GPU SMITH
COMPUTE ENGINEERING

GPU SMITH / RESEARCH & PRACTICAL GUIDANCE

LLM Workload Trace Selection: Representativeness vs Replay Loss

> Plan and build private AI compute with GPU Smith. We connect workload requirements with GPU hardware, networking, deployment and operating decisions so your infrastructure fits the work it must perform.

Published by GPU Smith · 2026-11-02 · llm workload trace selection · llm inference workload traces · llm trace replay benchmark · burstgpt · azure llm inference trace dataset · representative llm workload trace · llm serving benchmark · workload fingerprint · trace replay fidelity

Original article: https://gpusmith.com/articles/en/llm-workload-trace-selection-replay-loss-matrix



In this article

1. Executive Summary
2. Introduction and Background
3. BurstGPT
4. Azure LLM Inference Traces
5. Buyer-Supplied and Other Reference Traces
6. Feature Comparison
7. Transformation-Loss Matrix
8. Data Analysis and Evidence
9. The Workload Evidence Contract
10. Implications and Future Directions
11. Frequently Asked Questions (FAQs)
12. Conclusion

Executive Summary

Choosing an LLM workload trace is a measurement-design decision that happens before any GPU is powered on, and the choice silently bounds what a benchmark can prove. This report compares **BurstGPT**, the **Azure LLM inference traces** (2023 and 2024) and **buyer-supplied traces** as candidate inputs for a private inference pilot, and it records what sampling, timestamp scaling, prompt synthesis and retokenization destroy. The analysis is dated 2026-11-02 and draws on sources reviewed on that date.

The three candidates differ in provenance and in what each row can carry. BurstGPT's authors describe 10.31 million traces over 213 days in the current paper abstract (arxiv.org), but the repository documents separate cohorts of 121 days and 110 days (github.com), and an earlier paper version reported 5.29 million traces over 121 days (arxiv.org). Totals therefore belong to a named release and file, not to "BurstGPT" as a whole. Azure's 2023 files are a sample from several services (raw.githubusercontent.com) with three columns and no prompt text (raw.githubusercontent.com), while the 2024 release is labelled a one-week sample of two services (its documented collection window is May 10th-19th 2024) (raw.githubusercontent.com).

The replay layer then removes more. NVIDIA's AIPerf tutorial treats each BurstGPT row as an independent single-turn request and synthesizes prompts of the prescribed token lengths (docs.nvidia.com), and its loader ignores the Model, Total tokens and Log Type columns (raw.githubusercontent.com). A USENIX ATC 2025 paper separately reuses BurstGPT to illustrate real-world request loads (www.usenix.org). vLLM's BurstGPT dataset class keeps only GPT-4 rows, drops failed rows and samples rows (raw.githubusercontent.com). Output length is a cap unless the harness forces it (docs.nvidia.com) (docs.vllm.ai). Uniform time compression preserves interarrival variability but changes per-minute peaks, and random subsampling lowered the interarrival coefficient of variation in the hypothetical illustration computed for this report.

The practical outcome is a **workload evidence contract**: a versioned manifest recording provenance, observed period, row schema, units, tokenizer, failures, conversation identity, content availability, license and every transformation, plus a fingerprint compared against the buyer's aggregated distributions. No public

cohort is a proxy for current private traffic. Public traces are admissible as reference and stress inputs, and sizing claims need a buyer fingerprint. This report supplies no GPU throughput or forecast-accuracy numbers.

Introduction and Background

Private inference architects, capacity planners and procurement performance teams share a recurring problem: a pilot produces tokens per second and latency percentiles, but those outputs are only as meaningful as the traffic that drove them. Arrival bursts, joint input and output length, failure rows and conversation structure all shape queueing, batching and memory pressure, and none of them is a property of the GPU. This report addresses LLM workload trace selection for serving benchmarks and trace replay as a dataset-fitness question, in the register of a specification: what may a candidate trace represent, and what has a transformation already removed before results are read?

Three terms are used consistently. A **trace** is a per-request record, usually an arrival time plus token counts. A **replay** is a harness run that regenerates requests from that record. **Fidelity** is the set of trace properties that survive replay. The same text can produce different token counts depending on the model (help.openai.com), which is why tokenizer records matter. A trace without text cannot test semantic quality, and length-synthesized replay is never a semantic or quality validation. Early serving systems often had no public traces at all: the Orca authors synthesized client requests because there was no publicly available request trace for generative models (www.usenix.org). Public traces have since appeared, but each is a sample of one service in one period.

Scope and method matter. The sources are the dataset owners' repositories, the papers that introduced them, harness documentation and standards-body rules, each reviewed on 2026-11-02. The scope excludes harness migration guides, metrics dictionaries, cache-reuse audits, batching tutorials and autoscaling adoption. GPU Smith is not a provider of any trace or harness discussed, and nothing here reports a client engagement or a measured GPU result. Label conventions: "absent" means a source states or shows the field is not there; "not stated" means the pages read are silent; "hypothetical" marks any illustration generated for this report rather than measured on a public dataset.

