Large Language Models as Realistic Microservice Trace Generators

Donghyun Kim[1], Sriram Ravula[1], Taemin Ha[1], Alexandros G. Dimakis[2,3], Daehyeok Kim[1], Aditya Akella[1]

[1] University of Texas at Austin
[2] University of California, Berkeley
[3] BespokeLabs.ai

Why do we need synthetic workload traces?

Workload traces are essential for understanding computer system behavior and managing compute and memory resources.

Problem: Limited availability of systems data

- Limited public data availability
- Trace collection and maintenance overhead

Existing approaches do not work!

Packet trace generation with GAN models [NetShare]

- Inflexible to handle traces with variable structures (e.g., variable depths)
- Not learning structural properties (e.g., call graph hierarchy)

Cloud VM workloads generation with LSTM [Bergsma et al.]

- Predict only specific attributes in traces
- Unable to handle large size traces (e.g., large call graph traces)

[NetShare] Practical GAN-based synthetic IP header trace generation using NetShare, SIGCOMM 2022
[Bergsma et al.] Generating Complex, Realistic Cloud Workloads using Recurrent Neural Networks, SOSP 2021

Opportunity: Large Language Models

Recent advances in LLMs allow:

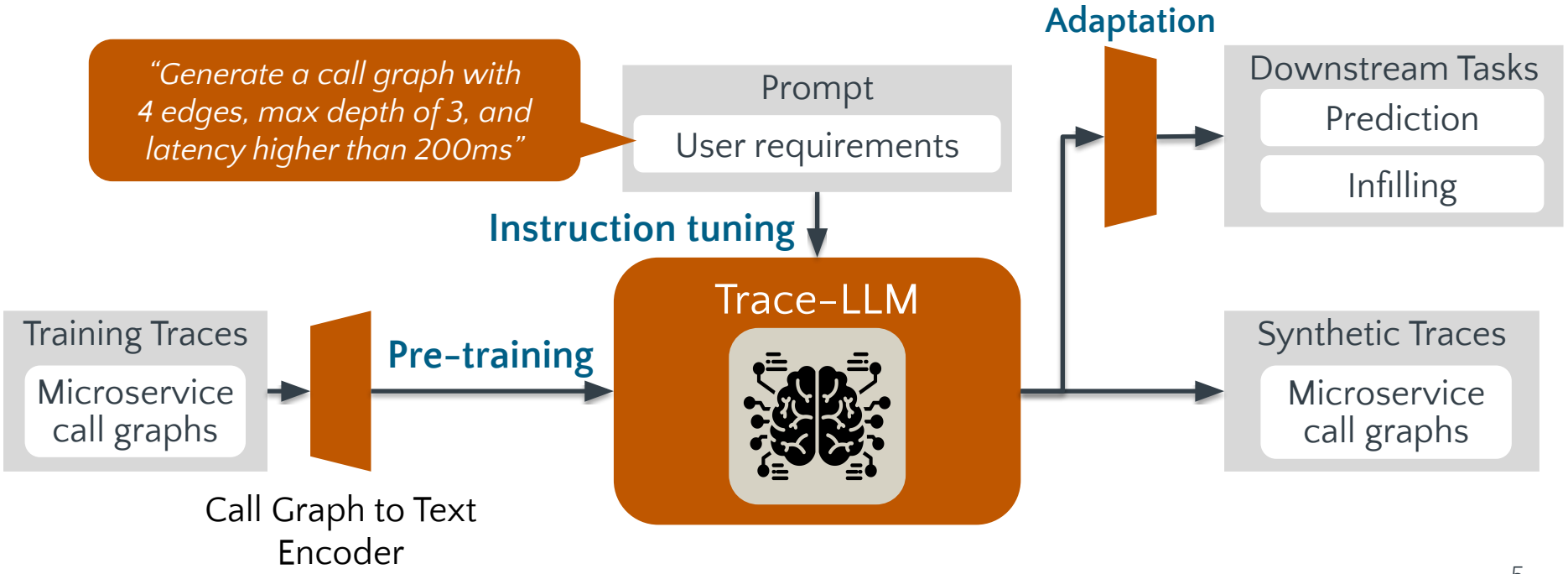
- Learning complex patterns in data from various domains
- Conditioning on constraints to guide generation toward desired behaviors

Large Language Models (LLMs) offer new ways to understand and generate complex systems data.

