Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 2 August 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 142-164
DOI: https://doi.org/10.51903/jtie.v5i2.565

Token-Burst-Aware Capacity Planning for LLM Inference Services: Request Arrival, Token Demand, and Failure Risk Modeling from BurstGPT Traces

Jiayi Nie[1], Yinchen Shi[2], Lucas Zhao[3]
Email: ogyinchen@gmail.com*[2], oritom.hezekiah-braye1@ust.edu.ng[3]
[1]*Operations Research, Columbia University, NY, USA*
[2]*Computer Science, New York University, NY, USA*
[3]*Computer Science, Georgia Institute of Technology, GA, USA*
*Corresponding Author

Abstract

Large language model (LLM) inference services convert request arrivals into coupled input-token prefill and output-token decoding workloads. Capacity planning therefore depends on token volume, temporal bursts, service mix, queueing behavior, and reliability signals rather than request counts alone. This study evaluates an integrated planning pipeline on BurstGPT v1.1, comprising 5,288,173 raw requests over 121 trace days and 5,188,507 completed requests. Strictly chronological experiments aggregate demand, forecast hourly completed tokens, detect minute-level burst pressure, estimate zero-response risk, simulate capacity policies, and replay representative test hours in Vidur. Random forest, selected on the validation interval, achieved 64.73% weighted absolute percentage error (WAPE) on the locked test interval; XGBoost achieved the lowest test WAPE (64.57%), while the last-hour baseline reached 67.42%, indicating limited forecastability under a pronounced level shift. A seasonal-residual burst detector achieved F1 = 0.653, although burst prevalence was sensitive to rolling-horizon and quantile settings. For minute-level zero-response risk, raw XGBoost achieved ROC-AUC = 0.813 and average precision = 0.116; isotonic calibration improved the Brier score (0.0230–0.0204) and 10-bin expected calibration error (0.0262–0.0122), despite low F1. Static P90/P95 capacity eliminated under-provisioned test hours at cost indices of 13.10 and 18.09. More economical dynamic baselines achieved 12.65% under-provisioned hours at a cost index of 2.37 and 13.63% at 2.43. The validation-selected random-forest policy was cheaper but less reliable (34.31% at 1.23). Vidur replay linked normalized demand to A100/H100 GPU counts, latency, batching, and memory pressure. The results support conservative interpretation of point forecasts and validation of reserve rules under distribution shift, rare-event calibration, and serving-stack constraints.

Keywords: *LLM inference serving, Capacity Planning, Token Demand Forecasting, Burst Detection, Failure Risk.*

I. INTRODUCTION

LLM inference services differ from conventional request-serving systems because a single request can create a short prefill, a long autoregressive decode, or both. The Transformer architecture made sequence modeling practical at scale (Vaswani et al., 2017), and later foundation models demonstrated that large language models can serve many tasks through a unified interface (Bommasani et al., 2021; Brown et al., 2020; Chowdhery et al., 2023; Touvron et al., 2023). The serving side of this development is a capacity-management problem: a cluster must absorb variable arrivals, preserve compute and key-value (KV) cache memory for long sequences, and protect latency service-level objectives (SLOs) during bursts (Chen et al., 2024; He et al., 2023; Tu et al., 2024). Request counts are insufficient because two hours with similar arrivals can differ substantially in prompt and generated-token volume.

Received on June, 2026; Revised on June, 2026; Accepted on July, 2026; Published on August, 2026

Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 2 August 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 142-164
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

This paper studies the problem with BurstGPT v1.1, a public trace of real LLM service traffic released in 2024 and subsequently documented by Wang et al. (2025). The records expose relative request time, model class, request-token length, response-token length, total tokens, and API or conversation service type. The two raw files include rows with zero response tokens, while corresponding filtered files remove those rows. Completed rows define observed token demand; raw rows additionally support modeling of zero-response events. The latter are treated as trace-observed noncompletion signals, not as proof of GPU exhaustion, queue overload, or any other unobserved root cause.

The research question is: how accurately can near-term token demand, burst state, and zero-response risk be modeled from production LLM traces, and how do these estimates affect capacity decisions under distribution shift? Modern serving systems use memory-aware scheduling, continuous or iteration-level batching, and KV-cache management to raise throughput (Kwon et al., 2023; Pope et al., 2023; Shazeer, 2019; Yu et al., 2022). A fleet sized on average request volume can still fail during short token bursts; one sized at a conservative historical percentile can waste accelerators after a level shift. The evaluation therefore includes static percentiles, naive and moving-average controllers, recent-window percentile policies, a queue/backlog-aware controller, split-conformal upper bounds, learned forecasts, and risk/burst reserves on the same locked test interval.

The contribution is fourfold. First, the study constructs completed-demand and raw-event views directly from BurstGPT fields and validates their arithmetic and coverage. Second, it uses chronological train, validation, calibration, threshold-tuning, and test intervals, with exact causal feature definitions and saved search grids. Third, it evaluates burst-label sensitivity and probability calibration before allowing rare-event signals to influence reserve capacity. Fourth, it grounds normalized token-hour results through Vidur replay of representative test hours on profiled A100 and H100 configurations (Agrawal et al., 2024). This final stage connects token demand to time to first token (TTFT), time between tokens (TBT), end-to-end (E2E) latency, scheduling delay, batching, memory, and restart behavior.

The operating target is deliberately multidimensional. Under-provisioning can appear as queueing, admission control, missed latency targets, or noncompletion; over-provisioning appears as idle accelerator capacity. Point error alone cannot characterize this trade-off, and rare-event discrimination alone does not establish whether a risk threshold produces a useful reserve. The analysis therefore reports forecasting errors, burst precision and recall, receiver operating characteristic area under the curve (ROC-AUC), average precision (AP), F1, Brier score, reliability curves, capacity shortage and cost, and hardware-level replay metrics. This decision-

oriented evaluation prevents a small predictive improvement from being interpreted as operational sufficiency.

II. LITERATURE REVIEW

LLM serving inherits the concerns of cloud-scale systems while adding token-level variability. Classic datacenter work emphasizes that cluster capacity must be managed as an integrated computer rather than as isolated machines (Barroso et al., 2013), and tail-latency analysis shows why a small fraction of slow or overloaded components can dominate user experience (Dean & Barroso, 2013). Distributed scheduling research similarly argues that low-latency services require fast placement under changing load (Ousterhout et al., 2015). Prediction-serving platforms such as Clipper and Clockwork established that online inference needs batching, scheduling, and model-aware resource control to sustain latency under fluctuating traffic (Crankshaw et al., 2017; Gujarati et al., 2020).

Transformer inference makes those controls sequence dependent. Multi-query attention reduces memory-bandwidth pressure during autoregressive decoding (Shazeer, 2019), while inference-scaling studies show that attention-cache and parallelism choices affect throughput (Pope et al., 2023). Orca introduced iteration-level scheduling because token-by-token generation does not match one-shot DNN execution (Yu et al., 2022). PagedAttention and vLLM made KV-cache memory management central to effective batching under variable sequence lengths (Kwon et al., 2023). Vidur models prefill, decode, batching, scheduling, parallelism, and profiled operator execution to estimate LLM-serving performance without exhaustively deploying each configuration (Agrawal et al., 2024). These systems motivate workload evaluation that preserves prompt length, output length, arrival timing, and context limits.

Capacity planning also has a forecasting tradition. ARIMA-style models provide a foundation for trend, autocorrelation, and seasonality (Box et al., 2015), and forecasting practice emphasizes temporally ordered holdouts (Hyndman & Athanasopoulos, 2021). Cloud workload studies translate forecast errors into quality-of-service and provisioning cost (Calheiros et al., 2015; He et al., 2023; He et al., 2024; Tu et al., 2024), while forecasting at scale demonstrates the operational value of calendar seasonality and changing levels (Taylor & Letham, 2018). Conformal methods add prediction sets or intervals around a point model; for dependent time series, assumptions and empirical coverage must be stated rather than inferred from an exchangeable-data result (Stankevičiūtė et al., 2021).

Tree ensembles are natural workload baselines because they represent nonlinear interactions among lags, rolling statistics, and calendar variables. Random forests aggregate decorrelated trees (Breiman, 2001), gradient boosting fits additive trees to residual structure (Friedman, 2001), XGBoost adds regularization and scalable computation (Chen & Guestrin, 2016), and LightGBM

uses histogram learning and leaf-wise growth (Ke et al., 2017). Recurrent models such as long short-term memory networks remain important for sequential prediction (Hochreiter & Schmidhuber, 1997), but the hourly sample size here favors transparent lag-feature models. Interpretability also matters in operations; feature-attribution methods connect model outputs to observable drivers (Lundberg & Lee, 2017).

