METIS: Fast Quality-Aware RAG Systems with Configuration Adaptation

SOSP '25

Siddhant Ray[1], Rui Pan[2], Zhuohan Gu[1], Kuntai Du[1 3], Shaoting Feng[1], Ganesh Ananthanarayanan[4], Ravi Netravali[2], Junchen Jiang[1 3]

[1]University of Chicago, [2]Princeton University, [3]TensorMesh, [4]Microsoft

Xiaoqi Li @Reading Group 2025/12/30

LLMs: A New Paradigm, But Not Perfect

- LLMs are enabling breakthroughs across many fields
  - e.g. code generation, creative writing, conversations, ...
- However, they suffer from inherent limitations
  - Hallucination, knowledge cutoff, high cost & latency, …

Who is the first person to walk on Mars?

Commander Ivan Kuznetsov on the fictional Ares 7 mission in 2035….

The Hallucination Problem

What was the biggest news story of yesterday?

My knowledge cutoff is in early 2024...

The Knowledge cutoff Problem

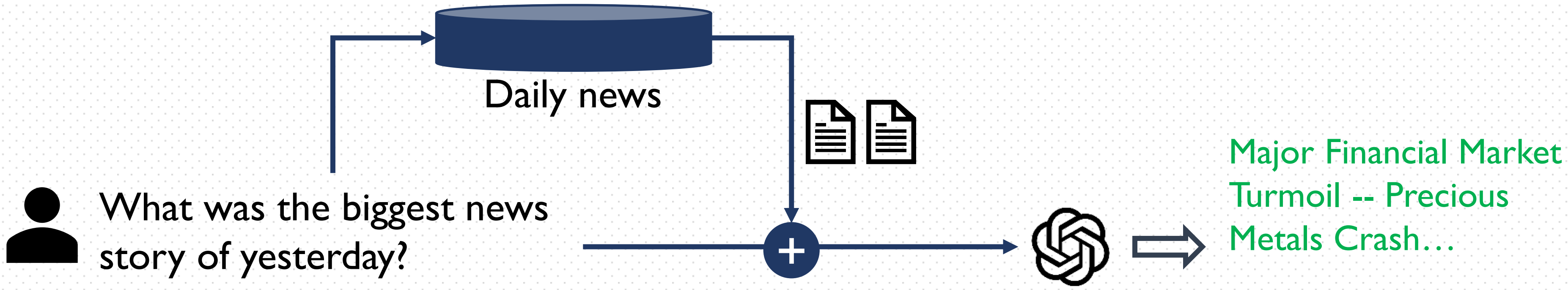
………

$ $

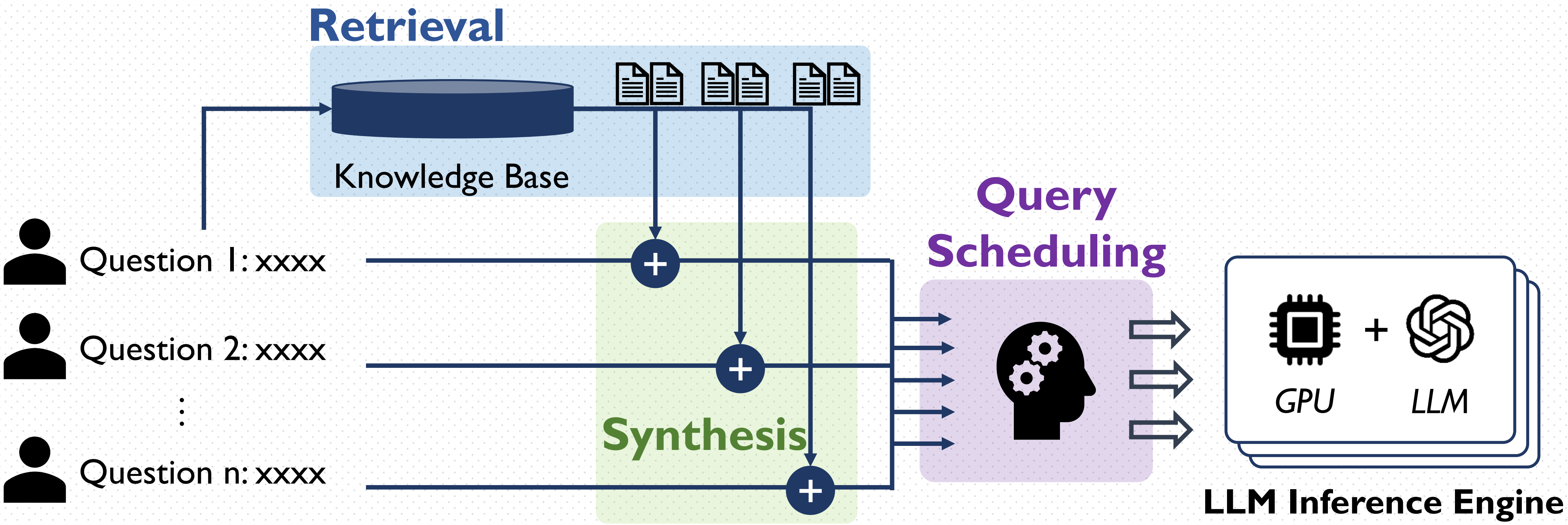
High cost & Latency

RAG: Augmenting LLMs with External Knowledge

- ❑ RAG (Retrieval Augmented Generation)
  - ❖ Retrieves relevant information from external knowledge bases
  - ❖ Feeds this context to the LLM along with the user's query
- ❑ This approach makes LLM responses more:
  - ❖ Factual & traceable
  - ❖ Up-to-date & relevant



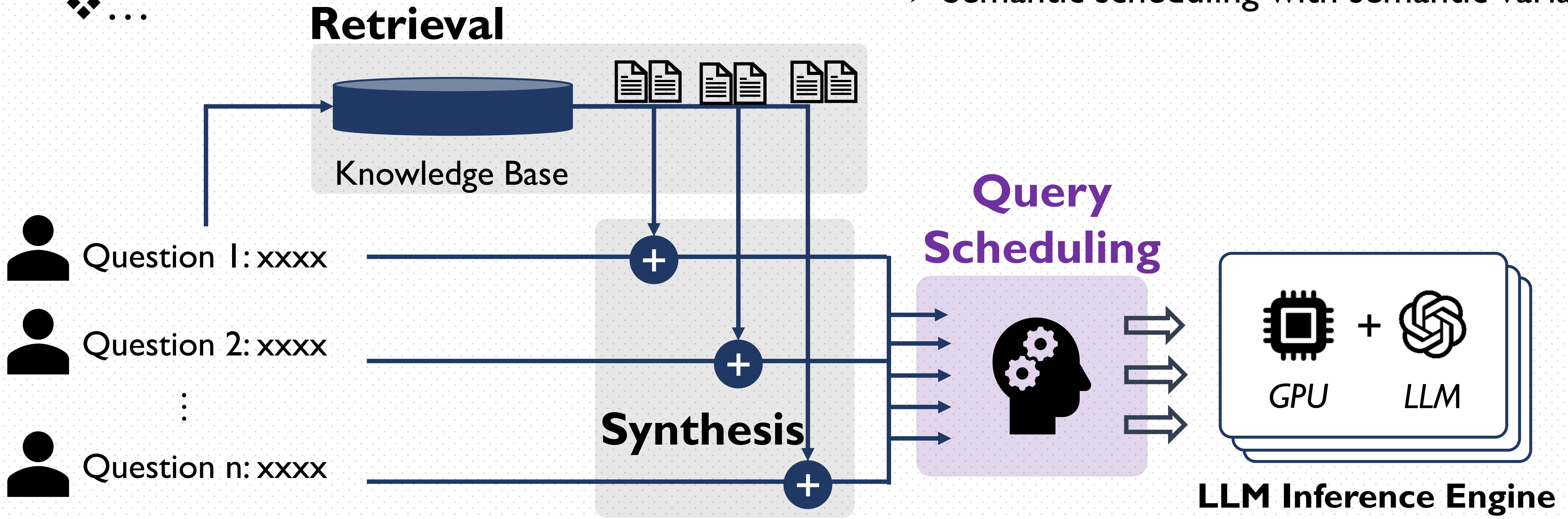
Optimization Opportunities in RAG Systems



Optimization Opportunities in RAG Systems

❑ Query scheduling
- ❖ Batching strategy
- ❖ Request reordering
- ❖ Computation reuse
- ❖ …