BurstGPT

Capabilities

BurstGPT is released by its authors for research use, with the repository describing the release as made for the benefit of the research and academic community (github.com). The README credits Microsoft Azure as the serving platform, and the paper says data came from one regional GPT service provider with over 3,000 users. The documented row schema includes the request submission time, a model label, request and response token counts, a total, and a log type; the README adds Session ID and Elapsed time columns for newer traces. Timestamps are seconds from midnight of the first day, calibrated to the local time zone, so no absolute calendar origin is encoded in the column.

The release structure is where version discipline starts. The README states six files in the current release (github.com): raw and failure-free versions of three parts. The first two parts together span 121 days and the third spans 110 days (github.com). The earliest release (v1.0) covered 61 days and 1.4 million lines (github.com). Line counts per file are stated as 1429.7k and 1404.3k for part one, 3858.4k and 3784.2k for part two, and 5344.0k and 4956.1k for part three. The pages read list no checksums, so a manifest should record a hash computed at download time.

Adoption

The dataset's reuse is documented in two independent places. The NVIDIA AIPerf tutorial links BurstGPT as a replayable trace (docs.nvidia.com), and a USENIX ATC 2025 paper uses it in Figure 3 to show real-world request loads (www.usenix.org), labeling subfigure (a) as the BurstGPT trace with 3k users. These are separate uses: one is tooling documentation and the other is academic illustration, and neither establishes that the data resembles another organization's traffic. The paper also describes an open-sourced benchmark suite called BurstGPT-Perf, which the repository pages read do not mention, so its availability should be verified before it is named in a specification. The paper itself was published at KDD 2025, according to the Hong Kong Baptist University research portal (scholars.hkbu.edu.hk).

Attribution requires care. NVIDIA's page describes the data as from Microsoft Research, while the README credits Azure as the serving platform and the paper's affiliations include universities in Hong Kong and Shenzhen and Huawei Technologies (arxiv.org). A procurement document should attribute the trace to the BurstGPT authors, cite the paper the README asks users to reference (github.com), and treat Azure as the serving platform rather than the data owner. The repository's LICENSE file is Creative Commons Attribution 4.0 International (github.com), though the pages read do not say which release assets it covers; record the license text and retrieval date per file.

Strengths and Limitations

The strengths are scale, a failure signal and documented dynamics. The paper reports failure rows with a response length of zero (arxiv.org), releases a cleaned and a raw trace, and reports a failure rate exceeding 5% for the ChatGPT conversation service. It characterizes the conversation service as peaking on weekdays and diminishing on weekends (arxiv.org), and the API service as aperiodic and bursty. It reports Zipf-like request lengths and bimodal ChatGPT response lengths (arxiv.org), with a linear request and response correlation.

The limitations are equally specific. Content is absent: an early version states that the content of requests and responses is not recorded (arxiv.org). The data comes from a single provider (arxiv.org), and an earlier version listed session ID and response time as missing fields, which later README columns partly address. Session ID exists only for conversation mode (github.com), and Elapsed time is end-to-end latency, explicitly not time to first token. No tokenizer is named in the paper versions or README pages read, so the token-count definition is not stated. The paper's 213 days and 10.31 million traces (arxiv.org) are not reconciled with the repository's 121-day and 110-day cohorts in the sources read; this report preserves both rather than combining them.

Azure LLM Inference Traces

Capabilities

Azure's public LLM inference data comes in two releases. The 2023 page calls it a sample of the traces from multiple LLM inference services (raw.githubusercontent.com), collected on November 11th 2023. The Splitwise paper that introduced it says the released data is a subset, with arrival time and prompt and output token sizes per request. The released files are separate conversation and code traces with three columns, TIMESTAMP, ContextTokens and GeneratedTokens (raw.githubusercontent.com). The 2024 release is a longer one-week sample of two services (raw.githubusercontent.com), tied to the HPCA 2025 DynamoLLM paper (raw.githubusercontent.com) and described as collected on May 10th-19th 2024. That window is ten days, while the README calls the sample one week, so a manifest should record the first and last timestamps actually observed in the files.

Content is explicitly unavailable: the owners state they have no visibility into the content of the prompts (raw.githubusercontent.com). The license is stated as a CC-BY Attribution License, and the repository LICENSE file contains the Creative Commons Attribution 4.0 International text (raw.githubusercontent.com); users are asked to cite the accompanying paper. The repository also states the traces are provided as they are.

Adoption

Splitwise and DynamoLLM used the data for service-level characterization. Splitwise reports a coding median prompt of 1500 tokens (arxiv.org) against 1020 for conversation (arxiv.org), and median outputs of 13 (arxiv.org) and 129 tokens (arxiv.org), with an almost bimodal conversation output distribution. That contrast makes the two files a natural pair of stress inputs: one prompt-heavy with short outputs, one more balanced. DynamoLLM reports a diurnal pattern with daily peaks and lower load at night (arxiv.org), and says the popularity of request types changes over time, so a single-hour window of a mix that moves over a week is a sample of a moving target.

