ALTO: An Efficient Network Orchestrator for Compound AI Systems

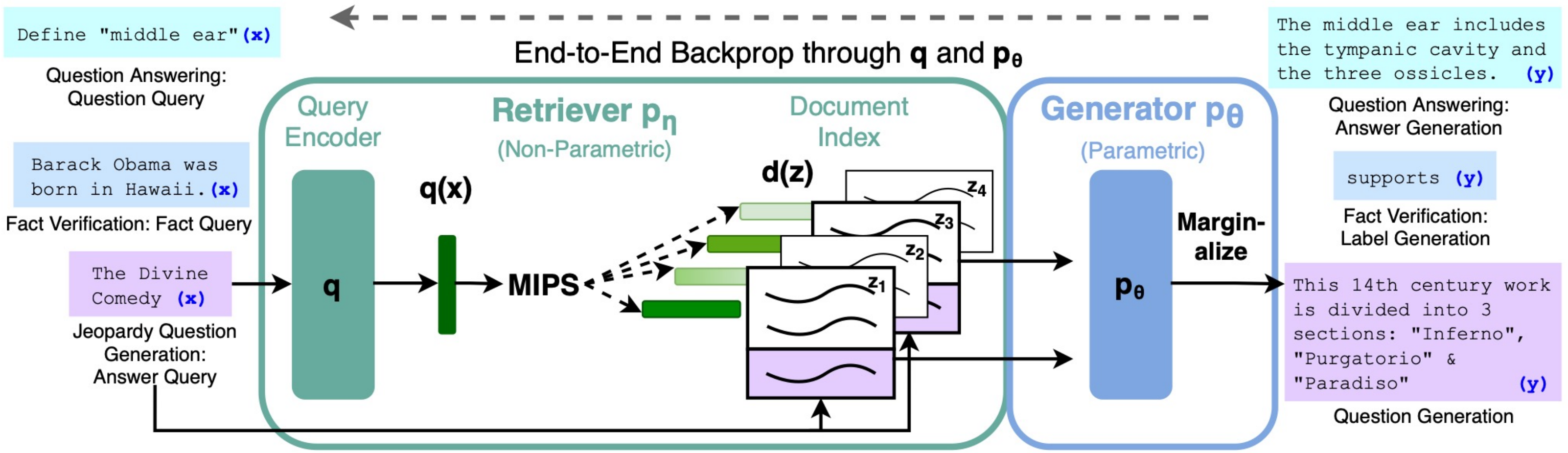
Keshav Santhanam[*1], Deepti Raghavan[*1], Muhammad Shahir Rahman[1], Thejas Venkatesh[1], Neha Kunjal[1], Pratiksha Thaker[2], Philip Levis[1], Matei Zaharia[3]

[*]Equal contribution

[1]Stanford University [2]CMU [3]UC Berkeley

Compound AI Systems

Compound AI systems combine AI models with external tools to solve challenging tasks

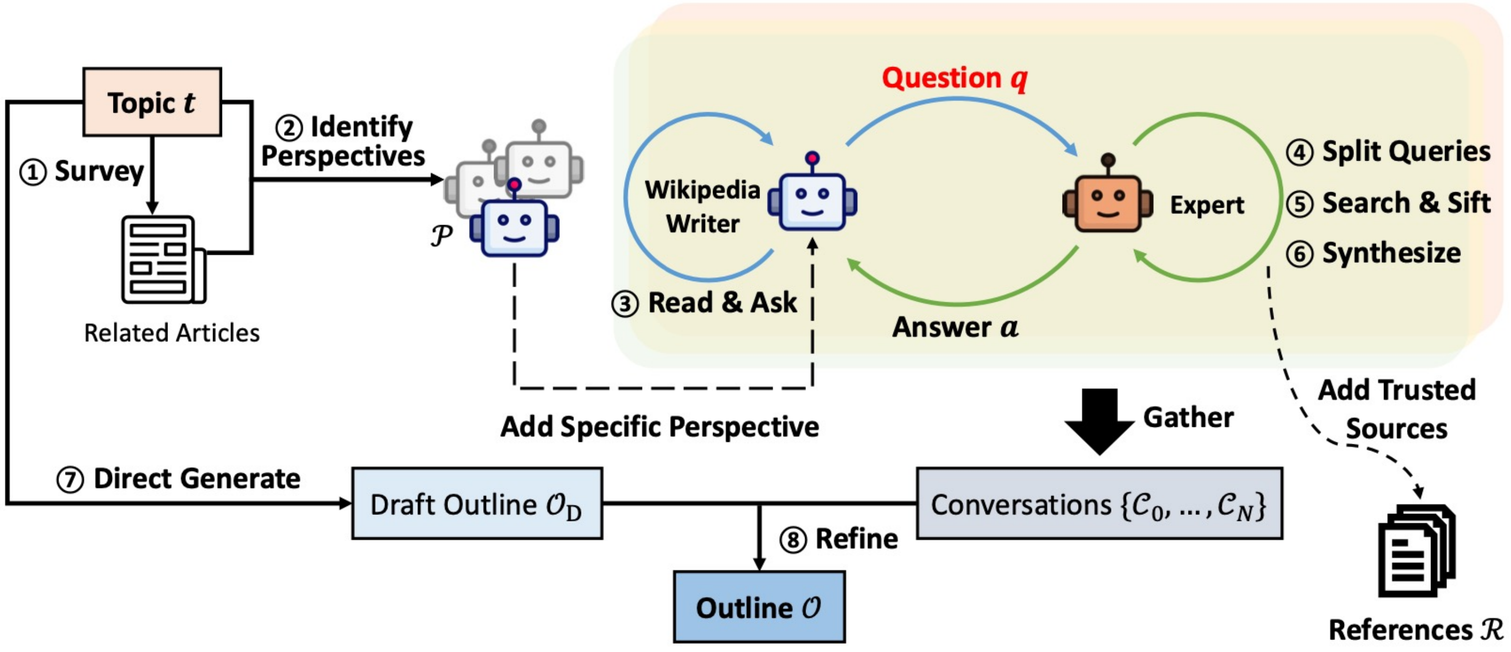


Retrieval-augmented generation (RAG)

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., & others. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in Neural Information Processing Systems, 33, 9459–9474.

