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

Wenqi Jiang

June 2025

ETH zürich

Systems@ETH zürich

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: 1st systematic performance optimization for RAG

Efficiently serving diverse and evolving RAG algorithms

Chameleon: 1st heterogeneous accelerator system for RAG

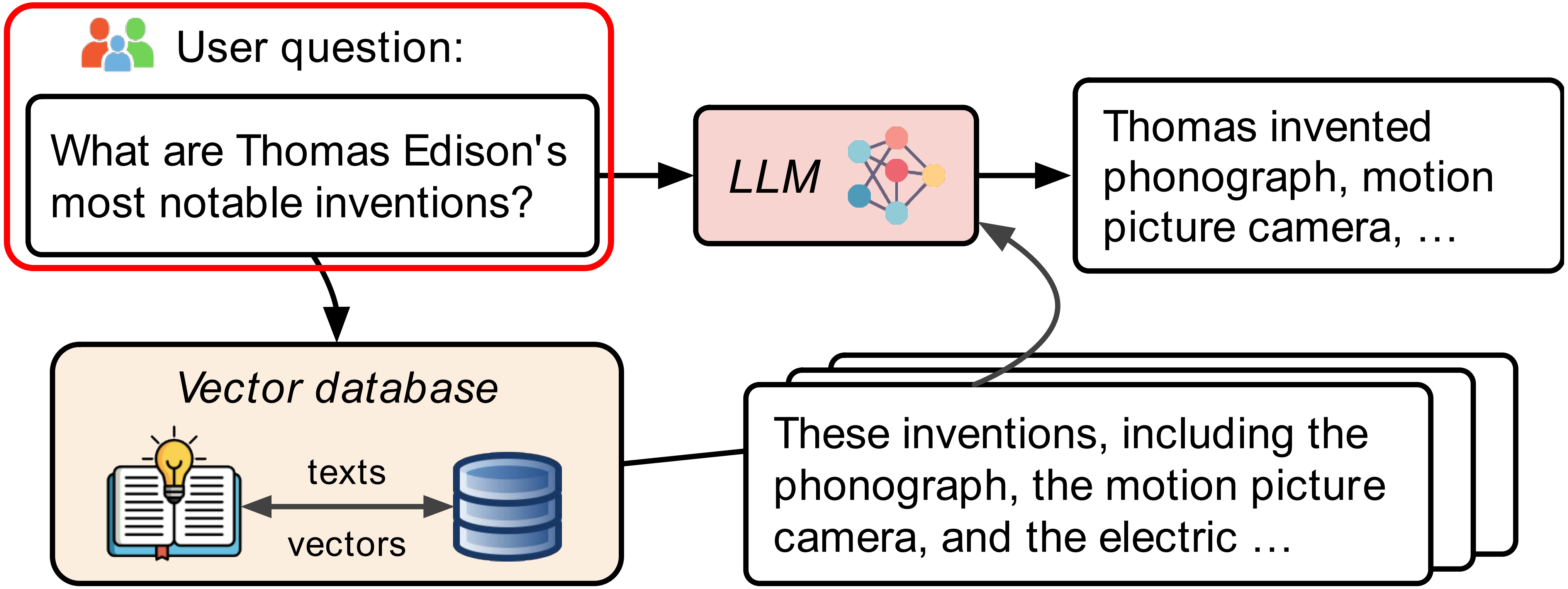
Explore hardware specialization for vector search

Future work: next-generation machine learning systems

Spanning algorithms, databases, systems, and hardware

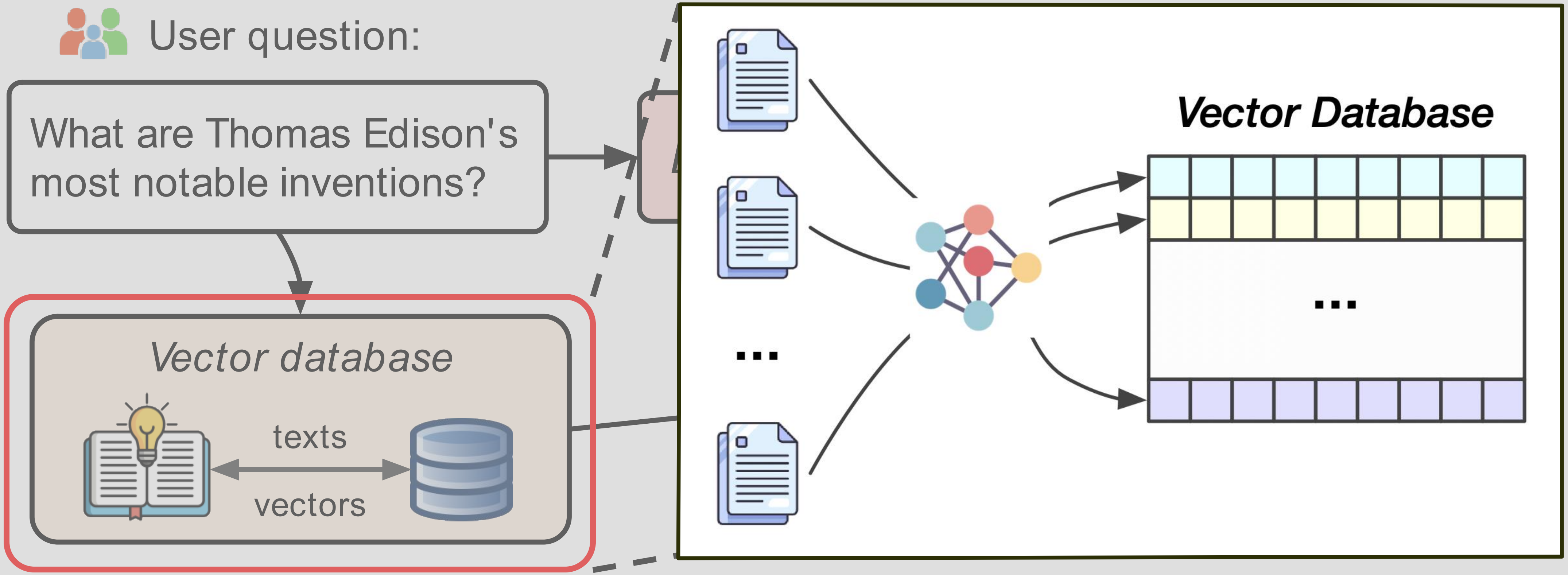
Retrieval-augmented generation (RAG)

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



Retrieval-augmented generation (RAG)

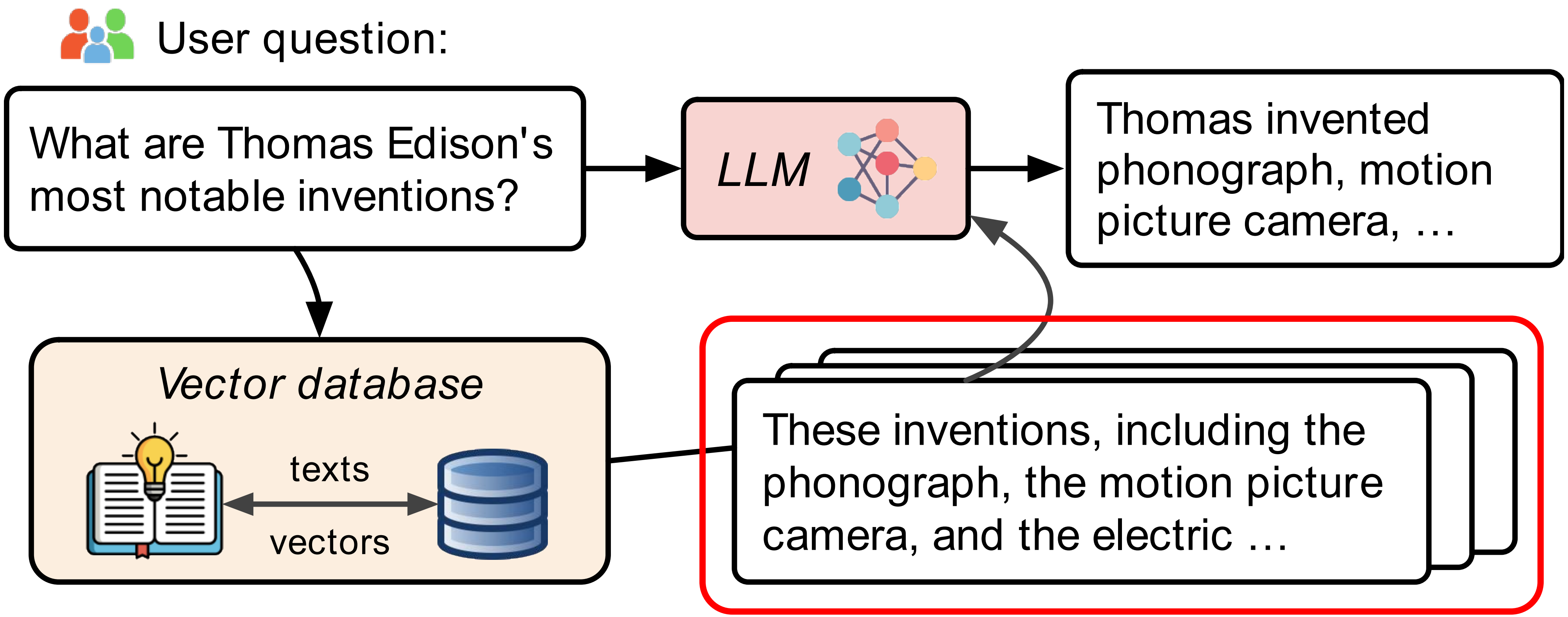
Key idea: pair LLMs with **retrievals from external databases**



Retrieval = Vector Search

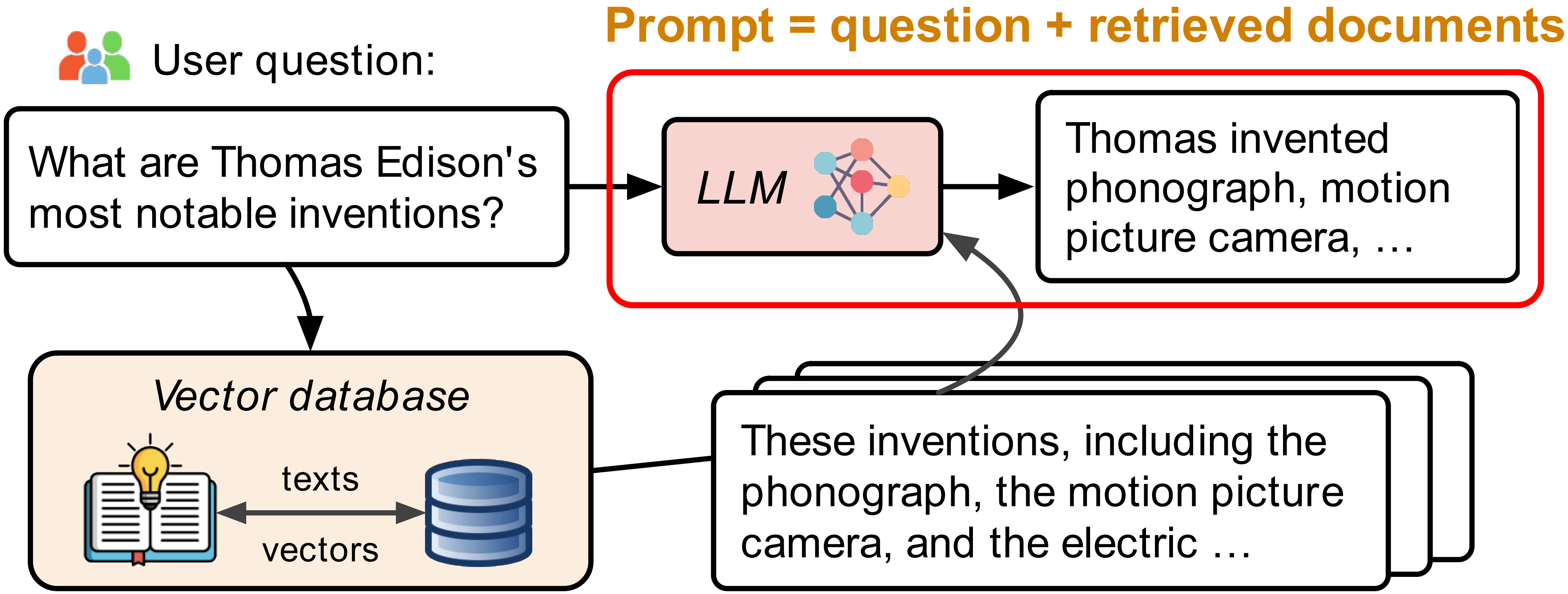
Retrieval-augmented generation (RAG)

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

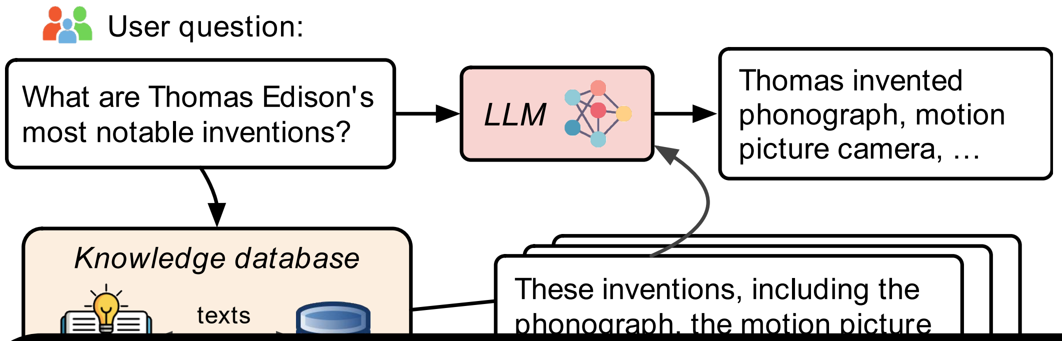


Retrieval-augmented generation (RAG)

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

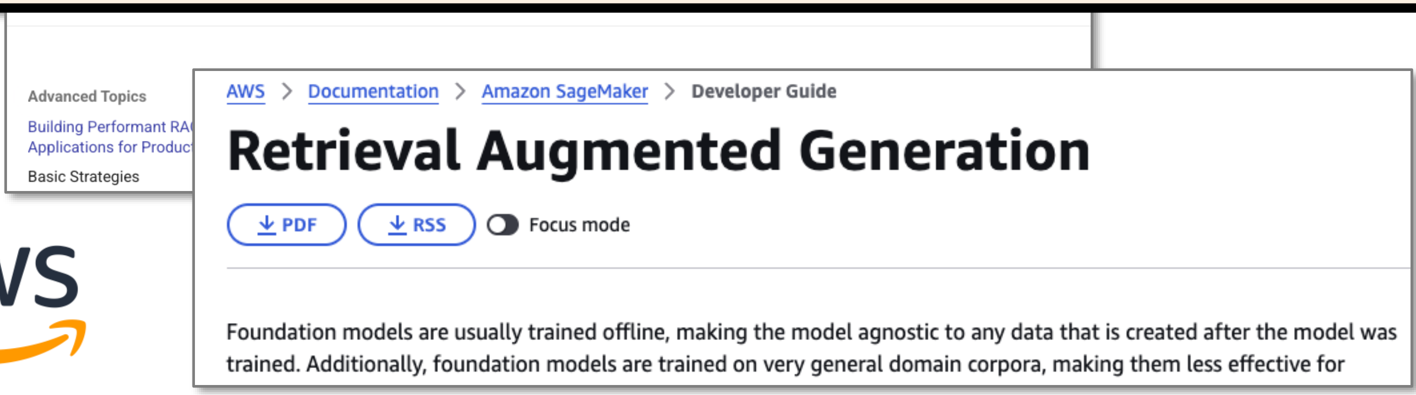


Retrieval-augmented generation (RAG)