Strengths and Limitations

The strengths are clear provenance of purpose and a plain schema. The timestamp is documented only as the invocation time, and the data page describes validation by reproducing the paper's characterization graphs. The first data row of the 2023 code file shows a seven-digit fractional-second wall-clock timestamp dated 2023-11-16 (raw.githubusercontent.com), while the documentation gives a collection date of November 11th. A manifest should therefore record the first and last timestamps observed in the downloaded file rather than inherit the documentation date. The Splitwise authors characterized 20-minute traces, and the released window length is not stated on the data page.

Absence is inferred from the three-column schema: no error column, session identifier or tokenizer is documented, and time zone and origin are not stated. DynamoLLM says its internal week-long traces are a super-set of the open-source ones, so conclusions in that paper may not be reproducible from the public files. Arrival-process assumptions made in the papers that introduced the traces are methodological choices, not properties of the public file, and a reader replaying the file does not inherit them.

Buyer-Supplied and Other Reference Traces

Capabilities

A buyer-supplied trace is the only candidate that can speak to the buyer's current population, and it is also the one least likely to be shareable. The enterprise-safe pattern is to hold raw logs inside the buyer boundary, compute aggregated fingerprints there, and export only distributions. Published releases show how much can be shared. The Mooncake traces carry timestamp, input and output length and hash IDs, and say raw text and tokens are not included for privacy protection (github.com). The Alibaba Bailian traces use randomized chat identifiers (github.com) and 16-token salted hash blocks, so session structure survives without content or user identity.

Adoption

Conversation-capable public data sets sit in different families. LMSYS-Chat-1M was collected from 210K unique IP addresses (huggingface.co) between April and August 2023, and WildChat adds timestamped transcripts, although its conversation-level timestamp marks the last turn in UTC (huggingface.co), a coarse arrival proxy. ShareGPT-derived data narrowed roughly 100k conversations to 53k by filtering (huggingface.co). vLLM's benchmark CLI documents ShareGPT as a dataset option but treats arrival as a client-side rate parameter (docs.vllm.ai), and the PagedAttention paper notes its datasets did not include timestamps, so arrival times were generated. These corpora supply text and turn structure but not production arrival timing, which is the opposite of the Azure and BurstGPT gap.

Strengths and Limitations

The limits are explicit in each release. Mooncake's conversation and tool traces were sampled from one hour of online data (github.com), its sampler prioritized requests within the same session, its synthetic trace uses Poisson-generated arrivals, and its authors report that reuse in their online traces is much smaller than in open benchmark datasets (arxiv.org). The Bailian trace is a two-hour sample from a single cluster (github.com), and removes absolute timestamps, so time-of-day patterns cannot be recovered. ServeGen released parameterized and sanitized data for confidentiality. Licenses also differ: the LMSYS agreement prohibits transferring the dataset to third parties (huggingface.co), which bears on sharing it with a vendor, and the WildChat card records a license update (huggingface.co). Each license should be read per file at the retrieval date, and the transfer restriction is a reason to keep such corpora out of vendor hands.

Feature Comparison

Table 1 below is the dataset ledger for the three primary candidates. It uses the three-state convention above (present, absent, not stated) so that an unknown field is never silently read as an absent one. Rows reflect sources reviewed on 2026-11-02.

Property	BurstGPT	Azure 2023 (conv/code)	Azure 2024 (conv/code)	Buyer-supplied (template)
Owner and attribution	Authors of the BurstGPT paper; Azure credited as serving platform (github.com)	Microsoft Azure public dataset; cite Splitwise	Cite the DynamoLLM paper (raw.githubusercontent.com)	Buyer; record data steward
Observed period	Cohorts of 121 and 110 days (github.com); calendar origin not stated	Doc says collected on November 11th 2023 (raw.githubusercontent.com); verify in file	May 10th-19th 2024 per page	Declare start, end, time zone
Row schema	Timestamp, model, request and response tokens, total, log type; newer files add Session ID and Elapsed time	TIMESTAMP, ContextTokens, GeneratedTokens (raw.githubusercontent.com)	Same three fields on the data page	Declare; hash identifiers
Timestamp unit and origin	Seconds from midnight of day one, local time zone	Documented only as invocation time ; absolute wall-clock strings seen in file	Not verified here	Declare unit, origin, UTC offset
Tokenizer	Not stated	Not stated	Not stated	Pin name and revision
Failures	Zero response tokens (arxiv.org); raw and cleaned files	No error field documented	No error field documented	Keep status codes, add error class
Conversation identity	Session ID in conversation mode only	Absent from schema	Absent from schema	Hash and keep if consented
Content availability	Absent; lengths only (arxiv.org)	No visibility into prompts (raw.githubusercontent.com)	Same statement	Keep inside boundary
License	CC BY 4.0 file in repo (github.com)	CC-BY Attribution	CC-BY wording repeated	Internal terms

Reading the table: the schema gap between BurstGPT and Azure is smaller than the gap in meaning. BurstGPT rows carry a service label and a log type, so conversation-mode and API-mode traffic can be separated, while Azure's two files are split by service but carry no such label inside the row. Both lack a documented tokenizer and both lack message text. Neither gives a timestamp that aligns with the other without an additional offset assumption, which matters when comparing BurstGPT with the Azure LLM inference traces: they are different services, periods and units, so they should be treated as two reference points, not interchangeable samples.

