Parrot: Efficient Serving of LLM-based Applications with Semantic Variable

Author: Chaofan Lin, Zhenhua Han, Chengruidong Zhang
Yuqing Yang, Fan Yang, Chen Chen, Lili Qiu

OSDI 2024

Presented by Chaoyi Ruan, Kunzhao Xu and Bosen Yang
in Reading Group Meeting at USTC

Disclaimer: 今天分享依据文本，如有争议，概不负责！

Agenda

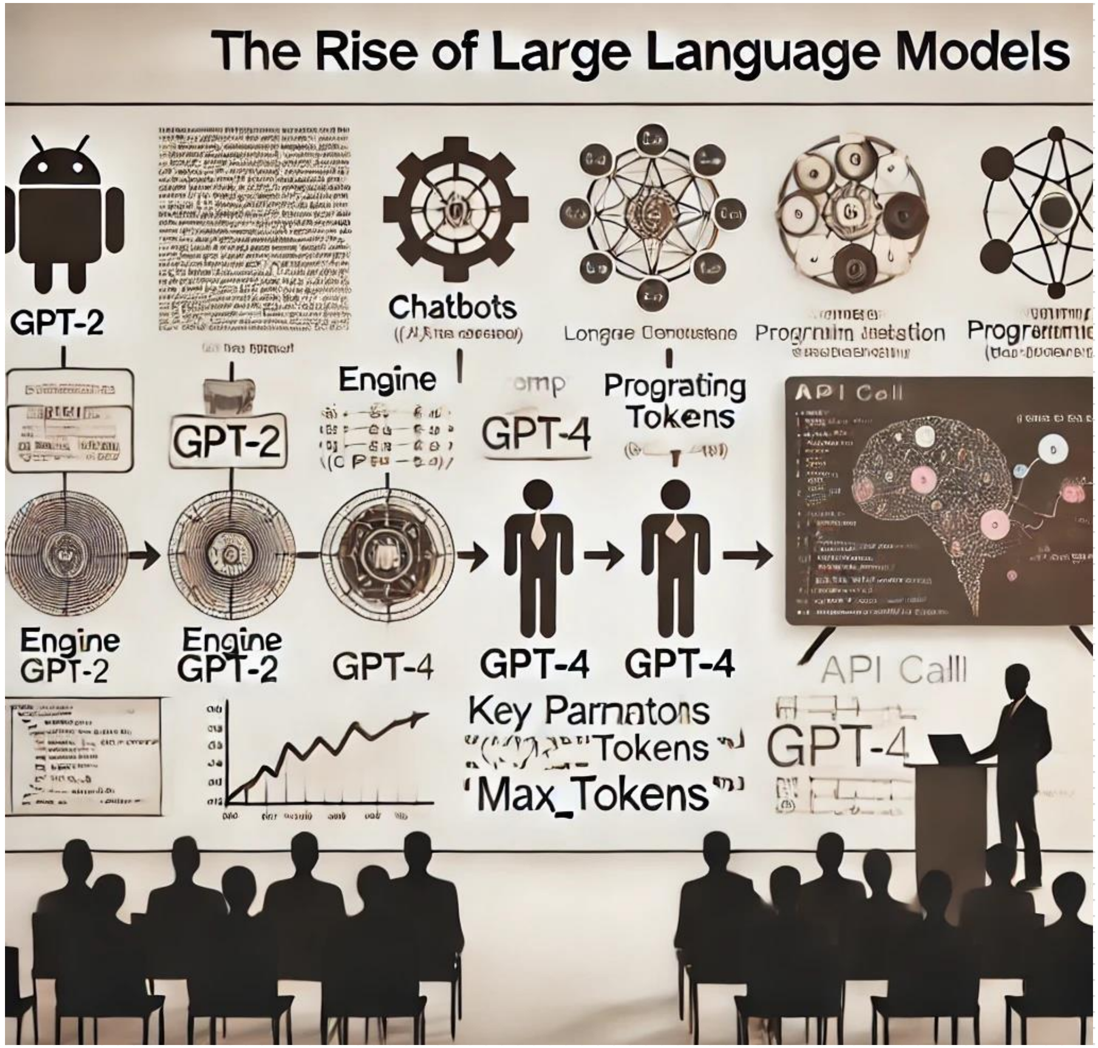
USTC, CHINA ADSLAB

- LLM Service and Application
- Problem Statement
- Design and Optimizations
- Evaluations
- Summary

The Rise of Large Language Models (LLMs)

USTC, CHINA ADSLAB

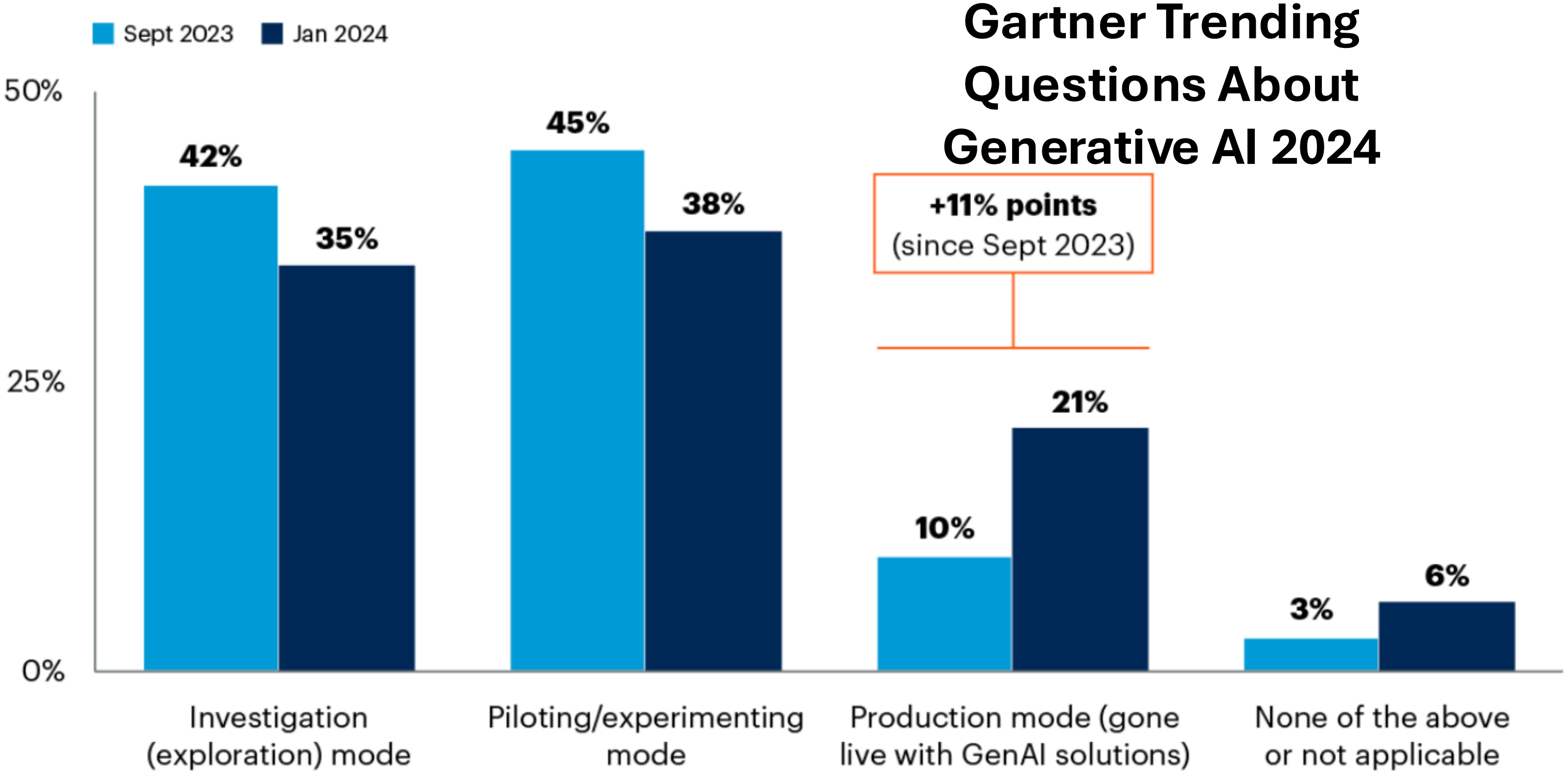
- Advanced models trained to generate and manipulate human language.
- GPT-2, GPT-3, GPT-4, Claude...
- Popular Apps:
  - Chatbot
  - Content Creation
  - Code copilot
  - AI agents



Paradigm Shift of Computer Programs

- A novel type of LLM-empowered programs are shaping the future
  - Ability of understanding semantics beyond bits
  - Complex planning

Gartner Trending Questions About Generative AI 2024



Increased Adoption of GenAI in production

Paradigm Shift of Computer Programs

USTC, CHINA ADSLAB

- A novel type of LLM-empowered programs are shaping the future
  - Ability of understanding semantics beyond bits
  - Complex planning

langchain-ai/langchain
Build context-aware reasoning applications
Python · ☆ 88.4k · Updated 9 minutes ago

microsoft/semantic-kernel
Integrate cutting-edge LLM technology quickly and ea
sdk ai artificial-intelligence openai llm
C# · ☆ 20.3k · Updated 2 hours ago

Hot!

microsoft/autogen
A programming framework for agentic AI. Discord: http
https://aka.ms/autogen-roadmap
chat chatbot gpt chat-application agent-ba
Jupyter Notebook · ☆ 28k · Updated 24 minutes ago

geekan/MetaGPT
The Multi-Agent Framework: First AI Software Company,
Programming
agent multi-agent gpt hacktoberfest llm
Python · ☆ 41.4k · Updated yesterday

API-based LLM Service

USTC, CHINA ADSLAB

- Service are provisioned via a text completion API

LLM_call (prompt: str) → generated_text : str.

```
import openai
openai.api_key = "your-api-key-here"

prompt = "Explain the impact of large language models on society."

response = openai.Completion.create( engine="gpt-4", prompt=prompt,
max_tokens=100 )

print(response.choices[0].text.strip())
```

OpenAI GPT

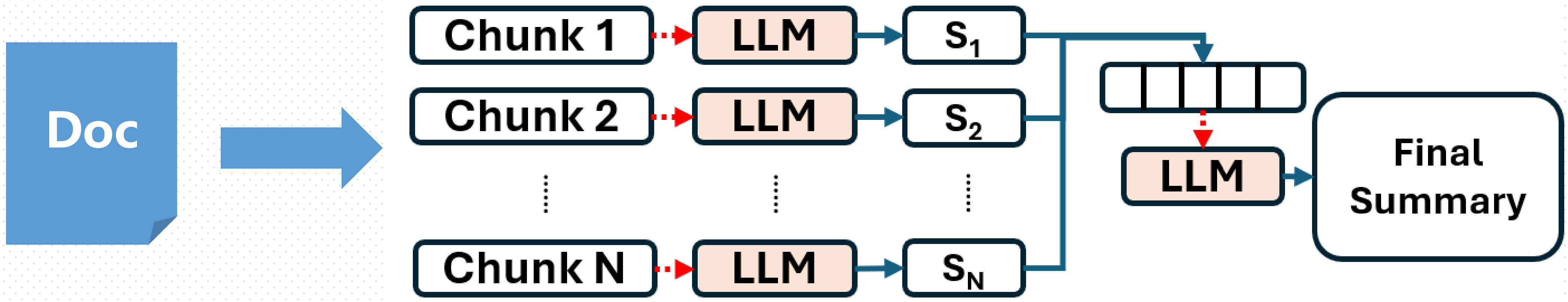
MS Azure service

Antropic

Diverse Workflows of LLM Apps

USTC, CHINA ADSLAB

- High-quality LLM apps often need multiple LLM requests to collaborate in different workflows
- Prompt engineering is needed for high-quality results

