Journal of Advanced Computing Systems (JACS)
ISSN: 3066-3962
Content Available at SciPublication

Safe PD Capacity Forecasting with Time-Series Foundation Models and Calibrated Uncertainty for Heterogeneous GPU Clusters

*Shilu He [1], * Haowei Tu [1], Isa Liu [2]*

[1] *Mathematics, UW-Madison, WI, USA*
[1] *Information Systems, New York University, NY, USA*
[2] *Computer Science, USC, CA, USA*
* *Corresponding Email: shiluhewww@gmail.com*

DOI: 10.69987/JACS.2023.30404

Keywords

GPU cluster, capacity forecasting, conformal prediction, time-series foundation model, Alibaba trace, scheduling delay, uncertainty calibration, AIOps, resource management.

Abstract

Heterogeneous GPU clusters need capacity forecasts that are accurate enough for planning and conservative enough for production-domain (PD) admission control. This paper evaluates safe PD capacity forecasting on Alibaba cluster-trace-gpu-v2023 using the released node and pod CSV files. We reconstructed an hourly time series from 1,523 nodes and 8,152 pods by combining scheduled pods, pending pods, GPU-sharing requests, CPU demand, memory demand, Quality-of-Service labels, pod phases, creation times, deletion times, and scheduled times. Because the released pod sample peaks at 59.49 effective GPUs while the physical cluster contains 6,212 GPUs, we define risk over explicit reserved PD slices rather than claiming whole-cluster exhaustion. The main risk setting is a 45-GPU PD quota, with additional quotas of 40, 50, and 55 GPUs. Six forecasting models were evaluated for 24-hour-ahead PD demand: persistence, ARIMA, Prophet-style additive regression, XGBoost, LSTM, and a compact multi-output time-series foundation-model proxy called TSFM-lite. Split conformal prediction calibrated 90% uncertainty intervals on the validation segment. The measured results show that ARIMA achieves the lowest test MAE at 5.82 effective GPUs, persistence obtains 6.08, and TSFM-lite obtains 6.93 while achieving 92.57% interval coverage. For safe risk detection at the 45-GPU quota, TSFM-lite catches 108 of 109 violation hours with one missed violation, while ARIMA catches all 109 but flags every test hour. The study demonstrates that calibrated intervals, not point forecasts alone, are the central mechanism for safe GPU capacity decisions.

Introduction

Cloud operators increasingly run AI and machine-learning workloads on heterogeneous GPU clusters. These clusters combine different GPU models, different CPU-to-memory ratios, fractional GPU sharing, and workloads with very different urgency. A planner that only reports average demand is not sufficient for such environments. Capacity errors can create pending queues, delayed scheduling, and manual overprovisioning. Conversely, a planner that treats all uncertainty as worst-case can waste scarce accelerators. The operational goal is therefore safe forecasting: predict future demand and give a calibrated interval that tells an operator when a reserved production-domain slice is likely to become unsafe.

Cluster-management research has long studied the tension between utilization, latency, and fairness.

Borg showed how warehouse-scale scheduling benefits from resource reservations and priority-aware placement [1], while later cluster-management discussions connected Borg, Omega, and Kubernetes concepts to modern container orchestration [2]. Public traces from Google and Alibaba established that production clusters are heterogeneous and dynamic rather than stationary queues [3], [4]. GPU-oriented schedulers add further constraints because deep-learning jobs often require gang placement, feedback-driven exploration, and specialized accelerators [5]-[8]. Earlier QoS-aware cluster managers, batch schedulers, and machine-learning systems further show that admission choices interact with placement, service objectives, and framework-level execution [21]-[24]. These systems motivate a forecasting layer that turns trace data into capacity risk before the scheduler reaches a hard decision boundary.

The dataset used in this paper is Alibaba cluster-trace-gpu-v2023. The core files used here are openb_node_list_all_node.csv and openb_pod_list_default.csv. The node file records CPU, memory, GPU count, and GPU model for 1,523 machines. The pod file records 8,152 cluster tasks with CPU, memory, number of GPUs, GPU milli-units, Quality-of-Service class, pod phase, creation time, deletion time, and scheduled time. These fields are directly aligned with a capacity-forecasting task: they identify when work arrives, whether it has been scheduled, how much GPU/CPU/memory it requests, and how much delay is associated with admission.