Burst and event-risk modeling connect forecasting to reliability. Anomaly-detection research distinguishes point, contextual, and collective anomalies, which is relevant because a token level may be routine at one time and exceptional at another (Chandola et al., 2009; Liu et al., 2023; Nie & Zheng, 2023, 2024). Rare positives require precision-recall analysis because accuracy and even ROC curves can conceal weak positive identification (Davis & Goadrich, 2006). Probability calibration is also important when reserve size depends on estimated risk rather than rank alone (Guo et al., 2017). Operational summaries and evidence chains can then translate quantitative signals into repeatable actions while retaining source context (Li et al., 2023; Li et al., 2024; Liu et al., 2024; Zhang et al., 2024).

Taken together, prior work suggests that a serving study should align each metric with its decision: point forecasting with allocation, contextual bursts with reserve activation, calibrated event probabilities with risk thresholds, and hardware replay with latency and memory constraints. This paper follows that alignment and treats discrepancies across layers as results rather than forcing them into a single favorable score.

III. RESEARCH METHODS

A. Data Source, Views, and Validation

BurstGPT v1.1 contains two chronological raw CSV files and two corresponding files with zero-response rows removed. The analysis reads the two raw files and constructs both views with one consistent rule: a completed row has Response tokens > 0 and Total tokens > 0; all other rows are excluded from served-demand targets and retained as noncompletion observations. There were 99,666 zero-response rows (1.8847% of 5,288,173); 99,648 also had zero total tokens, and 18 retained a nonzero total-token field. The completed rows contained no violations of Request tokens + Response tokens = Total tokens.

Table 1. BurstGPT fields and their roles in the analysis.

Field	Type	Role
Timestamp	Relative seconds	Minute, hour, and trace-day aggregation
Model	Categorical	ChatGPT/GPT-4 workload segmentation
Request tokens	Integer	Prefill-side demand indicator
Response tokens	Integer	Decode-side demand; zero defines a trace-observed noncompletion signal
Total tokens	Integer	Completed token-demand target and arithmetic check
Log Type	Categorical	API/conversation service segmentation

The exact study roles of the source fields are given in Table 1. Table 2 reports exact file coverage and checksums. Because the release has no request identifier, 204,190 rows duplicate all six visible fields exactly. These rows were retained: identical timestamp, model, token, and service fields can represent distinct simultaneous requests, and deduplication cannot be justified from the available schema. The timestamp is an anonymized relative clock; no civil date or timezone is inferred.

Table 2. Raw trace coverage used by the experiments.

File	Raw rows	Completed rows	Excluded rows	Timestamp range (s)	Completed tokens	SHA-256
BurstGPT_1.csv	1,429,737	1,404,294	25,443	5–5,269,973	1,049,252,253	4bb3783693d0a435…e12122
BurstGPT_2.csv	3,858,436	3,784,213	74,223	5,270,414–10,454,395	1,126,271,015	44bf5942b03fca42…8928bf7
Total	5,288,173	5,188,507	99,666	5–10,454,395	2,175,523,268	—

Completed records were summed into 174,240 minute indices and 2,904 hour indices, with absent intervals filled by zero. Minute aggregates contain request count, completed-token load, model/service shares, and zero-response count. Hour aggregates contain request, response, and total completed tokens together with request count and model/service shares. Figure 1 summarizes how these views feed forecasting, burst/event modeling, capacity simulation, and GPU replay.

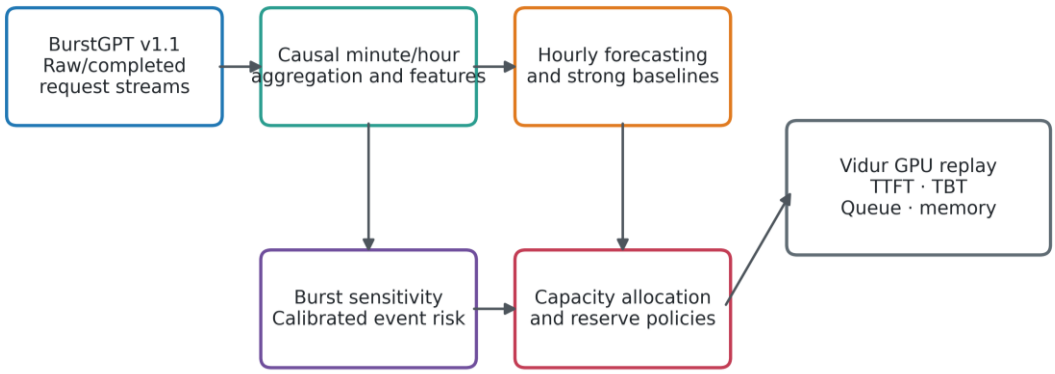

Figure 1. Experimental workflow from BurstGPT aggregation to capacity policies and hardware replay.

B. Chronological Splits and Causal Features

Every feature used at interval t ends at t - 1 or is a calendar variable known before t. Exact split indices are reported in Table 3. Forecasting reserves the first 168 hours for lag construction, then uses 1,915 training hours, a 410-hour validation block, and 411 test hours. The first validation half selects hyperparameters and the point-model family; the second half calibrates conformal residual bounds. Capacity-controller settings are selected on the full validation block after the point model is fixed. Failure-risk validation is divided into a calibration half and a threshold-selection half. The test intervals are consulted only after choices are frozen.

Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 2 August 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 142-164
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

Table 3. Exact chronological partitions on the relative trace clock.

Task	Training	Validation/calibration	Threshold or controller tuning	Locked test
Hourly forecast	h 168–2,082 (1,915)	h 2,083–2,287 (205) for model selection; h 2,288–2,492 (205) for conformal calibration	Full h 2,083–2,492 (410), after point-model selection	h 2,493–2,903 (411)
Failure risk	min 1,441–122,399 (120,959)	min 122,400–135,358 (12,959)	min 135,359–148,318 (12,960)	min 148,319–174,239 (25,921)
Burst detection	min 0–121,966 (121,967)	min 121,967–148,102 (26,136)	Included in validation interval	min 148,103–174,239 (26,137)

The hourly forecast has exactly 33 predictors: four calendar variables (hour_sin, hour_cos, week_sin, week_cos); total-token lags at 1, 2, 3, 6, 12, 24, 48, 72, and 168 hours; shifted rolling mean, population standard deviation, and maximum over 3, 6, 12, 24, 72, and 168 hours; and two log-growth contrasts, log1p(lag_1) − log1p(lag_24) and log1p(lag_1) − log1p(rolling_mean_24). The 89 failure-risk predictors are generated exactly from five source series—request count, completed tokens, GPT-4 share, API share, and zero-response count. Each source contributes lags at 1, 2, 5, 15, 60, 240, and 1,440 minutes plus rolling sum and mean over 5, 15, 60, 240, and 1,440 prior minutes (17 variables per source, 85 total); minute-of-day and day-of-week sine/cosine terms add four calendar variables. Seventeen predictors therefore use prior zero-response counts, while 72 do not. Table 4 records model search, calibration, and software details. Forecast regressors fit log1p(hourly tokens) and invert with expm1. The candidate grid contained 22 configurations: Ridge (4), ElasticNet (6), random forest (3), gradient boosting (3), XGBoost (3), and LightGBM (3). The selected configurations were Ridge $\alpha = 100$; ElasticNet $\alpha = 0.1$ and l1 ratio = 0.5; random forest with 500 trees, unlimited depth, minimum leaf size 4, and all features; Huber gradient boosting with 300 trees, depth 3, learning rate 0.03, minimum leaf size 4, and subsample 0.9; XGBoost with 400 trees, depth 4, learning rate 0.03, minimum child weight 2, subsample/column sample 0.9, and L2 = 1; and LightGBM with 400 trees, 31 leaves, learning rate 0.03, minimum child size 20, subsample/column sample 0.9, and L2 = 1.

Table 4. Reproducibility and selection specification.