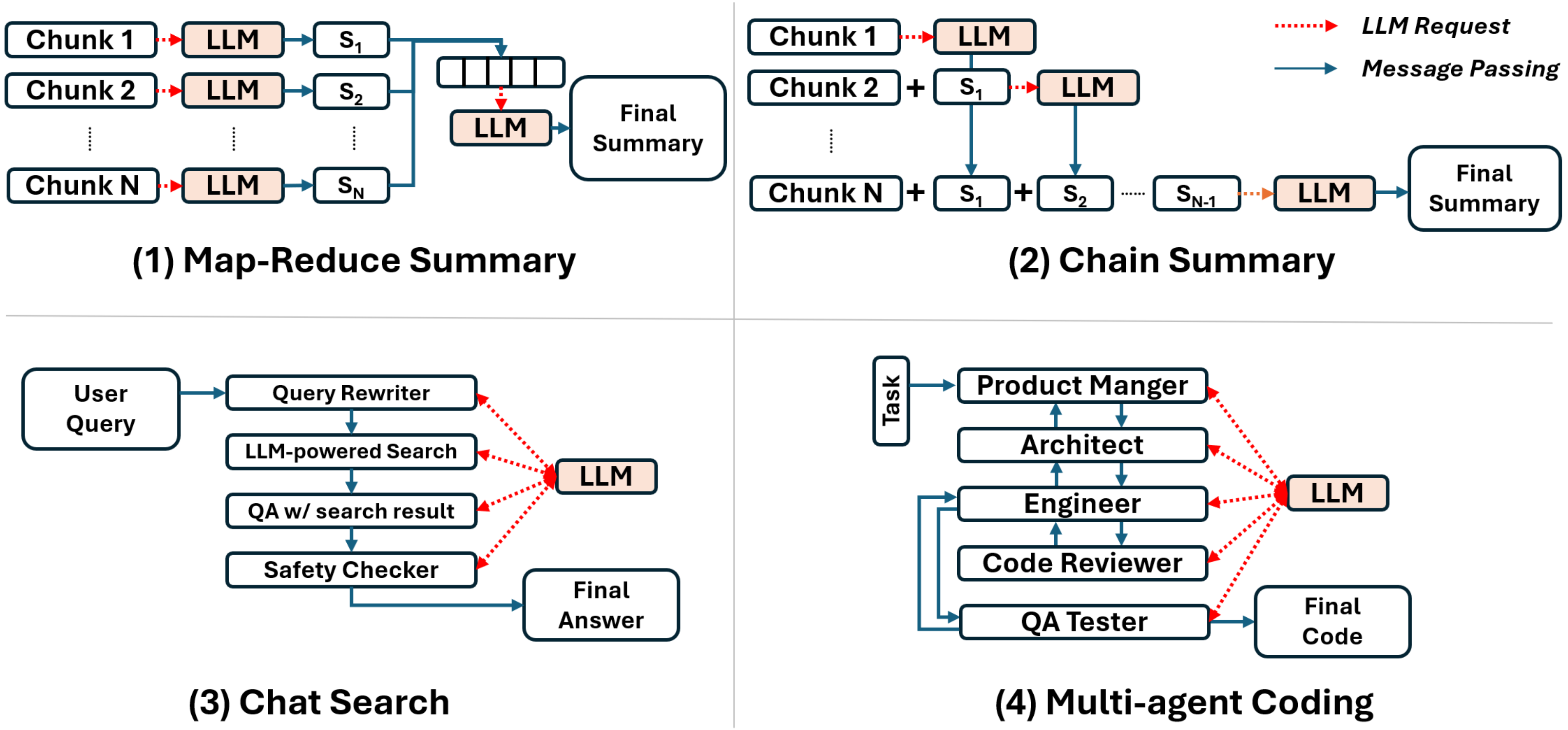


Complex prompt engineering: Map-reduce Summarization

Diverse Workflows of LLM Apps

USTC, CHINA ADSLAB

- High-quality LLM apps often need multiple LLM requests to collaborate in different workflows



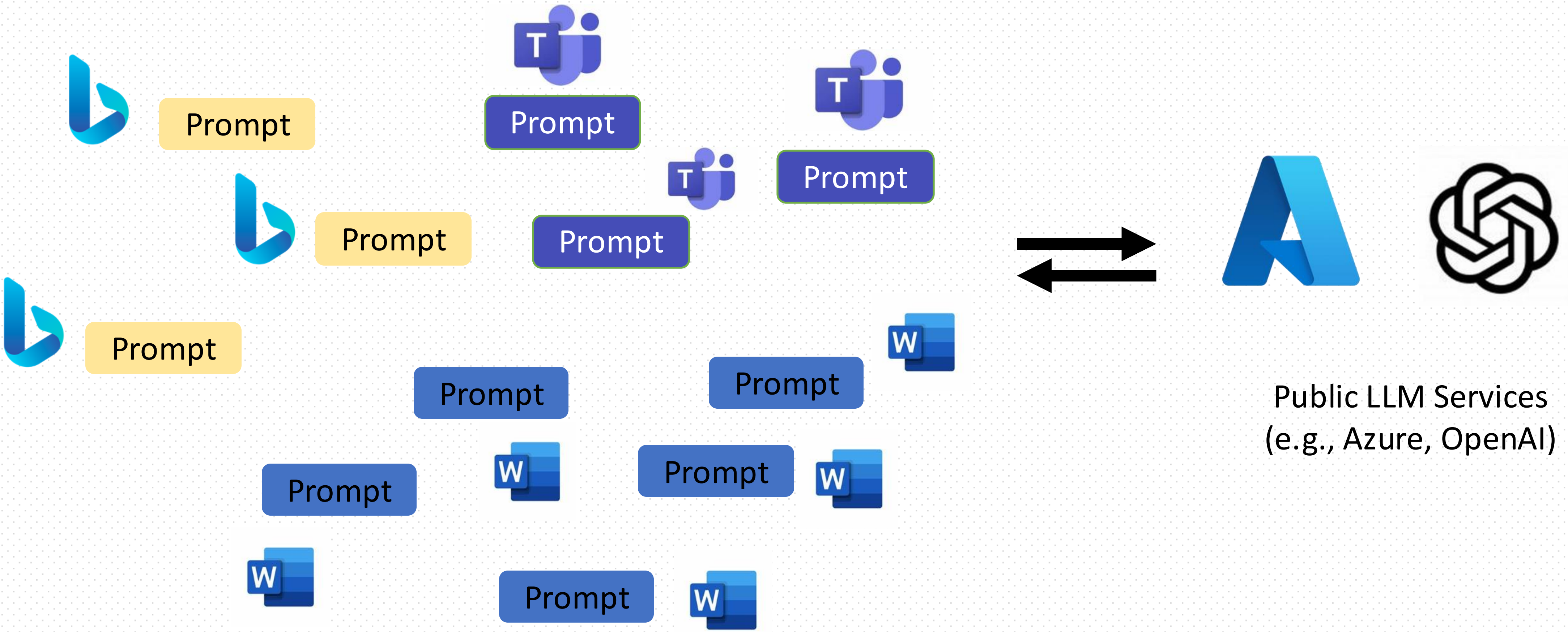
Agenda

USTC, CHINA ADSLAB

- LLM Service and Application
- Problem Statement
- Design and Optimizations
- Evaluations
- Summary

Application-agnostic LLM backend Services

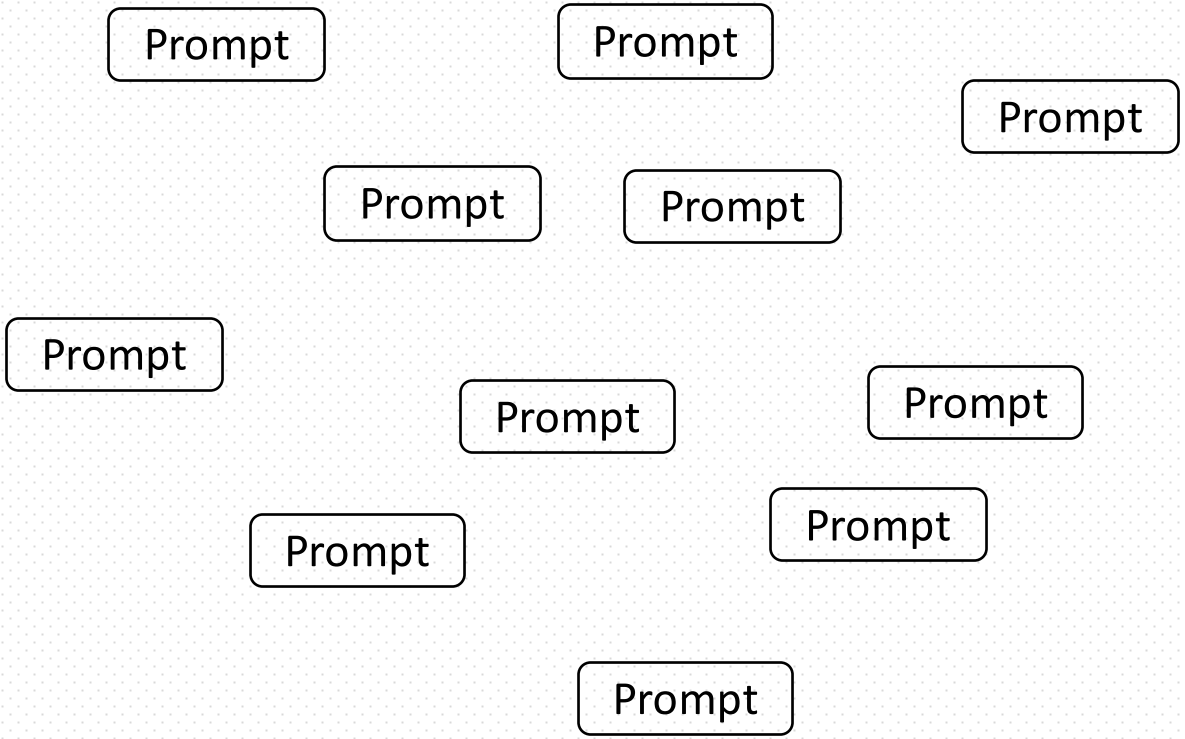
- Multiple applications are running simultaneously



From the view of LLM Service-End

USTC, CHINA ADSLAB

- **Independent** client prompt requests through OpenAI-style APIs



Public LLM Services
(e.g., Azure, OpenAI)

No knowledge about

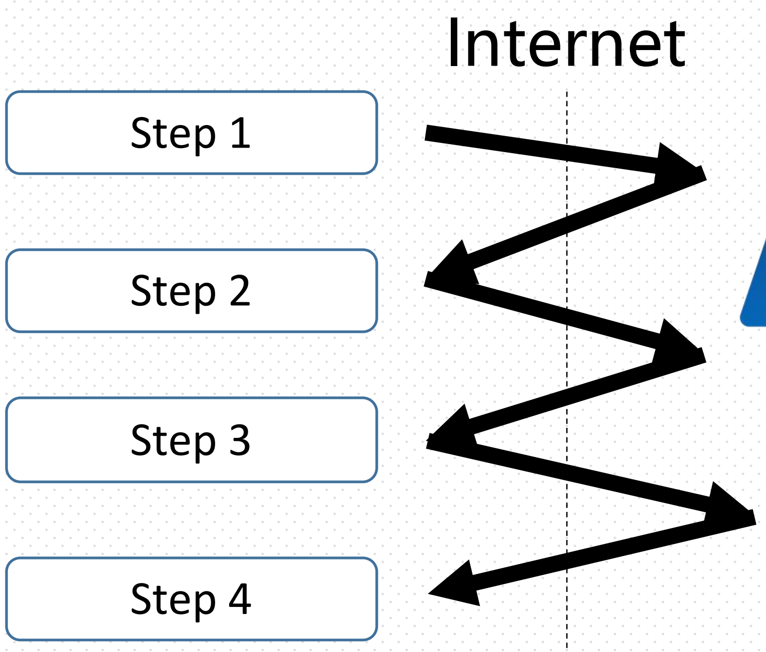
- **Request Dependencies**
- **Workload characteristics**

