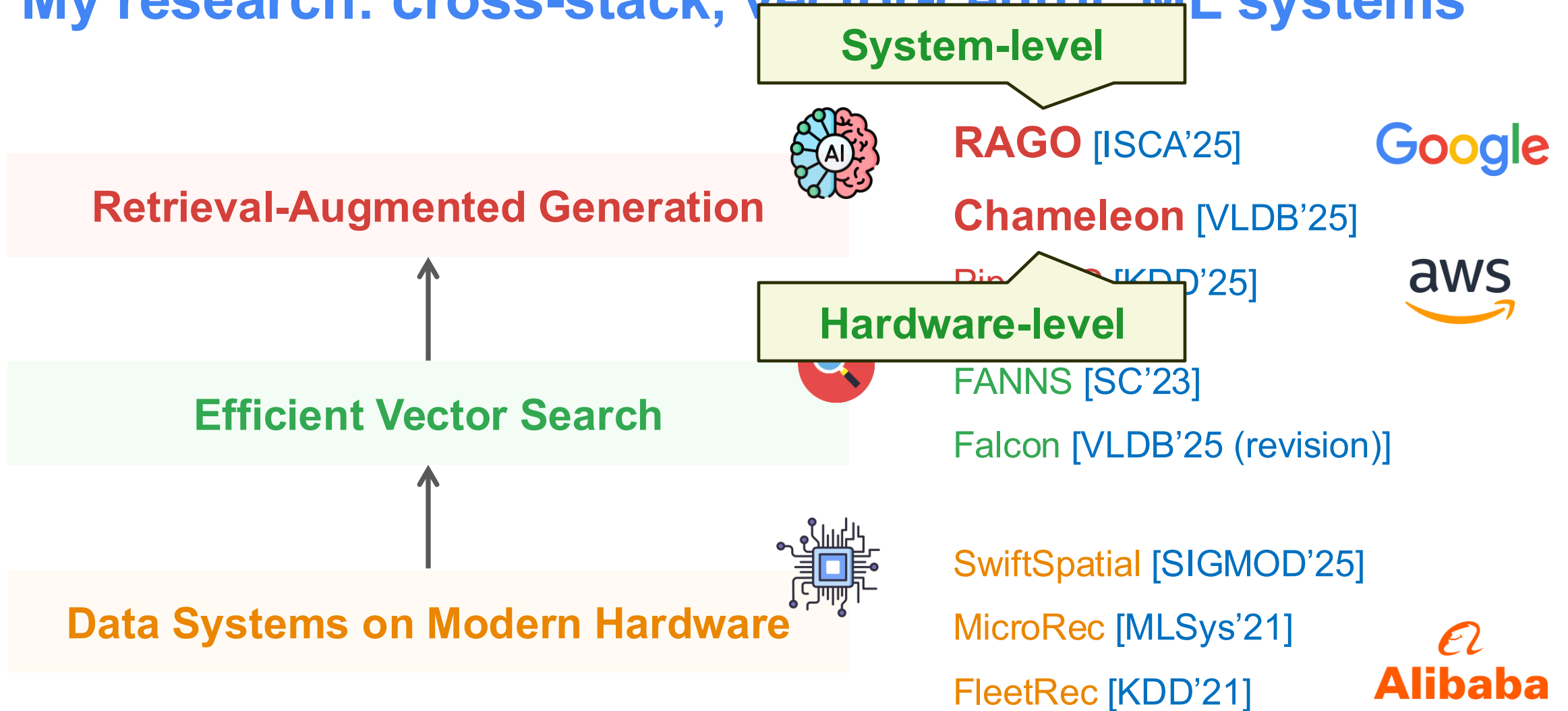
Only first-author papers are listed

# My research: cross-stack, vector-centric ML systems



Only first-author papers are listed

# My research: cross-stack, vector-centric ML systems



Only first-author papers are listed

# Presentation Outline

Overview: ML system efficiency is beyond model acceleration

My research: cross-stack, vector-centric ML systems

**RAGO: 1<sup>st</sup> systematic performance optimization for RAG**

Efficiently serving diverse and evolving RAG algorithms

Chameleon: 1<sup>st</sup> heterogeneous accelerator system for RAG

Explore hardware specialization for vector search

Future work: next-generation machine learning systems

Spanning algorithms, databases, systems, and hardware

# Optimizing RAG serving is challenging

## Many RAG algorithm variants, no clear sign of convergence

Published as a conference paper at ICLR 2020

REALM: Retrieval-Augmented Language Model Pre-Training

Kehin Om<sup>1</sup>, Kenton Lee<sup>2</sup>, Zora Tang<sup>1</sup>, Pansong Pang<sup>1</sup>, Ming-Wei Chang<sup>1</sup>

Abstract

Language model pre-training has been shown to capture a surprising amount of world knowledge, crucial for NLP tasks such as question answering. However, this knowledge is stored implicitly in the parameters of a neural network, requiring extra-large networks to cover more facts. To capture knowledge in a more modular and interpretable way, we augment language model pre-training with a latent knowledge retrieval, and allows the model to retrieve and attend over facts from a large corpus such as Wikipedia during pre-training. In our pre-training and inference, we show how to pre-train a knowledge retriever in an unsupervised manner using masked language modeling as the loss signal and backpropagating through a retrieval step that considers millions of documents. We demonstrate the effectiveness of Real-Augmented Language Model pre-training (REALM) by fine-tuning in the challenging of Open-domain Question Answering (OpenQA). We compare against state-of-the-art models both explicit and implicit knowledge models, these popular OpenQA benchmarks, and find we outperform all previous methods by a significant margin (4-16% absolute accuracy) while also providing qualitative benefits over interpretability and modularity.

1. Introduction

Recent advances in language model pre-training have shown that models such as BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019) and T5 (Raffel et al., 2019) “learn” continuous “Google Research Contributions to Knowledge” (C4) (google.github.io/knowledge-research-contributions/), Zora Tang (tangzora@google.com), Pansong Pang (pangpansong@google.com), Ming-Wei Chang (mchang@google.com).

Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, July 19-20, 2020. Copyright 2020 by the authors.

Published as a conference paper at ICLR 2020

GENERALIZATION THROUGH MEMORIZATION: NEAREST NEIGHBOR LANGUAGE MODELS

Urvashi Khambhwal<sup>1</sup>, Omor Levy<sup>1</sup>, Dan Jurafsky<sup>1</sup>, Luke Zettlemoyer<sup>1</sup> & Mike Lewis<sup>1</sup>

Abstract

We introduce ENN-LMs, which extend a pre-trained neural language model to incorporate nearest neighbor information. We show that this model can generalize to new tasks by leveraging the information stored in the nearest neighbor models. We demonstrate the effectiveness of ENN-LMs by fine-tuning on the challenging of Open-domain Question Answering (OpenQA). We compare against state-of-the-art models both explicit and implicit knowledge models, these popular OpenQA benchmarks, and find we outperform all previous methods by a significant margin (4-16% absolute accuracy) while also providing qualitative benefits over interpretability and modularity.

1. Introduction

Recent advances in language model pre-training have shown that models such as BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019) and T5 (Raffel et al., 2019) “learn” continuous “Google Research Contributions to Knowledge” (C4) (google.github.io/knowledge-research-contributions/), Zora Tang (tangzora@google.com), Pansong Pang (pangpansong@google.com), Ming-Wei Chang (mchang@google.com).

Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, July 19-20, 2020. Copyright 2020 by the authors.

Published as a conference paper at ICLR 2020

SELF-RAG: LEARNING TO RETRIEVE, GENERATE, AND CRITIQUE THROUGH SELF-REFLECTION

Akari Asai<sup>1</sup>, Zequn Wu<sup>1</sup>, Yisheng Wang<sup>1</sup>, Arjun SSB<sup>1</sup>, Hannaneh Hajishiraf<sup>1</sup>

Abstract

Despite their remarkable capabilities, large language models (LLMs) sometimes contain factual inaccuracies due to their reliance on knowledge they memorize. Retrieval-Augmented Generation (RAG) has been proposed to address this issue by retrieving relevant information from external knowledge sources. However, RAG often requires manual intervention to select relevant information, which is not scalable. In this paper, we propose SELF-RAG, a self-reflection framework that enables LLMs to learn to retrieve, generate, and critique their own outputs. We demonstrate the effectiveness of SELF-RAG by fine-tuning on the challenging of Open-domain Question Answering (OpenQA). We compare against state-of-the-art models both explicit and implicit knowledge models, these popular OpenQA benchmarks, and find we outperform all previous methods by a significant margin (4-16% absolute accuracy) while also providing qualitative benefits over interpretability and modularity.

1. Introduction

Recent advances in language model pre-training have shown that models such as BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019) and T5 (Raffel et al., 2019) “learn” continuous “Google Research Contributions to Knowledge” (C4) (google.github.io/knowledge-research-contributions/), Zora Tang (tangzora@google.com), Pansong Pang (pangpansong@google.com), Ming-Wei Chang (mchang@google.com).

Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, July 19-20, 2020. Copyright 2020 by the authors.

Published as a conference paper at ICLR 2020

Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions

Harsh Trivedi<sup>1</sup>, Niranjan Balasubramanian<sup>1</sup>, Tushar Khot<sup>1</sup>, Ashish Sabharwal<sup>1</sup>

Abstract

Prompting-based large language models (LLMs) are surprisingly powerful at generating natural language reasoning steps as part of their output. However, these models often struggle to maintain consistency across long reasoning chains, leading to errors in the final answer. In this paper, we propose a framework that interleaves retrieval with chain-of-thought reasoning to improve the consistency and accuracy of LLMs on knowledge-intensive multi-step questions. We demonstrate the effectiveness of our framework by fine-tuning on the challenging of Open-domain Question Answering (OpenQA). We compare against state-of-the-art models both explicit and implicit knowledge models, these popular OpenQA benchmarks, and find we outperform all previous methods by a significant margin (4-16% absolute accuracy) while also providing qualitative benefits over interpretability and modularity.

1. Introduction

Recent advances in language model pre-training have shown that models such as BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019) and T5 (Raffel et al., 2019) “learn” continuous “Google Research Contributions to Knowledge” (C4) (google.github.io/knowledge-research-contributions/), Zora Tang (tangzora@google.com), Pansong Pang (pangpansong@google.com), Ming-Wei Chang (mchang@google.com).

Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, July 19-20, 2020. Copyright 2020 by the authors.

# Drastically different workload characteristics

Figure 1: Datascale scaling improves language modelling and downstream task performance. Left: Datascale scaling performance on language modelling and a downstream task (NLM11) with LLaMA-2 and LLaMA-3 models. Right: Compute-optimized scaling of retrieval-based language models vs. LLaMA-3 models with PyTriton. By considering the size of the dataset as an additional dimension of scaling, we can improve model performance at lower training cost.

18th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada

# Case study 1: RAG with hyper-scale retrieval



Argument: **Smaller model + hyper-scale retrieval = Larger model**

10x model size saving given similar generation quality [1,2]



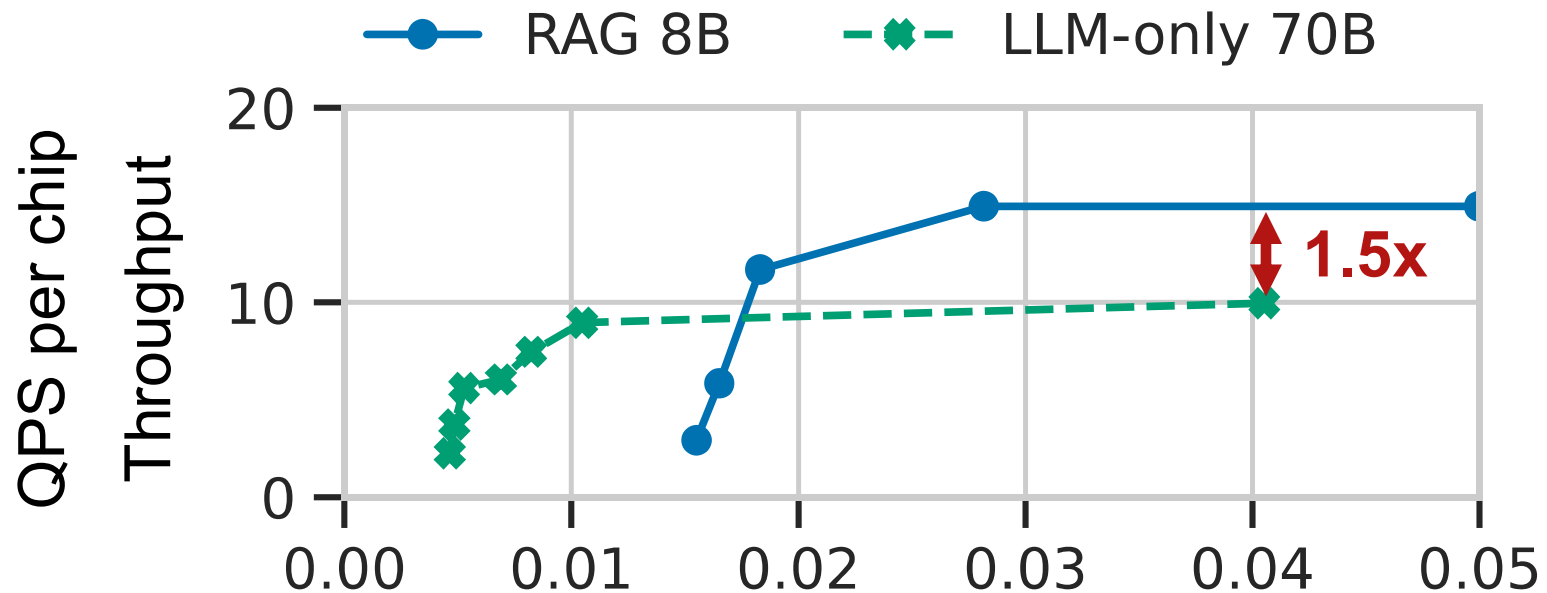
Internet-scale corpus with two trillion tokens according to DeepMind [1]

[1] Borgeaud et al. "Improving Language Models by Retrieving from Trillions of Tokens", 2022

[2] Wang et al. "InstructRetro: Instruction Tuning Post Retrieval-Augmented Retraining", 2023

