ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

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

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

https://github.com/microsoft/ParrotServe

上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

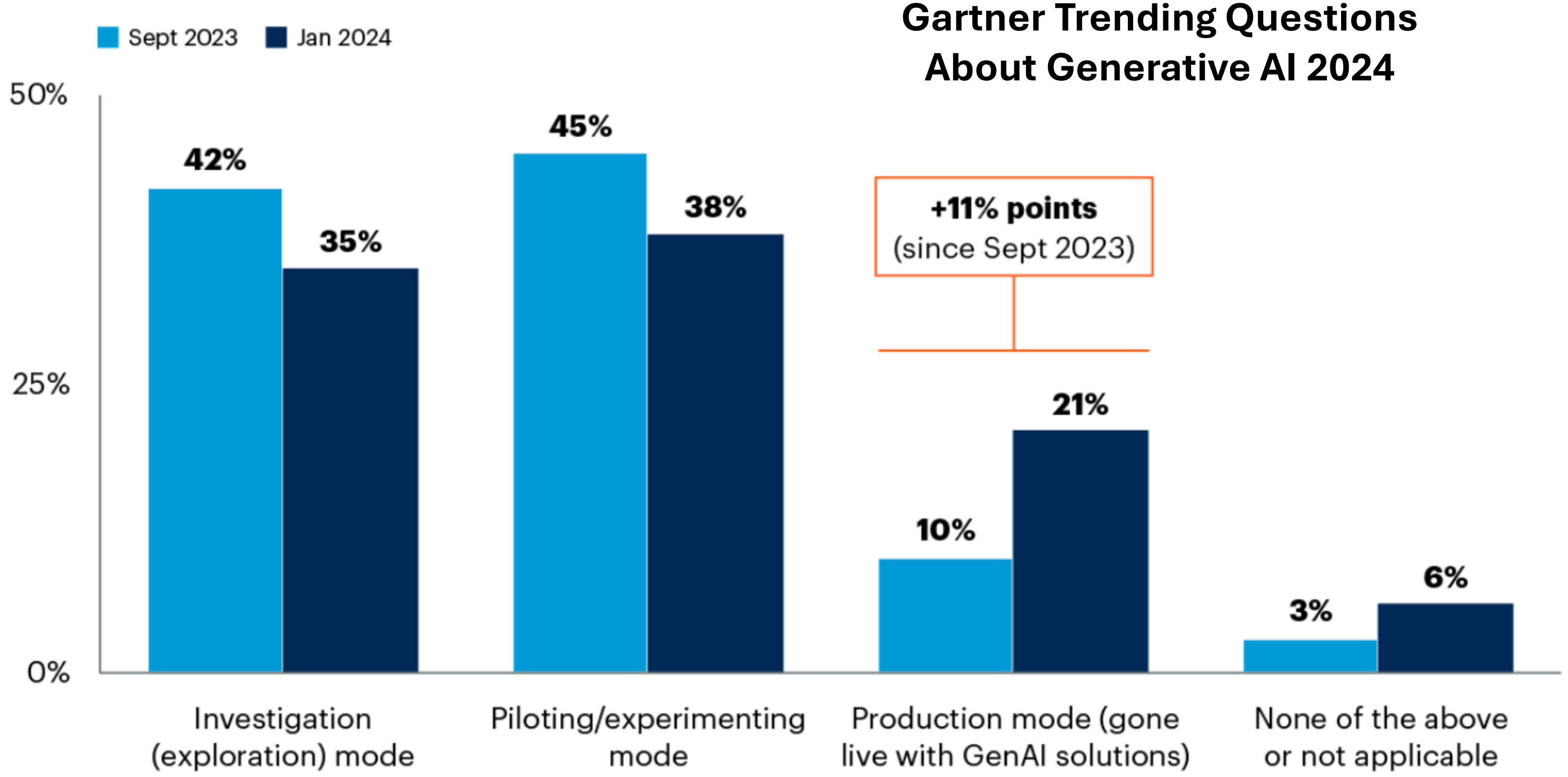
Microsoft Research

Generated by DALL·E

Paradigm Shift of Computer Programs

- A novel type of program (LLM + Code) 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

- A novel type of program (LLM + Code) 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

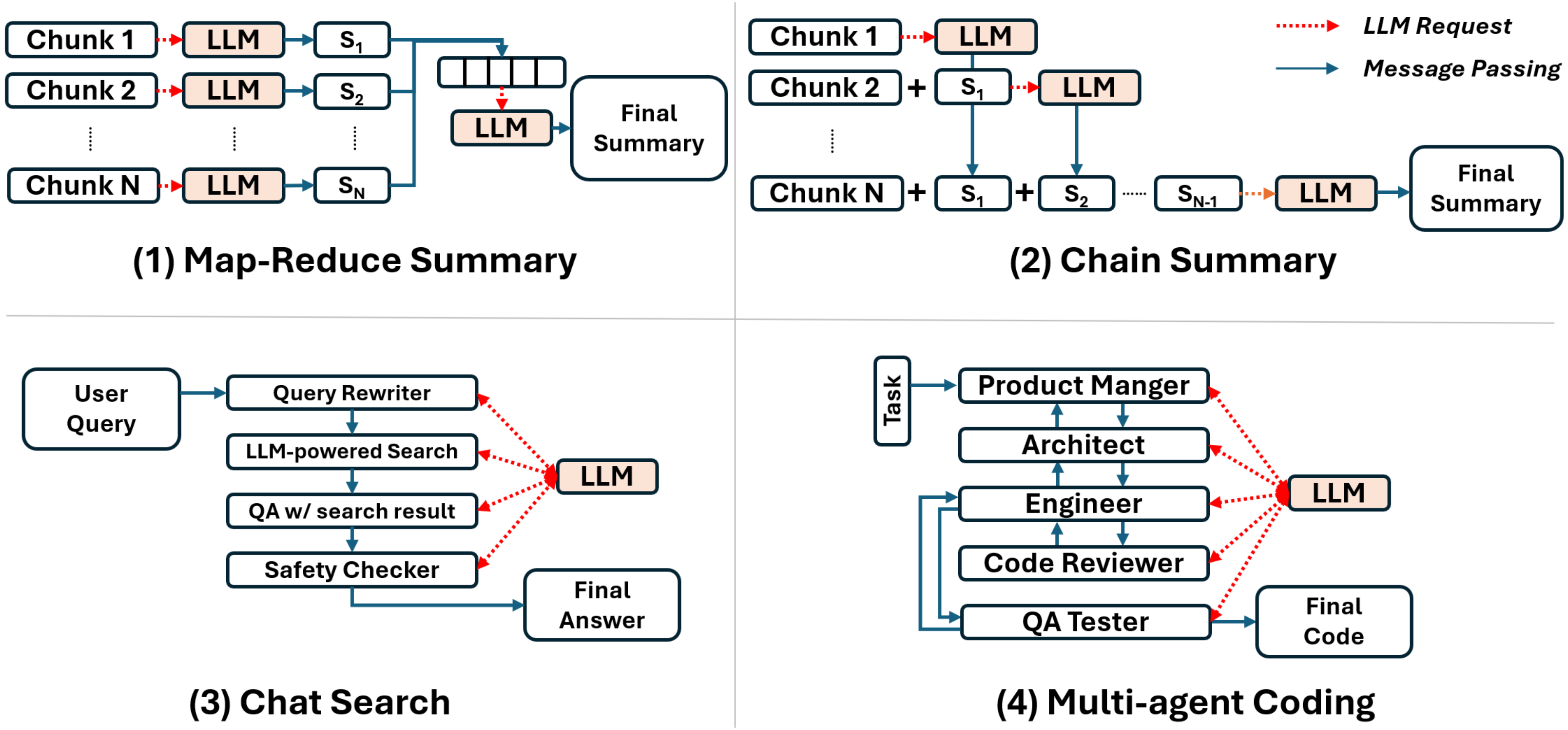
Hot!