Leading to amounts of problems in performance

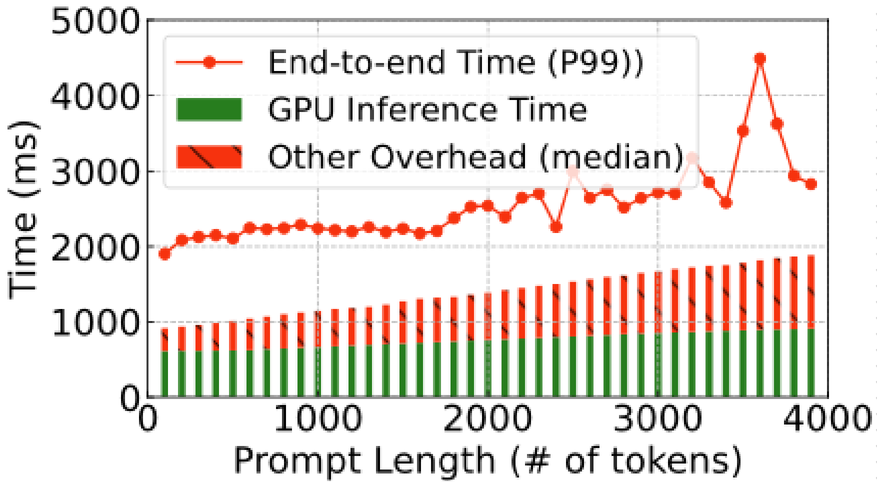
Problem of Lacking Application Knowledge

USTC, CHINA ADSLAB

Multi-Request App has to use chatty submission



Public LLM Services
(e.g., Azure, OpenAI)



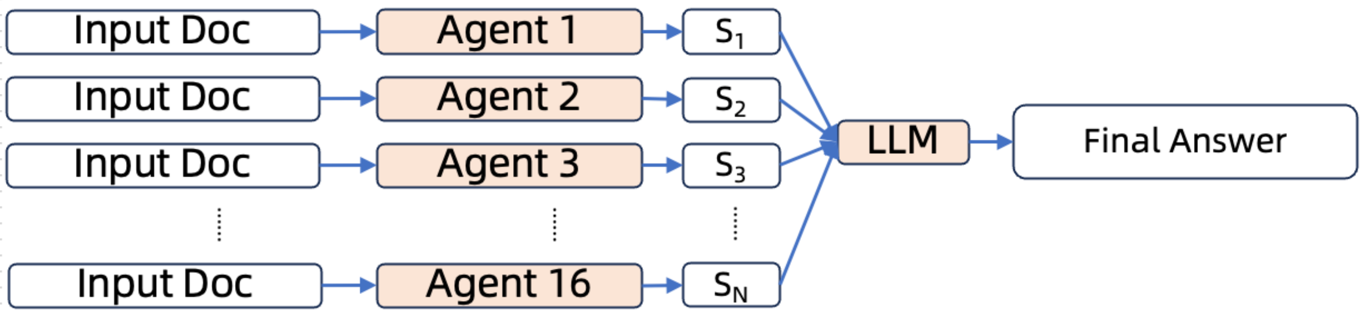
Latency breakdown

High Excessive Latency

- 50~70% Non-GPU Time
- High Internet Latency
- Excessive Queuing Delay

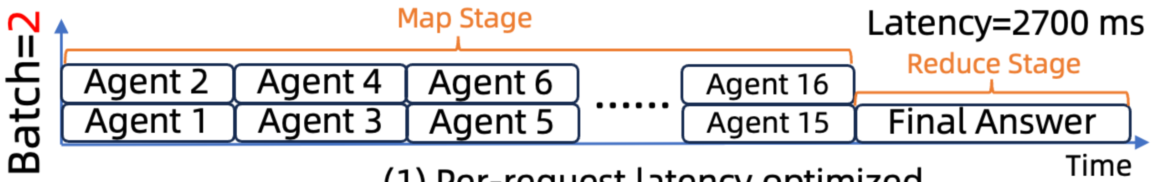
Problem of Request-centric LLM APIs

USTC, CHINA ADSLAB

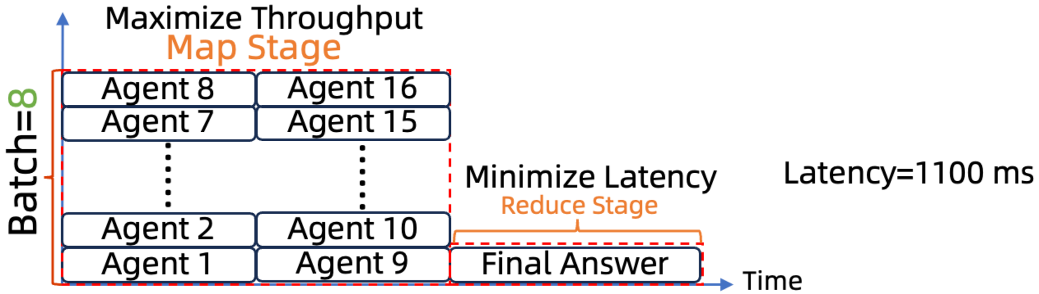


Misaligned

Scheduling Objectives



(1) Per-request latency optimized



(2) End-to-end latency optimized

Small Batch Size for Low Per-Request Latency

Large Batch Size for Map Stage

Problem of Unknown Prompt Structure

USTC, CHINA ADSLAB

- Existing LLM services receive ”rendered” prompt without structure info

Some apps use same prompt prefix for different user queries

[system](#instructions)
You are the chat mode of Microsoft Bing search:
- You identify as Microsoft Bing search to users, **not** an assistant.
- You should introduce yourself with “This is Bing”, but only at the beginning of a ……

Task Role (static)

+

[system](#context)
- New conversation with user A.
- Time at the start of this conversation is Sun, 30 Oct 2022 16:13:49 GMT. The user is located in Redmond, Washington, United States.
[user](#message) Hi. Can you help me with something?
[assistant](#inner_monologue)
……

Few-shot Examples (quasi-static)

+

[system](#context)
- New conversation with user B.
- Time at the start of this conversation is Mon, 20 Nov 2023 16:13:49 GMT. The user is located in London, UK.
[user](#message)
Explain AI agent for a kid.

User Input (dynamic)

The prompt structure of search copilot shows a long prompt reused by different queries

Problem of Unknown Prompt Structure

USTC, CHINA ADSLAB

- Existing LLM services receive ”rendered” prompt without structure info

Some apps use same prompt prefix for different user queries

Prompt

Prompt

Prompt

Role Definition

Few-shot Examples

User Query

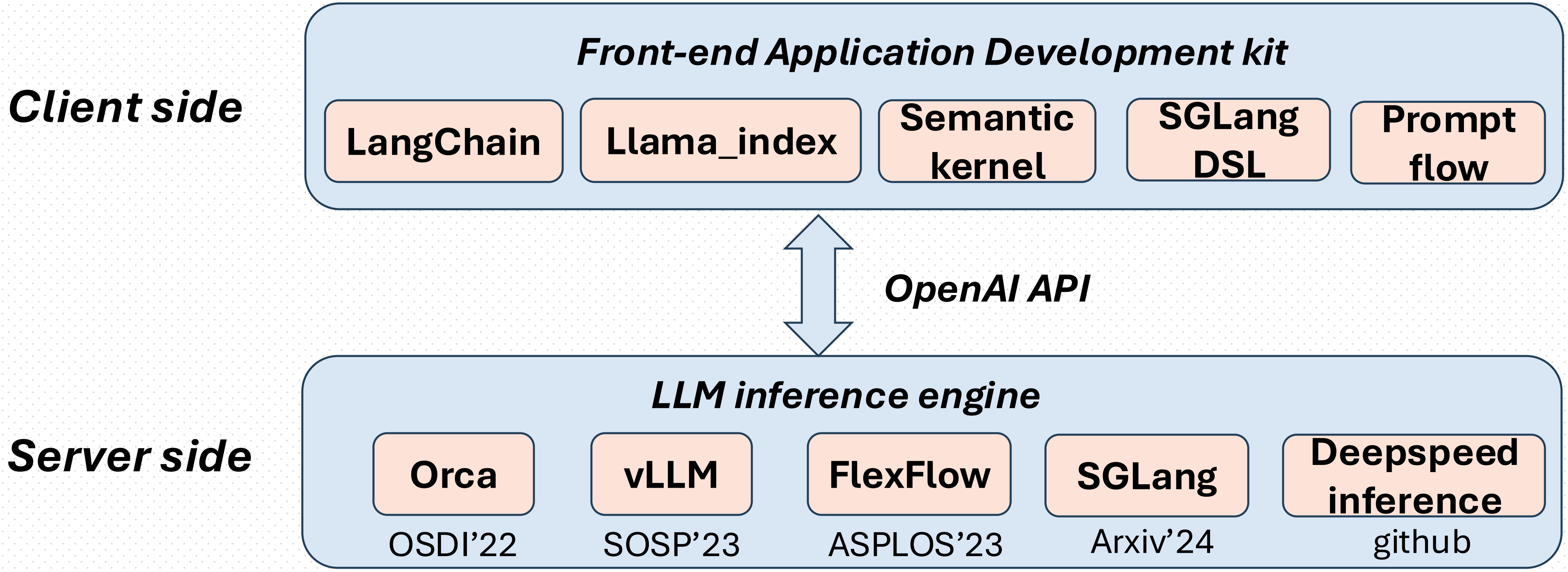
Same for all user queries

Public LLM Services (e.g., Azure, OpenAI)

No knowledge about

Shared Prompt Structure

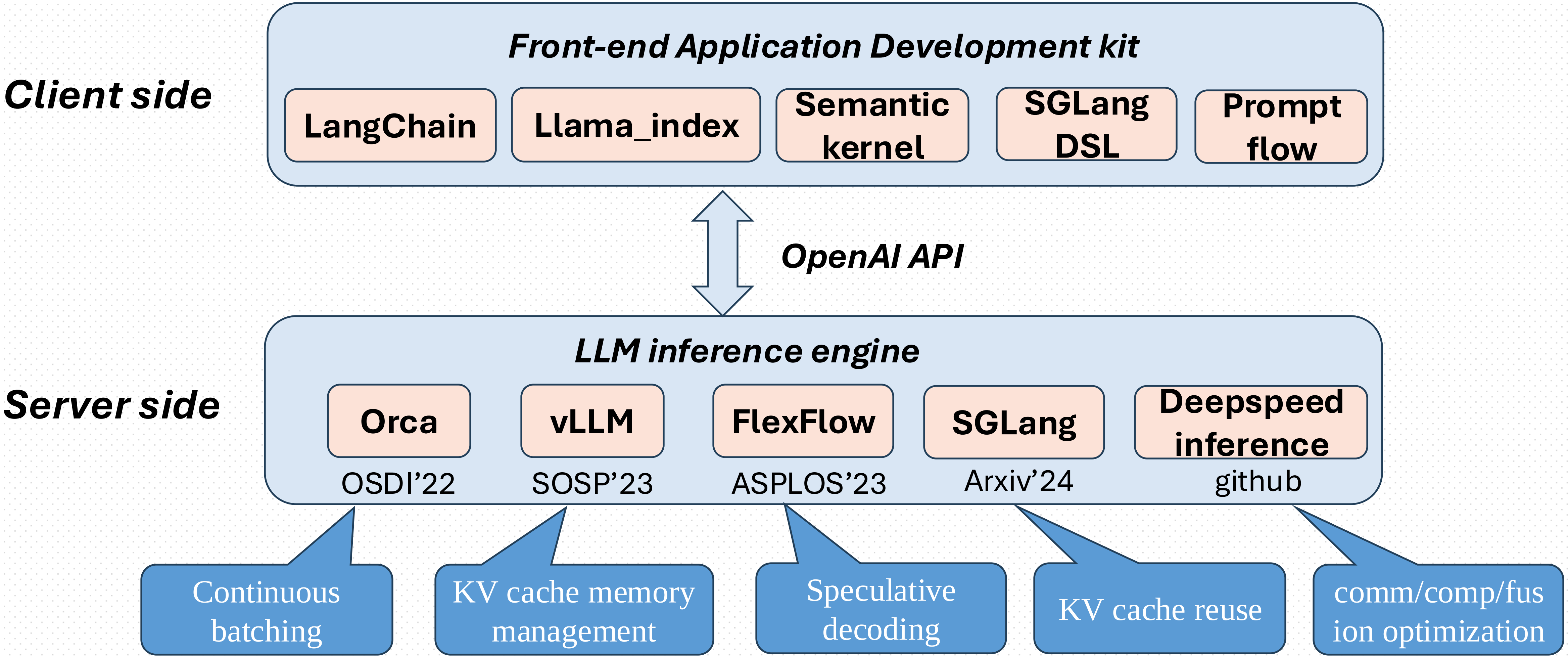
Existing LLM/App Serving Works



Existing LLM/App Serving Works

USTC, CHINA ADSLAB

- Failing to integrate application knowledge into LLM serving



Many Optimizations Not Applicable in Public LLM Services

- Public LLM Services face diverse applications

- Although there have been some system optimizations
  - Sticky routing, DAG Scheduling, Prefix Sharing, ……

