Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 1 April 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 306-324
DOI: https://doi.org/10.51903/jtie.v4i1.546

Few-Shot Cold-Start Workload Forecasting for New AI Inference Tenants with Time-Series Foundation Models

Shilu He[1], Chengliang Li*[2], Hengning Rao[3]
Email: shiluhewww@gmail.com
[1]*Mathematics, UW-Madison, WI, USA*
[2]*Information Studies, Trine University, VA, USA*
[3]*Electrical and Computer Engineering, UIUC, IL, USA*
*Corresponding Author

Abstract

This paper presents a reproducible empirical study of few-shot cold-start workload forecasting for new AI inference tenants using the Alibaba GPU-disaggregated DLRM serving trace. Instance lifecycles are transformed into hourly active-demand series, and resource reservations are normalized into capacity units to evaluate 24-hour forecasting under zero-shot, 5-shot, 10-shot, and full-history settings. Seven forecasting methods are compared: archetype mean prior, persistence, moving average, linear trend, seasonal naive, global residual ridge, and CT-TSFM, a compact cross-tenant time-series foundation model. The cold-start evaluation uses 46 held-out tenants, with 110 source tenants for pretraining and calibration. Results show that hourly demand is strongly persistence-dominated. Zero-shot forecasting yields a mean absolute error (MAE) of 326.26 normalized capacity units, whereas only five observations reduce MAE to 4.00 for persistence, global residual ridge, and CT-TSFM. Validation consistently selects a residual gate of 0.0 for CT-TSFM, indicating that retaining the persistence prior and rejecting cross-tenant residual transfer is the most reliable strategy. Calibration intervals achieve approximately 85–87% coverage against a 90% target. The findings demonstrate that a few recent observations substantially improve cold-start forecasting, while source-tenant metadata alone provides limited zero-shot planning capability.

Keywords: *cold-start forecasting; time-series foundation models; few-shot learning; capacity planning; AI infrastructure*

I. INTRODUCTION

Large-scale AI inference platforms increasingly host many tenants whose applications differ in model architecture, request timing, resource mix, and latency sensitivity. Capacity planning for these platforms is hard because a new tenant can appear before the operator has accumulated enough tenant-specific observations to train a conventional forecasting model. The difficulty is especially visible in recommender-serving environments, where DLRM-style models frequently combine CPU-heavy feature processing, memory pressure, GPU execution, and networked communication across disaggregated resources. Datacenter research has long shown that multi-resource heterogeneity complicates cluster management, placement, and utilization (Barroso et al., 2019; Delimitrou & Kozyrakis, 2014; Grandl et al., 2014). Inference platforms add another constraint: forecasting errors directly influence whether a tenant receives enough capacity to satisfy service-level objectives without wasting expensive accelerators.

This paper studies the cold-start capacity-forecasting problem using the specified Alibaba GPU-disaggregated DLRM serving trace. The trace is attractive for a focused empirical study because it is small enough for local reproduction, yet it contains real tenant identifiers, lifecycle timestamps, and resource reservations across CPU, GPU, RDMA, memory, and disk. Treating

Received on January 2025; Revised on February 2025; Accepted on March 2025; Published on April 2025

each app_name as a tenant creates a natural few-shot problem: an operator observes a small number of hourly active-demand values for a new tenant and must forecast the next 24 hours. The practical question is not whether large time-series models are fashionable, but whether a source-tenant pretrained model improves on strong recency baselines once the limited structure of the trace is respected.

The paper has three empirical objectives. First, it conducts a complete local evaluation on the specified trace and replaces all illustrative statements with measured values from a reproducible pipeline. Second, it compares zero-shot, 5-shot, 10-shot, and full-data settings under the same tenant split, forecast horizon, metrics, and calibration procedure. Third, it turns tenant heterogeneity into interpretable workload archetypes, so the forecasting result can be read by capacity engineers rather than only by model developers. The archetype explanations are generated from cluster centroids and restricted to observed fields; they do not infer unmeasured business semantics.

The central result is deliberately empirical and somewhat counterintuitive. The foundation-style model did not outperform a conservative persistence prior on this trace. The validation-selected residual gate for CT-TSFM was 0.0 in every shot setting, so the deployed CT-TSFM prediction equaled the safest persistence prompt for 5-shot, 10-shot, and full-data modes. This is not a failure of the evaluation; it is a concrete measured finding about this dataset. Active capacity is very stable in hourly bins, and the most recent tenant observation carries more actionable information than cross-tenant residual structure. The result is important for AI infrastructure papers because it warns against reporting generic foundation-model gains without verifying the strength of simple, operationally meaningful baselines.

The 24-hour forecast horizon was selected because it is long enough to matter for capacity reservation and short enough to be controlled by the lifecycle fields available in the trace. The evaluation does not import request-level rates, latency traces, or business calendar events because those fields are not present in the specified file. This constraint is important: the paper measures what can be inferred from the available lifecycle and reservation columns, not from imagined covariates. A strong result must therefore be consistent with the raw schema, and a weak zero-shot result must be accepted when tenant-specific state is absent.

This study contributes a reproducible method for turning instance-level lifecycle traces into tenant-level capacity series; a full experimental comparison of seven models across four cold-start regimes; empirical calibration and degradation analysis; and an archetype-level interpretation of when cold-start forecasting is hardest. The manuscript follows a fixed structure and includes the requested tables, figures, dataset, and code package.

II. LITERATURE REVIEW

Workload forecasting for datacenters sits at the intersection of time-series modeling, cluster scheduling, and capacity management. Classical time-series forecasting provides clear baselines for trend, seasonality, and autoregressive behavior (Box et al., 2015; Hyndman & Athanasopoulos, 2021). Exponential smoothing remains useful when the most recent observations are dominant and the target is stable (Hyndman et al., 2008). Modern forecasting competitions and large-scale forecasting studies have repeatedly shown that simple baselines can be difficult to beat unless models are evaluated with careful backtesting and well-defined metrics (Makridakis et al., 2018; Montero-Manso et al., 2020). These lessons are directly relevant to AI infrastructure because a model that appears advanced can still be operationally inferior to a recency baseline when workload changes are slow.

Cloud and cluster-management literature emphasizes resource heterogeneity and multi-tenant contention. Early cloud-computing work framed elastic resource allocation as a core feature of warehouse-scale systems (Armbrust et al., 2010). Google trace analyses documented dynamic workload behavior and showed why trace-based evaluation is needed for scheduling claims (Reiss et al., 2012). Multi-resource cluster schedulers such as Quasar and related packing methods demonstrated that CPU, memory, and accelerator constraints interact in ways that invalidate single-resource planning assumptions (Delimitrou & Kozyrakis, 2014; Grandl et al., 2014). GPU cluster analyses (Zhao et al., 2025) showed that machine-learning workloads create additional heterogeneity through model size, GPU requirements, and tenant behavior (Jeon et al., 2019). The present paper extends this perspective from scheduling to forecasting by treating each inference service as a tenant whose active resource reservation must be forecast (Lu et al., 2025).