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

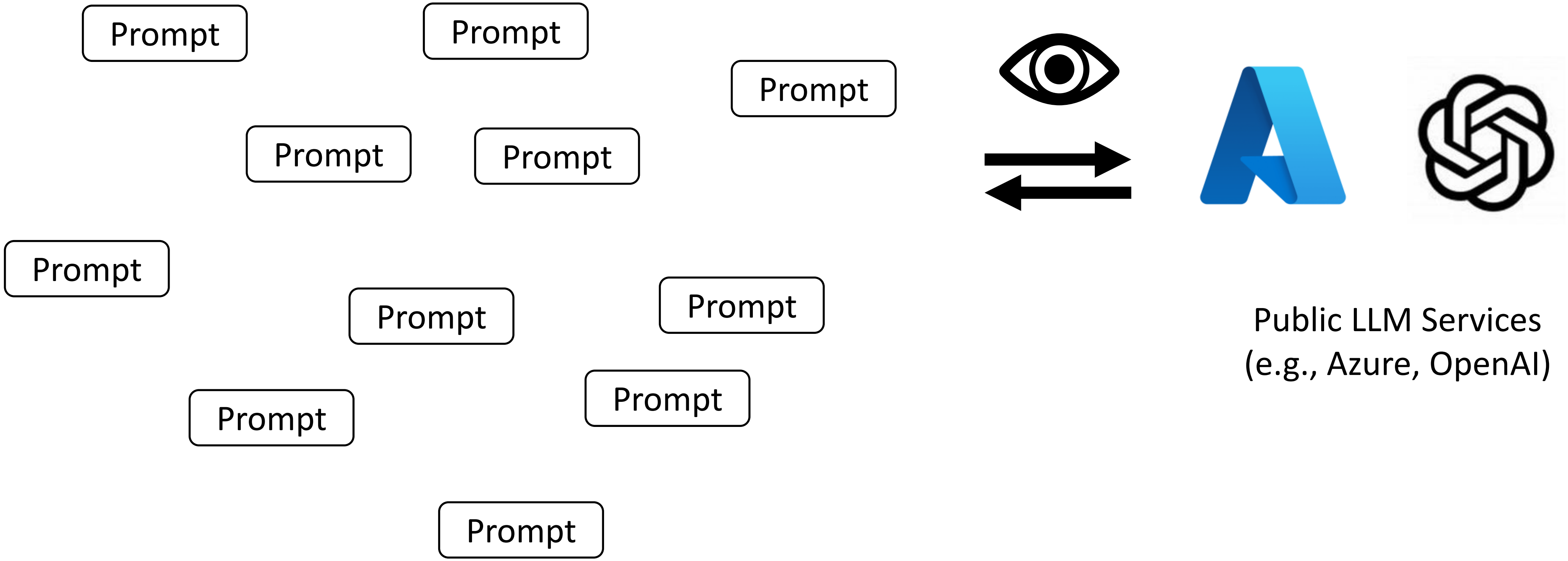
Diverse Workflows of LLM Apps (or Agents)