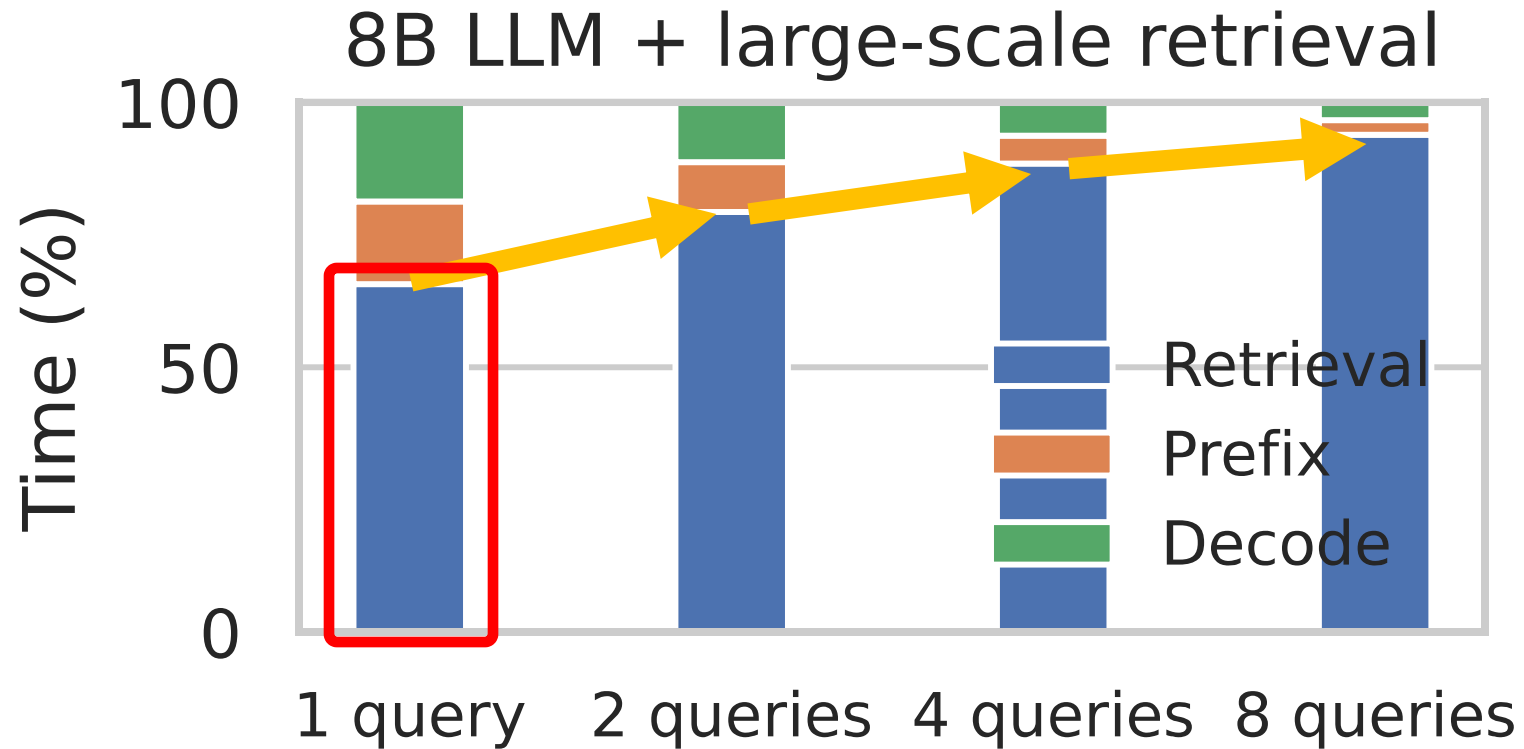
# Case study 1: RAG with hyper-scale retrieval

RAG with smaller models achieve **better QPS/chip** than larger LLMs  
**10x size difference** (RAG-8B vs LLM-70B) **but only 1.5x speedup**



**RAG overhead: (1) longer prompts and (2) hyper-scale retrieval**

# Case study 1: RAG with hyper-scale retrieval



**Hyper-scale retrieval can be a major bottleneck  
(2<sup>nd</sup> half of this talk addresses retrieval performance)**

# Case study 2: RAG for long-context processing

Answering questions of **user-defined long context** in real-time



**Naive solution:** include documents in the prompt (e.g., 1M tokens)

Possible but **very costly**, e.g., 60 USD / million token for GPT4

**RAG solution:** retrieve relevant passages

Significant **lower cost with comparable quality** [1,2]

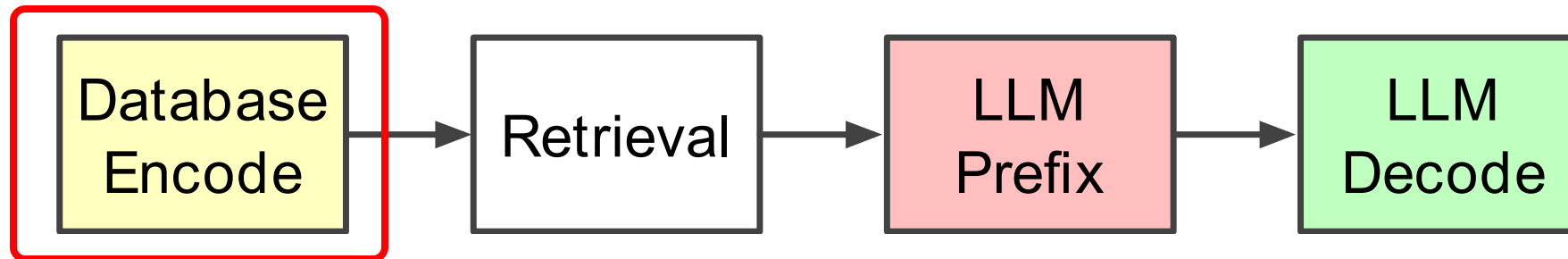
[1] Lee et al. "Can Long-Context Language Models Subsume Retrieval, RAG, SQL, and More?", 2024

[2] Yue et al. "Inference Scaling for Long-Context Retrieval Augmented Generation", 2024

## Case study 2: RAG for long-context processing

Divide the document into many passages

**Encode each passage** into a vector using a BERT-style model

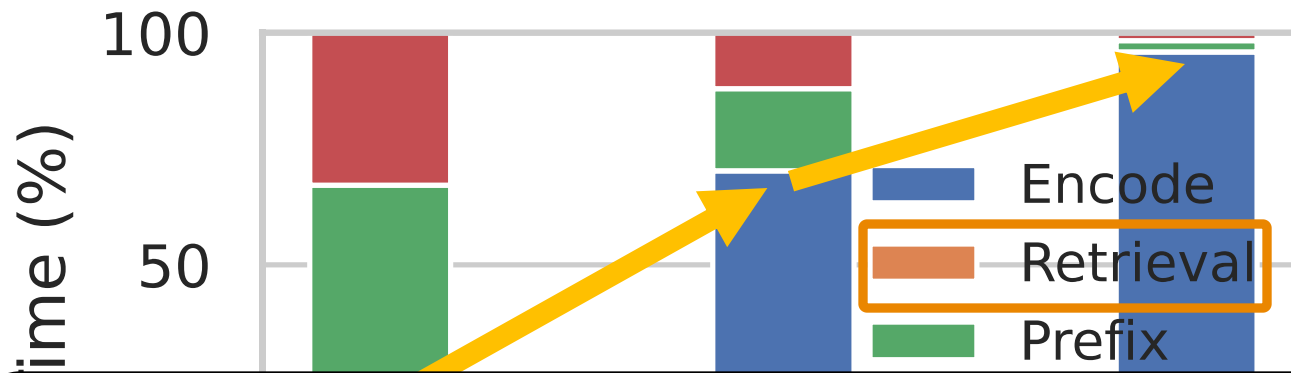


**Small model** (e.g., 100M~1B) + **small databases** (1K~1M vectors)

[1] Lee et al. "Can Long-Context Language Models Subsume Retrieval, RAG, SQL, and More?", 2024

[2] Yue et al. "Inference Scaling for Long-Context Retrieval Augmented Generation", 2024

## Case study 2: RAG for long-context processing



**Encoder (120M) << LLM (70B)**

Encoder 500x smaller

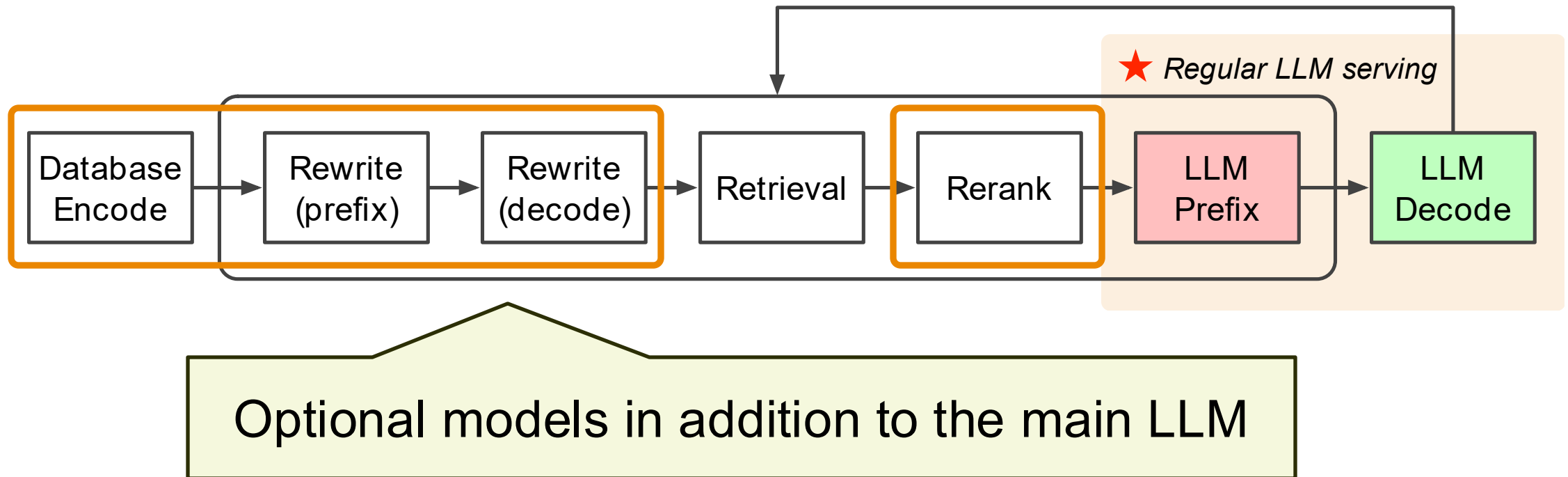
**Drastically different workloads across RAG algorithms**

Document lengths (tokens)

1. Even a small encoder model can become the bottleneck
2. Retrieval performance does not matter even with brute-force scan

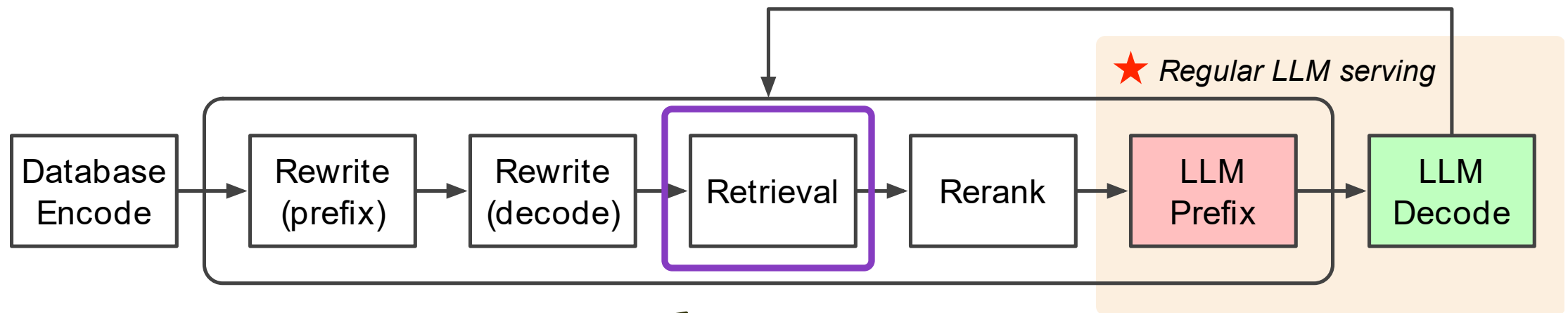
# RAGSchema: workload abstraction for RAG algorithms