Compound AI Systems

Compound AI systems combine AI models with external tools to solve challenging tasks

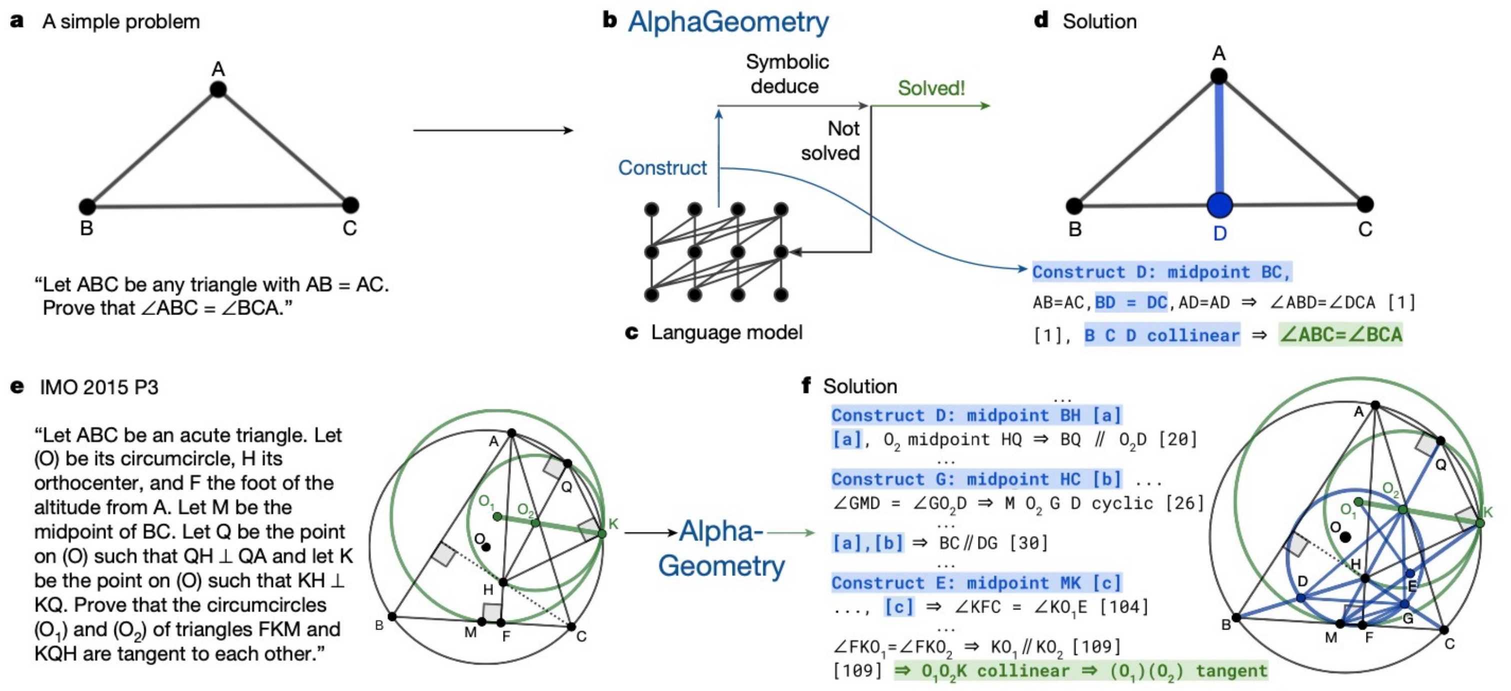


Writing Wikipedia articles from scratch (STORM)

Shao, Y., Jiang, Y., Kanell, T. A., Xu, P., Khattab, O., & Lam, M. S. (2024). Assisting in Writing Wikipedia-like Articles From Scratch with Large Language Models.

Compound AI Systems

Compound AI systems combine AI models with external tools to solve challenging tasks



Proving mathematical theorems (AlphaGeometry)

T. H. Trinh, Y. Wu, Q. V. Le, H. He, and T. Luong. Solving olympiad geometry without human demonstrations. Nature, 625(7995):476–482, 2024.