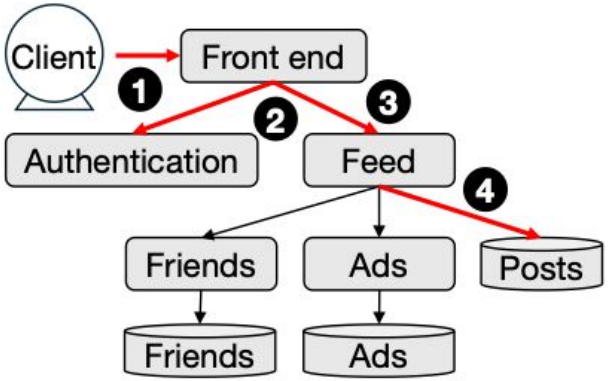
Our work: Trace-LLM

Can we synthesize realistic system data based on existing ones with LLMs?

- We use microservice call graph traces as a starting point



What do microservice call graphs look like?



	EDGE ID	Src.	Dest.	Type	Start (ms)	Finish (ms)
❶	0	Client	Front end	HTTP	0	24
❷	0.1	Front end	Authentication	RPC	0	1
❸	0.2	Front end	Feed	RPC	1	23
❹	0.2.1	Feed	Posts	DB	6	15

- Microservice traces consist of sequence of communications (❶--❹)
- Call graphs have structural properties

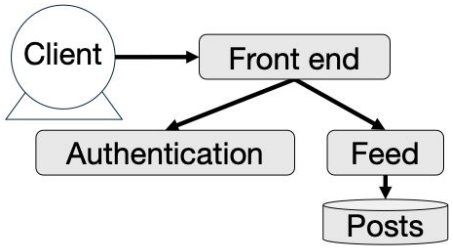
Requirements for synthesized traces

1. Validity
   - Traces must meet structural constraints
   - E.g., Parent node's destination must match child's source

2. Similarity in distribution
   - They should follow distribution of certain properties of existing one
   - E.g., distribution of popular microservices, communication types, …

Pre-training an existing LLM

Pre-train an LLM (e.g., Llama-2) to teach call graph structures and distribution



[GENERATE GRAPH] id:Social Network/num_edges:4/max_depth:3/latency:24

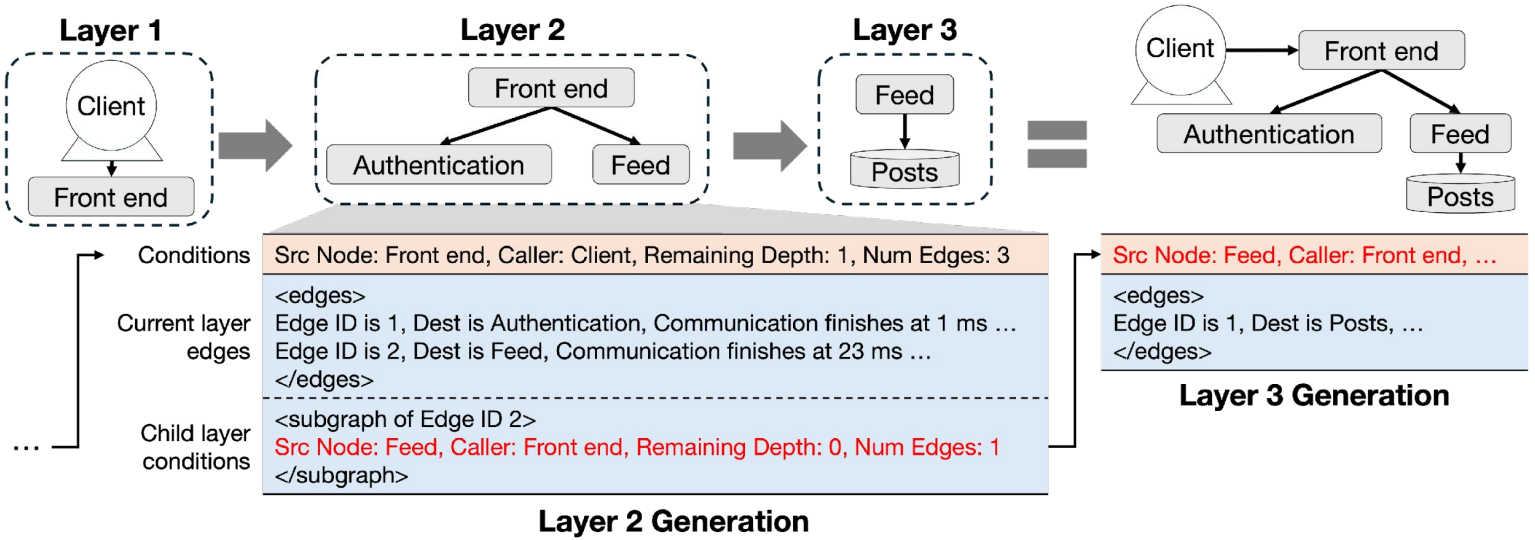
<edges>
(id is 0, src is Client, dest is Front end, type is http, start at 0 ms, finish at 24 ms)
(id is 0.1, src is Front end, dest is Authentication, type is rpc, start at 0 ms, finish at 1 ms)
(id is 0.2, src is Front end, dest is Feed, type is rpc, start at 1 ms, finish at 23 ms)
(id is 0.2.1, src is Feed, dest is Posts, type is db, start at 6 ms, finish at 15 ms)
</edges>