RAGSchema = **Model components** + **Retrieval configurations**



# RAGSchema: workload abstraction for RAG algorithms

RAGSchema = **Model components** + **Retrieval configurations**

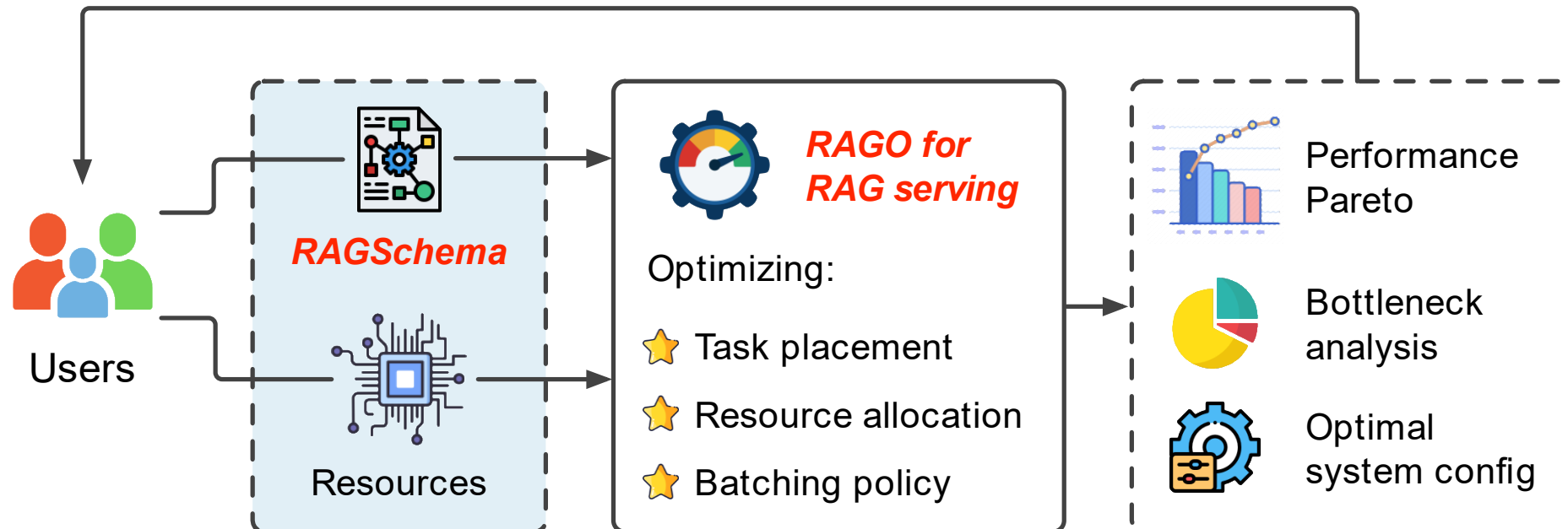


Database sizes; multi-query retrievals; iterative retrievals

# RAGO: Retrieval-Augmented Generation Optimizer

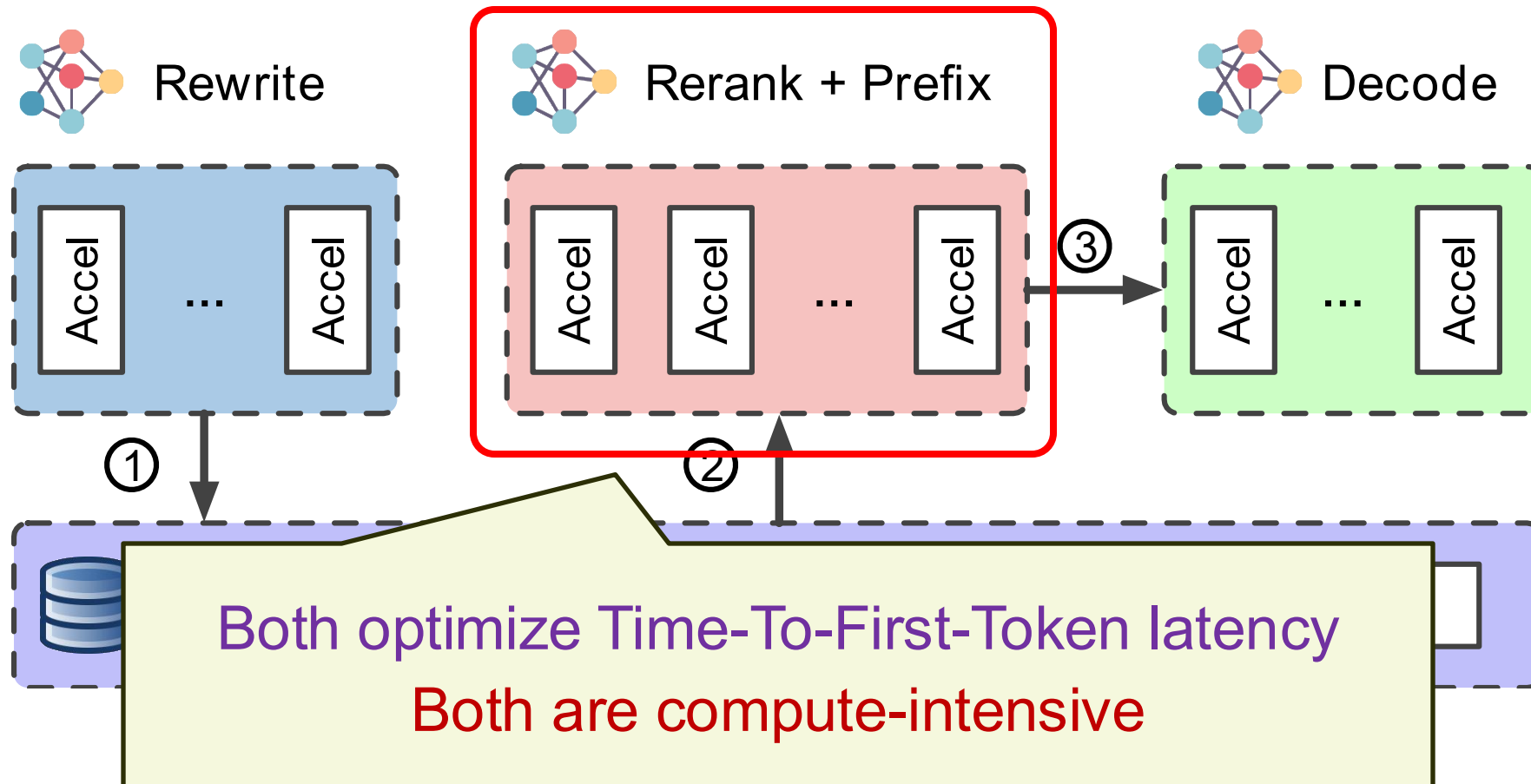
Inputs: **RAGSchema + Hardware resources**