- Less hallucination
- Better personalization
- Easier updates

RAG is becoming the industry standard for reliable LLM serving



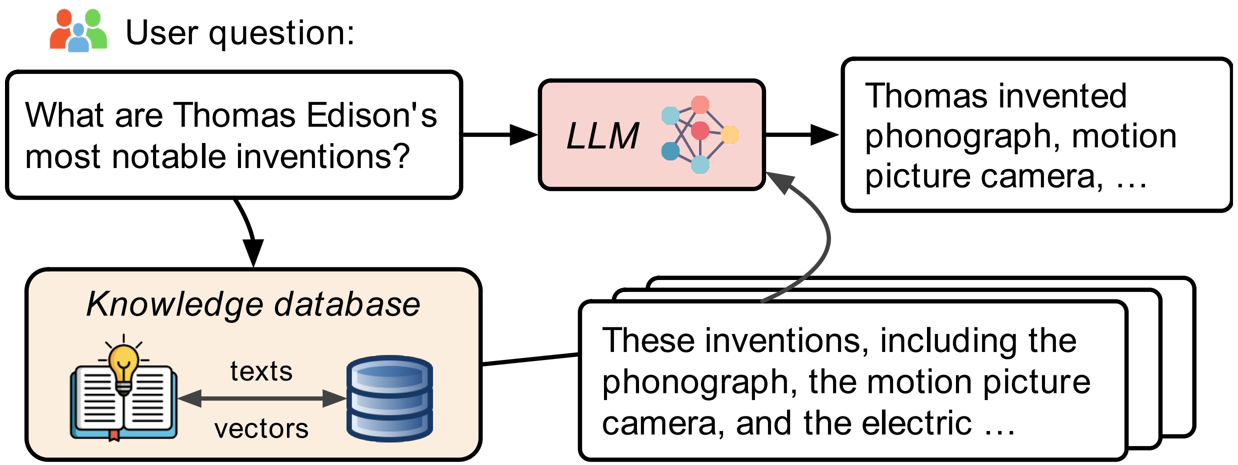
AWS > Documentation > Amazon SageMaker > Developer Guide

Retrieval Augmented Generation

PDF RSS Focus mode

Foundation models are usually trained offline, making the model agnostic to any data that is created after the model was trained. Additionally, foundation models are trained on very general domain corpora, making them less effective for

Retrieval-augmented generation (RAG)

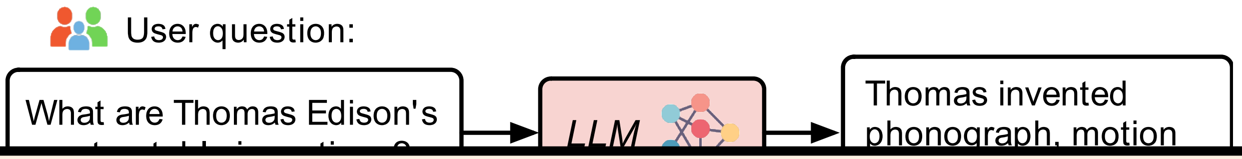


Vector database and retrieval play a key role in the pipeline

Various RAG algorithms of drastically different workload

Multiple system components on **heterogeneous hardware**

Retrieval-augmented generation (RAG)



Cross-stack design is the future of ML systems

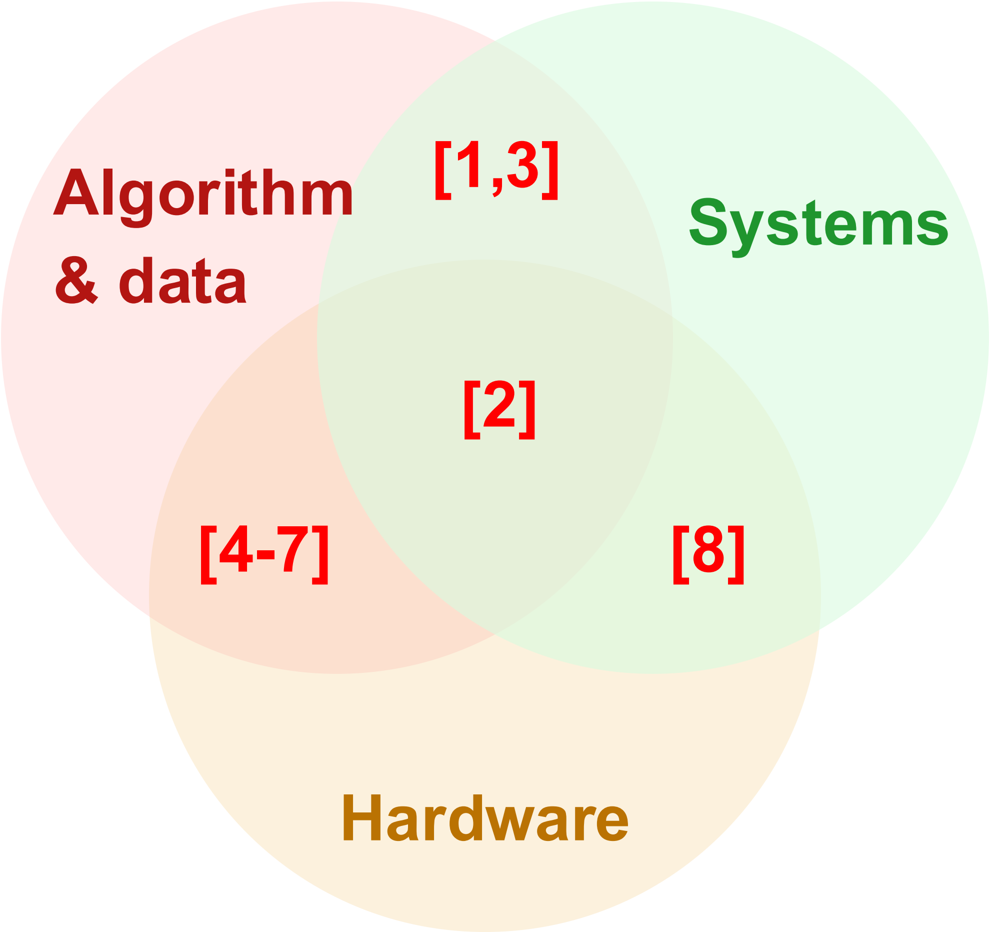
Algorithm, data, system, hardware, …

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

Retrieval-Augmented Generation

↑

Efficient Vector Search

↑

Data Systems on Modern Hardware

RAGO [ISCA'25]

Chameleon [VLDB'25]

PipeRAG [KDD'25]

FANNS [SC'23]

Falcon [VLDB'25 (revision)]

SwiftSpatial [SIGMOD'25]

MicroRec [MLSys'21]

FleetRec [KDD'21]

Only first-author papers are listed

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

System for algorithms

Retrieval-Augmented Generation

Efficient Vector Search

Data Systems on Modern Hardware

RAGO [ISCA'25]

Chameleon [VLDB'25]

PipeRAG [KDD'25]

FANNS [SC'23]

Falcon [VLDB'25 (revision)]

SwiftSpatial [SIGMOD'25]

MicroRec [MLSys'21]

FleetRec [KDD'21]

Only first-author papers are listed

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

Hardware for ML and data systems

Retrieval-Augmented Generation

Efficient Vector Search

Data Systems on Modern Hardware

Chameleon [VLDB'25]

PipeRAG [KDD'25]

FANNS [SC'23]

Falcon [VLDB'25 (revision)]

SwiftSpatial [SIGMOD'25]

MicroRec [MLSys'21]

FleetRec [KDD'21]

Only first-author papers are listed

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

Retrieval-Augmented Ge

Efficient Vector Search

Data Systems on Modern Hardware

RAGO [ISCA'25]

Algorithm for System Efficiency

PipeRAG [KDD'25]

FANNS [SC'23]

Falcon [VLDB'25 (revision)]

SwiftSpatial [SIGMOD'25]

MicroRec [MLSys'21]

FleetRec [KDD'21]

Google

aws

Alibaba

Only first-author papers are listed

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

Retrieval-Augmented Generation

Algorithm-Hardware Co-design

Efficient Vector Search

Data Systems on Modern Hardware

RAGO [ISCA'25]

Chameleon [VLDB'25]

FANNS [SC'23]

Falcon [VLDB'25 (revision)]

SwiftSpatial [SIGMOD'25]

MicroRec [MLSys'21]

FleetRec [KDD'21]

Only first-author papers are listed

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

Retrieval-Augmented Generation

Efficient Vector Search

Data Systems on Modern Hardware

RAGO [ISCA'25]

Chameleon [VLDB'25]

PipeRAG [KDD'25]

FANNS [SC'23]

n)]

Hardware for ML and data systems

SwiftSpatial [SIGMOD'25]

MicroRec [MLSys'21]

FleetRec [KDD'21]

Only first-author papers are listed

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

System-level

Retrieval-Augmented Generation

RAGO [ISCA’25]

Chameleon [VLDB’25]

[KDD’25]

Hardware-level

Efficient Vector Search

FANNS [SC’23]

Falcon [VLDB’25 (revision)]

Data Systems on Modern Hardware

SwiftSpatial [SIGMOD’25]

MicroRec [MLSys’21]

FleetRec [KDD’21]

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: 1st systematic performance optimization for RAG

Efficiently serving diverse and evolving RAG algorithms

Chameleon: 1st heterogeneous accelerator system for RAG

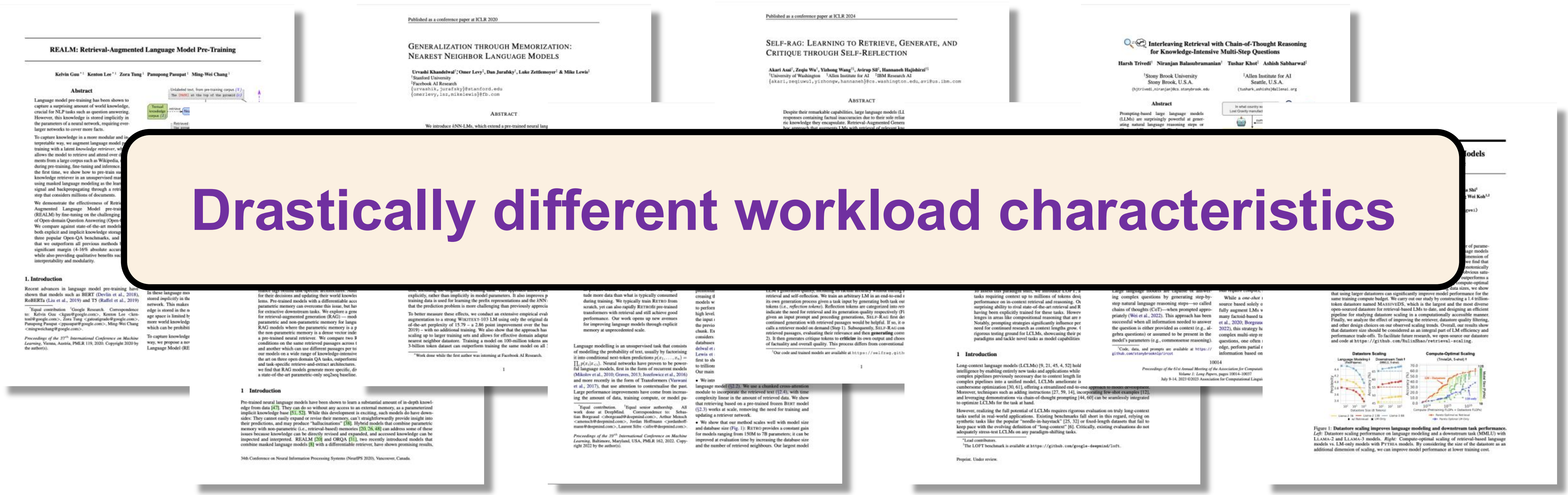
Explore hardware specialization for vector search

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

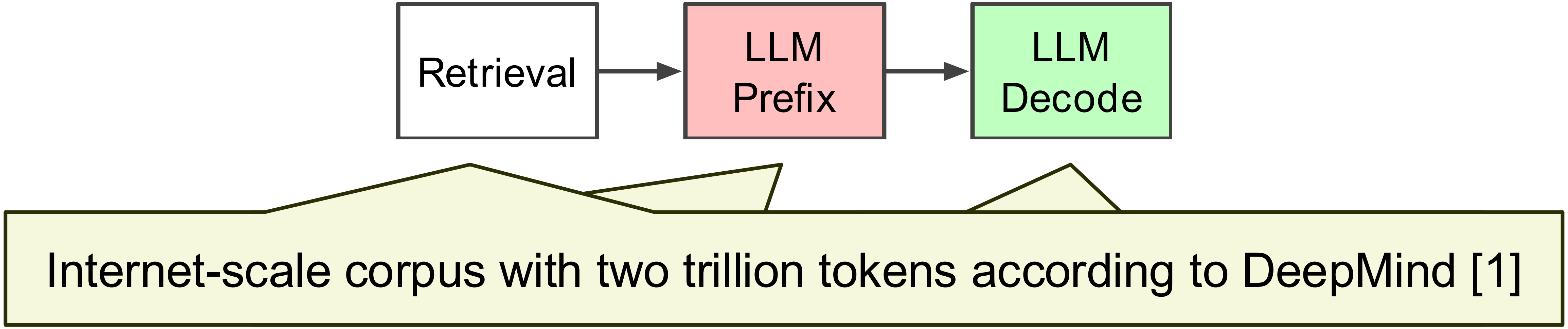


Drastically different workload characteristics

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]

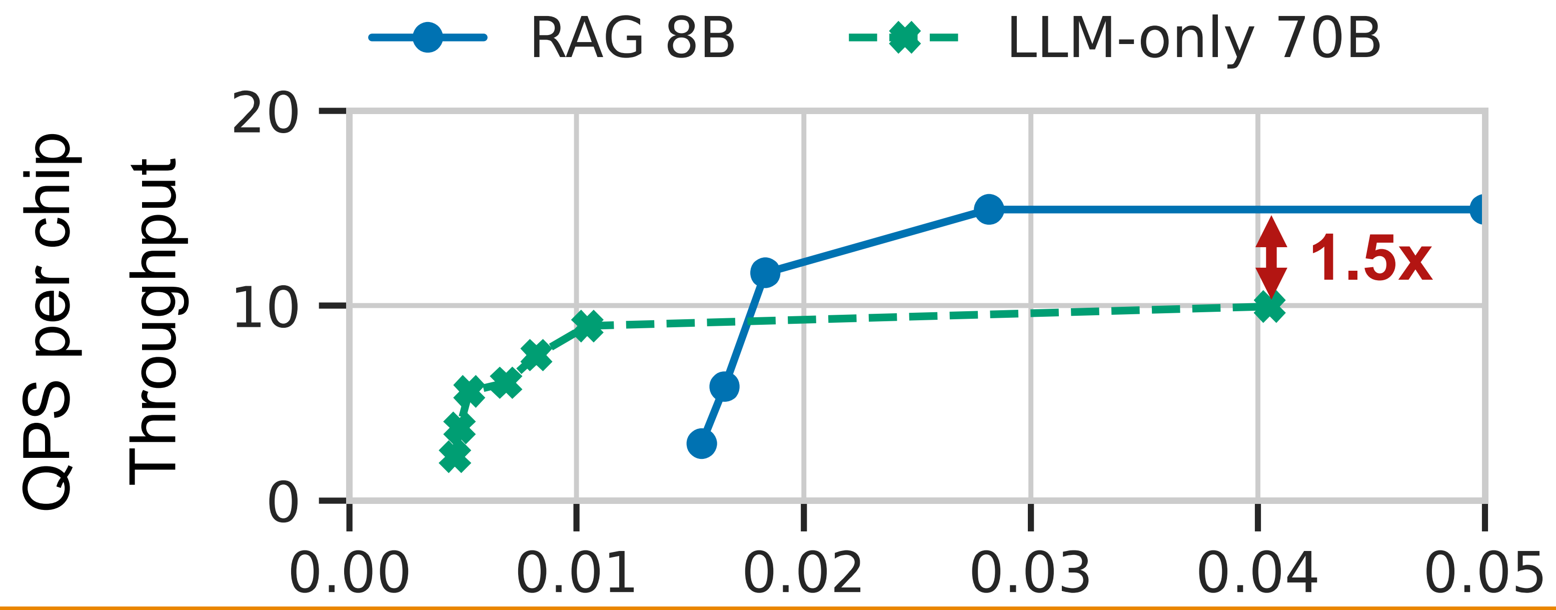


[1] Borgeaud et al. "Improving Language Models by Retrieving from Trillions of Tokens", 2022
[2] Wang et al. "InstructRetro: Instruction Tunning Post Retrieval-Augmented Retraining", 2023