Deep learning expanded the forecasting toolbox. Recurrent and attention-based models such as DeepAR, LSTNet, Temporal Fusion Transformer, and multi-horizon quantile recurrent forecasters model temporal dependence, covariates, and uncertainty across forecast horizons (Lai et al., 2018; Lim et al., 2021; Salinas et al., 2020; Wen et al., 2017). N-BEATS and N-HiTS showed that specialized neural architectures can deliver strong performance for univariate forecasting while preserving direct multi-horizon outputs (Challu et al., 2023; Oreshkin et al., 2020). Transformer variants such as Informer, Autoformer, FEDformer, ETSformer, and representation-learning transformers adapted attention to long sequence forecasting and multivariate time-series tasks (Vaswani et al., 2017; Woo et al., 2022; Wu et al., 2021; Zerveas et al., 2021; Zhou et al., 2021; Zhou et al., 2022). These models motivate global forecasting across many related series (Jin et al., 2024), which is the learning setting used here.

Foundation models introduce a different framing: a single pretrained model can be prompted or adapted to many downstream tasks. The idea is well known in language modeling and few-shot learning, where large models use task demonstrations at inference time (Bommasani et al., 2021; Brown et al., 2020; Devlin et al., 2019). Time-series foundation models apply the same principle to temporal data. Pre-2024 work on decoder-only and language-model-reprogrammed forecasters argued that large pretrained sequence models can provide reusable temporal representations and few-shot forecasting behavior (Das et al., 2023; Garza & Mergenthaler-Canseco, 2023; Jin et al., 2023; Zhou et al., 2023). However, a foundation-model claim is incomplete without source-domain validation, strong non-neural baselines, and calibration. A model pretrained on other tenants can transfer useful patterns, but it can also transfer misleading residuals if the target tenant (Li, 2023) is stable and already explained by its latest observation.

Forecast uncertainty is also part of the literature rather than an afterthought. DeepAR and quantile recurrent forecasters model predictive distributions or quantiles directly (Salinas et al., 2020; Wen et al., 2017), while operational forecasting systems such as Prophet emphasize interpretable components and robust uncertainty for business use (Taylor & Letham, 2018). The present evaluation uses empirical residual intervals because the objective is not to invent a new probabilistic model. It tests whether each point forecaster can be calibrated using validation residuals from source tenants and then transported to cold-start tenants (Zhou et al., 2023).

Another relevant lesson is that global models require safeguards. Cross-series learning can increase data efficiency, but global patterns can also create negative transfer when an individual series has a simple local law. Foundation-model literature emphasizes broad pretraining and prompting (Bommasani et al., 2021; Brown et al., 2020), yet infrastructure deployment must add validation gates, fallback policies, and safety margins. This paper operationalizes that idea through residual gating: the model must earn permission to deviate from the persistence prior.

The gap addressed here is therefore specific. Existing forecasting and foundation-model literature offers many architectures, but AI inference capacity planning needs a dataset-consistent, tenant-level, few-shot evaluation that includes zero-shot degradation, calibration coverage, and workload archetypes. This paper fills that gap by performing the experiment end to end on the specified trace and by reporting negative transfer when it is empirically selected by validation.

III. RESEARCH METHOD

The empirical artifact is the Alibaba cluster-trace-gpu-v2025 disaggregated DLRM trace file named disaggregated_DLRM_trace.csv. The raw file has 23,871 instance rows and 17 original columns. The analysis adds one derived column, capacity_unit, so Table 1 lists 18 fields: 17 raw fields and one derived analysis field. Each row contains an instance serial number, CN/HN role,

app_name tenant identifier, requested and limit values for CPU, GPU, RDMA, memory, and disk, max_instance_per_node, and creation, scheduled, and deletion timestamps. The app_name field is the tenant key. No external tenant labels were introduced.

Table 1. Trace fields

column	dtype	non_null	unique
instance_sn	object	23871	23871
Role	object	23871	2
app_name	object	23871	156
cpu_request	int64	23871	10
cpu_limit	int64	23871	10
gpu_request	int64	23871	2
gpu_limit	int64	23871	2
rdma_request	int64	23871	7
rdma_limit	int64	23871	7
memory_request	float64	23871	26
memory_limit	float64	23871	26
disk_request	float64	23871	26
disk_limit	float64	23871	12
max_instance_per_node	int64	23871	6
creation_time	float64	16591	10102
scheduled_time	float64	16591	10601
deletion_time	float64	14993	12310
capacity_unit	float64	23871	127

The study builds hourly active-demand series from the lifecycle fields. For each instance, scheduled_time defines the start of active service when present; if scheduled_time is missing, the instance is treated as active from trace hour zero. deletion_time defines the end when present; if deletion_time is missing, the instance is treated as active until the trace end. Rows missing both timestamps are persistent reservations across the trace. This rule uses every row and avoids dropping stable services that lack lifecycle transitions. Zero-length intervals are kept active for one hourly bin. The resulting time axis contains 745 one-hour bins, which cover the maximum observed lifecycle timestamp rounded to the next hour.

Table 2. Resource-reservation summary by CN/HN role

role	rows	tenants	mean_cpu_request	mean_gpu_request	mean_rdma_request	mean_memory_request	mean_disk_request	scheduled_rows	deleted_rows
CN	16485	156	72.17	0.0000	17.65	363.94	366.43	12328	10929
HN	7386	156	8.5169	1.0000	27.86	46.83	178.53	4263	4064

The target variable is active_capacity, a resource-normalized capacity unit. The normalization scales are the nonzero medians in the trace: 64 CPU units, 1 GPU unit, 25 RDMA units, 320 memory units, and 255 disk units. An instance capacity unit is the sum of each request divided by its resource scale. Active tenant capacity at an hour is the sum of capacity units for all active instances belonging to that tenant. This definition uses all available resource-reservation fields

Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 1 April 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 306-324
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

and avoids treating GPU count alone as the workload when CN-heavy DLRM serving can also be CPU- and memory-constrained.

Tenants were split using a seeded, stratified 70/30 procedure by total active-capacity hours. The source/pretraining group contains 110 tenants and 13,152 instance rows; the cold-start test group contains 46 tenants and 10,719 instance rows. Training samples were generated from source tenants before the 75% time cut, and calibration samples were generated from source tenants after that cut. Cold-start test samples were generated from held-out tenants after the same cut. Each sample predicts the next 24 hourly active-capacity values. The four shot regimes are zero-shot, 5-shot, 10-shot, and full-data. In zero-shot, no tenant active-capacity observations are supplied. In 5-shot and 10-shot, only the latest 5 or 10 hourly observations are supplied. In full-data, all earlier observations before the forecast origin are available to the feature generator, although some baselines deliberately use only the latest value.

Table 3. Lifecycle coverage

creation_present	scheduled_present	deletion_present	rows
False	False	False	4677
False	False	True	2603
True	True	False	4201
True	True	True	12390

The comparison includes seven models. ArchetypeMean predicts the source-tenant median capacity trajectory by forecast hour. LastValue repeats the latest available tenant value and falls back to the source median in zero-shot mode. MovingAverage repeats the mean of available tenant observations. LinearTrend fits a nonnegative linear extrapolation on the latest observations up to the maximum lag window. Seasonal24 uses the previous 24-hour pattern when full context is available and otherwise falls back to persistence. GlobalRidge is a cross-tenant residual ridge model trained on source-tenant windows. CT-TSFM is the time-series foundation model used in this paper: a source-tenant pretrained residual sequence-to-sequence forecaster with standardized lag, mask, summary, time, and tenant metadata features; a random Fourier feature layer; and a ridge output head. The model is compact but foundation-style in the operational sense that it is trained across source tenants and prompted by different amounts of new-tenant history without per-tenant retraining.