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



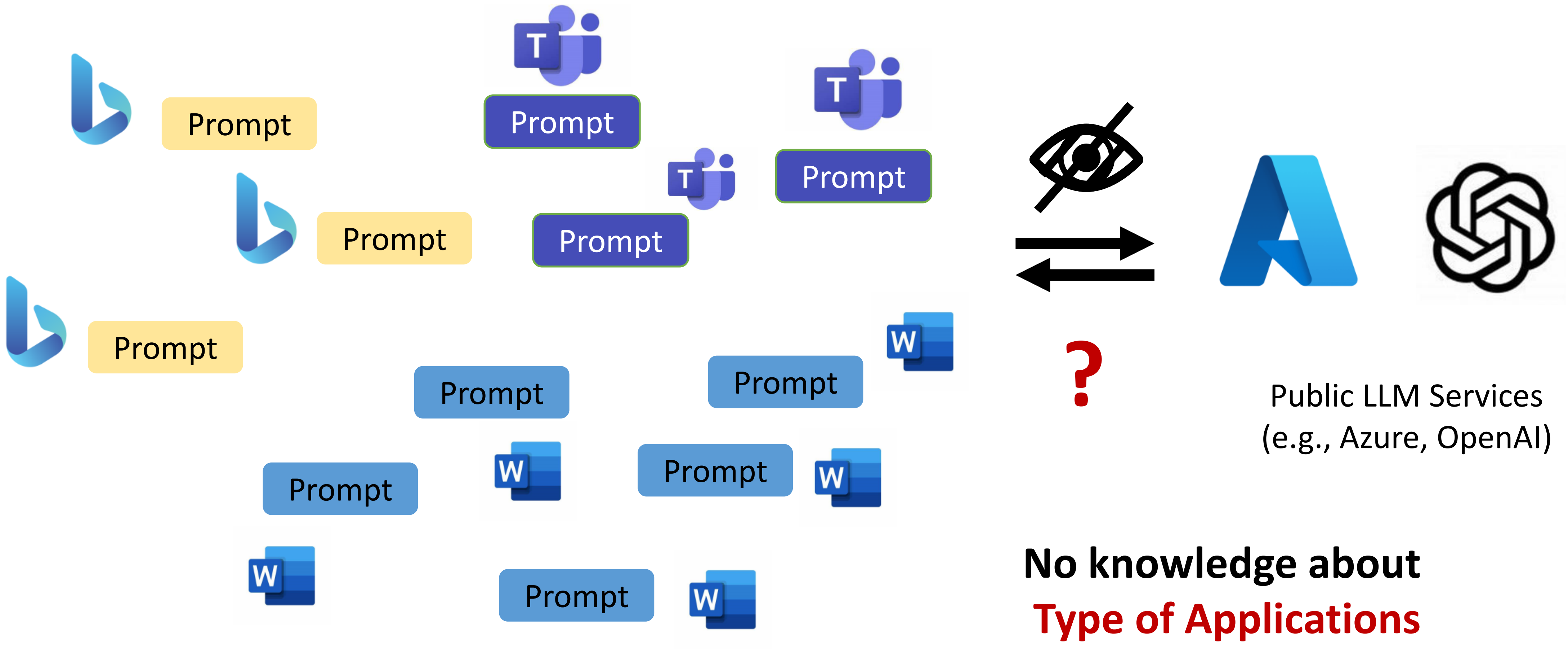
From the view of Multi-tenant LLM Services

- Face a lot of independent prompt requests through OpenAI-style APIs



From the view of Multi-tenant LLM Services

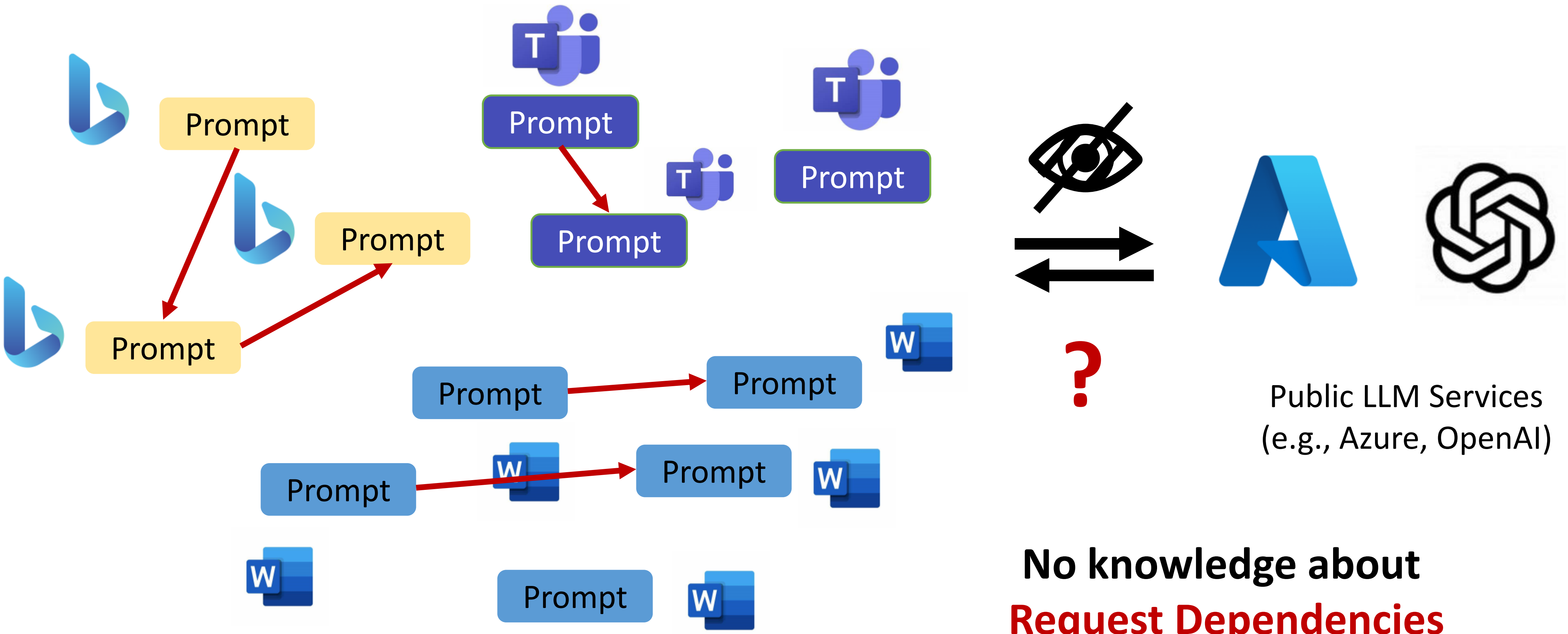
- Face a lot of independent prompt requests through OpenAI-style APIs