Case study 1: RAG with hyper-scale retrieval

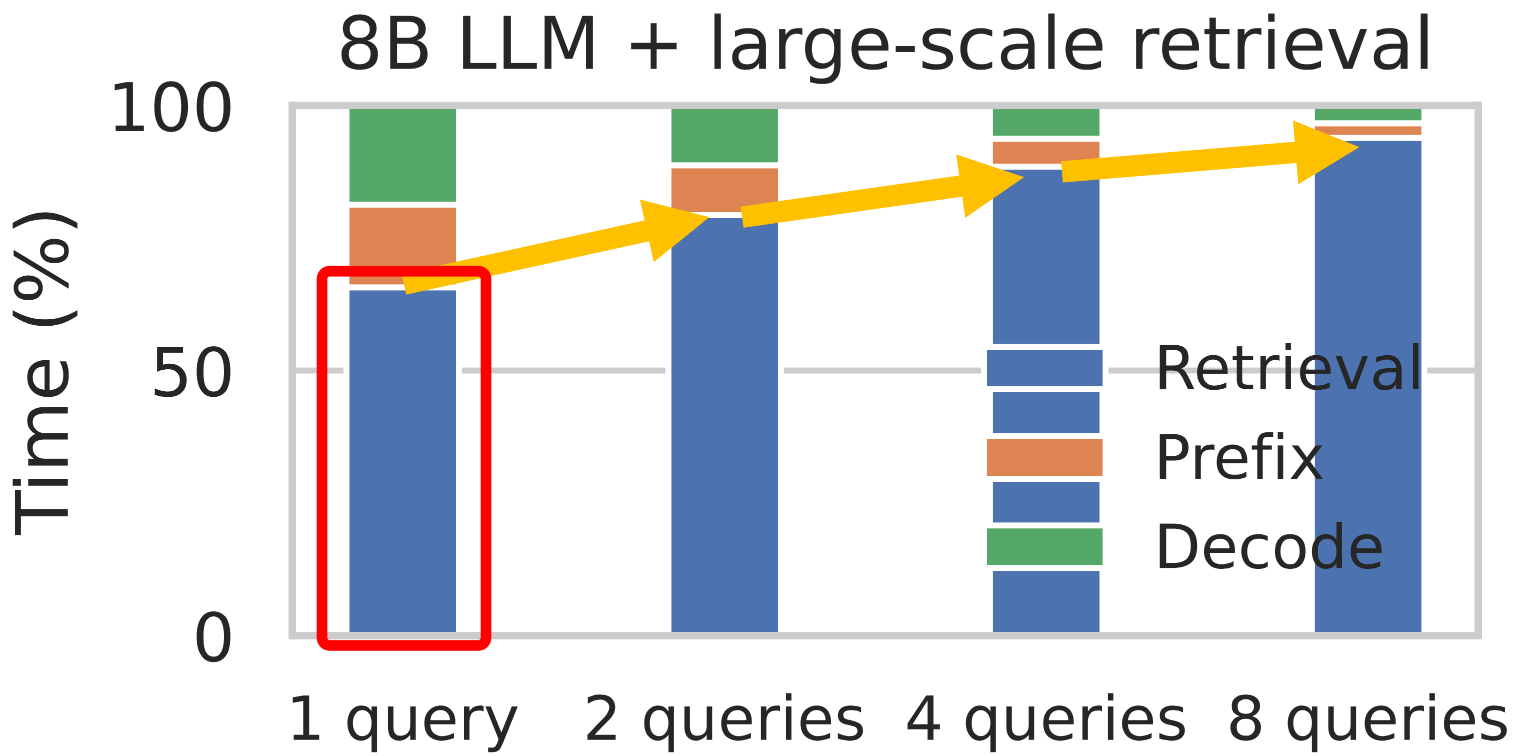
RAG with smaller models achieve **better QPS/chip** than larger LLMs

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

(2nd 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]

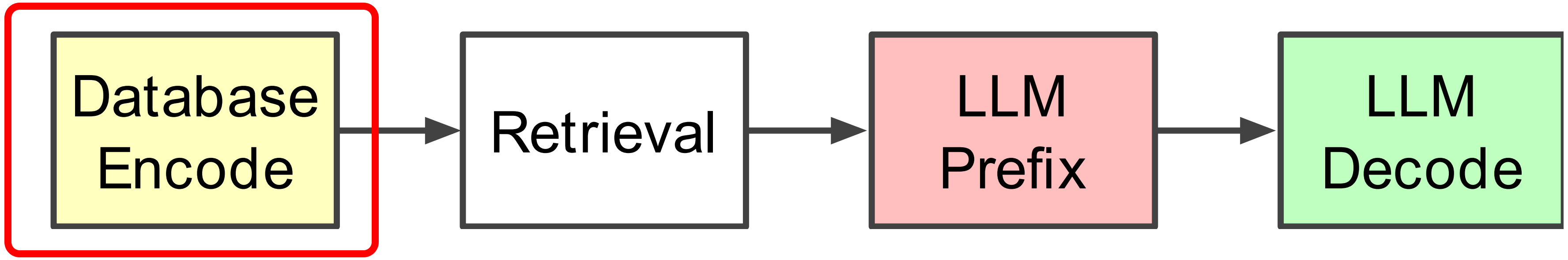
[1] Lee et al. "Can Long-Context Language Models Subsume Retrieval, RAG, SQL, and More?", 2024

[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

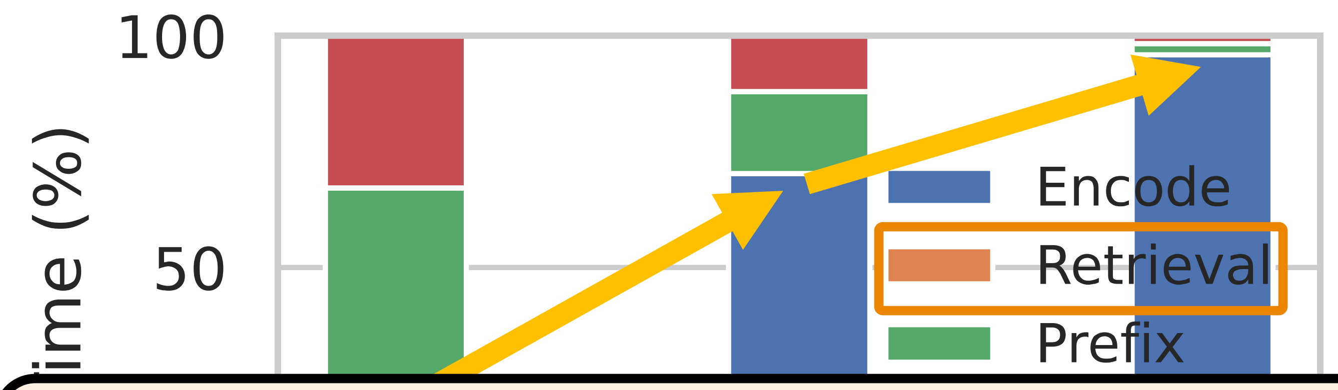
Encode each passage into a vector using a BERT-style model



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

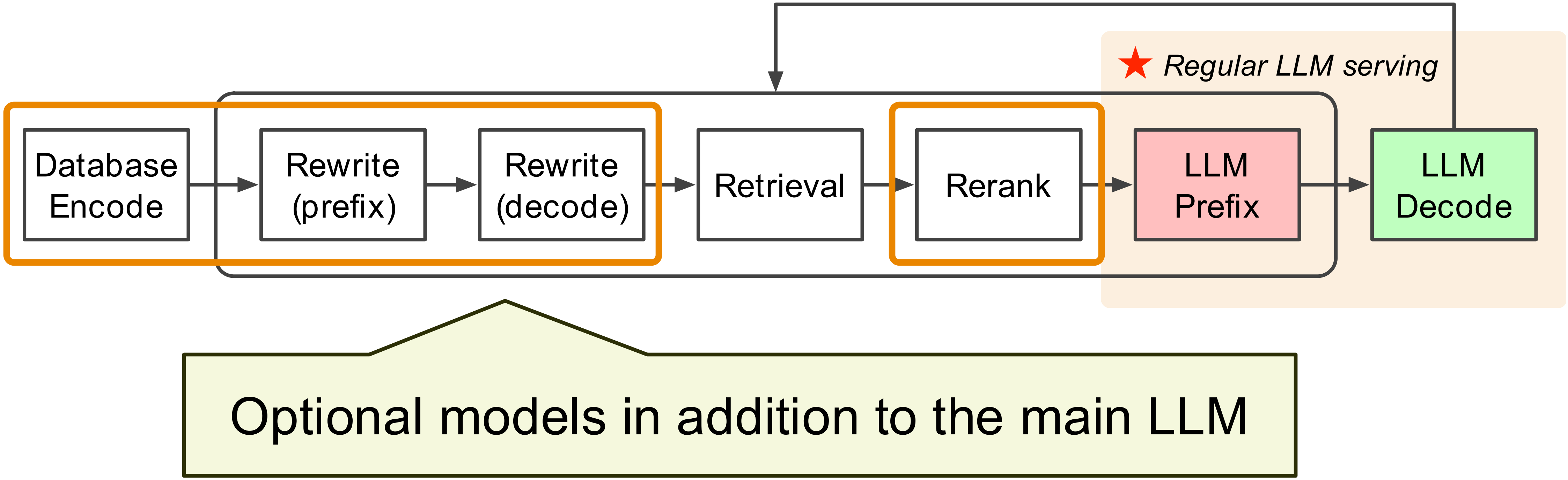
Document lengths (tokens)

1. Even a small encoder model can become the bottleneck

2. Retrieval performance does not matter even with brute-force scan

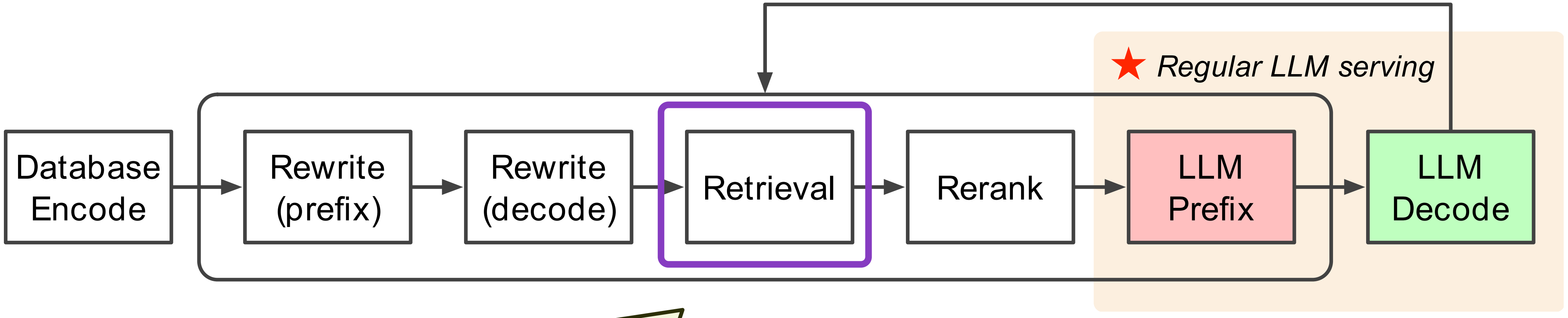
RAGSchema: workload abstraction for RAG algorithms

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



RAGSchema: workload abstraction for RAG algorithms

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

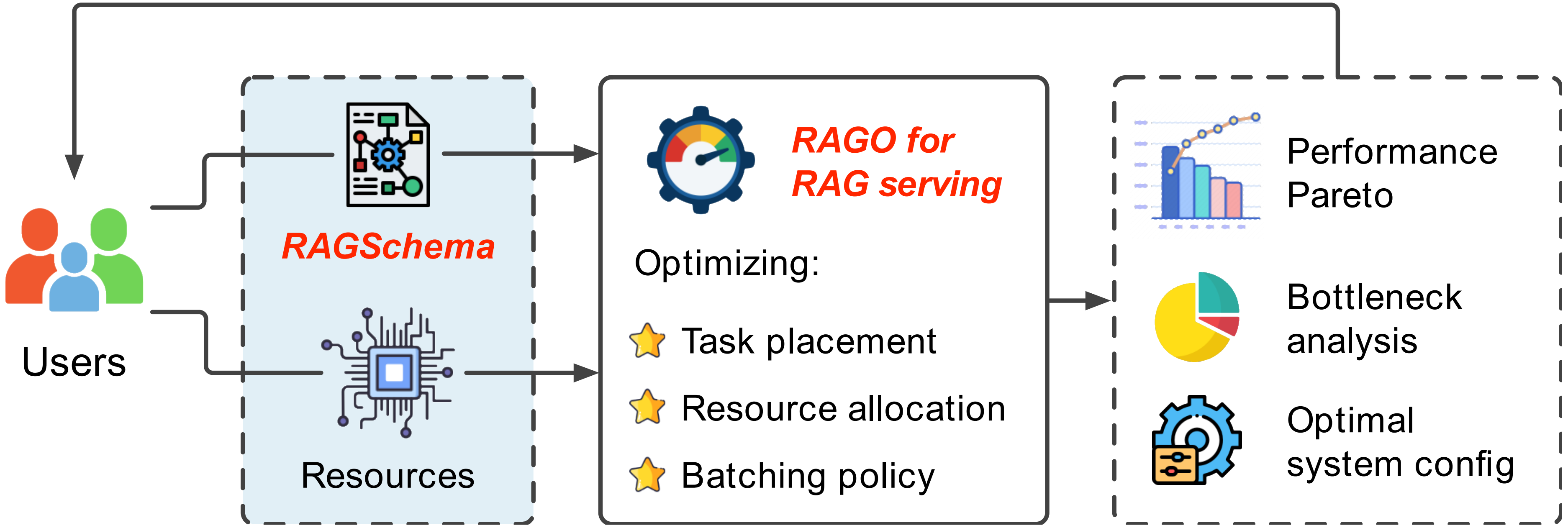


Database sizes; multi-query retrievals; iterative retrievals

RAGO: Retrieval-Augmented Generation Optimizer

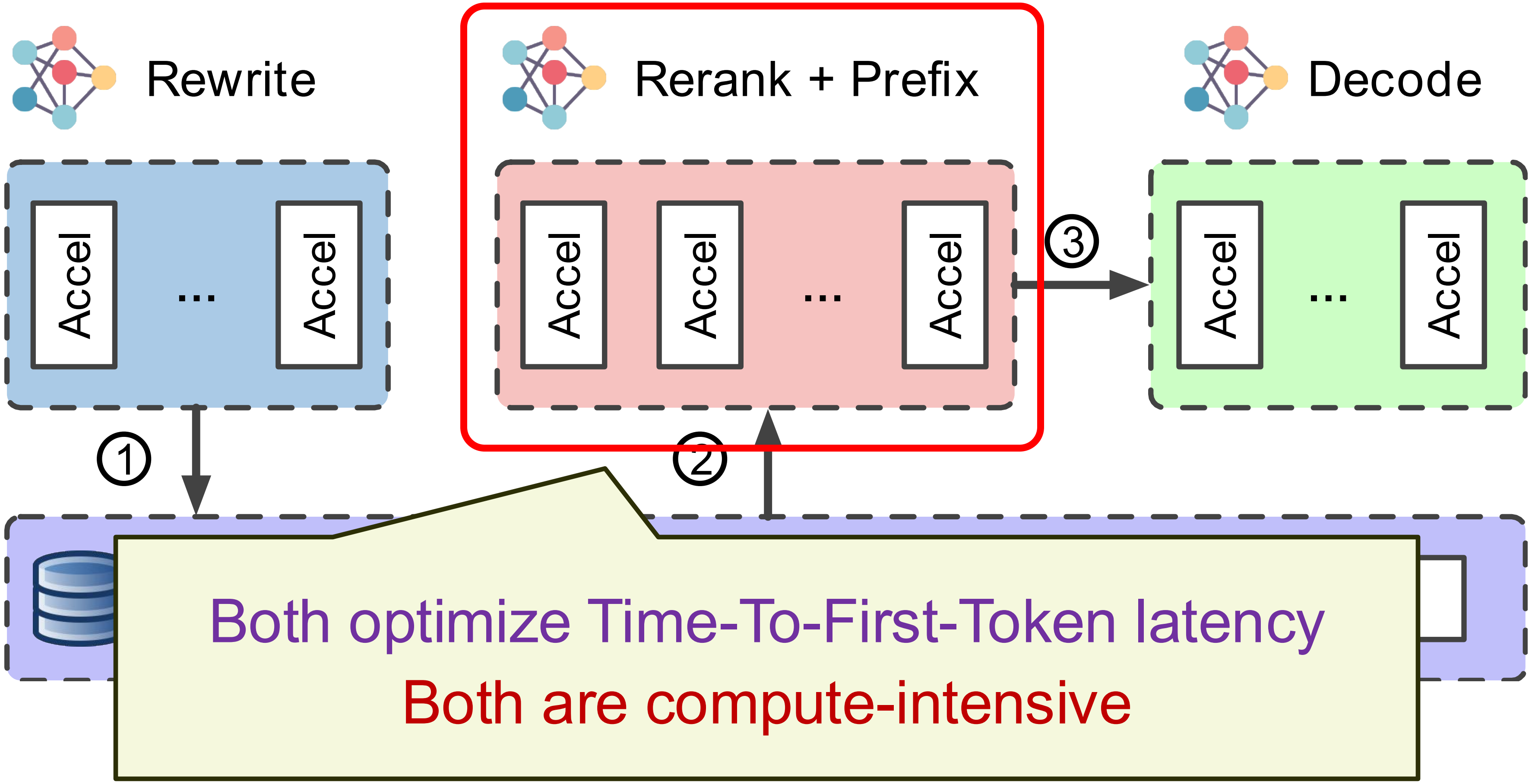
Inputs: **RAGSchema + Hardware resources**