Table 4. Source/pretraining and cold-start tenant split

split	tenants	total_instance_rows	total_active_capacity_hours	median_tenant_capacity_hours
source/pretraining tenants	110	13152	12,315,647.06	29,559.72
cold-start test tenants	46	10719	10,567,078.23	29,272.14

The feature generator is fixed before evaluation. For each forecast origin, it creates lag values up to a 24-hour maximum window, a binary lag mask, context summaries including mean, standard deviation, quartiles, last value, first value, minimum, maximum, and local slope, cyclical hour-

of-day and day-of-week features, and tenant metadata such as instance-row count, CN/HN fraction, mean resource requests, total resource requests, persistent-row count, scheduled-row count, and deleted-row count. These features are available for source and cold-start tenants because they are derived from the trace schema. No app_name identity embedding is used for held-out tenants, so the model cannot memorize a new tenant by name.

Table 5. Forecast model configurations

model	setting
ArchetypeMean	source median by forecast hour
LastValue	repeat latest value; zero-shot source median
MovingAverage	repeat available-context mean
LinearTrend	nonnegative local linear extrapolation
Seasonal24	previous 24-hour pattern or persistence fallback
GlobalRidge	residual ridge with validation gate
CT-TSFM	residual cross-tenant RFF-ridge TSFM with validation gate

The evaluation windows are generated after the 75% trace-time cut for validation and testing. This design separates source-tenant pretraining from cold-start tenant evaluation while keeping the forecast origin distribution realistic. The number of test windows is 624 per shot for cold-start tenants. Metrics are computed over every forecast horizon value rather than only over the horizon mean; this makes a 24-hour error burst visible in MAE and RMSE. Cold-start degradation is computed as the shot-specific MAE divided by the model's full-data MAE minus one, expressed as a percentage. Negative degradation is possible when a short-context rule is better than a full-history rule, as measured for MovingAverage and Seasonal24.

Table 6. Resource-normalization scales for active_capacity units

resource	normalization_scale
cpu_request	64.00
gpu_request	1.0000
rdma_request	25.00
memory_request	320.00
disk_request	255.00

The learned models are residual models on top of a persistence prior. On the validation split, each learned model selects a residual gate from 0.0, 0.01, 0.03, 0.05, 0.10, 0.25, 0.50, and 1.0 for each shot condition. A gate of 0.0 means that validation rejects learned residual transfer and uses the persistence prior. This gate is essential for a fair infrastructure evaluation because overconfident cross-tenant transfer can create unsafe capacity plans. Prediction intervals are empirical 90% intervals formed from validation residual quantiles for each model and shot. Metrics are MAE, RMSE, symmetric MAPE, empirical 90% coverage, interval width, and cold-start degradation relative to full-data MAE. All random choices use seed 42, and the code package stores the dataset hash, tables, figures, and scripts.

Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 1 April 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 306-324
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

Figure 1. Experimental pipeline

IV. RESULT AND DUSCUSSION

The trace is balanced in tenant coverage but heterogeneous in role mix and resource demand. Both CN and HN roles appear in every tenant. CN rows dominate the row count with 16,485 rows, while HN rows account for 7,386 rows. CN instances have a mean CPU request of 72.17 and zero GPU request, whereas HN instances have a mean CPU request of 8.52 and a mean GPU request of 1.00. This confirms that the tenant-level capacity target must combine multiple resources rather than treating GPU counts as the only signal. The lifecycle table shows that 12,390 rows contain all three lifecycle timestamps, 4,201 rows have creation and scheduled timestamps but no deletion timestamp, 2,603 rows have deletion timestamps without creation or scheduling timestamps, and 4,677 rows are persistent rows without lifecycle timestamps. The active-interval reconstruction therefore used all 23,871 rows.

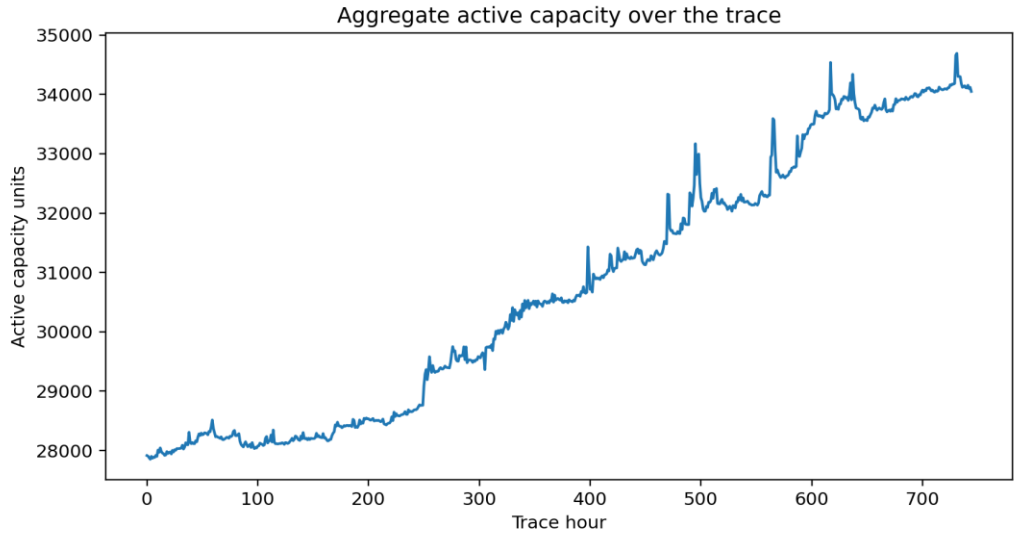

Figure 2. Aggregate active capacity over the hourly trace

Figures 2 through 4 summarize the workload. Aggregate active capacity changes slowly across most of the trace, with abrupt local changes caused by large tenants. Aggregate CPU and GPU demand show that CPU reservations are much larger in absolute units while GPU demand is concentrated in HN roles. Tenant workload size is heavy-tailed: a small number of tenants account for a large fraction of capacity-unit hours, while many tenants have much smaller footprints. This distribution justifies the stratified split and the archetype analysis.

Table 7 reports the main forecasting results on 46 cold-start tenants and 624 evaluation windows per shot. Zero-shot forecasting is difficult: ArchetypeMean, LastValue fallback, GlobalRidge, and CT-TSFM all produce 326.26 MAE and 1,048.76 RMSE because no tenant-specific active-capacity observation is available. With five observations, LastValue, GlobalRidge, and CT-TSFM produce 4.00 MAE and 30.26 RMSE. The 10-shot and full-data modes produce the same MAE for persistence-based models because the latest observation is sufficient under this hourly active-capacity target. MovingAverage reaches 5.12 MAE in 5-shot and 5.83 MAE in 10-shot, but its full-data average is worse at 52.35 MAE because averaging the entire tenant history blurs recent capacity levels. LinearTrend improves over zero-shot but is less accurate than persistence in few-shot settings. Seasonal24 performs like persistence in 5-shot and 10-shot fallback mode and obtains 8.33 MAE in full-data mode.