Compound AI Systems

Compound AI systems combine AI models with external tools to solve challenging tasks

LangChain
SGL
Nexusflow
DSPy
LlamaIndex
LMQL
Langdock
AUTOGPT by Mindstream
FlowiseAI

Many frameworks and DSLs exist to build compound AI systems

Compound AI Systems

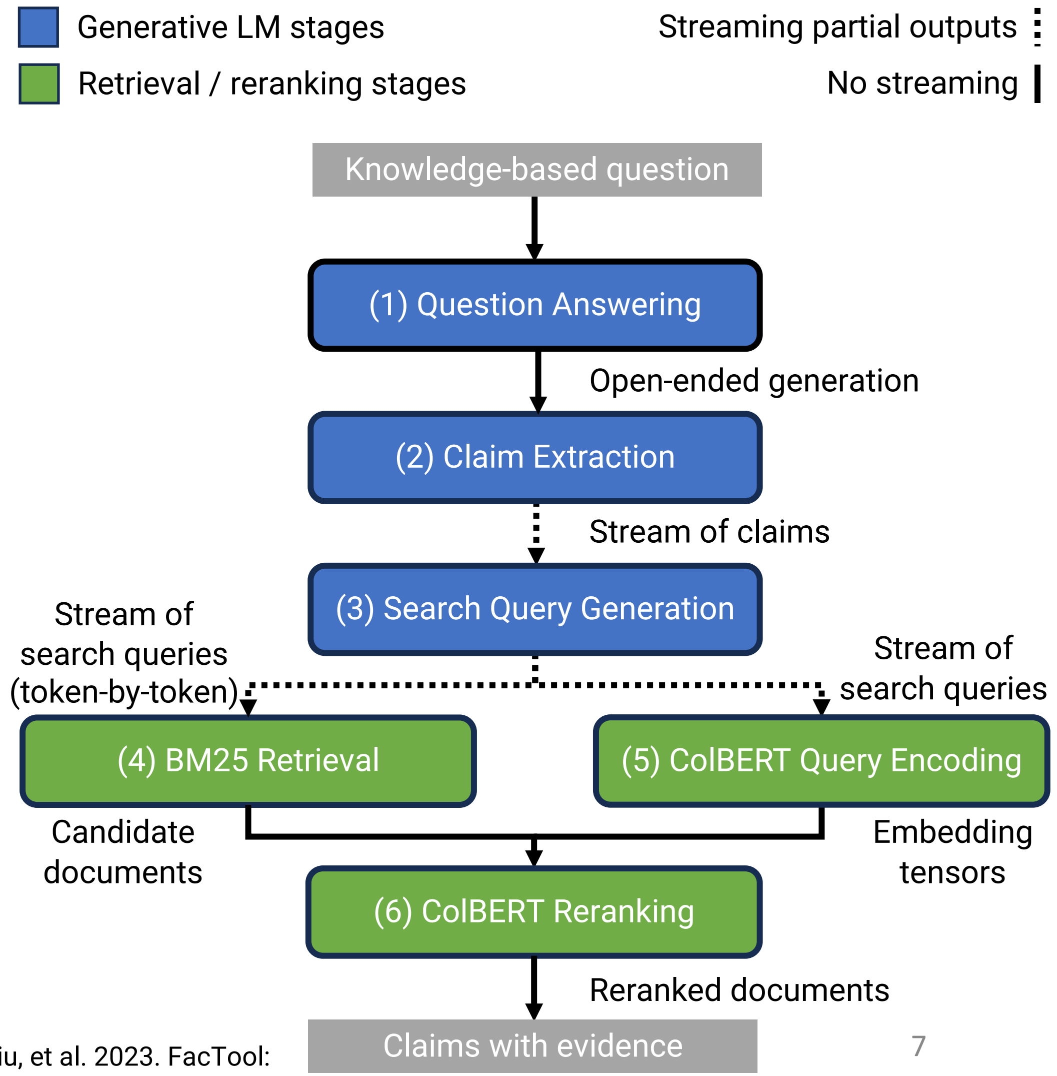
Compound AI systems combine AI models with external tools to solve challenging tasks

How do we serve compound AI systems efficiently at scale?

Streaming partial outputs

- Key optimization: *Streaming* partial outputs between pipeline stages
  - Partial outputs are segments of text such as words, sentences, or paragraphs
  - Generative LMs output text incrementally so we can emit these partial outputs as soon as they are generated
- FacTool[1] as a representative pipeline:
  1) Chat-bot answers question
  2) Extract factual claims
  3) Generate search queries for each claim

  4-6) Search for corroborating evidence

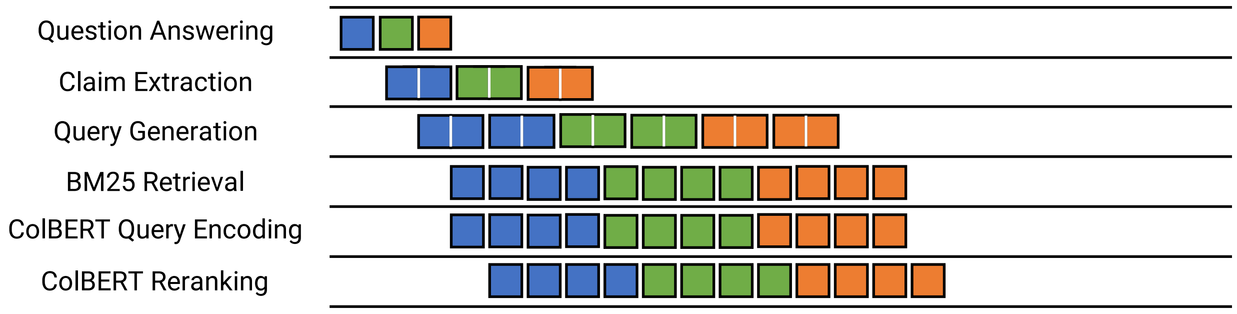


[1] I Chern, Steffi Chern, Shiqi Chen, Weizhe Yuan, Kehua Feng, Chunting Zhou, Junxian He, Graham Neubig, Pengfei Liu, et al. 2023. FacTool: Factuality Detection in Generative AI–A Tool Augmented Framework for Multi-Task and Multi-Domain Scenarios.