Outputs: **Optimal performance + System configurations**



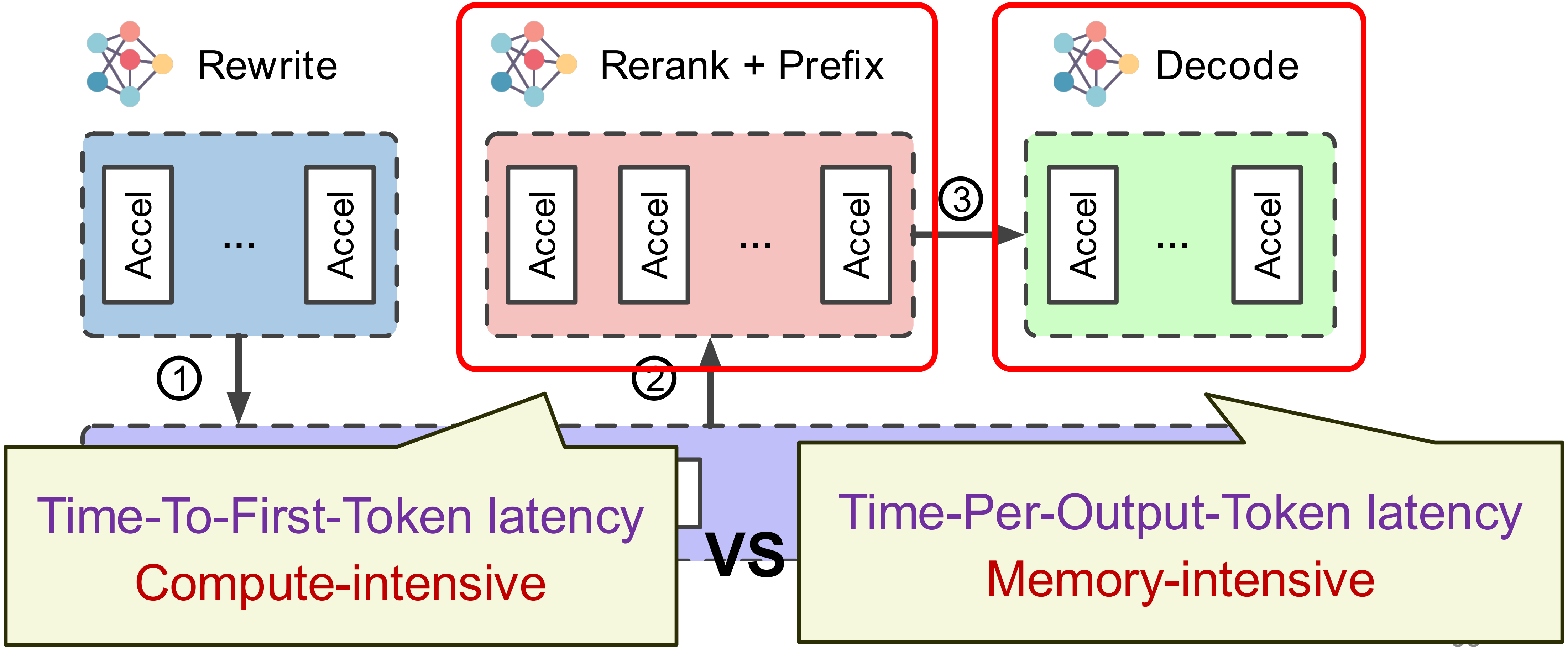
RAGO system design space

Task placement + Resource allocation + Batching



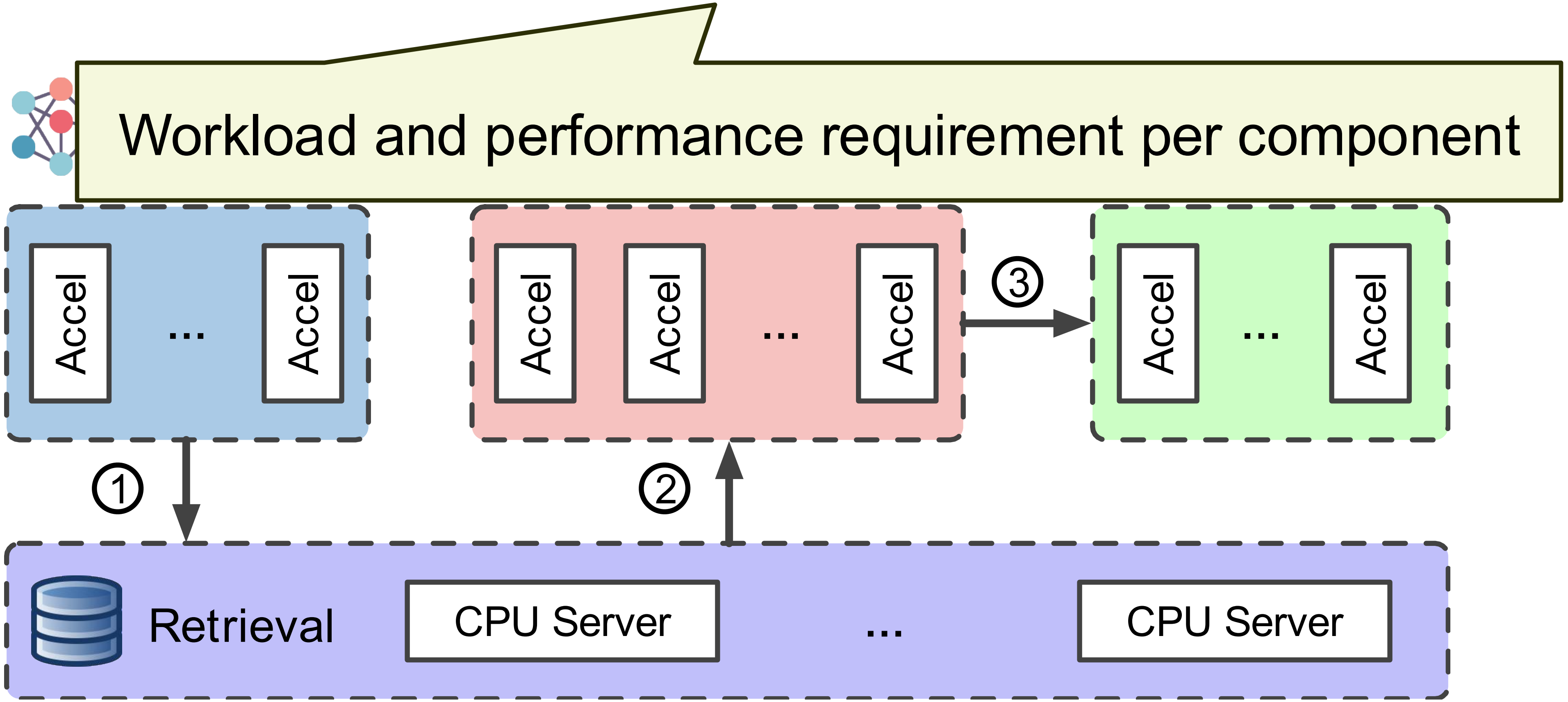
RAGO system design space

Task placement + Resource allocation + Batching



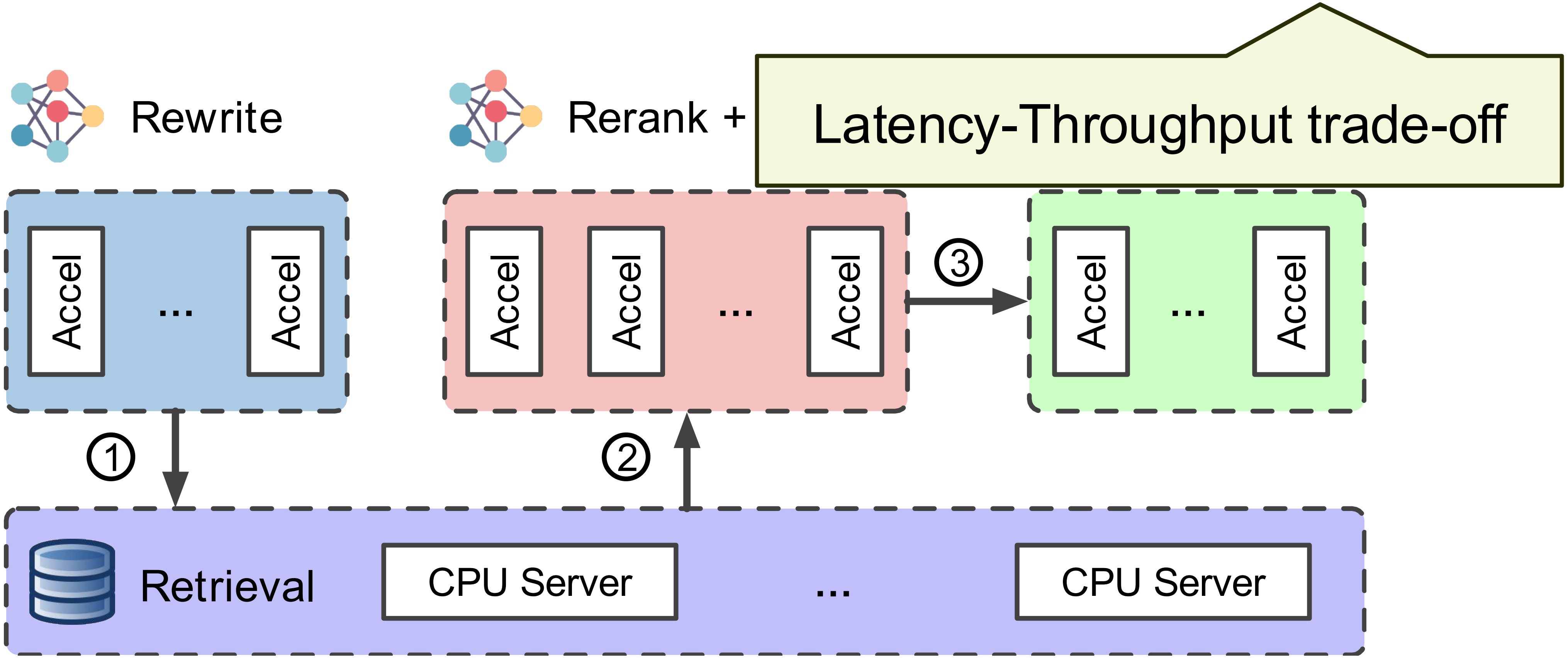
RAGO system design space

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

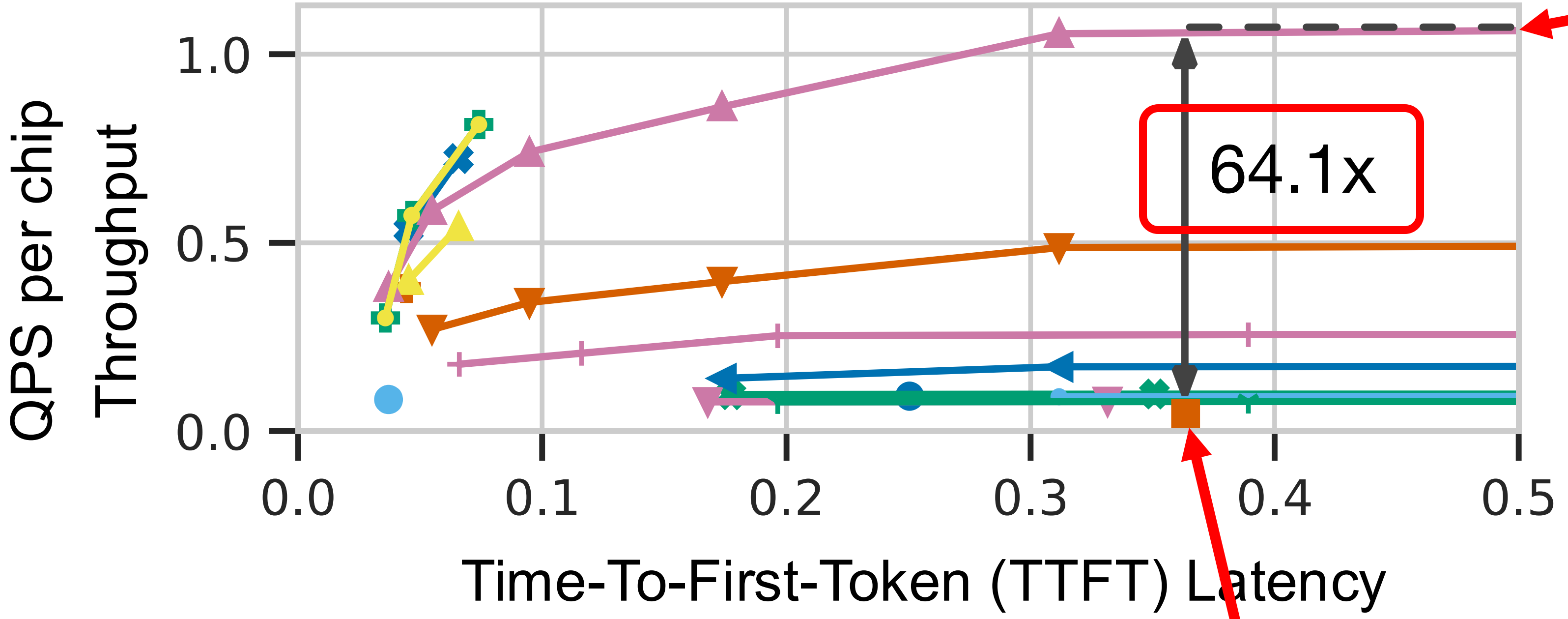
1. **Inference cost model**
2. **Retrieval cost model**

Well-tuned roofline models

3. **RAG cost assembler** to evaluate end-to-end performance
   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: proper resource allocation

Long-context processing:

database encoder << LLM

Naive plan: little resources for the small encoder

RAGO: 1st 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: 1st systematic performance optimization for RAG

Efficiently serving diverse and evolving RAG algorithms

Chameleon: 1st heterogeneous accelerator system for RAG

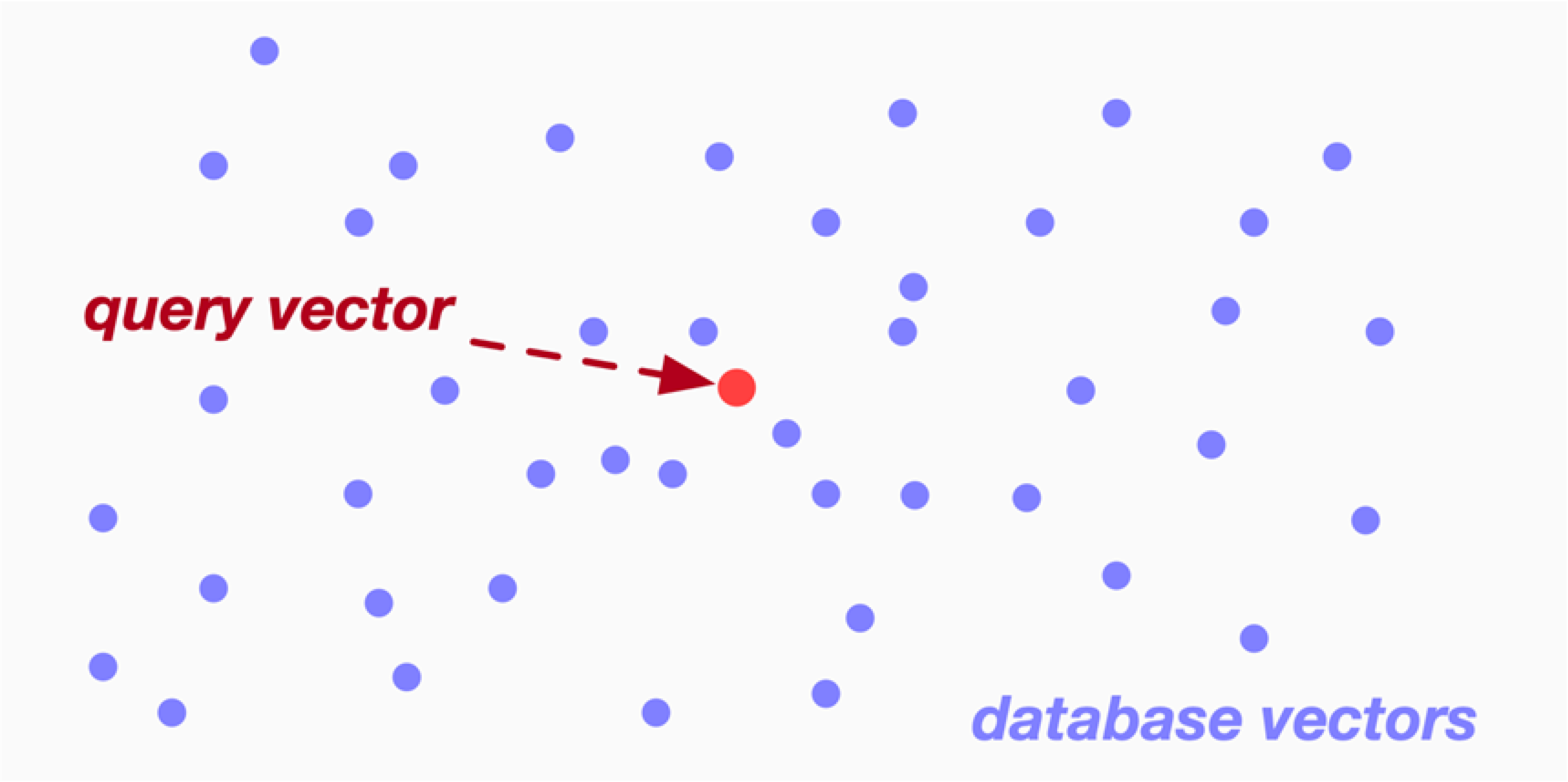
Explore hardware specialization for vector search

