

ServeGen

Workload Characterization and Generation of Large Language Model Serving in Production

Yuxing Xiang^{1,2}, Xue Li², Kun Qian², Yan Zhang¹, Wenyuan Yu², Ennan Zhai², Xin Jin¹, Jingren Zhou²

¹ Peking University ² Alibaba Group



LLM Serving Workloads Drive System Design

The scale

- LLMs power a new generation of applications
- Production clusters serve **billions of requests** daily
- Diverse demand: language, **multimodal**, **reasoning**

Workloads matter

- Scheduling, auto-scaling, provisioning all depend on workload assumptions
- Workloads **motivate** new optimizations
- Workloads **benchmark** new systems

***If workload assumptions are wrong,
system designs will be wrong too.***

But how well do we actually understand LLM serving workloads?

The Status Quo: Limited Understanding

What exists

	BurstGPT	LMM
Duration	4 months	2 days
Models	2	—
Requests	5.29M	—
Scope	Lang only	Image only

What's missing

- Omnimodal & reasoning workloads: **uncharacterized**
- Intricate patterns (bursts, shifts): **underexplored**
- Workload generation: **unrealistic**

The de facto approach

Poisson arrivals + ShareGPT lengths — produces workloads that are **misleadingly easier** to serve

ServeGen: Closing the Gap

Characterization

Comprehensive analysis of **language**, **multimodal**, and **reasoning** workloads from Alibaba Cloud

3.54B requests · 12 models · 4 months

Key Insight

Per-client decomposition reveals a **causal model**: top clients steer complex workload patterns

ServeGen Framework

Open-source tool for generating **realistic** workloads via per-client composition

Table-of-Contents

I. Characterizing Language Workloads

Arrivals, data distribution, and the client decomposition

II. Emerging Workloads

What's unique about multimodal and reasoning models

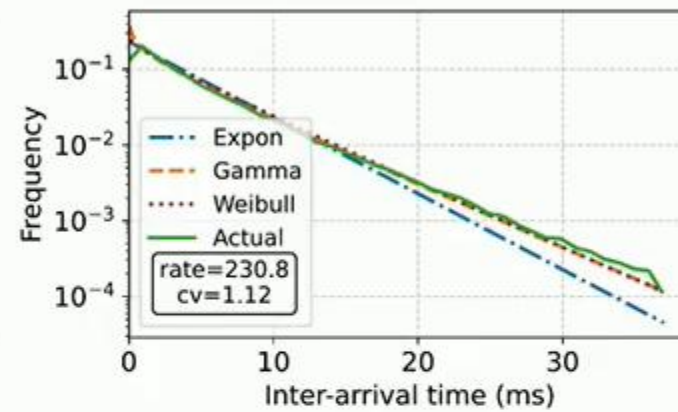
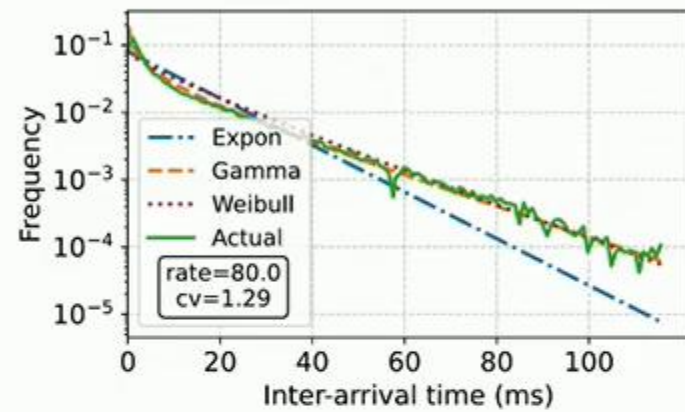
III. ServeGen Framework

Generating realistic workloads & Case studies

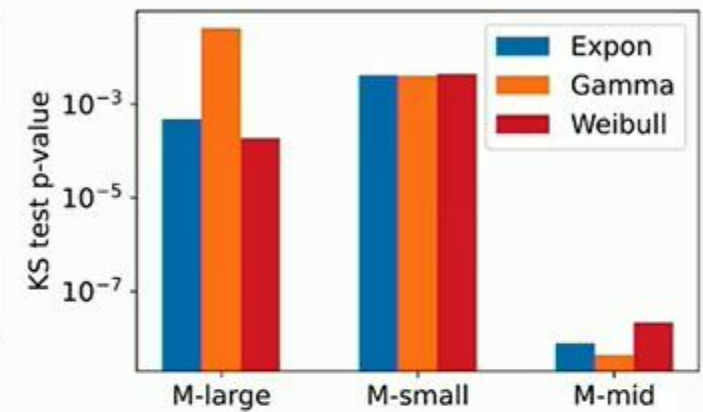
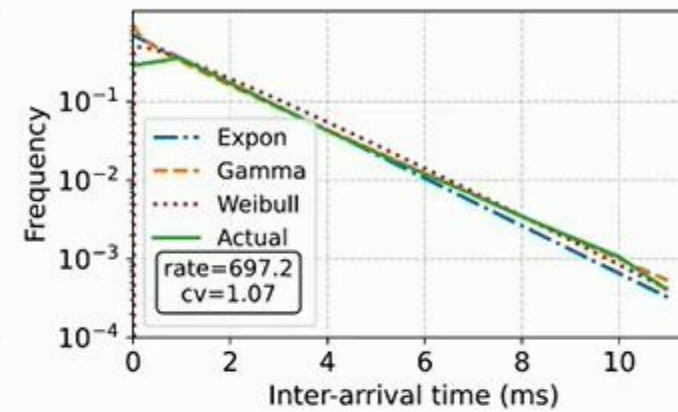
Part I: Characterizing Language Workloads

Arrivals, data distribution, and the client decomposition

Arrivals Are Bursty and Complex



Arrival Time Distribution (w/ fitting curves)

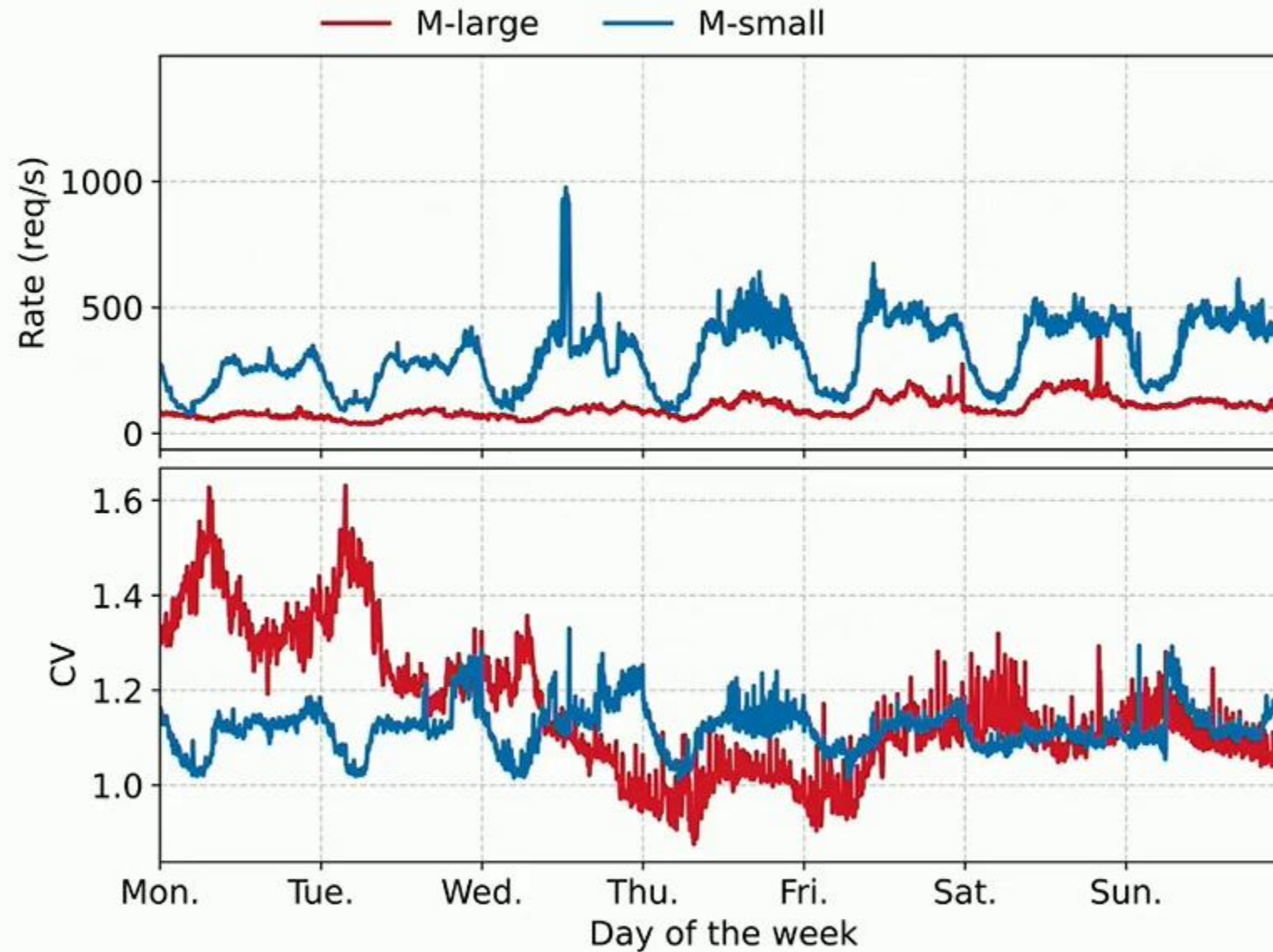


KS-Tests

- Arrivals are **bursty** ($CV > 1$)
- Arrivals are **complex**
 - No single distribution fits all workloads
 - Exponential / Gamma / Weibull / ...