Recursive generation of call graph layers

Problem: Learning all the intricacies in the traces is challenging for LLMs

Idea: Break a call graph generation into multiple easier steps

(1) Generate a layer at a time and (2) Recursively generate child layers



Conditioning through language descriptions

How to make the model generate call graphs following user instructions?

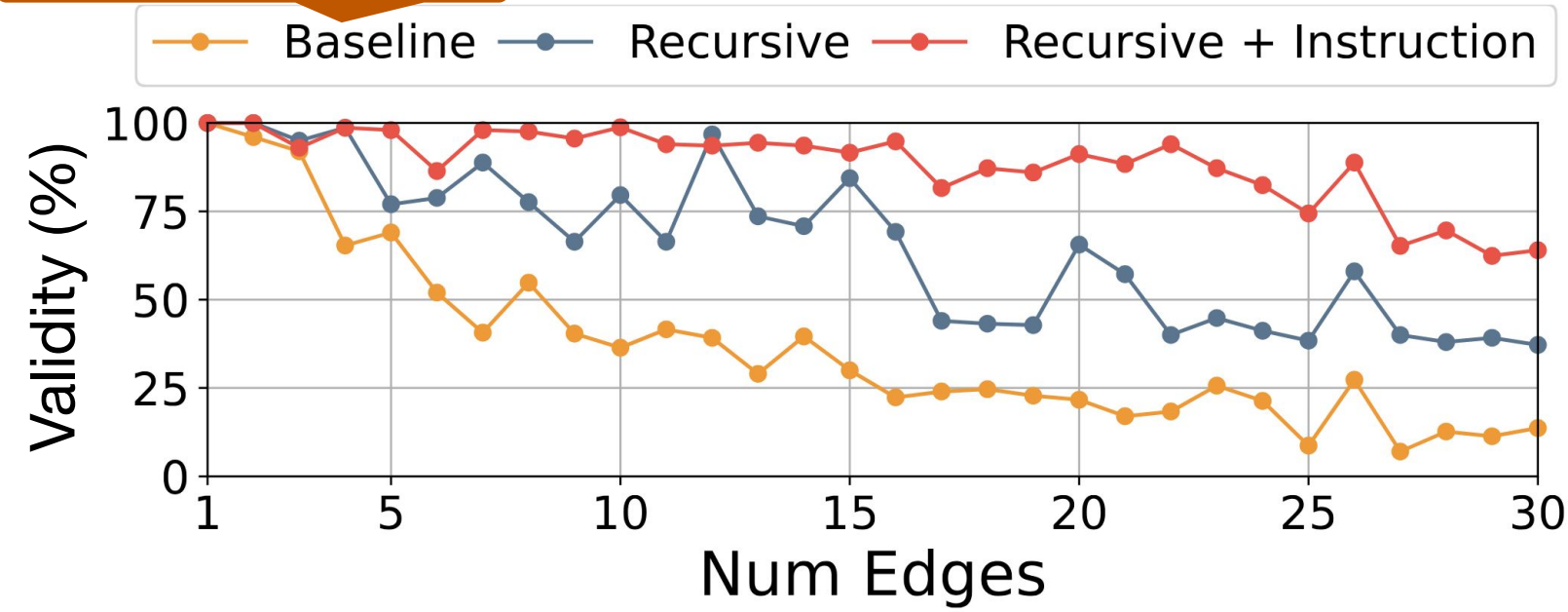
- E.g., "`Generate a call graph with latency higher than p90` "
- The model needs to learn the Instruction–call graph relationship!