Streaming partial outputs

- We can pipeline *across* requests by treating each stage as a microservice
- Streaming partial outputs enables pipelining *within* a single request
- This reduces per-request latency and enables higher serving throughput
- Our prototype system **ALTO** enables partial output streaming

Compound AI Systems Today

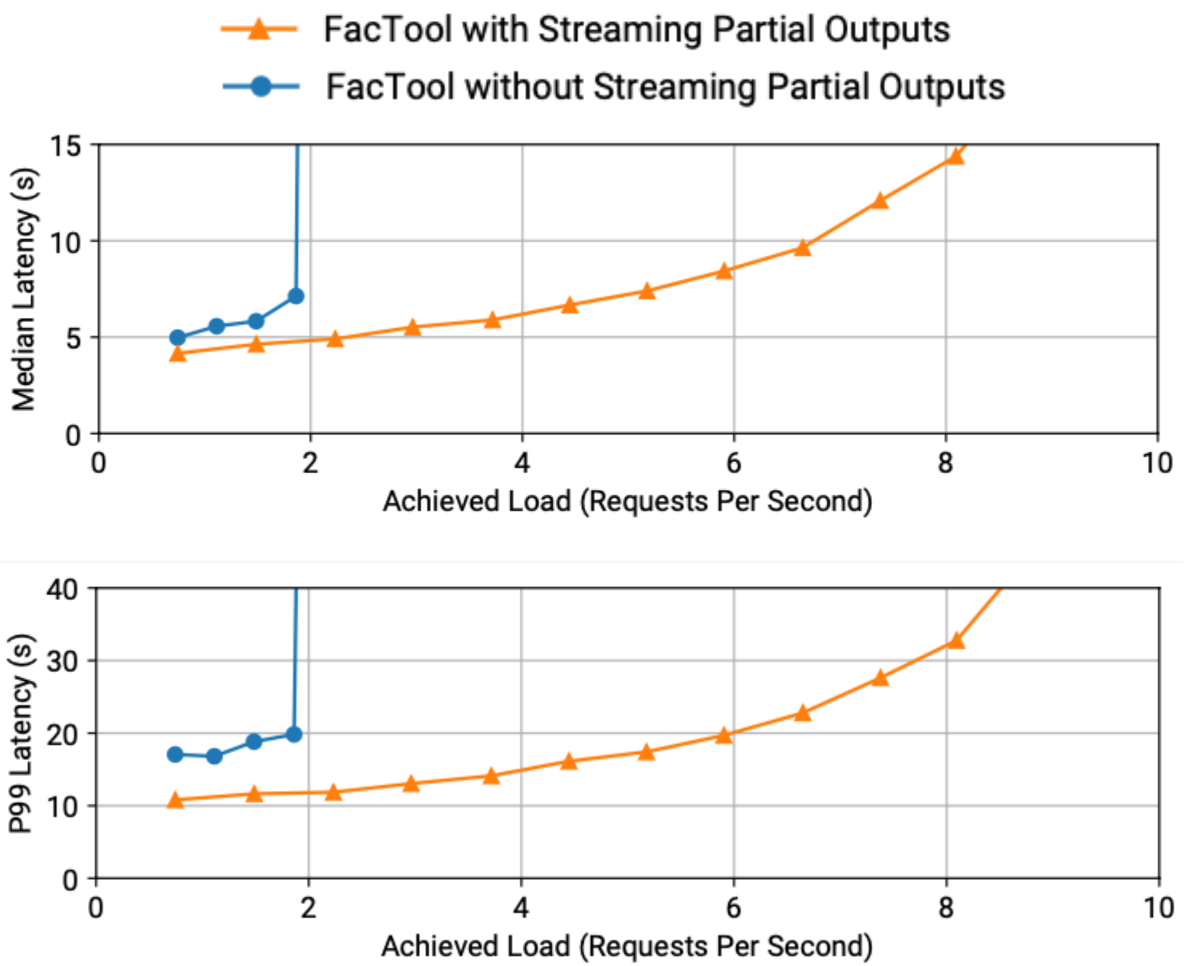


Partial Output Streaming

Empirical evaluation using FacTool

- We vary the input load (requests / second) and measure the resulting per-request latency
- We compare streaming partial outputs (orange) vs fully materializing LM outputs (blue)
- Streaming partial outputs enables up to **3x** higher serving throughput* while reducing tail latency by **1.8x**