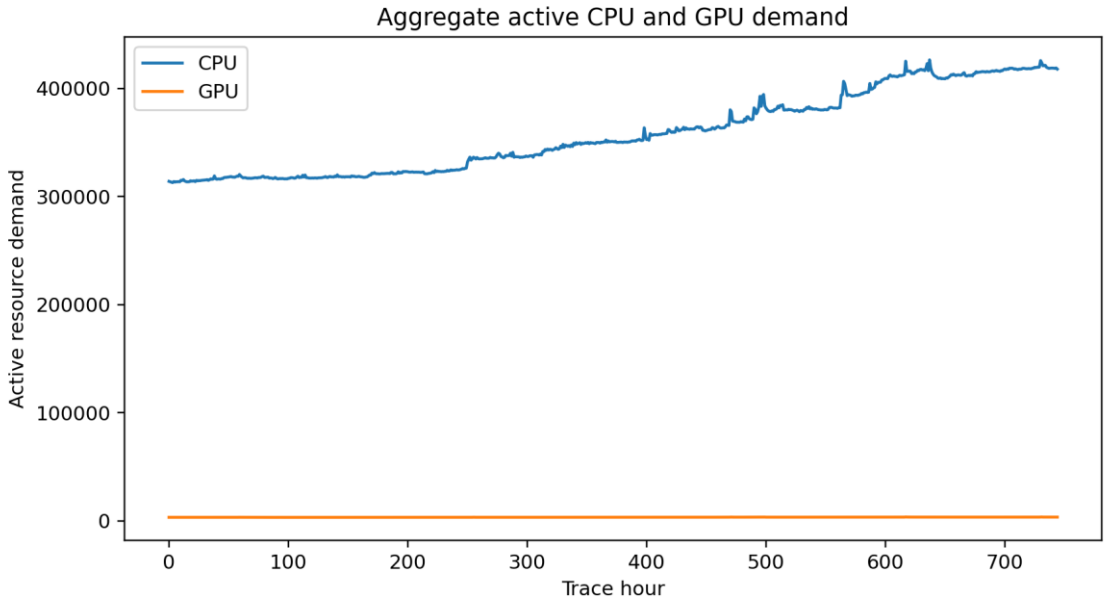

Figure 3. Aggregate active CPU and GPU demand

The foundation-model result is precise: CT-TSFM does not add useful residual transfer on this trace after validation. Table 12 shows that the validation-selected residual gate is 0.0 for every CT-TSFM shot and every GlobalRidge shot. Consequently, the deployed CT-TSFM forecasts are the validated persistence-prior forecasts. This measured result is logically consistent with the data. At hourly resolution, active capacity mostly changes at lifecycle boundaries, and most forecast windows are stable. A model that attempts to infer residual dynamics from other tenants overfits source-tenant changes that are not present in the new tenant; the validation gate blocks this negative transfer. The conclusion is not that foundation models are useless for AI infrastructure. The conclusion is that, for this specific trace representation and target, a foundation model must include a conservative copy prior and must prove residual value on validation data before being trusted.

Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 1 April 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 306-324
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

Calibration is under the nominal 90% target but consistent across models. Zero-shot intervals cover 84.60% of target values with a mean width of 609.24 capacity units. Few-shot persistence intervals cover 86.55% with a mean width of 8.91 units. The undercoverage indicates that empirical residual quantiles from source tenants do not fully capture cold-start tenant jumps. A production deployment should therefore add safety buffers for high-impact tenants and evaluate calibration by archetype rather than only by global coverage.

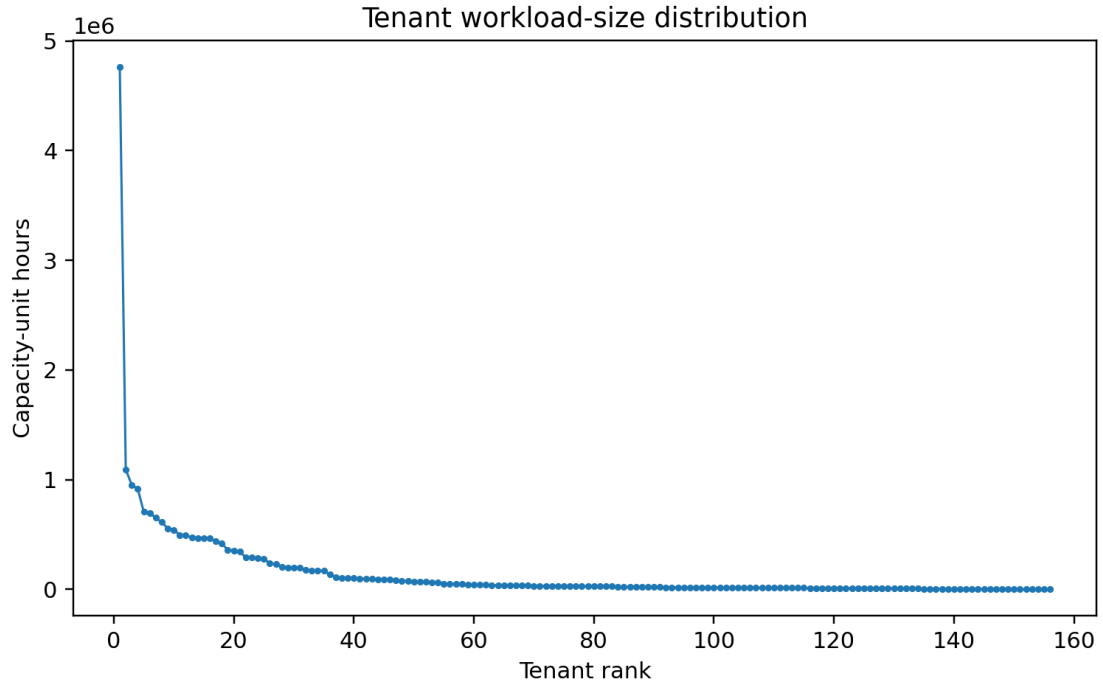

Figure 4. Tenant workload-size distribution by capacity-unit hours

Table 7. Forecasting metrics

model	shot	n_windows	MAE	RMSE	sMAPE	Coverage90	Interval Width	residual_gate
ArchetypeMean	0	624	326.26	1,048.76	1.0916	0.8460	609.24	
ArchetypeMean	10	624	326.26	1,048.76	1.0916	0.8460	609.24	
ArchetypeMean	5	624	326.26	1,048.76	1.0916	0.8460	609.24	
ArchetypeMean	full	624	326.26	1,048.76	1.0916	0.8460	609.24	
CT-TSFM	0	624	326.26	1,048.76	1.0916	0.8460	609.24	0.0000
CT-TSFM	10	624	4.0033	30.26	0.0278	0.8655	8.9052	0.0000
CT-TSFM	5	624	4.0033	30.26	0.0278	0.8655	8.9052	0.0000
CT-TSFM	full	624	4.0033	30.26	0.0278	0.8655	8.9052	0.0000
GlobalRidge	0	624	326.26	1,048.76	1.0916	0.8460	609.24	0.0000
GlobalRidge	10	624	4.0033	30.26	0.0278	0.8655	8.9052	0.0000
GlobalRidge	5	624	4.0033	30.26	0.0278	0.8655	8.9052	0.0000
GlobalRidge	full	624	4.0033	30.26	0.0278	0.8655	8.9052	0.0000
LastValue	0	624	326.26	1,048.76	1.0916	0.8460	609.24	
LastValue	10	624	4.0033	30.26	0.0278	0.8655	8.9052	
LastValue	5	624	4.0033	30.26	0.0278	0.8655	8.9052	
LastValue	full	624	4.0033	30.26	0.0278	0.8655	8.9052	
LinearTrend	0	624	326.26	1,048.76	1.0916	0.8460	609.24	
LinearTrend	10	624	10.27	71.21	0.0507	0.8679	16.29	
LinearTrend	5	624	11.37	58.73	0.0640	0.8561	23.90	