Transformation-Loss Matrix

Replay tools and sampling choices transform a trace before the GPU sees it. *Table 2* lists common transformations, what survives each, and what is destroyed, using documented harness behavior where it exists.

Transformation	Survives	Destroyed or distorted	Evidence and caveat
Contiguous window	Local burst structure, within-window length mix	Weekly and diurnal cycles outside the window	AIPerf offsets skip earlier rows (docs.nvidia.com); hypothetical result in Table 3
Random subsample	Marginal length histograms	Burst peaks and arrival variability; per-minute p95	Hypothetical result in Table 3; TraceSplitter warns that rescaling can exaggerate or eliminate fine-grained burstiness (microsoft.com)
Uniform time compression	Interarrival coefficient of variation, order	Absolute rates; per-minute counts scale with the factor	Authors multiply timestamps to reach a target rate (arxiv.org); TraceSplitter describes this as the common ad hoc approach (microsoft.com); AIPerf's replay speedup option is supported only by the baseten_trace loader (docs.nvidia.com), so BurstGPT compression is a recorded preprocessing step
Conversation-unaware upscaling	Aggregate rate	Session and turn interval structure	ServeGen contrasts a conversation-aware method with naive interarrival scaling (arxiv.org)
Poisson re-timing	Mean rate	Burstiness (coefficient of variation near 1)	Poisson is a generator default in vLLM (docs.vllm.ai); Poisson is documented as often poorly modelling bursty workloads
Length-synthesized prompts	Input token counts	All prompt meaning, prefix sharing, quality signal	AIPerf synthesizes prompts of prescribed lengths (docs.nvidia.com); text comes from a corpus (docs.nvidia.com); replay requests are text-only (docs.nvidia.com)
Row-independent replay	Request lengths and offsets	Session dependencies and turn intervals	AIPerf assigns each BurstGPT row its own session (raw.githubusercontent.com)
Column dropping or filtering	Selected columns only	Model mix, log-type mix, GPT-4 vs ChatGPT split	AIPerf ignores Model, Total tokens and Log Type (raw.githubusercontent.com); vLLM keeps only GPT-4 rows (raw.githubusercontent.com)
Failure removal	Successful-request lengths	Error share and failure-linked bursts	vLLM removes failed requests (raw.githubusercontent.com); raw and cleaned files exist
Retokenization	Nominal token counts	Equivalence of counts across tokenizers	Counts differ by model (help.openai.com); AIPerf's client tokenizer defaults to the model name
Output cap without forcing	Requested maximum	Realized output length	Output length is a maximum (docs.nvidia.com); EOS can stop generation early (docs.nvidia.com)

The matrix shows that most losses come from the harness rather than the file. A replay that preserves timestamps but synthesizes prompts retains arrival offsets and length pairs while losing meaning. A replay that samples rows and generates arrivals from a rate parameter, as vLLM's BurstGPT class does (raw.githubusercontent.com), retains marginal lengths and loses the original arrival process entirely.

What Survives in Practice

Four harness behaviors deserve explicit entries in a test specification:

- **Open versus closed load**
- **Output length enforcement**
- **Unit and warmup hazards**
- **Prefix caching**

The details follow. First, open versus closed load: Schroeder and colleagues show that open systems, where arrivals are independent of completions, and closed systems, where arrivals are triggered by completions, can give significantly different results (www.usenix.org). AIPerf fixed-schedule replay dispatches requests at recorded times, but a concurrency limit blocks requests at the limit (docs.nvidia.com), which makes the run a hybrid of the two models; AIPerf can report schedule lag to show how faithfully the recorded schedule was delivered (docs.nvidia.com). vLLM's max-concurrency parameter is likewise unlimited by default, and an infinite request rate sends all requests at time zero (docs.vllm.ai).

Second, output length. The AIPerf metrics reference states that EOS tokens can stop generation early (docs.nvidia.com), and provides mismatch metrics that help identify when the server is not honoring the requested length. vLLM defines max_tokens as a cap (docs.vllm.ai) and ignore_eos as continuing past the EOS token; its timed-trace code ignores EOS to reach the traced output length (raw.githubusercontent.com). Splitwise forced the traced output length with synthetic prompts, arguing prompt text does not affect the metrics it benchmarks (arxiv.org); that argument is scoped to its metrics and does not carry to quality evaluation.

Third, unit hazards. BurstGPT seconds are converted to milliseconds by AIPerf (docs.nvidia.com), Mooncake timestamps are already milliseconds (github.com), and vLLM's timed-trace mode needs an explicit multiplier to reach seconds (docs.vllm.ai). A thousandfold unit error compresses a trace to a thousandth of its duration and is easy to miss in a summary table. Options also change the test type: AIPerf can strip per-turn timestamps and inter-turn delays at load time (docs.nvidia.com), and a flag keeps a user-selected timing mode instead of fixed-schedule replay; either turns a replay into a rate or concurrency test, and the manifest should say which was run. Warmup adds a further wrinkle: the warmup phase always uses rate-based scheduling internally (docs.nvidia.com) and its metrics are discarded (docs.nvidia.com), so the first minutes of a replayed window are not a faithful copy of the recorded schedule. Fourth, prefix caching: repeating a benchmark can inflate throughput through prefix-cache reuse (docs.vllm.ai), and length-synthesized prompts do not reproduce production prefix sharing, so cache conclusions require a separate audit outside this scope.