Future work: next-generation machine learning systems

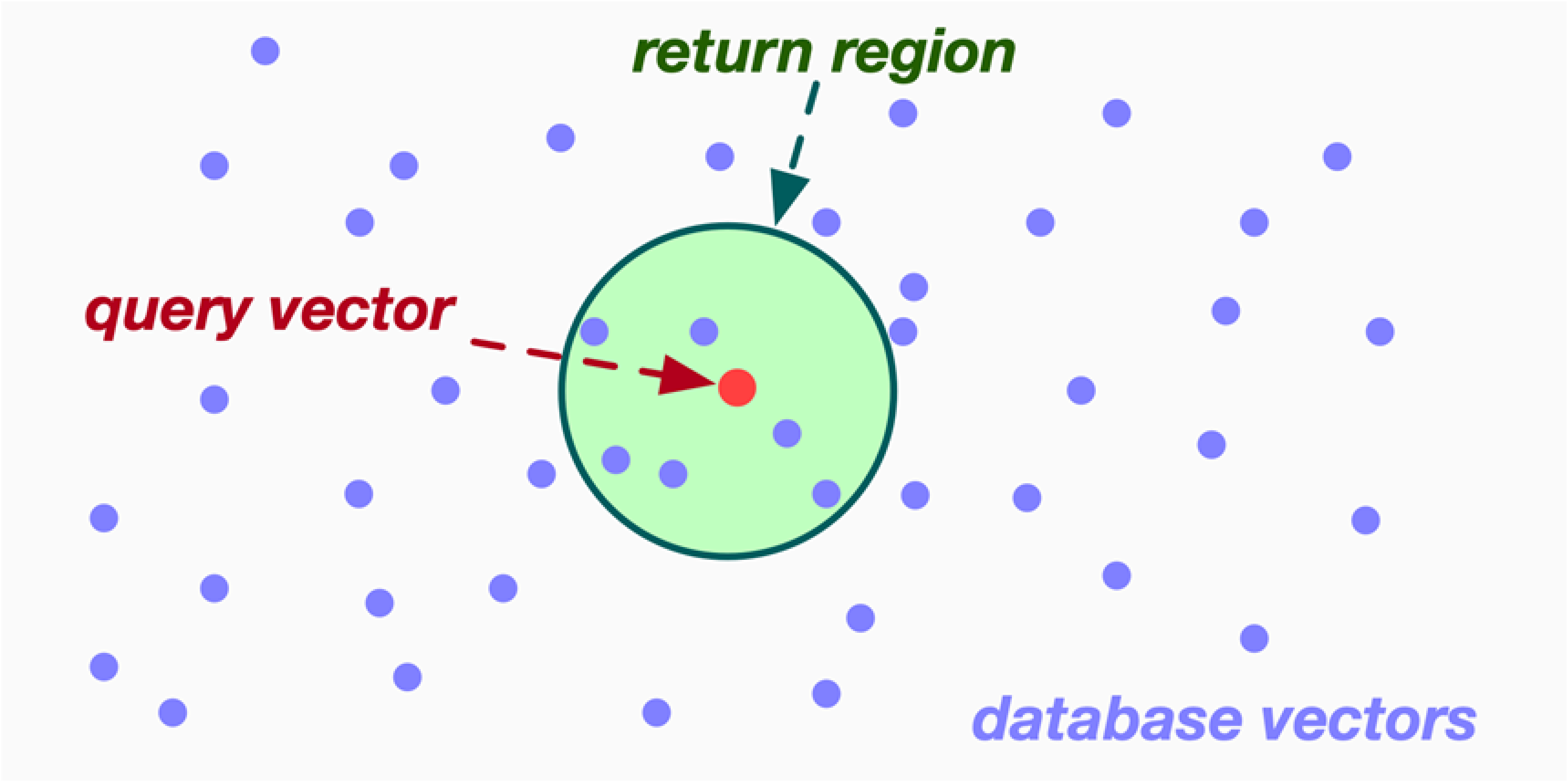
Spanning algorithms, databases, systems, and hardware

Vector search: problem definition

Vector search: problem definition



Vector search: problem definition

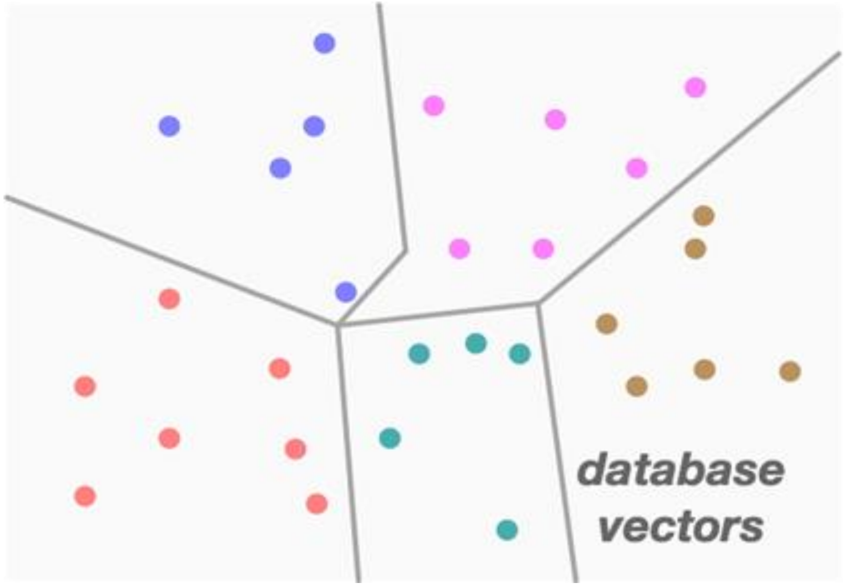


Vector search ≈ approximate nearest neighbor search

IVF-PQ: a popular vector search algorithm in RAG

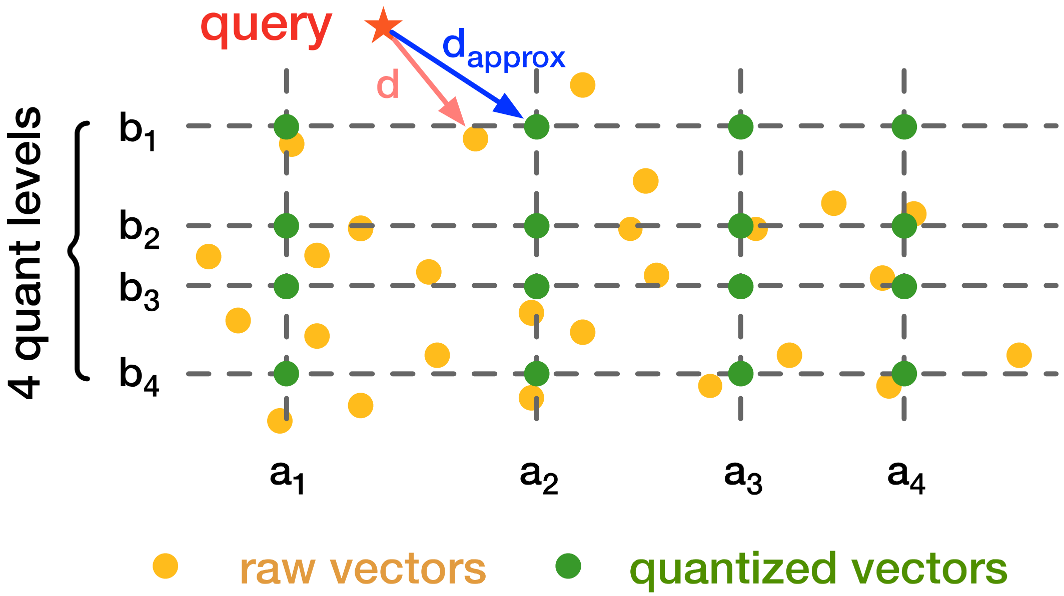
Inverted-file (IVF) index:

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

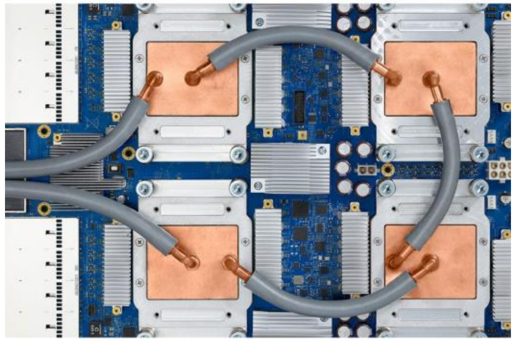
Energy wasted by idle compute units

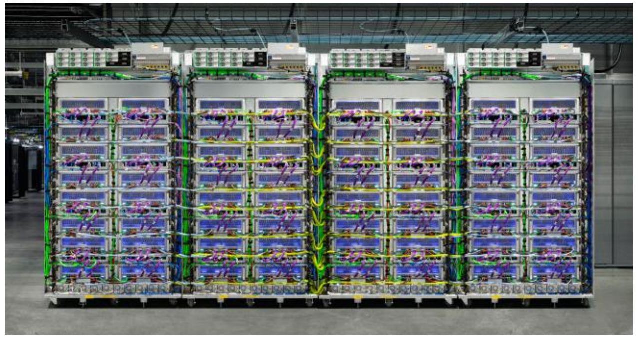
Hardware specialization is increasingly popular

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

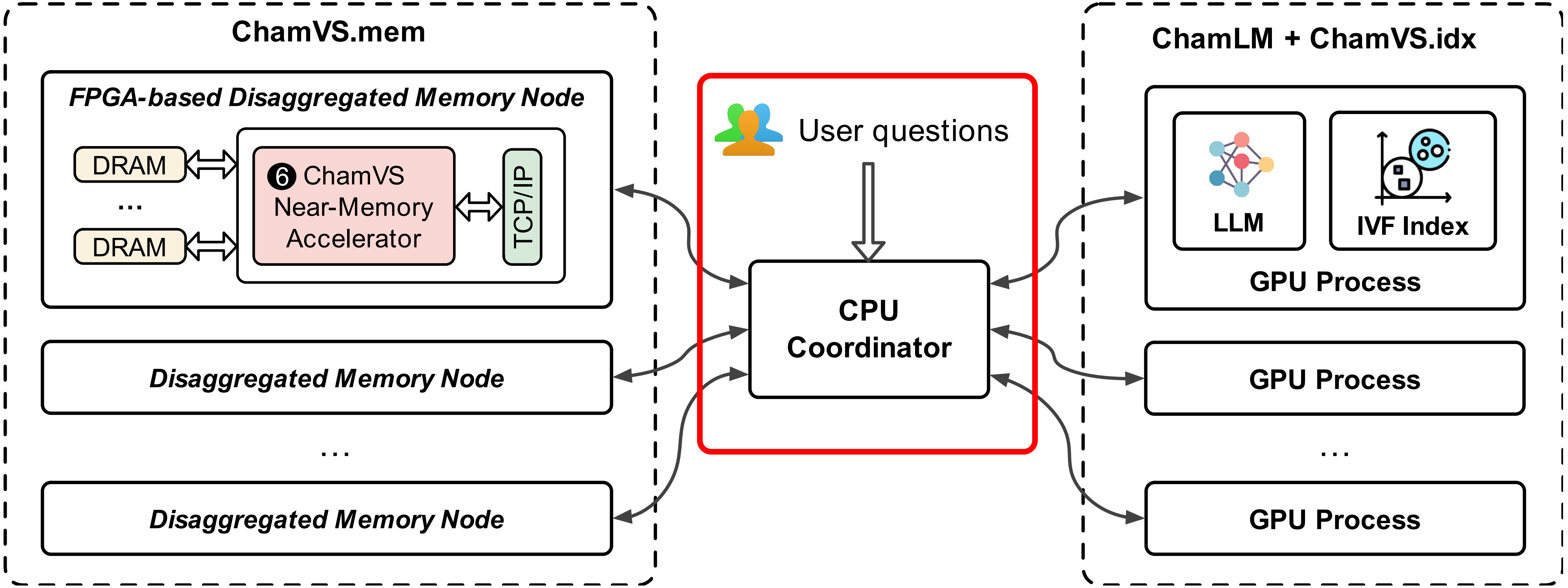
Inference accelerators + vector search accelerators