No knowledge about
Type of Applications

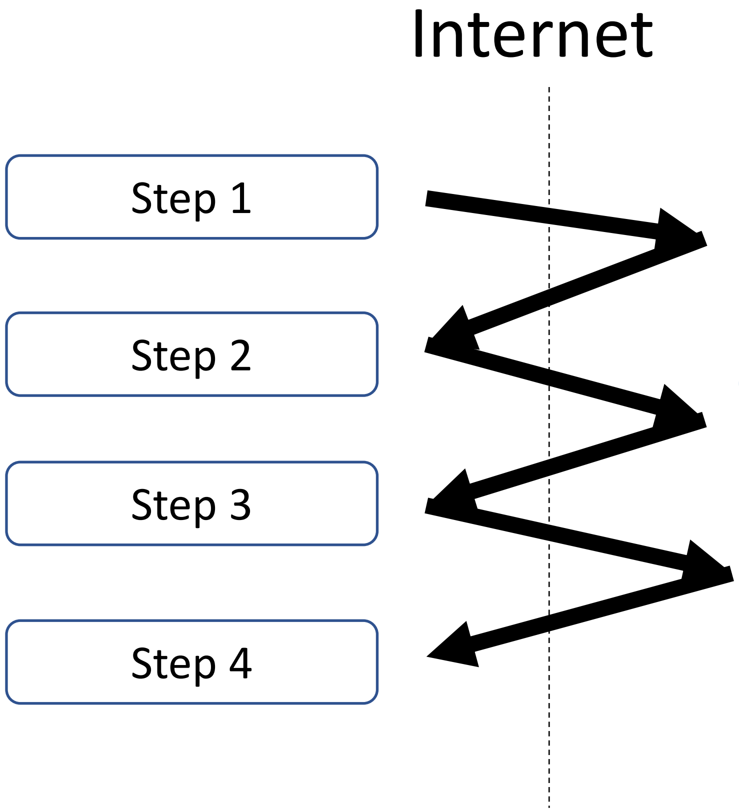
From the view of Multi-tenant LLM Services

- Face a lot of independent prompt requests through OpenAI-style APIs



No knowledge about
Request Dependencies

Problems of Lacking Application Knowledge



Multi-Request App has to use chatty submission

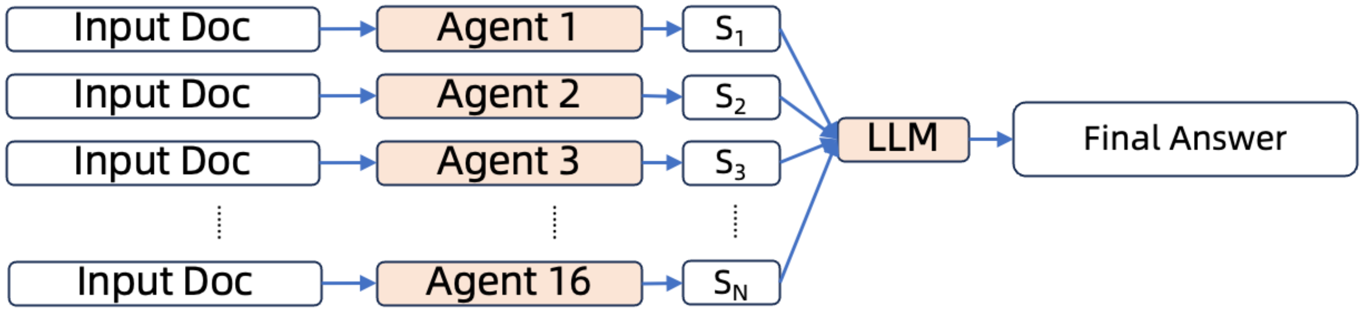
Public LLM Services
(e.g., Azure, OpenAI)

High Excessive Latency

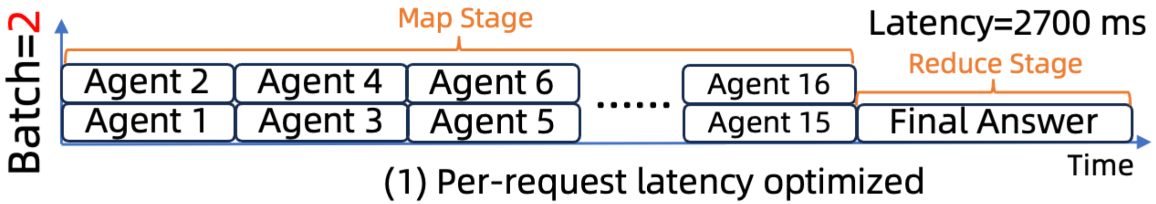
- 50~70% Non-GPU Time

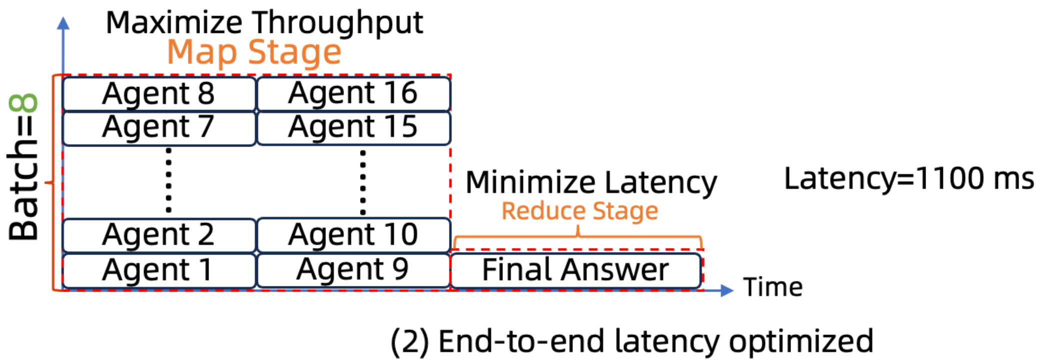
- High Internet Latency
- Excessive Queuing Delay

Problems of Request-centric LLM APIs



Misaligned
Scheduling Objectives





Small Batch Size for Low Per-Request Latency

Large Batch Size for Map Stage

Problem of Unknown Prompt Structure

- 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

Same for all user queries

User Query

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

No knowledge about
Shared Prompt Structure

Many Optimizations Not Applicable in Public LLM Services

- Public LLM Services face diverse applications

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

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

Our Goals in Parrot

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

Insight from Prompt Engineering

- 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

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

Prompt

w/ Semantic Variables as Placeholders

```
@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
"""
```