Component	Exact specification	Selection criterion
Forecast point models	33 strictly causal features; six learned families plus lag-1, seasonal-24, seasonal-168, MA-6/24/168, and recent-window P90/P95-24/168 baselines	Lowest WAPE on validation hours 2,083–2,287
Failure models	Logistic: standardized, C = 0.1, balanced; RF: 300 trees, depth 14, leaf 3, sqrt features; XGBoost: six-candidate grid	XGBoost grid selected by validation AP; thresholds by validation F1
Selected failure XGBoost	350 trees, depth 6, learning rate 0.05, child weight 5, subsample 0.85, column sample 0.80, L2 = 5, L1 = 0.1	Validation AP = 0.3595
Probability calibration	Raw, Platt scaling, and isotonic regression fit on min 122,400–135,358	Lowest Brier score on min 135,359–148,318
Randomness	Forecast seed 20,250,716; failure/burst seed 42	Fixed before evaluation
Software	Python 3.12.13; pandas 3.0.3; NumPy 2.5.1; scikit-learn 1.8.0; XGBoost 3.0.2; LightGBM 4.6.0	Single-threaded learned forecasts where supported

C. Forecasting Metrics and Interpretation

The primary forecast metric is WAPE, defined as total absolute error divided by total observed demand. Mean absolute error (MAE) and root mean squared error (RMSE) retain token-hour units; RMSE weights large misses more heavily. Mean absolute percentage error (MAPE) is computed only for nonzero hours, and symmetric MAPE (sMAPE) provides a bounded proportional summary. WAPE selects the model on the first validation half because low-demand and zero hours make per-hour percentage metrics unstable. Test WAPE is an evaluation result, not a model-selection rule.

D. Burst Labels and Detector Sensitivity

The primary planning-burst label is training-defined and uses a five-minute response horizon. A minute is positive if its five-minute completed-token total exceeds the training 97.5th percentile (430,938.6 tokens), or if its single-minute load exceeds the corresponding 97.5th percentile (89,573 tokens). In comparison, positive five-minute growth exceeds its 97.5th percentile (151,711.2 tokens). The five-minute window approximates a short reserve or routing reaction interval; the 97.5th percentile excludes routine busy periods while retaining rare high-pressure minutes in the test interval. This remains an operational proxy rather than a measured GPU-overload label.

The detectors are a static P95 rule, an exponentially weighted moving average (EWMA) z-score, a rolling median absolute deviation (MAD) z-score, a seasonal residual quantile, and an isolation forest. Their thresholds are selected on validation data. Robustness is tested with rolling-only labels at 95th, 97.5th, and 99th training percentiles over 1, 5, and 15 minutes. This sensitivity analysis is essential because a lower-demand test regime can eliminate positives defined by long or extreme training windows.

E. Zero-Response Risk and Calibration

The minute-level target equals one when at least one raw request has zero response tokens. Logistic regression, random forest, XGBoost, and LightGBM are compared, with the three originally specified models retained in the main results table and LightGBM used as a supplementary discrimination check. XGBoost probabilities are calibrated using Platt scaling and isotonic regression. The first validation half fits each calibrator; the second selects the calibration method by Brier score and its decision threshold by F1. Test results include ROC-AUC, AP, Brier score, 10-bin expected calibration error (ECE), precision, recall, and F1. Threshold sensitivity is evaluated from 0.01 to 0.50.

An additional episode-onset analysis labels only positive minutes preceded by 60 failure-free minutes. It compares the full model with an ablation that removes all prior-zero-response features. This test distinguishes within-episode persistence from warning. Segment-median completed

Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 2 August 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 142-164
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

tokens are imputed for excluded rows only to produce an unserved-token proxy in operational summaries; this quantity is not interpreted as observed lost work or as proof of a capacity failure.

F. Capacity policies

Capacity is first expressed in normalized completed token-hours. Static policies use P90 or P95 of all causal pre-validation hours 0–2,082. Dynamic baselines include lag-1, seasonal naive at 24 and 168 hours, moving averages at 6, 24, and 168 hours, recent-window P90/P95 over 24 and 168 hours, and a queue/backlog-aware controller. One-sided 90% and 95% split-conformal upper bounds add finite-sample quantiles of calibration residuals to the selected point forecast (Stankevičiūtė et al., 2021). The conformal results are reported as empirical holdout performance; nominal coverage is not asserted under the observed level shift.

All dynamic policies have a 10,900-token/hour warm floor, the training P10. Multiplicative factors and queue parameters are chosen on the full validation interval by minimizing mean provisioned capacity among candidates with at most 10% under-provisioned hours; if none is feasible, the rule minimizes under-provisioning first and capacity second. The queue policy uses

$$C[t] = max(floor, factor \times MA24[t] + gain \times B[t-1])$$

and updates virtual backlog after demand is observed:

$$B[t] = max(0, B[t-1] + demand[t] - C[t]).$$

The selected factor/gain pair is 1.75/0.50. The selected random-forest factor is 2.0. Risk and burst reserves use only the previous completed hour. A 10% reserve is activated by any previous-hour burst detection or by previous-hour maximum raw XGBoost risk above 0.9591; that threshold and reserve size are fixed on validation. Outcomes are under-provisioned hours, mean and P95 shortage across all hours, mean unused capacity, served-demand percentage, and cost index (mean capacity divided by mean test demand).

G. GPU-serving replay

Normalized demand is mapped to serving constraints with Vidur commit 8383d2935bc62723a212090baa9f98ada206fc14, whose supplied profiles cover A100-80GB and H100-80GB DGX systems. Three one-hour windows are selected from the 411-hour test region: median completed demand (trace hour 2,757), the closest hour to test P95 demand (2,801), and peak demand (2,862). Replays use Llama-2-70B, FP16, vLLM scheduling, tensor parallelism 2, and pipeline parallelism 1. The study-level latency criterion is TTFT-P90 ≤ 2 s and TBT-P99 ≤ 0.2 s; the reliability-aware criterion additionally requires zero simulator restarts. Vidur supports a 4,096-token context for this configuration. Of 216,444 completed requests in the test region, 977 (0.451%) exceed that limit and account for 7.171% of completed tokens. They are excluded rather than truncated. The median window is fully replayable; the P95 window retains 4,064 of 4,066 requests and 809,715 of 818,673 tokens; the peak window retains 668 of 860 requests and

1,132,969 of 2,142,605 tokens. The peak results therefore characterize the supported-context subset, not the full peak-hour demand. TTFT measures arrival-to-first-token delay, TBT is the inter-token decode delay, E2E is completion latency, and model flops utilization (MFU) is the ratio of achieved to theoretical floating-point work.

III. RESULTS/FINDINGS

A. Workload Composition and Temporal Structure

The trace is uneven across model and service types. As Table 5 shows, ChatGPT API traffic supplies 80.01% of completed tokens, while GPT-4 API and conversation traffic have higher zero-response percentages. These differences justify service-mix predictors but do not establish why a row has zero output.

Table 5. Workload composition by model and service type.

Model	Service type	Raw requests	Zero-response rows	Event %	Completed-token share %	Completed median
ChatGPT	API log	4,810,532	77,238	1.61	80.01	227
ChatGPT	Conversation log	159,736	6,999	4.38	6.82	701
GPT-4	API log	242,362	12,301	5.08	9.03	646
GPT-4	Conversation log	75,543	3,128	4.14	4.13	883

Table 6 shows that conversation records are longer than API records within each model class and that GPT-4 has a larger output/input ratio. Thus, identical arrival counts can imply different prefill, decode, and KV-cache pressure.

Table 6. Completed request-token distribution by model and service type.

Model/service	Requests	Req med.	Req P95	Resp med.	Resp P95	Total med.	Total P95	Output/ input
ChatGPT API	4,733,294	216	795	7	165	227	928	0.128
ChatGPT conversation	152,737	494	1,951	196	623	701	2,423	0.330
GPT-4 API	230,061	481	1,586	79	1,179	646	2,170	0.387
GPT-4 conversation	72,415	598	2,799	262	823	883	3,406	0.338

At the model level, Table 7 confirms a completed-token median of 691 for GPT-4 versus 228 for ChatGPT and a higher GPT-4 zero-response percentage. Figure 2 shows nonstationary daily demand and a substantial low-demand interval before the final rebound. Figure 3 shows recurring hour-of-day and day-of-week structure, while Figure 4 shows the long-tailed completed-token distribution.

Table 7. Model-level completed demand and zero-response comparison.

Model	Completed requests	Event %	Req med.	Req P95	Resp med.	Resp P95	Total med.	Total P95	Output/input
ChatGPT	4,886,031	1.69	216	892	7	193	228	1,161	0.141
GPT-4	302,476	4.85	498	2,016	119	1,126	691	2,396	0.371

B. Forecasting under distribution shift

Random forest was selected by validation WAPE (35.87%). On the locked test interval, it obtained WAPE = 64.73%, while XGBoost had the numerical minimum of 64.57% (Table 8).

Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 2 August 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 142-164
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

The 0.15-percentage-point test difference does not replace the validation-selected model. The last-hour baseline remained close at 67.42%, and LightGBM and gradient boosting were also within two percentage points of the selected model. Consequently, the learned-model advantage is small relative to the absolute error.

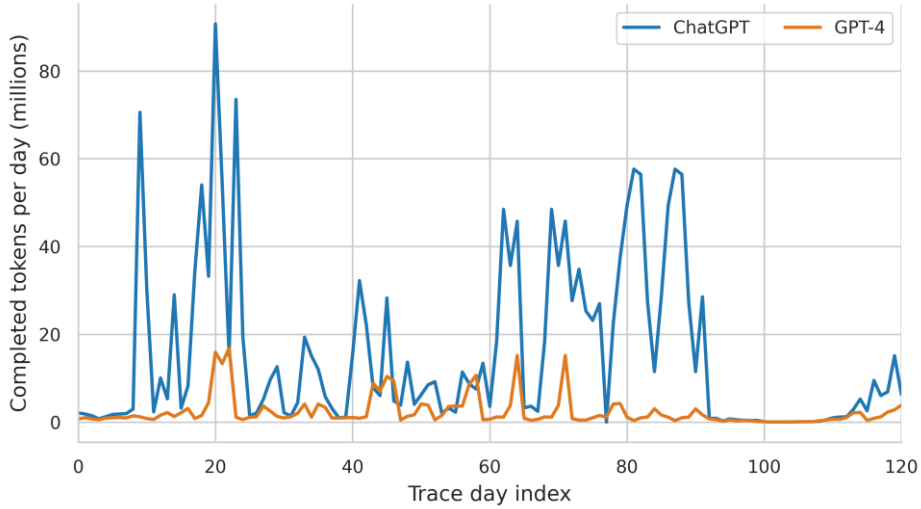

Figure 2. Daily completed-token demand by model over the 121-day trace.

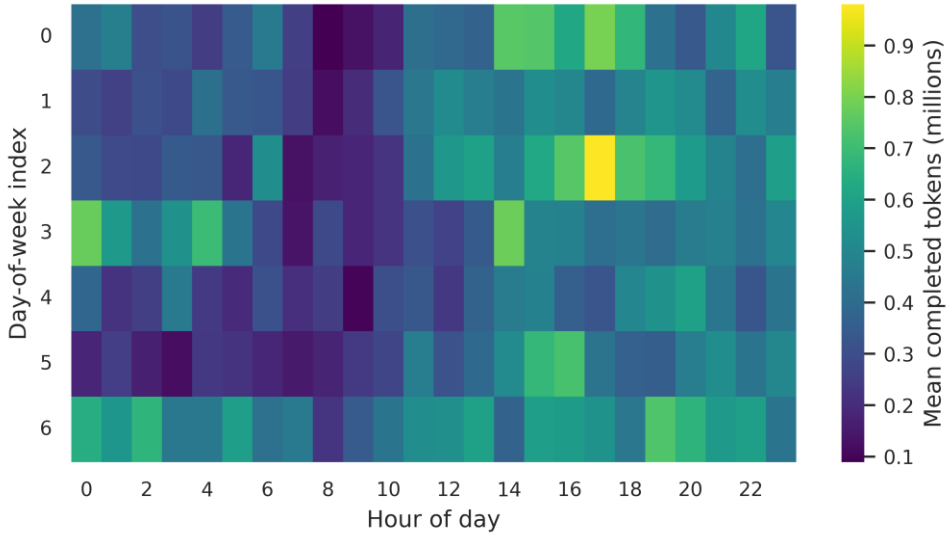

Figure 3. Mean completed-token demand by day-of-week index and hour of day.

Figure 5 explains the high errors. The test interval begins at a low level and then rebounds with sharp spikes. Local persistence helps the lag-1 baseline, but it cannot anticipate rebounds; week-ahead replay inherits a different demand level; and learned models underpredict many peaks. The results support using point forecasts as one controller input, not describing them as accurate forecasts of every high-demand hour.

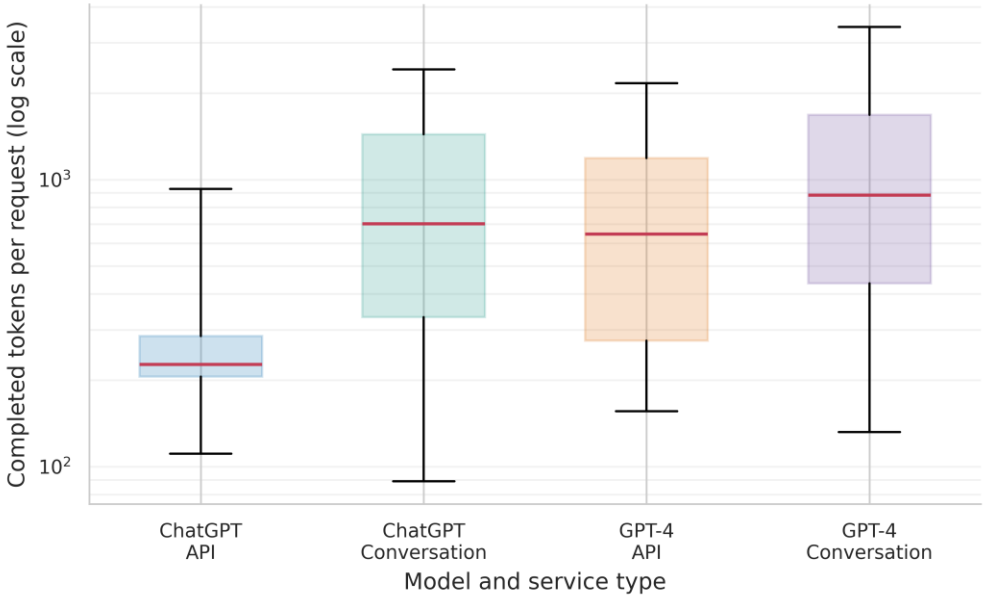

Figure 4. Completed total-token distribution by model and service type; whiskers show the 5th and 95th percentiles.

Table 8. Hourly completed-token forecasting on the locked test interval.

Model	MAE	RMSE	MAPE nonzero %	sMAPE %	WAPE %	Mean bias
XGBoost log-target	122,741	245,214	161.88	118.91	64.57	−69,339
Random forest log-target	123,033	258,081	129.31	100.90	64.73	−74,073
LightGBM log-target	125,172	246,436	231.11	106.04	65.85	−44,433
Gradient boosting log-target	126,466	245,131	127.70	119.21	66.53	−66,149
Last-hour naive	128,158	254,505	205.42	81.24	67.42	−721
Six-hour moving average	135,491	256,286	486.81	96.68	71.28	−5,568
Seasonal naive 24 h	165,291	345,002	291.96	97.47	86.96	−24,639
Seasonal naive 168 h	170,528	329,256	597.40	140.88	89.72	−152,972
ElasticNet log-target	217,917	320,216	2,124.46	121.29	114.65	67,152
Ridge log-target	222,752	328,957	2,136.31	121.53	117.19	76,544

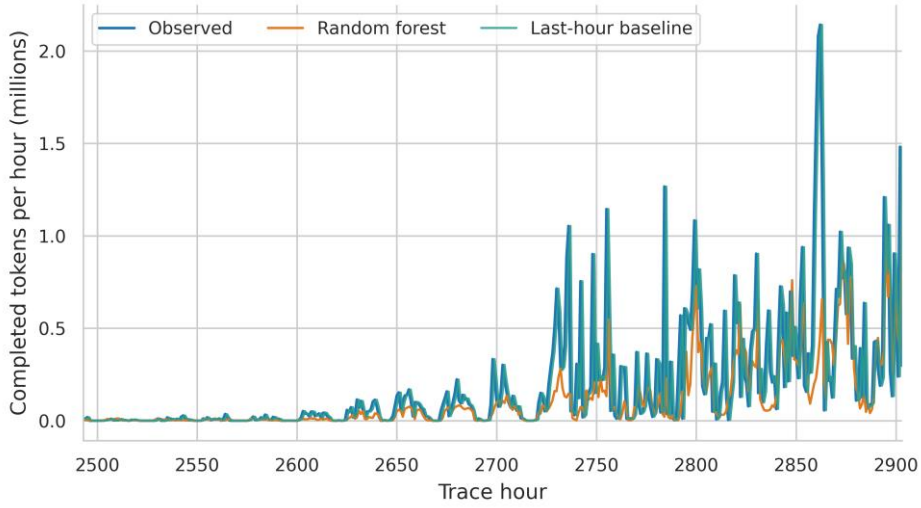

Figure 5. Locked-test hourly demand with the validation-selected random forest and last-hour baseline.

Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 2 August 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 142-164
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

C. Burst detection and label sensitivity

The primary compound burst label is positive in 0.287% of test minutes. The seasonal-residual detector obtains precision 0.534, recall 0.840, and F1 0.653, the best primary-label F1 in Table 9. Static P95 and isolation forest recover most positives but flag more minutes. EWMA and rolling MAD are substantially less precise.

Table 9. Primary burst detectors and rolling-label sensitivity on the test interval.

Analysis	Setting	Test positive %	Precision	Recall	F1	Flagged %
Primary detector	Seasonal residual quantile	0.287	0.534	0.840	0.653	0.451
Primary detector	Static P95 token rule	0.287	0.385	0.867	0.533	0.647
Primary detector	Isolation forest	0.287	0.312	0.987	0.474	0.907
Primary detector	EWMA z-score	0.287	0.034	0.853	0.066	7.181
Primary detector	Rolling MAD z-score	0.287	0.018	0.947	0.035	15.338
Sensitivity	1 min, q95	0.647	0.954	0.988	0.971	0.670
Sensitivity	1 min, q97.5	0.363	0.949	0.989	0.969	0.379
Sensitivity	1 min, q99	0.080	0.457	1.000	0.627	0.176
Sensitivity	5 min, q95	0.302	0.949	0.949	0.949	0.302
Sensitivity	5 min, q97.5	0.088	0.920	1.000	0.958	0.096
Sensitivity	5 min, q99	0.000	0.000	0.000	0.000	1.710
Sensitivity	15 min, q95	0.000	0.000	0.000	0.000	0.000
Sensitivity	15 min, q97.5	0.000	0.000	0.000	0.000	0.000
Sensitivity	15 min, q99	0.000	0.000	0.000	0.000	1.106

The lower panel of Table 9 shows that this result is conditional on label design. Rolling-only one-minute labels retain positives across all three quantiles; the five-minute 99th-percentile label and every 15-minute label have no test positives because the test demand level is below those training thresholds. Therefore, the primary label is operationally interpretable for a short response horizon, but its prevalence and apparent detector performance are not stable under all reasonable horizons and quantiles. Figure 6 displays the compound labels, including their sparse occurrence and association with short-lived token spikes.

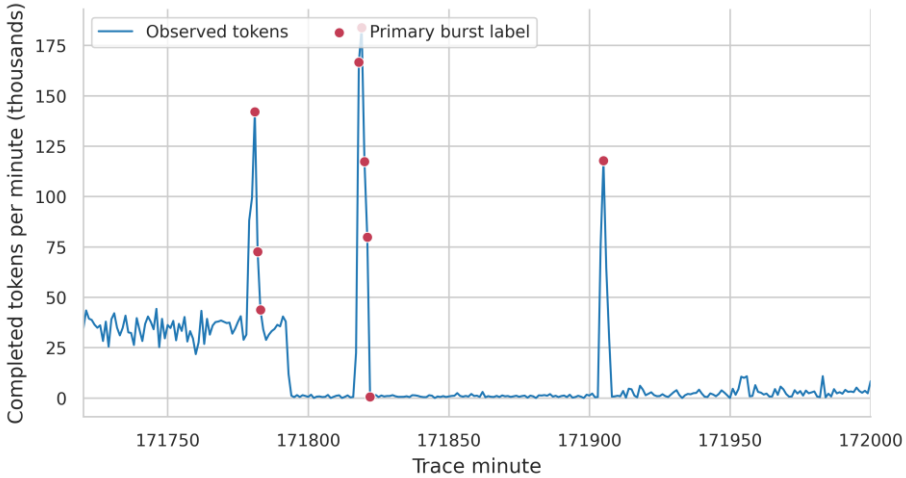

Figure 6. Minute-level completed-token demand with primary planning-burst labels.

D. Zero-response risk, calibration, and threshold sensitivity

Test prevalence is 2.141% of minutes. Raw XGBoost has the strongest discrimination among the three main models, with ROC-AUC 0.813 and AP 0.116, but its validation-selected threshold yields F1 0.172 (Table 10). This is adequate for ranking some risky intervals, not for claiming reliable classification of most positive minutes. Random forest has lower ROC-AUC and AP, while logistic regression performs poorly at its threshold. Figure 7 shows the XGBoost ROC curve.

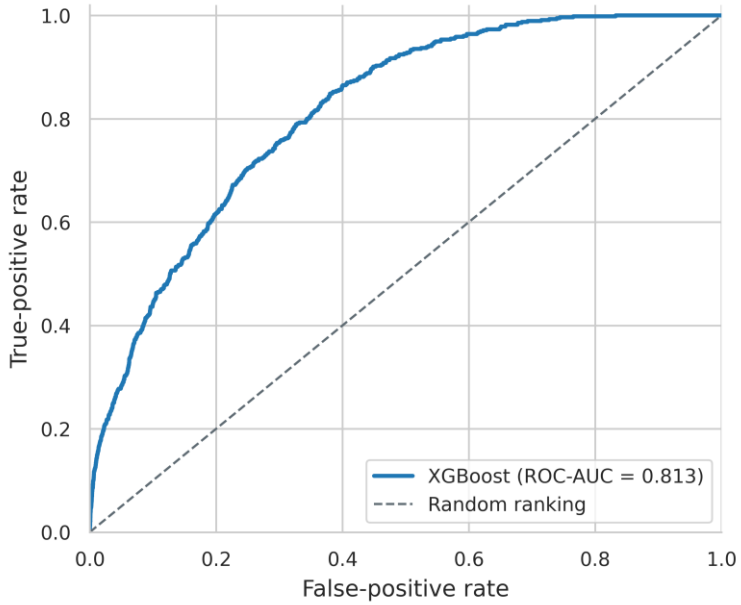

Figure 7. ROC curve for raw XGBoost zero-response risk on the locked test interval.

Calibration improves probability error but does not solve the rare-event task. Isotonic regression is selected by validation Brier score. On test, it reduces Brier from 0.0230 to 0.0204 and ECE from 0.0262 to 0.0122, while AP falls to 0.099 and F1 remains 0.169 because isotonic mapping creates probability ties. Platt scaling happens to have the lowest test Brier and ECE, but it was not the validation-selected method. Figure 8 shows that all variants remain imperfect in higher-risk bins.

Table 10. Test results for zero-response risk and XGBoost calibration.

Model/calibration	ROC-AUC	AP	Brier	ECE	Threshold	Precision	Recall	F1
Logistic regression, raw	0.639	0.050	0.1883	0.3805	0.7534	0.126	0.052	0.074
Random forest, raw	0.774	0.099	0.0375	0.0999	0.2875	0.092	0.391	0.149
XGBoost, raw	0.813	0.116	0.0230	0.0262	0.2404	0.138	0.227	0.172
XGBoost, Platt	0.813	0.116	0.0201	0.0091	0.0844	0.138	0.227	0.172
XGBoost, isotonic	0.810	0.099	0.0204	0.0122	0.1538	0.132	0.236	0.169

At isotonic thresholds 0.01, 0.05, 0.1538, and 0.50, precision/recall are 0.067/0.573, 0.100/0.382, 0.132/0.236, and 0.438/0.013, respectively. Reserve behavior is therefore threshold-sensitive. The episode-onset test is still weaker: the full model has ROC-AUC 0.613 and AP 0.0032 for 69 onset

Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 2 August 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 142-164
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

minutes, while the load-only ablation has ROC-AUC 0.679 and AP 0.0045. Most discrimination in the full task comes from persistence within clusters after a zero-response event has already appeared, not from dependable advance warning of a new episode.

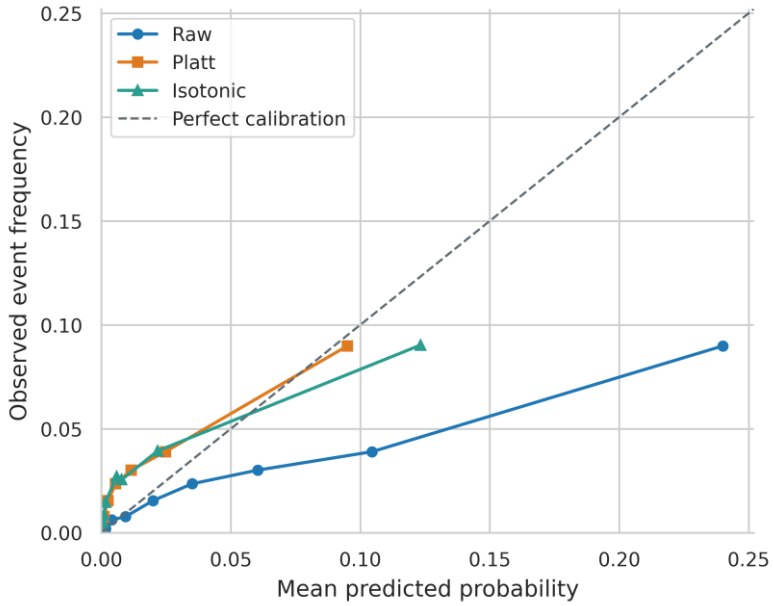

Figure 8. Reliability curves for raw, Platt-scaled, and isotonic XGBoost probabilities.