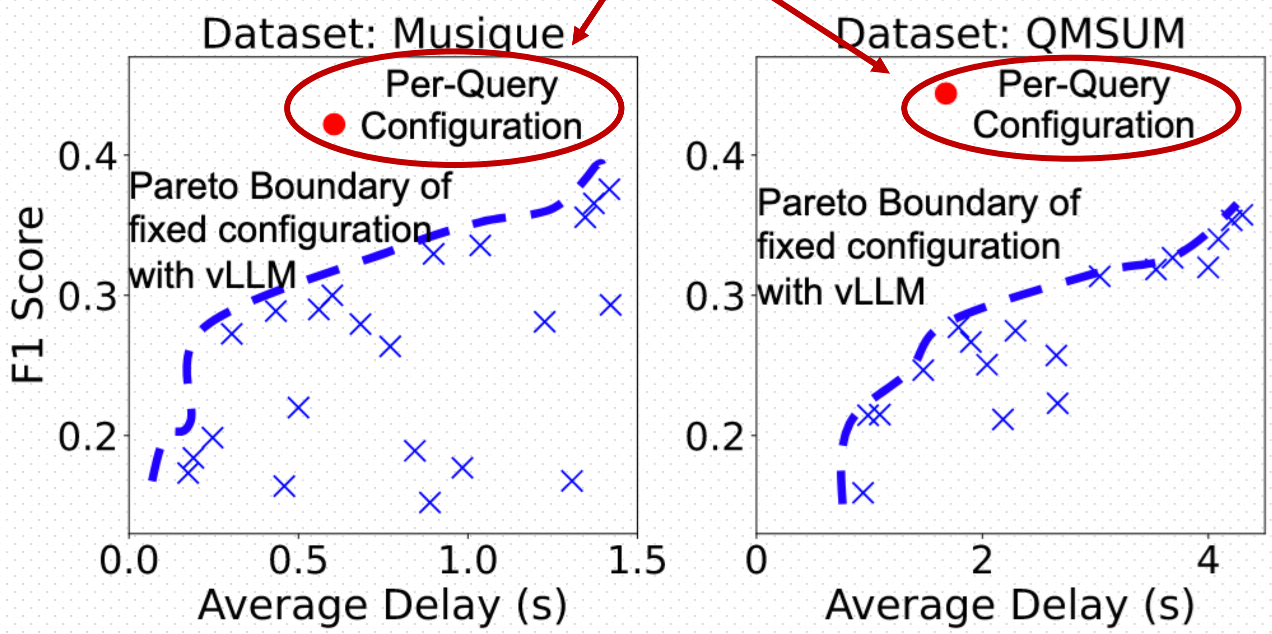
❑ Prior work
- ❖ vLLM (SOSP '23)
  - ➢ Continuous batching with PagedAttention
- ❖ Parrot (OSDI '24)
  - ➢ Semantic scheduling with semantic variable



Optimization Opportunities in RAG Systems

- ❑Per-query RAG configuration

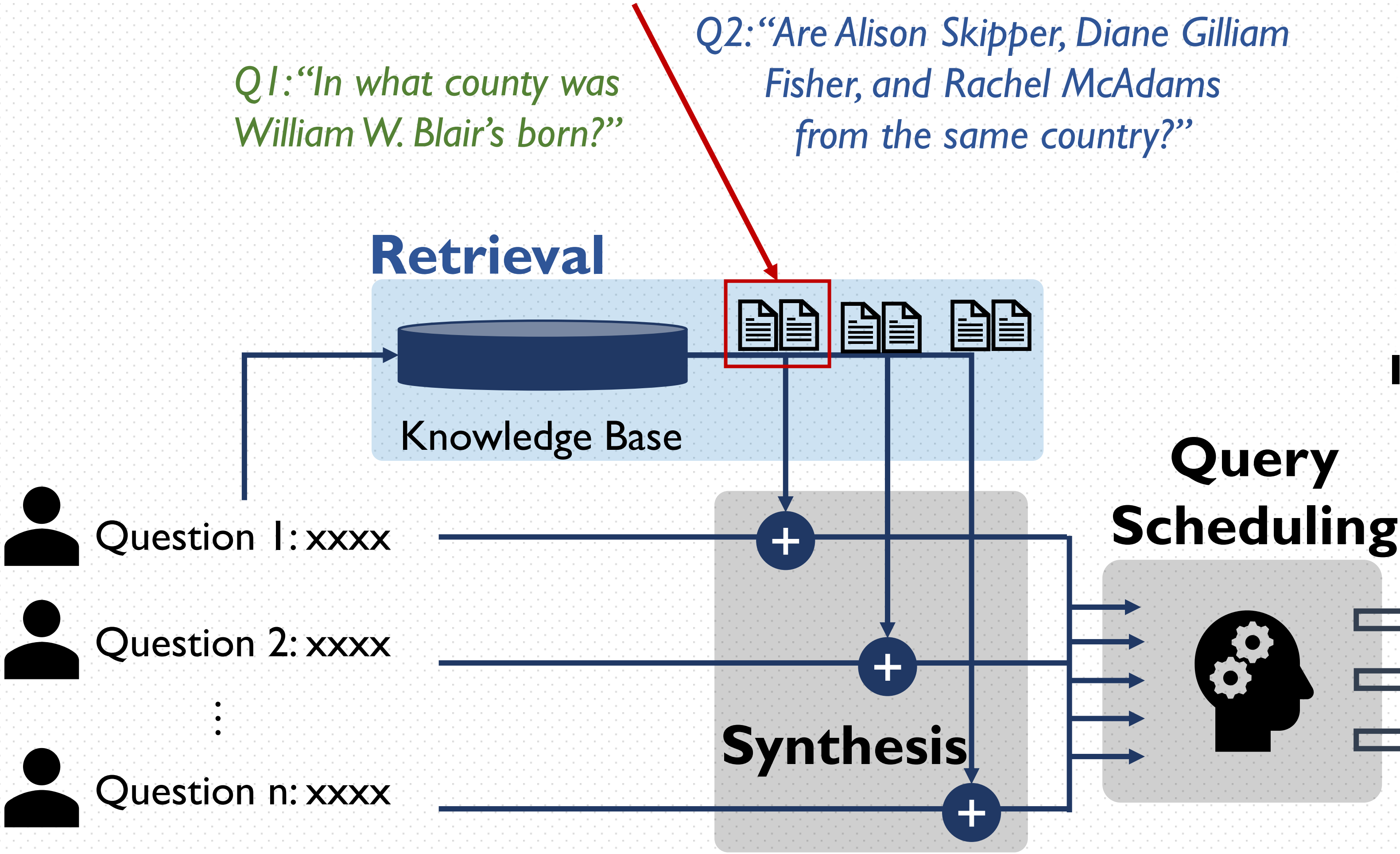
Per-query configuration can achieve significantly better quality-delay tradeoffs

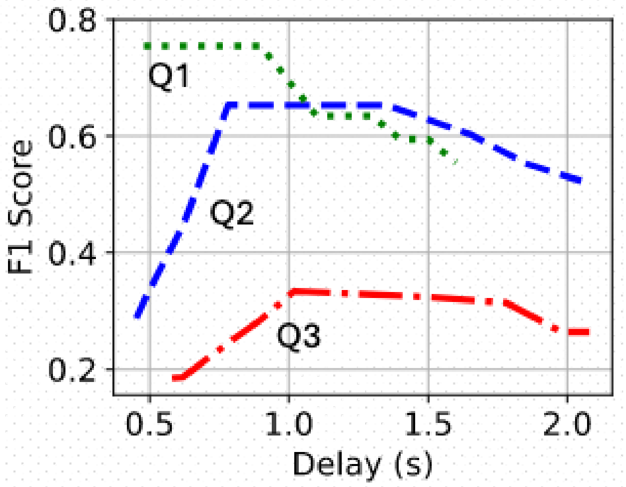


Per-query configuration vs. static configuration

Optimization Opportunities in RAG Systems

- ❑ Per-query RAG configuration
  - ❖ How many text chunks to retrieve?