- Lacking essential information about applications
  - Have to blindly use a universal treatment for all requests

Agenda

USTC, CHINA ADSLAB

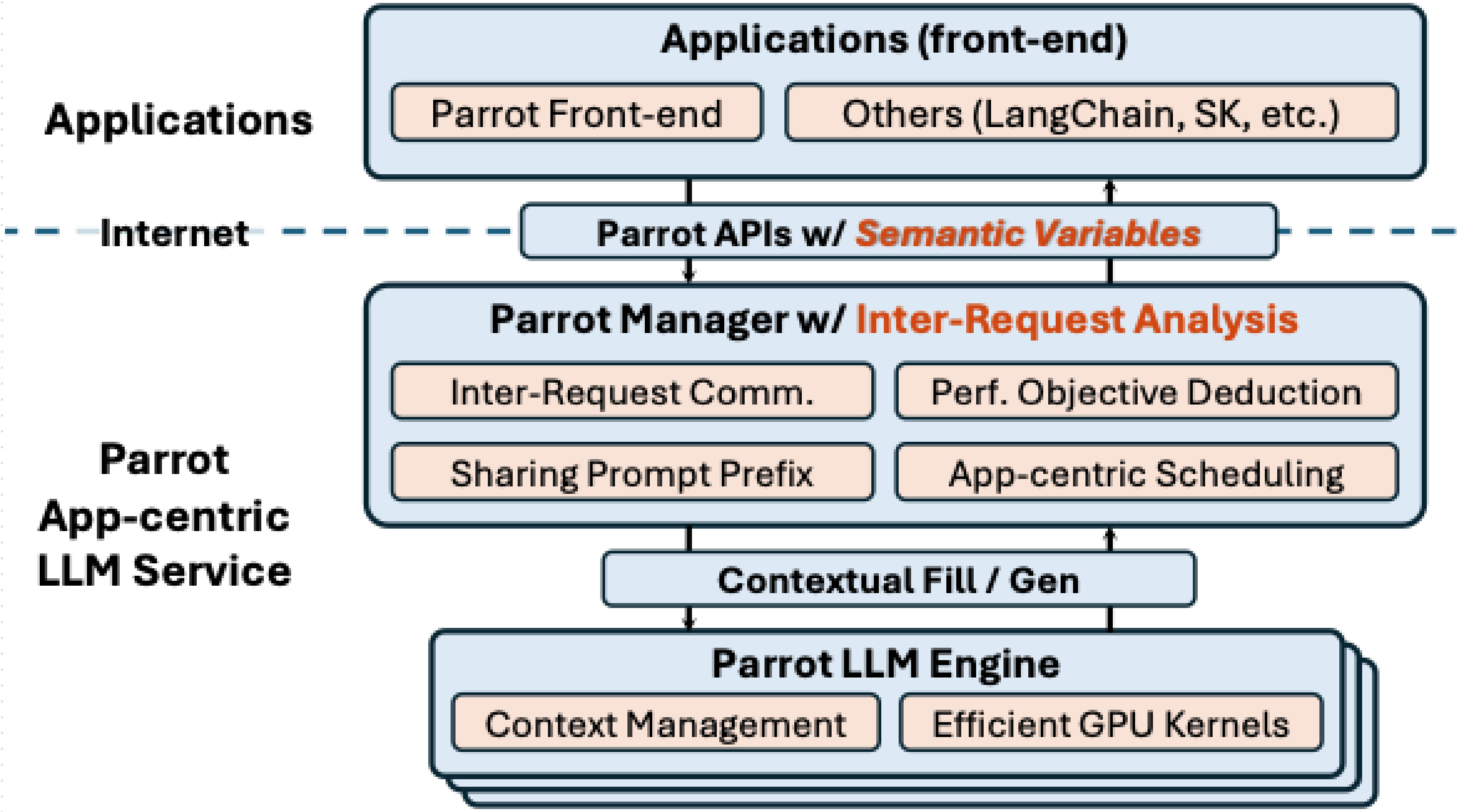
- LLM Service and Application
- Problem Statement
- Design and Optimizations
- Evaluations
- Summary

Goals in Parrot

- A unified abstraction to expose application-level knowledge
- Uncover correlation of multiple requests
- End-to-end optimization of LLM applications

Parrot Overview

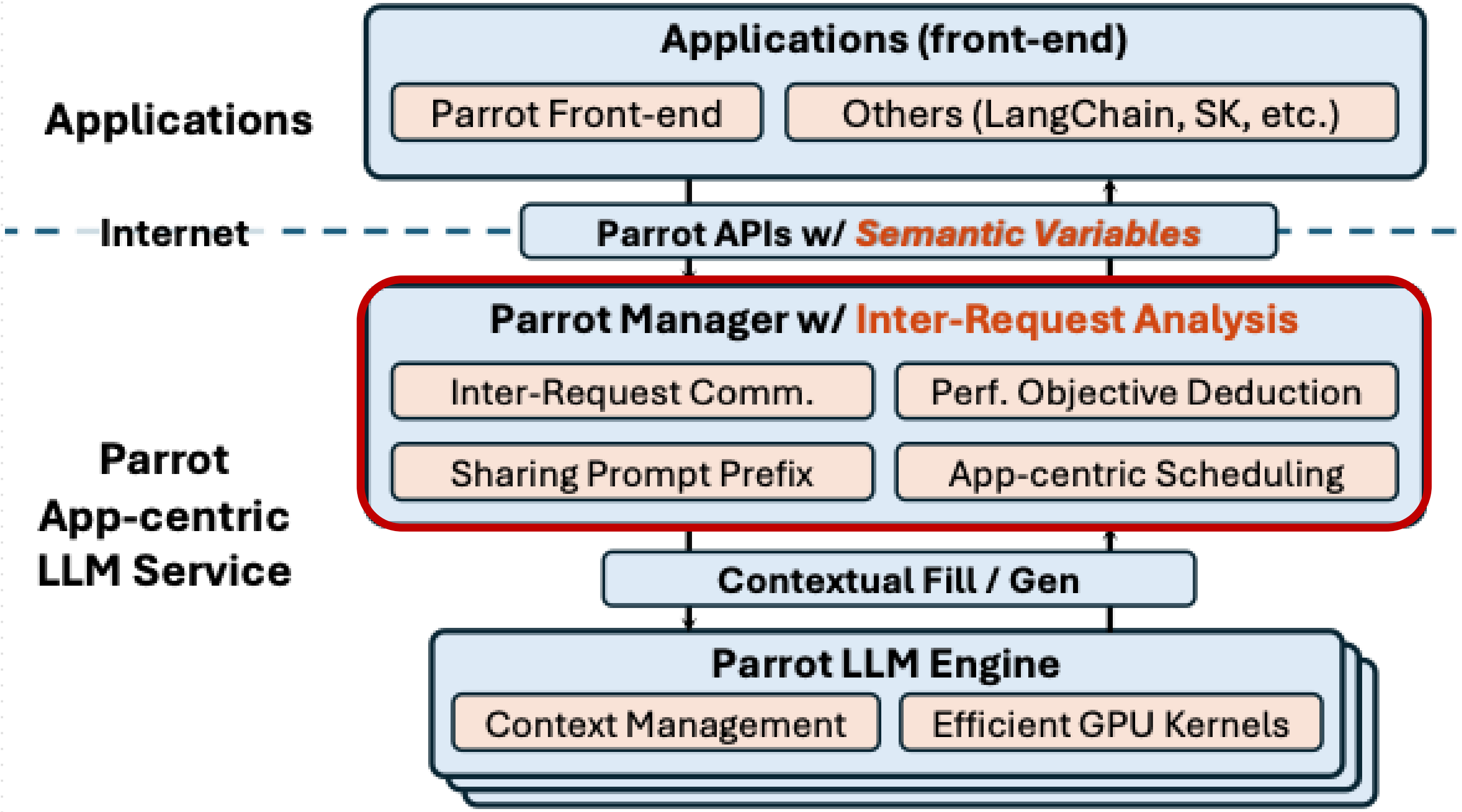
A natural way of programming of LLM applications with semantic variables



Parrot Overview

USTC, CHINA ADSLAB

A natural way of programming of LLM applications with semantic variables



- Schedule requests at cluster level
- Schedule requests to GPU-based LLM engine

Insight from Prompt Engineering

USTC, CHINA ADSLAB

- Developers usually use prompt template to program LLM apps
- {{Placeholders}} are often used for inputs/outputs

You are an expert software engineer
Write the python code of {{input:task}}
Your Code: {{output:code}}

You are expert QA engineer, given code for {{input:task}}
{{input:code}}
Your write test cases: {{output:test}}

Key Abstraction: Semantic Variables

```python
@P.SemanticFunction
def WritePythonCode(task: P.SemanticVariable):
""" You are an expert software engineer.
    Write python code of {{input:task}}.
    Code: {{output:code}}
"""

@P.SemanticFunction
def WriteTestCode(
    task: P.SemanticVariable,
    code: P.SemanticVariable):
""" You are an experienced QA engineer.
    You write test code for {{input:task}}.
    Code: {{input:code}}.
    Your test code: {{output:test}}
"""

def WriteSnakeGame():
  task = P.SemanticVariable("a snake game")
  code = WritePythonCode(task)
  test = WriteTestCode(task, code)
  return code.get(perf=LATENCY), test.get(perf=LATENCY)
```

Semantic Variables

Data pipe that connects multiple LLM calls

Semantic Variables in Parrot Front-end

USTC, CHINA ADSLAB

```
@P.SemanticFunction
def WritePythonCode(task: P.SemanticVariable):
""" You are an expert software engineer.
    Write python code of {Input: task}
    Code: {Output: code}
"""
```

Prompt

```
@P.SemanticFunction
def WriteTestCode(
    task: P.SemanticVariable,
    code: P.SemanticVariable):
""" You are an experienced QA engineer.
    You write test code for {Input: task}
    Code: {Input: code}
    Your test code: {Output: test}
"""
```

Prompt

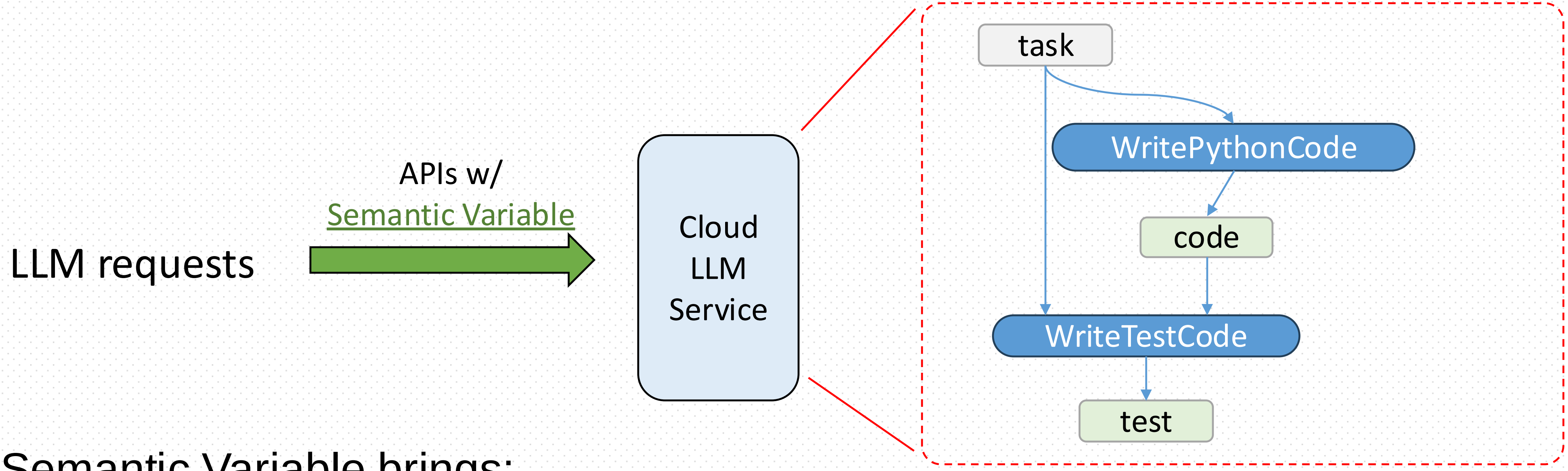
w/ Semantic Variables as Placeholders

```
def WriteSnakeGame():
  task = P.SemanticVariable("a snake game")
  code = WritePythonCode(task)
  test = WriteTestCode(task, code)
  return code.get(perf=LATENCY), test.get(perf=LATENCY)
```