*For a fixed latency target of 4 seconds / request

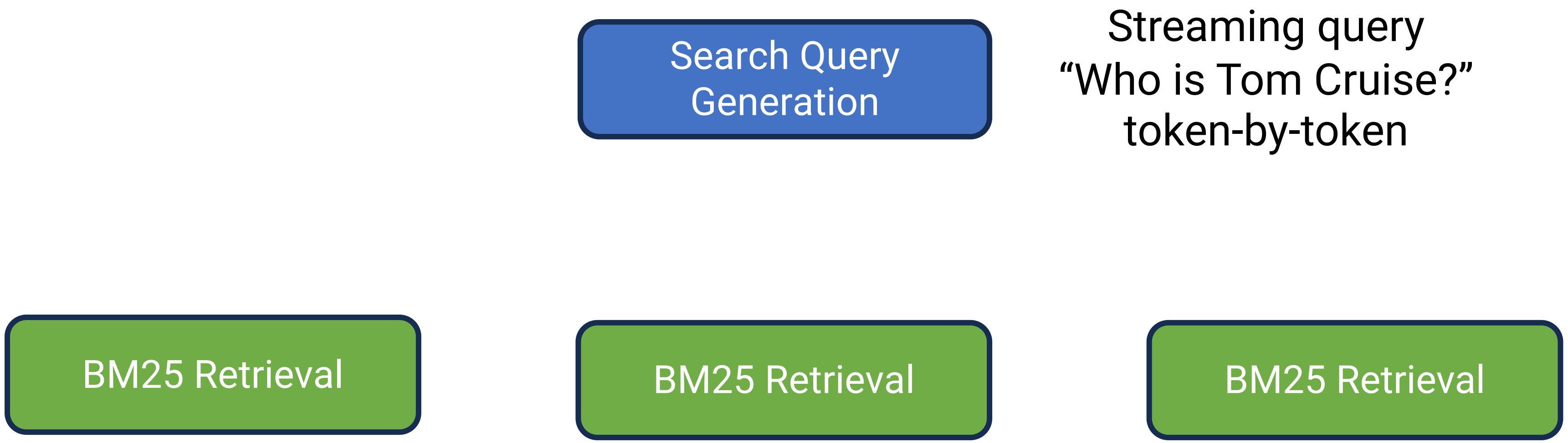


ALTO: **A**utomatic **L**anguage **T**oken **O**rchestrator

- **ALTO**: a serving system for automatically distributing and parallelizing compound, streaming AI pipelines
- Key challenges when streaming partial outputs:
  - **Correctness**: How do we segment partial outputs automatically and route them through the correct pipeline stage instances?
  - **Efficient load balancing**: Given a pipeline with heterogeneous prompts, how should we dynamically schedule prompts across instances?
- We discuss the need for **aggregation constraints** and **distributed prompt-aware scheduling** to address these challenges

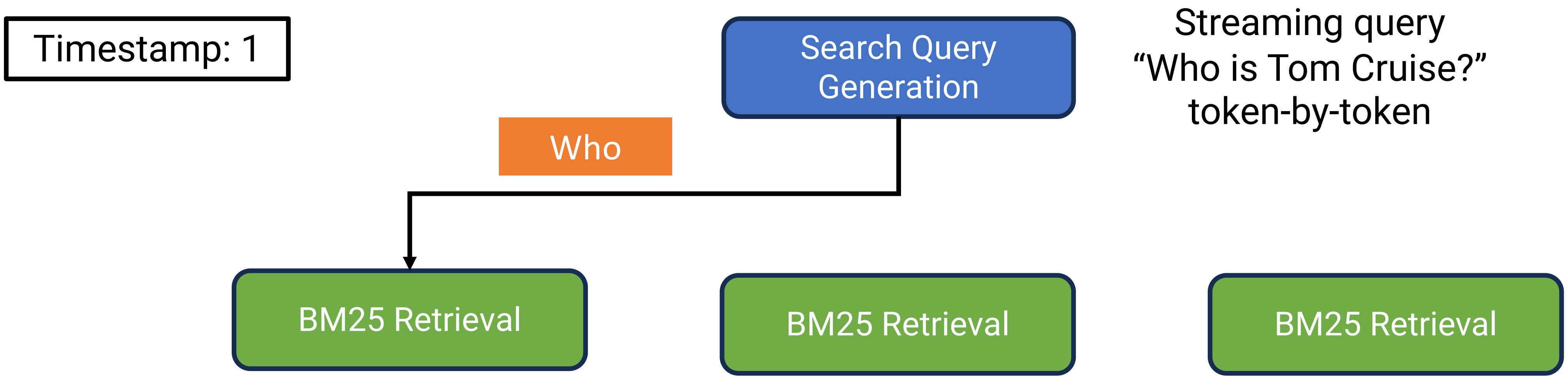
Stateful pipeline stages

- Some pipeline stages are **stateful**, which means all partial outputs routed through this stage must follow a consistent path



Stateful pipeline stages

- Some pipeline stages are **stateful**, which means all partial outputs routed through this stage must follow a consistent path

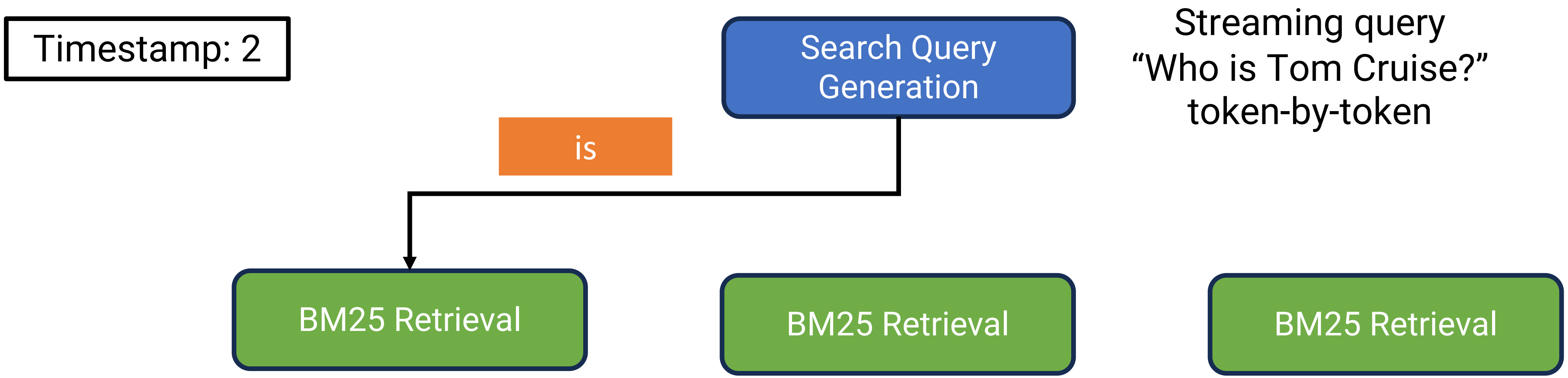


BM25 stage is **stateful**, so every token in this query’s stream must be routed through the same instance