Data Analysis and Evidence

This section turns the documented numbers into checks a reviewer can reproduce. All derived values below are arithmetic on the repository's stated line counts and durations, labeled as derived.

Failure share by file (derived). From the README counts, the raw-minus-cleaned difference is 25.4k rows of 1429.7k for part one (1.78%) (github.com), 74.2k of 3858.4k for part two (1.92%), and 387.9k of 5344.0k for part three (7.26%). These are row shares, not the paper's service-level statistic of above 5% for ChatGPT conversation (arxiv.org), and the README does not define the removed rows beyond the failure convention. A buyer who adopts a cleaned file inherits a failure share of zero by construction.

Mean arrival rate by cohort (derived). Part one's 1429.7k lines over 61 days (github.com) give about 0.27 requests per second, treating the v1.0 release's 61-day span as the part-one window (an inference: the sources give 61 days for v1.0 and 121 days for parts one and two together, but do not state part one's span on its own); part three's 5344.0k lines over 110 days (github.com) give about 0.56. The paper separately reports 0.019 requests per second for the conversation service and 0.21 for the API service (arxiv.org). These figures describe different cohorts and aggregations, so a rate is meaningful only with its file name attached. These rates are low for loading a multi-GPU node, which is why BurstGPT's authors suggest scaling average RPS to the evaluation setup; in vLLM the arrival rate is a client-side parameter that controls the target request generation rate (docs.vllm.ai).

Unit-checked scaling. For a trace with timestamps in seconds, time compression by a factor `scale` is t_new = (t_old - t0) / scale, where t0 is the first timestamp. The mean rate rises by `scale`, per-minute counts rise accordingly, and the interarrival coefficient of variation is unchanged, because both the mean and standard deviation of gaps divide by `scale`. NIST defines the sample coefficient of variation as the ratio of standard deviation to mean (www.itl.nist.gov); the Azure Functions paper notes Poisson arrivals would ideally yield a CV of 1 and values above 1 indicate variability (www.usenix.org). In vLLM, lower burstiness values produce bursty traffic and higher values uniform traffic (docs.vllm.ai), and for a Gamma process the interarrival CV equals one over the square root of the shape parameter (a mathematical identity, not a measured value).

Fingerprint definition. The workload fingerprint used in this report has seven components, computed with the same code on original and transformed rows:

- **Per-minute arrivals:** mean, p95 and max request count.
- **Interarrival CV:** standard deviation of gaps divided by their mean.
- **Error share:** fraction of rows with zero response tokens.
- **Joint histogram:** log2-binned input and output tokens.
- **Length correlation:** log-scale input and output correlation on non-failed rows.
- **Session-turn intervals:** the gap distribution between turns, where session identity exists.
- **Divergence:** Jensen-Shannon divergence on the joint histogram between original and transformed rows.

(Hypothetical Example) Transformation Effects on a Synthetic Trace

Table 3 is generated from a synthetic trace, not from any public dataset: 20,000 requests with Gamma-distributed gaps (shape 0.25, scale 16 seconds), lognormal input lengths, correlated lognormal outputs, and a 2% share of zero-output rows, produced with Python 3.13.5 standard library and seed 7. The lognormal input, output and correlation parameters are not published here, so the table is illustrative only and is not meant to be reproduced exactly; it shows direction of effect only.

Transformation (hypothetical)	Rows	Per-minute mean / p95 / max	Interarrival CV	Error share	Log in/out corr.	JS divergence (joint)
Original	20,000	14.83 / 28.0 / 47	1.98	0.018	0.64	0.0000
Contiguous 25% window	5,000	15.24 / 29.0 / 45	1.96	0.019	0.64	0.0016
Random 25% subsample	5,000	3.71 / 8.0 / 16	1.35	0.019	0.65	0.0021
Output lengths shuffled	20,000	14.83 / 28.0 / 47	1.98	0.018	0.00	0.0860
Time compression x4	20,000	59.17 / 85.1 / 98	1.98	0.018	0.64	0.0000
Poisson re-timing, same mean rate	20,000	14.73 / 21.0 / 29	1.01	0.018	0.64	0.0000
Failure rows removed	19,638	14.56 / 28.0 / 46	1.98	0.000	0.64	0.0091
Input truncated to 1024	20,000	14.83 / 28.0 / 47	1.98	0.018	0.62	0.0204

In this illustration, the contiguous window kept the interarrival CV and peak structure while the random subsample lowered the CV from 1.98 to 1.35 and cut the per-minute p95 roughly in proportion to the row count (28.0 to 8.0 on one quarter of the rows), so the distortion shows up in the CV and the peak structure rather than in the p95 alone. Poisson re-timing preserved the mean rate but moved the CV to 1.01 and cut the p95 from 28.0 to 21.0, which is the loss that matters for queueing. Shuffling outputs left the arrival columns untouched but erased the length correlation, a loss visible only in a joint comparison. Uniform compression left the CV unchanged, as the algebra predicts, yet it raised the p95 per-minute count because a minute now contains four original minutes. Among the changes that kept the arrival process, shuffling outputs produced the largest joint-histogram divergence (0.0860), followed by input truncation (0.0204). Real traces may behave differently; the point is that each transformation moves a specific fingerprint component, which is why fingerprints, not row counts, should gate acceptance.