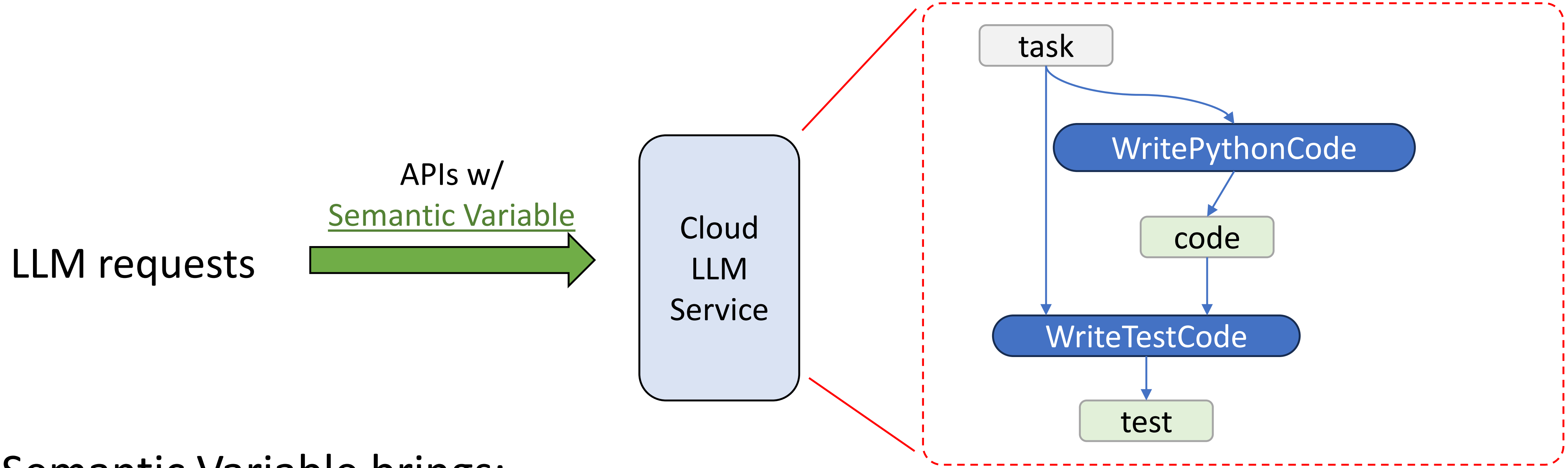
Prompt

```
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:

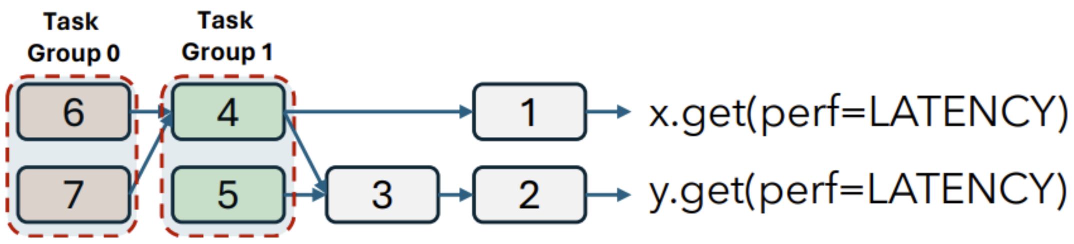
- DAG construction between requests
- Prompt structure analysis
- Data pipelining between requests

…

Parrot Overview

Optimization: App-centric Scheduling

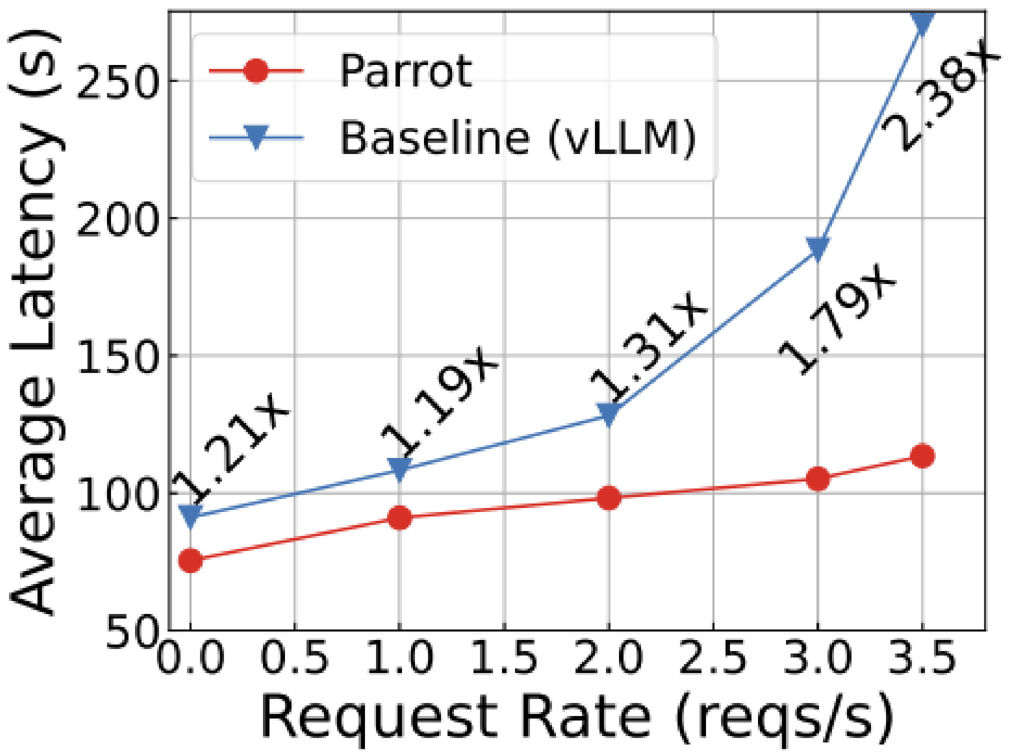
- With DAG of application requests & E2E requirement
- Derive the performance requirement of each LLM call



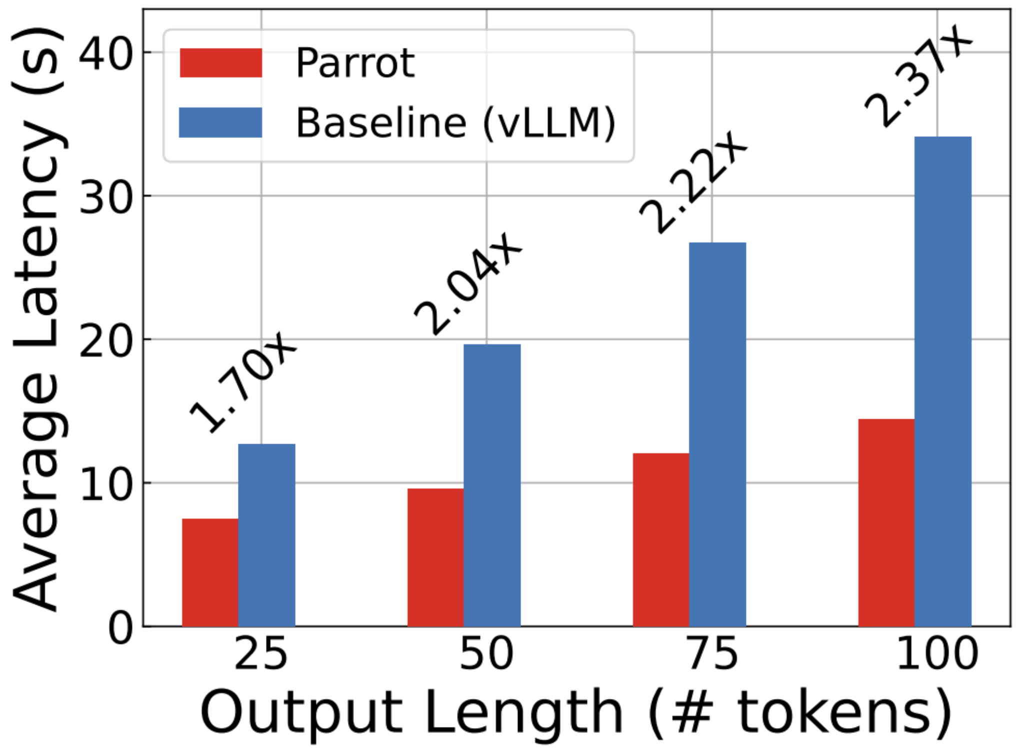
From the DAG, derive requests can be executed in parallel

Evaluation: Chain/Map-Reduce Summary

Chain Summary

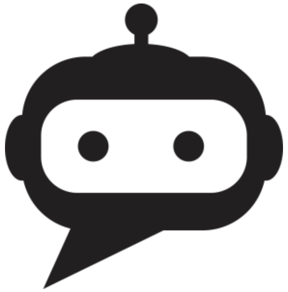