Outputs: **Optimal performance + System configurations**



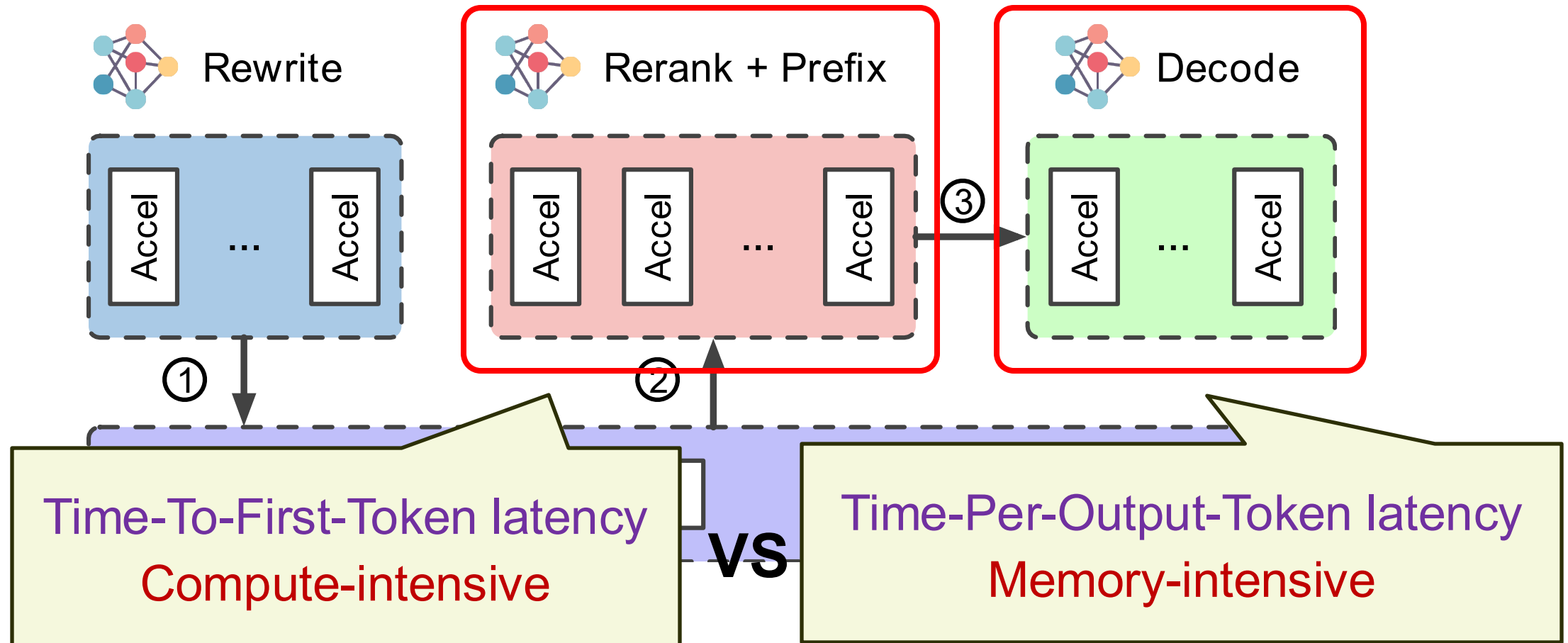
# RAGO system design space

**Task placement** + Resource allocation + Batching



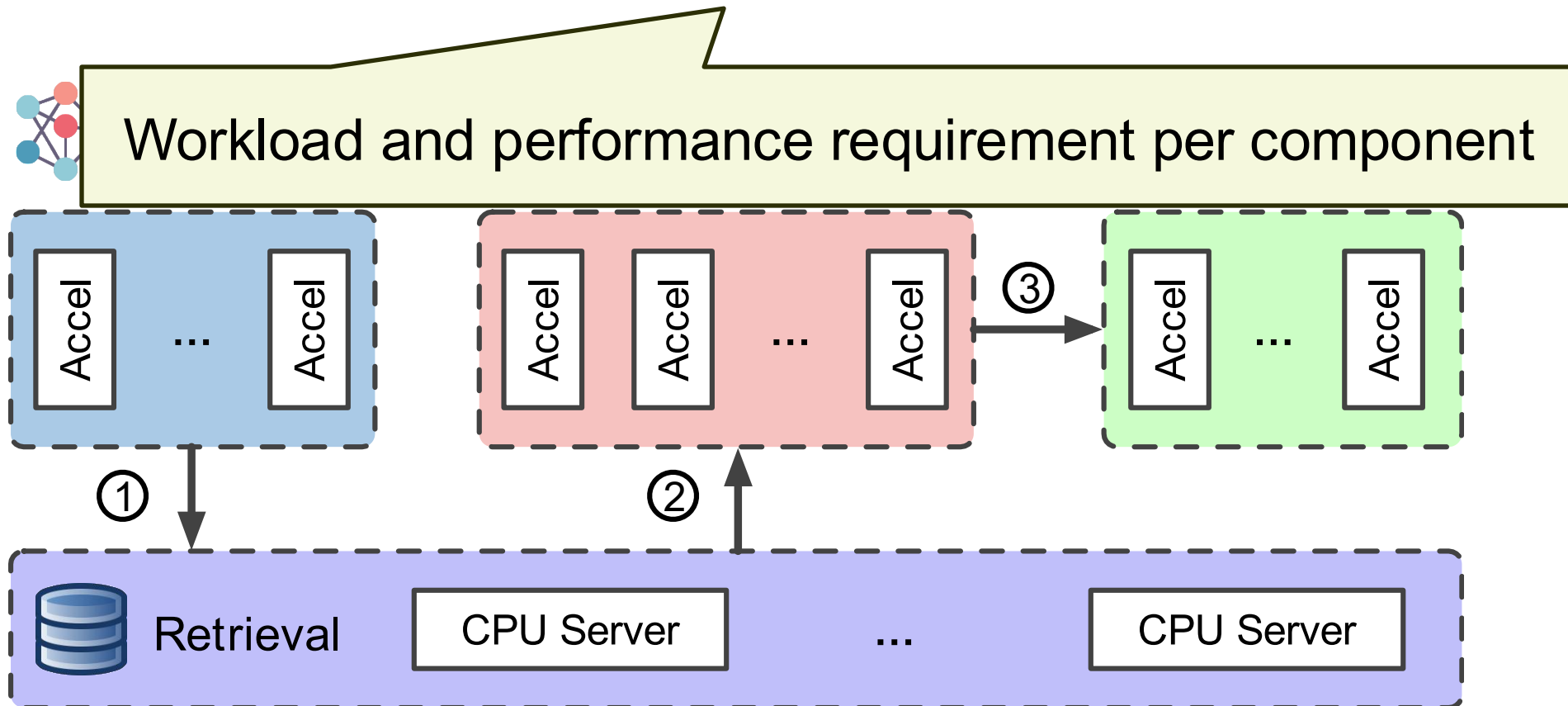
# RAGO system design space

**Task placement** + Resource allocation + Batching



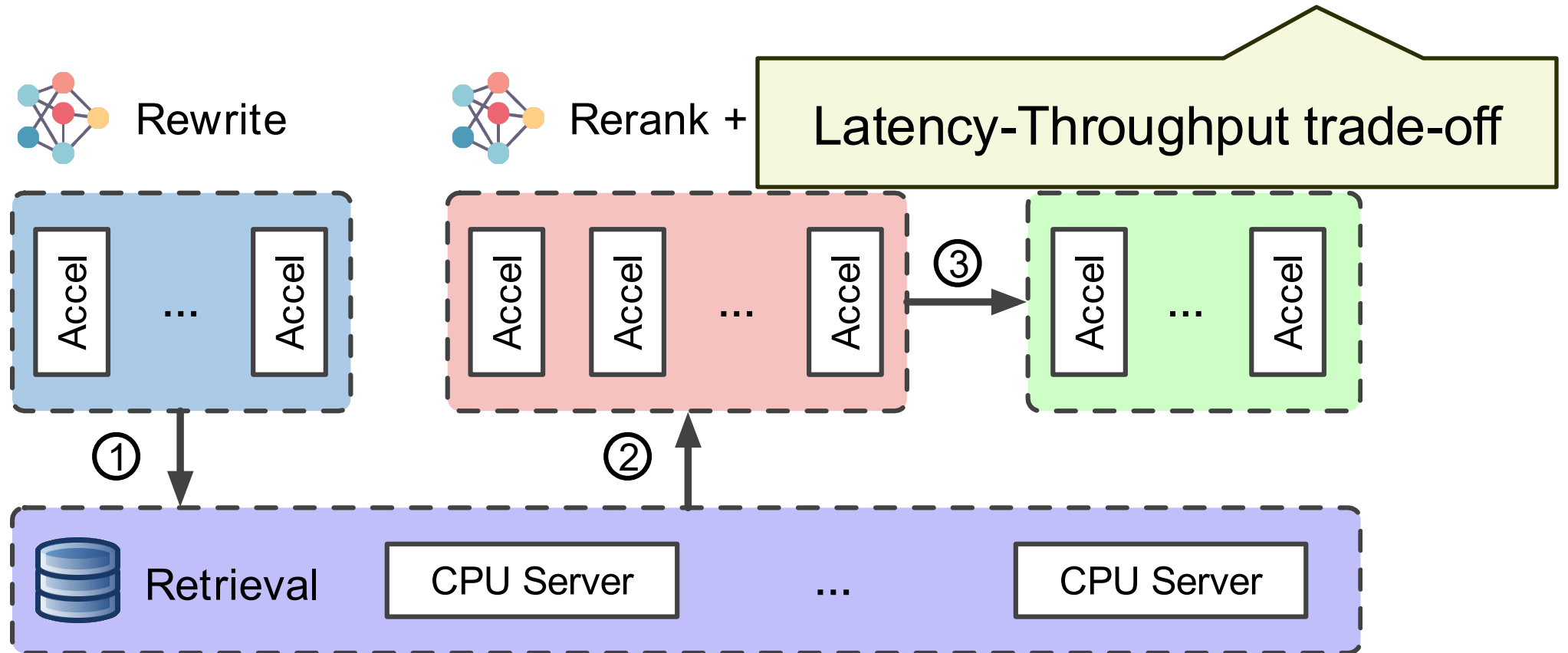
# RAGO system design space

Task placement + **Resource allocation** + Batching



# RAGO system design space

Task placement + Resource allocation + **Batching**



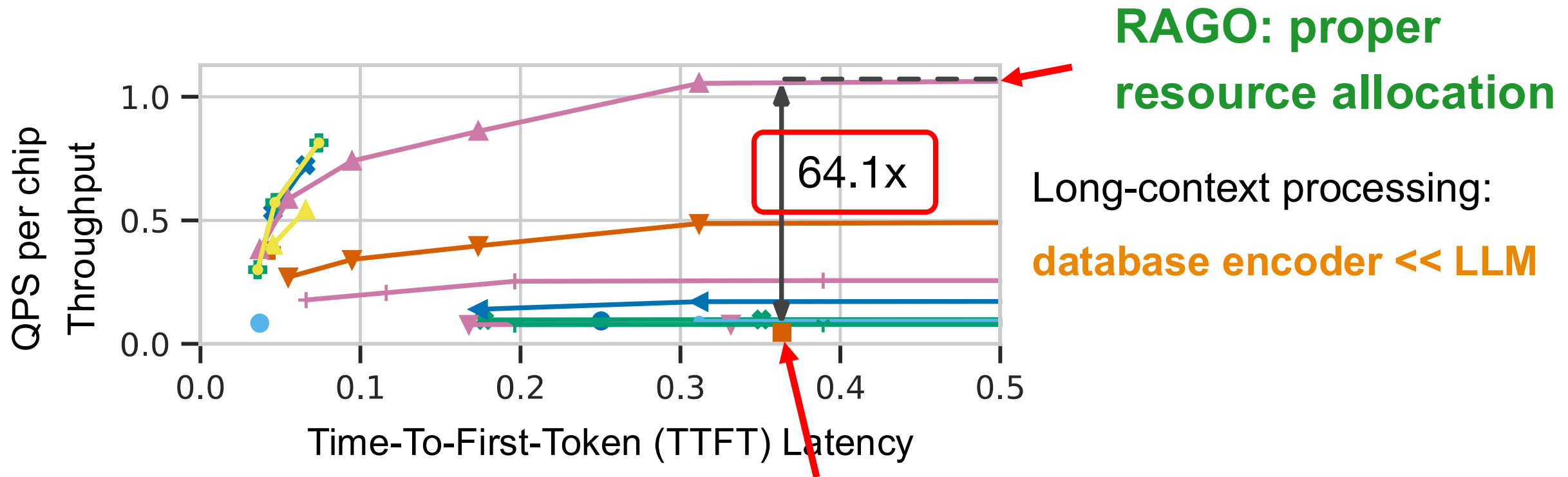
# Finding optimal schedules in RAGO

## RAGO: cost-model-based system design space exploration

1. **Inference cost model**
  2. **Retrieval cost model**
  3. **RAG cost assembler** to evaluate end-to-end performance
- } Well-tuned roofline models
- a) Calculate performance Pareto per RAG component
  - b) Explore schedule combinations between components

# Evaluation: performance of various schedules

Each curve is a resource allocation plan with various batch sizes:



# **RAGO: 1<sup>st</sup> systematic RAG serving optimization**

## **Characterizing performance across RAG paradigms**

Drastically different performance characteristics

## **RAGSchema: RAG workload abstraction**

Unified representation for various RAG algorithms

## **RAGO: cost-model-based performance optimization**

Optimize placement, allocation, and batching policies

# Presentation Outline

**Overview: ML system efficiency is beyond model acceleration**

My research: cross-stack, vector-centric ML systems

**RAGO: 1<sup>st</sup> systematic performance optimization for RAG**

Efficiently serving diverse and evolving RAG algorithms

**Chameleon: 1<sup>st</sup> heterogeneous accelerator system for RAG**

Explore hardware specialization for vector search

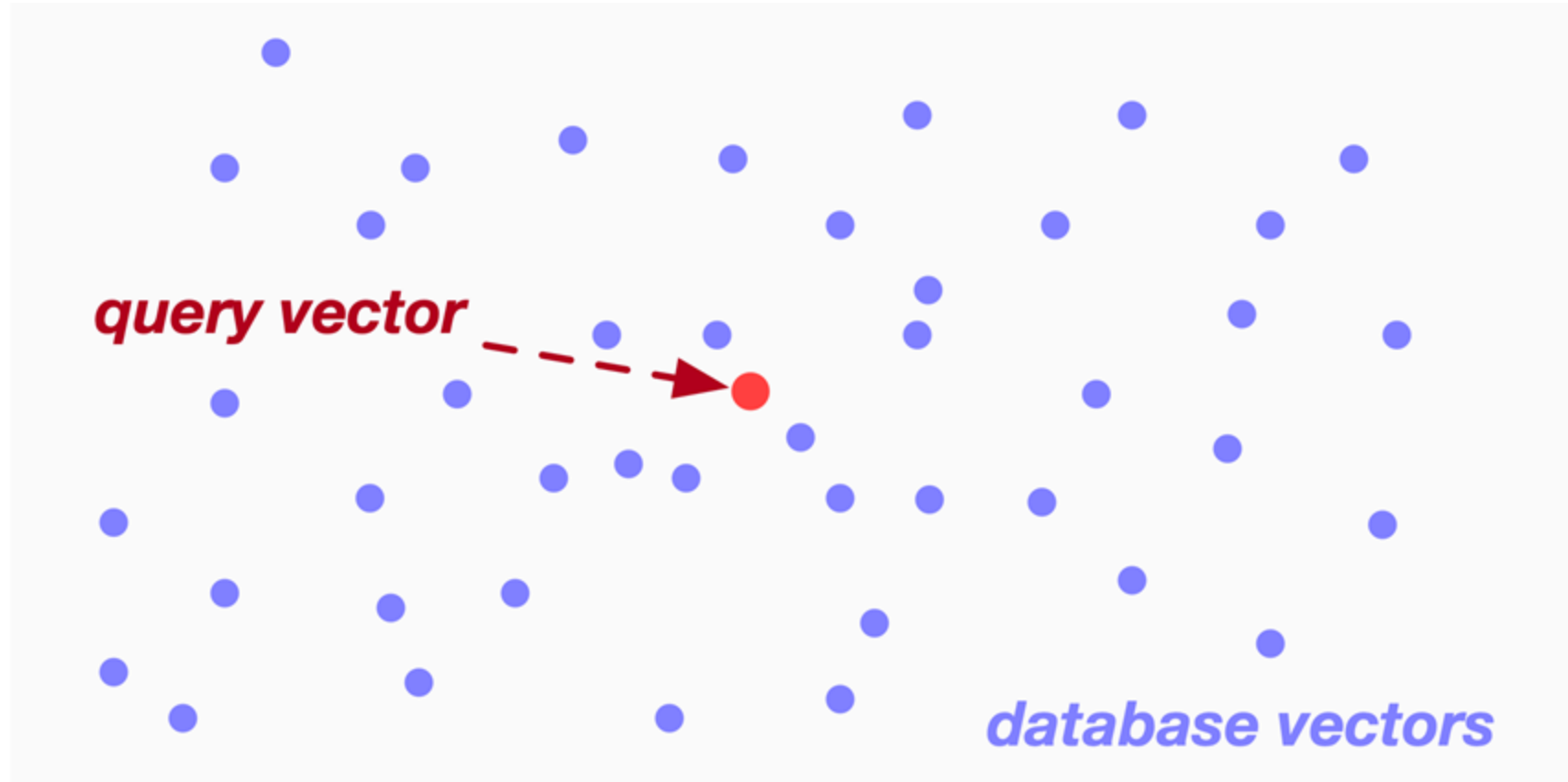
**Future work: next-generation machine learning systems**

Spanning algorithms, databases, systems, and hardware

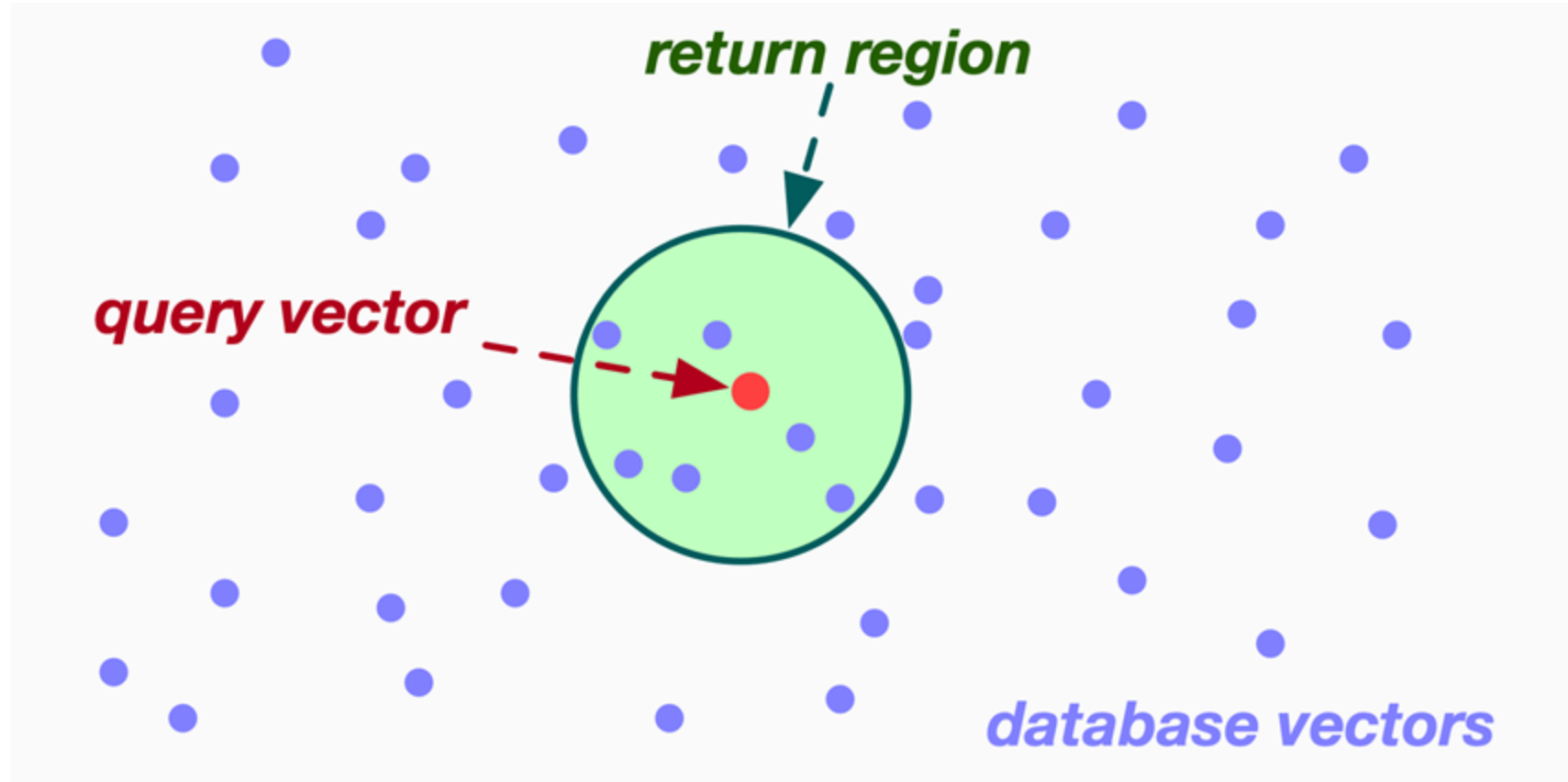
# Vector search: problem definition



# Vector search: problem definition



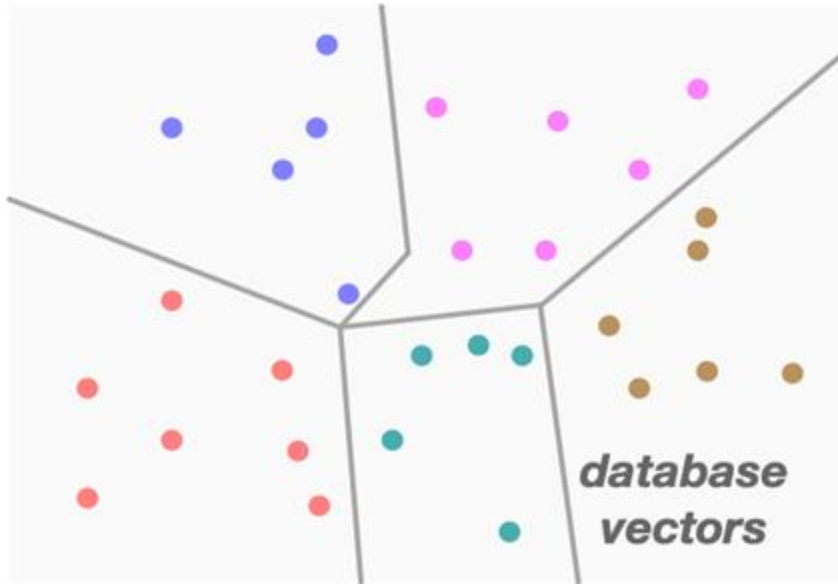
# Vector search: problem definition



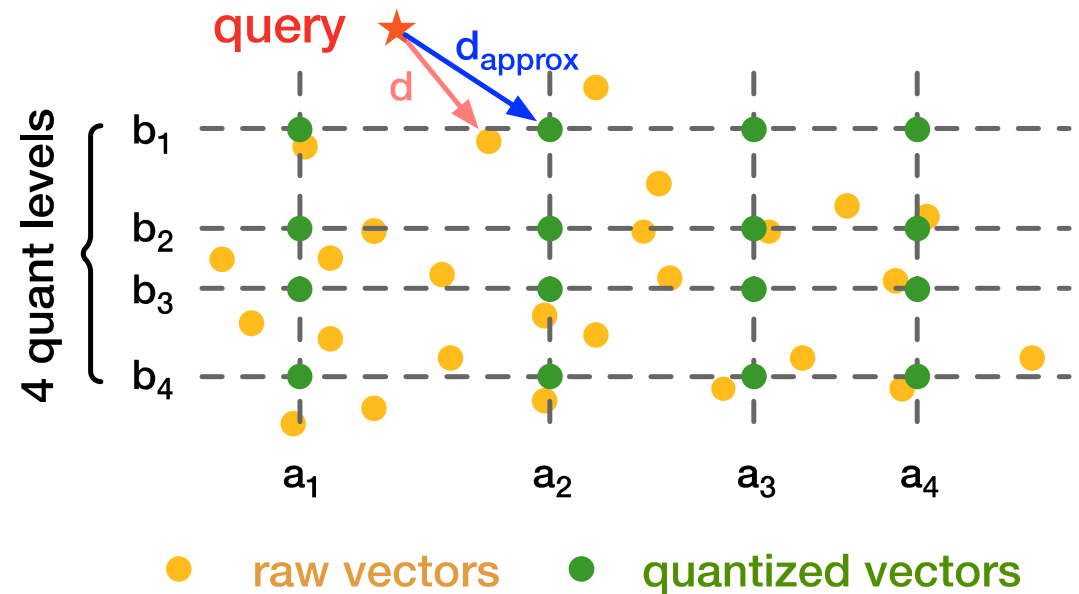
# Vector search $\approx$ approximate nearest neighbor search