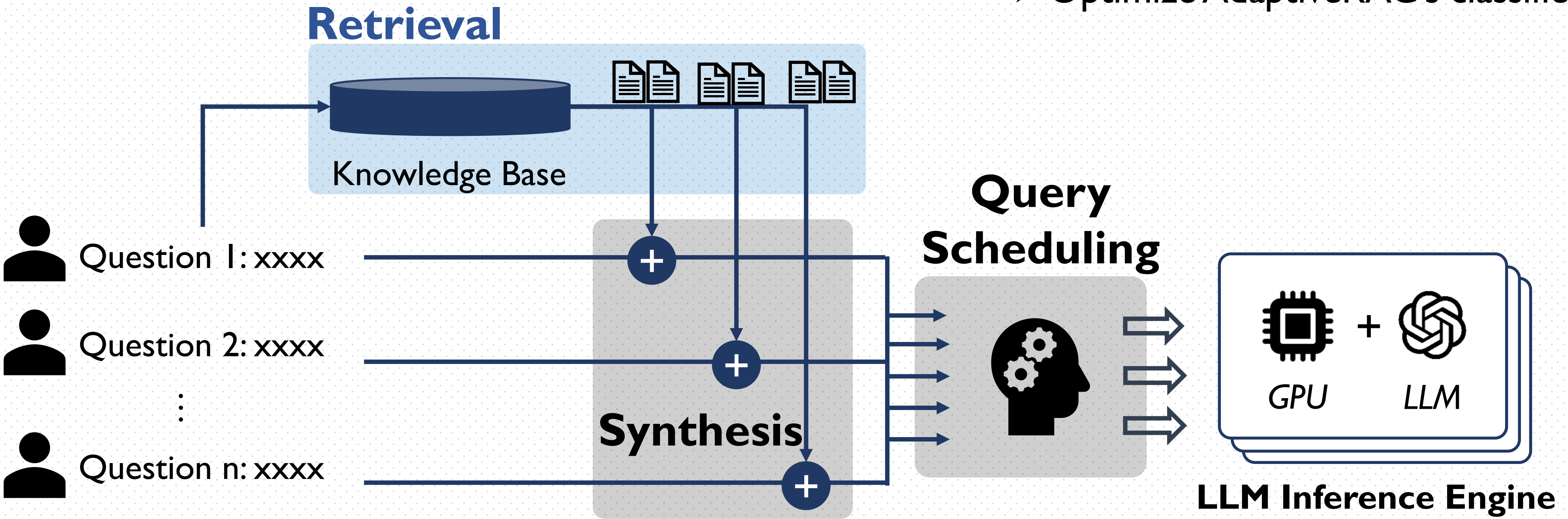




Increase # of chunks from 1 to 35

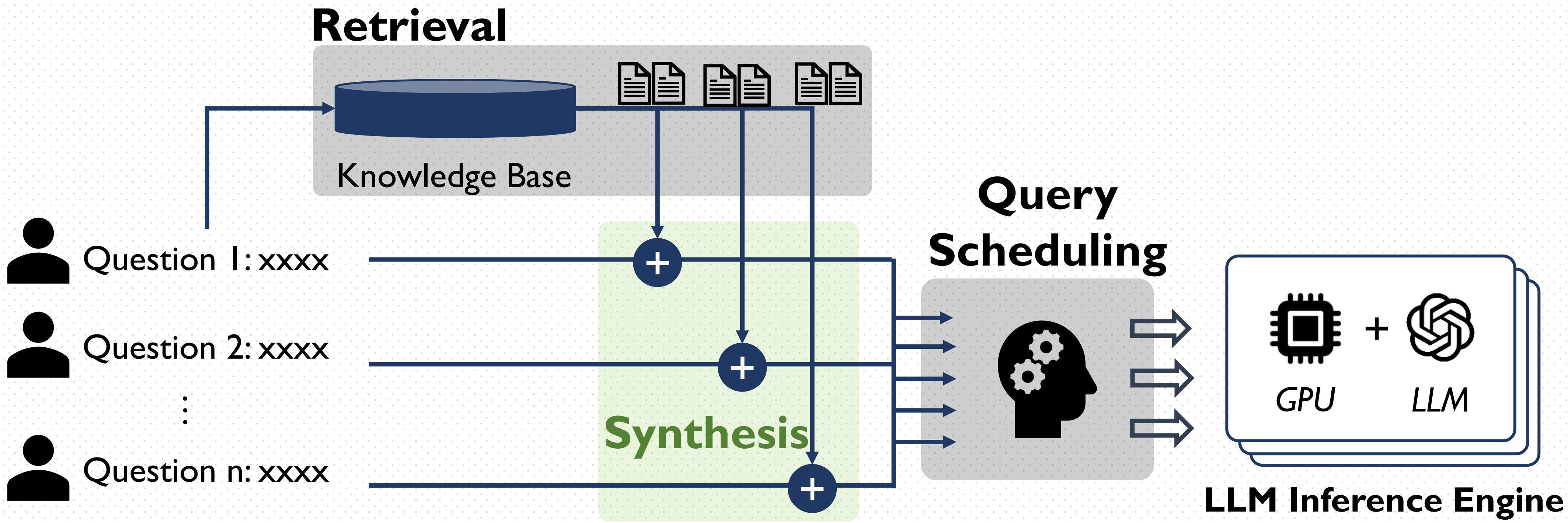
Optimization Opportunities in RAG Systems

- ❑Per-query RAG configuration
  - ❖How many text chunks to retrieve?
- ❑Prior work
  - ❖AdaptiveRAG (NAACL '24)
    - ➢Use a classifier to decide #chunks
  - ❖MBA-RAG (COLING '25)
    - ➢Optimize AdaptiveRAG's classifier



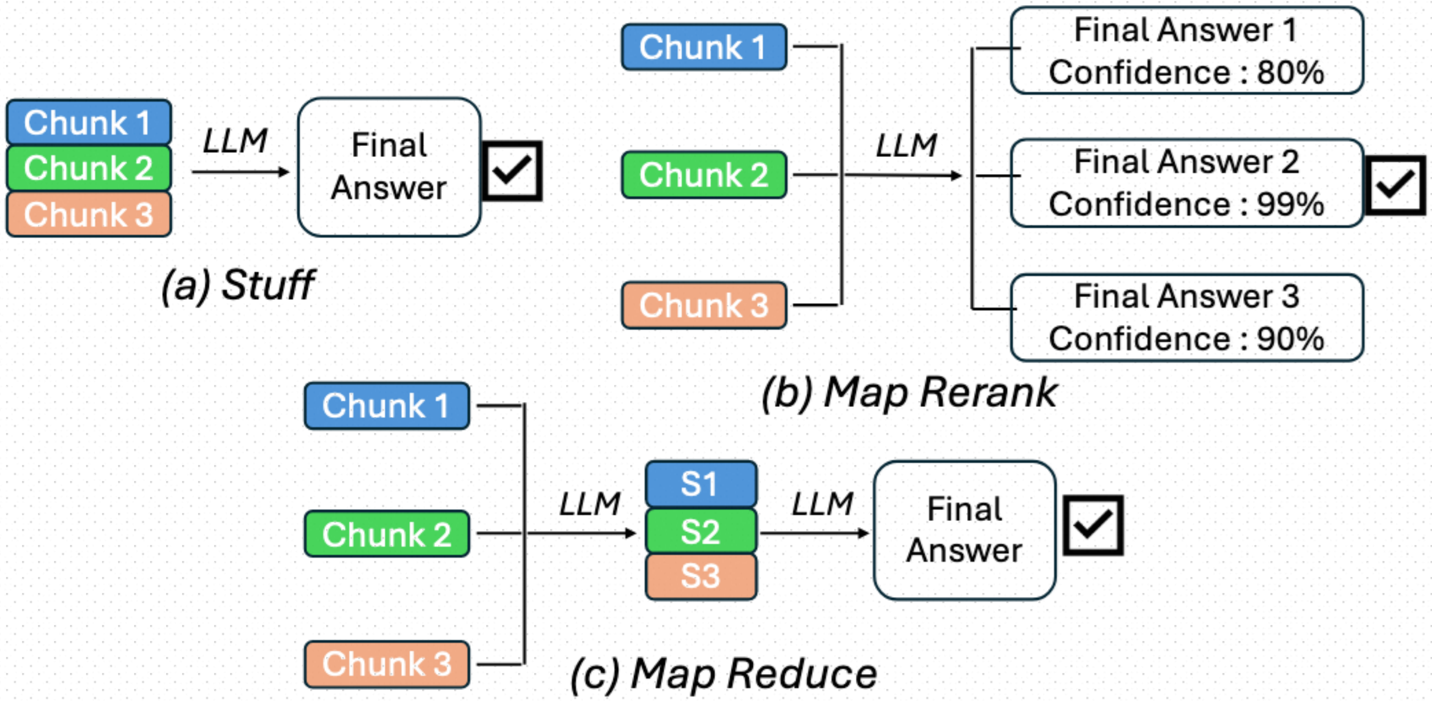
Optimization Opportunities in RAG Systems

- ❑Per-query RAG configuration
  - ❖How to synthesize the retrieved chunks?