Map-Reduce Summary

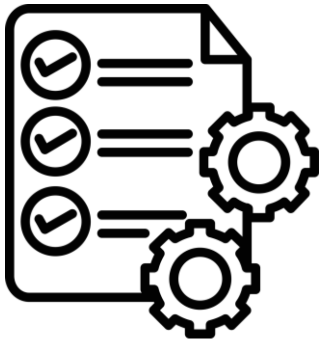


Optimization: Multi-app Serving

- Public LLM Service w/ apps with different performance criteria



Chatbot: Low Latency



Data Analytics: High Throughput

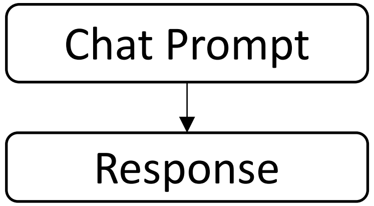
Batch Size	Small	Large

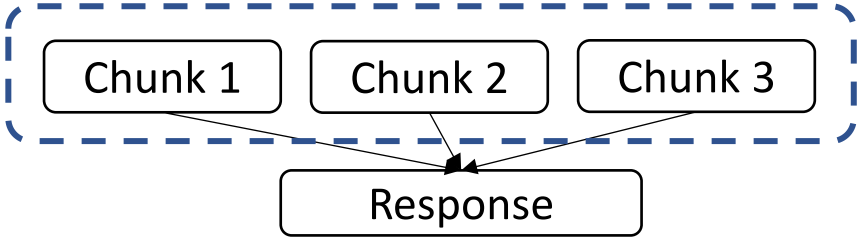
Conflict when scheduled to the same GPU engine

Optimization: Multi-app Serving

- Public LLM Service w/ apps with different performance criteria

Application DAG





High Throughput

response.get(perf=LATENCY)

Chatbot: Low Latency

response.get(perf=LATENCY)

Data Analytics: High Throughput

Batch Size

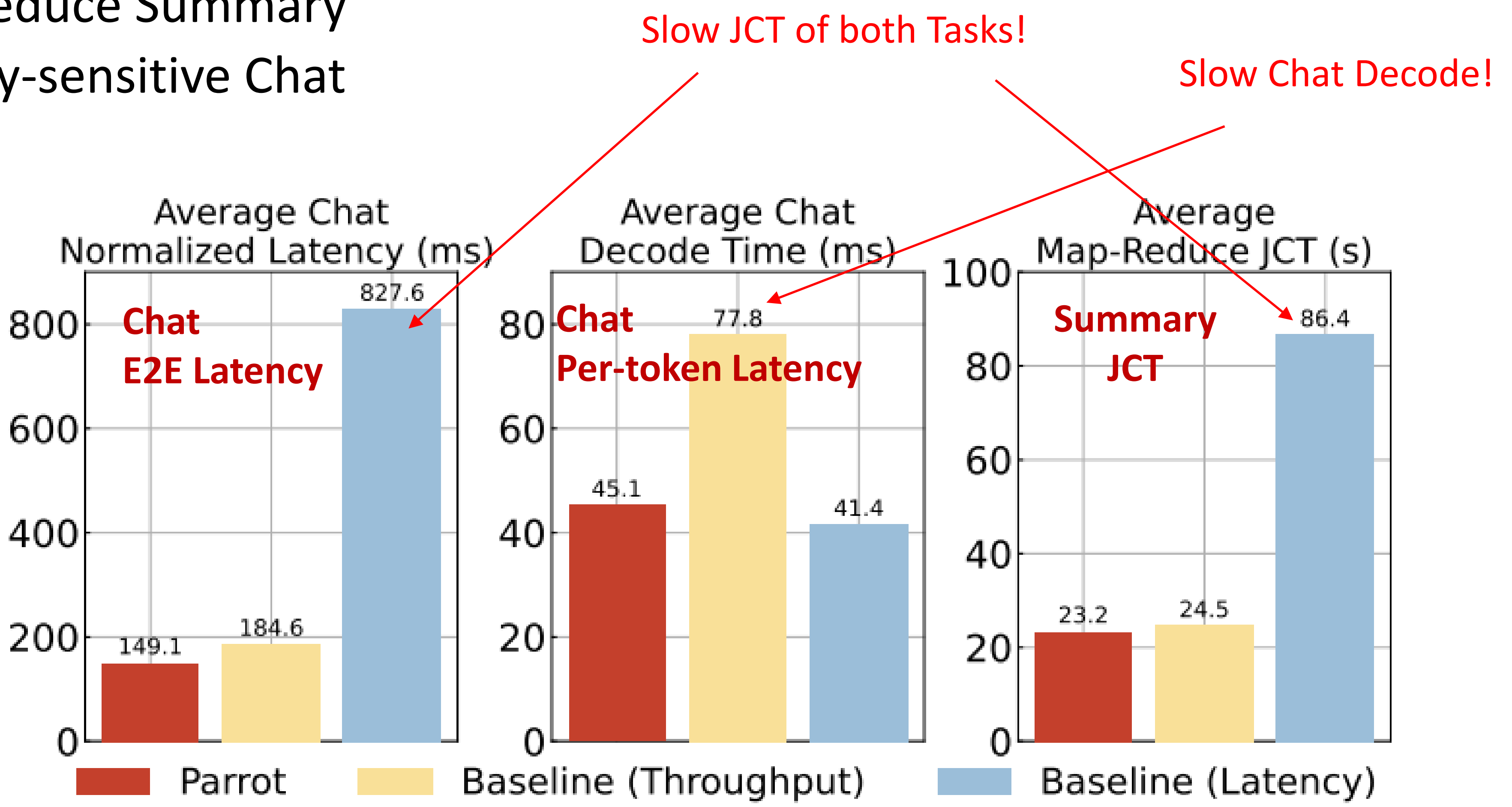