- K-S tests confirm: **different models need different fits**

Burstiness Shifts Over Time



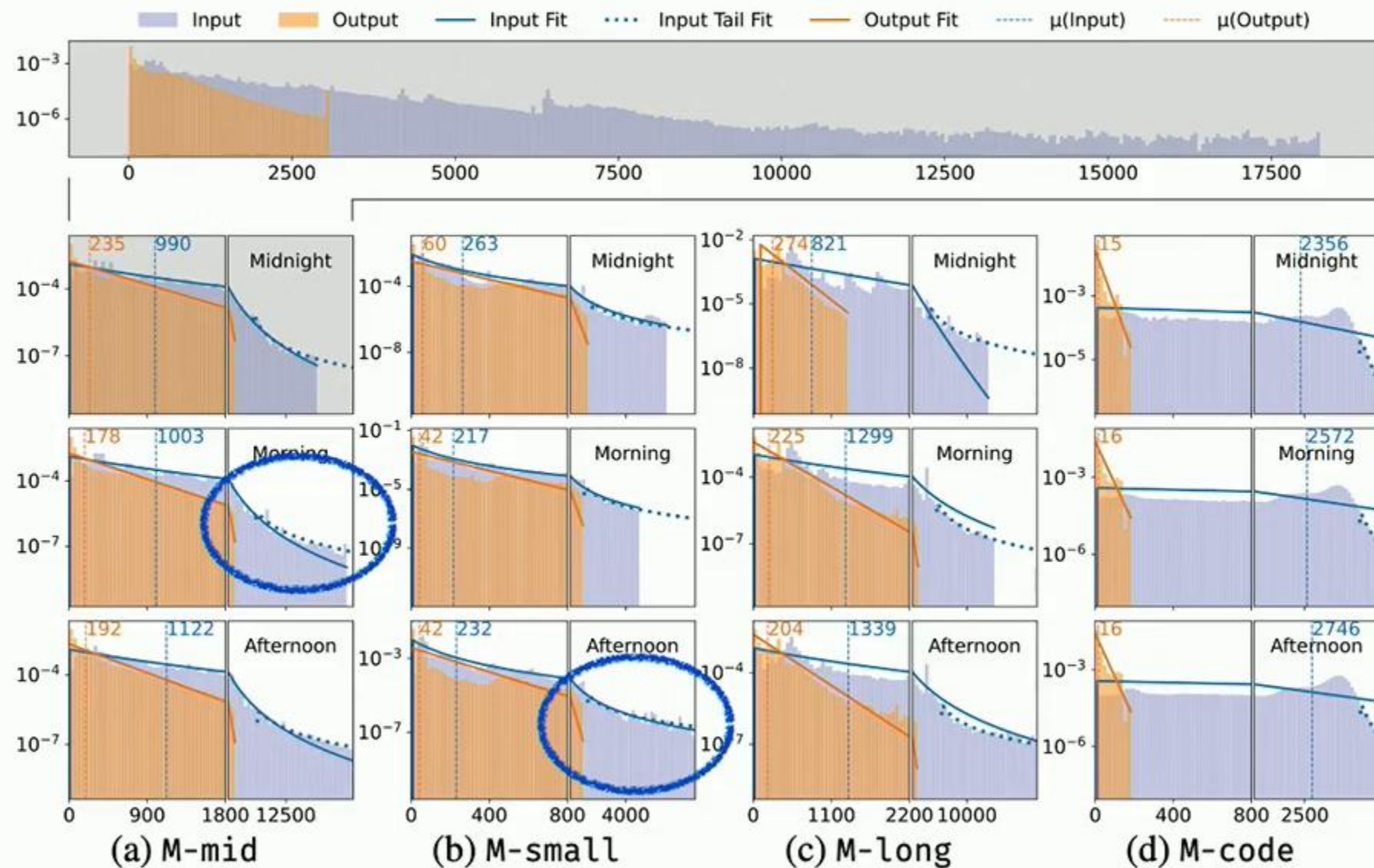
- M-large: continuously bursty Mon–Tue, then stable Thu–Fri
- Rate and burstiness shifts are **not** synchronized

Arrival Pattern

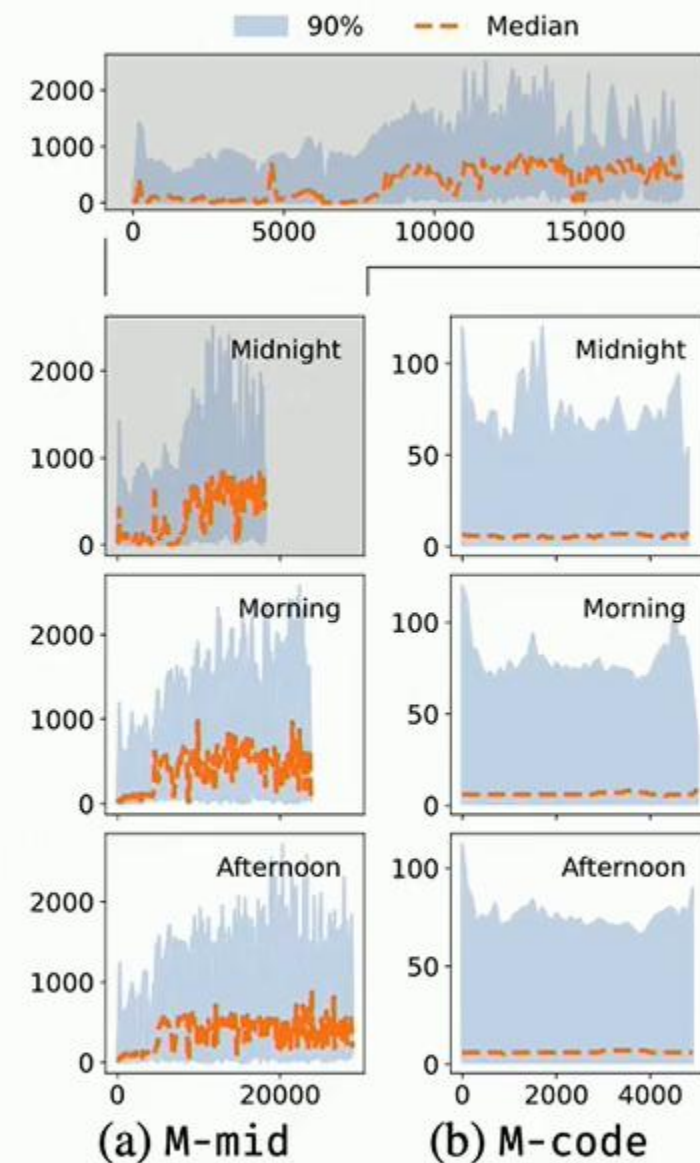
Systems Implication

Static workload assumptions fail — systems need **adaptive design** (e.g., scheduling and scaling) that respond to shifting patterns beyond rates.