The Workload Evidence Contract

The contract below converts the matrix into acceptance rules. It determines whether an input dataset is fit for a stated pilot decision; it does not select a harness or interpret results.

Step 1: Define the population and decision

State whether the pilot population is single-turn chat, code completion, multi-turn sessions or internal document prompts. A code assistant resembles the Azure code file on the dimension Splitwise measured (median prompt 1500 tokens, median output 13) (arxiv.org), but session timing for agents is not covered: ServeGen reports inter-turn times concentrated around 100 seconds with a long tail (arxiv.org), a structure that independent-row replay cannot represent.

Step 2: Fill the ledger and mark absent versus not stated

Use Table 1's fields for every candidate: owner, observed period, release and hash, schema, units, tokenizer, failures, conversation identity, content availability and license. Reject any candidate whose observed period, unit or origin cannot be stated, and record observed first and last timestamps from the file itself.

Step 3: Record every transformation

List windowing, subsampling, scaling factor, prompt synthesis, tokenizer, truncation and filters, with the Table 2 property each one loses. Time-scaling must state the formula and factor, and conversation-unaware scaling must be flagged when sessions exist (arxiv.org).

Step 4: Fingerprint and compare

Compute the fingerprint on the original and transformed trace, and on the buyer's logs inside the buyer boundary. Compare to aggregated buyer distributions; export no text by default. A mismatch in interarrival CV, per-minute p95 or joint-histogram divergence is a rejection reason for sizing use.

Step 5: Select a portfolio and admissible uses

Choose a **reference** trace (a contiguous BurstGPT window with raw failure rows retained), a **stress** trace (an Azure prompt-heavy code window or a scaled burst window) and a **buyer** trace (aggregated distributions or a sanitized replay). Write admissible uses before any GPU result is read: for example, "relative comparison of two platforms under identical replay" is admissible on public traces, while "fleet size for production" requires the buyer fingerprint match.

Step 6: Publish a manifest and rejection reasons

An illustrative manifest template, with a pinned command pattern adapted from NVIDIA's tutorial (which uses a `latest` container tag (docs.nvidia.com) and a ten-row smoke subset, neither suitable for a pilot):

```
trace_manifest_version: 1
dataset: BurstGPT
file: BurstGPT_without_fails_3.csv      # or the raw file, stated
sha256: <computed at download>
retrieved_utc: 2026-11-02T00:00:00Z
license: CC BY 4.0 (repo LICENSE, retrieved same day)
observed_period: "110 consecutive days (README); calendar origin not stated"
timestamp_unit: seconds
tokenizer: not stated by dataset; replay uses served-model tokenizer
transformations:
```

```
  - window: [t_start_s, t_end_s]
  - time_scale: "t_new = (t_old - t0) / 8"
  - prompts: length-synthesized (no semantic validity)
  - failures: removed | retained
harness: aiperf==0.13.0   # pypi.org lists 0.13.0
server_image: vllm/vllm-openai@sha256:<digest>
model: Qwen/Qwen3-0.6B@<revision>
output_length_enforcement: "--extra-inputs ignore_eos:true; OSL-mismatch metrics checked"
rejection_reasons: []
```

AIPerf's published version at the time of reading was 0.13.0 (pypi.org) (docs.nvidia.com); the loader source examined was the main branch, which may differ from a tagged release, so pin and re-read the loader at the tag used. Rejection reasons should be coded so that every refusal is traceable:

- **R1:** no stated observed period or time origin.
- **R2:** unknown timestamp unit.
- **R3:** failures removed without a recorded note.
- **R4:** content required (for example prefix or semantic claims) but absent.
- **R5:** session dependencies required but replay is row-independent.
- **R6:** tokenizer or model revision unrecorded.
- **R7:** license or sharing terms unverified.

Implications and Future Directions

Several implications follow for pilots. First, headline totals are not interchangeable across versions: the same repository and paper have described 5.29 million traces over 121 days (arxiv.org) and 10.31 million over 213 days (arxiv.org), and the repository documents more than one cohort. Manifests should name file and hash, not the dataset brand. Second, harness choices are part of the dataset. When vLLM's BurstGPT class filters to GPT-4 rows and drops failures (raw.githubusercontent.com), the same CSV becomes a different workload from the one AIPerf replays with timestamps (docs.nvidia.com), so cross-harness comparisons need identical transformations.