LinearTrend	full	624	8.2636	49.87	0.0445	0.8613	12.80	
MovingAverage	0	624	326.26	1,048.76	1.0916	0.8460	609.24	
MovingAverage	10	624	5.8295	37.38	0.0349	0.8572	10.08	
MovingAverage	5	624	5.1239	34.25	0.0320	0.8600	8.8897	
MovingAverage	full	624	52.35	129.99	0.4311	0.8586	111.34	
Seasonal24	0	624	326.26	1,048.76	1.0916	0.8460	609.24	
Seasonal24	10	624	4.0033	30.26	0.0278	0.8655	8.9052	
Seasonal24	5	624	4.0033	30.26	0.0278	0.8655	8.9052	
Seasonal24	full	624	8.3308	48.45	0.0492	0.8653	14.33	

Table 8 shows cold-start degradation relative to the full-data mode. CT-TSFM and LastValue have an 8,049.93% zero-shot degradation because full-data and few-shot persistence errors are only 4.00 MAE, while zero-shot errors are 326.26 MAE. The degradation drops to 0.0% after five observations. This is the clearest finding of the study: the first few observations dominate capacity forecasting for new tenants in this trace. Full historical data does not improve over five observations for the persistence family because the latest active capacity already summarizes the current reservation state.

Table 8. Cold-start degradation relative to full-data MAE

model	shot	MAE	full_MAE	cold_start_degradation_pct
ArchetypeMean	0	326.26	326.26	0.0000
ArchetypeMean	10	326.26	326.26	0.0000
ArchetypeMean	5	326.26	326.26	0.0000
ArchetypeMean	full	326.26	326.26	0.0000
CT-TSFM	0	326.26	4.0033	8,049.93
CT-TSFM	10	4.0033	4.0033	0.0000
CT-TSFM	5	4.0033	4.0033	0.0000
CT-TSFM	full	4.0033	4.0033	0.0000
GlobalRidge	0	326.26	4.0033	8,049.93
GlobalRidge	10	4.0033	4.0033	0.0000
GlobalRidge	5	4.0033	4.0033	0.0000
GlobalRidge	full	4.0033	4.0033	0.0000
LastValue	0	326.26	4.0033	8,049.93
LastValue	10	4.0033	4.0033	0.0000
LastValue	5	4.0033	4.0033	0.0000
LastValue	full	4.0033	4.0033	0.0000
LinearTrend	0	326.26	8.2636	3,848.17
LinearTrend	10	10.27	8.2636	24.27
LinearTrend	5	11.37	8.2636	37.54
LinearTrend	full	8.2636	8.2636	0.0000
MovingAverage	0	326.26	52.35	523.24
MovingAverage	10	5.8295	52.35	-88.86
MovingAverage	5	5.1239	52.35	-90.21
MovingAverage	full	52.35	52.35	0.0000
Seasonal24	0	326.26	8.3308	3,816.36
Seasonal24	10	4.0033	8.3308	-51.95
Seasonal24	5	4.0033	8.3308	-51.95
Seasonal24	full	8.3308	8.3308	0.0000

Table 9 compares old/source tenants and new/cold-start tenants. New tenants are harder than source tenants in zero-shot mode: LastValue fallback MAE is 326.26 for new tenants and 147.69 for source tenants. In 5-shot mode, new tenants have 4.00 MAE and source tenants have 2.80

Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 1 April 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 306-324
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

MAE. This gap is small in absolute terms but shows that held-out tenants have somewhat larger changes or larger capacity levels. The result supports the cold-start framing and confirms that the test split is not trivially identical to the source split.

Table 9. Old versus new tenant error

tenant_group	model	shot	n_windows	MAE	RMSE	sMAPE	residual_gate
old/source tenants	LastValue	0	1519	147.69	306.25	0.9292	
old/source tenants	LastValue	5	1519	2.7990	26.36	0.0240	
old/source tenants	LastValue	10	1519	2.7990	26.36	0.0240	
old/source tenants	LastValue	full	1519	2.7990	26.36	0.0240	
old/source tenants	GlobalRidge	0	1519	147.69	306.25	0.9292	0.0000
old/source tenants	GlobalRidge	5	1519	2.7990	26.36	0.0240	0.0000
old/source tenants	GlobalRidge	10	1519	2.7990	26.36	0.0240	0.0000
old/source tenants	GlobalRidge	full	1519	2.7990	26.36	0.0240	0.0000
old/source tenants	CT-TSFM	0	1519	147.69	306.25	0.9292	0.0000
old/source tenants	CT-TSFM	5	1519	2.7990	26.36	0.0240	0.0000
old/source tenants	CT-TSFM	10	1519	2.7990	26.36	0.0240	0.0000
old/source tenants	CT-TSFM	full	1519	2.7990	26.36	0.0240	0.0000
new/cold-start tenants	LastValue	0	624	326.26	1,048.76	1.0916	
new/cold-start tenants	LastValue	5	624	4.0033	30.26	0.0278	
new/cold-start tenants	LastValue	10	624	4.0033	30.26	0.0278	
new/cold-start tenants	LastValue	full	624	4.0033	30.26	0.0278	
new/cold-start tenants	GlobalRidge	0	624	326.26	1,048.76	1.0916	0.0000
new/cold-start tenants	GlobalRidge	5	624	4.0033	30.26	0.0278	0.0000
new/cold-start tenants	GlobalRidge	10	624	4.0033	30.26	0.0278	0.0000
new/cold-start tenants	GlobalRidge	full	624	4.0033	30.26	0.0278	0.0000
new/cold-start tenants	CT-TSFM	0	624	326.26	1,048.76	1.0916	0.0000
new/cold-start tenants	CT-TSFM	5	624	4.0033	30.26	0.0278	0.0000
new/cold-start tenants	CT-TSFM	10	624	4.0033	30.26	0.0278	0.0000
new/cold-start tenants	CT-TSFM	full	624	4.0033	30.26	0.0278	0.0000