Input/Output Length Distributions



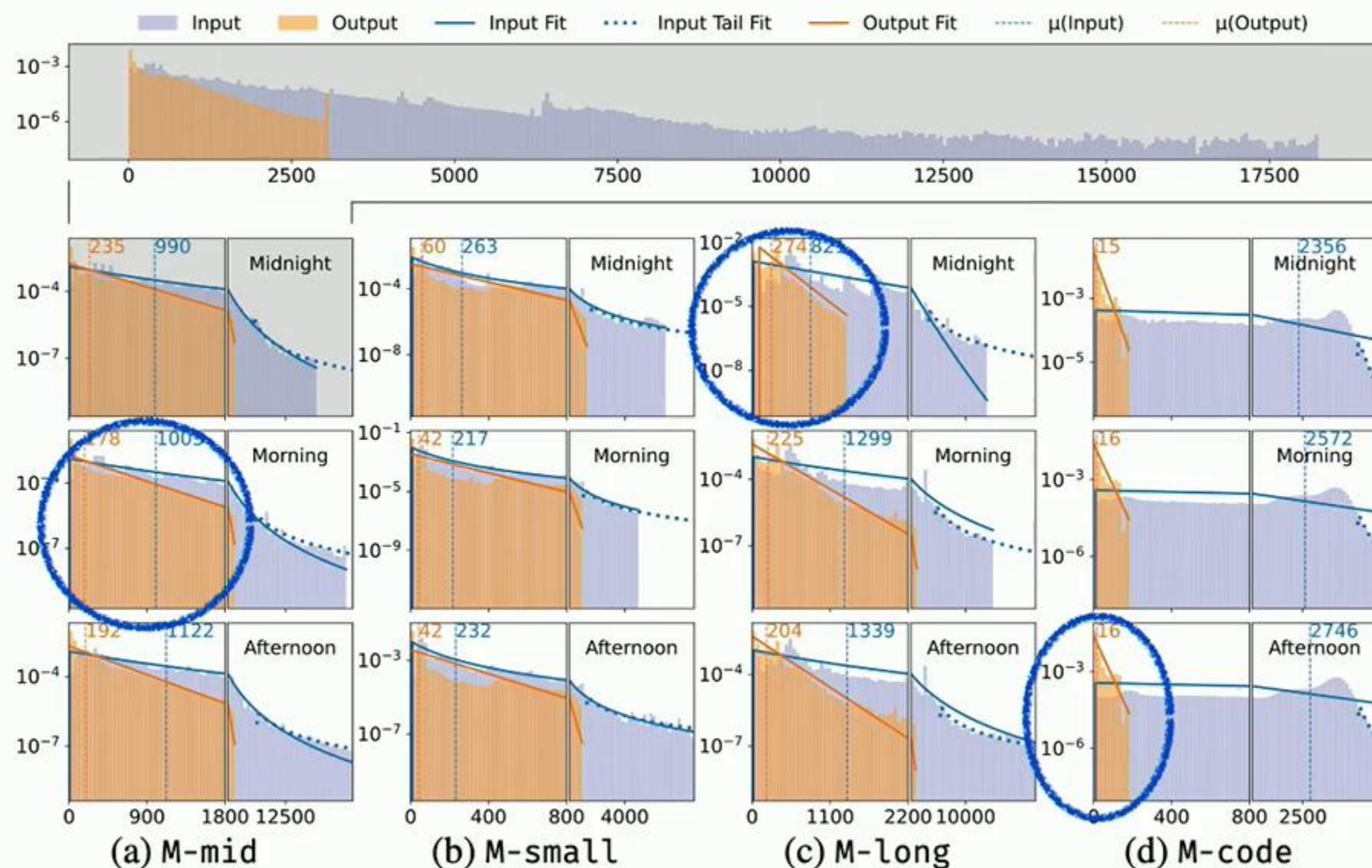
Length distribution across models



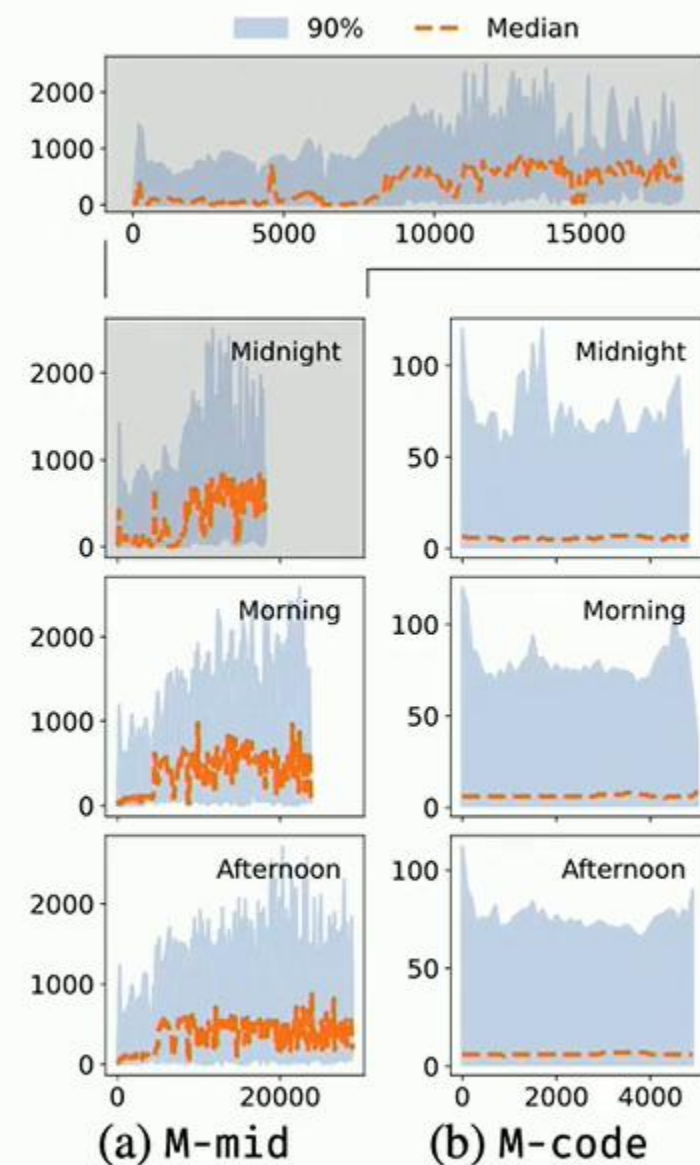
Length correlation

- Input lengths follow a **Pareto + Log-normal** mixture — heavy-tailed with long prompts
- Fit body and tail separately