Licenses can also change after download, as the WildChat card's license update shows (huggingface.co). Third, public traces age. The Azure data is from November 2023 and May 2024, and serving workloads keep changing. As agentic and reasoning workloads grow, row-independent traces under-represent multi-step structure; the likelihood that request-length distributions change over time suggests that a single window should not be treated as stationary. Fourth, borrowed chat corpora can mislead on cache behavior, since Mooncake's authors report much smaller reuse in their online traces than in open benchmark datasets.

For procurement test specifications, the practical recommendation is modest: require a manifest, require a buyer fingerprint comparison before sizing statements, and treat public traces as reference and stress inputs. The main uncertainty cannot be reduced by better public data: how closely any historic public cohort matches a specific enterprise's current traffic is an empirical question answered only by the buyer's own aggregated logs. Open items include a documented tokenizer for BurstGPT counts, calendar origins for BurstGPT timestamps, and published checksums for all trace releases.

Frequently Asked Questions (FAQs)

How do BurstGPT and the Azure LLM inference trace dataset differ?

They differ in service, period, unit and schema. BurstGPT documents cohorts of 121 and 110 days with model, log type and optional session columns (github.com), whereas Azure's 2023 files are a sample of several services with three columns (raw.githubusercontent.com). BurstGPT timestamps are seconds from day one; the Azure 2023 file uses wall-clock strings, and the 2024 timestamp format was not verified here. Compare them as separate reference points.

What makes a representative LLM workload trace, and how is workload representativeness judged?

Representativeness is a match between trace and target on arrival variability, joint length distribution, error share, session structure and content type, measured by a fingerprint. No public trace is representative of every private workload. Single-service samples such as BurstGPT's single provider (arxiv.org) and Mooncake's one hour (github.com) should be called what they are.

How faithful is production LLM trace replay (replay fidelity)?

Fidelity depends on the harness. AIPerf replays timestamps but synthesizes prompts and treats rows as independent (docs.nvidia.com), and realized output length may differ from the trace unless forced (docs.nvidia.com). Schedule-lag metrics can show whether the recorded schedule was delivered.

Which request arrival pattern traces are public?

BurstGPT and Azure supply arrival offsets; Mooncake supplies relative millisecond arrivals (github.com); Bailian removes absolute timestamps (github.com). ShareGPT-style corpora (huggingface.co) carry no production arrival timing.

Is the MLPerf approach an alternative?

MLPerf Inference's Server scenario generates Poisson arrivals (github.com) with latency limits such as 2000 ms and 200 ms for time to first token and time per output token in a conversational category (github.com). The rules standardize comparison across systems rather than replaying a particular service, so it answers a different question from trace replay.

Should tokenizers be pinned?

Yes. The same text can produce different token counts depending on the model (help.openai.com), Anthropic's documentation advises counting the same request under two models to measure a tokenizer difference (docs.anthropic.com), tokenizers use differing subword algorithms (huggingface.co), and MLCommons notes the reading-speed mapping of TPOT depends on the tokenizer (mlcommons.org). Record the tokenizer revision beside the dataset.

Conclusion

LLM workload trace selection is a documentation and measurement discipline before it is a benchmarking one. BurstGPT offers arrival timing, token counts, a failure signal and, in newer files, conversation identity, but its totals depend on the release, its tokenizer is not stated, and it contains no prompt content. The Azure traces offer service-split samples with a plain schema and explicit statements that content is not visible. Buyer-supplied traces alone can represent current private traffic, and they are best used through aggregated fingerprints.

The transformation-loss matrix shows that replay tools, not just files, determine fidelity: row-independent sessions, length-synthesized prompts, filtered columns, removed failures and uncapped output lengths each remove a specific property. The recommended output is a versioned workload manifest with coded rejection reasons, a portfolio of reference, stress and buyer traces, and admissible uses written before GPU results are read. Output length is a maximum unless the harness forces it (docs.vllm.ai), and mismatch metrics exist to expose when it was not honored (docs.nvidia.com). Conclusions about fleet size should be withheld until the buyer fingerprint is compared with the trace actually replayed. In short, a public trace answers the question it was recorded for, and the manifest states which question that was. Where the manifest cannot state a unit, an origin, a tokenizer or a transformation, the correct response is to reject the input for sizing use and say why, rather than to read the GPU results more carefully.

Source links

Links cited in this article, in order of first appearance. Full addresses are provided for readers using extracted text.