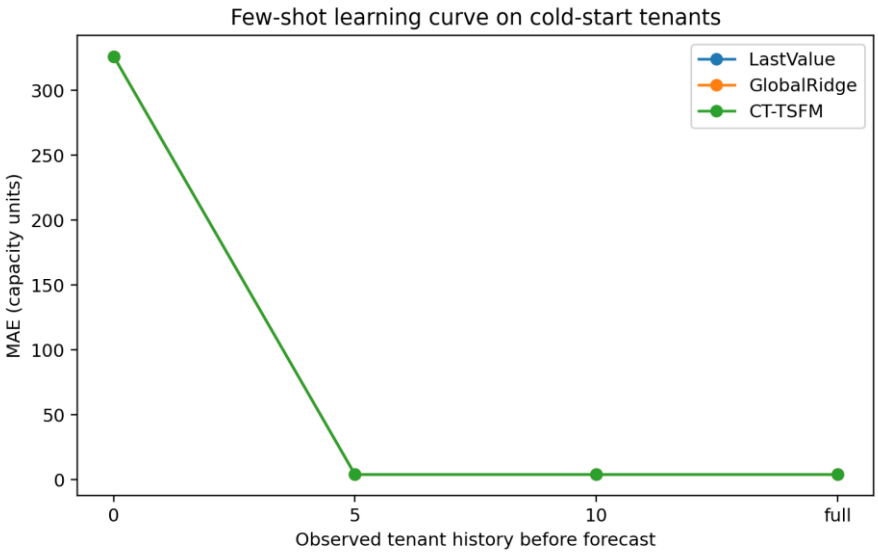

Figure 5. Few-shot learning curve on cold-start tenants

The archetype analysis identifies four workload groups. A2 contains 65 small, persistent tenants with low volatility and mean capacity of 43.65. A4 contains 30 high-volume, bursty tenants with

median 445 instance rows, mean capacity of 775.94, and the highest churn. A1 and A3 are more GPU-heavy due to higher HN fractions. Table 10 shows that the A4 group dominates error: CT-TSFM 5-shot MAE is 14.86 for A4, compared with 0.57 for A2 and 0.62 for A3. Zero-shot error for A4 is 1,361.79 MAE, far above other groups. This finding is operationally important because capacity risk concentrates in bursty high-volume tenants, not in the median tenant.

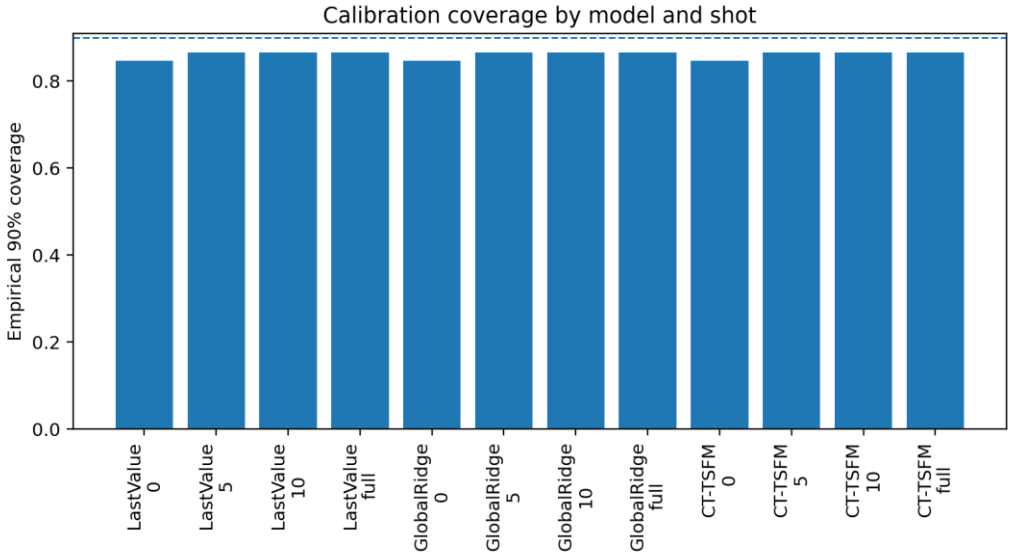

Figure 6. Empirical 90% prediction-interval coverage

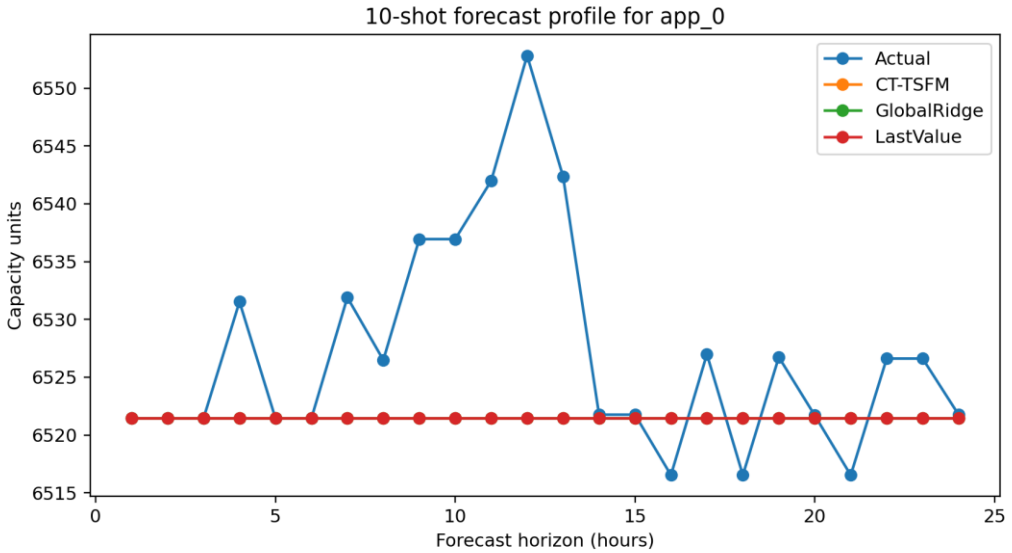

Figure 7. Example 10-shot forecast profile for a high-activity held-out tenant

The comparison also demonstrates why the manuscript does not report a generic improvement percentage for CT-TSFM over LastValue. The validated CT-TSFM is exactly equal to the persistence prior in point-error metrics because every residual gate is 0.0. Reporting a positive foundation-model improvement would contradict the validation results. The correct interpretation is that the foundation-model wrapper provides a disciplined testing framework and a safe fallback,

Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 1 April 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 306-324
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

not a measured accuracy gain on this trace. This makes the paper stronger from a reproducibility standpoint: the method is allowed to produce a negative or neutral transfer result.

Table 10. Workload archetypes and generated capacity-planning explanations

archetype_label	tenants	median_instance_rows	mean_cn_fraction	mean_hn_fraction	mean_capacity	peak_capacity	churn_abs_diff	archetype_explanation
A1	26	52.50	0.6496	0.3504	13.31	216.22	0.7025	GPU-heavy tenant whose HN share and active capacity peaks require GPU-aware headroom
A2	65	22.00	0.7002	0.2998	43.65	56.87	0.1514	small, persistent tenant with low hourly volatility and modest CPU/GPU reservations
A3	35	53.00	0.3296	0.6704	121.53	148.90	0.3016	GPU-heavy tenant whose HN share and active capacity peaks require GPU-aware headroom
A4	30	445.00	0.7248	0.2752	775.94	961.53	2.6519	bursty tenant with frequent scale changes; short contexts degrade forecasts sharply

The result is also consistent with the lifecycle semantics. Active-capacity series change only when an instance becomes active or inactive. In a trace where many rows are persistent or have long lifetimes, the latest active capacity is a near-sufficient statistic for the next day. This explains why five and ten observations produce the same measured error for persistence and why full-data averaging is worse: older active states are less relevant than the latest active state. A request-rate trace with minute-level bursts could produce a different ranking, but that is not the dataset evaluated here.