Small

Large

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

Evaluation: Scheduling Mixed Workloads

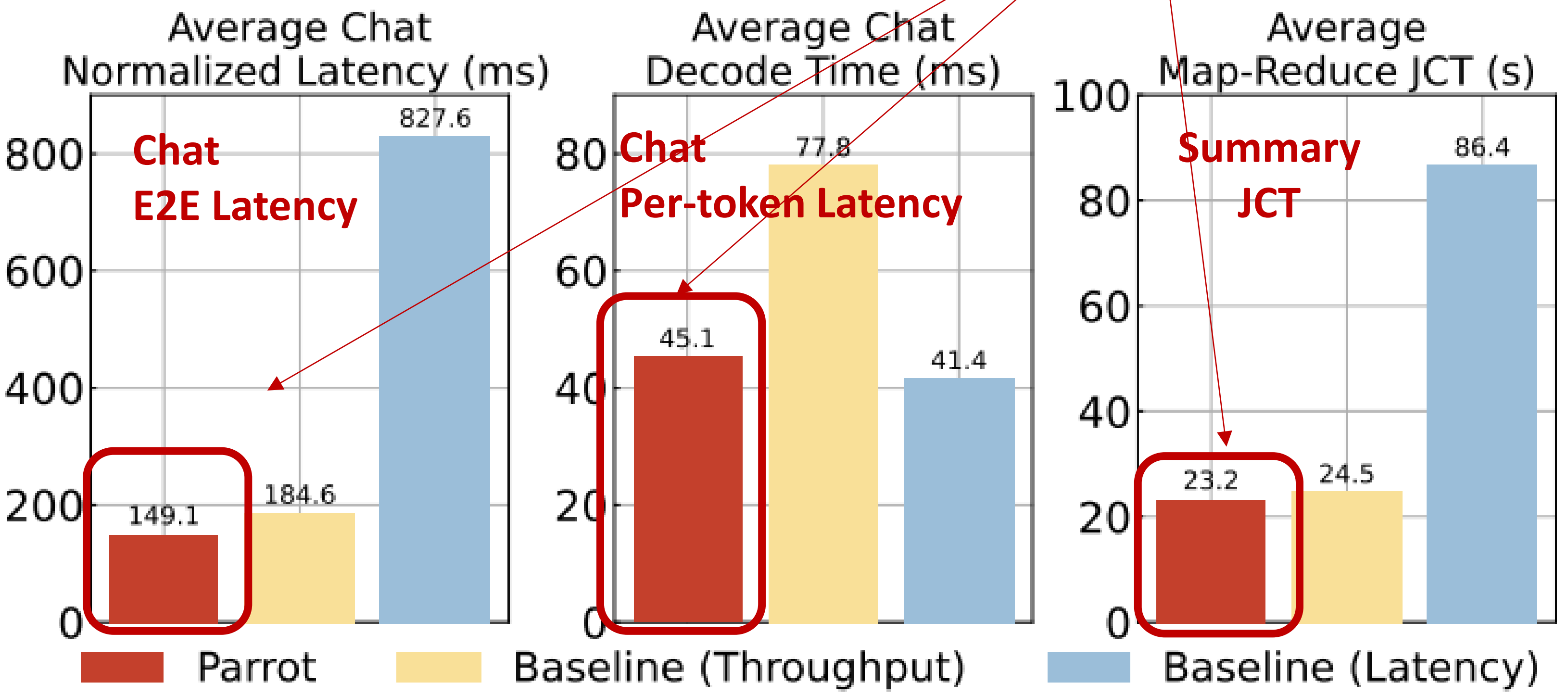
- Mixed workloads
  - Map-reduce Summary
  - Latency-sensitive Chat



Evaluation: Scheduling Mixed Workloads

- Mixed workloads
  - Map-reduce Summary
  - Latency-sensitive Chat

Parrot achieves low latency and high-throughput for both apps



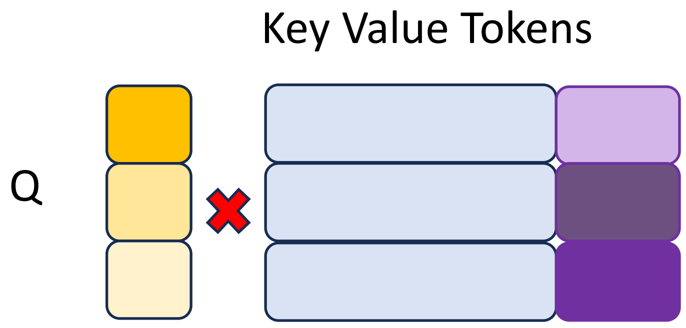
Optimization: Sharing Prompt Prefix

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

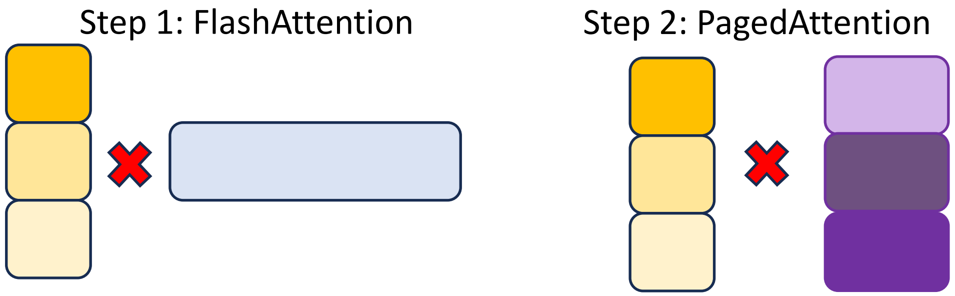
Prefix 1 Prefix 2

> Your are expert of {task}, here are some examples: {example}, your response: {response}