1. https://arxiv.org/abs/2401.17644
2. https://github.com/HPMLL/BurstGPT
3. https://arxiv.org/html/2401.17644v3
4. https://raw.githubusercontent.com/Azure/AzurePublicDataset/master/AzureLLMInferenceDataset2023.md
5. https://raw.githubusercontent.com/Azure/AzurePublicDataset/master/data/AzureLLMInferenceTrace_code.csv
6. https://raw.githubusercontent.com/Azure/AzurePublicDataset/master/README.md
7. https://docs.nvidia.com/aiperf/tutorials/datasets-inputs/profile-with-burst-gpt-traces
8. https://raw.githubusercontent.com/ai-dynamo/aiperf/main/src/aiperf/dataset/loader/burst_gpt.py
9. https://www.usenix.org/system/files/atc25-he-yongjun.pdf
10. https://raw.githubusercontent.com/vllm-project/vllm/main/vllm/benchmarks/datasets/datasets.py
11. https://docs.nvidia.com/aiperf/latest/tutorials/load-patterns-scheduling/fixed-schedule-benchmarking.md
12. https://docs.vllm.ai/en/latest/api/vllm/sampling_params
13. https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them
14. https://www.usenix.org/system/files/osdi22-yu.pdf
15. https://github.com/HPMLL/BurstGPT/releases
16. https://docs.nvidia.com/aiperf/tutorials/datasets-inputs/profile-with-burst-gpt-traces.md
17. https://scholars.hkbu.edu.hk/en/publications/burstgpt-a-real-world-workload-dataset-to-optimize-llm-serving-sy

18. https://arxiv.org/html/2401.17644v5
19. https://github.com/HPMLL/BurstGPT/blob/main/LICENSE
20. https://arxiv.org/html/2401.17644v1
21. https://raw.githubusercontent.com/Azure/AzurePublicDataset/master/AzureLLMInferenceDataset2024.md
22. https://raw.githubusercontent.com/Azure/AzurePublicDataset/master/LICENSE
23. https://arxiv.org/pdf/2311.18677
24. https://arxiv.org/pdf/2408.00741
25. https://github.com/kvcache-ai/Mooncake/blob/main/FAST25-release
26. https://github.com/alibaba-edu/qwen-bailian-usagetraces-anon
27. https://huggingface.co/datasets/lmsys/lmsys-chat-1m
28. https://huggingface.co/datasets/allenai/WildChat-1M
29. https://huggingface.co/datasets/anon8231489123/ShareGPT_Vicuna_unfiltered
30. https://docs.vllm.ai/en/latest/benchmarking/cli
31. https://arxiv.org/pdf/2407.00079
32. https://docs.nvidia.com/aiperf/latest/reference/command-line-options.md
33. https://microsoft.com/en-us/research/uploads/prod/2021/04/tracesplitter-eurosys21.pdf
34. https://arxiv.org/pdf/2505.09999
35. https://docs.nvidia.com/aiperf/latest/tutorials/datasets-inputs/synthetic-dataset-generation.md
36. https://docs.nvidia.com/aiperf/latest/tutorials/datasets-inputs/custom-dataset-guide.md
37. https://docs.nvidia.com/aiperf/latest/reference/ai-perf-metrics-reference.md
38. https://www.usenix.org/legacy/event/nsdi06/tech/full_papers/schroeder/schroeder.pdf
39. https://docs.nvidia.com/aiperf/latest/benchmark-modes/load-generator-options-reference.md
40. https://docs.vllm.ai/en/latest/cli/bench/serve.html
41. https://raw.githubusercontent.com/vllm-project/vllm/main/vllm/benchmarks/serve.py
42. https://docs.nvidia.com/aiperf/latest/tutorials/load-patterns-scheduling/warmup-phase-configuration.md
43. https://www.itl.nist.gov/div898/software/dataplot/refman2/auxillar/coefvari.htm
44. https://www.usenix.org/system/files/atc20-shahrad.pdf
45. https://pypi.org/project/aiperf
46. https://github.com/mlcommons/inference_policies/blob/master/inference_rules.adoc
47. https://docs.anthropic.com/en/docs/build-with-claude/token-counting
48. https://huggingface.co/docs/transformers/tokenizer_summary
49. https://mlcommons.org/2024/03/mlperf-llama2-70b

THE PUBLISHER

About GPU Smith

GPU Smith provides independent engineering and research for private AI infrastructure. We help technical buyers and operators reason about GPU workload sizing, hardware selection, procurement, deployment and inference operations, from the compute system to the networking, power and cooling requirements around it.

Start with the workload

Useful infrastructure decisions begin with the models, data, throughput, latency and operating constraints a team actually has. GPU Smith helps connect those requirements with system design and deployment choices. The goal is a privately controlled AI environment whose capacity and operational demands are understood before a purchasing decision.

Hardware and supplier research

Our public reference library covers GPUs, complete systems and networking components. The server-vendor directory and manufacturing research help teams investigate suppliers, compare options and follow sources behind technical claims. We distinguish manufacturer specifications from measured benchmarks and advertised capabilities from tested configurations. Reference pages are research resources, not a live inventory listing or a binding equipment quote.

Deployment and operations

GPU Smith's engineering scope includes infrastructure integration, networking, power, cooling and the practical operation of inference workloads. Our published articles explain the tradeoffs behind deployment, procurement and ongoing operations so teams can ask better questions and document decisions.

Work with GPU Smith

Explore the hardware library, GPU references, networking references and vendor research. Contact GPU Smith with your workloads and deployment constraints to discuss a project and confirm current scope, pricing and availability.

Inclusion of a manufacturer or product in our research does not imply a partnership, certification or endorsement.

Website: https://gpusmith.com

This article is educational. Verify consequential medical, legal, financial, regulatory and technical decisions with appropriate professionals and the cited primary sources. Company services are described separately from the article's research findings.