Requirement: accommodate diverse RAG algorithms

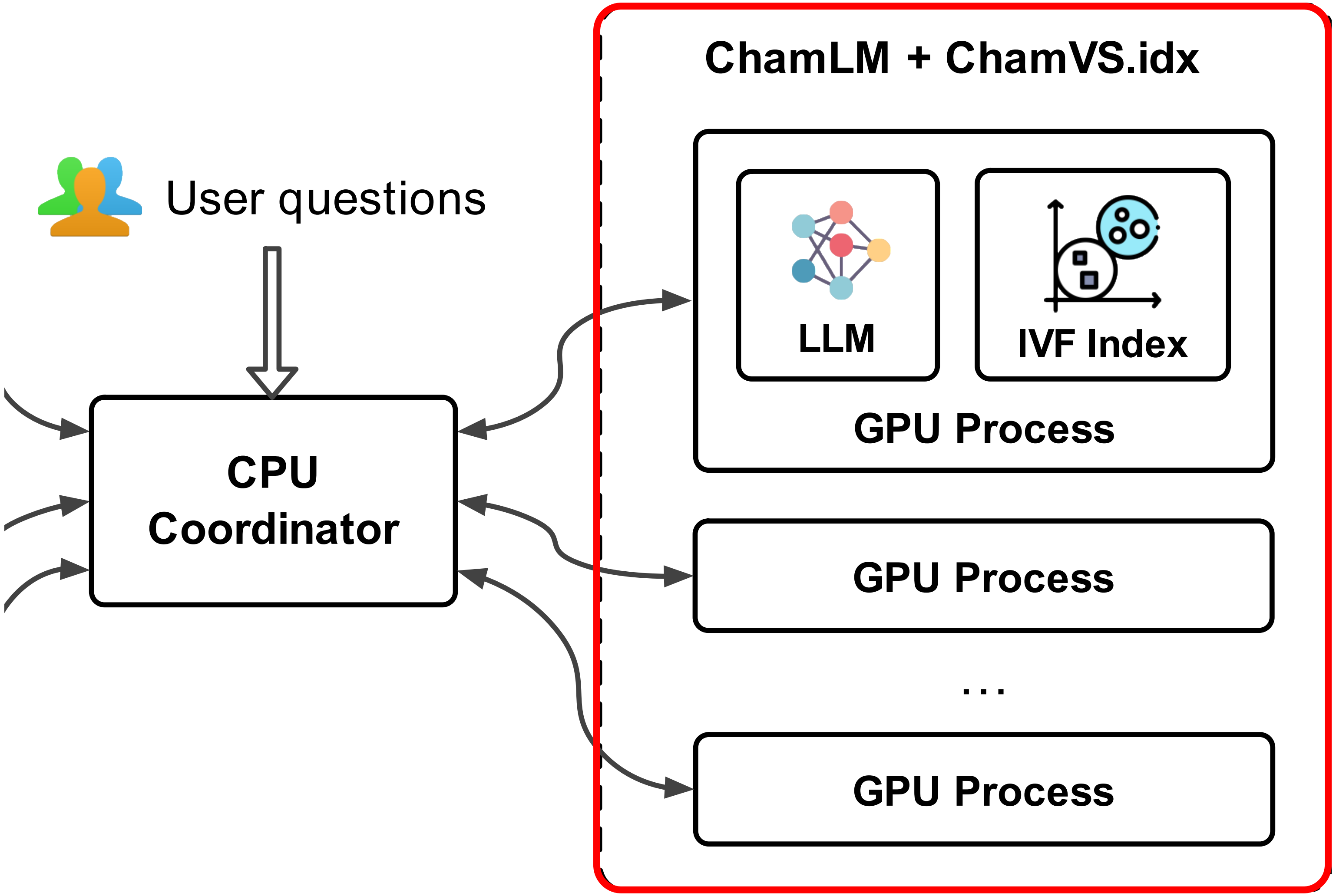
Principle 2: accelerator disaggregation

Handle various performance bottlenecks across RAGs

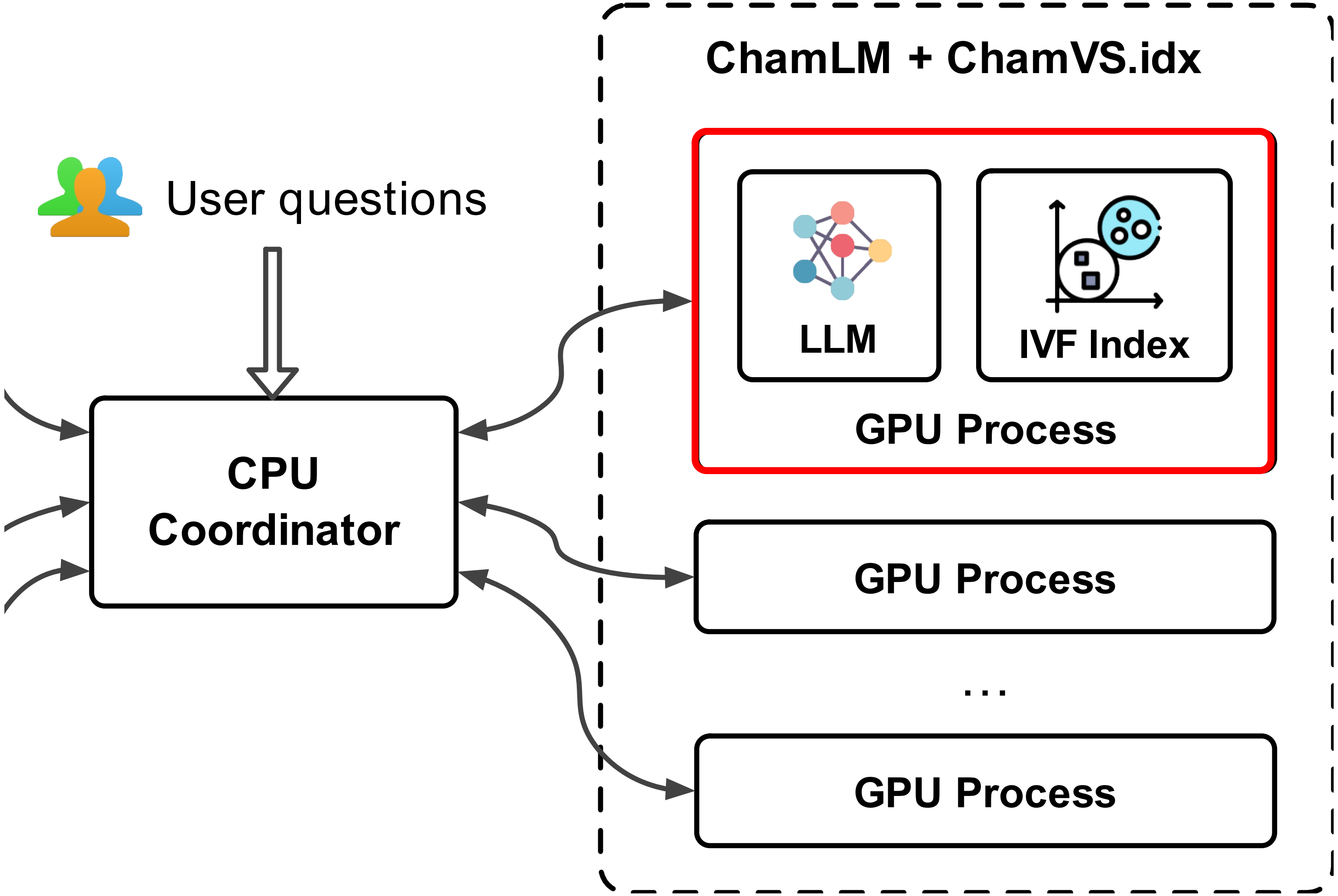
Chameleon: accelerator heterogeneity + disaggregation



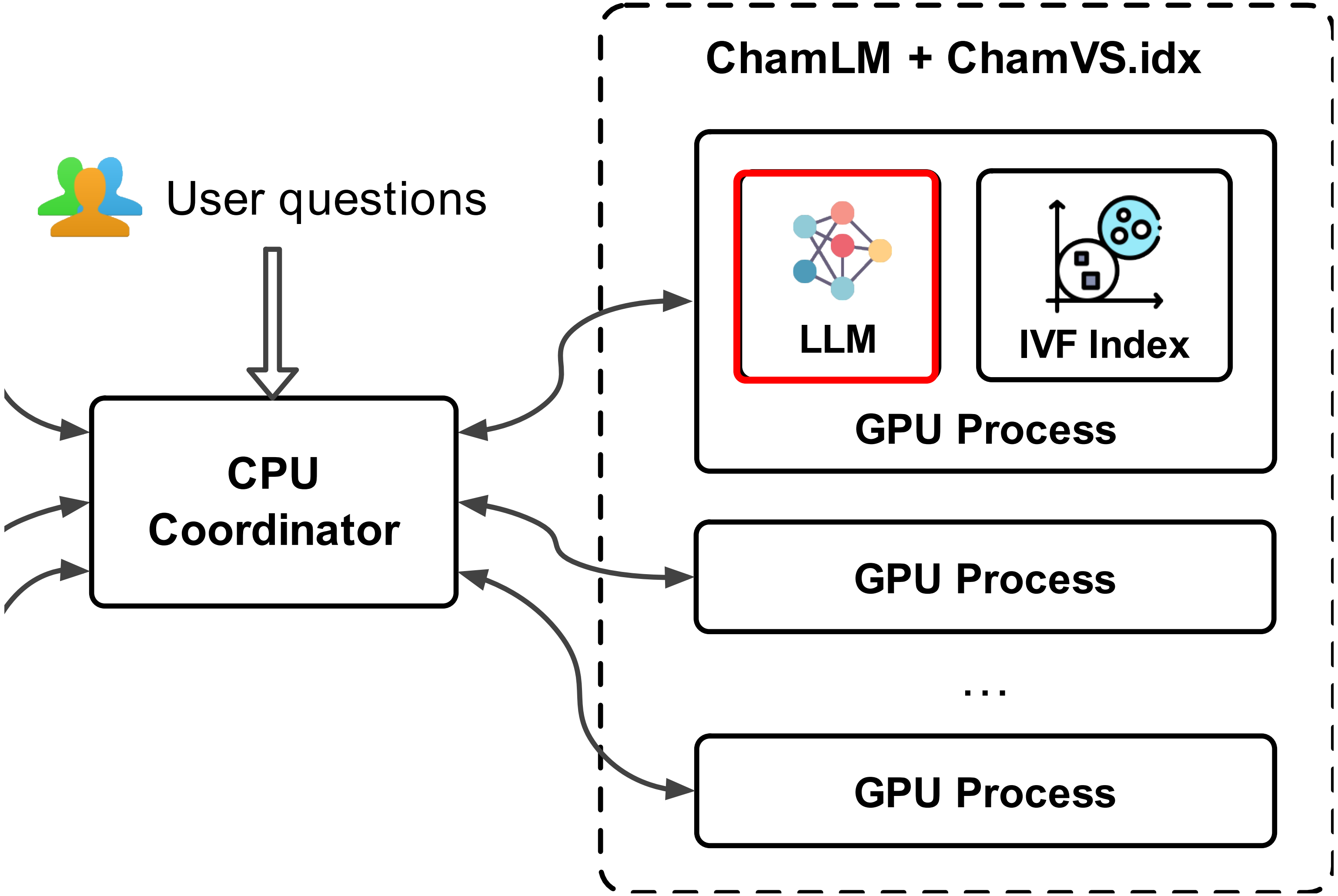
Chameleon: accelerator heterogeneity + disaggregation



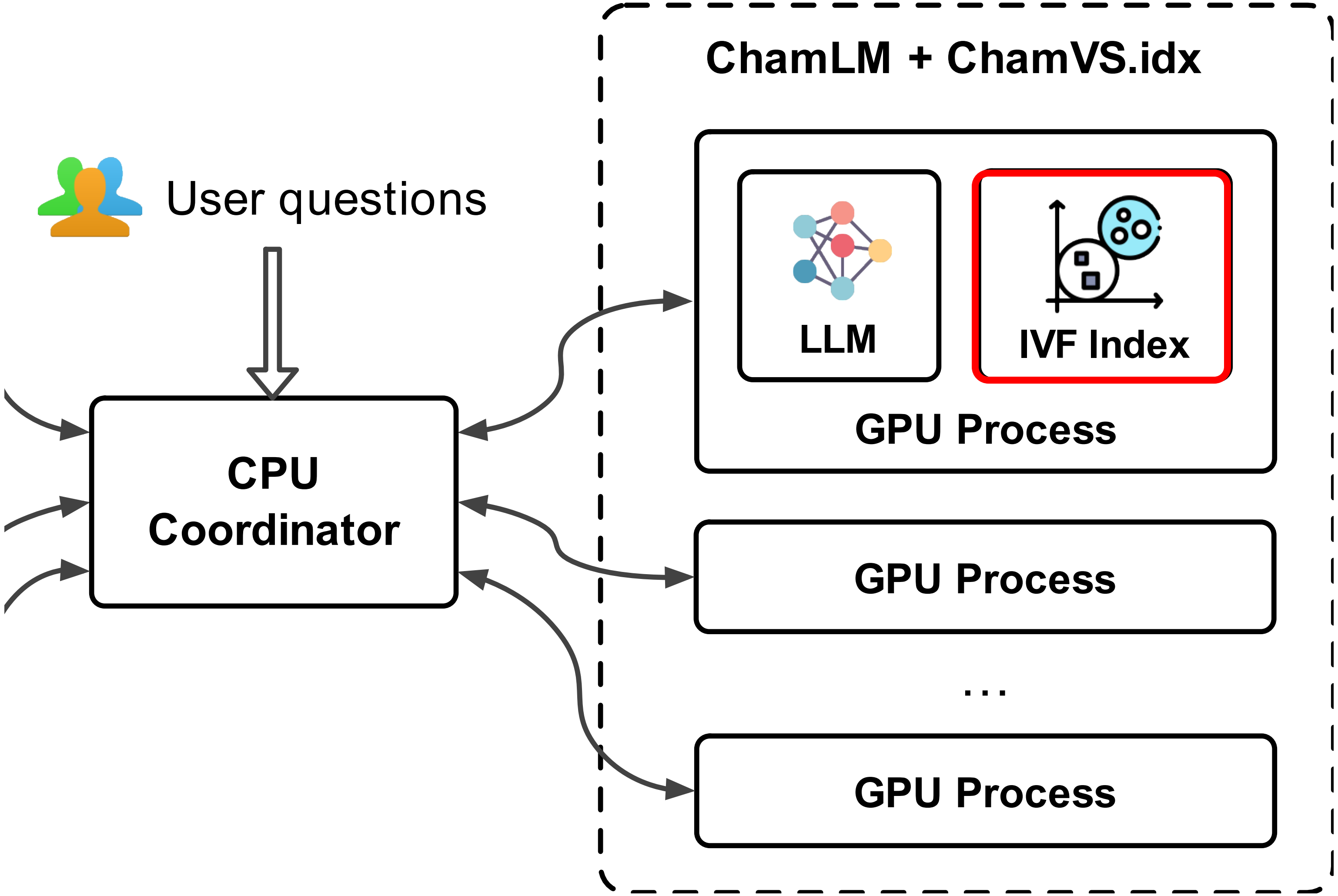
Chameleon: accelerator heterogeneity + disaggregation



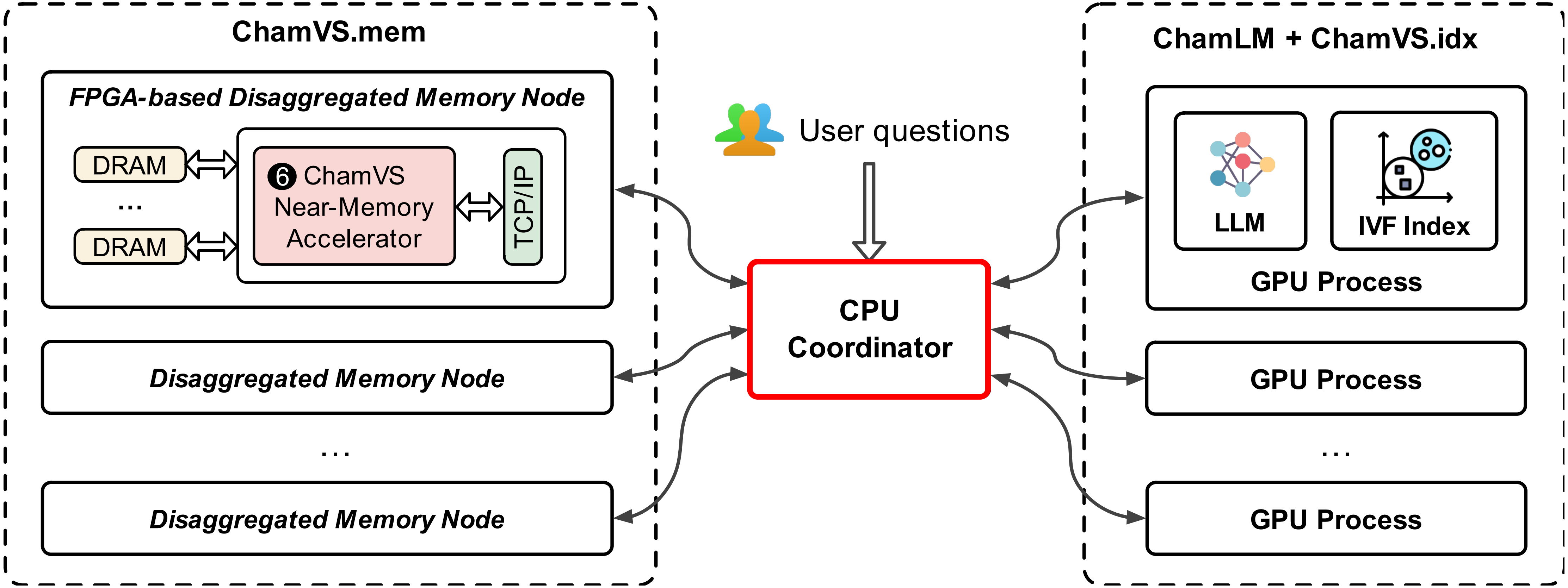
Chameleon: accelerator heterogeneity + disaggregation