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



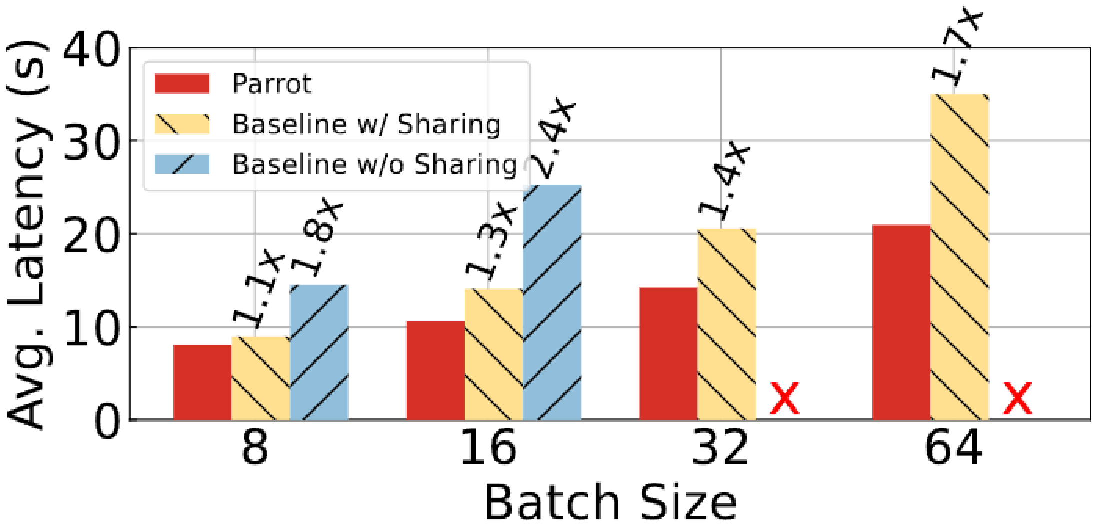
Standard Attention



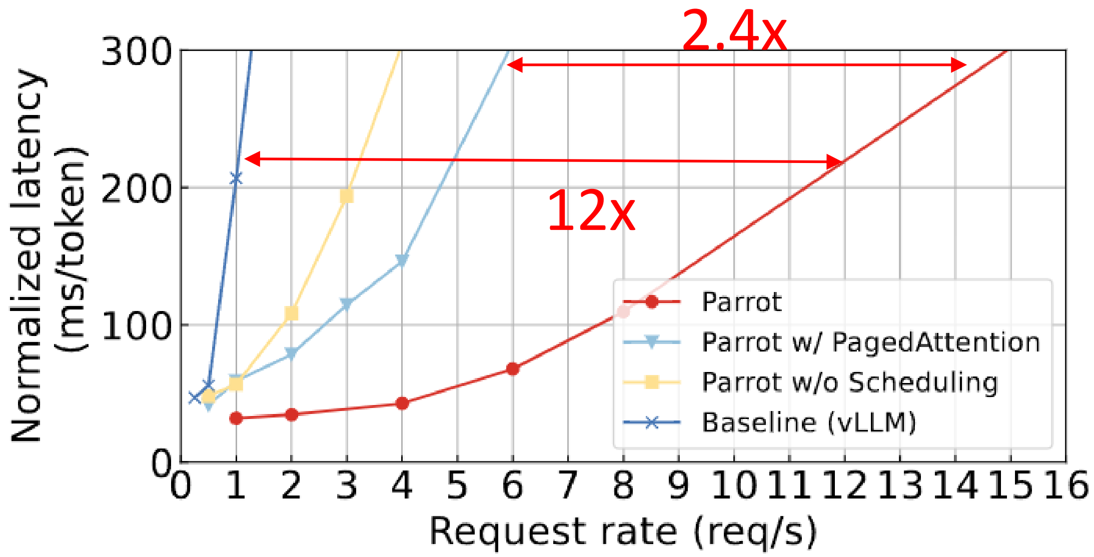
Our Algorithm

Evaluation: Popular Apps (Bing Copilot, GPTs)

Synthesized requests following Bing Copilot length distribution



Synthesized requests from 4 different popular GPTs applications



Summary

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

- 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

Microsoft Research Asia is hiring

Beijing, Shanghai, Vancouver, Singapore, Hong Kong, Tokyo, Seoul

Thanks

Zhenhua Han
hzhua201@gmail.com

Microsoft® Research