Input/Output Length Distributions



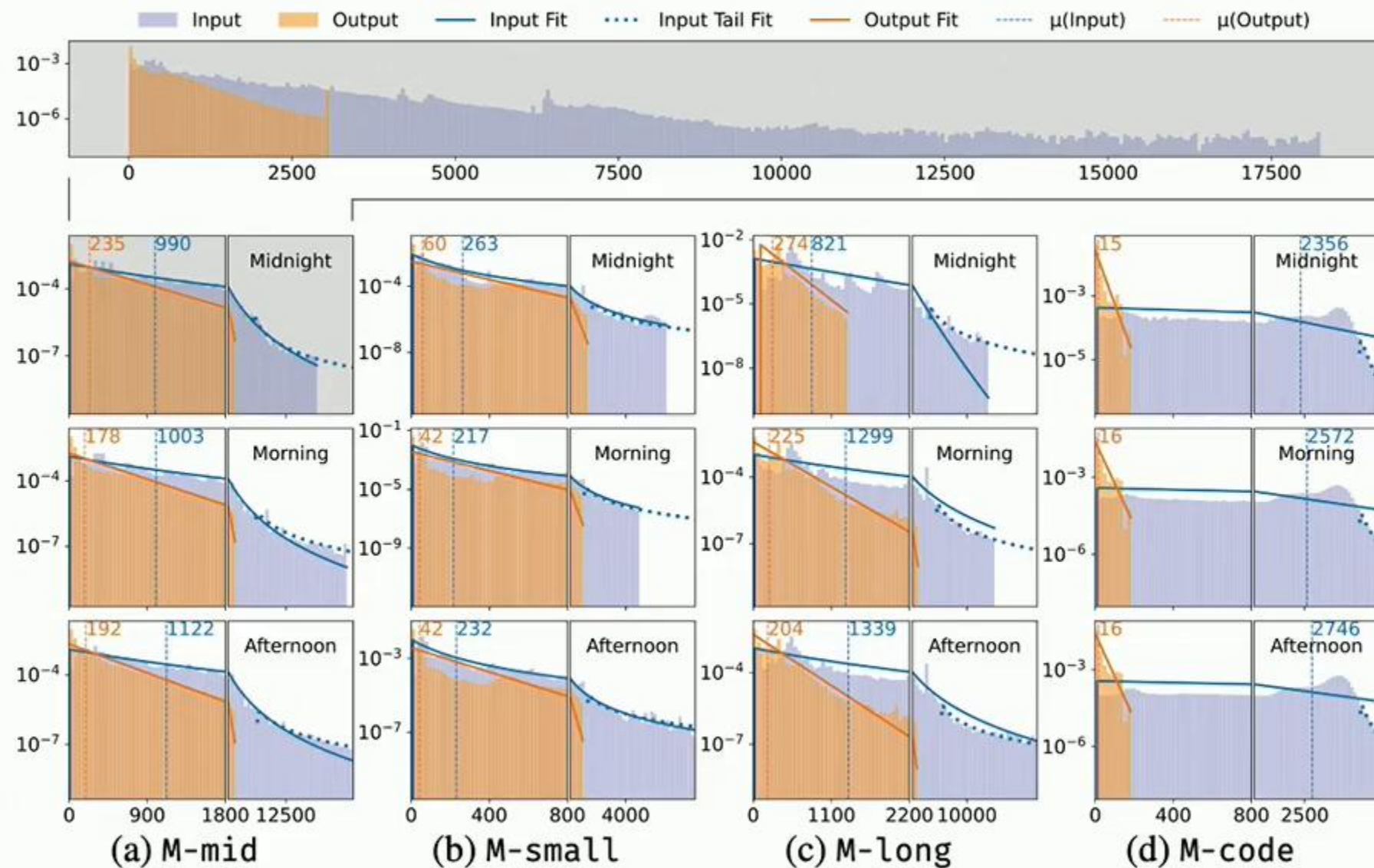
Length distribution across models



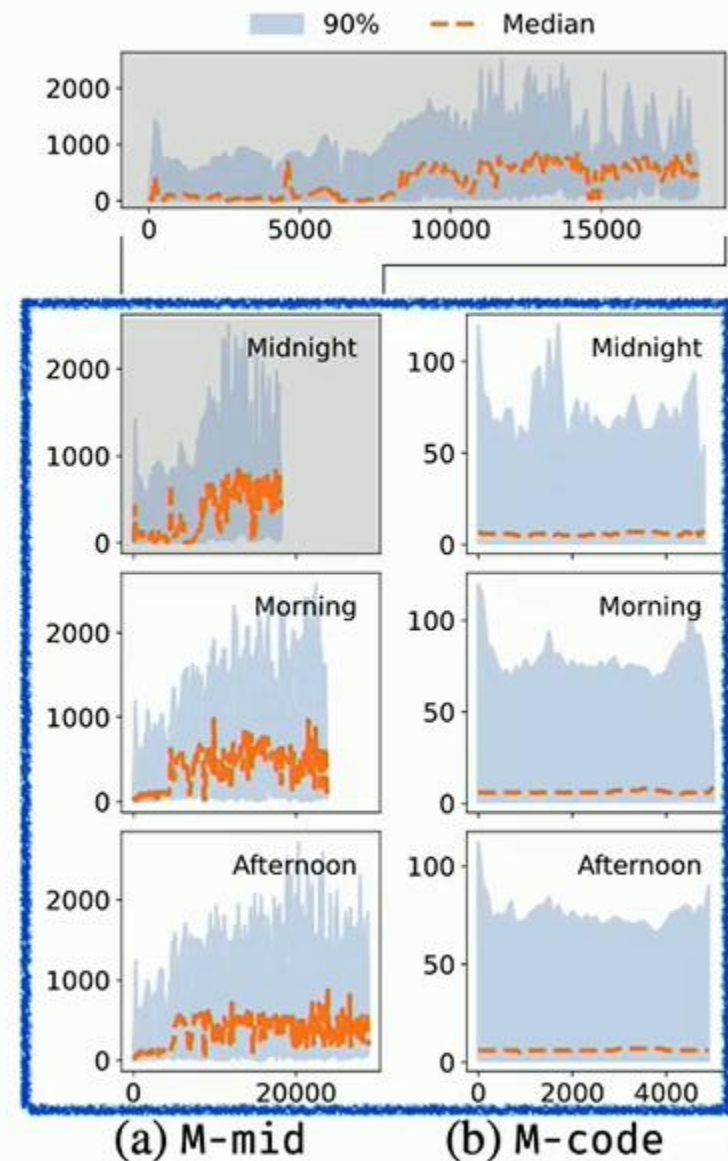
Length correlation

- Output lengths are surprisingly **Exponential** — *memoryless*
- Except M-small, where output is more structured (template-driven)

Input/Output Length Correlation



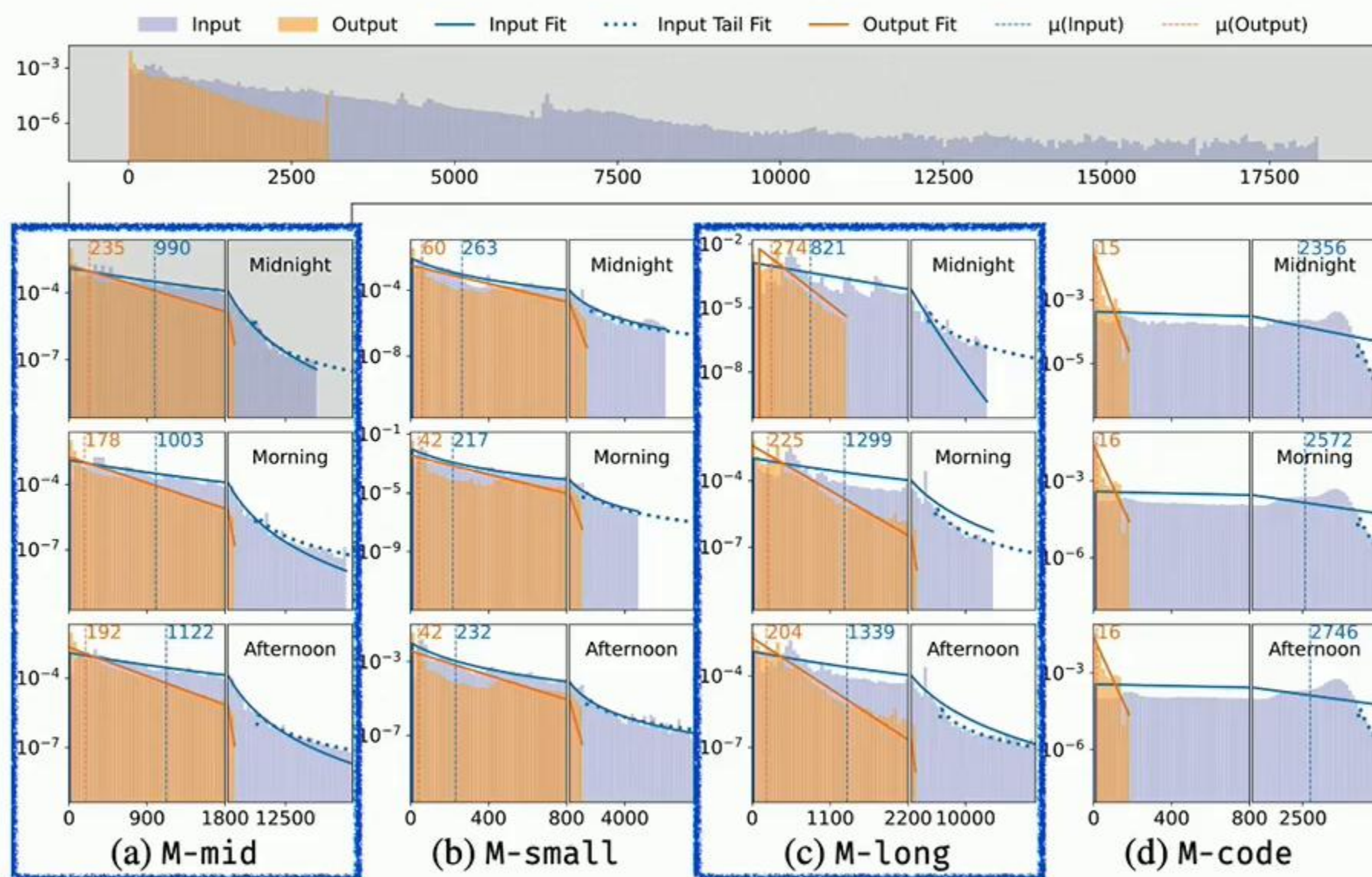
Length distribution across models



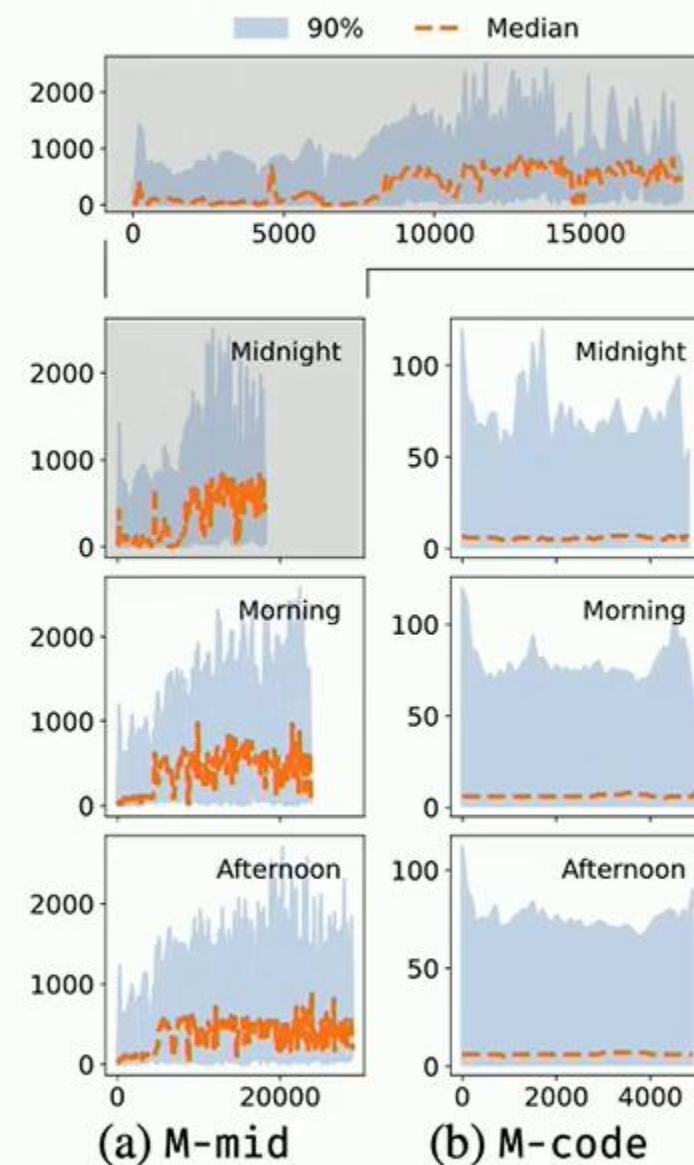
Length correlation

- Input and output lengths show **weak correlation**
- Predicting request lengths can be difficult in general

Length Distributions Shift Over Time



Length distribution across models



Length correlation

- Distributions shift **up to 1.63×** for input, **1.46×** for output
- Input and output shifts are **independent** — not synchronized