**IVF-PQ:** a popular vector search algorithm in RAG

**Inverted-file (IVF) index:**  
**prune** the search space



**Product quantization (PQ):**  
**lossy compression** of vectors



# Large-scale vector search on existing systems

**Ideal system: sufficient memory capacity + fast PQ decoding**

Decode: each byte code involves two fetch operations

**CPU: too slow for PQ decoding** 

Intensive table lookup operations overload the cache

Low throughput of 1~1.5 GB/s per core

**GPU: prohibitively expensive at scale** 

Limited High-Bandwidth Memory (HBM) capacity

Energy wasted by idle compute units

# Hardware specialization is increasingly popular

Compute

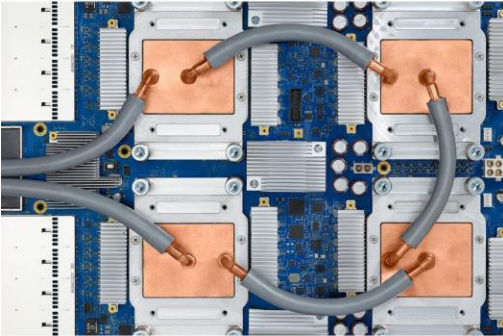
Announcing Trillium, the sixth generation of  
Google Cloud TPU

May 15, 2024

 Meta

 aws

 Microsoft



It's time to think about retrieval acceleration

# Proposed RAG system design principles

**Requirement:** fast inference + fast vector search

**Principle 1: accelerator heterogeneity**

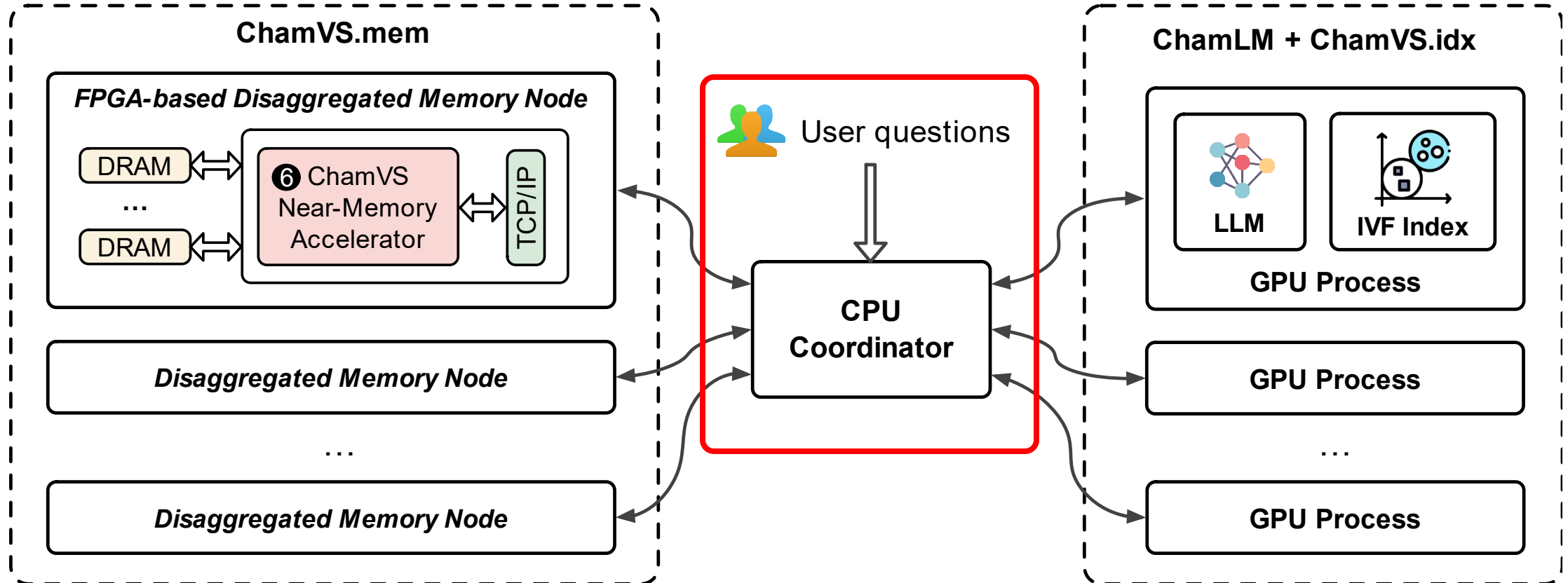
Inference accelerators + vector search accelerators

**Requirement:** accommodate diverse RAG algorithms

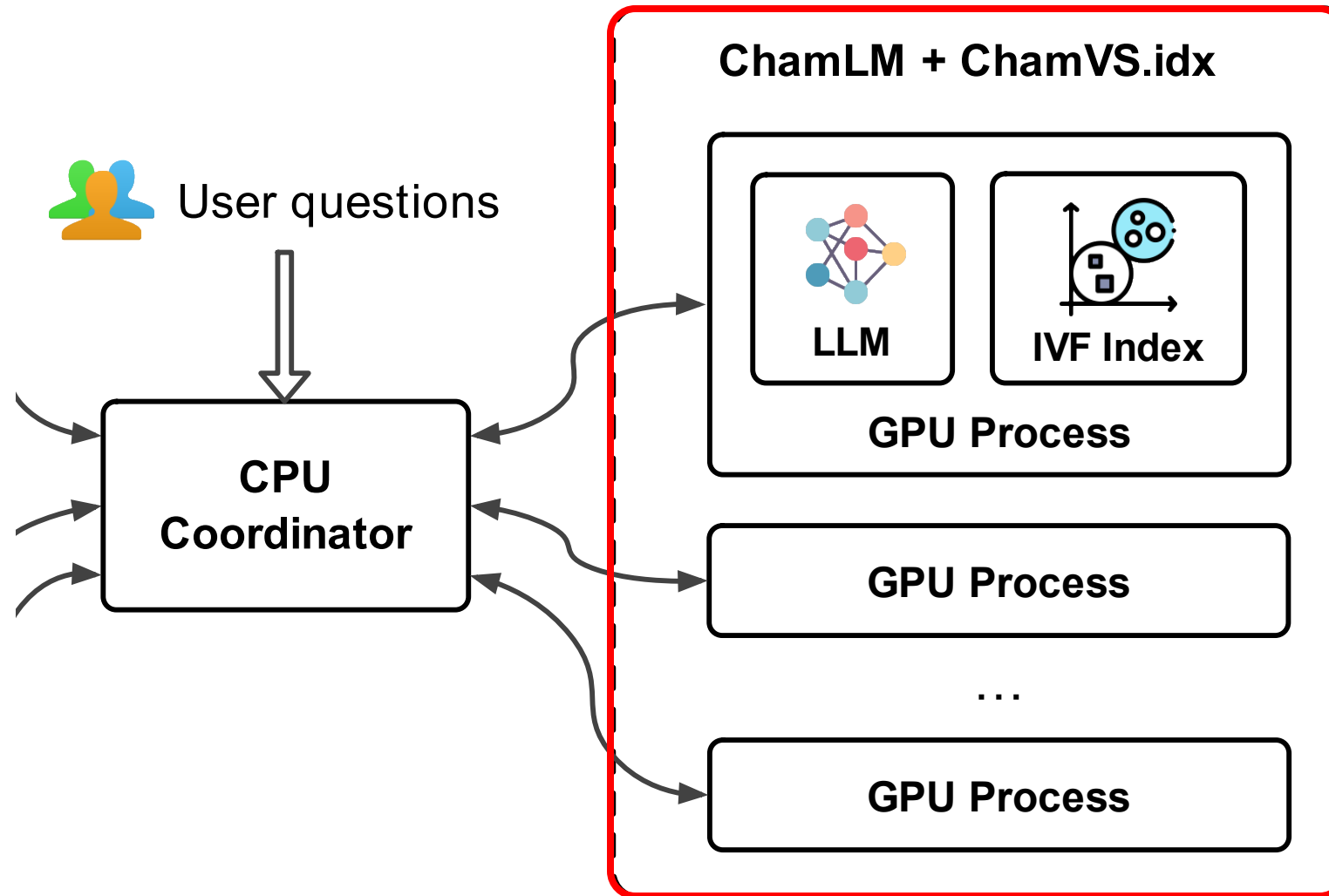
**Principle 2: accelerator disaggregation**

Handle various performance bottlenecks across RAGs

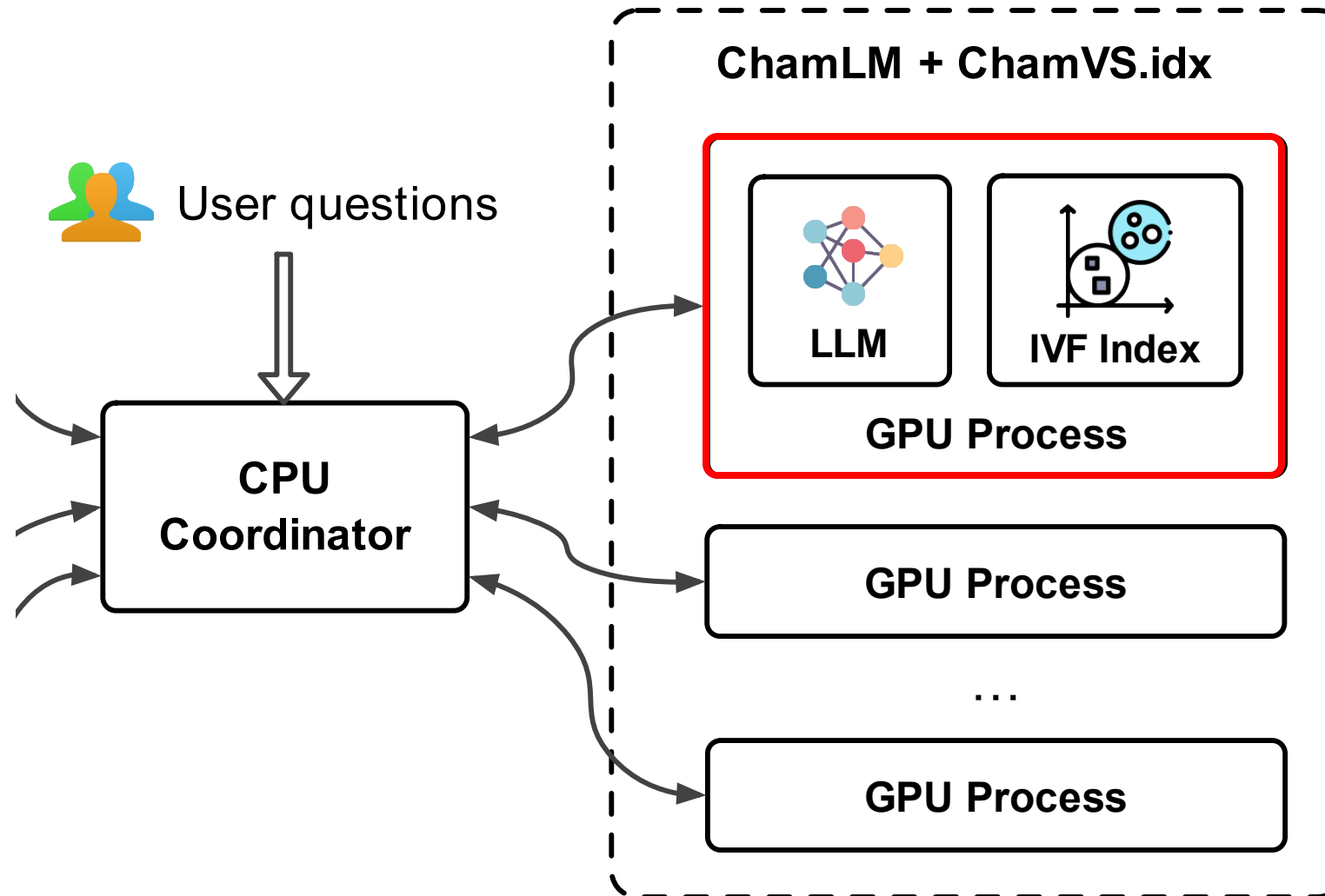
# Chameleon: accelerator heterogeneity + disaggregation



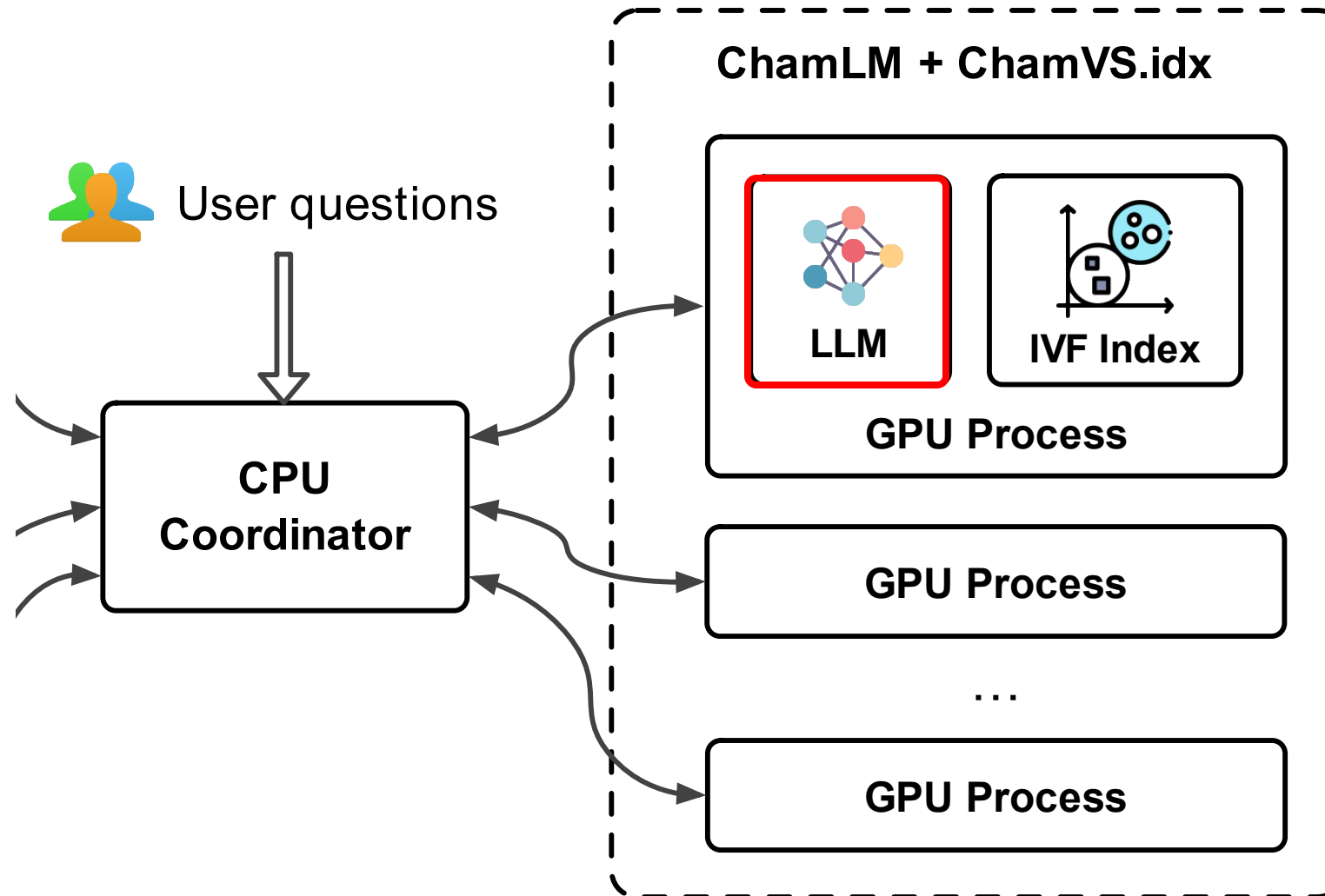
# Chameleon: accelerator heterogeneity + disaggregation