We call this an **aggregation constraint**

Stateful pipeline stages

- Some pipeline stages are **stateful**, which means all partial outputs routed through this stage must follow a consistent path

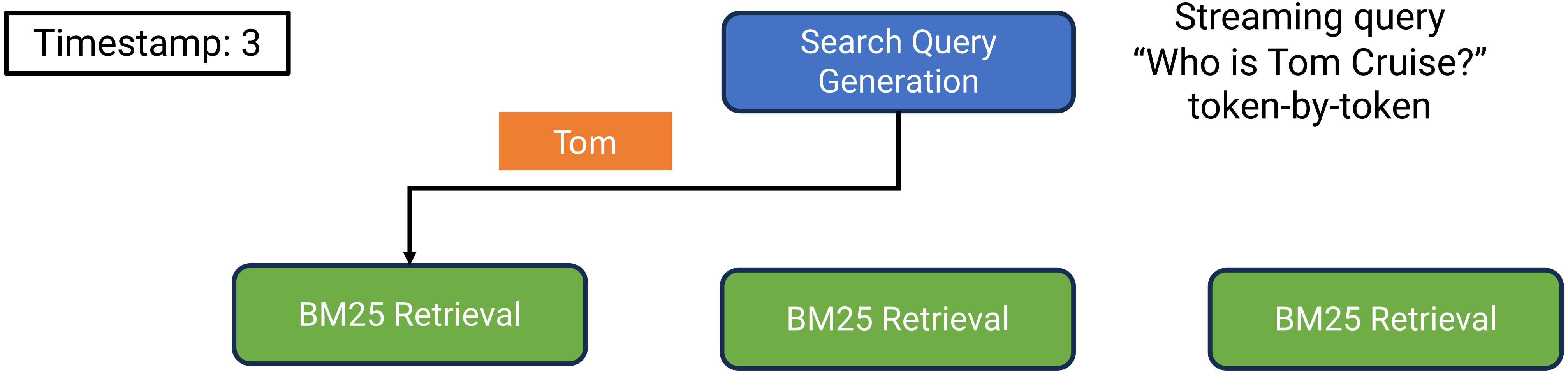


BM25 stage is **stateful**, so every token in this query’s stream must be routed through the same instance

We call this an **aggregation constraint**

Stateful pipeline stages

- Some pipeline stages are **stateful**, which means all partial outputs routed through this stage must follow a consistent path



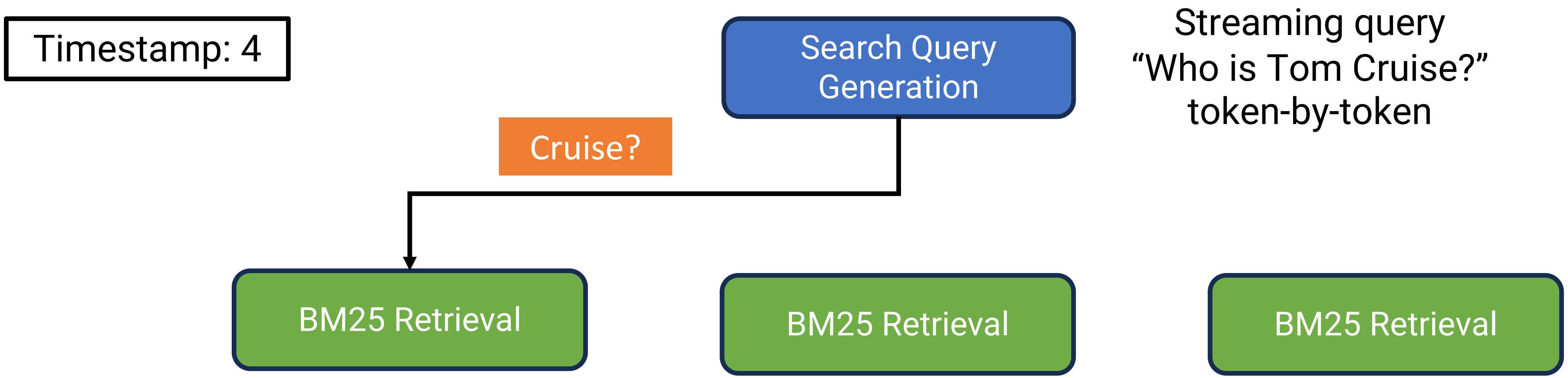
Streaming query
“Who is Tom Cruise?”
token-by-token

BM25 stage is **stateful**, so every token in this query’s stream must be routed through the same instance