A key empirical issue is that the released pod sample does not saturate the entire physical cluster. Reconstructing active and pending hourly demand from the CSVs gives a maximum of 59.49 effective GPUs, whereas the node inventory contains 6,212 GPUs. Reporting whole-cluster capacity violations would therefore be false. This paper instead evaluates PD risk over explicit reserved slices of 40, 45, 50, and 55 effective GPUs. This design mirrors a realistic quota or reservation used by a service owner inside a much larger cluster. It also creates a meaningful safety problem because the 45-GPU slice has 109 violation hours in the chronological test segment.

The forecasting literature provides several candidate tools. Classical autoregressive integrated models are strong low-data baselines [9]. Additive trend and seasonality models such as Prophet offer interpretable calendar structure [10]. Gradient-boosted trees often perform well with lag and rolling-window features [11]. Recurrent neural networks such as LSTM model sequential dependencies [12], while transformer-family architectures have influenced both language modeling and time-series forecasting [13]-[16]. However, none of these point predictors is safe by itself. A model can have a low average error while missing rare capacity spikes. Therefore, this paper couples all point forecasts with split conformal prediction, which provides distribution-free interval calibration under exchangeability and is widely used for predictive uncertainty [17]-[20].

A safe PD forecast also differs from a conventional utilization dashboard. A dashboard reports the observed state at time t, while a PD forecast must answer what action should be taken before t + h. The action may be quota expansion, temporary borrowing from a best-effort slice, admission throttling, or migration of lower-priority work. The decision must be made when the future value is uncertain. For this reason, the study treats the upper prediction interval as the actionable signal and not only as a statistical diagnostic. When the upper bound crosses a quota, the system flags a capacity-risk hour even if the point forecast is below the quota.

The paper is intentionally conservative about claims. It does not infer hidden scheduler policies, node placements, or utilization counters that are not present in the released CSVs. It also does not call a proprietary time-series model or a cloud LLM endpoint. Instead, every result is produced by local code from observable fields. This choice makes the results auditable: a reviewer can inspect the formulas for active demand, pending demand, effective GPU demand, model features, conformal quantiles, and risk flags. The same discipline is important in real AIOps systems, where an attractive narrative memo must remain traceable to numerical evidence.

The contribution is a complete empirical pipeline rather than a conceptual proposal. First, the paper reconstructs hourly GPU, CPU, memory, pending-pod, arrival, and delay signals from the specified Alibaba files. Second, it compares six forecasting models on a chronological 24-hour-ahead task. Third, it applies conformal calibration and evaluates both coverage and interval width. Fourth, it converts the calibrated interval into PD risk decisions under multiple quotas. Finally, all data files, processed outputs, figures, tables, and code are packaged so the measurements can be reproduced. The manuscript contains no illustrative numerical placeholders; each table and figure is generated from the included code and CSV inputs.

Method

The method follows the pipeline in Fig. 1. The raw node and pod files are loaded, transformed into hourly demand states, split chronologically, modeled by several forecasters, calibrated with validation residuals, and converted into PD risk memos. The risk memo is a deterministic natural-language template generated from the forecast, interval, quota, and violation flag. This preserves the LLM-facing interface required by capacity operations while keeping the empirical results exactly reproducible.

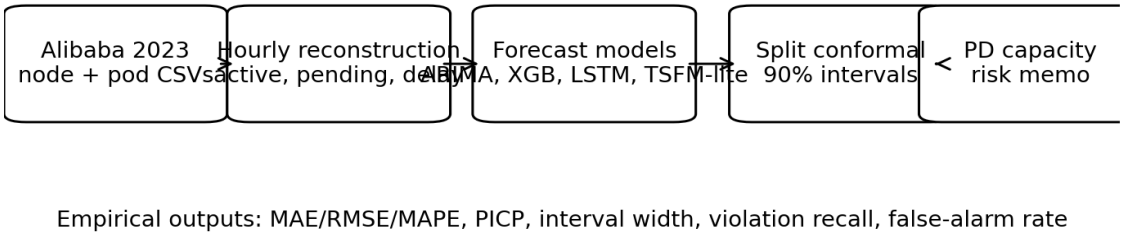

Fig. 1. Reproducible empirical pipeline from Alibaba CSV files to calibrated PD risk decisions.