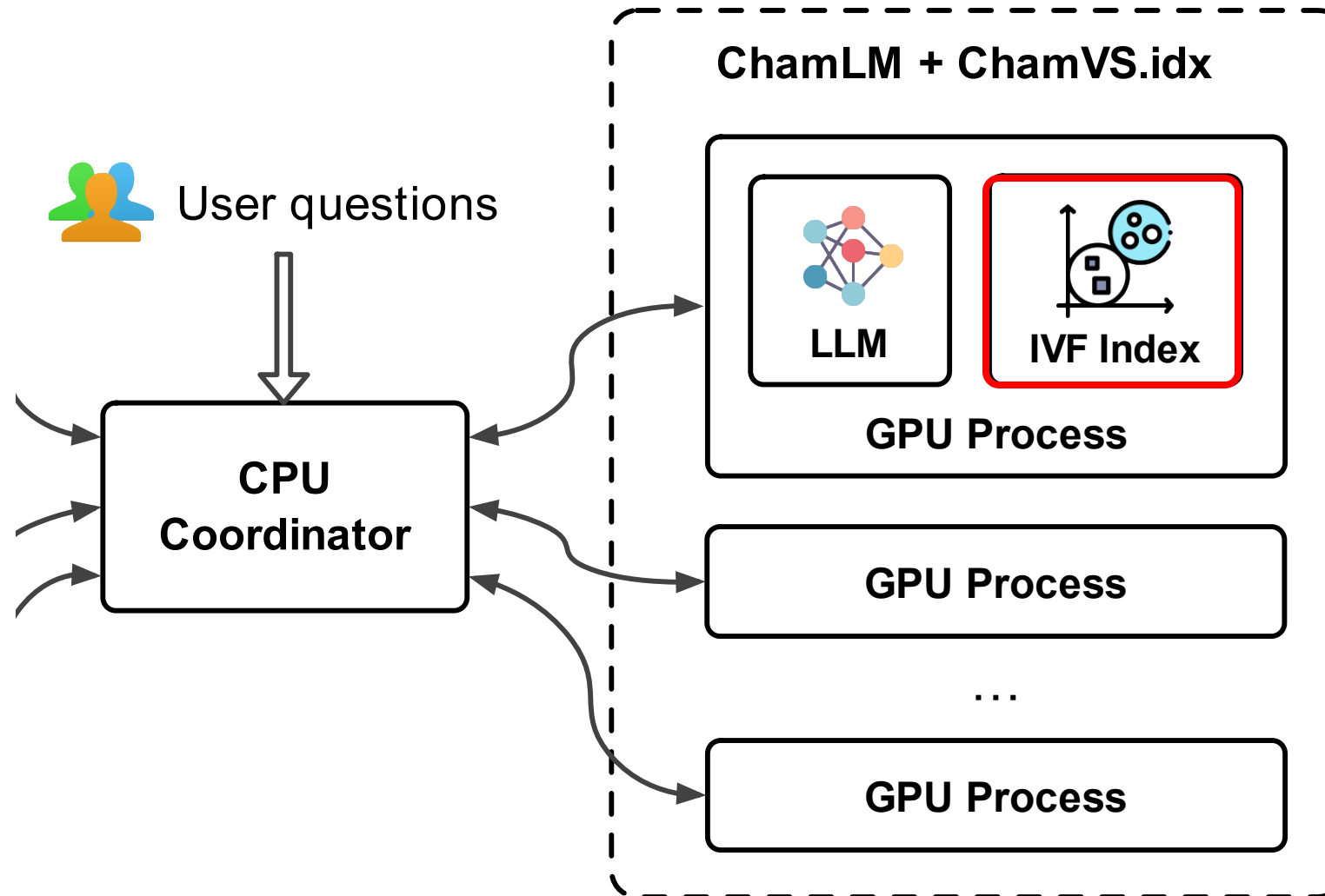
# Chameleon: accelerator heterogeneity + disaggregation



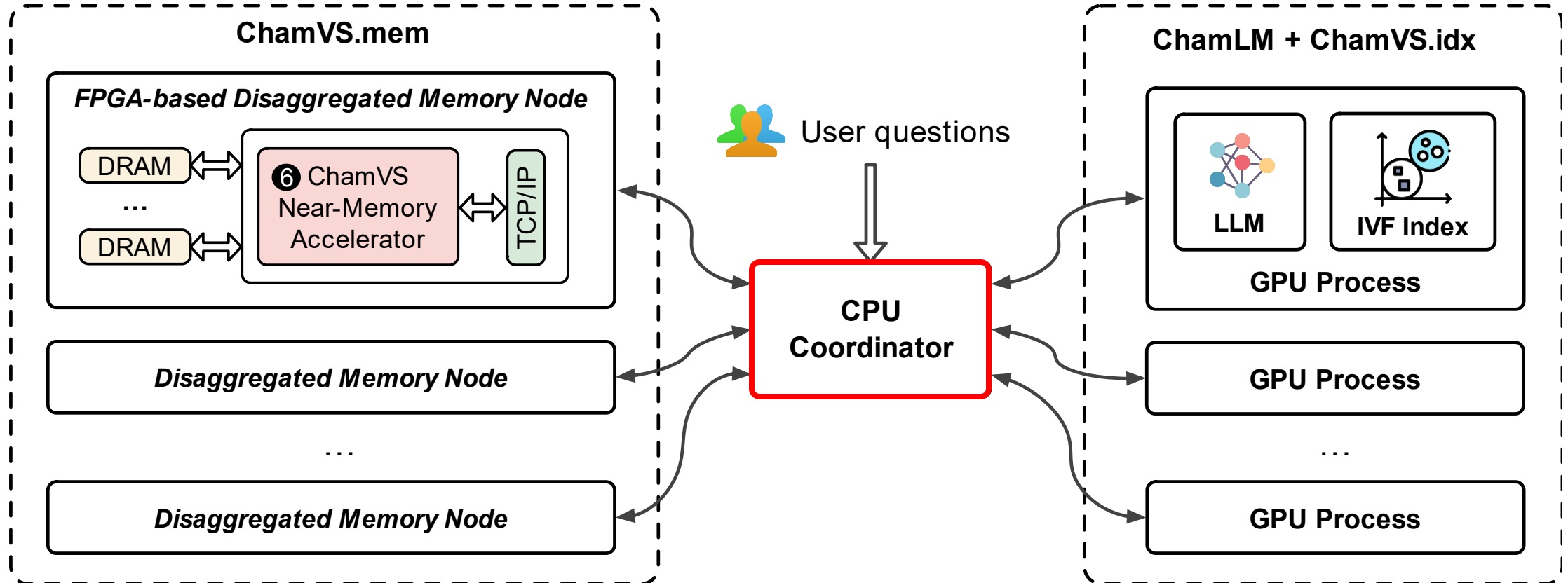
# Chameleon: accelerator heterogeneity + disaggregation



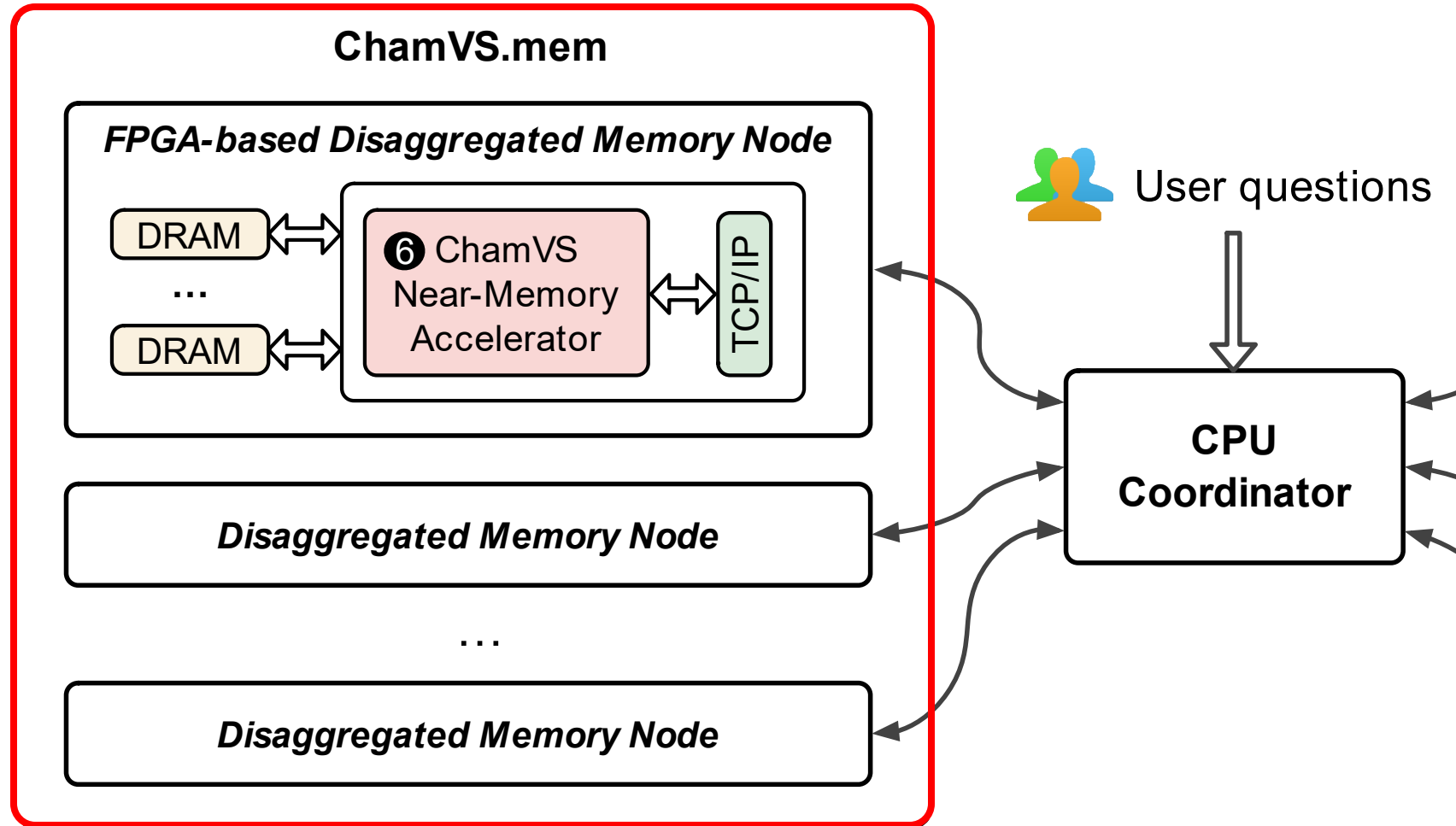
# Chameleon: accelerator heterogeneity + disaggregation



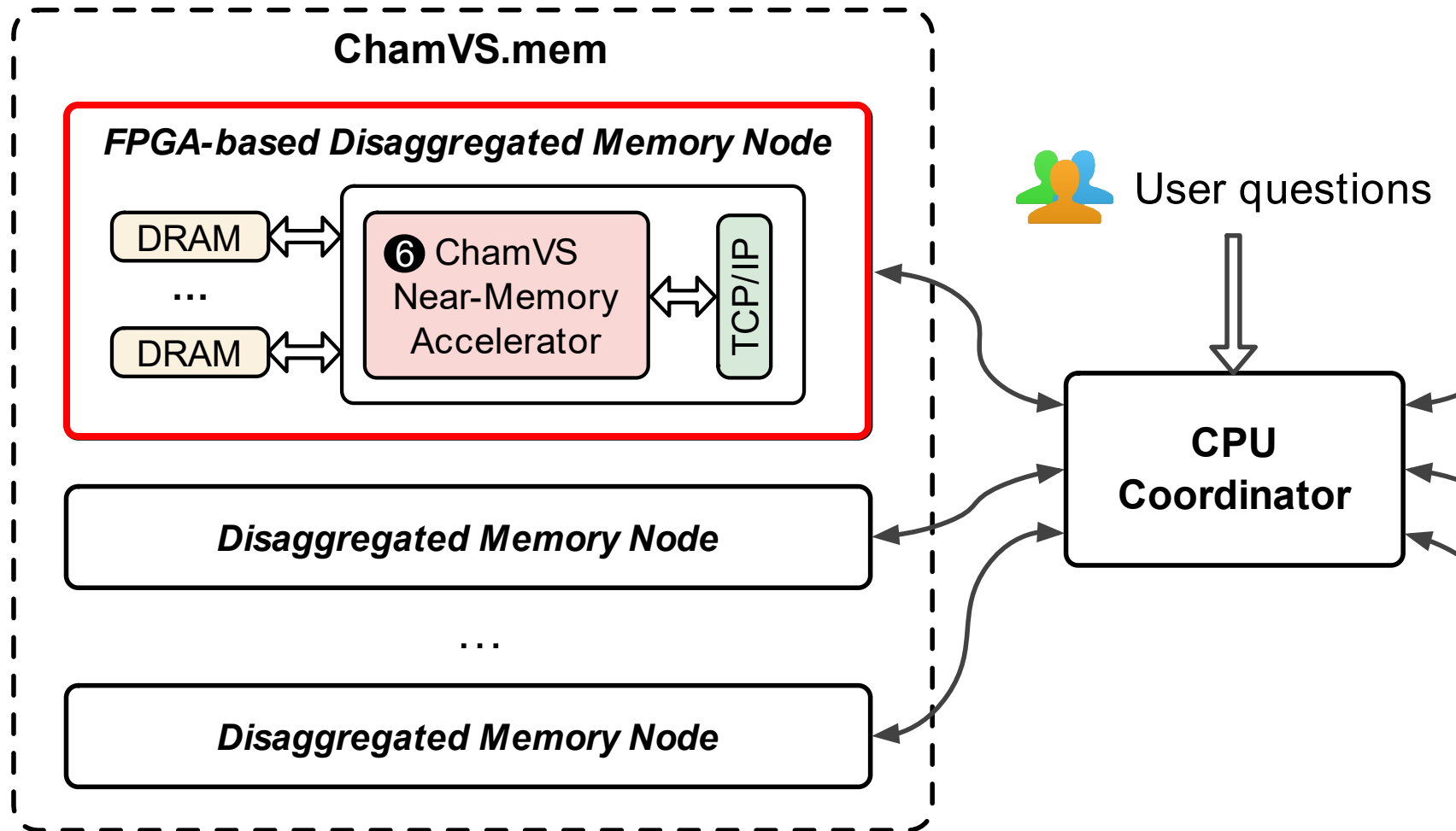
# Chameleon: accelerator heterogeneity + disaggregation