Optimization Opportunities in RAG Systems

- ❑Per-query RAG configuration
  - ❖How to synthesize the retrieved chunks?



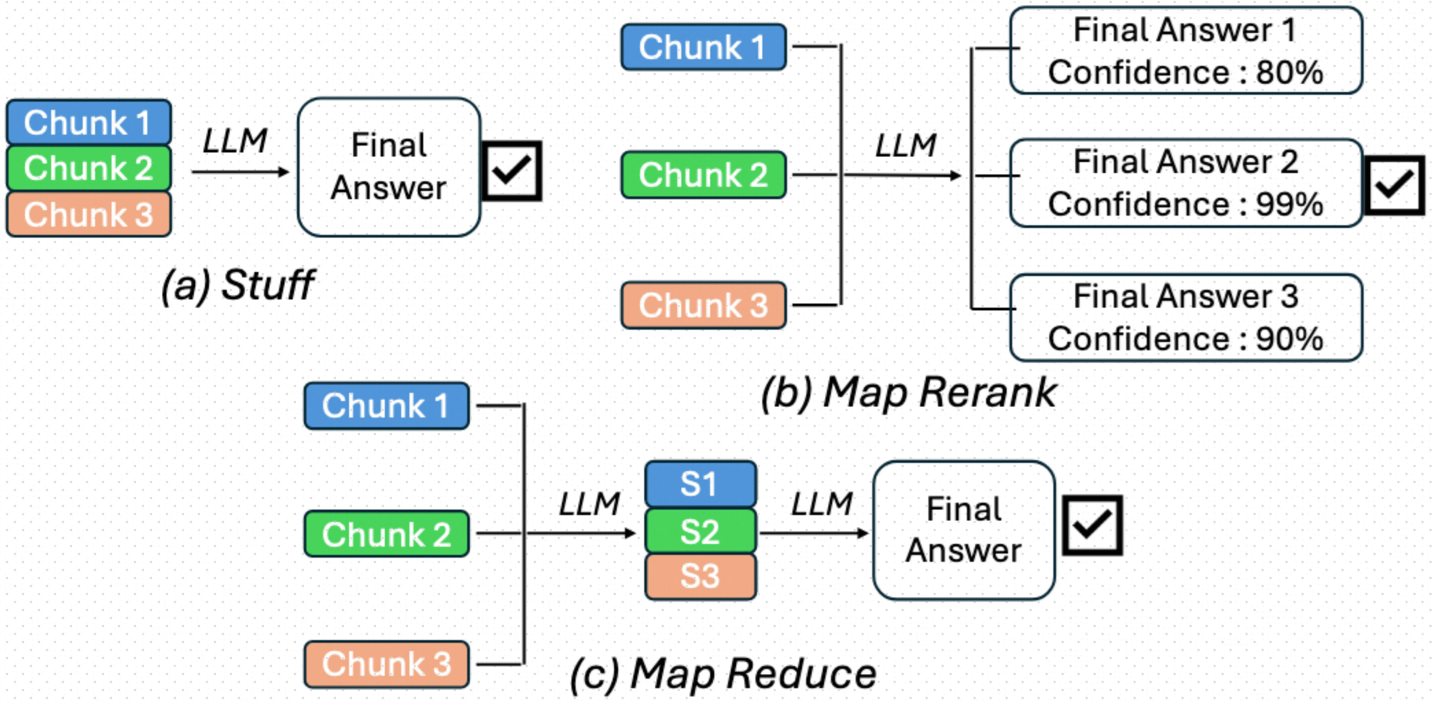
Different RAG synthesis methods

	Stuff	Map Rerank	Map Reduce
Quality	★★	★	★★★
Speed	★★★	★★	★
Scalability	★	★★★	★★★

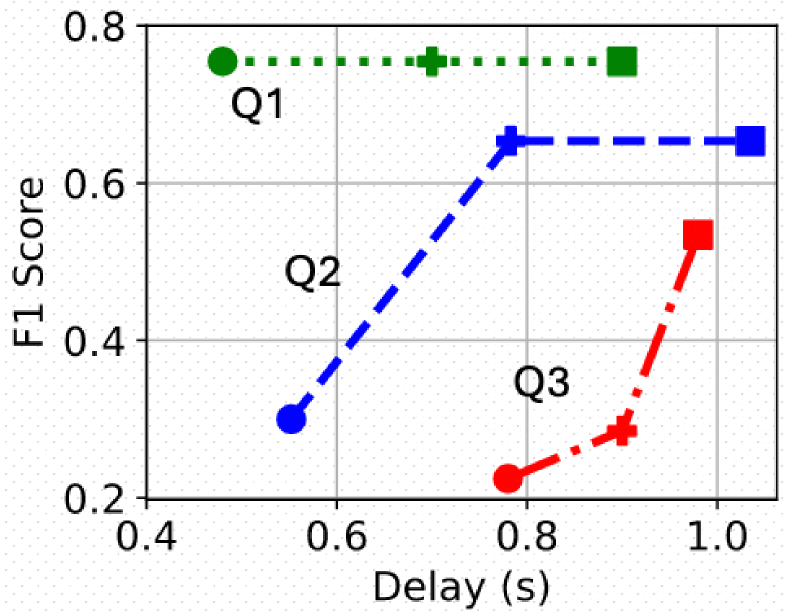
Comparison of different RAG synthesis methods

Optimization Opportunities in RAG Systems

- Per-query RAG configuration
  - How to synthesize the retrieved chunks?



Different RAG synthesis methods

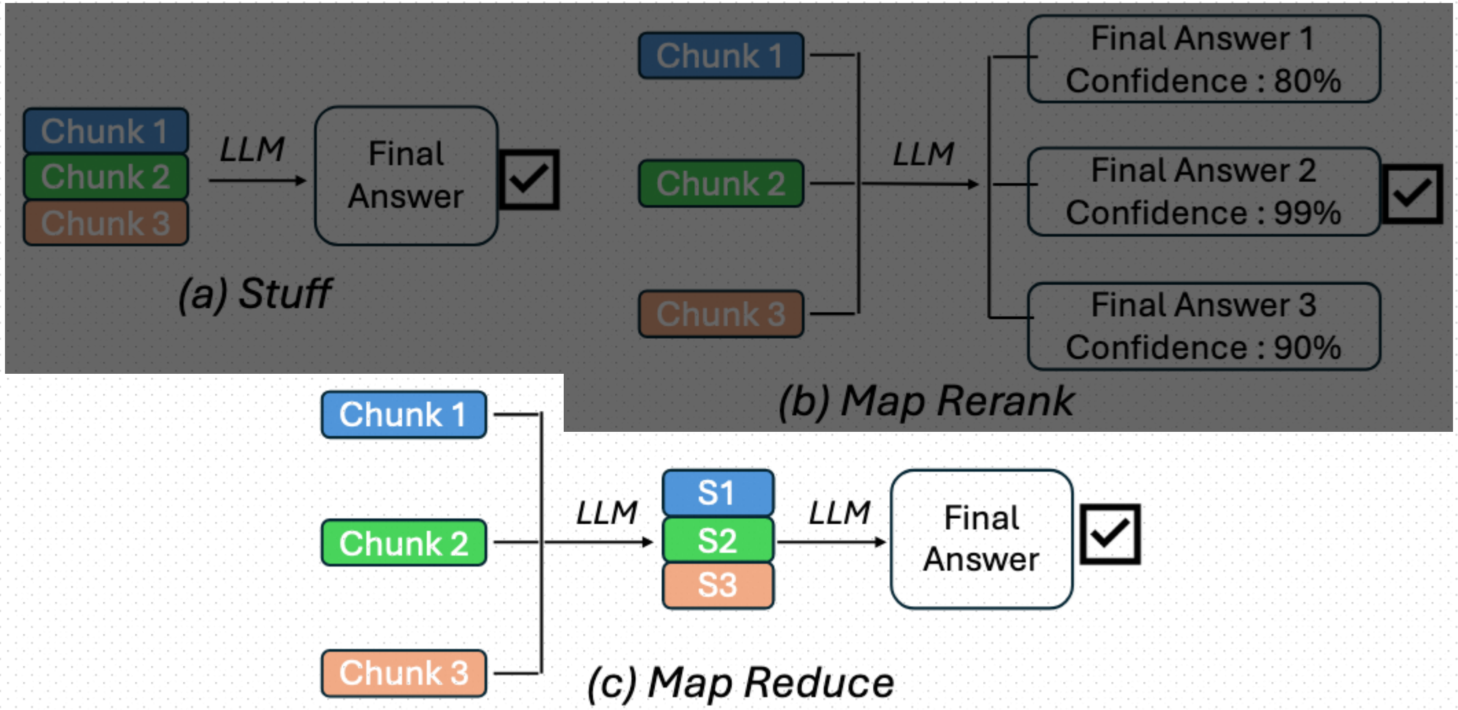


Change synthesis method from map_rerank (circle) , stuff (plus) to map_reduce (square)