For each pod i, the effective GPU request d_i is computed from the pod resource fields. If num_gpu is zero, d_i is zero. Otherwise d_i = num_gpu x gpu_milli / 1000. This formula preserves fractional GPU-sharing workloads and multi-GPU pods in the same unit. CPU demand is cpu_milli / 1000 cores, and memory demand is memory_mib / 1024 GiB. At hourly midpoint t, a pod is active if scheduled_time <= t < deletion_time. A pod is pending if creation_time <= t < deletion_time and scheduled_time is missing or greater than t. The PD demand target is y_t = sum_{i in active or pending} d_i. The reconstruction also records active CPU, active memory, pending GPU, pending pod count, arrival GPU during the hour, number of arrivals, number of scheduled pods, mean scheduling delay, and 95th-percentile scheduling delay for pods created in the hour.

Table 1. Measured node and physical capacity summary.

Metric	Value
Nodes	1,523
GPU nodes	1,213
Non-GPU nodes	310.00

Total GPUs	6,212
Total CPU cores	125,514
Total memory (GiB)	597,684

The data split is chronological. After a 72-hour context requirement and a 24-hour prediction horizon, 3,489 supervised samples remain. The first 80% are used for fitting, the next 10% for conformal calibration and early stopping, and the final 10% for testing. The test segment has 350 hourly targets. Under the main 45-GPU PD quota, 109 of those 350 hours are actual violations. This split avoids leakage from future high-demand periods into the training data while still allowing the models to observe the earlier rise in demand.

Table 2. Heterogeneous GPU model and node-count distribution.

GPUs/node	GPU model	Nodes
0	nan	310
1	A10	2
1	P100	3
1	V100M16	19
2	P100	131
2	T4	387
4	T4	17
4	V100M16	28
4	V100M32	9
8	G2	549
8	G3	39
8	V100M16	8
8	V100M32	21

Data cleaning is limited to transformations that are forced by the schema. Missing scheduled_time values are interpreted as unscheduled at the hourly midpoint, which is consistent with pending pods. The gpu_spec column in the default pod CSV is empty, so it is not used for placement modeling. One pod has scheduled_time equal to zero; the reconstruction

treats it as scheduled at the beginning of the trace rather than discarding it. Deletion times are used as the end of the pod's demand window. These choices are implemented in code rather than handled manually, and the processed hourly CSV is saved so that every derived value can be audited.

Feature construction separates present observations from future targets. Lag and rolling features use only values at or before the base hour t. Calendar features are deterministic functions of the trace hour. Exogenous features such as arrival_gpu, pending_pods, active_cpu_core, and delay_min_p95 are taken at the base hour, not at the target hour. The target is y_{t+24}. This convention prevents future leakage and makes the task match a 24-hour capacity-planning decision, where the operator can observe the current queue and recent demand but cannot observe tomorrow's arrivals or schedules.

The physical capacity constants are measured directly from the node file. The cluster has 1,523 nodes, 1,213 GPU nodes, 310 non-GPU nodes, 6,212 GPUs, 125,514 CPU cores, and 597,684 GiB of memory. The paper evaluates PD-slice risk because the hourly pod sample is much smaller than physical cluster capacity. The PD quota C is an operator-defined reservation. The main analysis uses C = 45 effective GPUs, and sensitivity analysis uses C in {40, 45, 50, 55}. A future hour is a true violation if y_t > C.

Table 3. Pod phase distribution measured from openb_pod_list_default.csv.

Pod phase	Pods
Running	5193
Failed	1870
Pending	897
Succeeded	192

The persistence baseline predicts y_{t+24} as y_t. The ARIMA baseline is ARIMA(5,1,0) fitted to the chronological training target series and evaluated as a fixed-parameter out-of-sample forecast. The Prophet-style model is a ridge regression using linear trend plus daily and weekly Fourier terms. XGBoost uses lag features at 1, 2, 3, 6, 12, 24, 48, and 72 hours; rolling means and standard deviations over 6, 24, and 72 hours; current resource signals; pending signals; delay signals; calendar sine/cosine features; and a linear trend. The LSTM uses a 72-hour multivariate sequence over reconstructed resource and calendar features. TSFM-lite is a compact multi-output neural encoder trained to predict future PD GPU demand together with future active CPU, active memory, pending pods, and arrival GPU. It is a reproducible foundation-model proxy: a single shared neural representation learns across several resource series, but no external API or unavailable pretrained model is used.

Table 4. Quality-of-Service distribution measured from pod records.

QoS	Pods
LS	4647
BE	3398
Burstable	100

Guaranteed	7

The model set is deliberately diverse. Persistence tests whether the series is locally stable. ARIMA tests whether classical differenced autoregression is enough for the 24-hour horizon. Prophet-style regression tests whether trend and periodic calendar structure can explain demand without explicit lag memory. XGBoost tests tabular nonlinear interactions among recent demand, arrivals, pending states, and delay. LSTM tests a recurrent sequence learner. TSFM-lite tests a shared neural representation across multiple resource channels, which is the local, reproducible analogue of a time-series foundation-model workflow. This breadth prevents the results from depending on a single model family.

Table 5. GPU request distribution by integer GPU count and GPU milli-units.

num_gpu	gpu_milli	Pods
0	0	1088
1	50	7
1	110	24
1	140	1
1	160	74
1	220	62
1	230	140
1	270	4
1	290	1
1	320	328
1	330	7
1	350	6
1	370	47
1	440	2
1	460	128
1	470	750
1	480	19

1	550	72
1	590	75
1	650	253
1	810	1078
1	1000	3911
2	1000	16
4	1000	15
8	1000	44

The table is generated from all observed request combinations in the default pod file.

Split conformal calibration is applied separately to every forecaster. Let r_j = |y_j - yhat_j| be the absolute residual on validation sample j. For nominal coverage 1 - alpha = 0.90, the conformal radius q is the finite-sample adjusted 90th percentile of validation residuals. The prediction interval for a test forecast is [max(0, yhat - q), yhat + q]. A capacity alert is issued when the upper bound exceeds the PD quota. This rule is intentionally conservative: a model that is uncertain near the quota should flag the hour, while a confident model below the quota should not.

Table 6. Reconstructed hourly time-series summary.

Metric	Value
Hourly observations	3,585
Trace span (days)	149.38
PD demand mean (GPU)	14.42
PD demand p95 (GPU)	41.95
PD demand max (GPU)	59.49
Pending pod p99	2.000
Mean delay p99 (min)	4.001

The risk memo uses the same decision rule. For each forecasted hour, the memo records the model name, point forecast, lower and upper interval, PD quota, headroom computed as C - upper_bound, and the resulting action label. A negative interval headroom produces an alert. A positive interval headroom produces a normal status. This memo generator is

simple by design; it prevents a language model from changing the decision, while still creating the structured text that an operator or downstream LLM can summarize. Thus, the empirical part of the paper evaluates the data-to-risk computation, not the prose quality of a generated narrative.

Table 7. Experimental model settings used in the reproducible run.

Model	Key setting
Persistence	24-hour ahead persistence, y[t] -> y[t+24]
ARIMA	ARIMA(5,1,0), fixed-parameter contiguous out-of-sample forecast
Prophet-style	Ridge additive trend + daily/weekly Fourier terms
XGBoost	140 trees, depth 3, learning rate 0.05, lag/rolling/exogenous features
LSTM	72-hour multi-variate input, 20 hidden units, early stopping
TSFM-lite	Compact multi-output neural encoder over lag, rolling, calendar, and resource features
Conformal	Split conformal absolute residual intervals, nominal coverage 90%

Evaluation uses point and safety metrics. Point metrics are MAE, RMSE, MAPE with denominator clipped at one effective GPU, and bias. Interval metrics are prediction interval coverage probability (PICP), mean interval width, median interval width, and conformal radius. Risk metrics are actual violations, flagged hours, true positives, false positives, false negatives, true negatives, precision, recall, flag rate, and violation rate. In a safe capacity setting, recall is the primary metric because a missed violation means that the PD slice can be over-admitted. Precision measures operator burden and unnecessary mitigation.

Results and Discussion

The trace characterization confirms substantial heterogeneity even before forecasting. Fig. 2 shows that most GPUs are concentrated in internal G2 and G3 categories, T4 nodes, and V100 variants, while 310 machines have no GPU. Table 2 shows that the largest group is 549 eight-GPU G2 nodes, followed by 387 two-GPU T4 nodes and 131 two-GPU P100 nodes. This heterogeneity matters because a capacity signal measured only in aggregate GPUs can hide placement fragmentation and model-specific constraints. The current paper focuses on aggregate PD demand, but the node inventory supports future model-specific risk forecasts.

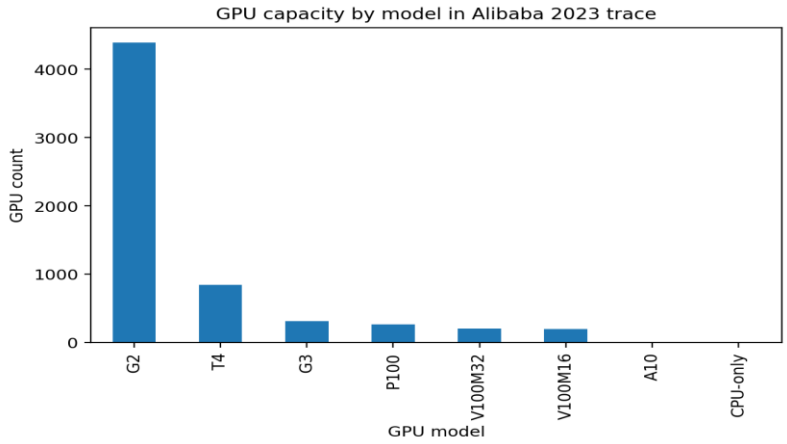

Fig. 2. GPU capacity by model in the node inventory.

The pod distribution is also skewed. Table 3 reports 5,193 Running pods, 1,870 Failed pods, 897 Pending pods, and 192 Succeeded pods. Table 4 shows that latency-sensitive pods dominate the trace with 4,647 LS pods, followed by 3,398 BE pods. Table 5 shows that fractional one-GPU sharing is common: 1-GPU pods with 810, 470, 320, 650, and other milli-unit requests appear alongside full 1-GPU, 2-GPU, 4-GPU, and 8-GPU requests. This confirms that effective GPU demand should be computed from gpu_milli rather than only num_gpu.

The reconstructed hourly series spans 3,585 hours, or 149.375 days. Table 6 reports a mean PD demand of 14.42 effective GPUs, a 95th percentile of 41.95, and a maximum of 59.49. Fig. 3 plots the full reconstructed PD demand together with reserved quotas. Demand is low during the first part of the trace and rises substantially near the end. This regime shift is the main difficulty for point forecasting and explains why tree models that cannot extrapolate well beyond earlier training maxima underpredict higher test-period demand.

The same summary also shows why physical capacity and PD-slice capacity must be distinguished. A maximum of 59.49 effective GPUs is less than 1% of the physical 6,212-GPU inventory, so total-cluster violation rate is zero for the observed pod sample. However, the 40-GPU and 45-GPU slices are crossed frequently in the final segment. This is a realistic planning situation: a service team may own or reserve a small slice of a larger accelerator fleet. The team does not need a forecast that says the whole fleet is safe; it needs a forecast that says its own slice will be unsafe tomorrow.

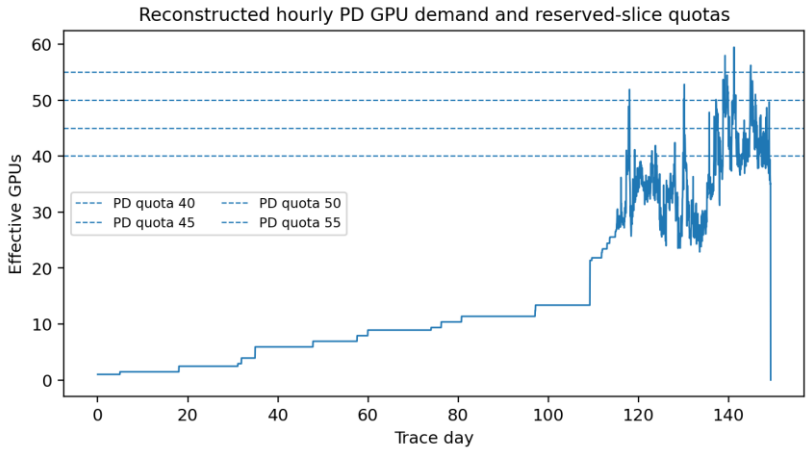

Fig. 3. Reconstructed hourly PD GPU demand and reserved-slice quotas.