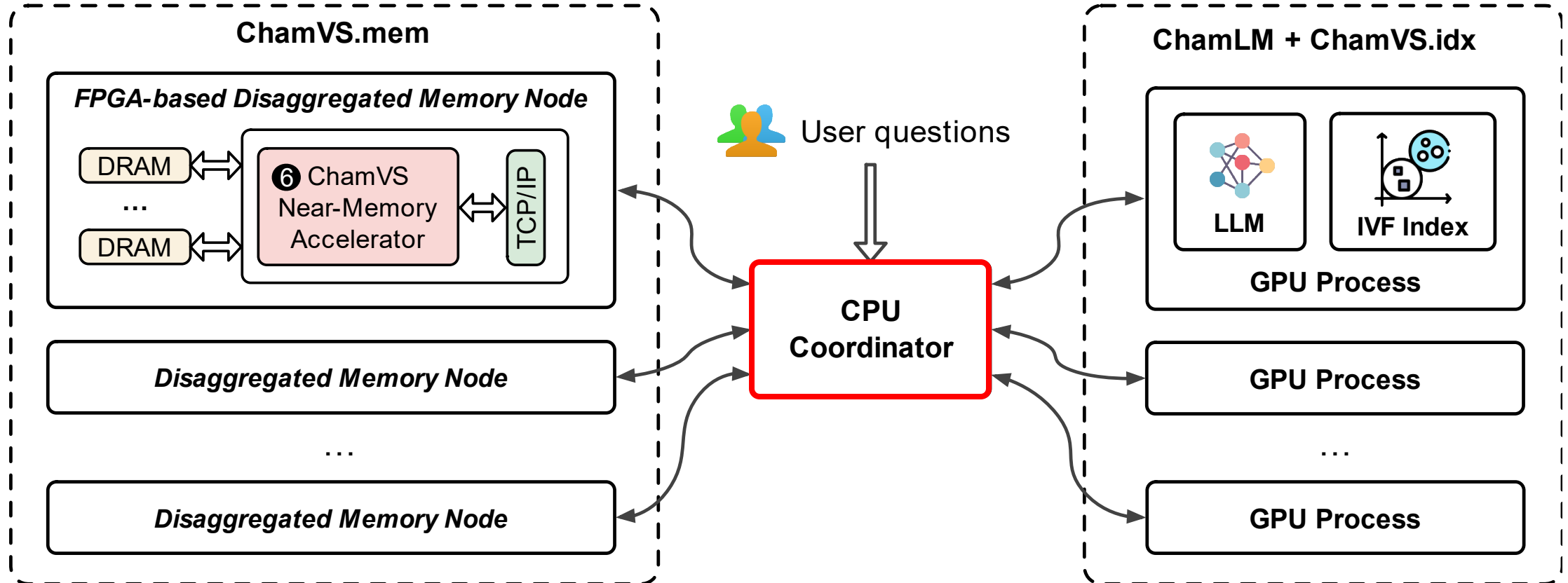
# Chameleon: accelerator heterogeneity + disaggregation



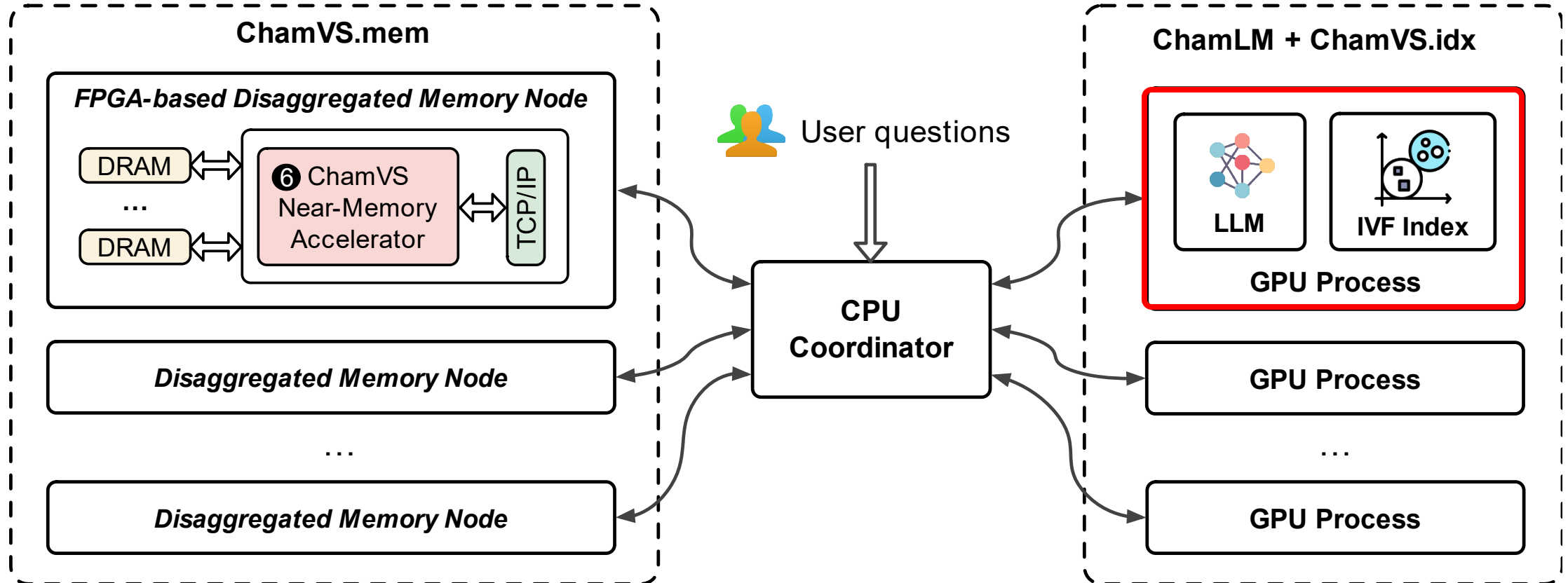
# Chameleon: accelerator heterogeneity + disaggregation



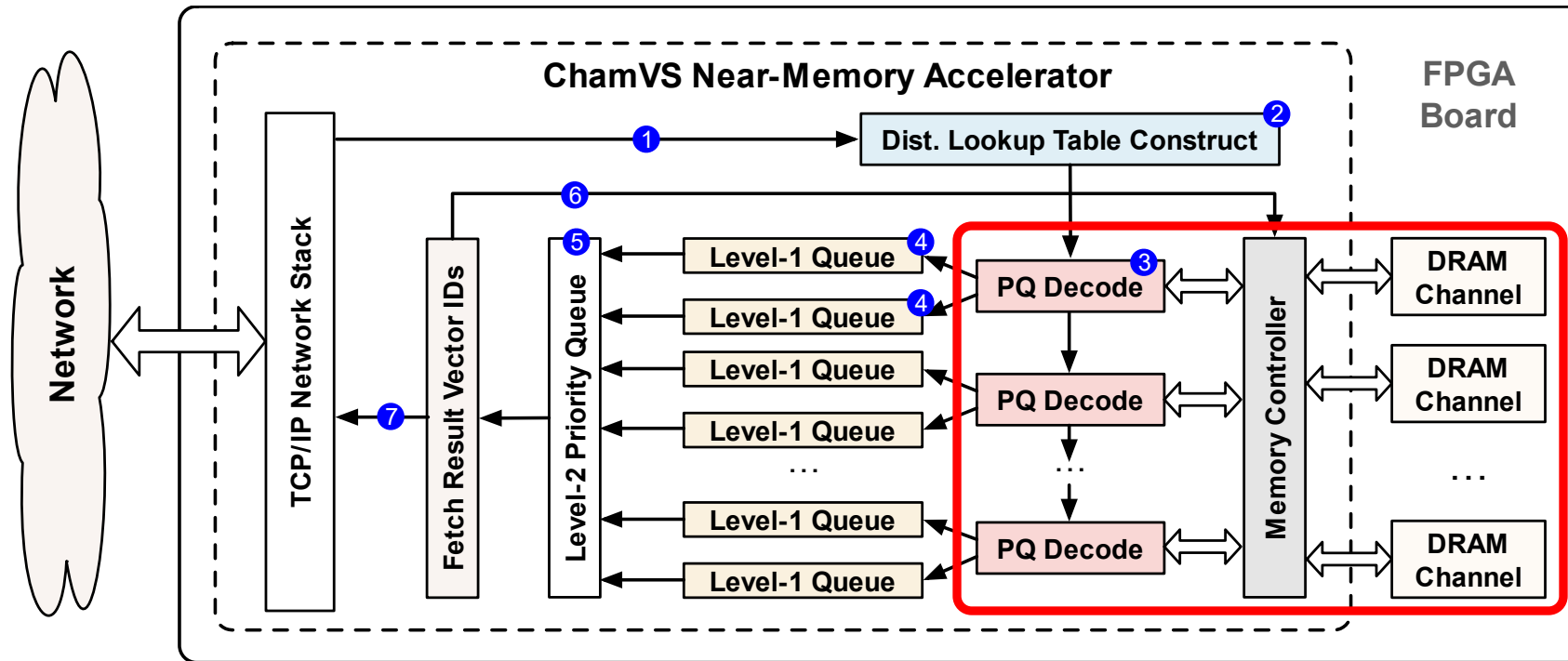
# Chameleon: accelerator heterogeneity + disaggregation



# Chameleon: accelerator heterogeneity + disaggregation



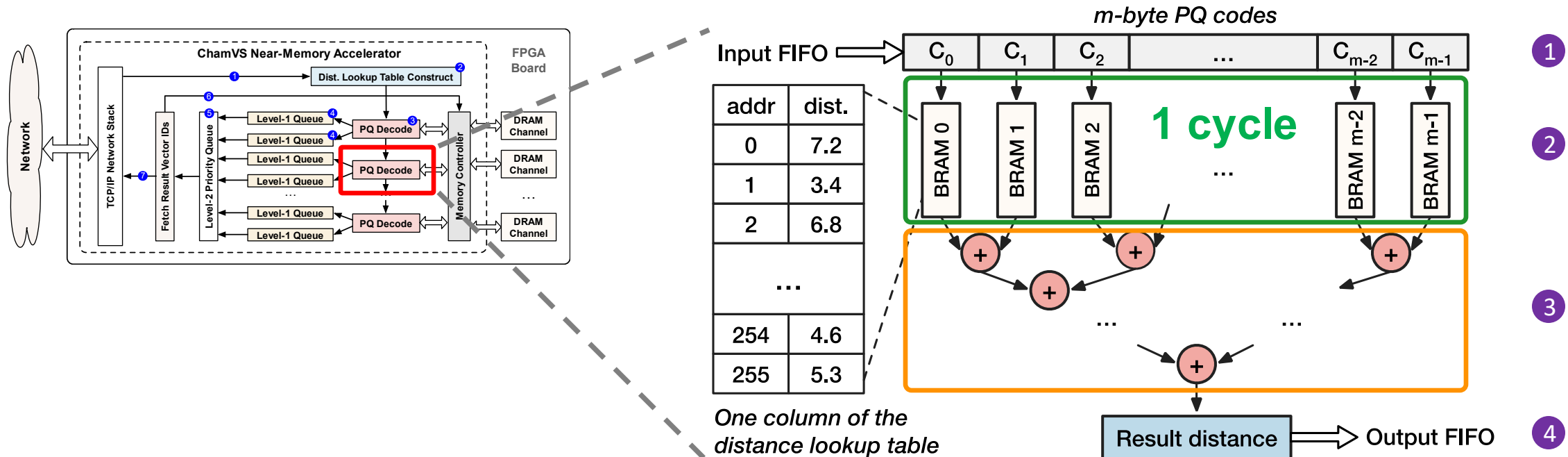
# ChamVS: near-memory retrieval acceleration