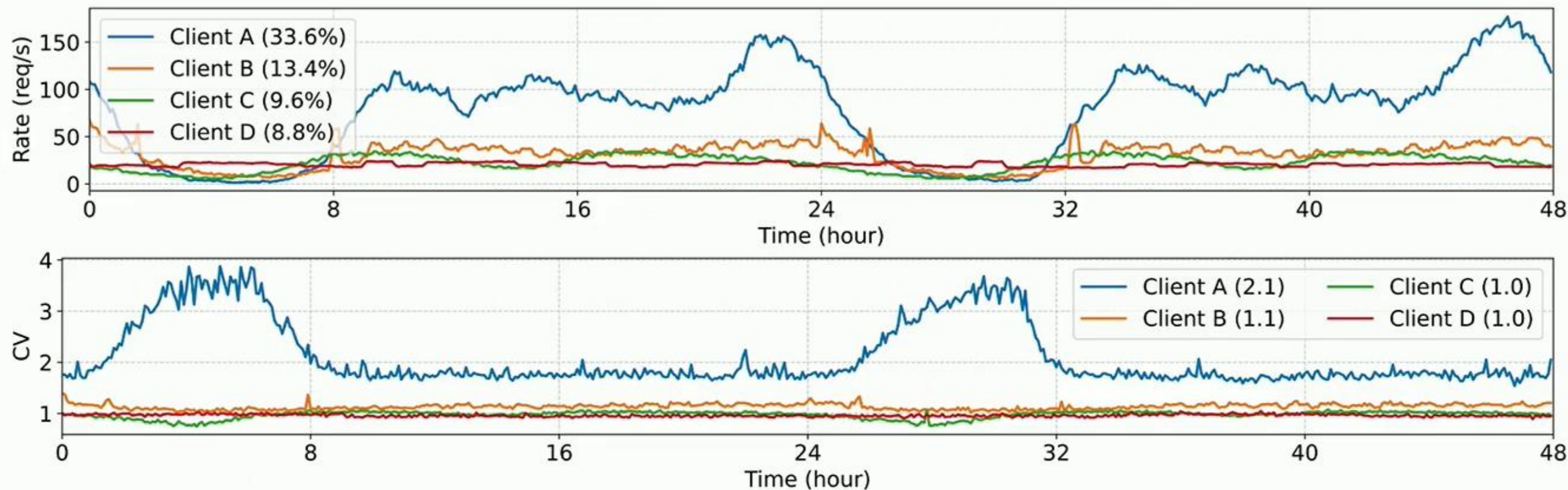
Length Distributions

Systems Implication

Input/output distributions are unique and shift over time — **dynamic prefill/decoding load patterns** that require fine-grained management

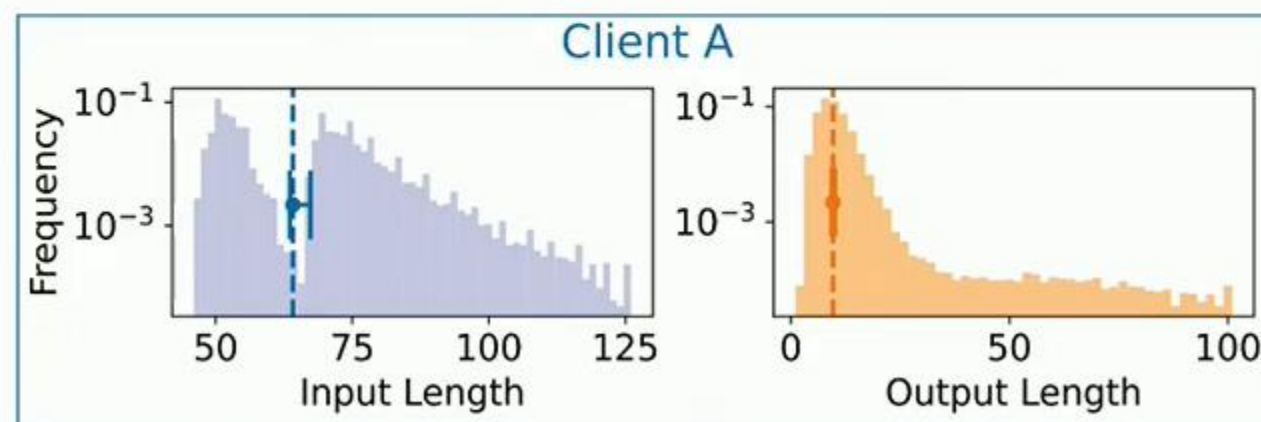
Is the conclusion just "it's complex"?

The Key Insight: Client Decomposition

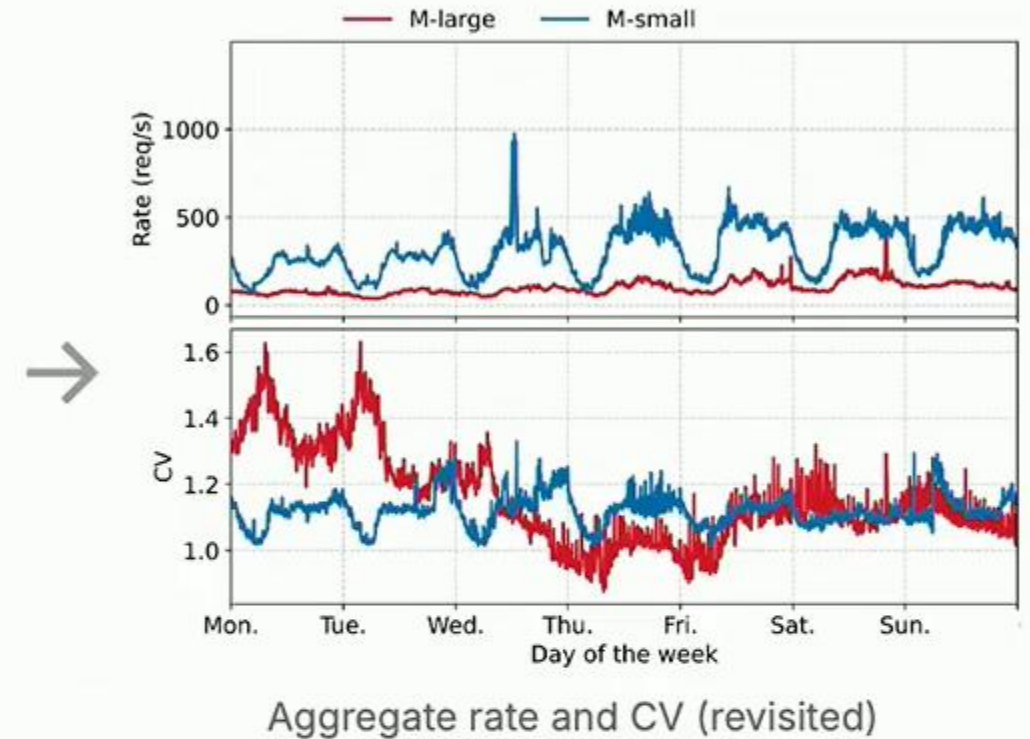
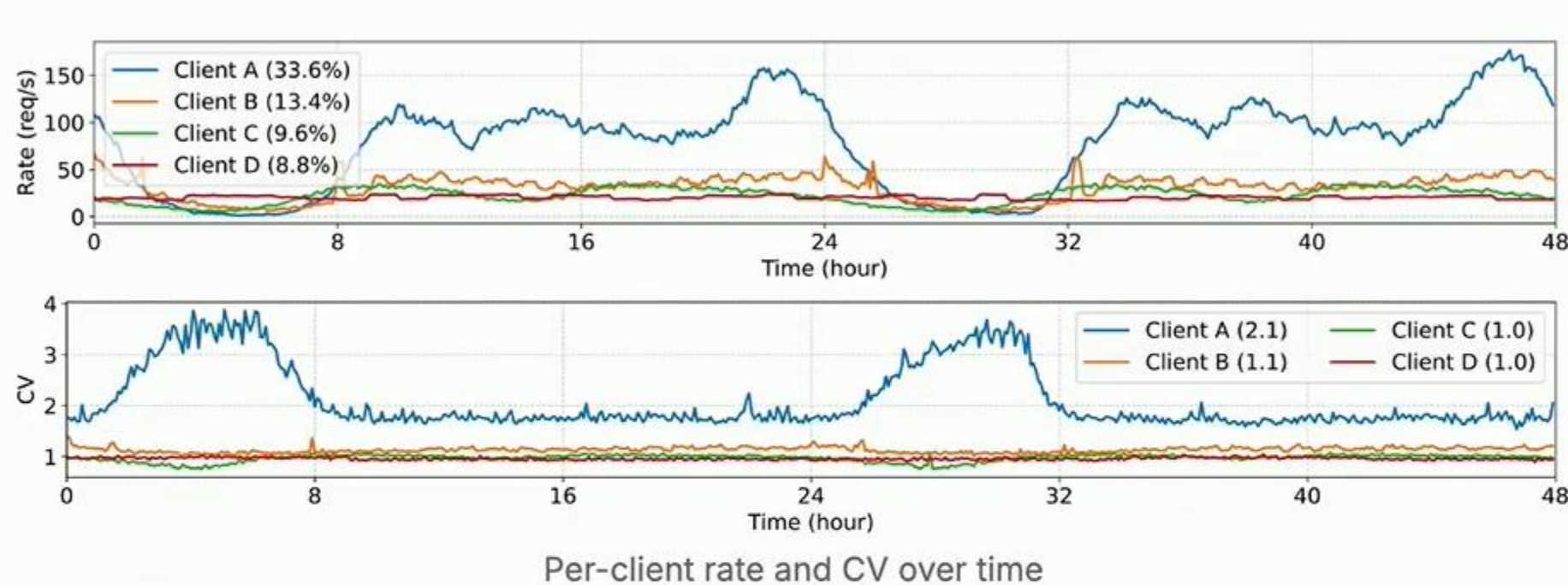


Heterogeneous but predictable

- Top clients are individually relatively **stable** in burstiness and lengths
- **Causal model:** client rate fluctuations steer the entire workload...