Idea: Adding conditions to instructions during fine-tuning

Requirements:
start_communication_at:0/start_node:USER/remaining_depth:2/num_current_edges:1/num_edges:4/latency:12/id:S_032647104
Conditions:
In each edge, communication start time should NOT be greater than latency 12 milliseconds
Generate subgraph instructions if necessary
the first start_communication_at should be requirement's start_communication_at 0
Also, communication should finish before latency 12 milliseconds
copy caller from requirement's start_node:USER
generate 1 edges following num_current_edges
Build a call graph with high latency

Does Trace-LLM generate valid graphs?

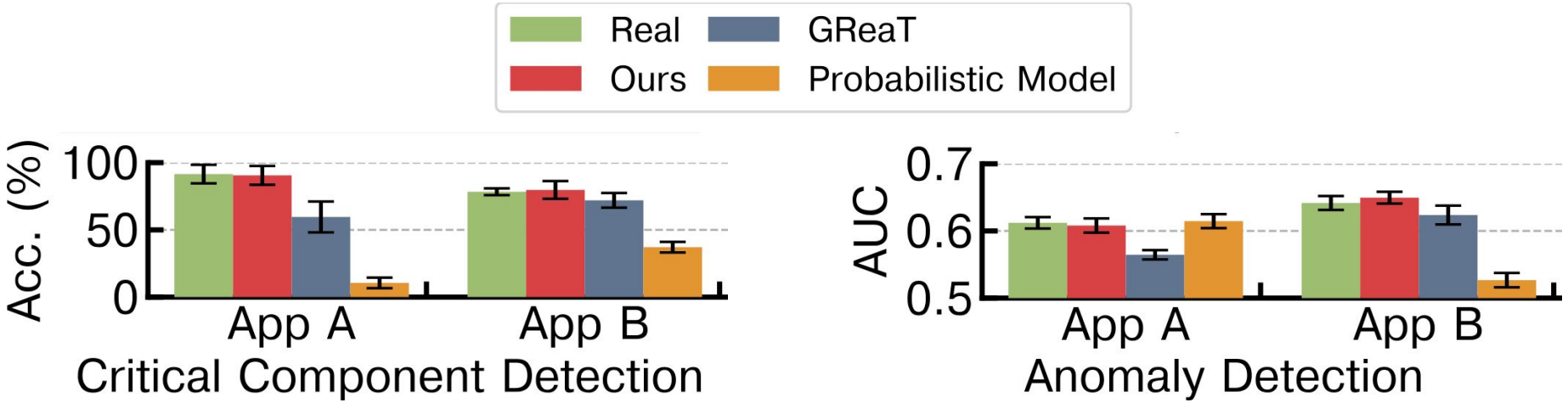
Llama-2 7B w/o pre-train



Similar results for different depth, temperature parameters and model sizes

Using Synthetic Traces for Downstream ML Tasks

TraceLLM performs comparably to real data on microservice management tasks.



More evaluation results (e.g., similarity, adaptation) available in the paper!

Conclusion

TraceLLM

- An LLM trained to generate **synthetic microservice workload traces**.
  - A **recursive generation framework** to model complex hierarchical call graph structures.
  - **Instruction tuning** to enforce structural constraints and simulate rare or extreme scenarios.

- Generated traces are realistic and valid, outperforming existing synthetic generation methods.
  - Synthetic traces can **replace real data** for key system management tasks and **adapt** to downstream applications (e.g., feature prediction, trace infilling).

Thank you!

Paper:

https://arxiv.org/pdf/2502.17439

Code:

https://github.com/ldos-project/tracellm

Research supported by NSF: