PT Jurnal Cendekia Indonesia
Journal of Information Systems and Business Technology
Homepage: https://journal.jci.co.id/jisbt
Vol. 04 No. 02 (2026) Page: 90-100
E-ISSN: 3109-8886

Conformal GPU Demand Envelopes and Power-Aware Scheduling for Heterogeneous AI Clusters under Cold Start and Cross-Cloud Shift

Sijia Chen[1], Marcus Reed[2], Hong Zhang[3]

[1]Department of Computer Science, University of Florida, Gainesville, FL, USA
[2]Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA
[3]Department of Computer Science, University of Utah, Salt Lake City, UT, USA

Email: [1*]sijia_chen_uf@icloud.com

Abstract

Production LLM traffic is bursty, and capacity decisions couple GPU availability with power and carbon. We convert probabilistic demand forecasts into risk-controlled admission, heterogeneous allocation, calibrated overbooking, and cross-site dispatch. The evaluation uses all 1,429,737 requests in the 61-day BurstGPT trace and SustainDC carbon, weather, workload, configuration, and physical power model. Requests form a complete five-minute grid with a weighted-token target. A chronological split assigns 42 days to fitting, nine to conformal calibration, and ten to testing. Point models include persistence, seasonality, Ridge, histogram gradient boosting, and Extra Trees; interval models include direct quantiles, split, rolling, regime-conditioned, and conformalized quantile regression. Histogram gradient boosting attained 49.68% WAPE versus 115.81% for daily seasonality and reduced mean absolute error by 31,545.90 weighted tokens/bin (95% block-bootstrap interval, 23,152.07–37,746.37). Rolling conformal achieved 90.00% coverage at mean width 86,728.29. With a calibration-selected 0.80 overbooking factor, the proposed policy recorded 5.11% bin-level SLO violations and reduced emissions from 52.51 to 12.57 tCO2e through SustainDC-aware dispatch. The remaining 8.81% workload shortfall quantifies burst risk; GPU type, site, and price remain scenario inputs.

Keywords: AI Cluster Scheduling, Conformal Prediction, GPU Demand Forecasting, Heterogeneous Accelerators, Cold Start, Cross-Cloud Dispatch, Power-Aware Computing, Risk-Controlled Overbooking, Burstgpt, Sustaindc.

Abstrak

*Lalu lintas LLM dalam lingkungan produksi bersifat fluktuatif (melonjak-lonjak), dan keputusan kapasitas mengaitkan ketersediaan GPU dengan aspek daya serta emisi karbon. Kami mengubah prakiraan permintaan probabilistik menjadi mekanisme penerimaan yang mengontrol risiko, alokasi heterogen, overbooking yang terkalibrasi, serta distribusi tugas lintas lokasi. Evaluasi ini menggunakan seluruh 1.429.737 permintaan dari data trace BurstGPT selama 61 hari serta model SustainDC yang mencakup aspek karbon, cuaca, beban kerja, konfigurasi, dan daya fisik. Permintaan disusun dalam kisi lima menit yang lengkap dengan target weighted-token. Pembagian data secara kronologis menetapkan 42 hari untuk tahap penyesuaian model (fitting), sembilan hari untuk kalibrasi konformal, dan sepuluh hari untuk pengujian. Model titik (point models) yang digunakan meliputi persistensi, musiman, Ridge, histogram gradient boosting, dan Extra Trees; sedangkan model interval mencakup kuantil langsung, split, rolling, kondisi rezim (regime-conditioned), dan regresi kuantil konformal. Metode histogram gradient boosting mencapai nilai WAPE 49,68% dibandingkan 115,81% pada metode musiman harian, serta mengurangi mean absolute error sebesar 31.545,90 weighted-token*bin (interval block-bootstrap 95%: 23.152,07–37.746,37). Metode rolling conformal mencapai cakupan 90,00% dengan lebar rata-rata 86.728,29. Dengan faktor overbooking 0,80 yang dipilih melalui kalibrasi, kebijakan yang diusulkan mencatat pelanggaran SLO tingkat bin sebesar 5,11% dan mengurangi emisi dari 52,51 menjadi 12,57 tCO2e melalui distribusi tugas yang mempertimbangkan parameter SustainDC. Kekurangan beban kerja sebesar 8,81% mencerminkan risiko lonjakan (burst risk); sementara jenis GPU, lokasi, dan harga tetap menjadi input skenario.*

***Kata Kunci**: Penjadwalan Klaster AI, Prediksi Konformal, Prakiraan Permintaan GPU, Akselerator Heterogen, Cold Start, Distribusi Lintas-Cloud, Komputasi Sadar-Daya, Overbooking Terkendali-Risiko, BurstGPT, SustainDC.*

A. INTRODUCTION

Interactive generative-AI services turn token streams into capacity planning. Minute-scale arrivals and unknown output length can create queues that outlive bursts; model mix shifts memory pressure. Operators choose accelerators with different throughput, memory, power, and price under site-varying electrical conditions. Forecast uncertainty must therefore feed admission and scaling.

BurstGPT records production Azure OpenAI requests rather than synthetic arrivals [1]. Its fields are elapsed timestamp, model identity, request, response and total

Diterima Redaksi: 01-08-2026 | Selesai Revisi: 06-08-2026 | Diterbitkan Online: 07-08-2026

tokens, and log type. SustainDC adds carbon intensity, EPW weather, workload data, and a physical power model [2]. BurstGPT supplies observed demand; SustainDC supplies exogenous site conditions and power calculations. The pairing does not imply that Azure hardware, region, latency, or electricity price appears in the trace.

Forecasting spans boosting [3], temporal convolutions [4], foundation models [5], calibrated GPU capacity [6], and multi-horizon workload semantics [7]. GPU-memory estimation complements demand prediction [8], while seasonal-regime probabilistic forecasting tests exogenous drift [9]. Few-shot inference forecasting [10] and shared-feature transfer [11] motivate cold-start and shift tests; hybrid deployment changes where demand is served [12]. We compare persistence, regularized lags, and nonlinear models chronologically.

Conformalized quantile regression corrects conditional quantiles [13]; adaptive methods handle shift [14], sequential methods address dependence [15], and localized procedures relax exchangeability [16]. Confidence-calibrated fusion illustrates heterogeneous-signal uncertainty [17], while GPU demand envelopes bound oversubscription risk [18]. We compare static, rolling, regime-conditioned, and quantile-based intervals by coverage and width.

Serving systems address memory fragmentation [19], iteration scheduling [20], model-parallel multiplexing [21], prefill/decode separation [22], live rescheduling [23], phase-specific hardware [24], cold loading [25], preemptible capacity [26], and GPU heterogeneity [27], [28]. Complementary trace-driven work links prediction to spot admission [29], power-aware inventory [30], and noisy-neighbor risk [31]. This study inserts a calibrated envelope before admission and placement.

InferLine, FA2, and MArk connect provisioning to latency, SLA, and cost objectives [32], [33], [34]. μ-Serve and DynamoLLM add power and energy constraints [35], [36]; Carbon Explorer, carbon-aware computing, and CarbonScaler optimize facility or workload carbon [37], [38], [39]. The evaluation unifies these objectives through prediction error, interval quality, SLO risk, utilization, power, carbon, cost, and profit metrics.

We contribute a leakage-safe chronological protocol; point and interval tests across burst, horizon, regime, and cold-start settings; calibration-selected heterogeneous overbooking; and four-site SustainDC replay with the official 1 MW model. The design exposes coverage, SLO, shortfall, cost, and carbon tradeoffs (Figure 1).

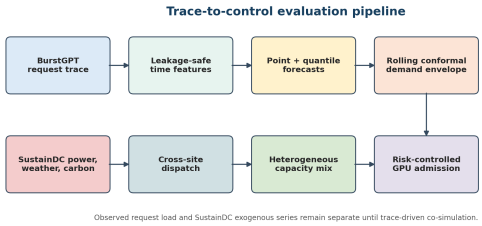

Figure 1 Trace-to-control pipeline with separate request-demand and SustainDC exogenous inputs.

A GPU-equivalent unit normalizes weighted-token demand; it is not a benchmarked accelerator throughput. Efficient, Balanced, and Fast are scenario pool names rather than vendor product names. Cost and profit are normalized monetary units (NMU), not provider invoices. Cross-site results are counterfactual replays over observed demand and SustainDC environmental series.

B. METHODS

Data Audit And Demand Construction

BurstGPT_1 has 1,429,737 records over 61 days and complete required columns, as documented by the source dataset [1]. We retain 25,427 zero-total failed requests and 54,040 exact duplicates because no identifier supports safe deduplication (Table 1). SustainDC contributes 17 carbon files, 11 EPW files, and three workload files [2]; carbon and weather cover 8,760 hours, and workloads span 8,760–8,904 hours.

Table 1 Dataset Audit And Preprocessing Decisions.

Dataset	Record type	Count	Fields	Missing values	Coverage / treatment
BurstGPT_1	Requests	1,429,737	6	0	61 days
BurstGPT_1	Zero-total failed requests	25,427	1	0	1.78% of rows
BurstGPT_1	Exact duplicate rows retained	54,040	6	0	Concurrent events are not deduplicated
SustainDC	Carbon-intensity files	17	10	0	8760-8760 hourly rows/file
SustainDC	EPW weather files	11	35	0	8760-8760 hourly rows/file
SustainDC	Workload files	3	2	0	8760-8904 hourly rows/file

Note: Counts and treatments follow the source-file audit.

The trace comprises four observed service streams formed by Model × Log Type: ChatGPT API, GPT-4 API, ChatGPT conversation, and GPT-4 conversation. Table 2 shows that ChatGPT API accounts for 77.43% of requests, while conversation traffic has markedly higher failure rates: 5.52% for ChatGPT and 5.50% for GPT-4, versus 1.38% and 1.00% for the two API streams. The file contains no tenant or account identifier; consequently, the cold-start

evaluation holds out service classes rather than constructing unsupported tenant labels.

Table 2 Observed Burstgpt Service-Stream Composition

Stream	Requests	Share (%)	Median input tokens	Median output tokens	P95 total tokens	Failure rate (%)
ChatGPT | API log	1,106,998	77.43	218.00	24.00	3,009.00	1.38
GPT-4 | API log	168,520	11.79	460.00	38.00	2,208.00	1.00
ChatGPT | Conversation log	103,111	7.21	481.00	210.00	2,326.00	5.52
GPT-4 | Conversation log	51,108	3.57	520.00	222.00	2,761.00	5.50
All streams	1,429,737	100.00	251.00	34.00	2,801.00	1.78

Note: A zero total-token value is counted as a failed request.

Output-token generation is more compute intensive than reading an input token in autoregressive inference. The target for bin t is therefore

$$y_t = \sum_{i \in t} (r_i + 4o_i)$$

where r_i and o_i are request and response tokens. Four is a fixed workload conversion, not a hardware measurement. Integer timestamp division assigns events to bins; missing bins are zero-filled. The main series has 17,567 five-minute bins with mean 89,833.18, median 19,820.00, 95th percentile 467,788.80, 99th percentile 1,056,931.16, and maximum 2,452,346 weighted tokens. One- and fifteen-minute grids support sensitivity analysis. Figure 2 shows the daily volume and heavy-tailed distribution.

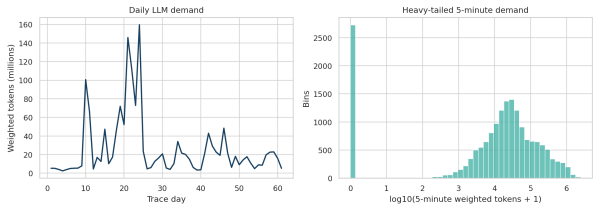

Figure 2 Burstgpt Demand Profile: Daily Weighted-Token Volume And The Heavy-Tailed Distribution Of Complete Five-Minute Bins.

Chronological Protocol And Leakage-Safe Features

The chronological protocol uses elapsed days 1–42 for fitting, days 43–51 for conformal calibration, and days 52–61 for final testing. Weekly-lag construction leaves 10,080 usable training bins, 2,592 calibration bins, and 2,879 test bins at five-minute resolution. No random split mixes future and past observations. Model selection and the overbooking factor use only the training or calibration partitions; test observations enter only metric calculation and sequential rolling updates after each decision.

At resolution m, one day contains d = 1440/m bins. Features include lags 1, 2, 3, 6, 12, 24, d/4, d/2, d, 2d, and 7d; shifted rolling mean, standard deviation, and maximum; daily and weekly sine/cosine pairs; and normalized trend. Duplicate lags are removed. Every feature at t uses only earlier observations. At horizon h, origin features predict.

$$y_{t+h-1}$$

Point and quantile forecasting

Five point methods cover increasing complexity. Last value returns y_{t-1}; seasonal naive returns y_{t-d}. Ridge-lag standardizes features, fits log(1 + y), and uses penalty 10. Histogram GBDT uses squared-error log-target loss, 160 iterations, learning rate 0.055, 31 leaves, minimum leaf size 30, and L2 regularization 2. Extra Trees uses 60 trees, depth 18, minimum leaf size 4, and 60% of features per split. Both use seed 20260804; negative inverse transforms are clipped to zero. The resolution–horizon study uses 80 GBDT iterations with other settings unchanged.

The interval base learner fits gradient-boosted 5th and 95th conditional quantiles on the log target using 100 trees, learning rate 0.045, depth 3, and minimum leaf size 25. The two-sided construction follows conformalized quantile regression [13]. For nominal coverage $1 - \alpha = 0.90$, static split conformal computes calibration scores $s_j = |y_j - \hat{y}_j|$ and the finite-sample higher quantile

$$q = Q_{\lceil (n+1)(1-\alpha) \rceil / n}(s_1, \ldots, s_n)$$

The interval is $[\max(0, \hat{y}_t - q), \hat{y}_t + q]$. Rolling conformal uses an up-to-14-day residual window initialized with the nine calibration days; it reaches the full window after five test days, updates after each observed outcome, and then drops the oldest residual. This sequential update follows time-series conformal logic [15] while allowing recent errors to respond to shift [14]. Regime-conditioned conformal partitions calibration residuals by three predicted-demand levels and four time-of-day blocks; cells with fewer than 50 points use the global quantile, a pragmatic localization consistent with nonexchangeable conformal concerns [16]. Conformalized quantile regression scores $s_j = \max(\hat{q}_{0.05,j} - y_j, y_j - \hat{q}_{0.95,j})$ and expands both quantile endpoints by the finite-sample score quantile [13].

Point metrics are WAPE, MAE, RMSE, sMAPE, P95 absolute error, and burst WAPE. Interval metrics are coverage, upper-tail violations, mean and normalized width, Winkler score, and endpoint pinball loss; together they distinguish sharp calibration from conservative width.

Cold-Start And Distribution-Shift Evaluation

Each cold-start test holds out one of four service streams at fifteen-minute resolution. The other streams provide source training on days 1–42; the held-out class contributes 48 adaptation hours from day 43, followed by ten test days. Lag, shifted-statistic, and phase features use only prior data. Target-window normalization separates shape from scale. Methods are daily seasonal, 48-hour local Ridge, pooled Ridge with target-scale normalization, and pooled Ridge with affine correction. Training quantiles bound predictions before rescaling. Limited-history inference forecasting [10] and approximately shared-feature transfer [11] motivate the pooled baselines. Because BurstGPT

contains neither tenant identifiers nor physical-cloud topology, this protocol tests model/log-class shift, not physical cross-cloud transfer.

GPU-Equivalent Admission And Heterogeneous Allocation

The controller maps weighted tokens to a normalized GPU-equivalent load. One unit equals one eighth of the training-set 99th-percentile five-minute demand, or 140,666.24375 weighted tokens per five minutes. This scale makes capacity readable while avoiding a claim about a specific GPU. The high-memory demand fraction is derived from the training ratio of GPT-4 weighted tokens and clipped to 0.65; its realized value is 0.65.

Three pools form the inventory. Efficient supplies 0.18 capacity at 0.015/0.072 kW idle/peak and 0.11 NMU/h; Balanced supplies 0.55 at 0.060/0.400 kW and 0.24 NMU/h; Fast supplies 1.00 at 0.080/0.700 kW and 0.40 NMU/h. Inventories remain 100/50/30 throughout the test. Power ordering assigns high-memory demand through Balanced–Fast and general demand through Efficient–Balanced–Fast, favoring smaller incremental power blocks. Cost ordering uses Fast–Balanced–Efficient, while the homogeneous ablation fills Fast first. Gavel [40] and Pollux [41] optimize heterogeneous accelerator allocation with effective-throughput or goodput models; Helix and Mélange extend heterogeneity to LLM serving [27], [28]. Our fixed pool rates remain declared scenario inputs, so the study does not claim measured per-accelerator speedups. Utilization is served demand divided by capacity. Full-period and second-half metrics reveal temporal risk variation without changing the declared inventory.

Six policies are compared: Reactive +20%, Point GBDT +15%, direct 95th-quantile GBDT, static 95% conformal, rolling 95% conformal, and the proposed rolling-envelope policy with heterogeneous power ordering and four-site dispatch. Factors 0.80–1.00 in 0.01 steps are tested on the second calibration half against 5% upper violations. The smallest feasible factor, 0.80, is fixed before testing, so admitted capacity is $0.80U_t$ for rolling upper bound U_t. QLM addresses request-level queue operations under SLOs [42], Sarathi-Serve controls prefill/decode batching [43], and Sia optimizes cluster goodput under heterogeneous accelerators [44]. Vidur supports configuration what-if analysis [45]; these queue, batch, scheduler, and simulator layers operate below or complement the five-minute capacity envelope evaluated here. The reserve and admission choices follow the distinct risk-control motivations of conformal oversubscription [18] and spot admission [29].

Sustaindc Power And Cross-Site Co-Simulation

The SustainDC physical model is instantiated at its 1 MW data-center capacity [2]. For reproducible and efficient replay, its IT and HVAC calculations are evaluated on a grid of utilization from 0 to 100% in one-percentage-point increments and dry-bulb temperature from −20 to 50 °C in one-degree increments; bilinear interpolation produces each five-minute value. GPU-pool power is converted to a fraction of the declared normal-inventory peak and supplied as utilization to the facility model. Total facility power includes IT and HVAC. Energy is the five-minute integral of power, and carbon equals energy times contemporaneous grid carbon intensity. Power-aware serving [35], [36] and geo-distributed LLM sustainability scheduling [46] motivate reporting both service and environmental objectives.

California, Washington, Texas, and Virginia use their SustainDC carbon files and matching San Jose, Seattle–Tacoma, Dallas–Fort Worth, and Leesburg EPW files. Hourly values are repeated into five-minute bins. Because BurstGPT timestamps are elapsed seconds rather than calendar datetimes, the 61 demand days align deterministically to the first 61 days of each annual SustainDC series. This is a controlled counterfactual alignment, not a claim that the Azure requests originated at those sites or dates.

The dispatcher starts in California, observes current site signals, enforces a six-bin dwell time, and migrates when another site lowers instantaneous carbon rate by more than 5%. Each completed migration adds 0.05 NMU. MAST demonstrates global GPU demand-supply balancing [47], while SkyPilot provides an intercloud brokerage context [48]. Carbon-aware systems motivate spatial placement [37], [38], [39], but empirical limits on temporal and spatial workload shifting require conservative interpretation [49]. Here, the one-switch replay is a controlled sensitivity analysis, not a deployed migration system. Compute cost is the pool-hour sum, and the normalized electricity term is 0.02 NMU per MWh. Revenue equals served GPU-equivalent load, so profit is revenue minus compute, electricity, and migration indices. These normalized quantities compare policies consistently and do not represent public-cloud prices. Scheduling metrics are bin-level SLO violation, demand-weighted workload shortfall, mean active-capacity utilization, facility power, energy, carbon, cost, profit, migration count, and second-half risk.

Statistical Analysis And Reproducibility

All experiments use seed 20260804. Prediction and scheduling contrasts preserve temporal pairing. A 2,000-replicate moving-block bootstrap with one-day blocks estimates 95% intervals for the seasonal-minus-GBDT absolute-error difference and the static-minus-proposed carbon and cost differences. A one-sided paired Wilcoxon test provides a distribution-free check for the point-error contrast. The run records the BurstGPT SHA-256 digest, SustainDC commit, software platform, split, calibrated constants, runtime, full test predictions, scheduler time series, and every table and figure; the retained artifacts support downstream diagnosis and audit..

C. RESULTS AND DISCUSSION

Point Forecasting Accuracy

Table 3 reports five-minute test accuracy. Histogram GBDT led with 49.68% WAPE, MAE 23,701.45, and RMSE 54,523.94 weighted tokens; last value reached 50.10% WAPE but higher RMSE (56,549.36) and P95 error (104,982.50 versus 98,787.38). Extra Trees, Ridge-lag, and seasonal naive reached 52.80%, 104.13%, and 115.81% WAPE. The nonlinear covariates improved the central forecast, consistent with trace-based GPU forecasting [6], [7].

Table 3 Five-Minute One-Step Point-Forecast Performance On The Ten-Day Test Period.

Model	WAPE (%)	MAE	RMSE	sMAPE (%)	P95 abs. error	Burst WAPE (%)	Fit time (s)
Histogram GBDT	49.68	23,701.45	54,523.94	90.86	98,787.38	51.24	530.193
Last value	50.10	23,899.95	56,549.36	64.85	104,982.50	41.59	0.000
Extra Trees	52.80	25,185.68	57,425.24	92.77	102,773.44	56.12	1.073
Ridge-lag	104.13	49,674.00	101,914.58	134.87	177,238.30	87.63	0.010
Seasonal naive	115.81	55,247.35	105,019.83	110.34	171,335.90	75.03	0.000

Note: Burst WAPE uses test bins above the 90th percentile of realized demand.

In the highest demand decile, last value achieved 41.59% burst WAPE versus 51.24% for Histogram GBDT and 75.03% for seasonal naive. Aggregate rank therefore did not guarantee burst protection, motivating calibrated upper bounds (Figure 3).

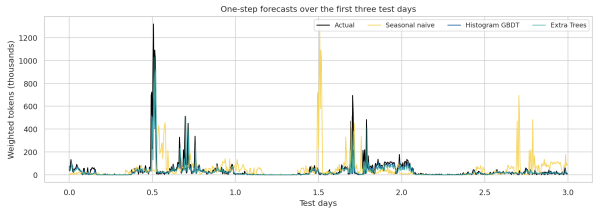

Figure 3 One-Step Forecasts Over The First Three Test Days.

Table 4 separates denominator effects. Extra Trees led low demand (175.41% WAPE; RMSE 8,598.27) and narrowly led medium demand (54.48% versus 55.79%). Histogram GBDT led high demand at 45.95% WAPE, versus 50.15% for Extra Trees and 72.81% for seasonal naive, supporting its use as the conformal center.

Table 4 Point-Forecast Accuracy By Realized Demand Tercile

Regime	Model	Bins	WAPE (%)	RMSE	sMAPE (%)
Low	Seasonal naive	960	1,653.59	87,719.98	134.84
Low	Ridge-lag	960	265.50	14,757.01	164.66
Low	Histogram GBDT	960	183.08	9,987.28	155.61
Low	Extra Trees	960	175.41	8,598.27	156.51
Medium	Seasonal naive	959	185.76	85,471.24	98.22
Medium	Ridge-lag	959	144.33	55,474.46	116.91
Medium	Histogram GBDT	959	55.79	22,794.09	60.86
Medium	Extra Trees	959	54.48	21,018.10	60.34
High	Seasonal naive	960	72.81	134,474.85	97.93
High	Ridge-lag	960	93.01	166,904.01	123.04
High	Histogram GBDT	960	45.95	91,086.27	56.07
High	Extra Trees	960	50.15	96,821.02	61.43

Note: Regimes are terciles of realized test demand.

Forecast resolution and horizon

Predictability declined with horizon (Table 5). Histogram GBDT WAPE increased from 47.85% to 74.54% across 1-6 min, 49.50% to 69.75% across 5-30 min, and 54.62% to 69.72% across 15-90 min. Seasonal naive exceeded 103% in every 15-min setting and 115% on the 5-min grid, matching the risk-curve motivation in multi-horizon cluster forecasting [7].

Table 5 Resolution And Forecast-Horizon Sensitivity.

Resolution (min)	Steps	Horizon (min)	Model	WAPE (%)	RMSE	Burst WAPE (%)
1	1	1	Seasonal naive	126.05	23,284.76	76.70
1	1	1	Histogram GBDT	47.85	9,855.01	39.02
1	3	3	Seasonal naive	126.61	23,365.02	77.16
1	3	3	Histogram GBDT	63.31	13,438.53	62.92
1	6	6	Seasonal naive	126.76	23,443.51	77.51
1	6	6	Histogram GBDT	74.54	15,472.49	77.51
5	1	5	Seasonal naive	115.81	105,019.83	75.03
5	1	5	Histogram GBDT	49.50	54,273.97	50.68
5	3	15	Seasonal naive	116.48	105,016.62	73.51
5	3	15	Histogram GBDT	65.48	68,092.53	73.95
5	6	30	Seasonal naive	116.47	105,630.57	73.68
5	6	30	Histogram GBDT	69.75	73,298.70	78.91
15	1	15	Seasonal naive	105.34	268,279.60	68.22
15	1	15	Histogram GBDT	54.62	147,832.24	58.54
15	3	45	Seasonal naive	104.38	270,143.89	66.29
15	3	45	Histogram GBDT	66.53	185,207.49	67.45
15	6	90	Seasonal naive	104.00	268,182.94	69.06
15	6	90	Histogram GBDT	69.72	189,013.58	73.52

Note: Each horizon is evaluated from the same chronological boundaries.

Interval calibration and sharpness

Table 6 shows three interval regimes. Raw quantile GBDT and CQR were identical because the calibration correction was zero; both covered 95.31% with 4.69% upper misses and width 118,277.88. Static conformal covered 92.12% at width 99,927.98; regime conditioning covered 90.07% at 109,040.06. Rolling conformal reached 89.9965%

(effectively 90.00%), with the smallest conformal width, 86,728.29.

Table 6 Ninety-Percent Predictive-Interval Calibration And Sharpness.

Method	Nominal (%)	Coverage (%)	Upper miss (%)	Mean width	Norm. width (%)	Winkler score	Pinball loss
Raw 5%-95% quantile GBDT	90.00	95.31	4.69	118,277.88	247.94	171,954.91	4298.873
Conformalized quantile regression	90.00	95.31	4.69	118,277.88	247.94	171,954.91	4298.873
Regime-conditioned conformal	90.00	90.07	9.41	109,040.06	228.58	214,125.11	5353.128
Static split conformal	90.00	92.12	6.50	99,927.98	209.47	217,636.24	5440.906
Rolling conformal (14-day window)	90.00	90.00	7.95	86,728.29	181.80	226,681.87	5667.047

Note: All intervals have 90% nominal two-sided coverage.

Figures 4-5 show coverage and the two-day rolling envelope. Rolling conformal traded 2.12 coverage points for a 13.21% narrower interval than static conformal. Its 7.95% upper-miss rate differed from total misses because lower-bound failures do not cause undercapacity. The scheduler therefore uses a separately calibrated 95% upper envelope, aligning interval control with oversubscription risk [18] rather than two-sided coverage alone.

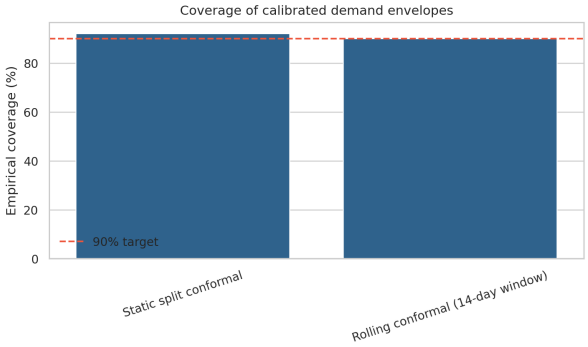

Figure 4 Empirical Coverage Of The Calibrated Demand Envelopes Relative To The 90% Target

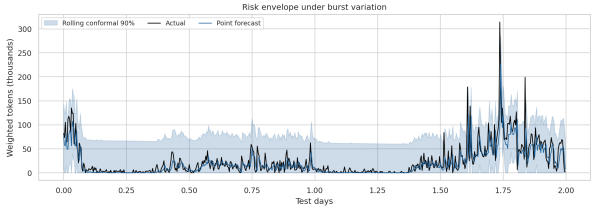

Figure 5 Rolling Conformal Demand Envelope During Two Test Days.

Cold-Start Transfer Across Service Classes

Table 7 and Figure 6 show heterogeneous cold-start transfer. Affine-adapted pooled Ridge led conversation traffic at 45.58% WAPE for ChatGPT and 56.77% for GPT-4; local Ridge reached 51.17% and 60.27%. Aligned source, 48-hour adaptation, and test windows avoid leakage and match the limited-history [10] and shared-feature [11] settings.

Table 7 Leave-One-Service-Stream-Out Cold-Start Performance.

Held-out stream	Method	WAPE (%)	sMAPE (%)	Burst WAPE (%)	Adapt. (h)
ChatGPT | API log	Daily seasonal	188.53	80.42	92.92	0
ChatGPT | API log	48-hour local Ridge	192.98	158.01	71.14	48
ChatGPT | API log	Pooled Ridge + 48-hour scale normalization	116.60	145.40	81.97	48
ChatGPT | API log	Pooled Ridge + 48-hour affine adaptation	147.74	142.68	93.69	48
ChatGPT | Conversation log	Daily seasonal	65.18	82.85	41.72	0
ChatGPT | Conversation log	48-hour local Ridge	51.17	82.71	51.15	48
ChatGPT | Conversation log	Pooled Ridge + 48-hour scale normalization	52.90	76.23	62.74	48
ChatGPT | Conversation log	Pooled Ridge + 48-hour affine adaptation	45.58	67.19	38.51	48
GPT-4 | API log	Daily seasonal	142.62	86.43	81.87	0
GPT-4 | API log	48-hour local Ridge	978.71	152.93	267.35	48
GPT-4 | API log	Pooled Ridge + 48-hour scale normalization	84.74	132.91	62.89	48
GPT-4 | API log	Pooled Ridge + 48-hour affine adaptation	85.77	162.43	67.43	48
GPT-4 | Conversation log	Daily seasonal	85.93	88.50	58.39	0
GPT-4 | Conversation log	48-hour local Ridge	60.27	89.48	56.52	48
GPT-4 | Conversation log	Pooled Ridge + 48-hour scale normalization	61.05	97.16	64.98	48
GPT-4 | Conversation log	Pooled Ridge + 48-hour affine adaptation	56.77	87.65	43.55	48

Note: Adaptation uses target-class hours 1–48 after 42 days of source-domain training.

API transfer was harder. Scale-normalized pooled Ridge led ChatGPT API at 116.60% WAPE and GPT-4 API at 84.74%; the 48-hour GPT-4 local model was unstable at 978.71%. Affine adjustment helped conversation streams, whereas API streams favored scale-only adaptation but retained large errors.

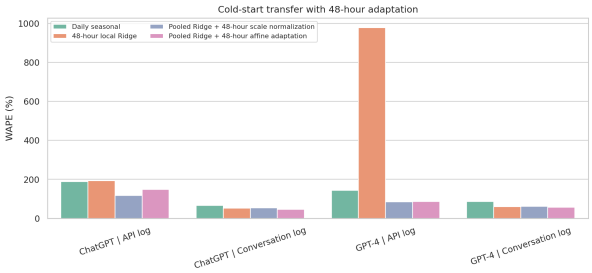

Figure 6 Service-Class Cold-Start Performance With And Without 48-Hour Adaptation

Scheduling, Risk, And Heterogeneous Capacity

Across 2,879 test bins (Table 8), Reactive +20% cost 100.54 NMU with 8.72% SLO violations. Point GBDT +15% reached 5.80% violations and 8.52% shortfall at 106.34 NMU. Direct 95% quantile control was most conservative (1.35%, 1.95%, 142.95 NMU); static and rolling conformal reached 2.47%/5.23%/140.80 and 3.06%/5.86%/133.30, respectively.

Table 8 Trace-Driven Admission, Capacity, Power, And Economic Results.

Policy	SLO bins (%)	Shortfall (%)	GPU util. (%)	Energy (MWh)	Carbon (tCO2e)	Cost (NMU)	Profit (NMU)	Second-half SLO (%)
Reactive +20%	8.72	6.25	29.21	218.79	52.51	100.54	814.74	10.42
Point GBDT +15%	5.80	8.52	27.52	218.79	52.51	106.34	786.85	9.79
Quantile GBDT 95%	1.35	1.95	19.21	218.92	52.54	142.95	814.34	1.88
Static conformal 95%	2.47	5.23	22.38	218.90	52.54	140.80	784.47	3.26
Rolling conformal 95%	3.06	5.86	23.70	218.86	52.53	133.30	785.85	4.44
Proposed power-aware conformal	5.11	8.81	28.03	218.73	12.57	110.54	779.83	8.19

Note: Values are computed on the fixed chronological test period. NMU denotes normalized monetary units.

The proposed controller reached 5.11% violation bins, 8.81% shortfall, 28.03% utilization, and 110.54 NMU cost. Relative to un-overbooked rolling conformal, factor 0.80 saved 22.76 NMU but added 2.05 SLO points and 2.95 shortfall points. Figure 7 exposes this cost-risk frontier, the same decision tension targeted by profit-aware spot admission [29] and power-aware inventory planning [30]. Presenting calibration, policy, risk, and resource context together follows the capacity-dashboard rationale in [50].

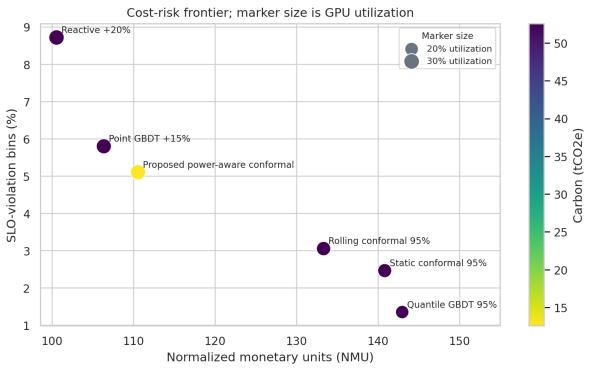

Figure 7 Scheduling Cost–Risk Frontier; Marker Size Encodes Utilization And Color Encodes Carbon.

Second-half violations rose to 8.19% from 5.11% overall, while second-half shortfall fell to 6.64% because earlier bursts carried more missed demand. Direct quantile retained 1.88% violations and 1.19% shortfall. Calibration over the full test therefore did not make risk uniform across regimes.

Cross-site power and carbon

Holding demand and allocation fixed isolates site effects (Table 9). Carbon was 52.51 tCO2e in California, 12.52 in Washington, 82.90 in Texas, and 80.69 in Virginia. One migration yielded 12.57 tCO2e, 76.06% below California; energy changed by less than 0.12 MWh, so carbon intensity drove the gap. Hybrid deployment similarly separates placement from demand [12].

Table 9.

Table 9 Cross-site SustainDC replay under an identical demand envelope

Policy	SLO bins (%)	Shortfall (%)	Energy (MWh)	Carbon (tCO2e)	Cost (NMU)	Migrations
Fixed California	5.11	8.81	218.77	52.51	110.49	0
Fixed Washington	5.11	8.81	218.73	12.52	110.49	0
Fixed Texas	5.11	8.81	218.73	82.90	110.49	0
Fixed Virginia	5.11	8.81	218.66	80.69	110.49	0
Four-site dispatcher	5.11	8.81	218.73	12.57	110.54	1

Note: Values are computed on the fixed chronological test period. NMU denotes normalized monetary units.

Figure 8 shows site signals and selection over seven days. The dispatcher uses only current-bin information, a 30-minute dwell, and a 5% improvement threshold. Its single switch reflects an observed advantage rather than future knowledge or oscillation.

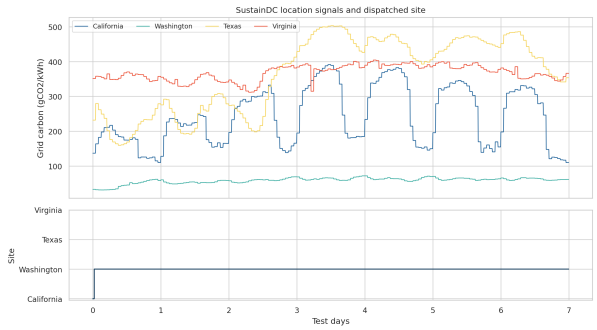

Figure 8 Sustaindc Grid-Carbon Signals And The Dispatched Site During The First Seven Test Days.

Ablation And Statistical Evidence

Table 10 isolates components. Removing conformal control raised violations/shortfall from 5.11%/8.81% to 16.85%/21.01% while reducing cost to 69.24 NMU. Fast-only allocation cost 119.29 NMU; no migration raised carbon to 52.51 tCO2e; no overbooking produced 3.06% violations, 5.86% shortfall, and 133.35 NMU. Reserve, pool mix, migration, and overbooking therefore control distinct risks.

Table 10 Ablation Of The Proposed Controller.

Policy	SLO bins (%)	Shortfall (%)	GPU util. (%)	Carbon (tCO2e)	Cost (NMU)	Profit (NMU)
Full method	5.11	8.81	28.03	12.57	110.54	779.83
No conformal envelope	16.85	21.01	40.44	12.56	69.24	702.00
No heterogeneous pools	3.37	6.05	23.33	12.58	119.29	797.98
No cross-cloud migration	5.11	8.81	28.03	52.51	110.49	779.88
No calibrated overbooking	3.06	5.86	23.70	12.58	133.35	785.81

Note: Values are computed on the fixed chronological test period. NMU denotes normalized monetary units.

Table 11 confirms paired effects. Seasonal naive exceeded GBDT absolute error by 31,545.90 tokens/bin (95% block interval 23,152.07-37,746.37; bootstrap and Wilcoxon p<0.001). Static conformal exceeded proposed carbon by 13.8825 kgCO2e/bin and cost by 0.010510 NMU/bin; both block-bootstrap intervals excluded zero (p<0.001).

Table 11 Paired Moving-Block-Bootstrap And Wilcoxon Evidence.

Paired contrast	Mean difference	95% CI low	95% CI high	Bootstrap p	Wilcoxon p	Unit
Seasonal naive minus Histogram GBDT absolute error	31,545.9002	23,152.0709	37,746.3739	<0.001	<0.001	weighted tokens/bin
Static conformal minus proposed carbon	13.8825	12.7668	15.4658	<0.001	—	kgCO2e/bin
Static conformal minus proposed cost	0.0105	0.0095	0.0118	<0.001	—	NMU/bin

Note: Intervals use 2,000 one-day moving-block-bootstrap replicates; p-values shown as <0.001 were below the reporting precision.

Operational Observability And Human Oversight

Deployment needs observability beyond forecast and dispatch. Conformal alert control provides a compatible but separate path from anomaly scores to rate-controlled, evidence-grounded incident tickets [51]. The retained aligned telemetry also supports multi-source root-cause attribution [52], while grounded operations documents preserve approved evidence [53]. Deep-autoencoder IoT traffic profiling adds a network view [54]; HDFS evidence bundles turn ambiguous scores into auditable narratives [55]. Self-supervised log representations add an event-sequence view [56]. CloudTrail stage detection with linear models and HMM smoothing adds an inspectable audit stream [57], although its assembled multi-stage chains are a controlled baseline rather than production evidence. OpenStack normal-prototype comparisons retain temporal context [58]. These methods diagnose breaches; they do not replace demand prediction.

Oversight also needs explicit interfaces. Risk cards show provenance, permission, consequence, and alternatives before approval [59]. Host system-call localization adds window-level forensic narratives [60]. Incident cards add timelines, severity, confidence, and controls [61]; repeated replanning, tool errors, and loops can trigger takeover [62]. State-changing copilot commands need an independent checker [63].

At scale, cost-aware AIOps routing can reserve expensive analysis for uncertain incidents [64]. These prospective controls were not part of the BurstGPT-SustainDC experiment.

BurstGPT lacks queueing latency, decode rate, GPU memory, accelerator type, region, and energy; the study therefore evaluates a demand-capacity SLO, not latency. Pools and inventories are scenario inputs; SustainDC supplies environment and facility power. GPU-memory prediction [8] is complementary, and hardware claims require accelerator telemetry. Carbon comparisons remain controlled because policies share the same demand and power model.

D. CONCLUSIONS

Using one chronological protocol, Histogram GBDT reduced WAPE from 115.81% for daily seasonality to 49.68%, and rolling conformal achieved 90.00% coverage at width 86,728.29. A 95% admission envelope with calibration-selected factor 0.80 and SustainDC dispatch recorded 5.11% SLO bins, 110.54 NMU, and 12.57 tCO2e; fixed California emitted 52.51 tCO2e.

Demand-weighted shortfall still reached 8.81%, risk shifted over time, and cold-start accuracy varied by service class. Accelerator type, inventory, and price remain scenario inputs; SustainDC alignment is counterfactual. Within these limits, conformal envelopes provide an auditable bridge from forecast error to admission, overbooking, scaling, and power-aware placement.

E. REFERENCES

[1] Y. Wang, Y. Chen, Z. Li, X. Kang, Y. Fang, Y. Zhou, Y. Zheng, Z. Tang, X. He, R. Guo, X. Wang, Q. Wang, A. C. Zhou, and X. Chu, "BurstGPT: A real-world workload dataset to optimize LLM serving systems," in Proc. 31st ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD), vol. 2, pp. 5831–5841, 2025, doi: 10.1145/3711896.3737413.

[2] A. Naug, A. Guillen, R. Luna, V. Gundecha, C. Bash, S. Ghorbanpour, S. Mousavi, A. Ramesh Babu, D. Markovikj, L. D. Kashyap, D. Rengarajan, and S. Sarkar, "SustainDC: Benchmarking for sustainable data center control," in Advances in Neural Information Processing Systems, vol. 37, Datasets and Benchmarks Track, 2024, doi: 10.52202/079017-3192.

[3] G. Ke et al., "LightGBM: A highly efficient gradient boosting decision tree," in Advances in Neural Information Processing Systems, vol. 30, 2017.

[4] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," arXiv preprint arXiv:1803.01271, 2018.

[5] A. Das, W. Kong, R. Sen, and Y. Zhou, "A decoder-only foundation model for time-series forecasting," in Proc. 41st Int. Conf. Machine Learning, PMLR, vol. 235, pp. 10148–10167, 2024.

[6] S. He, H. Tu, and I. Liu, "Safe PD capacity forecasting with time-series foundation models and calibrated uncertainty for heterogeneous GPU clusters," J. Adv. Comput. Syst., vol. 3, no. 4, pp. 48–66, Apr. 2023, doi: 10.69987/JACS.2023.30404.

[7] S. Zhao, J. Bai, and D. Roberson, "Multi-horizon GPU demand forecasting with workload semantics and operational risk curves: An empirical study on Alibaba Clusterdata GPU Trace," J. Technol. Informatics Eng., vol. 4, no. 3, pp. 544–571, Dec. 2025, doi: 10.51903/jtie.v4i3.498.

[8] C. Wang, Z. Wen, R. Zhang, P. Xu, and Y. Jiang, "GPU memory requirement prediction for deep learning task based on bidirectional gated recurrent unit optimization Transformer," in Proc. 5th Int. Conf. Artificial Intelligence, Virtual Reality and Visualization (AIVRV), Chengdu, China, pp. 31–35, 2025, doi: 10.1109/AIVRV67401.2025.11350369.

[9] Q. Xin, "Probabilistic bike-sharing demand forecasting under changing weather and seasonal regimes with transformer-based models," Findings, Mar. 2026, doi: 10.32866/001c.157499.

[10] S. He, C. Li, and H. Rao, "Few-shot cold-start workload forecasting for new AI inference tenants with time-series foundation models," J. Technol. Informatics Eng., vol. 4, no. 1, pp. 306–324, Apr. 2025, doi: 10.51903/jtie.v4i1.546.

[11] Z. S. Zhong, X. Pan, and Q. Lei, "Bridging domains with approximately shared features," in Proc. 28th Int. Conf. Artificial Intelligence and Statistics (AISTATS), PMLR, vol. 258, pp. 559–567, 2025.

[12] Q. Xin, "Hybrid cloud architecture for efficient and cost-effective large language model deployment," J. Inf. Syst. Informatics, vol. 7, no. 3, pp. 2182–2195, Sep. 2025, doi: 10.51519/journalisi.v7i3.1170.

[13] Y. Romano, E. Patterson, and E. J. Candès, "Conformalized quantile regression," in Advances in Neural Information Processing Systems, vol. 32, 2019.

[14] I. Gibbs and E. Candès, "Adaptive conformal inference under distribution shift," in Advances in Neural Information Processing Systems, vol. 34, 2021.

[15] C. Xu and Y. Xie, "Sequential predictive conformal inference for time series," in Proc. 40th Int. Conf. Machine Learning, PMLR, vol. 202, 2023.

[16] R. F. Barber, E. J. Candès, A. Ramdas, and R. J. Tibshirani, "Conformal prediction beyond exchangeability," Ann. Statist., vol. 51, no. 2, pp. 816–845, 2023, doi: 10.1214/23-AOS2276.

[17] Q. Xin, "Uncertainty-aware late fusion for 3D perception (confidence calibration + fusion rule learning)," J. Technol. Informatics Eng., vol. 4, no. 1, pp. 215–238, Apr. 2025, doi: 10.51903/jtie.v4i1.485.

[18] [18] S. Chen, S. He, and E. Sun, "Risk-bounded GPU resource oversubscription via conformal demand envelopes in production AI clusters," J. Adv. Comput. Syst., vol. 4, no. 5, pp. 119–134, May 2024, doi: 10.69987/JACS.2024.40509.

[19] W. Kwon et al., "Efficient memory management for large language model serving with PagedAttention," in Proc. 29th ACM Symp. Operating Systems Principles (SOSP), pp. 611–626, 2023, doi: 10.1145/3600006.3613165.

[20] G.-I. Yu et al., "Orca: A distributed serving system for Transformer-based generative models," in Proc. 16th USENIX Symp. Operating Systems Design and Implementation (OSDI), pp. 521–538, 2022.

[21] Z. Li et al., "AlpaServe: Statistical multiplexing with model parallelism for deep learning serving," in Proc. 17th USENIX Symp. Operating Systems Design and Implementation (OSDI), pp. 663–679, 2023.

[22] Y. Zhong et al., "DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving," in Proc. 18th USENIX Symp. Operating Systems Design and Implementation (OSDI), pp. 193–210, 2024.

[23] B. Sun et al., "Llumnix: Dynamic scheduling for large language model serving," in Proc. 18th USENIX Symp. Operating Systems Design and Implementation (OSDI), pp. 173–191, 2024.

[24] P. Patel et al., "Splitwise: Efficient generative LLM inference using phase splitting," in Proc. 51st Annu. Int. Symp. Computer Architecture (ISCA), pp. 118–132, 2024, doi: 10.1109/ISCA59077.2024.00019.

[25] Y. Fu et al., "ServerlessLLM: Low-latency serverless inference for large language models," in Proc. 18th USENIX Symp. Operating Systems Design and Implementation (OSDI), pp. 135–153, 2024.

[26] X. Miao et al., "SpotServe: Serving generative large language models on preemptible instances," in Proc. 29th ACM Int. Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS), vol. 2, pp. 1112–1127, 2024, doi: 10.1145/3620665.3640411.

[27] Y. Mei et al., "Helix: Serving large language models over heterogeneous GPUs and network via max-flow," in Proc. 30th ACM Int. Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS), vol. 1, pp. 586–602, 2025, doi: 10.1145/3669940.3707215.

[28] T. Griggs et al., "Mélange: Cost efficient large language model serving by exploiting GPU heterogeneity," arXiv preprint arXiv:2404.14527, 2024.

[29] S. Zhao, Y. Ren, and X. Chang, "Profit-aware spot GPU admission control with cost-sensitive loss and evidence-grounded policy memos for AI workload supply-demand matching," J. Technol. Informatics Eng., vol. 5, no. 2, pp. 45–59, 2026, doi: 10.51903/jtie.v5i2.545.

[30] S. He, J. Nie, and C. Li, "Power-aware inventory planning for AI infrastructure using job-level forecasting and LLM workload explanations," J. Technol. Informatics Eng., vol. 5, no. 1, Apr. 2026, doi: 10.51903/jtie.v5i1.548.

[31] J. Nie and D. Zheng, "Noisy-neighbor-aware VM degradation risk modeling with unsupervised residual fusion," J. Adv. Comput. Syst., vol. 4, no. 4, pp. 112–123, Apr. 2024, doi: 10.69987/JACS.2024.40409.

[32] D. Crankshaw et al., "InferLine: Latency-aware provisioning and scaling for prediction serving pipelines," in Proc. 11th ACM Symp. Cloud Computing (SoCC), pp. 477–491, 2020, doi: 10.1145/3419111.3421285.

[33] K. Razavi et al., "FA2: Fast, accurate autoscaling for serving deep learning inference with SLA guarantees," in Proc. IEEE 28th Real-Time and Embedded Technology and Applications Symp. (RTAS), pp. 146–159, 2022, doi: 10.1109/RTAS54340.2022.00020.

[34] C. Zhang et al., "MArk: Exploiting cloud services for cost-effective, SLO-aware machine learning inference serving," in Proc. USENIX Annu. Technical Conf. (USENIX ATC), pp. 1049–1062, 2019.

[35] H. Qiu et al., "Power-aware deep learning model serving with μ-Serve," in Proc. USENIX Annu. Technical Conf. (USENIX ATC), pp. 75–93, 2024.

[36] J. Stojkovic et al., "DynamoLLM: Designing LLM inference clusters for performance and energy efficiency," in Proc. IEEE Int. Symp. High-Performance Computer Architecture (HPCA), pp. 1348–1362, 2025, doi: 10.1109/HPCA61900.2025.00102.

[37] B. Acun et al., "Carbon Explorer: A holistic framework for designing carbon-aware datacenters," in Proc. 28th ACM Int. Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS), vol. 2, pp. 118–132, 2023, doi: 10.1145/3575693.3575754.

[38] A. Radovanović et al., "Carbon-aware computing for datacenters," IEEE Trans. Power Syst., vol. 38, no. 2, pp. 1270–1280, 2023, doi: 10.1109/TPWRS.2022.3173250.

[39] W. A. Hanafy, Q. Liang, N. Bashir, D. Irwin, and P. Shenoy, "CarbonScaler: Leveraging cloud workload elasticity for optimizing carbon-efficiency," Proc. ACM Meas. Anal. Comput. Syst., vol. 7, no. 3, Art. no. 57, 2023, doi: 10.1145/3626788.

[40] D. Narayanan, K. Santhanam, F. Kazhamiaka, A. Phanishayee, and M. Zaharia, "Heterogeneity-aware cluster scheduling policies for deep learning workloads," in Proc. 14th USENIX Symp. Operating Systems Design and Implementation (OSDI), pp. 481–498, 2020.