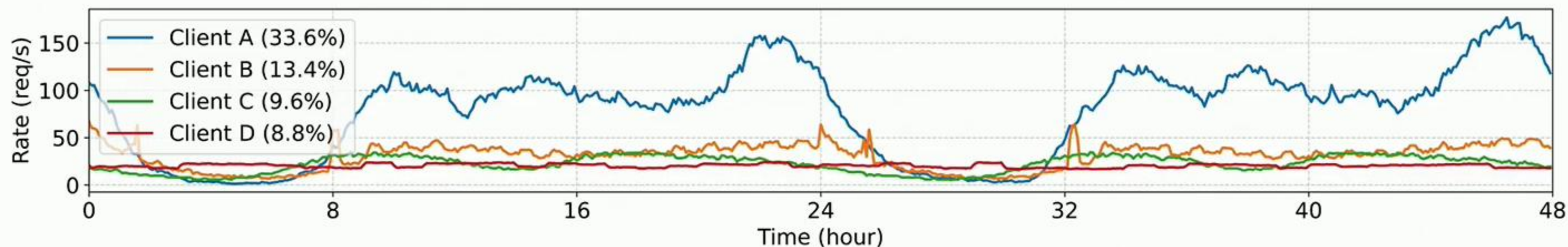


Impact of Top Clients: Explaining Burstiness

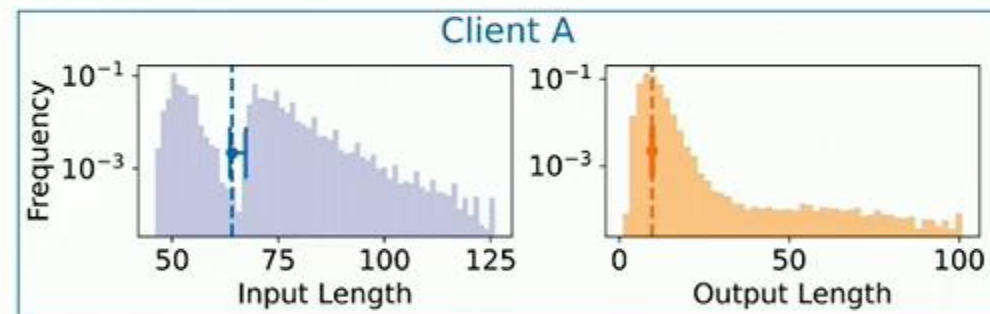


- Client A alone (CV=2.1) drives aggregate burstiness — when its rate rises, the aggregate CV spikes.
- Client A's rate curve (upper, blue) almost mirrors the aggregate CV curve.

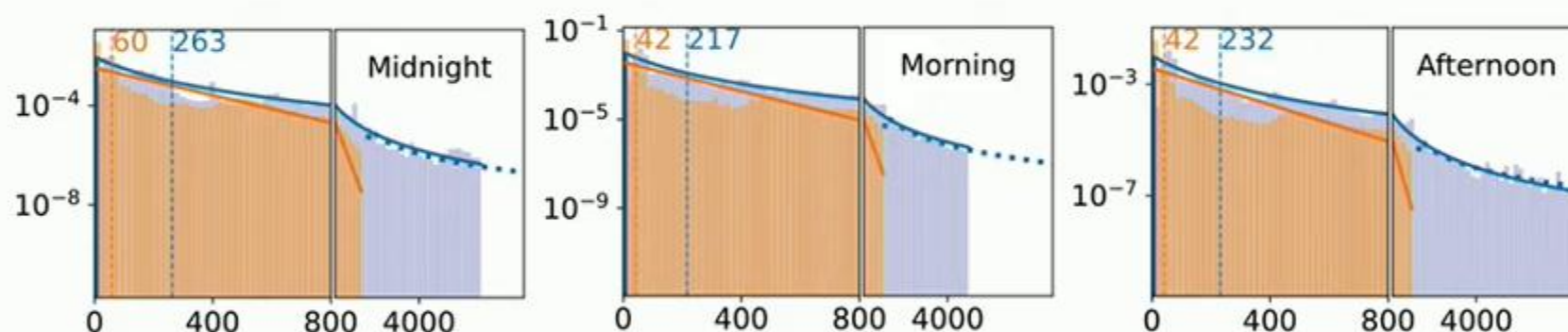
Impact of Top Clients: Explaining Length Shifts



Per-client rate over time



Client A: input/output lengths



Aggregate length distribution shifts (revisited)

- Client A has distinctive short inputs (~65 tokens) — when its rate goes up (Morning), the aggregate input length drops accordingly.

The Causal Model

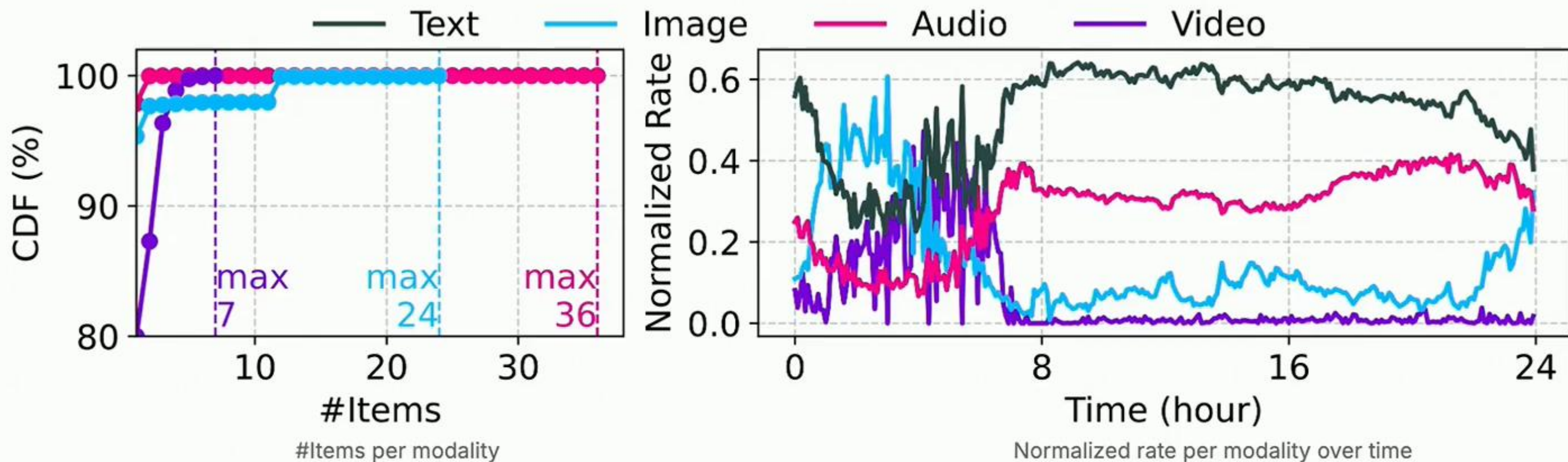
Complex workload patterns — bursts, distribution shifts — are **caused by the mixing** of a few dominant clients with different but individually stable behaviors

ServeGen utilizes client decomposition to model
and generate realistic workloads.

Part II: Emerging Workloads

What's unique about omnimodal and reasoning models

Omni-modal: Multimodal Load

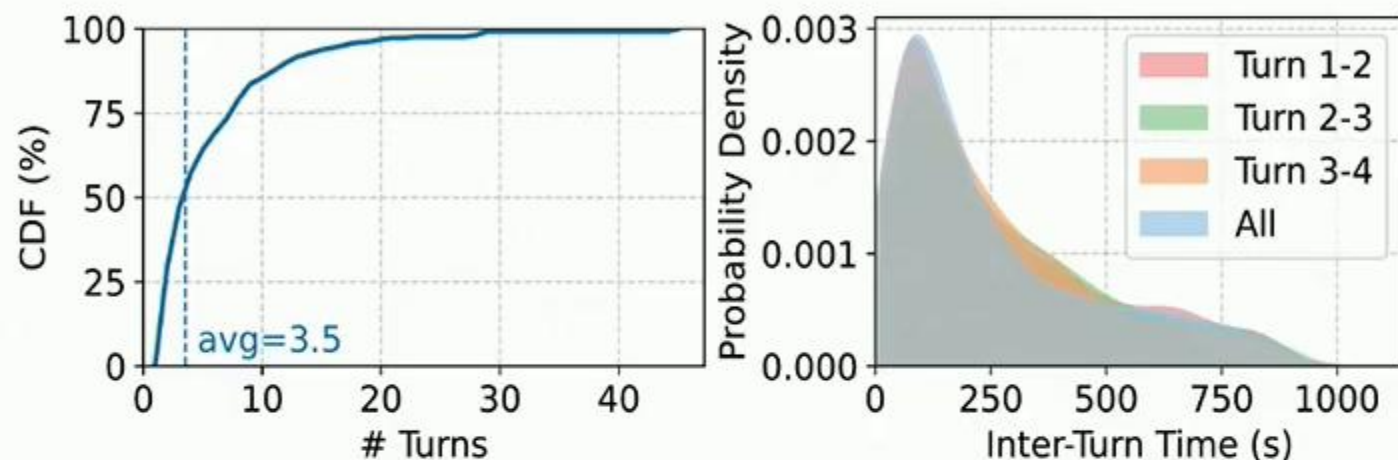


- Long-tailed multimodal load in omnimodal requests (up to **36** audio inputs per message!)
- Image, audio, and video loads shift **independently** of text