E. Capacity policies and stronger baselines

Table 11 shows a broader frontier than static-versus-learned comparison alone. Static pre-validation P90/P95 capacity eliminates shortages but costs 13.10 and 18.09 times mean test demand. The recent-window P95-24 policy has the lowest under-provisioning among dynamic policies (12.65%) at cost 2.37. Tuned lag-1 has the same under-provisioning at higher cost 3.00, while tuned MA-6 has 13.63% under-provisioning at cost 2.43. These common controllers outperform the validation-selected random-forest capacity policy on reliability.

Table 11. Capacity simulation on the 411-hour test interval.

Policy	Under %	Mean shortage	P95 shortage	Mean unused	Cost index
Static pre-validation P90	0.00	0	0	2,300,160	13.101
Static pre-validation P95	0.00	0	0	3,248,505	18.090
Recent-window P95, 24 h	12.65	19,958	72,554	281,031	2.374
Lag-1, tuned factor	12.65	20,626	142,760	400,611	2.999
Moving average 6 h, tuned	13.63	17,081	86,355	289,319	2.432
Recent-window P90, 24 h	15.57	28,368	154,059	208,513	1.948
Queue/backlog-aware MA24	19.71	26,835	168,092	171,562	1.761
Moving average 24 h, tuned	19.95	31,281	194,462	189,985	1.835
Random forest, tuned factor	34.31	53,720	300,334	98,160	1.234
Random forest + burst reserve	34.31	52,882	295,893	106,689	1.283
Random forest + failure-risk reserve	34.31	53,720	300,334	98,160	1.234
Random forest + failure/burst reserve	34.31	52,882	295,893	106,689	1.283
Split-conformal upper 95%	37.23	86,277	466,518	39,182	0.752
Seasonal naive 168 h, raw	63.99	158,507	708,807	10,511	0.221

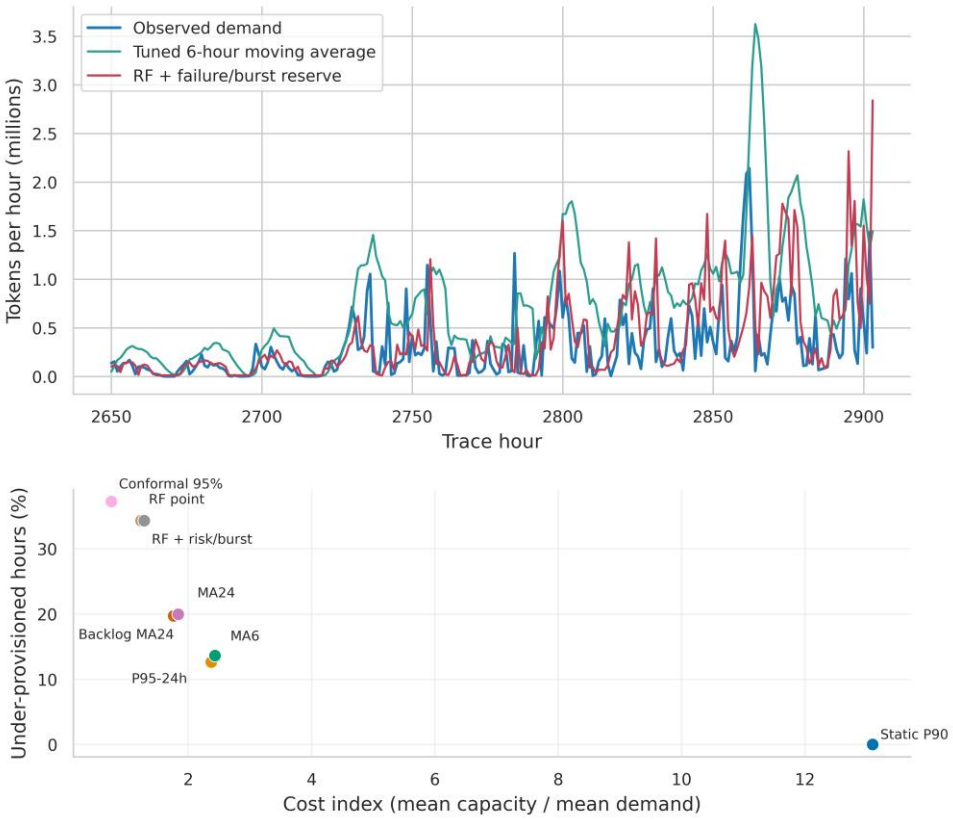

Figure 9. Capacity trajectories and reliability-cost positions for representative policies.

The random-forest policy has cost 1.234 but under-provisions 34.31% of hours. Adding the 10% burst reserve does not change the number of under-provisioned hours; it lowers mean shortage from 53,720 to 52,882 tokens and P95 shortage from 300,334 to 295,893 while raising cost to 1.283. The failure-risk trigger does not activate in the test interval because the validation-selected 0.9591 threshold is never exceeded by the prior-hour test signal, so the risk-only and combined rows match their corresponding base/burst rows. This is a negative operational result: the calibrated event model cannot be assumed to improve capacity without trigger coverage in the deployment regime.

Table 12. Reliability-aware Vidur replay at the minimum evaluated GPU count.

Device/window	Trace hour	Replay requests	Replay tokens	GPUs	TTFT P90 (s)	TBT P99 (s)	E2E P95 (s)	Sched. P95 (s)	Restarts
A100 median	2,757	50	52,902	2	0.561	0.059	55.033	0.034	0
A100 P95	2,801	4,064	809,715	4	0.747	0.152	2.633	0.049	0
A100 peak subset	2,862	668	1,132,969	4	0.744	0.061	207.957	0.056	0
H100 median	2,757	50	52,902	2	0.209	0.032	29.215	0.004	0
H100 P95	2,801	4,064	809,715	2	0.465	0.106	1.388	0.030	0
H100 peak subset	2,862	668	1,132,969	2	0.483	0.038	125.739	0.036	0

The split-conformal 95% upper policy under-provisions 37.23% of test hours despite its nominal level. Its additive calibration residual is learned from an unusually quiet interval before demand rebounds; empirical test coverage fails under this shift. The queue/backlog controller and moving

Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 2 August 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 142-164
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

averages occupy a lower-cost middle region. Figure 9 displays both hourly trajectories and the reliability-cost frontier.

F. Operational mapping to GPU serving

The Vidur results in Table 12 convert representative token windows into concrete serving outcomes. Because tensor parallelism is two, the minimum evaluated configuration has two GPUs. Two H100s satisfy the latency and zero-restart criteria for all three replayable windows. The A100 median window also requires two GPUs. The A100 P95 window needs four GPUs because two GPUs produce TBT-P99 = 0.244 s, above the 0.2-s criterion. The two-GPU A100 peak replay meets the two latency thresholds but records 47 restarts affecting 31 requests; four GPUs remove those restarts. E2E latency remains high for long decoded outputs and is reported rather than used as a fixed SLO.

Table 13. Batching and resource metrics for the replay configurations in Table 12.

Device/window	Batch mean	Batch P95	Memory mean %	Busy mean %	MFU mean %
A100 median	1.067	2	2.236	26.682	0.310
A100 P95	1.351	2	1.125	60.557	2.246
A100 peak subset	7.265	13	25.123	100.000	3.074
H100 median	1.015	1	2.127	14.837	0.097
H100 P95	1.432	2	1.199	59.997	1.402
H100 peak subset	8.525	17	29.498	99.960	1.961

Table 13 adds batching and resource pressure. P95 traffic increases device busy time substantially, and both peak configurations approach 100% busy time. Memory rises with concurrency and sequence state, especially in the peak replay, while batching differs across device and replica count. These results show why a token-hour policy cannot be converted by a single universal tokens-per-second constant: prefill/decode composition, sequence lengths, KV-cache occupancy, continuous batching, parallelism, and latency targets jointly determine the feasible GPU count (Kwon et al., 2023; Yu et al., 2022; Agrawal et al., 2024).