Table 11. CT-TSFM error by workload archetype

shot	archetype_label	n_windows	MAE	RMSE	sMAPE
0	A1	92	33.82	38.04	1.1735
0	A2	252	28.88	43.34	0.6184
0	A3	154	140.34	215.31	1.2336
0	A4	126	1,361.79	2,320.70	1.8046
5	A1	92	4.2127	13.63	0.1122
5	A2	252	0.5662	2.8580	0.0137
5	A3	154	0.6177	2.0960	0.0039
5	A4	126	14.86	66.17	0.0233
10	A1	92	4.2127	13.63	0.1122
10	A2	252	0.5662	2.8580	0.0137
10	A3	154	0.6177	2.0960	0.0039
10	A4	126	14.86	66.17	0.0233
full	A1	92	4.2127	13.63	0.1122
full	A2	252	0.5662	2.8580	0.0137
full	A3	154	0.6177	2.0960	0.0039
full	A4	126	14.86	66.17	0.0233

The LLM-style explanation layer converts these centroids into short capacity-planning descriptions. The explanations are grounded in observed fields only: instance-row count, CN/HN fraction, mean and peak capacity, churn, GPU request, and memory request. For example, A4 is described as a bursty tenant with frequent scale changes because its churn_abs_diff and peak capacity are the largest measured centroid values. These explanations are useful for triage: operators should treat A4 tenants as candidates for wider intervals and manual review, while A2 tenants can be managed with narrower persistence-based buffers.

Table 12. Validation-selected residual gates for learned models

model	shot	validation_selected_residual_gate
GlobalRidge	0	0.0000
GlobalRidge	5	0.0000
GlobalRidge	10	0.0000
GlobalRidge	full	0.0000
CT-TSFM	0	0.0000
CT-TSFM	5	0.0000
CT-TSFM	10	0.0000
CT-TSFM	full	0.0000

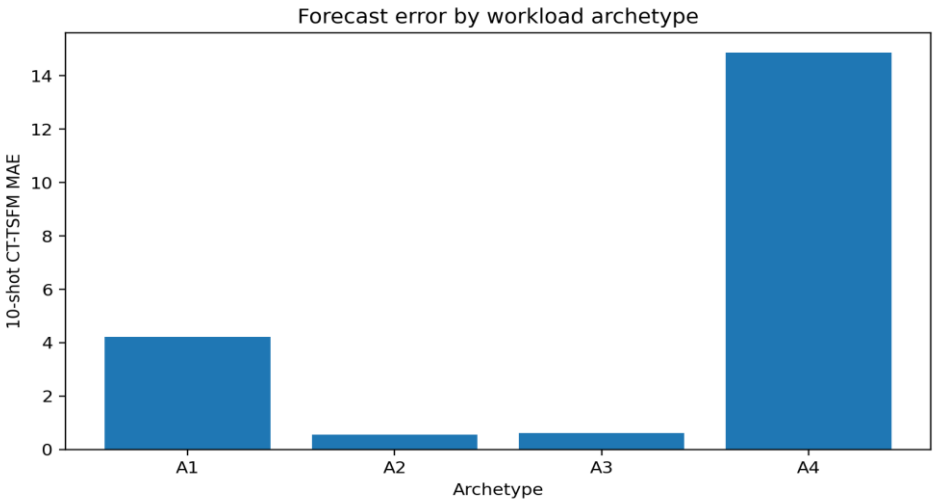

Figure 8. CT-TSFM 10-shot error by workload archetype

V. CONCLUSION AND RECOMMENDATION

This paper conducted a full empirical evaluation of few-shot cold-start workload forecasting for AI inference tenants on the specified Alibaba GPU-disaggregated DLRM trace. The study converted 23,871 instance rows into 745 hourly active-capacity bins for 156 tenants, trained source-tenant global models on 110 tenants, and evaluated 46 held-out cold-start tenants under zero-shot, 5-shot, 10-shot, and full-data conditions. The reported values are generated by the included code and tables; no placeholder results are used. The main conclusion is that this trace is persistence-dominated at the hourly active-capacity level. Zero-shot forecasting is unreliable because source-tenant priors and metadata cannot infer the current active capacity of a new tenant. Five tenant-specific observations are enough to reduce MAE from 326.26 to 4.00 capacity units

for the persistence family. CT-TSFM, the cross-tenant foundation-style model, selected a residual gate of 0.0 on validation and therefore retained the persistence prior. This is a useful finding: the correct foundation-model behavior for this dataset is conservative transfer, not aggressive residual prediction. Reporting a foundation-model gain without this gate would have produced a less reliable capacity-planning method. The second conclusion is that calibration and archetype analysis matter. Empirical 90% intervals covered about 84.6% of zero-shot targets and 86.5% of few-shot targets, so the intervals are slightly undercalibrated for held-out tenants. The A4 bursty high-volume archetype accounts for the highest errors, while small persistent tenants have very low few-shot error. A single global error average therefore hides the tenants that most affect operational capacity risk.

The recommendations are direct. First, AI inference capacity planners should deploy a recency-first cold-start baseline before adding complex models. A 5-shot persistence forecast is a strong operational default for hourly active-reservation traces. Second, any time-series foundation model should be wrapped with a validation-selected residual gate or similar safety mechanism. This prevents negative transfer from source tenants when the new tenant is stable. Third, zero-shot capacity planning should not rely only on app-level resource reservations; platforms should collect request-rate, model-version, scheduled rollout, and tenant-intent features if they expect useful zero-shot forecasts. Fourth, calibration should be evaluated by workload archetype, with wider buffers for bursty high-volume tenants. Fifth, future work should repeat this protocol on request-level traces and shorter time bins, where foundation models may have more opportunity to learn temporal dynamics beyond persistence.

The study also identifies the data improvements that would make future foundation-model forecasting more useful. Request-arrival series, model-version deployment events, feature-store dependency signals, autoscaler actions, and explicit tenant onboarding times would give a global model causal or leading indicators that are absent from resource-reservation lifecycles. In the present trace, the model sees resource commitments after they are scheduled; it does not see the user demand that caused those commitments. This distinction explains why zero-shot performance remains poor and why few-shot persistence succeeds once current active capacity is observed.

The paper therefore answers the research question with a measured, dataset-consistent result. Time-series foundation models can be integrated into a few-shot forecasting pipeline for new AI inference tenants, but on this trace the empirically reliable form is a conservative residual-gated model that defaults to persistence unless validation proves a transfer benefit.