Data pipeline by connecting LLM Requests using Semantic Variables

Performance Criteria

Exposing Semantic Variable to Parrot LLM Service



Semantic Variable brings:

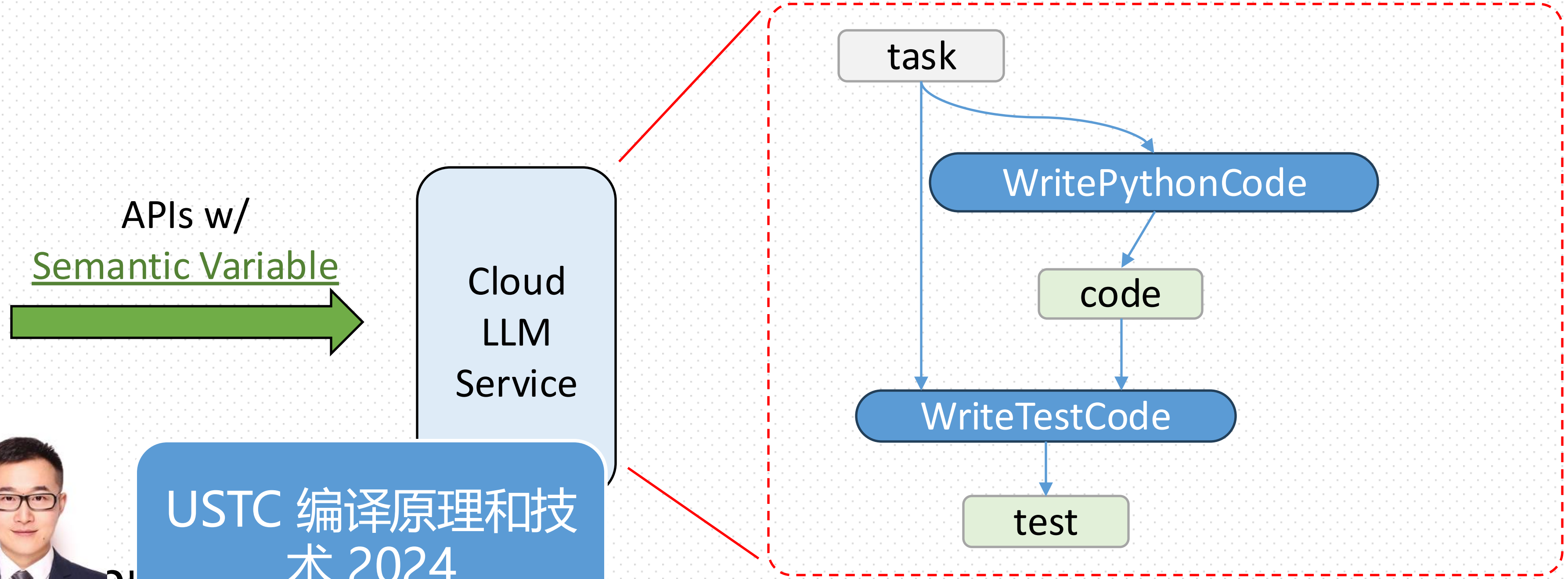
- DAG construction between requests
- Prompt structure analysis
- Data pipelining between requests

…

Parrot Overview

Exposing Semantic Variable to Parrot LLM Service

LLM requests



Semantic Va pı

- *DAG* construction between requests
- Prompt structure analysis
- Data pipelining between requests

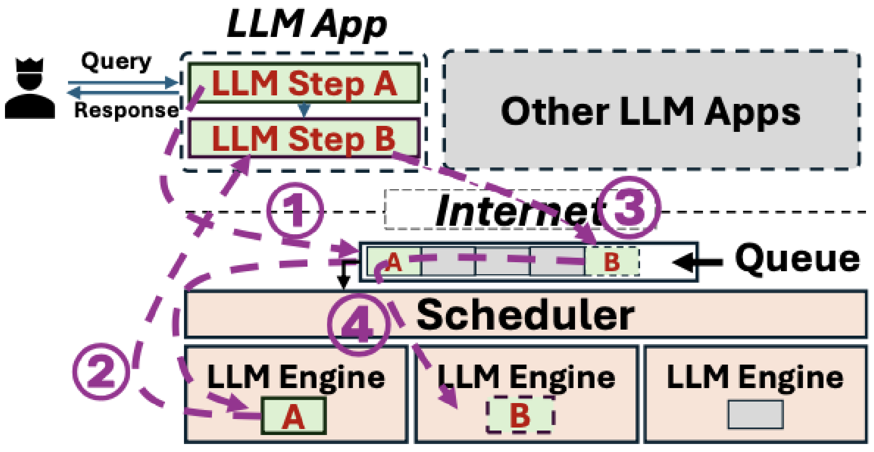
…

Parrot Overview

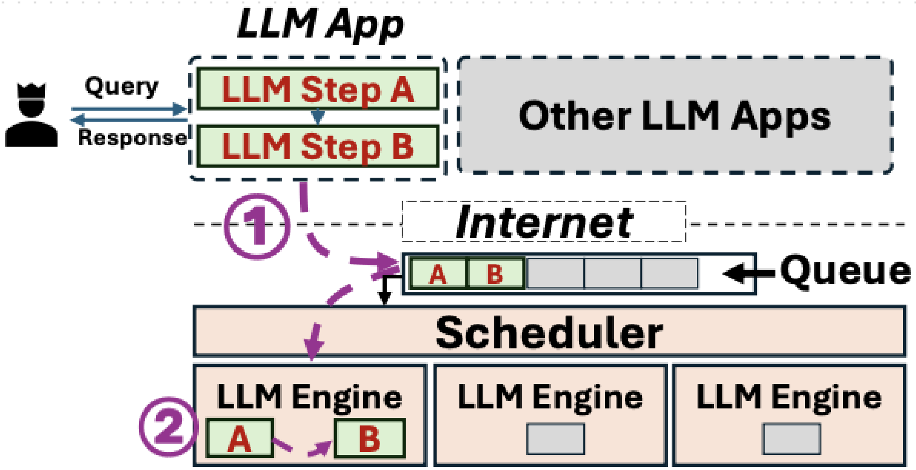
Optimization: Scheduling Dependent Requests

USTC, CHINA ADSLAB

- Optimizing dependent requests by using semantic variables
  - Decreased Network Communication



Current LLM service



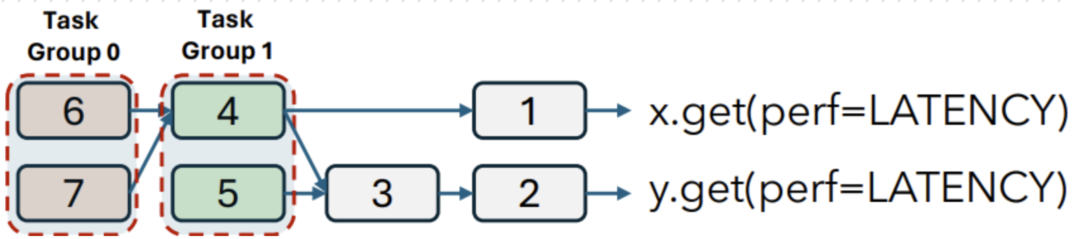
Parrot Design

Two steps are scheduled together with result of A be fed int B directly

- Avoid unnecessary network communication
- Avoid queuing delay from other apps

Optimization: Performance Criteria

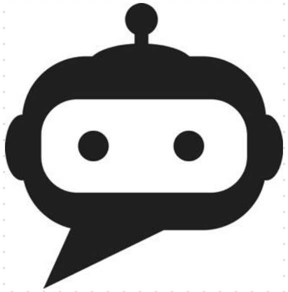
- With DAG of application requests & E2E requirement
- Derive the performance requirement of each LLM call
  - High throughput Variables: all relevant requests are marked as thpt-preferred
  - Latency sensitive variables:
    - Reverse topological order analysis
    - Direct-linked requests and predecessor are marked as latency-preferred
    - Parallel requests at the same stage are grouped together, higher batch size



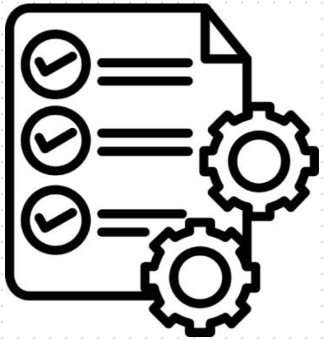
From the DAG, derive requests can be executed in parallel

Optimization: Performance Criteria

- Public LLM Service w/ apps with different performance criteria



Chatbot: Low Latency



Data Analytics: High Throughput

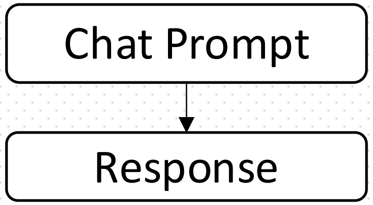
Batch Size	Small	Large

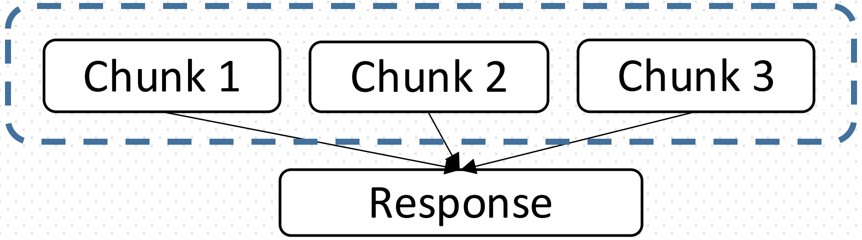
Conflict when scheduled to the same GPU engine

Optimization: Performance Criteria

- Public LLM Service w/ apps with different performance criteria

Application DAG





High Throughput

response.get(perf=LATENCY)

response.get(perf=LATENCY)