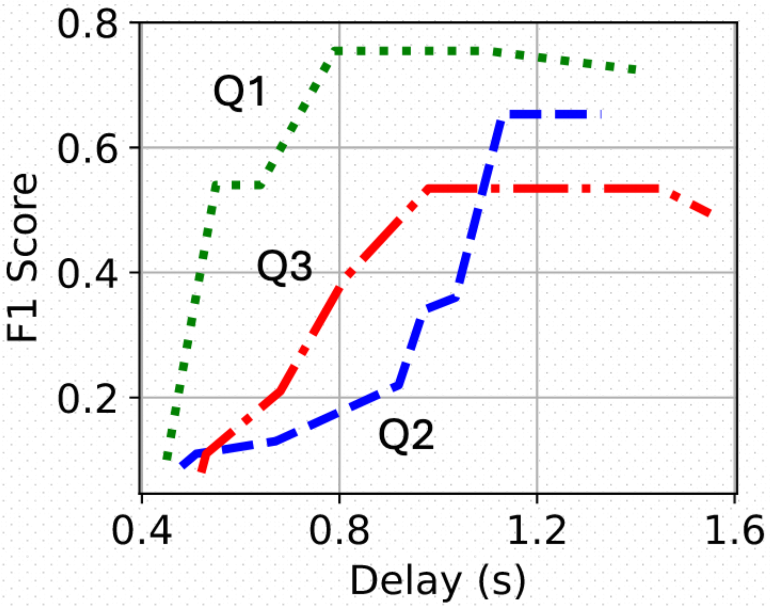
Optimization Opportunities in RAG Systems

❑Per-query RAG configuration

❖How long is each summary if Map-Reduce is selected?



Different RAG synthesis methods



Increase summary length from 1 to 100 with map_reduce

Optimization Opportunities in RAG Systems

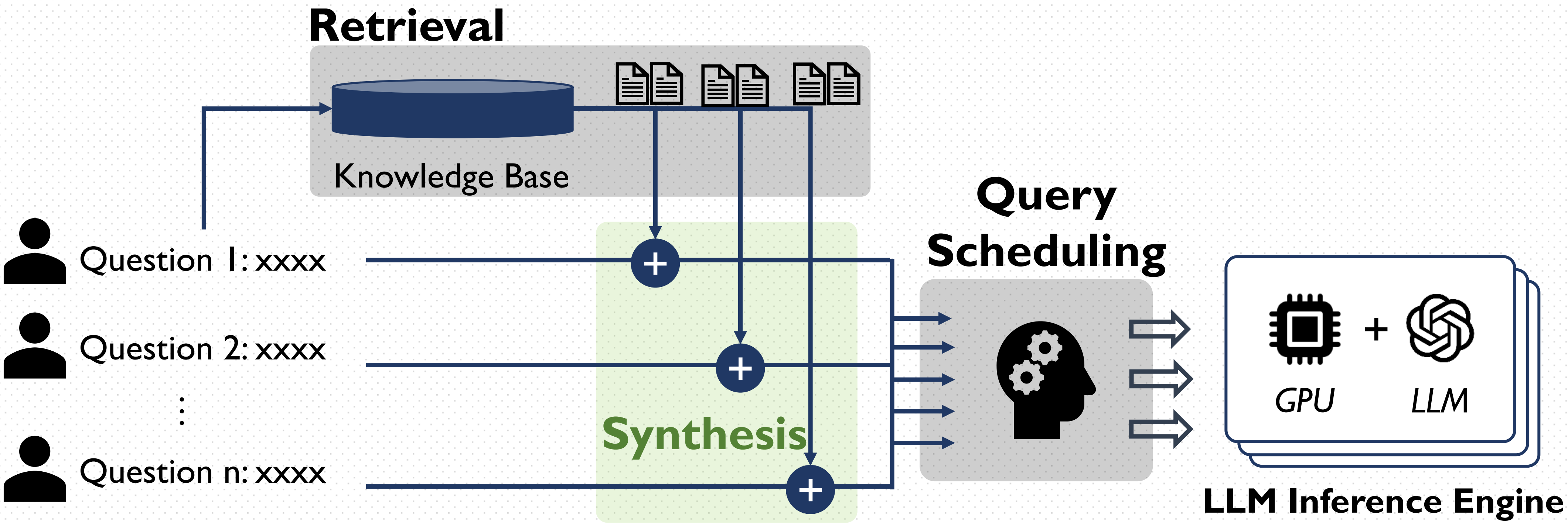
❑Per-query RAG configuration

❖How to synthesize the retrieved chunks?

❖How long is each summary if Map-Reduce is selected?

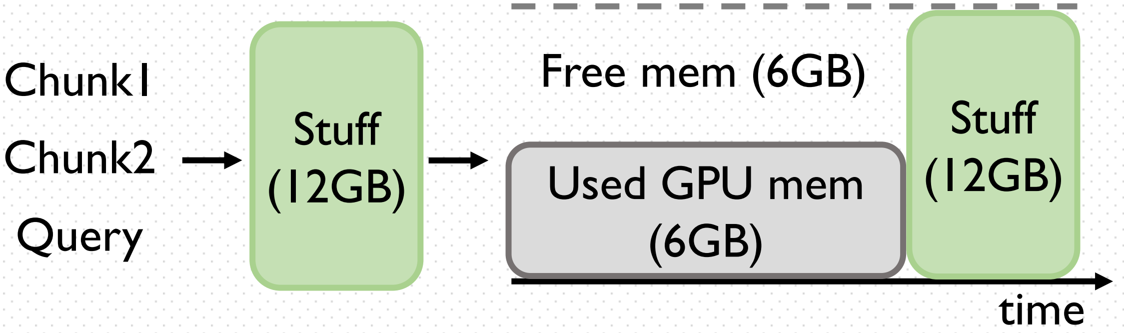
❑Prior work

❖Perhaps none :(

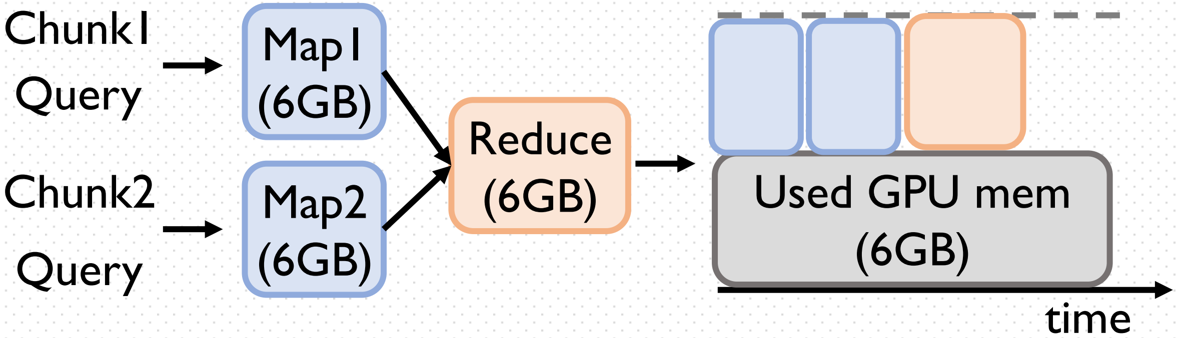


Overlooked Optimization Opportunities

- ❑ Existing work either:
  - ❖ Selects a static config then optimize scheduling
  - ❖ Tunes individual config only
- ❑ Multiple configuration should be tuned together to achieve optimal quality-delay tradeoffs
- ❑ The RAG configuration should be tuned jointly with scheduling