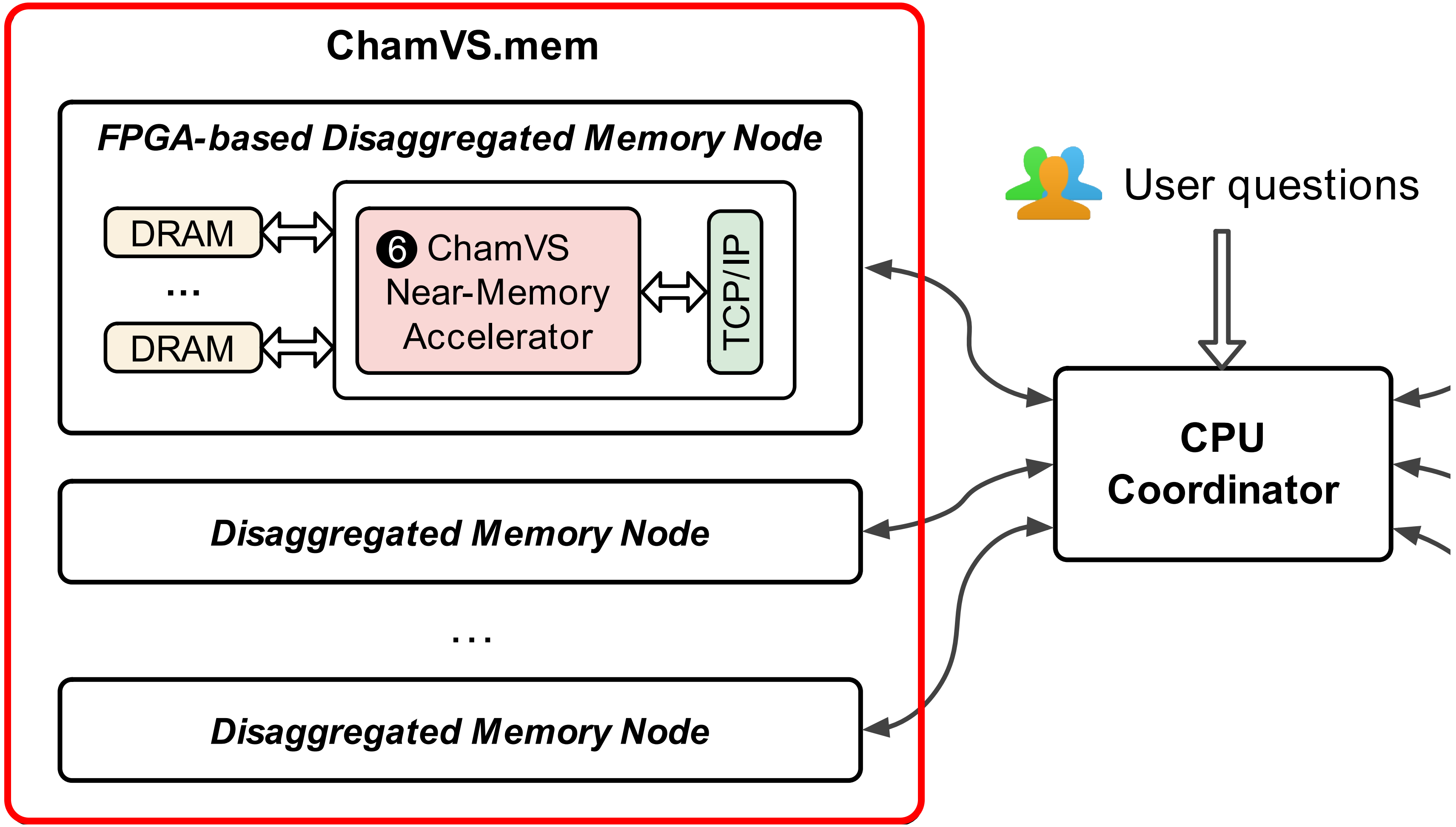
Chameleon: accelerator heterogeneity + disaggregation



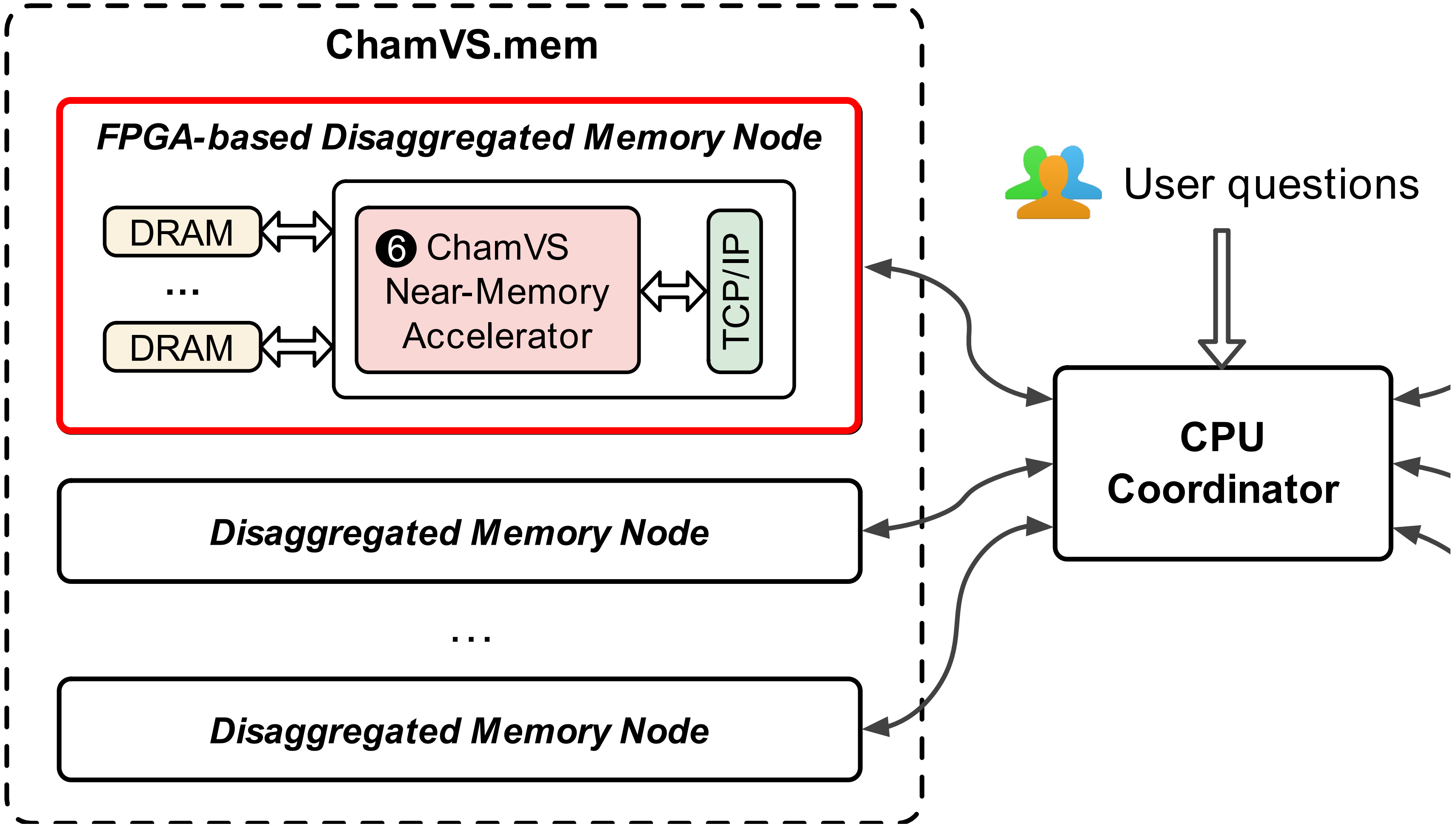
Chameleon: accelerator heterogeneity + disaggregation



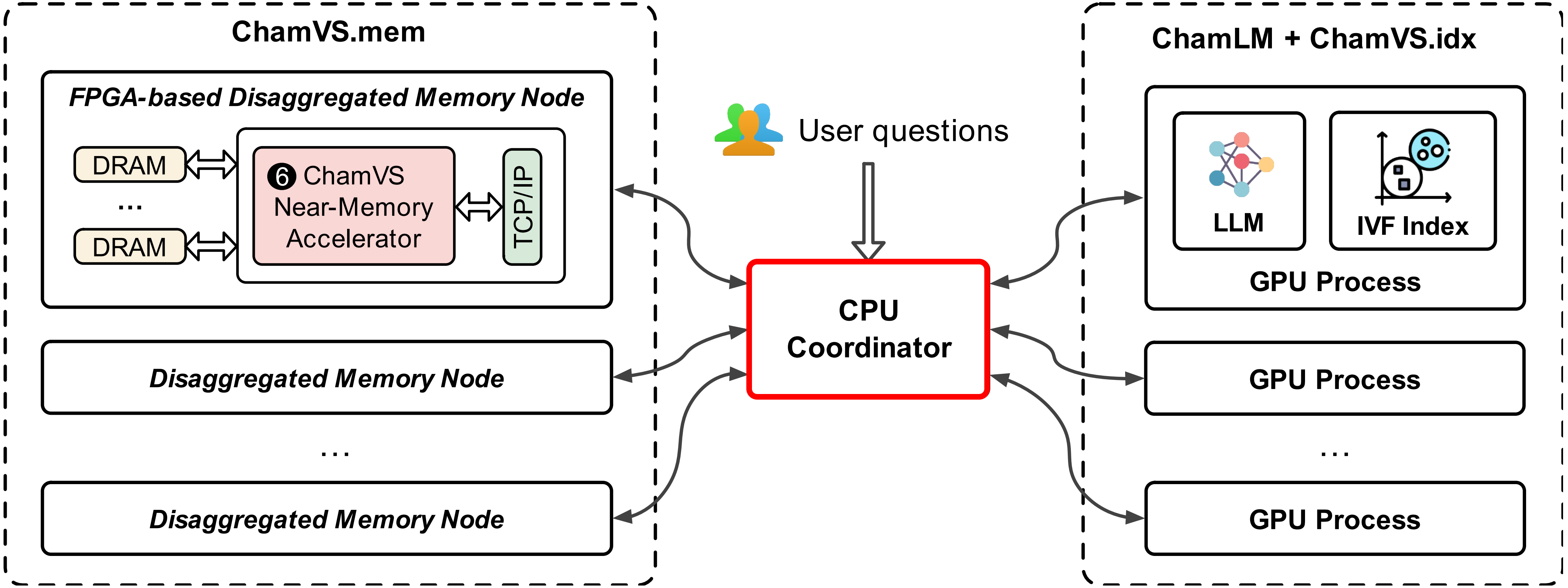
Chameleon: accelerator heterogeneity + disaggregation



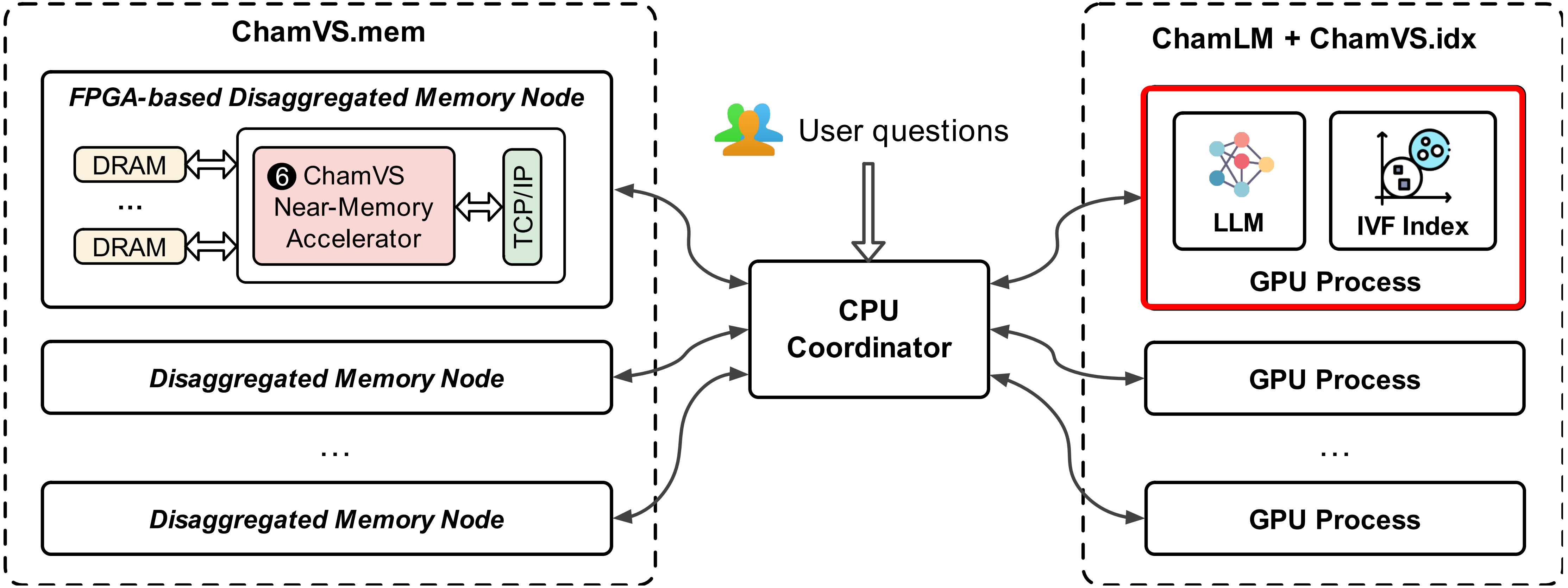
Chameleon: accelerator heterogeneity + disaggregation



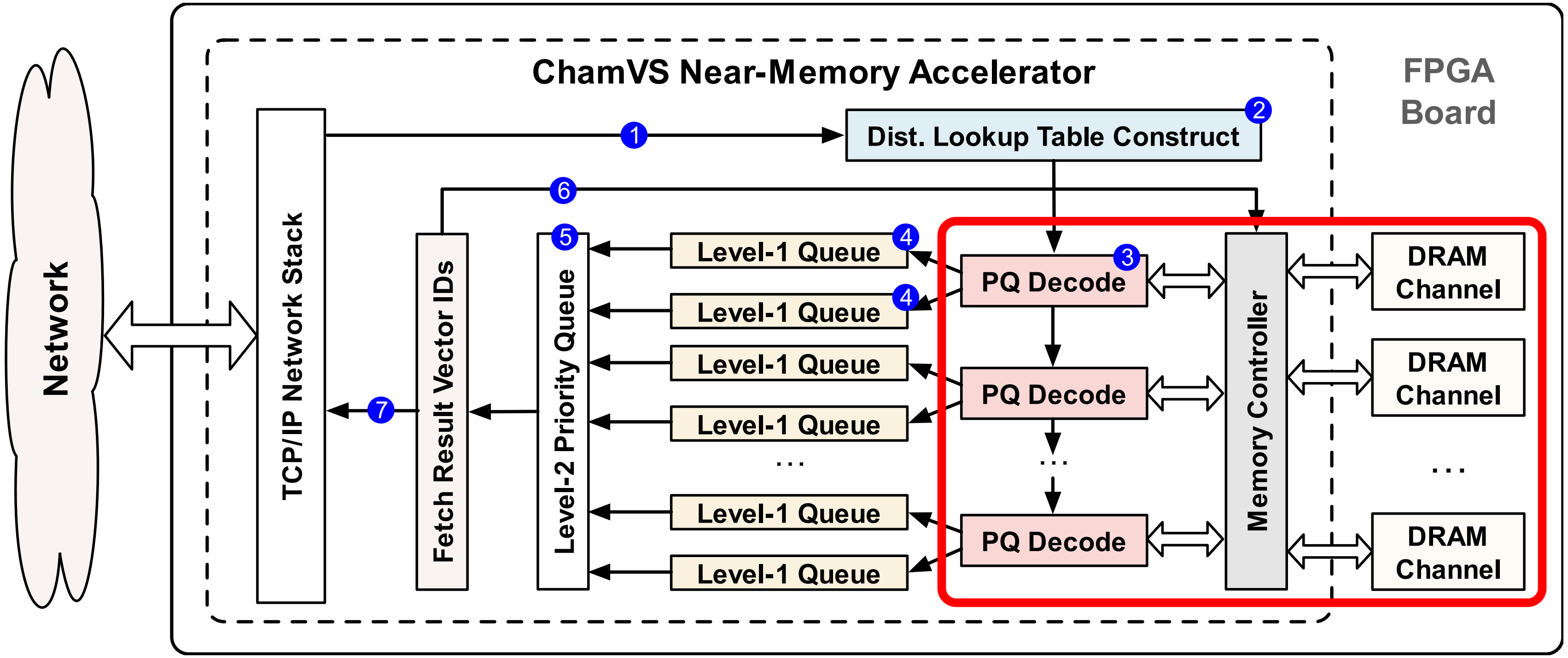
Chameleon: accelerator heterogeneity + disaggregation