Chatbot: Low Latency

Data Analytics: High Throughput

Batch Size

Small

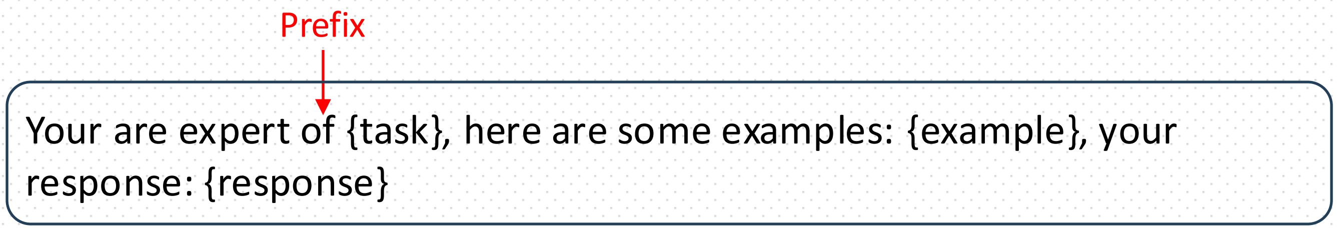
Large

Parrot can derive request-level scheduling goal from end-to-end requirement

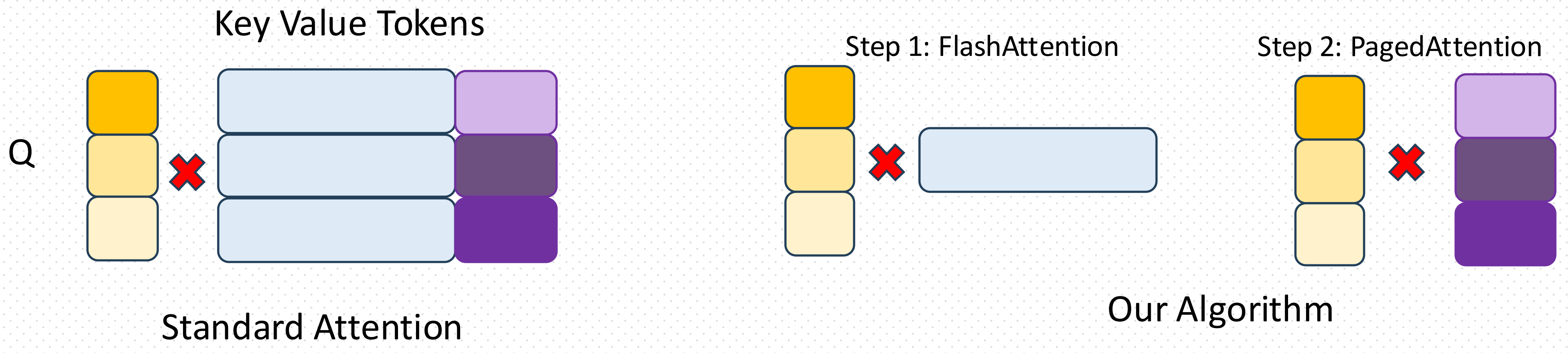
Optimization: Sharing Prompt Prefix

USTC, CHINA ADSLAB

- With prompt structure, Parrot can **automatically** detect shared prefix



- Optimized CUDA Kernel
  - Two-phase attention: avoid recomputing and reloading shared prefix



Optimization: App-centric Scheduling

USTC, CHINA ADSLAB

Topological order

Performance criteria

Schedule task group together

Shared prefix

Algorithm 1: Parrot's Request Scheduling.

Data: Q: the request queue

Q.sort() ; `/* Topological order */`

for $r \in Q$ **do**

  SharedReqsInQueue, CtxInEngine = FindSharedPrefix(r);

  if $r.TaskGroup \neq \varnothing$ **then**

    r^* = FindEngine(r.TaskGroup);

  else if $SharedReqsInQueue \neq \varnothing$ **then**

    r^* = FindEngine(SharedReqsInQueue);

  else if $CtxInEngine \neq \varnothing$ **then**

    r^* = FindEngine(r, filter=CtxInEngine);

  if $r^* = \varnothing$ **then**

    r^* = FindEngine(r);

  Q.remove(r^*);

Discussion

- Dynamic Applications and Function calling
  - Security risk of public service
  - New optimization space when offloading them into LLM server-side
- Inter-Request Analysis with Other Applications
  - Unexplored features: outlier handling, job failures, scheduling fairness, etc.
- Parrot compatibility with LLM orchestration frameworks
  - Template-based programming style
  - Extending existing LLM calls with semantic variables

Agenda

USTC, CHINA ADSLAB

- LLM Service and Application
- Problem Statement
- Design and Optimizations
- Evaluations
- Summary

Experimental Setup

USTC, CHINA ADSLAB

- Testbed
  - 1 server with a 24-core CPU and 1 A100 GPU
  - 1 server with a 64-core CPU and 4 A6000 GPUs
  - 200-300ms emulating the Internet latency
- Workloads
  - Model utilized: LlaMA 7/13B model
  - Task-1: long document analysis with Arxiv dataset
  - Task-2: BingCopilot with synthesized user queries
  - Task-3: Multi-agent application via MetaGPT
  - Task-4: Mixed workload (chat application + task-1)

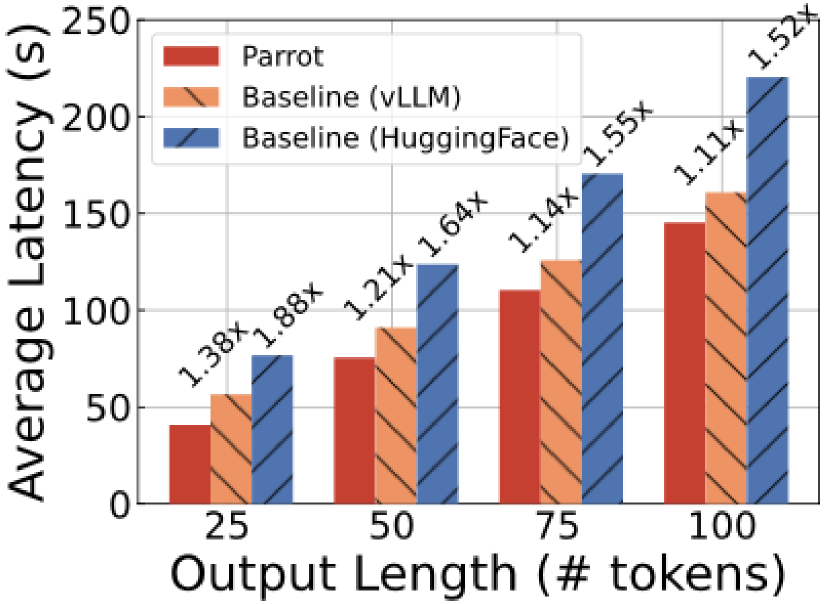
Workload	Serving Dependent Requests.	Perf. Obj. Deduction	Sharing Prompt	App-centric Scheduling
Data Analytics	✓	✓		✓
Serving Popular LLM Applications			✓	✓
Multi-agent App.	✓	✓	✓	✓
Mixed Workloads	✓	✓		✓

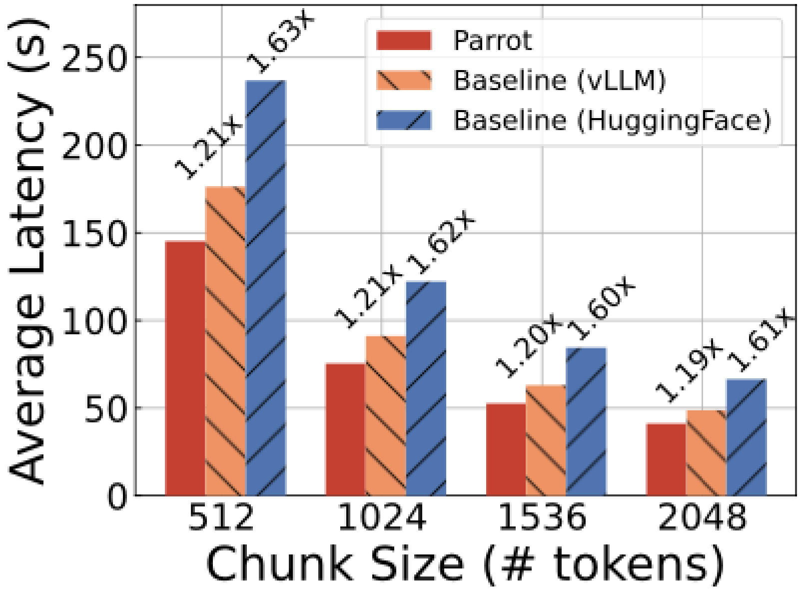
- Baseline
  - $\underbrace{\text{Application framework}}_{\text{Langchain}}$ + $\underbrace{\text{LLM serving + Engine Backend}}_{\text{FastChat + HG transformer/vLLM}}$

Evaluation: Chain/Map-Reduce Summary

USTC, CHINA ADSLAB

Average E2E latency of chain summarization

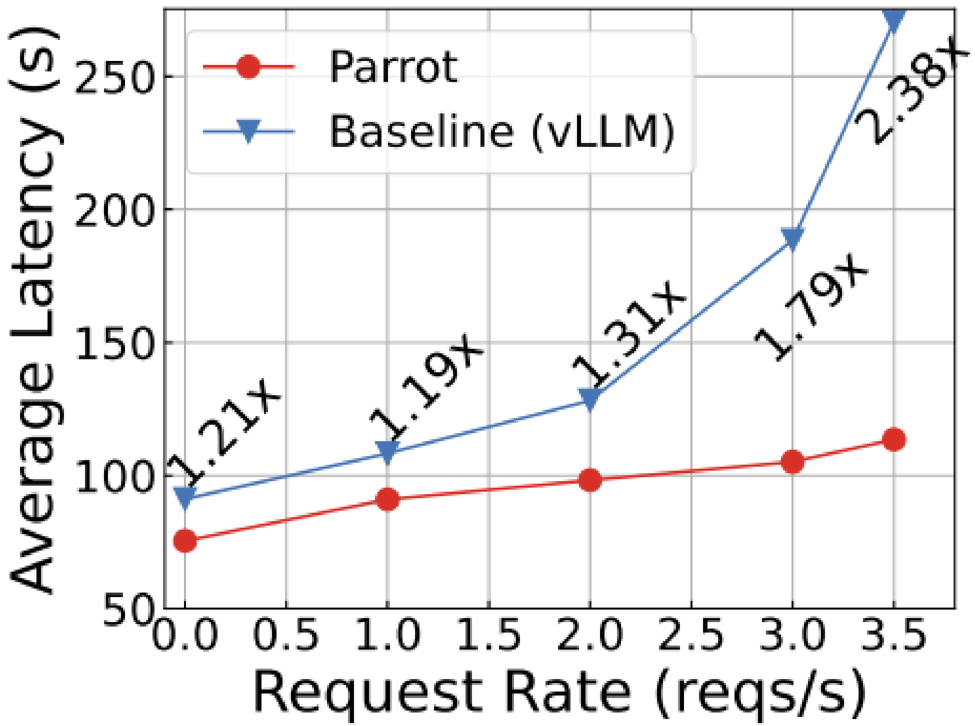




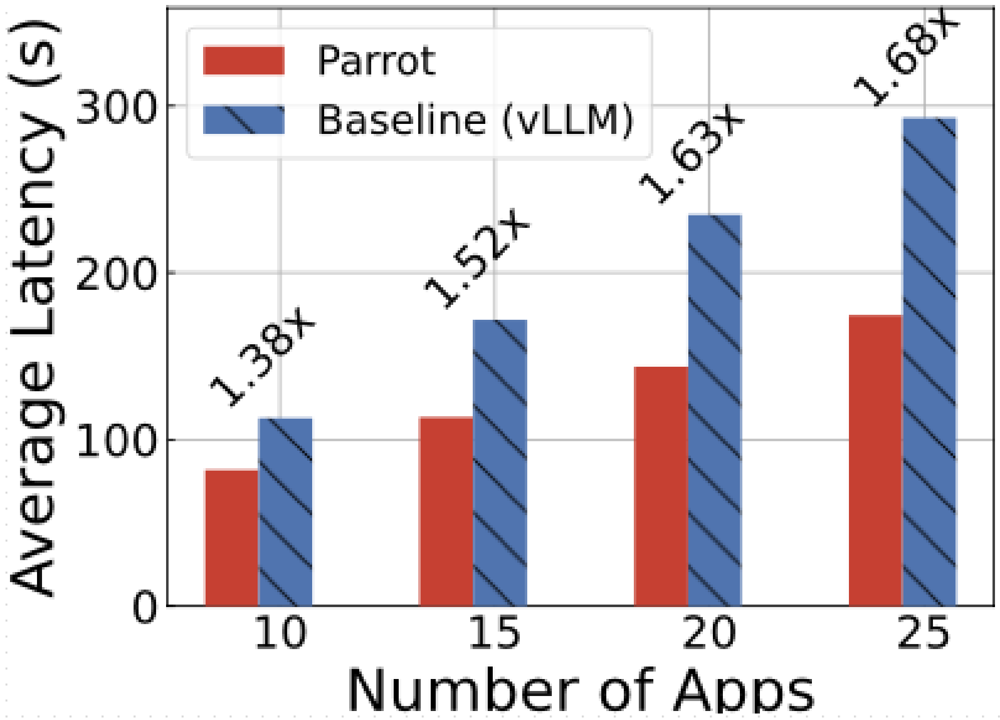
Parrot achieves a 1.38× and 1.88× reduction in latency over baselines due to decreased network latency.