Table 14. Leading forecast and zero-response-risk feature importances.

Forecast feature	RF importance	Risk feature	XGBoost importance
Tokens, lag 1 h	0.592	Zero-response rolling mean, 5 min	0.170
Hour sine	0.052	Zero-response rolling sum, 5 min	0.135
Tokens, lag 72 h	0.033	Zero-response rolling sum, 15 min	0.119
Tokens, lag 168 h	0.032	Zero-response lag 1 min	0.061
Tokens, lag 24 h	0.027	Zero-response rolling mean, 15 min	0.057
Hour cosine	0.021	Zero-response rolling mean, 60 min	0.039
Tokens, lag 48 h	0.020	Zero-response rolling sum, 60 min	0.030
Tokens, lag 2 h	0.020	Zero-response lag 2 min	0.011

G. Feature interpretation and operational summaries

Random-forest impurity importance is dominated by the immediately preceding hour (0.592), followed by hour-of-day and longer lags. The failure model instead emphasizes recent zero-response history (Table 14). Figure 10 presents the forecast ranking. The separation supports two

models: one estimates demand magnitude, while the other mostly describes persistence within unhealthy intervals. Neither ranking proves causality, and correlated rolling variables divide importance among related features.

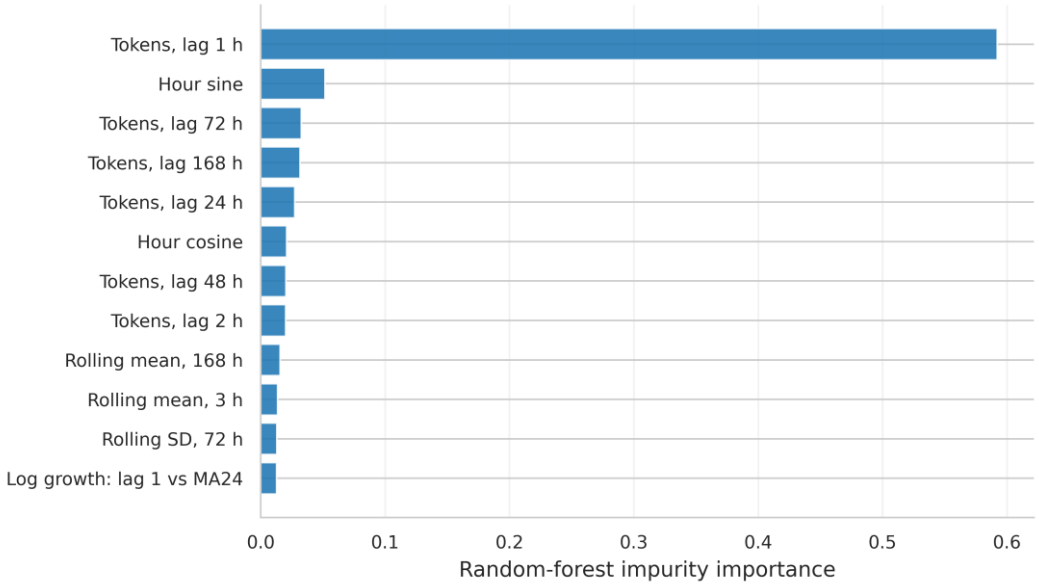

Figure 10. Impurity importance for the validation-selected random-forest forecast.

Table 15 translates the five highest pressure ratios under the combined reserve into concise actions. Failed-request counts are observed; imputed unserved tokens are segment-median proxies. The repeated rebound recommendation follows from the same failure mode visible in Figure 5: demand rises from a low base faster than the point controller. These summaries organize evidence for an operator but do not diagnose root cause.

Table 15. Highest-pressure test hours under the combined reserve policy.

Hour	Day	HOD	Actual tokens	Capacity	Pressure	Zero-response rows	Imputed tokens	Operational action
2,784	116	0	1,266,823	42,383	29.89	4	1,564	Raise the rebound-hour warm reserve and route long-context traffic to the higher-throughput pool
2,740	114	4	307,225	10,900	28.19	1	646	Raise the rebound-hour warm reserve and route long-context traffic to the higher-throughput pool
2,791	116	7	286,949	10,900	26.33	2	1,292	Raise the rebound-hour warm reserve and route long-context traffic to the higher-throughput pool
2,762	115	2	294,556	14,641	20.12	12	2,724	Raise the next-hour reserve and isolate the queue showing recent event concentration
2,787	116	3	318,544	23,083	13.80	0	0	Raise the rebound-hour warm reserve and route long-context traffic to the higher-throughput pool

Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 2 August 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 142-164
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

Discussion and Limitations

The forecasting results require cautious interpretation. Random forest is selected correctly on validation, yet its test WAPE is 64.73%, and the last-hour baseline is only 2.70 percentage points worse. XGBoost's 64.57% is a post-selection test result, not evidence that it should replace the selected model. The negative biases and Figure 5 show that high-demand rebounds are the main operational problem. Point forecasts can reduce average capacity, but recent-window percentile and moving-average controllers offer stronger reliability in this test regime.

The burst results are similarly conditional. The primary five-minute compound rule is linked to a plausible short reaction horizon, and seasonal residuals detect it better than the other tested methods. However, the sensitivity grid demonstrates that training-quantile labels can vanish after a level shift. Burst F1 therefore measures agreement with a planning proxy, not detection of independently observed GPU overload. A deployment should define the label from an explicit queue, latency, or admission-control objective and re-estimate thresholds when workload level changes.

Zero-response risk is useful mainly as a state signal. Calibration lowers Brier score and ECE, but AP and F1 remain low, and performance on episode onsets is near the rare-event base rate. Recent zero-response history dominates importance, so the model is better at identifying persistence after an event has appeared than at predicting a new incident from load alone. Moreover, BurstGPT v1.1 contains no queue length, latency, GPU utilization, error code, or routing state. A zero response cannot be attributed to insufficient capacity from this trace. The inactive risk trigger in the test interval illustrates why a ranked risk score does not automatically produce a useful capacity reserve.

Normalized capacity remains useful for comparing controller logic, but it is not a GPU count. The Vidur replays supply an operational bridge by modeling prefill, decode, batching, memory, parallelism, and latency. Their scope is nevertheless limited: three representative hours are replayed with fixed replica counts; the peak case excludes requests beyond the supported 4,096-token context; and simulator profiles are not direct measurements of the original BurstGPT serving stack. The replay establishes configuration-dependent feasibility, not a causal explanation of historical zero-response rows or a full dynamic autoscaling experiment.

External validity is also bounded by the trace. It covers two model labels and two service types over 121 anonymized relative days. Exact civil dates, region, hardware, scheduler, tenant identifiers, and request IDs are unavailable. Visible-field duplicates are retained because they cannot be resolved. Results may differ for modern long-context models, disaggregated prefill/decode fleets, speculative decoding, or workloads with different model mixes. These

limitations favor transparent thresholds, empirical coverage checks, and periodic backtesting over fixed claims of portability.

IV. CONCLUSION

Token-aware planning changes the interpretation of LLM serving demand because prompt length, output length, model mix, and temporal clustering produce pressure that request counts cannot represent. BurstGPT supports an integrated evaluation of completed-token demand and zero-response signals, but its final test segment also demonstrates the limits of prediction under level shift. The validation-selected random forest only modestly improves on lag-1 in test WAPE, and strong recent-window or moving-average capacity baselines outperform the learned policy on reliability.

The practical recommendation is a layered controller: maintain a measured warm floor, compare learned forecasts with recent-window percentile and queue-aware baselines, and activate burst or risk reserves only when validation shows both trigger coverage and an improved reliability-cost frontier. Rare-event probabilities should be calibrated and subjected to threshold and episode-onset tests before they control capacity. Hardware allocation should then be derived from serving-stack replay or benchmark measurements that include prefill/decode throughput, KV-cache memory, batching, TTFT/TBT targets, and restart behavior.

For this trace, static P90/P95 is reliable but prohibitively expensive; recent-window P95-24 and tuned MA-6 provide stronger dynamic reliability than the learned point controller; and the burst reserve reduces shortage severity only slightly. Vidur replay further shows that the same trace window can require different A100 and H100 counts once latency and restarts are considered. Effective LLM capacity planning therefore combines causal workload measurement, conservative backtesting, calibrated but limited event signals, and hardware-specific validation.