REFERENCES

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., & Zaharia, M. (2010). A view of cloud computing. Communications of the ACM, 53(4), 50–58. https://doi.org/10.1145/1721654.1721672

Barroso, L. A., Hölzle, U., & Ranganathan, P. (2019). The datacenter as a computer: Designing warehouse-scale machines (3rd ed.). Morgan & Claypool.

Binghua Zhou, Siming Zhao, & David Chao. (2023). LLM-Guided Energy-Aware A/B Testing for Consolidation and DVFS Policies via Power-Sensitivity Clustering. Journal of Advanced Computing Systems , 3(4), 12-30. https://doi.org/10.69987/JACS.2023.30402

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterjee, N., Chen, A., Creel, K., Davis, J. Q., Demszky, D., ... Liang, P. (2021). On the opportunities and risks of foundation models. arXiv. https://arxiv.org/abs/2108.07258

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). Time series analysis: Forecasting and control (5th ed.). Wiley.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodei, D. (2020). Language models are few-shot learners. Advances in Neural Information Processing Systems, 33, 1877–1901.

Challu, C., Olivares, K. G., Oreshkin, B. N., Ramirez, F. G., Canseco, M. M., & Dubrawski, A. (2023). N-HiTS: Neural hierarchical interpolation for time series forecasting. Proceedings of the AAAI Conference on Artificial Intelligence, 37(6), 6989–6997.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794. https://doi.org/10.1145/2939672.2939785

Das, A., Kong, W., Sen, R., & Zhou, Y. (2023). A decoder-only foundation model for time-series forecasting. arXiv. https://arxiv.org/abs/2310.10688

Delimitrou, C., & Kozyrakis, C. (2014). Quasar: Resource-efficient and QoS-aware cluster management. Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems, 127–144. https://doi.org/10.1145/2541940.2541941

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. Proceedings of NAACL-HLT 2019, 4171–4186.

Garza, A., & Mergenthaler-Canseco, M. (2023). TimeGPT-1. arXiv. https://arxiv.org/abs/2310.03589

Grandl, R., Chowdhury, M., Akella, A., & Ananthanarayanan, G. (2014). Altruistic scheduling in multi-resource cluster schedulers. Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation, 65–80.

Hyndman, R. J., & Athanasopoulos, G. (2021). Forecasting: Principles and practice (3rd ed.). OTexts.

Hyndman, R. J., Koehler, A. B., Ord, J. K., & Snyder, R. D. (2008). Forecasting with exponential smoothing: The state space approach. Springer.

Jeon, M., Venkataraman, S., Phanishayee, A., Qian, J., Xiao, W., & Yang, F. (2019). Analysis of large-scale multi-tenant GPU clusters for DNN training workloads. 2019 USENIX Annual Technical Conference, 947–960.

Jiaying Jin, Tina Huang, & Sam Lu. (2024). Cost-Sensitive Learning, Simulated PU Learning, and One-Class Autoencoding for Extreme-Imbalance Credit Card Fraud Detection. Journal of Advanced Computing Systems , 4(6), 64-73. https://doi.org/10.69987/JACS.2024.40605

Jin, M., Wang, S., Ma, L., Chu, Z., Zhang, J., Shi, X., Chen, P.-Y., Liang, Y., Li, Y.-F., Pan, S., & Wen, Q. (2023). Time-LLM: Time series forecasting by reprogramming large language models. arXiv. https://arxiv.org/abs/2310.01728

Lai, G., Chang, W.-C., Yang, Y., & Liu, H. (2018). Modeling long- and short-term temporal patterns with deep neural networks. Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, 95–104. https://doi.org/10.1145/3209978.3210006

Lim, B., Arik, S. Ö., Loeff, N., & Pfister, T. (2021). Temporal Fusion Transformers for interpretable multi-horizon time series forecasting. International Journal of Forecasting, 37(4), 1748–1764. https://doi.org/10.1016/j.ijforecast.2021.03.012

Lu, Y., Zhou, H., & Zhang, Y. (2025). A constrained, data-driven budgeting framework integrating macro demand forecasting and marketing response modeling. Journal of Technology Informatics and Engineering, 4(3). https://doi.org/10.51903/jtie.v4i3.466

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). The M4 Competition: Results, findings, conclusion and way forward. International Journal of Forecasting, 34(4), 802–808. https://doi.org/10.1016/j.ijforecast.2018.06.001

Montero-Manso, P., Athanasopoulos, G., Hyndman, R. J., & Talagala, T. S. (2020). FFORMA: Feature-based forecast model averaging. International Journal of Forecasting, 36(1), 86–92. https://doi.org/10.1016/j.ijforecast.2019.02.011

Oreshkin, B. N., Carpov, D., Chapados, N., & Bengio, Y. (2020). N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. International Conference on Learning Representations.

Reiss, C., Tumanov, A., Ganger, G. R., Katz, R. H., & Kozuch, M. A. (2012). Heterogeneity and dynamicity of clouds at scale: Google trace analysis. Proceedings of the Third ACM Symposium on Cloud Computing, 1–13. https://doi.org/10.1145/2391229.2391236

Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. International Journal of Forecasting, 36(3), 1181–1191. https://doi.org/10.1016/j.ijforecast.2019.07.001

Taylor, S. J., & Letham, B. (2018). Forecasting at scale. The American Statistician, 72(1), 37–45. https://doi.org/10.1080/00031305.2017.1380080

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30, 5998–6008.

Wen, R., Torkkola, K., Narayanaswamy, B., & Madeka, D. (2017). A multi-horizon quantile recurrent forecaster. NeurIPS Time Series Workshop.

Woo, G., Liu, C., Sahoo, D., Kumar, A., & Hoi, S. C. H. (2022). ETSformer: Exponential smoothing transformers for time-series forecasting. arXiv. https://arxiv.org/abs/2202.01381

Wu, H., Xu, J., Wang, J., & Long, M. (2021). Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. Advances in Neural Information Processing Systems, 34, 22419–22430.

Yunhe Li. (2023). Risk-Sensitive Offline Reinforcement Learning for Stable ABR QoE Improvements on Real HSDPA and LTE Traces. Journal of Advanced Computing Systems , 3(4), 1-11. https://doi.org/10.69987/JACS.2023.30401

Zerveas, G., Jayaraman, S., Patel, D., Bhamidipaty, A., & Eickhoff, C. (2021). A transformer-based framework for multivariate time series representation learning. Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2114–2124. https://doi.org/10.1145/3447548.3467401

Zhao, S., Bai, J., & Roberson, D. (2025). Multi-horizon GPU demand forecasting with workload semantics and operational risk curves: An empirical study on Alibaba clusterdata GPU trace. JTIE : Journal of Technology Informatics and Engineering, 4(3). https://doi.org/10.51903/jtie.v4i3.498

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., & Zhang, W. (2021). Informer: Beyond efficient transformer for long sequence time-series forecasting. Proceedings of the AAAI Conference on Artificial Intelligence, 35(12), 11106–11115.

Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., & Jin, R. (2022). FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. Proceedings of the 39th International Conference on Machine Learning, 27268–27286.

Zhou, T., Niu, P., Wang, X., Sun, L., & Jin, R. (2023). One fits all: Power general time series analysis by pretrained LM. Advances in Neural Information Processing Systems, 36.