Chameleon: accelerator heterogeneity + disaggregation



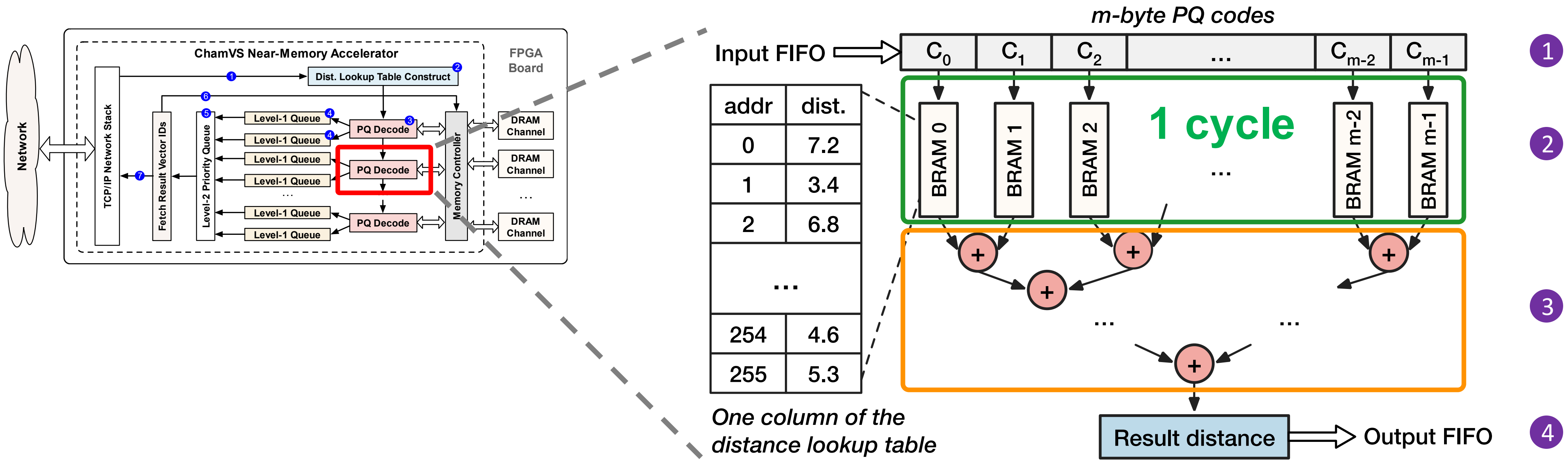
ChamVS: near-memory retrieval acceleration



Compared to CPUs: **faster PQ decoding**

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

ChamVS: near-memory retrieval acceleration

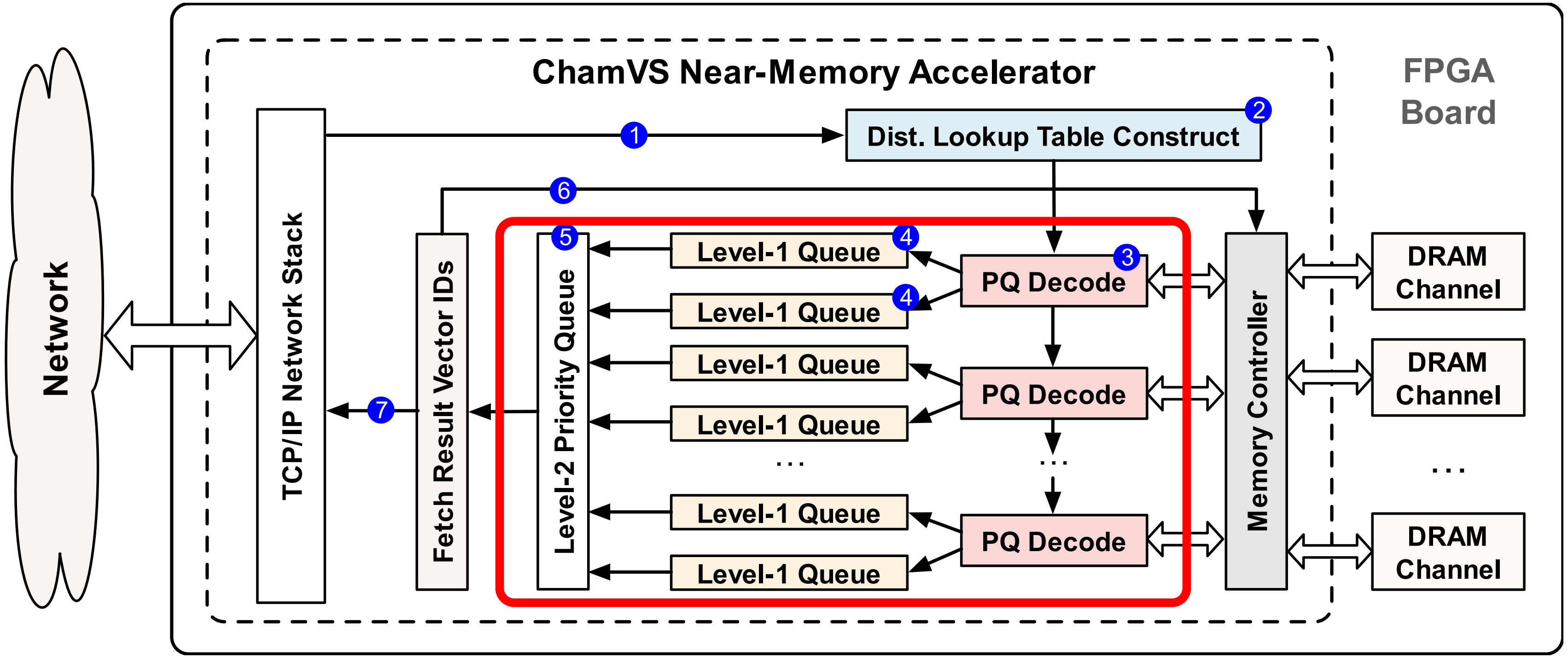


Parallel lookup + **Parallel computation** + **Pipeline parallelism**

=

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

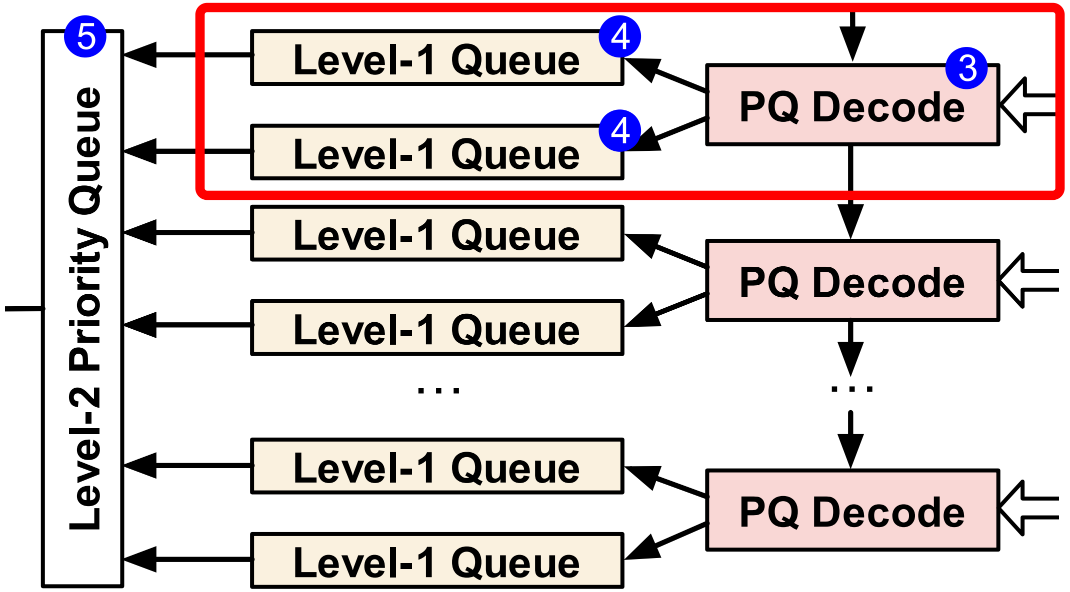
Systolic priority queue:

High throughput

one ingestion / two cycles

High resource consumption

queue length x queue num

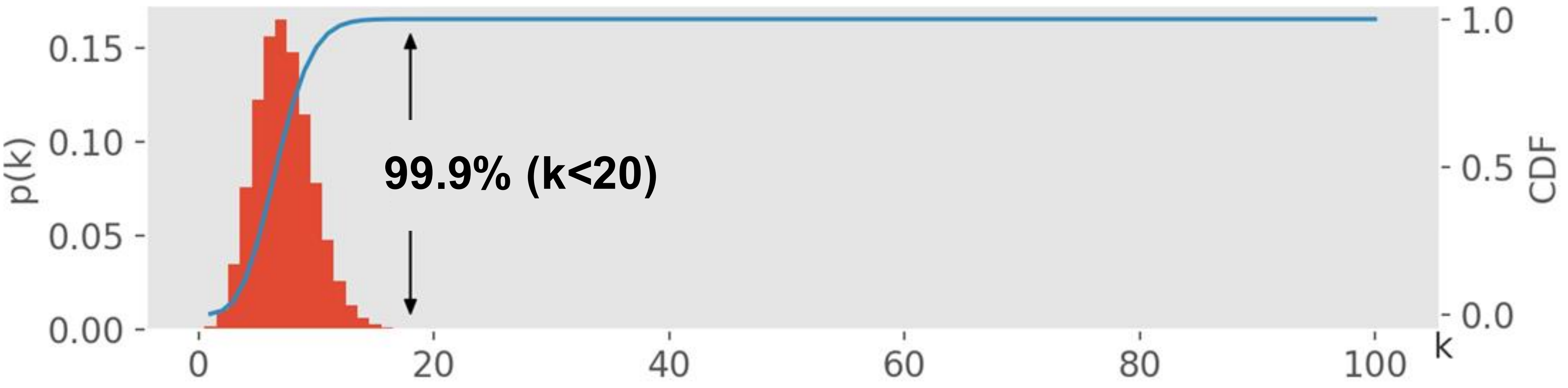


Question: how to reduce hardware resource consumption?

Approximate hierarchical priority queue

Example: 16 queues to collect 100 nearest neighbors

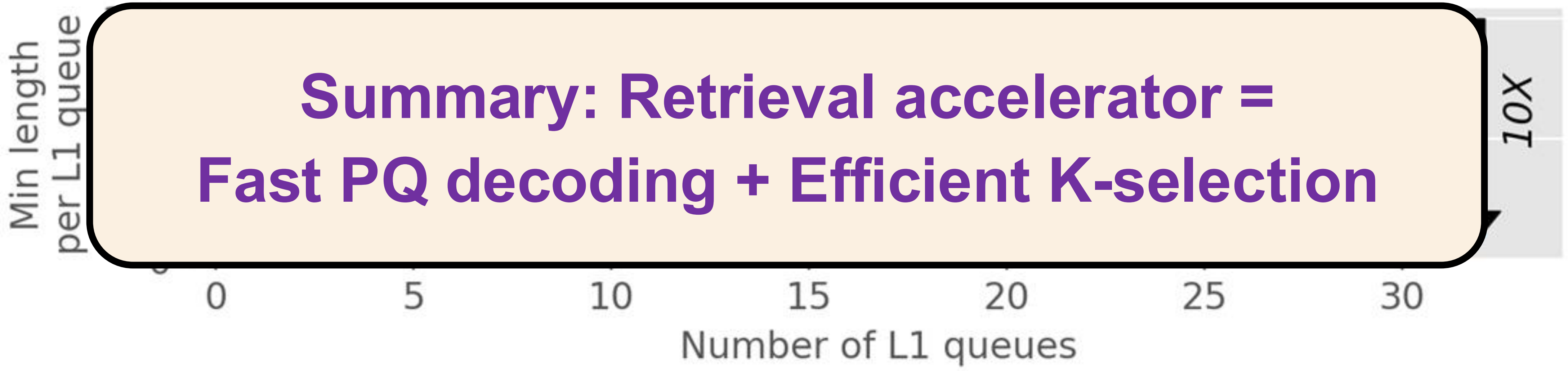
Is it likely that all 100 results are located in one queue?



Finding: Most queues only contain less than 20 results

Approximate hierarchical priority queue

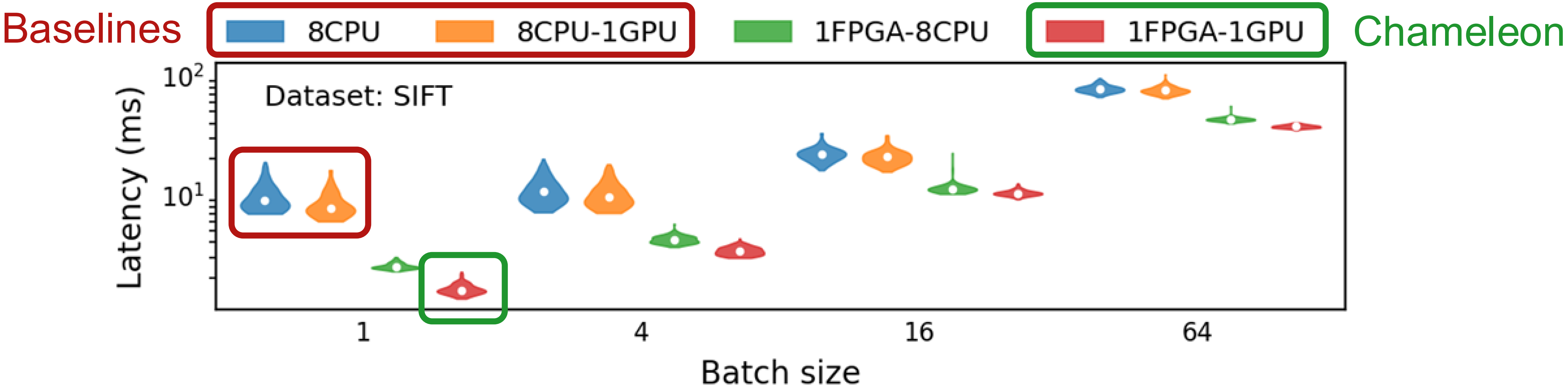
Idea: Truncate the queues significantly while achieving similar K-selection quality (e.g., 99% identical results)



Summary: Retrieval accelerator =
Fast PQ decoding + Efficient K-selection

10x resource saving without notable recall degrade

Vector search performance and energy efficiency

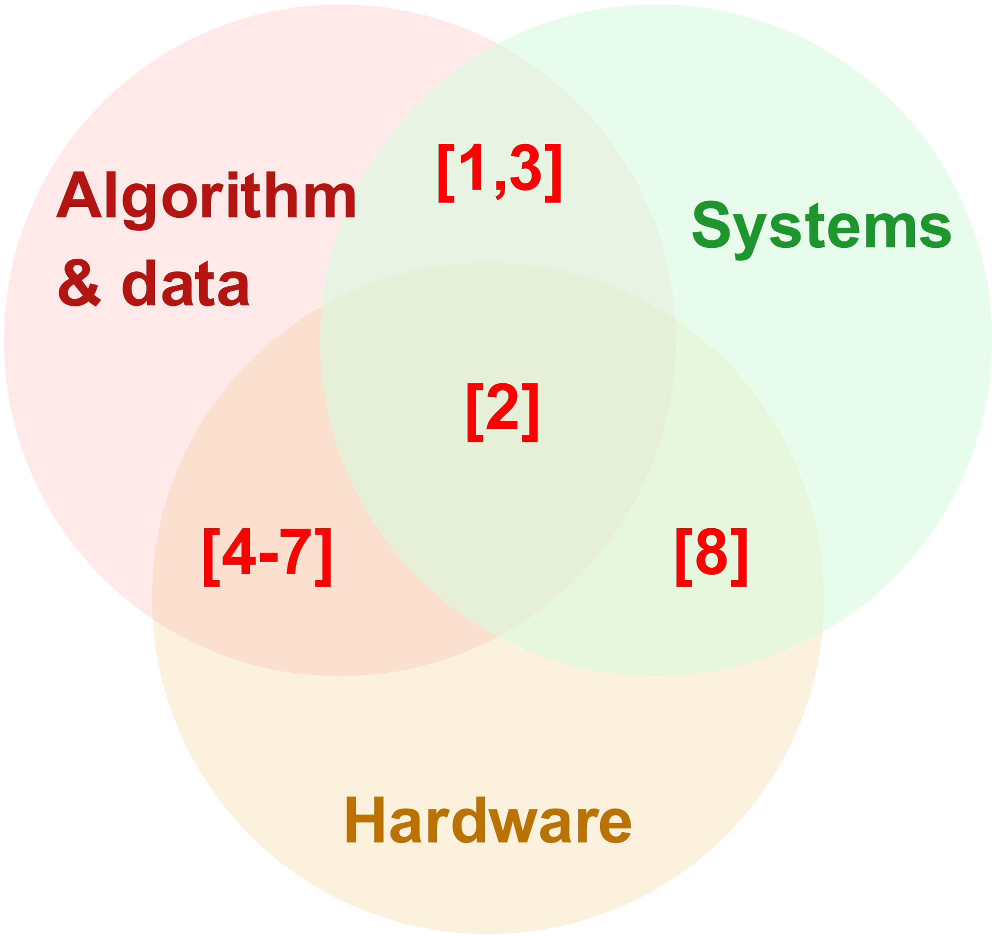


Chameleon achieves up to **16.6x speedup over CPU** baseline

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