Stuff may be slower when GPU memory is limited



MapReduce achieves faster response time

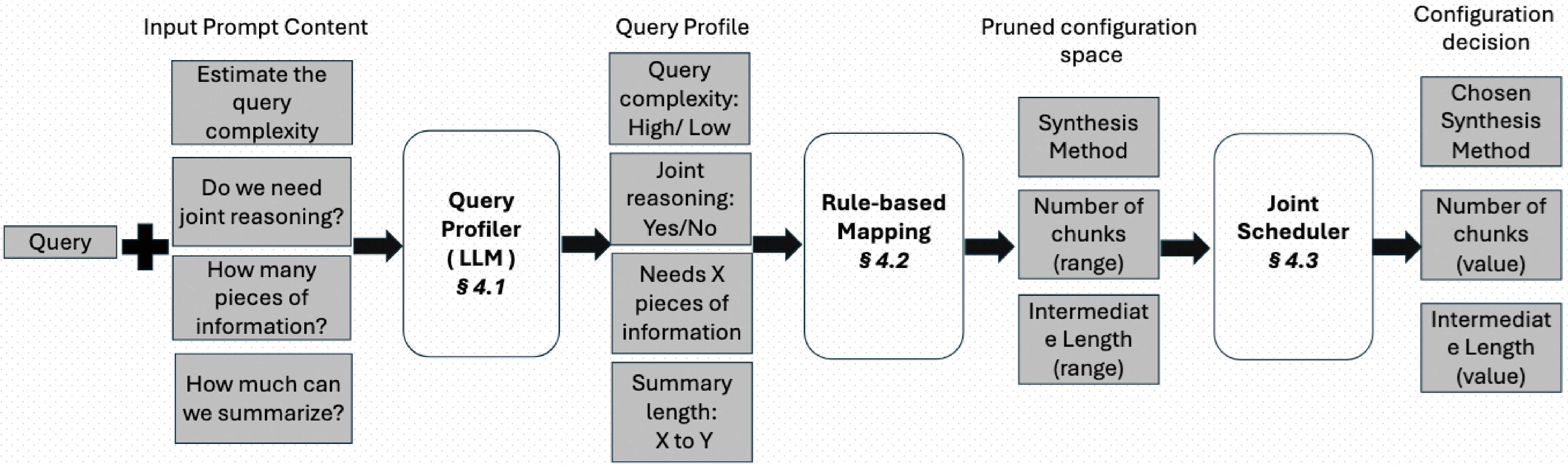
The Challenge: A Combinatorial Explosion

- Just for Map-Reduce, the options explode:
  - Tuning 30 values for num_chunks and 50 values for intermediate_length
  - Leads to ~1,500 configurations per query

- Exhaustive search is infeasible
- How to efficiently find a small set of "good enough" configurations?

The METIS Workflow

1. Estimate each query's profile using LLM
2. Map profile result to a pruned configuration space
3. Select the optimal configuration that fits the current batch



METIS workflow

Query Profiler

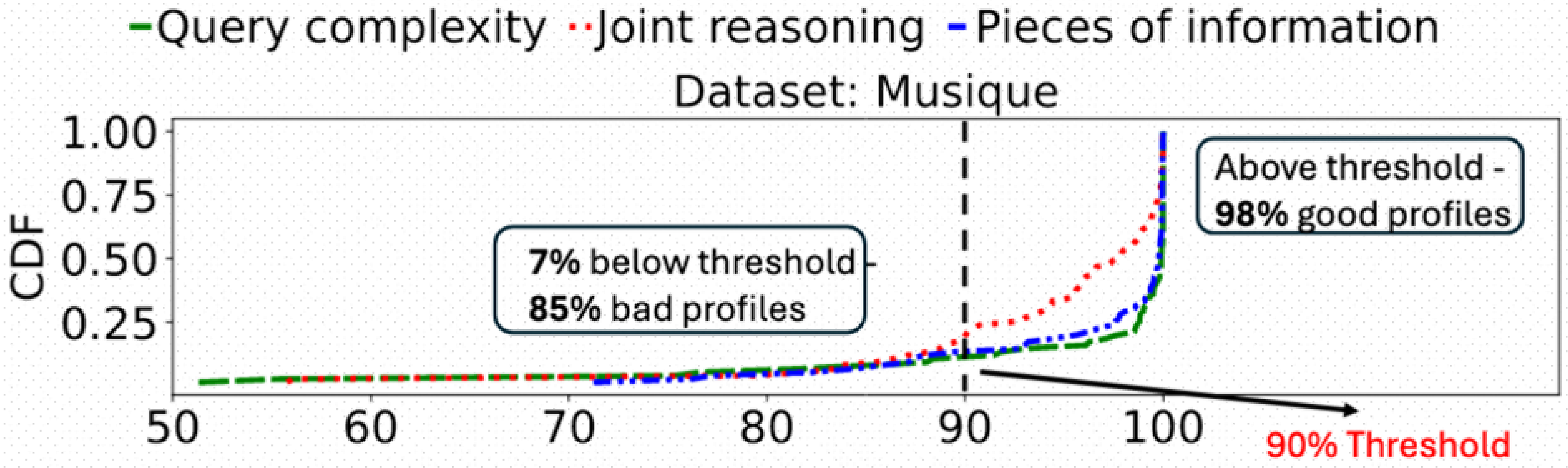
- Four dimensions to profile:
  1. Query complexity
  2. Joint reasoning requirement
  3. Pieces of information required
  4. The length of the summarization
- The metadata of the database is also provided
- Profiler is a larger LLM than the serving LLM, but the cost is cheap
  - Only query and metadata are provided

```
f"""
For the given query = {get.query()}: Analyse
    the language and internal structure of
   the query and provide the following
   information :

     1. Does it needs joint reasoning across
        multiple documents or not.
     2. Provide a complexity profile for the
        query:
        Complexity: High/Low \n \
        Joint Reasoning needed: Yes/No \n "
     3. Does this query need input chunks to
        be summarized and if yes, provide a
        range in words for the summarized
        chunks.
     4. How many pieces of information is
        needed to answer the query?

     database_metadata = {get.metadata()}
     chunk_size = {get.chunk_size()}

Estimate the query profile along with the
   database_metadata and chunk_size to
   provide the output.

"""
```

Prompt for METIS' profiler

Is The Quality Profile Reliable?

- ❑Use a confidence score threshold (90%) to decide
  - ❖If confidence > 90%: accept the generated profile
  - ❖Else: fallback to the space of recent 10 queries



Profile quality below/above the confidence score thredshold

How to Improve The Profiler Over Time?

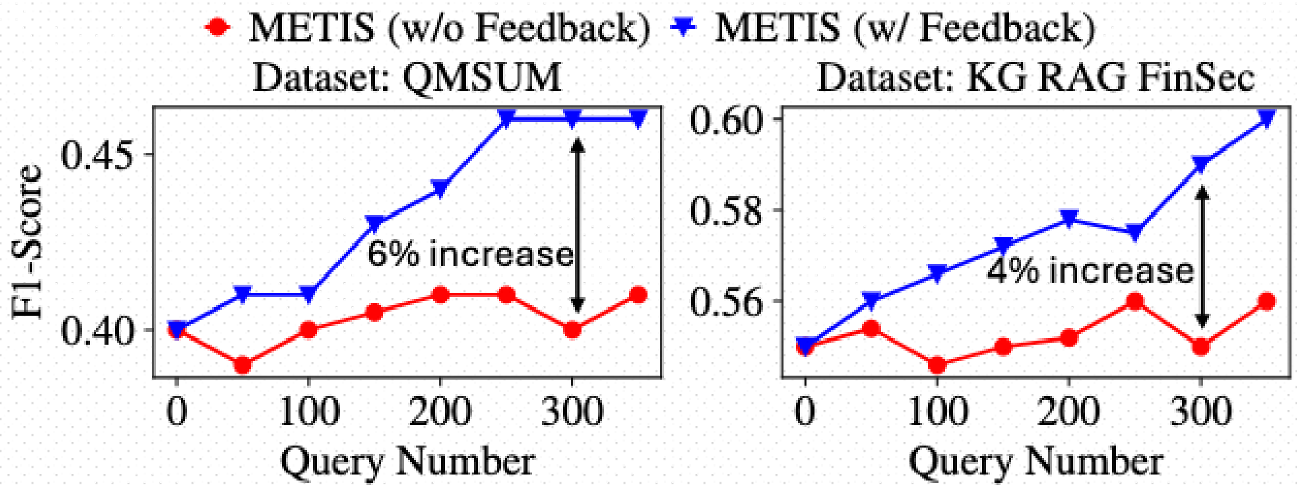
❑The feedback mechanism (executed periodically)

1. Generate a golden answer with the most expensive configuration
2. Run profiling with query & answer both provided
3. In-context learning with query/answer/profile