Evaluation: Chain/Map-Reduce Summary

Chain Summary with queued delay



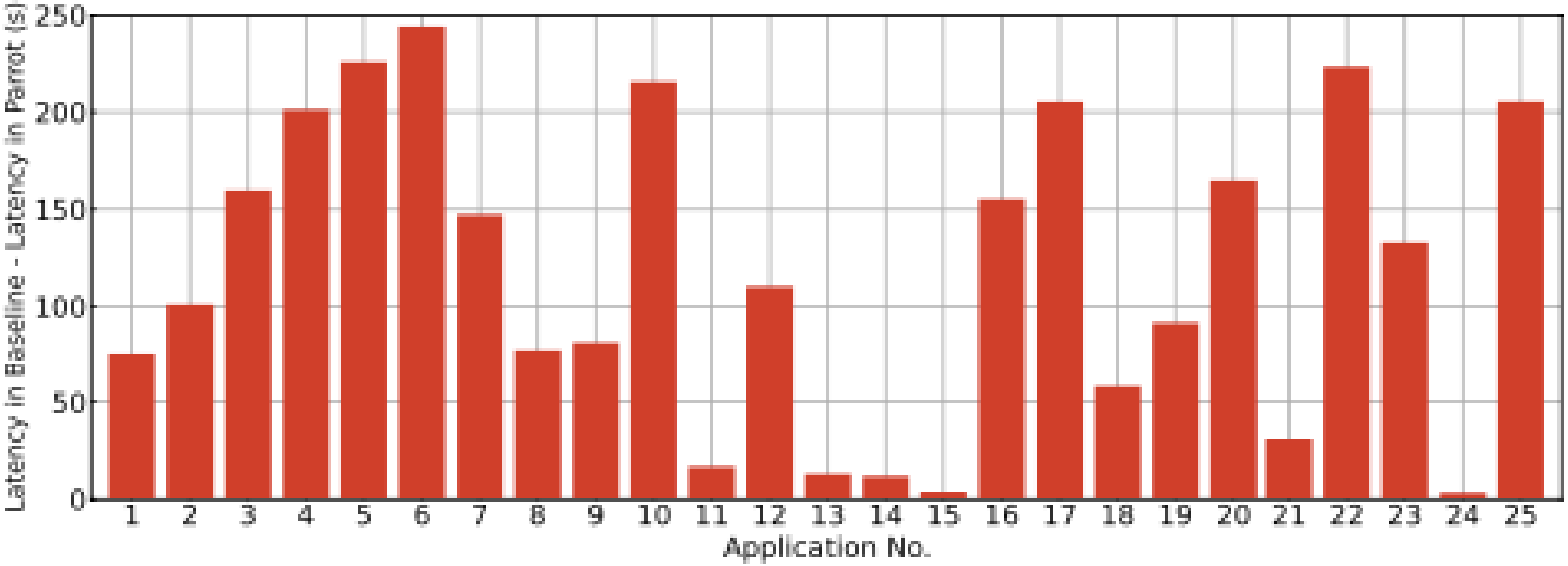
Multiple summary apps



- Parrot slashes latency by up to 2.38× since it further reduces queuing latency
- Slowdown due to interleaved execution of all applications

Evaluation: Chain/Map-Reduce Summary

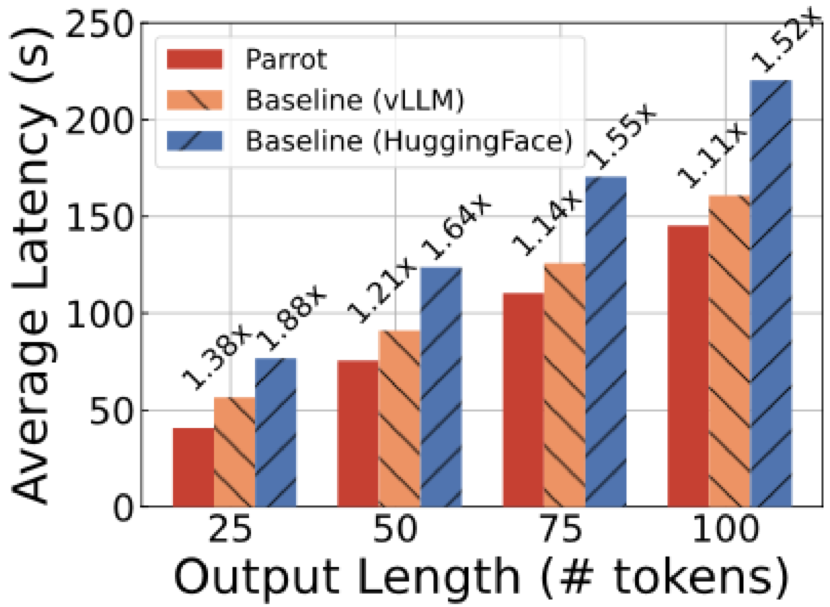
USTC, CHINA ADSLAB

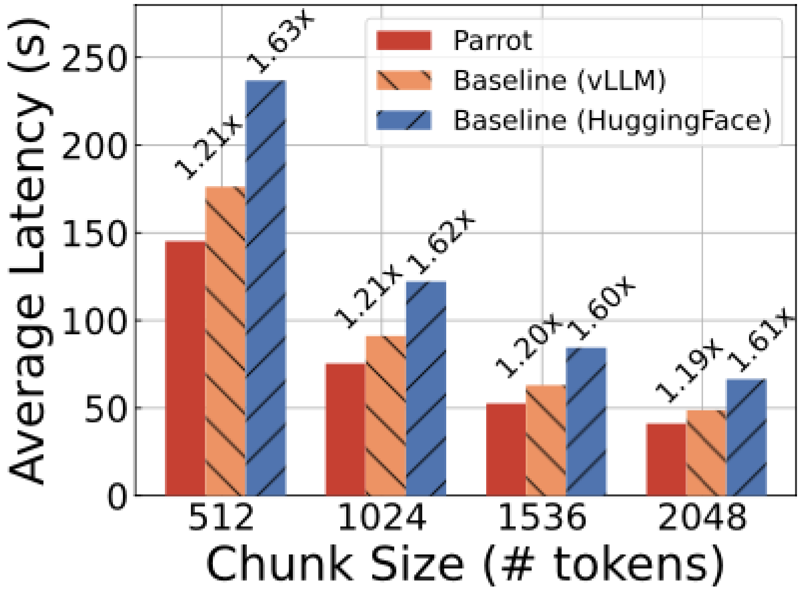


The difference in E2E latency of the 25 chain-summary application between Baseline and Parrot.

Evaluation: Chain/Map-Reduce Summary

Average E2E latency of map-reduce summarization



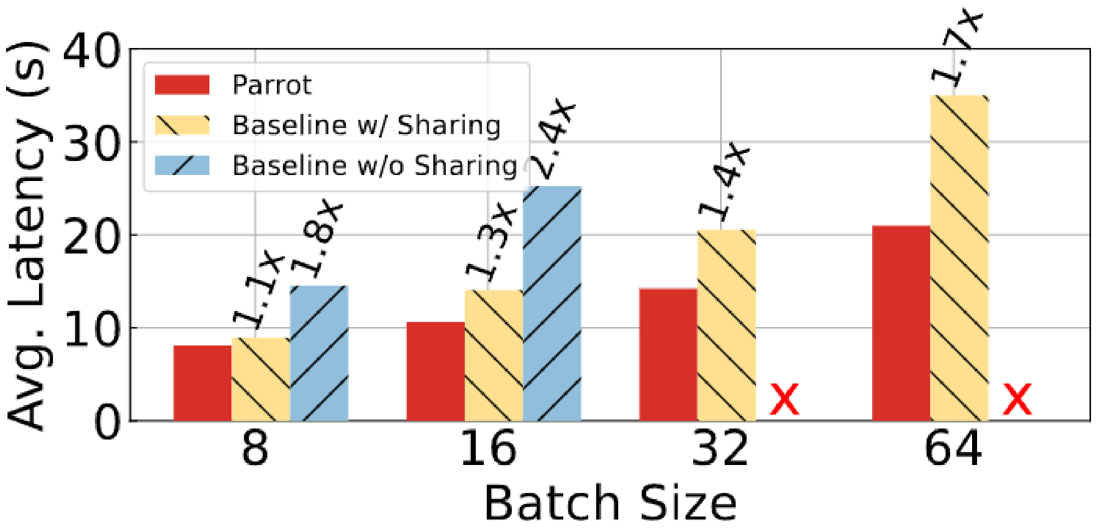


Parrot realizes a 2.37× acceleration over baselines by identifying the map task as a task group (higher batch)

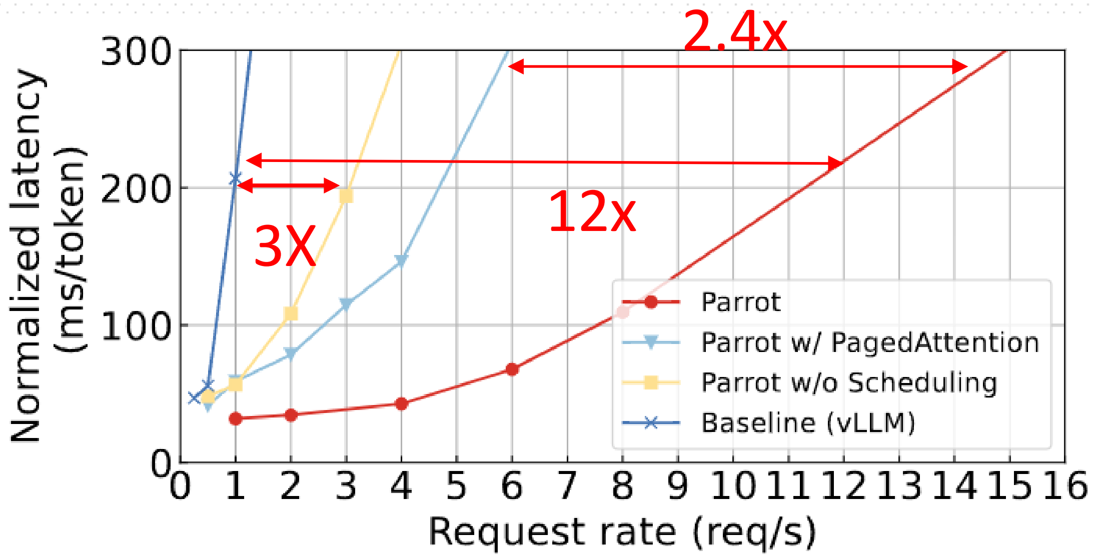
Evaluation: Popular Apps (Bing Copilot, GPTs)

USTC, CHINA ADSLAB

Synthesized requests following Bing Copilot length distribution



Synthesized requests from 4 different popular GPTs applications



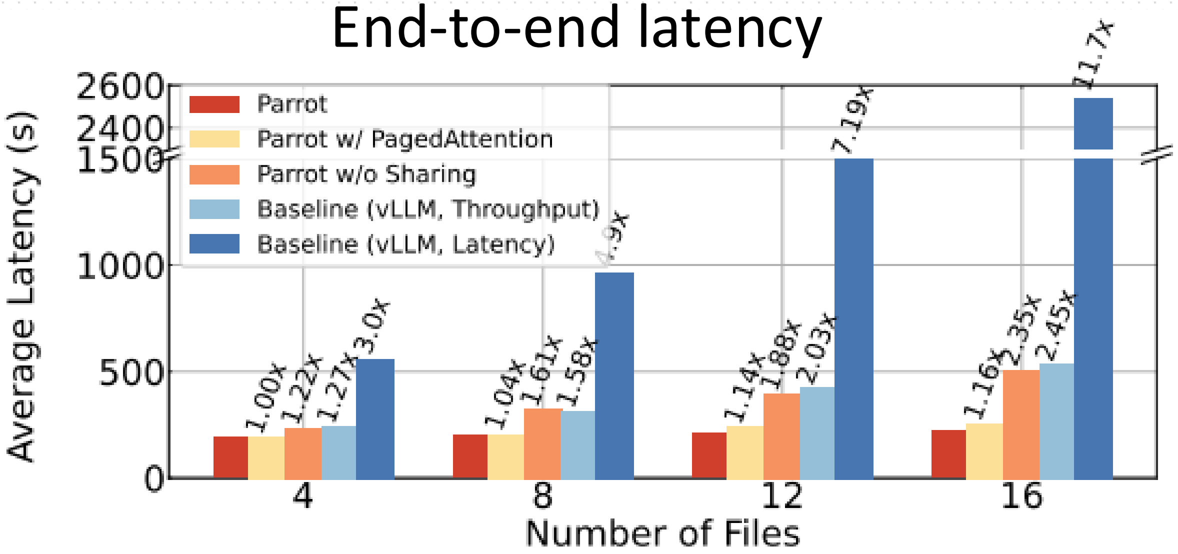
- Production prompts show up to 1.7x latency reduction due to better GPU kernel
- Parrot can sustain 12× higher request rates compared to the baseline without sharing.
  - Only 3X higher request rates without co-locate requests from the same app.
  - Even compared with paged attention, Parrot achieves 2.4x throughput improvement.