i Systems Implication

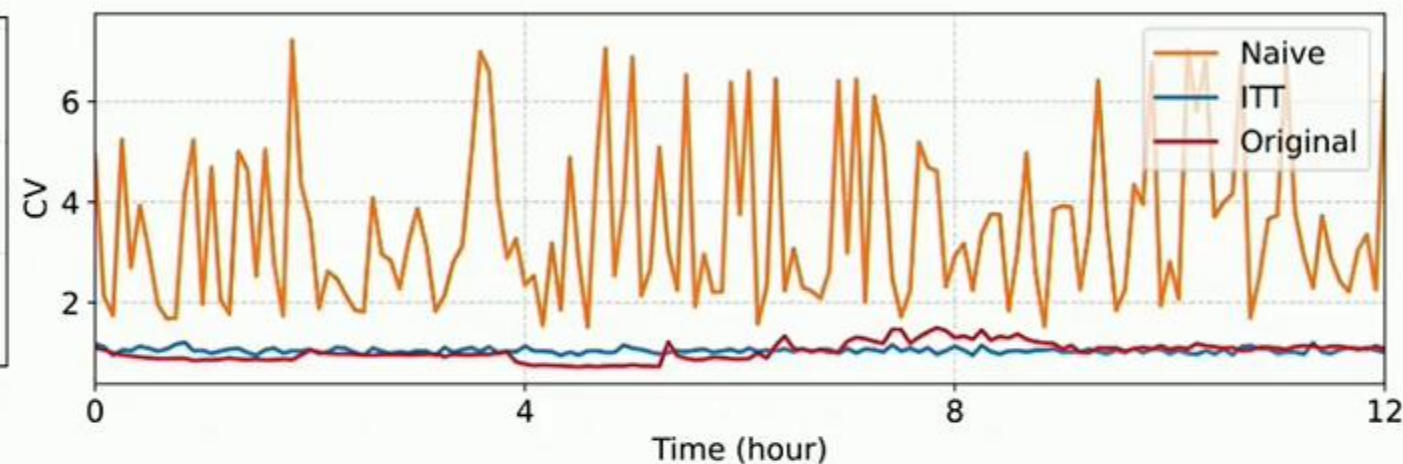
Call for modality-specific scaling and optimizations beyond the LLM backbone.

Reasoning: Multi-Turn Conversations



CDF of conversation turns

Inter-turn time distribution



CV Comparison (w/ and w/o multi-turn requests)

- Reasoning workloads are heavily **multi-turn** (avg. **3.5 turns** per conversation)
- Inter-turn times peak at ~100s with a long tail — users think before responding
- Multi-turn structure **smooths arrivals** ($CV \approx 1$, near-Poisson)

i Systems Implication

Multi-turn patterns enable **short-term predictability** — an opportunity for proactive scheduling and scaling.

More in the Paper

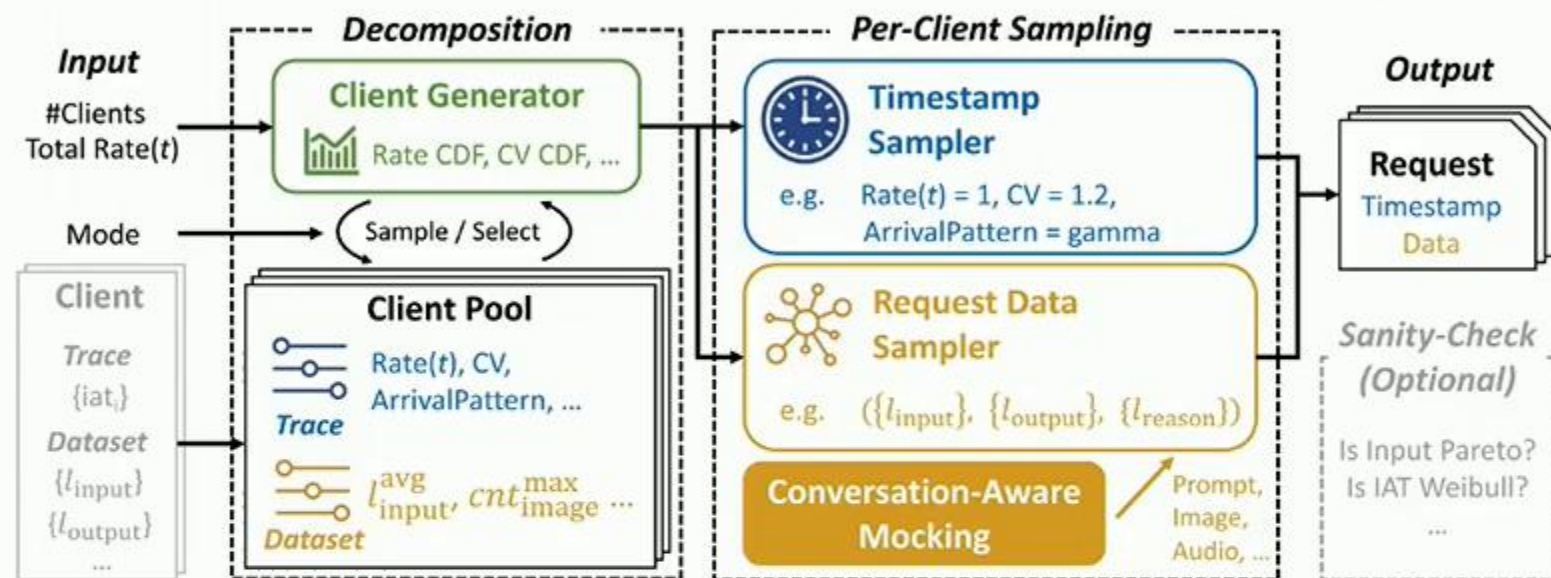
#	Finding	Systems Implication
1-2	Arrivals are bursty with shifting rate/CV	Adaptive scheduling, auto-scaling
3-4	Length distributions shift independently	Phase-aware resource provisioning
5	Top clients steer workload patterns	Per-client modeling for realism
6-8	Multimodal load variance & heterogeneity	Full-stack optimization beyond LLM
9-10	Reasoning: longer outputs, non-bursty	Decoding optimization; predictability
11	Client decomposition generalizes to emerging workloads	Causal model applies broadly

Importantly: causal modeling via client decomposition generalizes across workloads.

Part III: ServeGen Framework

From findings to realistic workload generation

ServeGen: Generating Realistic Workloads



Core principle: compose workloads per-client

Client Generator

Sample heterogeneous clients from a pre-configured pool with realistic rate/CV distributions

Timestamp Sampler

Generate per-client arrivals with time-varying rates; conversation-aware for multi-turn