❑Cost control

❖Low Frequency: 1/30

❖Limited History: last 4



Improvement for METIS using feedback

Rule-based Mapping

- Translate the query profile into an actionable configuration space

- The result: a pruned space of high-quality configurations

Algorithm 1: Rule based mapping algorithm

Input: *Query complexity, Joint reasoning required*
Input: *Pieces of information* , *Summarization length range*
Result: synthesis_method, num_chunks, intermediate_length

```
if Joint reasoning required == "no" then
    synthesis_method = map_rerank
else
    if Query complexity == "low" then
        synthesis_method = stuff
    else
        synthesis_method = stuff, map_reduce
num_chunks = [Pieces of information , 3× Pieces of information]
intermediate_length_range = Summarization length range
```

Joint Scheduler

- ❑Select the best-fit configuration
  - ❖Given a pruned range of good configurations
  - ❖Choose the best configuration which fits in memory
  - ❖Without considering quality anymore

- ❑In the case that none of the configurations fits
  - ❖Fall back to a cheaper configuration
    - ➢ MapRerank with as many chunks
    - ➢ Or Stuff with as many chunks

Evaluation Setup

- ❑ Inference model
  - ❖ Mistral-7B-v3 with 1 A40
  - ❖ Llama3.1-70B with 2 A40
- ❑ Profiling model
  - ❖ GPT-4o (OpenAI's Chat Completion API)
  - ❖ LLama-3.1-70B (HuggingaceAPI)
- ❑ Datasets
  - ❖ Squad
  - ❖ Musique
  - ❖ KG RAG FinSec
  - ❖ QMSUM
- ❑ Metric
  - ❖ F1-score
  - ❖ Delay
  - ❖ Dollar

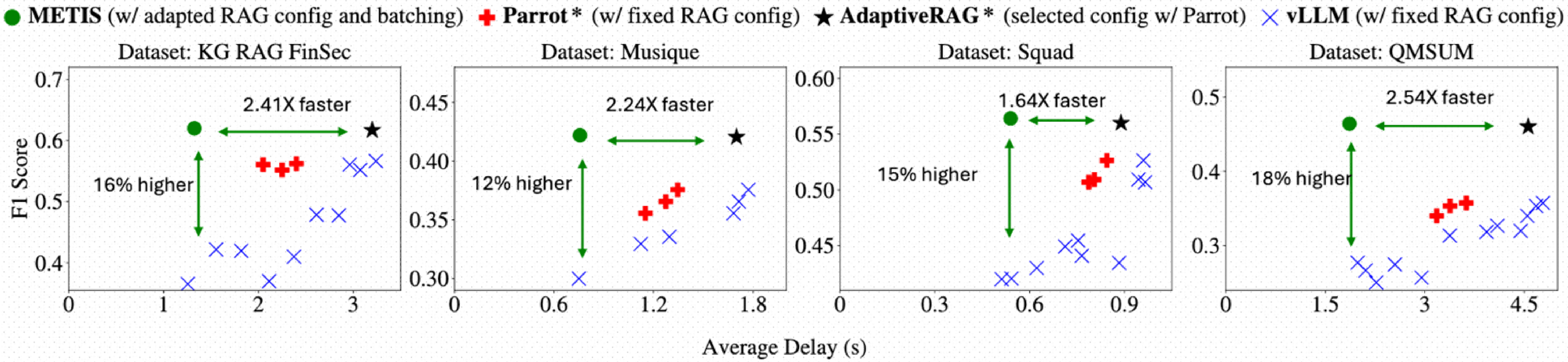
Evaluation Setup

- Baselines
  - vLLM (SOSP '23)
    - A highly-optimized inference engine using a static RAG configuration
  - Parrot* (OSDI '24)
    - One of SOTA LLM schedulers (with static configuration) using semantic variable
  - AdaptiveRAG* (NAACL '24):
    - Always picks the best possible configuration for quality
    - But is system-unaware

Delay and Throughput Improvement

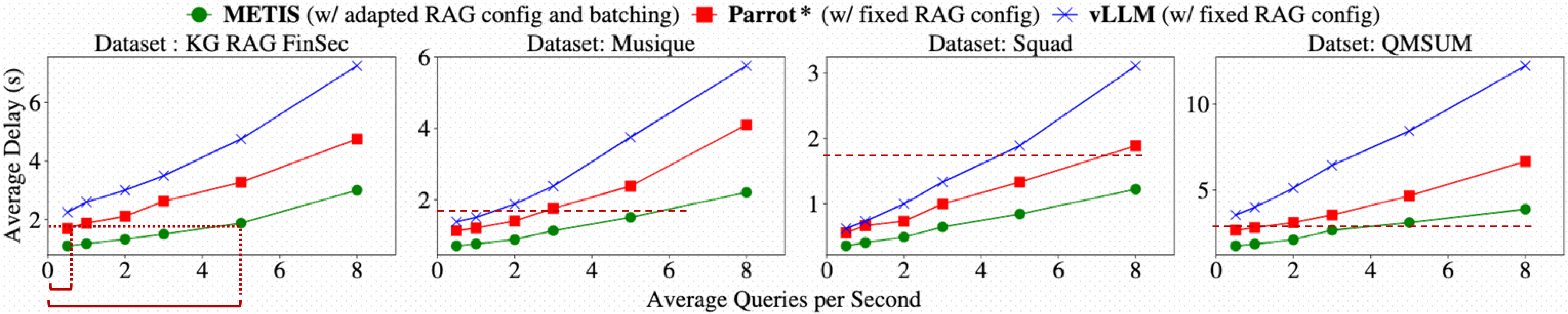
❑Lower delay without sacrificing generation quality

- ❖1.64-2.54x Lower delay
- ❖12-18% Higher F1-score



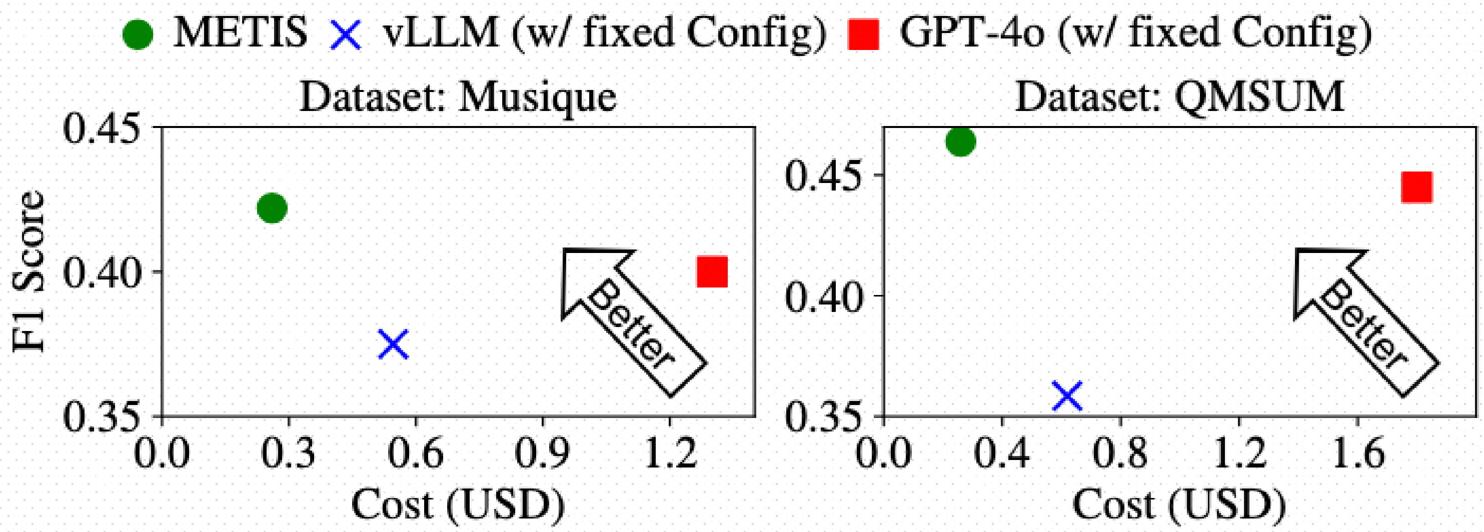
Delay and Throughput Improvement

- ❑Higher throughput at lower delay
  - ❖1.8-4.5x Higher throughput (at 1.8 seconds)
  - ❖With higher quality



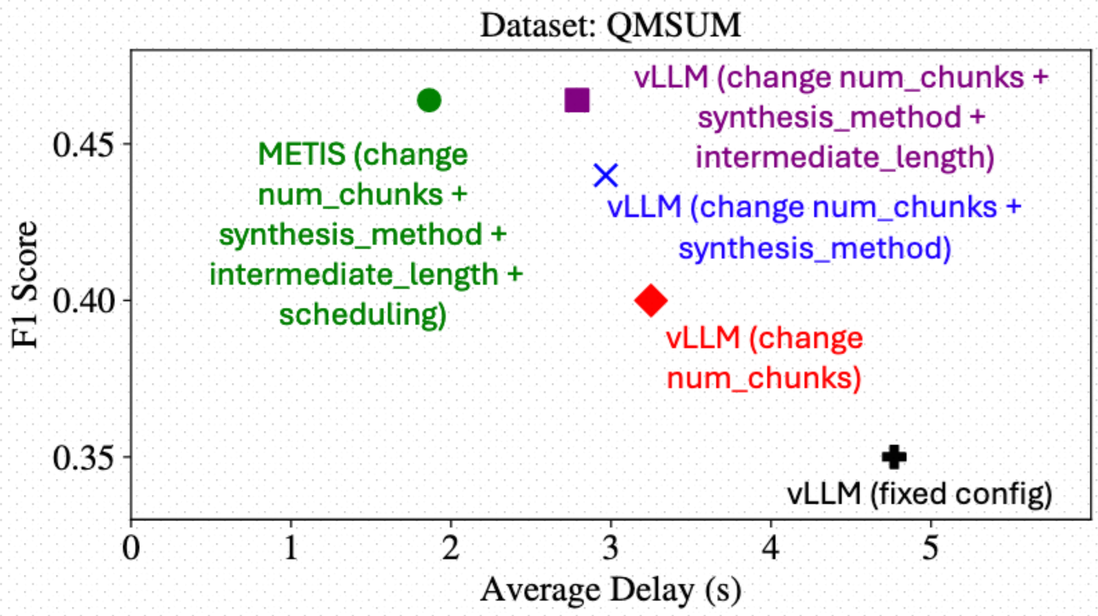
Cost Saving

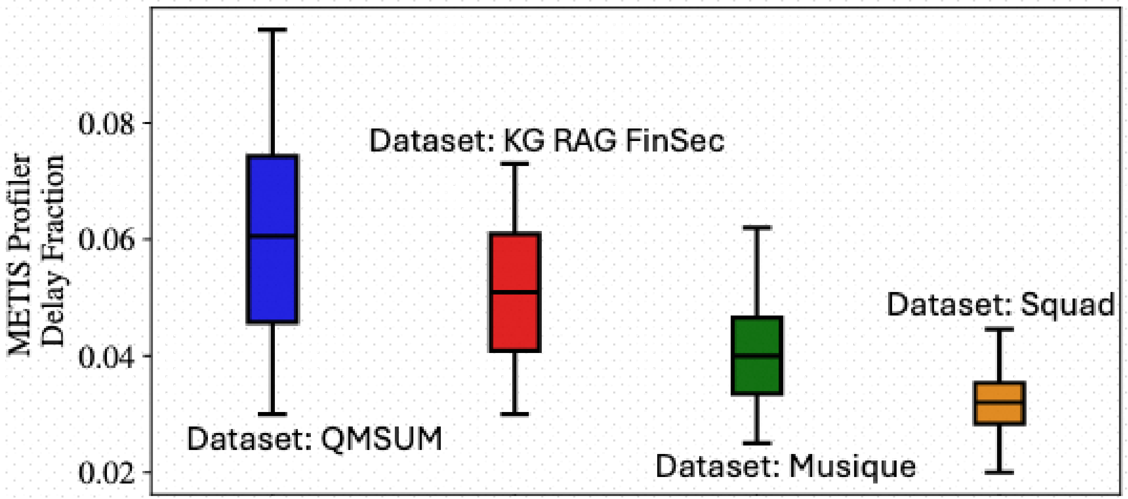
- ❑Significant lower dollar cost and higher F1-score
  - ❖METIS outperforms GPT-4o
    - ➢6.8x Lower cost
    - ➢Higher F1-score



Breakdown Analysis

- ❑Enabling more knobs unlocks better trade-offs
- ❑Profiler delay is at most 1/10 of end-to-end response delay

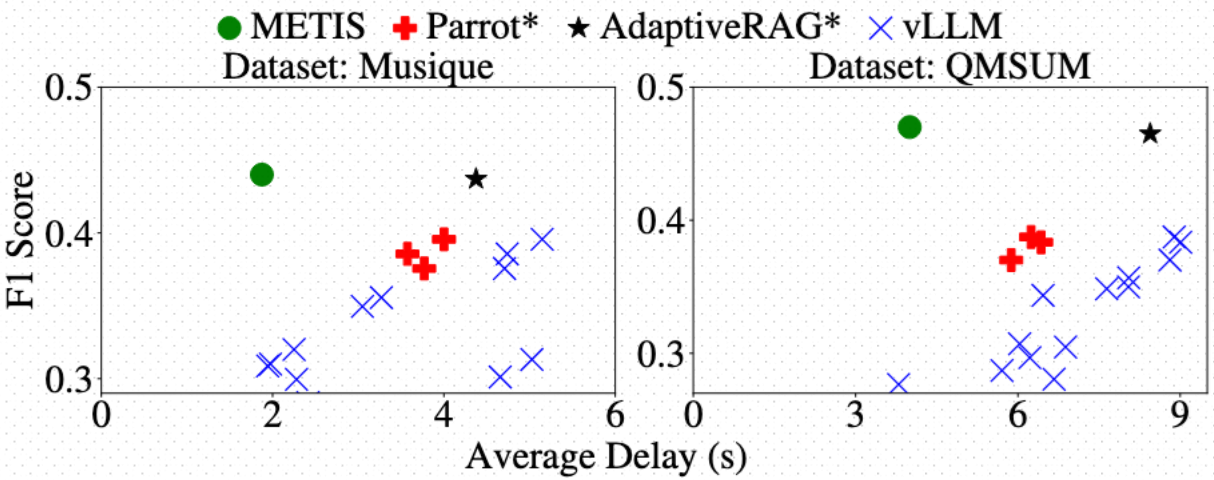


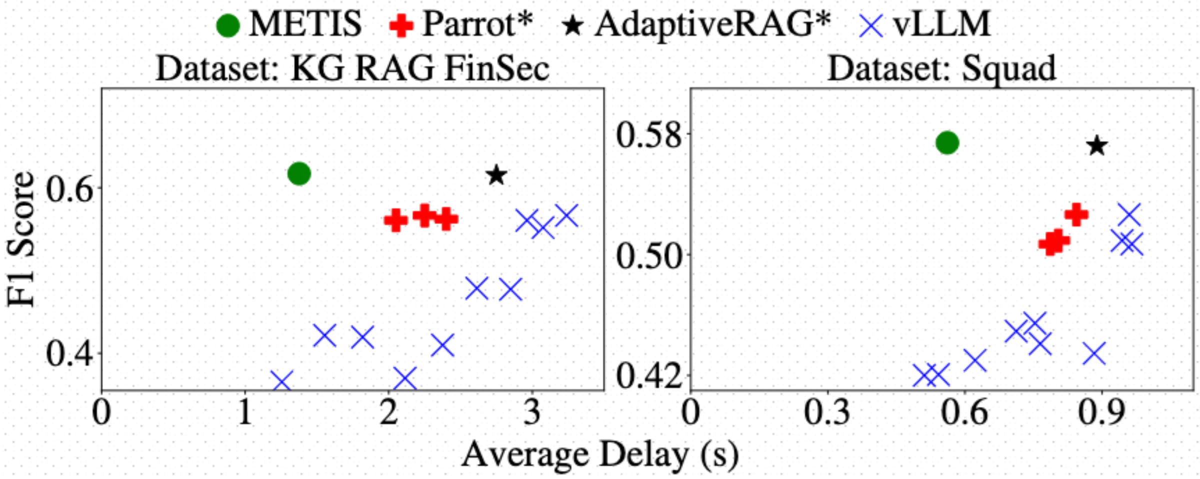


Sensitivity Analysis

- ❑METIS's advantages persist with larger inference model
  - ❖Mistral-7B-v3 ⇨ Llama3.1-70B

- ❑Performance gains remain even with a smaller, low-cost profiler
  - ❖GPT-4o ⇨ LLama-3.1-70B





Conclusion

❑Highlights

- ❖A simple and efficient RAG optimization framework
- ❖Extensive and insightful experimentation
- ❖Clear and well-crafted story

❑Potential problems

- ❖The risk of quality collapse under high load