Data and Code Availability

BurstGPT v1.1 was released on June 13, 2024 under CC BY 4.0. The release page is https://github.com/HPMLL/BurstGPT/releases/tag/v1.1 and the archival record is https://doi.org/10.5281/zenodo.11638888. Direct source files are available at:

- https://github.com/HPMLL/BurstGPT/releases/download/v1.1/BurstGPT_1.csv
- https://github.com/HPMLL/BurstGPT/releases/download/v1.1/BurstGPT_2.csv
- https://github.com/HPMLL/BurstGPT/releases/download/v1.1/BurstGPT_without_fails_1.csv
- https://github.com/HPMLL/BurstGPT/releases/download/v1.1/BurstGPT_without_fails_2.csv

The Vidur source snapshot is https://github.com/microsoft/vidur/commit/8383d2935bc62723a212090baa9f98ada206fc14. Analysis scripts, feature dictionaries, hyperparameter grids, run metadata, predictions, capacity-controller searches, replay commands, and machine-readable table outputs are available from the corresponding author. Input checksums, split indices, software versions, and random seeds are reported in Tables 2–4.

REFERENCES

Agrawal, A., Kedia, N., Mohan, J., Panwar, A., Kwatra, N., Gulavani, B. S., Ramjee, R., & Tumanov, A. (2024). Vidur: A large-scale simulation framework for LLM inference. Proceedings of Machine Learning and Systems, 6, 351-366. https://proceedings.mlsys.org/paper_files/paper/2024/file/b74a8de47d2b3c928360e0a011f48351-Paper-Conference.pdf

Barroso, L. A., Clidaras, J., & Hölzle, U. (2013). The datacenter as a computer: An introduction to the design of warehouse-scale machines (2nd ed.). Morgan & Claypool.

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., Chatterji, 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.

Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5-32. https://doi.org/10.1023/A:1010933404324

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. In Advances in Neural Information Processing Systems, 33, 1877-1901.

Calheiros, R. N., Masoumi, E., Ranjan, R., & Buyya, R. (2015). Workload prediction using ARIMA model and its impact on cloud applications' QoS. IEEE Transactions on Cloud Computing, 3(4), 449-458. https://doi.org/10.1109/TCC.2014.2350475

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM Computing Surveys, 41(3), 1-58. https://doi.org/10.1145/1541880.1541882

Chen, S., He, S., & Sun, E. (2024). Risk-bounded GPU resource oversubscription via conformal demand envelopes in production AI clusters. Journal of Advanced Computing Systems, 4(5), 119–134. https://doi.org/10.69987/JACS.2024.40509

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

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A.,

Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., … Fiedel, N. (2023). PaLM: Scaling language modeling with Pathways. Journal of Machine Learning Research, 24(240), 1-113.

Crankshaw, D., Wang, X., Zhou, G., Franklin, M. J., Gonzalez, J. E., & Stoica, I. (2017). Clipper: A low-latency online prediction serving system. In 14th USENIX Symposium on Networked Systems Design and Implementation (pp. 613-627).

Davis, J., & Goadrich, M. (2006). The relationship between precision-recall and ROC curves. In Proceedings of the 23rd International Conference on Machine Learning (pp. 233-240). https://doi.org/10.1145/1143844.1143874

Dean, J., & Barroso, L. A. (2013). The tail at scale. Communications of the ACM, 56(2), 74-80. https://doi.org/10.1145/2408776.2408794

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. The Annals of Statistics, 29(5), 1189-1232.

Gujarati, A., Karimi, R., Alzayat, S., Hao, W., Kaufmann, A., Vigfusson, Y., & Mace, J. (2020). Serving DNNs like clockwork: Performance predictability from the bottom up. In 14th USENIX Symposium on Operating Systems Design and Implementation (pp. 443-462).

Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. In Proceedings of the 34th International Conference on Machine Learning (pp. 1321-1330).

He, S., Chang, X., & Sun, E. (2024). Cross-cloud transfer learning for AI training capacity forecasting under workload and topology distribution shift. Journal of Advanced Computing Systems, 4(1), 100–120. https://doi.org/10.69987/JACS.2024.40108

He, S., Tu, H., & Liu, I. (2023). Safe PD capacity forecasting with time-series foundation models and calibrated uncertainty for heterogeneous GPU clusters. Journal of Advanced Computing Systems, 3(4), 48–66. https://doi.org/10.69987/JACS.2023.30404

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

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

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. In Advances in Neural Information Processing Systems, 30.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., & Stoica, I. (2023). Efficient memory management for large language model serving with PagedAttention. In Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles (pp. 611-626). https://doi.org/10.1145/3600006.3613165

Li, C., Bai, J., & Wang, S. (2024). Evidence-chain reliable RAG: Word-level hallucination detection, source attribution, and provenance explanation for LLM applications. Journal of Advanced Computing Systems, 4(2), 76–92. https://doi.org/10.69987/JACS.2024.40207

Li, C., Su, W., & Zhang, E. (2023). Lightweight hallucination firewall for enterprise LLM applications: Evidence consistency, self-checking, and small-model detection on TruthfulQA.

Journal of Advanced Computing Systems, 3(1), 49–65. https://doi.org/10.69987/JACS.2023.30104

Liu, G., He, S., & Liu, I. (2023). LLM-augmented multi-source root cause attribution for CPU and network faults in microservices. Journal of Advanced Computing Systems, 3(6), 39–57. https://doi.org/10.69987/JACS.2023.30604

Liu, G., Li, C., & Zhang, E. (2024). OpsLLM for cloud incident triage: Bilingual RAG-based root cause analysis and alert summarization for AI infrastructure operations. Journal of Advanced Computing Systems, 4(4), 97–111. https://doi.org/10.69987/JACS.2024.40408

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In Advances in Neural Information Processing Systems, 30.

Nie, J., & Zheng, D. (2023). Ambiguity-aware HDFS log anomaly detection with retrieval-augmented failure narratives and selective refusal. Journal of Advanced Computing Systems, 3(1), 66–80. https://doi.org/10.69987/JACS.2023.30105

Nie, J., & Zheng, D. (2024). Noisy-neighbor-aware VM degradation risk modeling with unsupervised residual fusion. Journal of Advanced Computing Systems, 4(4), 112–123. https://doi.org/10.69987/JACS.2024.40409

Ousterhout, K., Wendell, P., Zaharia, M., & Stoica, I. (2015). Sparrow: Distributed, low latency scheduling. In Proceedings of the 24th ACM Symposium on Operating Systems Principles (pp. 69-84). https://doi.org/10.1145/2517349.2522716

Pope, R., Douglas, S., Chowdhery, A., Devlin, J., Bradbury, J., Heek, J., Xiao, K., Agrawal, S., & Dean, J. (2023). Efficiently scaling transformer inference. Proceedings of Machine Learning and Systems, 5, 606-624.

Shazeer, N. (2019). Fast transformer decoding: One write-head is all you need. arXiv. https://arxiv.org/abs/1911.02150

Stankevičiūtė, K., Alaa, A. M., & van der Schaar, M. (2021). Conformal time-series forecasting. In Advances in Neural Information Processing Systems, 34, 6216-6228. https://proceedings.neurips.cc/paper_files/paper/2021/hash/312f1ba2a72318edaaa995a67835fad5-Abstract.html

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

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., & Lample, G. (2023). LLaMA: Open and efficient foundation language models. arXiv. https://arxiv.org/abs/2302.13971

Tu, H., Zhao, S., & He, S. (2024). LLM-augmented salable GPU supply forecasting for disaggregated recommendation serving: Predicting instance readiness, scheduling delay, and capacity risk. Journal of Advanced Computing Systems, 4(9), 85–102. https://doi.org/10.69987/JACS.2024.40908

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

Wang, Y., Chen, Y., Li, Z., Kang, X., Fang, Y., Zhou, Y., Zheng, Y., Tang, Z., He, X., Guo, R., Wang, X., Wang, Q., Zhou, A. C., & Chu, X. (2025). BurstGPT: A real-world workload dataset to optimize LLM serving systems. In Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (pp. 5831-5841). ACM. https://doi.org/10.1145/3711896.3737413

Yu, G.-I., Jeong, J. S., Kim, G.-W., Kim, S., & Chun, B.-G. (2022). Orca: A distributed serving system for Transformer-Based Generative Models. In 16th USENIX Symposium on Operating Systems Design and Implementation (pp. 521-538).

Zhang, B., Rao, H., & Zhao, D. (2024). Evidence-grounded RAG for cloud-native DevOps: Hallucination-resistant AIOps question answering over private operations documents. Journal of Advanced Computing Systems, 4(3), 109–125. https://doi.org/10.69987/JACS.2024.40308