Compared to CPUs: **faster PQ decoding**

Compared to GPUs: **abundant capacity; lower latency**

# ChamVS: near-memory retrieval acceleration

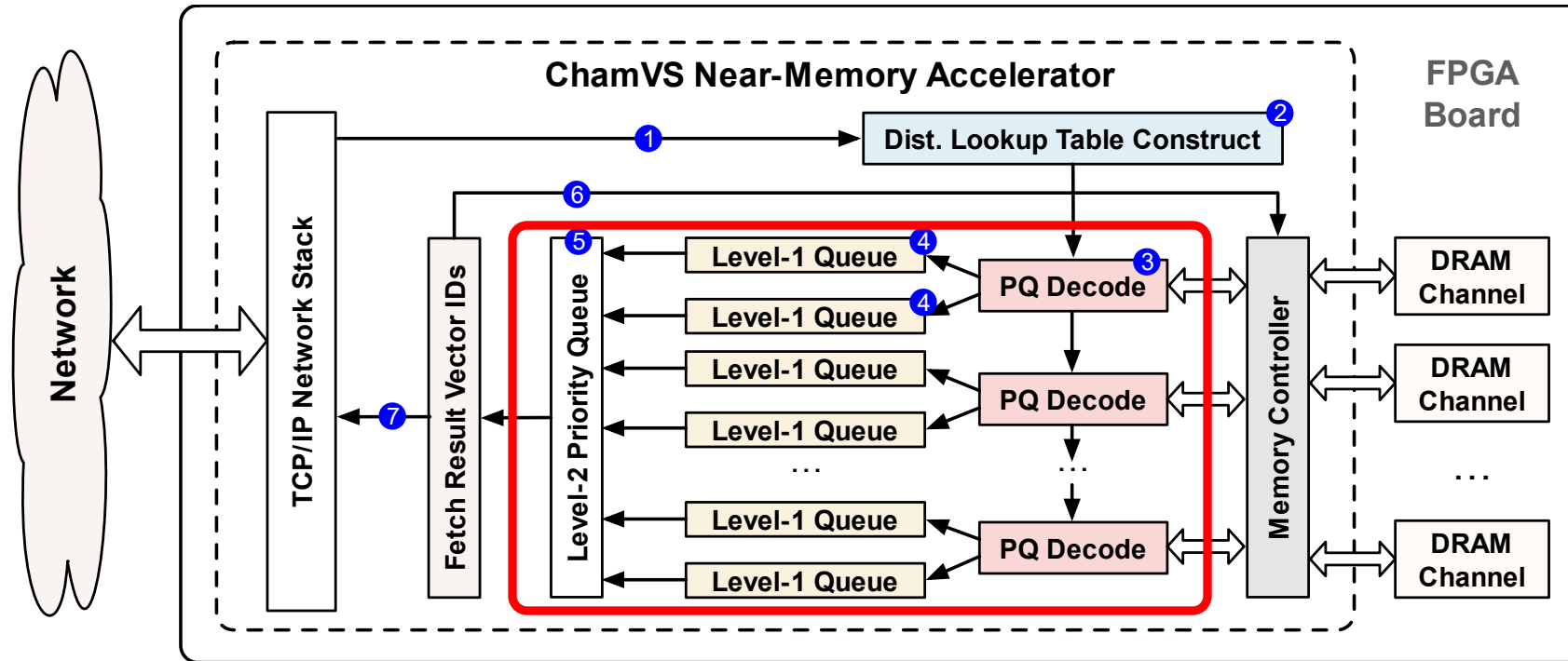


**Parallel lookup** + **Parallel computation** + **Pipeline parallelism**

=

High throughput of **one result distance per clock cycle**

# ChamVS: near-memory retrieval acceleration



Now we have very fast PQ decoding: **dozens of results per cycle**

**Challenge: inserting many distances into top-K queue per cycle**

# ChamVS: near-memory retrieval acceleration

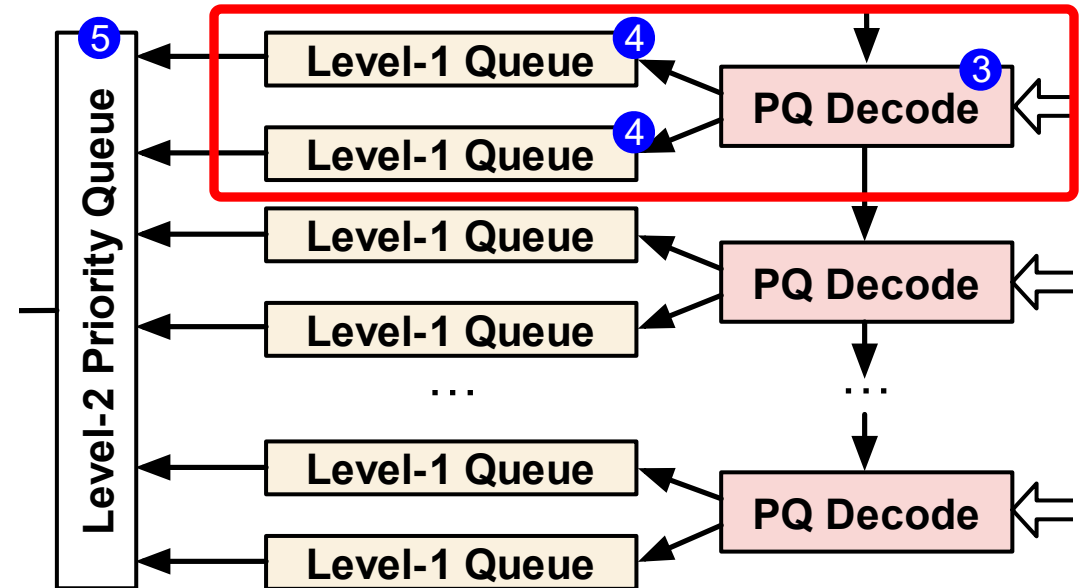
Systolic priority queue:

👍 **High throughput**

one ingestion / two cycles

👎 **High resource consumption**

queue length x queue num

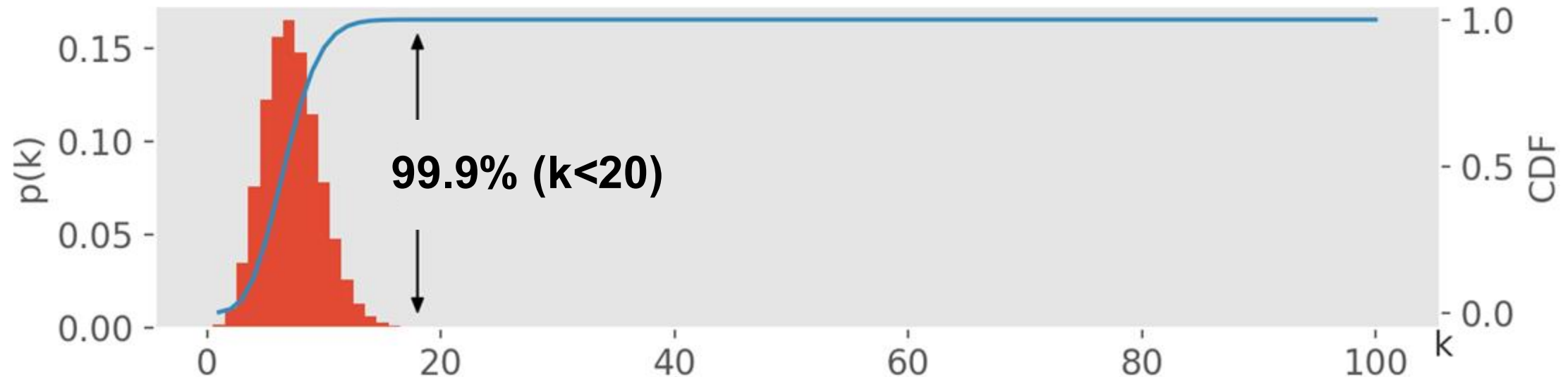


**Question: how to reduce hardware resource consumption?**

# Approximate hierarchical priority queue

Example: 16 queues to collect 100 nearest neighbors

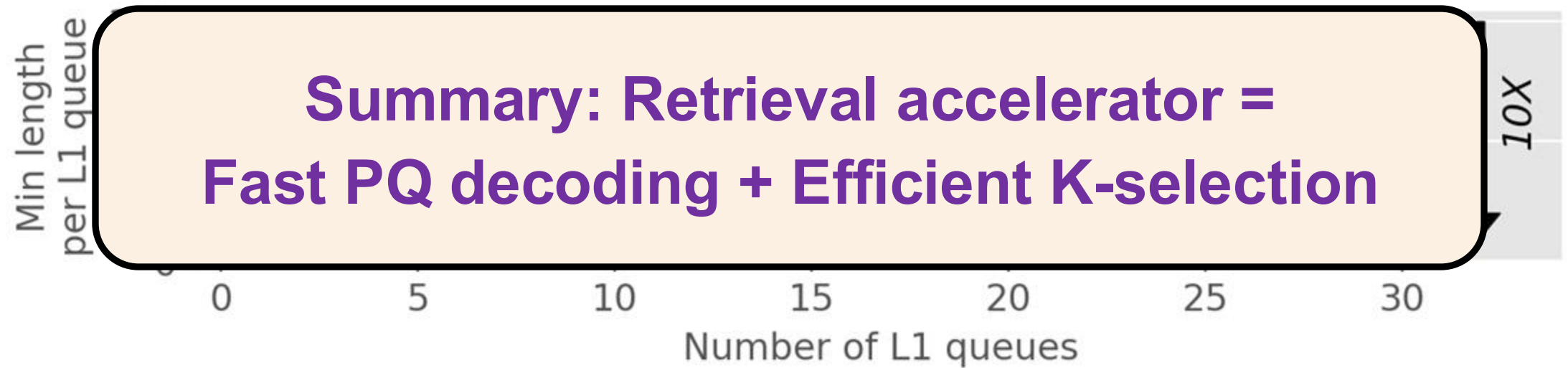
**Is it likely that all 100 results are located in one queue?**



**Finding: Most queues only contain less than 20 results**

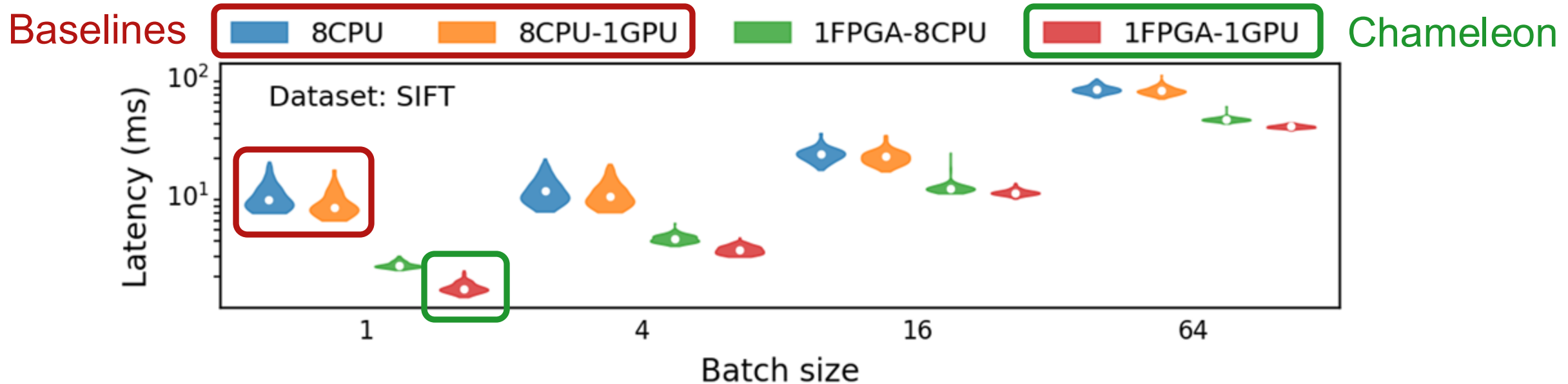
# Approximate hierarchical priority queue

**Idea: Truncate the queues significantly** while achieving similar K-selection quality (e.g., 99% identical results)



**10x resource saving without notable recall degrade**

# Vector search performance and energy efficiency

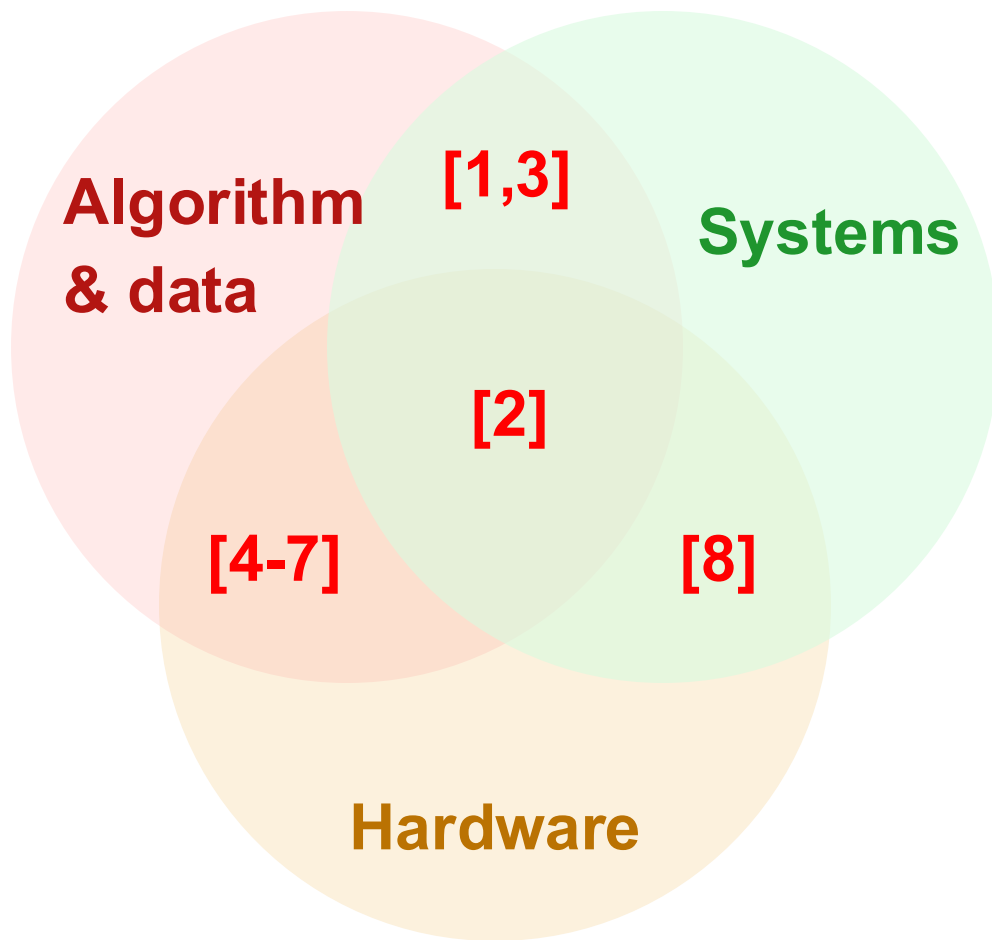


**Chameleon** achieves up to **16.6x speedup over CPU** baseline

**Energy efficiency** (Joule/query) is up to **26.2x** better than CPU

**End-to-end RAG speedup: 2.2x in latency and 3.2x in throughput**

# My research: next-generation ML infrastructure



**Cross-stack design is the future:**  
Strong interplays between algorithm, data, system, hardware, ...

[1] RAGO [ISCA'25]

[2] Chameleon [VLDB'25]

[3] PipeRAG [KDD'25]

[4] FANNS [SC'23]

[5] Falcon [VLDB'25 (revision)]

[6] SwiftSpatial [SIGMOD'25]

[7] MicroRec [MLSys'21]

[8] FleetRec [KDD'21]

Only first-author papers are listed