We call this an **aggregation constraint**

Stateful pipeline stages

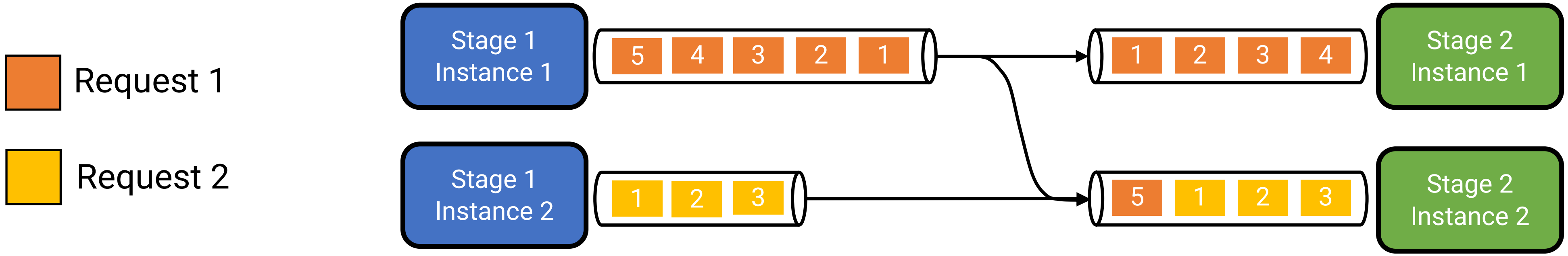
- Some pipeline stages are **stateful**, which means all partial outputs routed through this stage must follow a consistent path



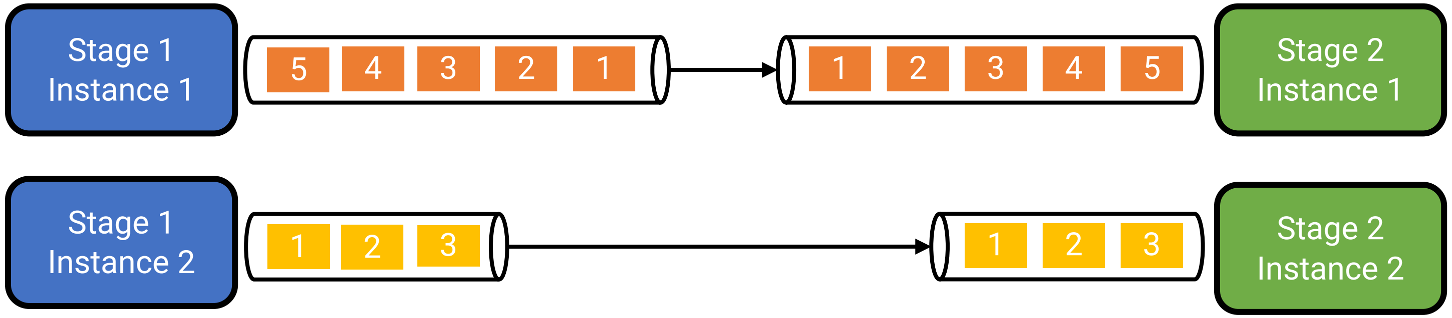
BM25 stage is **stateful**, so every token in this query's stream must be routed through the same instance

We call this an **aggregation constraint**

Aggregation constraints



Load balancing without an aggregation constraint



Load balancing with an aggregation constraint

Enforcing aggregation constraints can limit load balancing efficiency

Interface for aggregation-aware routing

- Currently in ALTO we provide this interface to specify aggregation constraints:

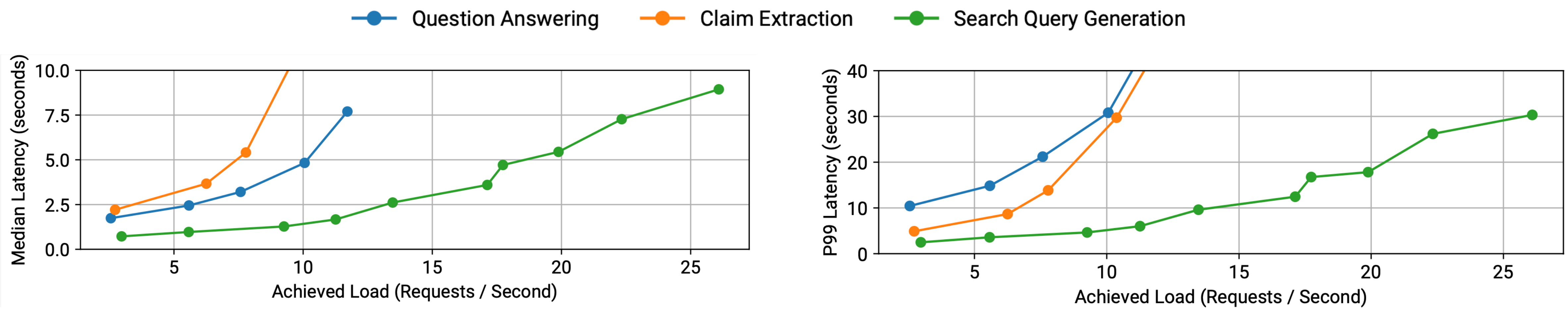
```
write(
    queue="bm25", obj=Token(...), id=obj_id,
     constraints=[obj_id, claim_id, query_id]
)
```

- We can enforce a consistent path by taking `hash(constraints) % N` (# of downstream stage instances)
- This requires explicitly encoding hierarchical ids in the object definitions, but we are working on a design for automating this

Prompt heterogeneity

- Prompts can vary significantly in their output type, frequency, and output volume
- We can observe this empirically within the FacTool pipeline