[41] A. Qiao, S. K. Choe, S. J. Subramanya, W. Neiswanger, Q. Ho, H. Zhang, G. R. Ganger, and E. P. Xing, "Pollux: Co-adaptive cluster scheduling for goodput-optimized deep learning," in Proc. 15th USENIX Symp. Operating Systems Design and Implementation (OSDI), pp. 1–18, 2021.

[42] A. Patke, D. Reddy, S. Jha, H. Qiu, C. Pinto, C. Narayanaswami, Z. Kalbarczyk, and R. Iyer, "Queue management for SLO-oriented large language model serving," in Proc. ACM Symp. Cloud Computing (SoCC), pp. 18–35, 2024, doi: 10.1145/3698038.3698523.

[43] A. Agrawal, N. Kedia, A. Panwar, J. Mohan, N. Kwatra, B. Gulavani, A. Tumanov, and R. Ramjee, "Taming throughput-latency tradeoff in LLM inference with Sarathi-Serve," in Proc. 18th USENIX Symp. Operating Systems Design and Implementation (OSDI), pp. 117–134, 2024.

[44] S. J. Subramanya, D. Arfeen, S. Lin, A. Qiao, Z. Jia, and G. R. Ganger, "Sia: Heterogeneity-aware, goodput-optimized ML-cluster scheduling," in Proc. 29th ACM Symp. Operating Systems

Principles (SOSP), pp. 642–657, 2023, doi: 10.1145/3600006.3613175.

[45] A. Agrawal, N. Kedia, J. Mohan, A. Panwar, N. Kwatra, B. S. Gulavani, R. Ramjee, and A. Tumanov, "Vidur: A large-scale simulation framework for LLM inference," Proc. Mach. Learn. Syst., vol. 6, pp. 351–366, 2024.

[46] H. Moore, S. Qi, N. Hogade, D. S. Milojicic, C. E. Bash, and S. Pasricha, "Sustainable carbon-aware and water-efficient LLM scheduling in geo-distributed cloud datacenters," in Proc. 35th Great Lakes Symp. VLSI (GLSVLSI), pp. 929–934, 2025, doi: 10.1145/3716368.3735301.

[47] A. Choudhury et al., "MAST: Global scheduling of ML training across geo-distributed datacenters at hyperscale," in Proc. 18th USENIX Symp. Operating Systems Design and Implementation (OSDI), pp. 563–580, 2024.

[48] Z. Yang et al., "SkyPilot: An intercloud broker for sky computing," in Proc. 20th USENIX Symp. Networked Systems Design and Implementation (NSDI), pp. 437–455, 2023.

[49] T. Sukprasert, A. Souza, N. Bashir, D. Irwin, and P. Shenoy, "On the limitations of carbon-aware temporal and spatial workload shifting in the cloud," in Proc. 19th European Conf. Computer Systems (EuroSys), pp. 924–941, 2024, doi: 10.1145/3627703.3650079.

[50] G. Liu, S. He, and H. Wong, "LLM-compatible visual brief cards for AI infrastructure capacity dashboards: A UI/UX framework for turning forecast risk into graphic design decisions," Int. J. Graph. Des., vol. 3, no. 1, pp. 196–213, May 2025, doi: 10.51903/ijgd.v3i1.3723.

[51] Q. Xin, "Log anomaly detection with conformal alert control and evidence-grounded incident ticket generation," AVITEC, vol. 8, no. 2, pp. 247–264, 2026, doi: 10.28989/avitec.v8i2.3974.

[52] G. Liu, S. He, and I. Liu, "LLM-augmented multi-source root cause attribution for CPU and network faults in microservices," J. Adv. Comput. Syst., vol. 3, no. 6, pp. 39–57, Jun. 2023, doi: 10.69987/JACS.2023.30604.

[53] B. Zhang, H. Rao, and D. Zhao, "Evidence-grounded RAG for cloud-native DevOps: Hallucination-resistant AIOps question answering over private operations documents," J. Adv. Comput. Syst., vol. 4, no. 3, pp. 109–125, Mar. 2024, doi: 10.69987/JACS.2024.40308.

[54] Q. Xin, Z. Xu, L. Guo, F. Zhao, and B. Wu, "IoT traffic classification and anomaly detection method based on deep autoencoders," Appl. Comput. Eng., vol. 69, no. 1, pp. 64–70, Jul. 2024, doi: 10.54254/2755-2721/69/20241511.

[55] [55] X. Sun, Z. S. Zhong, and Q. Wu, "Retrieval-grounded HDFS log anomaly detection and deterministic failure narrative generation," J. Comput. Syst. Appl., vol. 3, no. 1, pp. 15–30, Jun. 2026, doi: 10.64229/j6d7fr94.

[56] Q. Xin, "Self-supervised log anomaly detection with LogBERT-style transformers: Full empirical evaluation on a reproducible SynHDFS benchmark," J. Electr. Eng. Comput. Sci., vol. 11, no. 1, pp. 23–35, May 2026, doi: 10.54732/jeecs.v11i1.3.

[57] J. Bai, S. Chen, D. Zheng, and M.-J. Kuo, "Interpretable attack-chain stage detection from AWS CloudTrail event sequences via linear models and HMM smoothing," Inf. Electr. Electron. Eng., vol. 6, no. 1, pp. 28–43, May 2026, doi: 10.33474/infotron.v6i1.24923.

[58] Q. Xin, "Explaining OpenStack failure-injection log anomalies with retrieved normal prototypes," Emerg. Inf. Sci. Technol., vol. 6, no. 2, pp. 125–146, Nov. 2025, doi: 10.18196/eist.v6i2.31232.

[59] W. Su, H. Rao, and E. Ma, "Privacy and data-integrity risk cards for LLM agents: A UI/UX design framework for secure human oversight under prompt-injection attacks," Int. J. Graph. Des., vol. 4, no. 1, pp. 186–191, Apr. 2026, doi: 10.51903/ijgd.v4i1.3699.

[60] [60] Q. Xin, "Host-based intrusion detection with system call sequences: Window localization and forensic narratives," AVITEC, vol. 8, no. 2, pp. 325–334, Aug. 2026, doi: 10.28989/avitec.v8i2.3973.

[61] J. Nie, G. Liu, C. Li, and T. Zou, "Evidence-constrained incident visualization cards for distributed cloud logs: A UI/UX framework for turning Hadoop, OpenStack, and ZooKeeper logs into actionable SRE design interfaces," Int. J. Graph. Des., vol. 4, no. 1, pp. 179–185, Apr. 2026, doi: 10.51903/ijgd.v4i1.3703.

[62] Z. S. Zhong, C. Li, and H. Rao, "Trajectory reliability prediction for generalist AI agents: Tool-use failure analysis and success forecasting on ZClawBench," J. Technol. Informatics Eng., vol. 5, no. 1, pp. 341–360, Apr. 2026, doi: 10.51903/jtie.v5i1.539.

[63] B. Zhang, X. Sun, G. Liu, and B. Zhou, "LLM-style DevOps copilot for cloud-native troubleshooting: Retrieval-augmented runbook generation and command-safety evaluation," J. Technol. Informatics Eng., vol. 5, no. 2, pp. 104–118, Aug. 2026, doi: 10.51903/jtie.v5i2.534.

[64] C. Li, G. Liu, and Z. Zhao, "Cost-aware LLM-style routing for AIOps log analysis: Log parsing, anomaly detection, fault diagnosis, and incident summarization on LogEval task files," J. Technol. Informatics Eng., vol. 5, no. 2, pp. 91–103, 2026, doi: 10.51903/jtie.v5i2.538.