Data Sampler

Sample request lengths per client; preserve conversation histories via mocking

Generation Accuracy

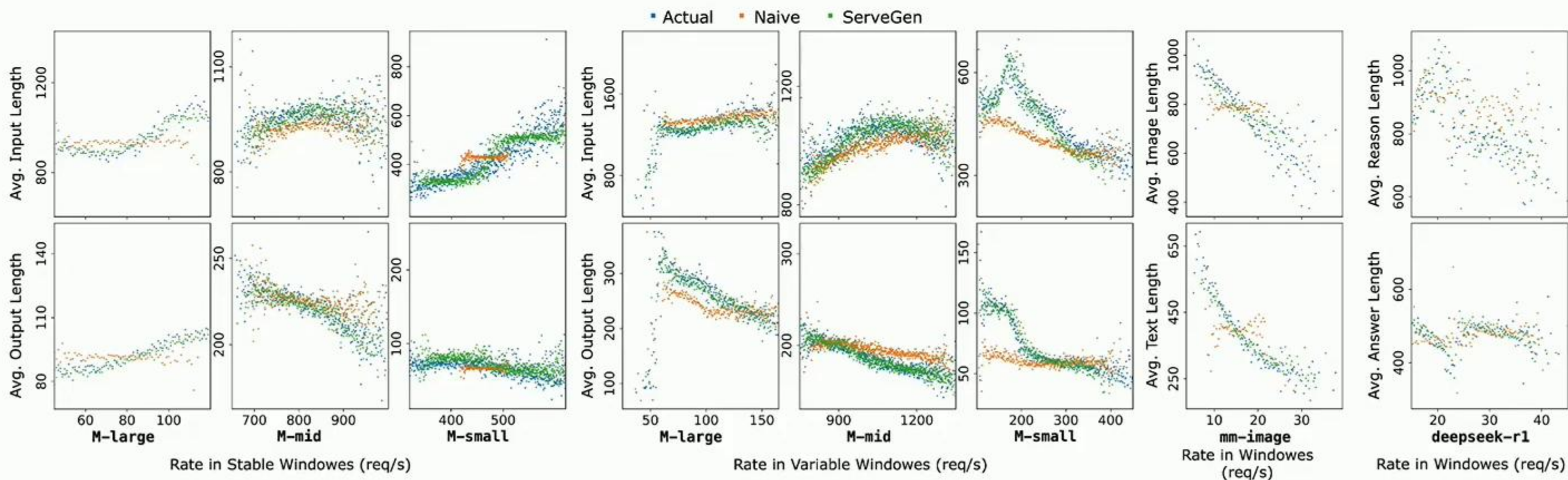
NAIVE: Poisson + Sampling from datasets

ServeGen

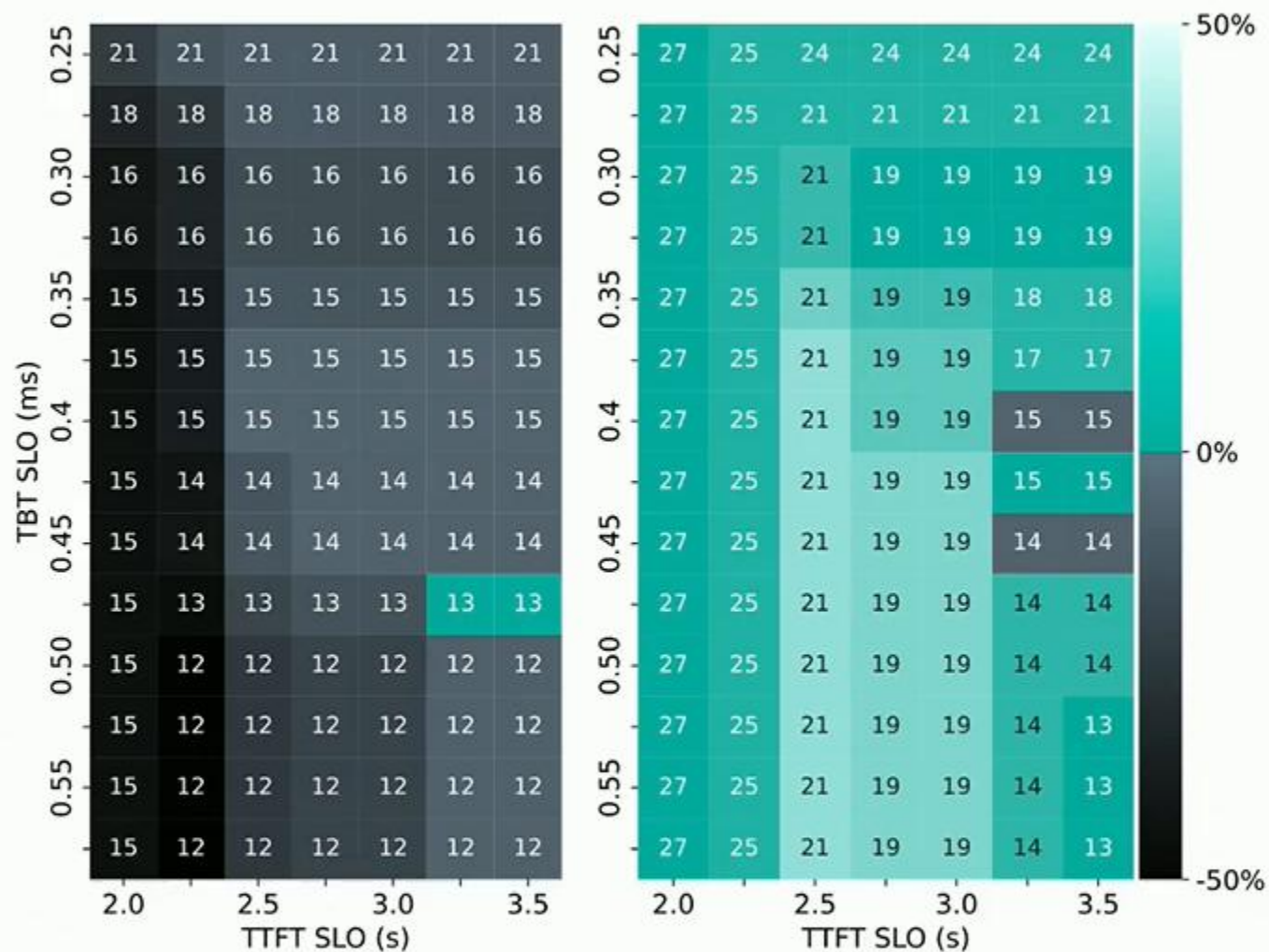
Preserves **realistic** workload patterns (e.g., correlation between rate and data distributions)

NAIVE

Less variable; misses rate–data correlations entirely



Use Case #1: Instance Provisioning



Setup

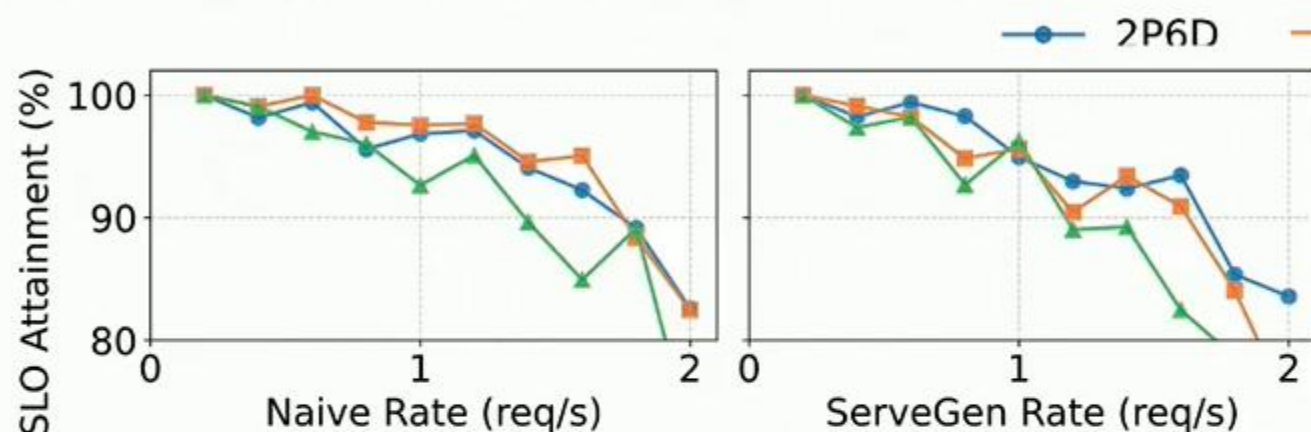
- 30K requests from M-large (10 min)
- vLLM with Qwen2.5-14B on A100 GPUs
- Find minimum #inst to meet TTFT/TBT SLOs

Results

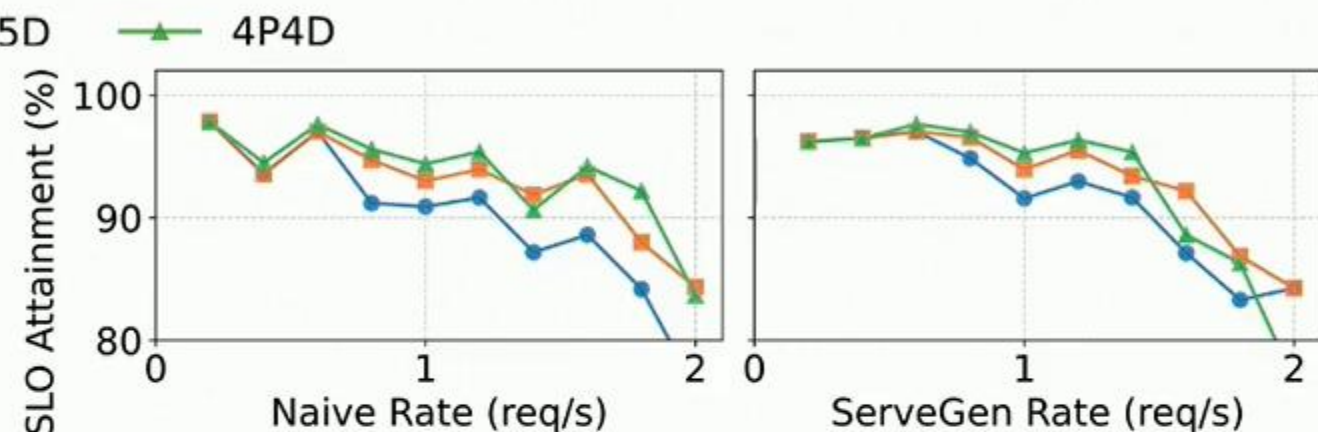
- NAIVE: mostly **under-provisioning** (gray)
- ServeGen: **accurate** provisioning (cyan)

NAIVE workloads are *misleadingly easier* to serve than real workloads.

Use Case #2: PD-Disaggregation



(a) Tight TBT SLO



(b) Tight TTFT SLO

Setup

- Qwen2.5-72B on 4×8 H20 GPUs, PD-disaggregated SGLang
- Vary SLO requirements and prefill/decode split (2P6D, 3P5D, 4P4D)

Results

- Best PD config **differs** between NAIVE and ServeGen (**workload-sensitivity**)
- Aligns with our production experience: **"Unpredictable performance drops"** under realistic workloads

Realistic workloads are crucial for *understanding* newer design choices.

...and more

- ServeGen can **foster further research**
 - Full-stack multimodal optimization?
 - Proactive scheduling for reasoning (or *agentic*) workloads?
 - High-fidelity serving simulation?
- Already benefited production teams at Alibaba on inference optimization



[alibaba/ServeGen](#)

A framework for generating realistic LLM serving workloads



Python



116



12

Thank You

ServeGen: Workload Characterization and Generation of LLM Serving in Production

Open-source: github.com/alibaba/ServeGen

Yuxing Xiang · Xue Li · Kun Qian · Yan Zhang · Wenyuan Yu · Ennan Zhai · Xin Jin · Jingren Zhou

Peking University · Alibaba Group