Evaluation: Multi-agent Applications

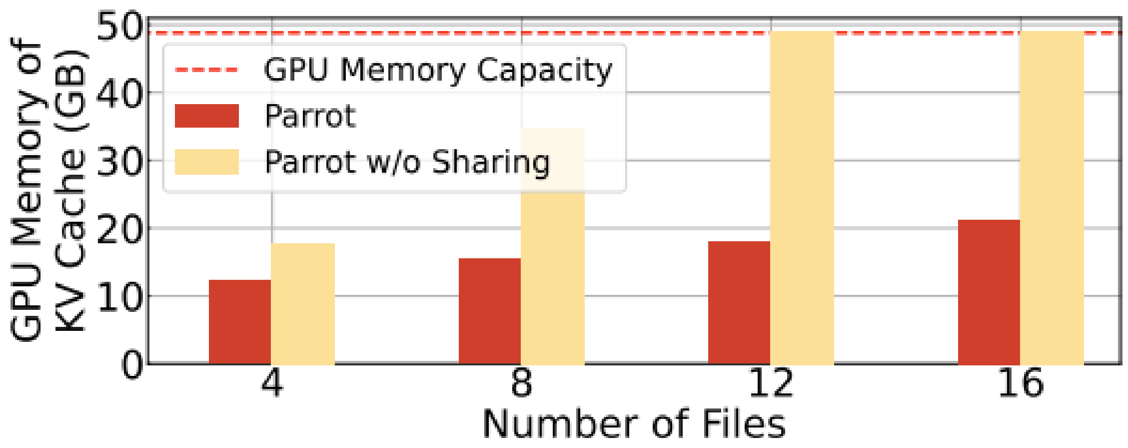
USTC, CHINA ADSLAB

- MetaGPT: code review and revision task
  - Architect outlines files structures and APIs
  - Reviewers leave comments for each file
  - Coders revise codes based on comments

End-to-end latency



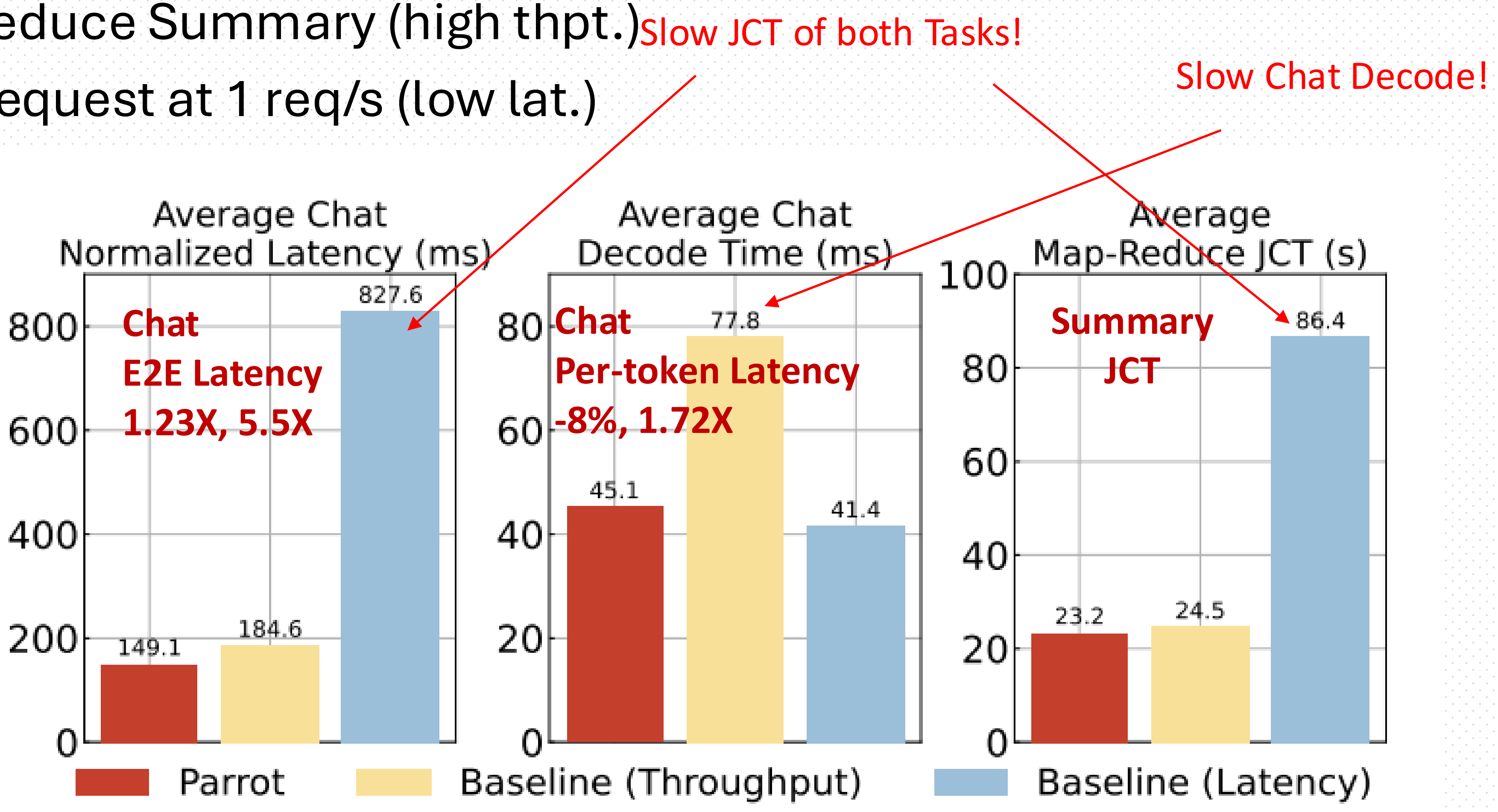
GPU Memory of KV cache



- Parrot achieves a speedup of up to 11.7× compared with the latency-centric baseline. (higher batch size)
- Even compared with throughput-centric baseline, Parrot achieves 2.45x throughput improvement. (sharing prefix)

Evaluation: Scheduling Mixed Workloads

- Mixed workloads
  - Map-reduce Summary (high thpt.)
  - Chat request at 1 req/s (low lat.)

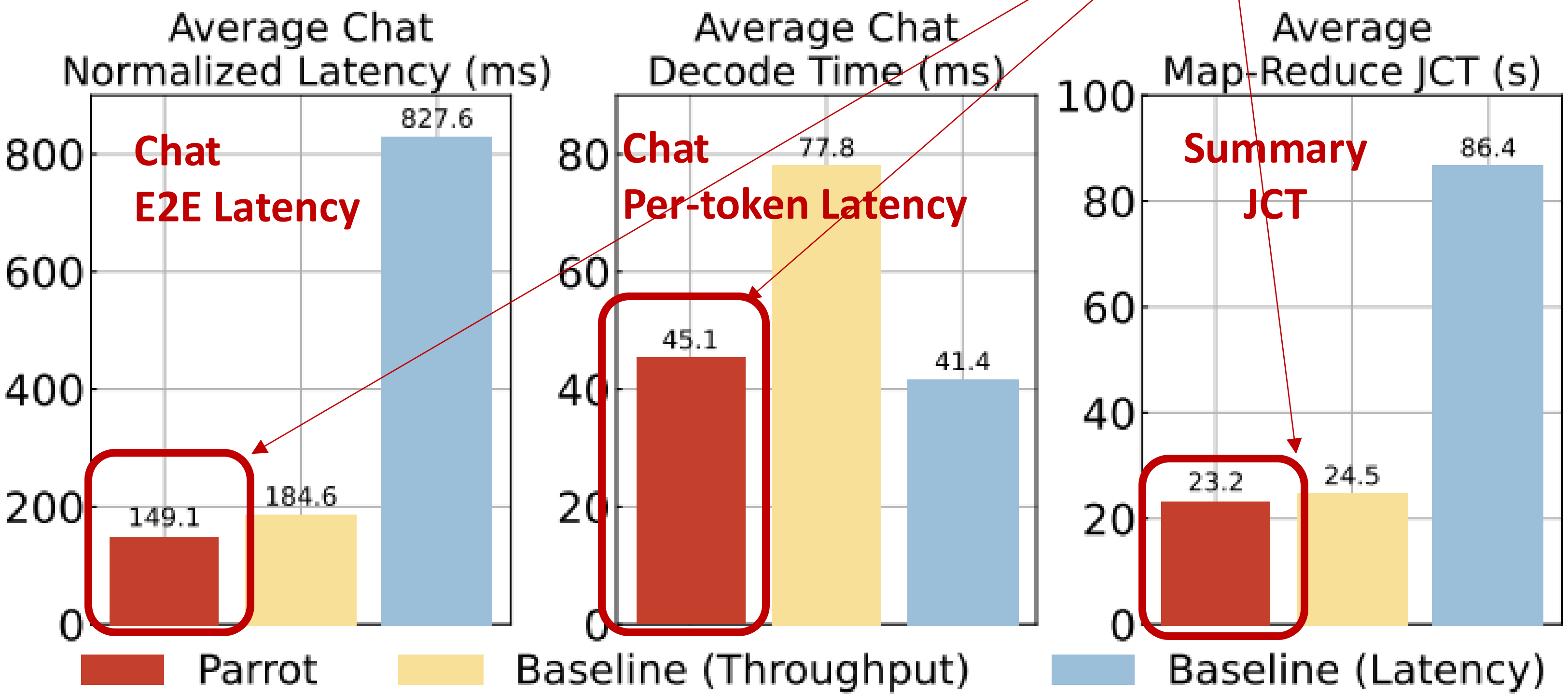


Evaluation: Scheduling Mixed Workloads

USTC, CHINA ADSLAB

- Mixed workloads
  - Map-reduce Summary (high thpt.)
  - Chat request at 1 req/s (low lat.)

Parrot achieves low latency and high-throughput for both apps



Parrot optimizes application performance by scheduling them on different engines

Agenda

USTC, CHINA ADSLAB

- LLM Service and Application
- Problem Statement
- Design and Optimizations
- Evaluations
- Summary

Pros and Cons

USTC, CHINA ADSLAB

- Pros
  - Innovative Abstraction (Semantic Variables)
  - End-to-end application-level optimization instead of request level
  - High performance gains and support for multiple workflows
- Cons
  - Potential overhead in terms of analyzing and managing variables
  - Lack of comparison to SGLang

Summary

USTC, CHINA ADSLAB

- Multi-tenant cloud LLM services running diverse apps
  - Lacking app knowledge misses many optimization opportunities

- Parrot: uses a unified abstraction Semantic Variable
  - To expose essential application-level information
  - End-to-end optimizations with dataflow analysis

- Evaluation shows order-of-magnitude efficiency improvement for practical use-cases