Stage	Overall Count	Per-output Quantum	Average # Outputs	Average Length / Output (words)	Average Time / Output (ms)
Question Answering	10795	Response (paragraphs)	-	62.5 ± 57.2	1292.3 ± 1175.0
Claim Extraction	10795	Claim	3.3 ±1.8	9.8 ± 3.5	403.6 ± 1175.0
Search Query Generation	35516	Search query	2.5 ± 1.3	5.5 ± 3.1	326.6 ± 252.4
		Search query token	5.5 ± 3.1	-	59.8 ± 79.3



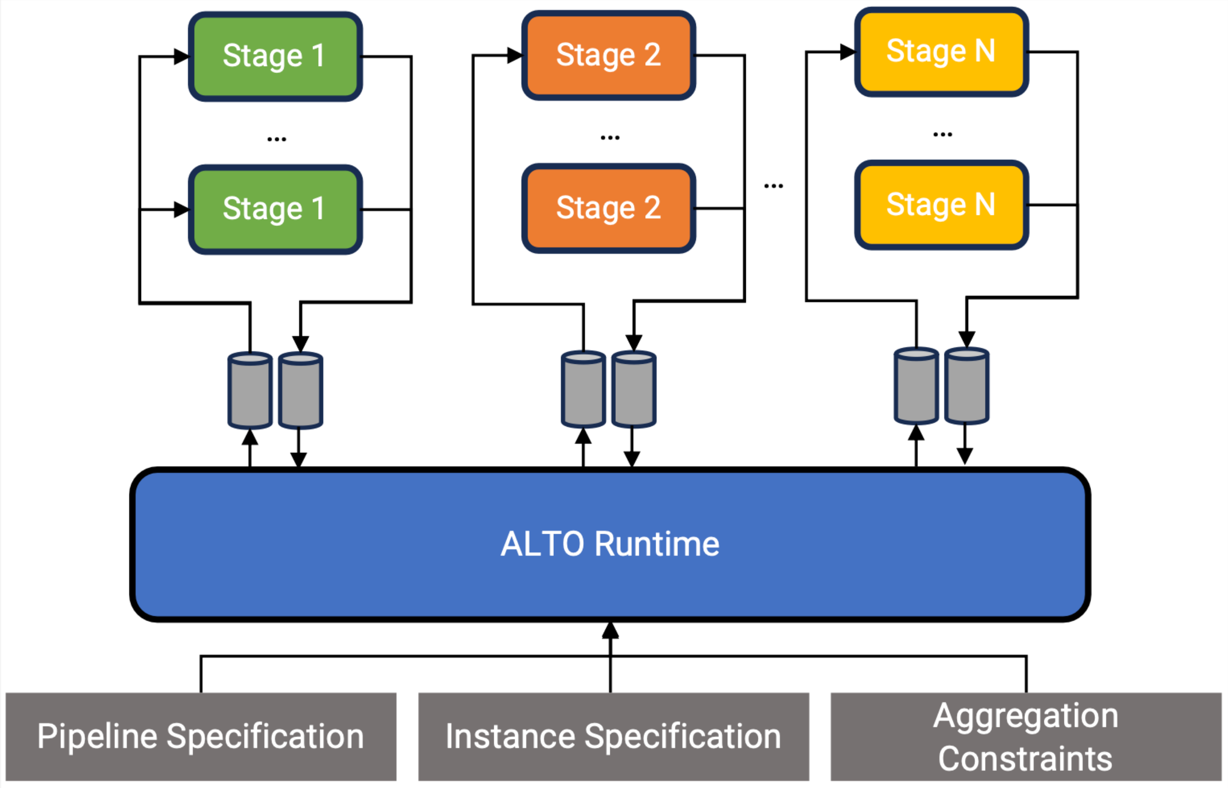
Distributed Prompt-aware Scheduling

- Many LM serving engines (e.g. vLLM, SGLang Runtime, Hydragen) benefit from serving the same prompts repeatedly on a single GPU by re-using prompt prefixes in the KV cache
- In a distributed setting we want to balance (1) dynamically load balancing prompts across stage instances with (2) maximizing prefix locality – these goals may be in conflict!
- This requires **distributed prompt-aware scheduling**
- ALTO does not currently implement distributed prompt-aware scheduling, but see our paper for some initial design ideas

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In Proceedings of the 29th Symposium on Operating Systems Principles. 611–626.
Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Jeff Huang, Chuyue Sun, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. 2023. Efficiently Programming Large Language Models using SGLang. arXiv preprint arXiv:2312.07104 (2023).
Jordan Juravsky, Bradley Brown, Ryan Ehrlich, Daniel Y. Fu, Christo- pher Ré, and Azalia Mirhoseini. 2024. Hydragen: High-Throughput LLM Inference with Shared Prefixes. arXiv:2402.05099 [cs.LG]

ALTO Implementation

- ALTO is implemented using an asynchronous queue interface over UNIX domain sockets
- Applications (written in Python) send Protobuf messages to a centralized runtime (written in Rust) which routes messages to downstream stage instances
- We are also working on a Ray-based implementation



Conclusion

- We can optimize compound AI system serving by **streaming partial outputs** between pipeline stages
- Our prototype system ALTO demonstrates this on the FacTool pipeline by improving throughput by up to **3x** while reducing tail latency by **1.8x**
- Streaming partial outputs introduces the new challenges of correctness and efficient load balancing, which require **aggregation constraints** and **distributed prompt-aware scheduling** to solve

keshav2@stanford.edu https://cs.stanford.edu/~keshav2