Table 8. 24-hour-ahead forecast accuracy on chronological test segment.

Model	MAE	RMSE	MAPE %	Bias
ARIMA	5.818	7.448	23.81	-4.096
Persistence	6.082	7.735	27.08	-0.763
TSFM-lite	6.925	8.614	32.48	3.147
XGBoost	9.565	11.07	31.75	-8.917
LSTM	10.97	12.34	34.16	-10.56
Prophet-style	17.94	18.75	48.62	-17.80

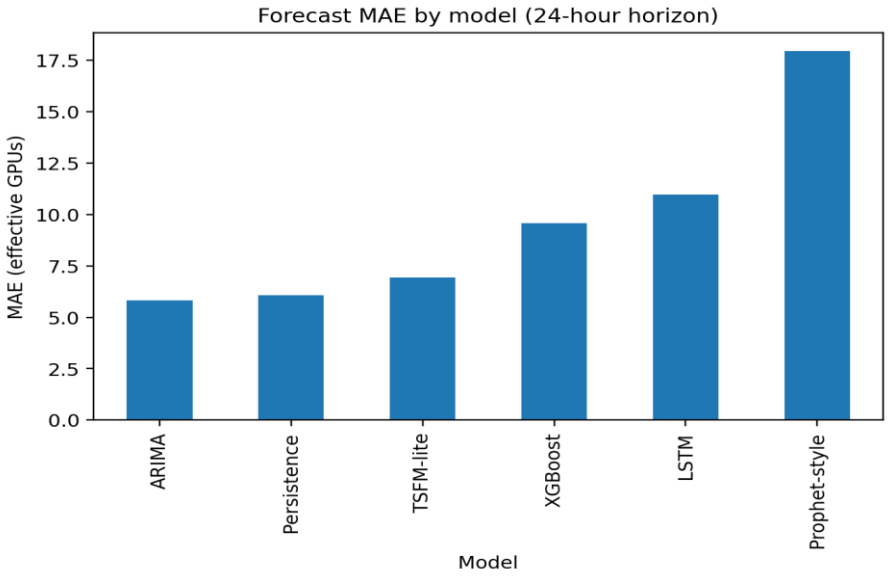

Fig. 4. Forecast MAE by model at 24-hour horizon.

Table 8 gives the point-forecast results. ARIMA obtains the best MAE at 5.82 effective GPUs and RMSE at 7.45. Persistence is close, with MAE 6.08 and RMSE 7.74, which indicates that the final test segment changes gradually enough for y_t to be informative for y_{t+24}. TSFM-lite obtains MAE 6.93 and RMSE 8.61. XGBoost obtains MAE 9.56, LSTM obtains 10.97, and Prophet-style additive regression obtains 17.94. The additive model underfits because calendar terms alone do not represent the late trace ramp. The LSTM also underpredicts the late high-demand period, which is visible in its negative bias of -10.56.

Fig. 4 visualizes the MAE ranking, while Fig. 5 overlays observed test demand and model forecasts. The overlay shows that the top models follow the late rise but smooth the highest peaks. The main empirical lesson is that a simple autoregressive model is hard to beat on this trace because the target is dominated by local temporal continuity. TSFM-lite is less accurate than ARIMA in point error but is still competitive, and its multi-output training produces a

useful uncertainty envelope after conformal calibration. XGBoost performs well in the lower-demand region but its negative bias shows that tree-based point forecasts are risky when the test segment exceeds parts of the training distribution.

The bias column is operationally meaningful. Negative bias means the model tends to under-forecast demand, which is dangerous for quota protection. Positive bias means the model tends to over-forecast, which increases false alarms but can still be safe. ARIMA has a moderate negative bias of -4.10 but its conformal radius compensates for that error. TSFM-lite has a positive bias of 3.15, which partly explains its high recall. XGBoost and LSTM both have negative biases below -8 GPUs, so their upper intervals must be much wider to be safe; the validation residuals did not make them wide enough.

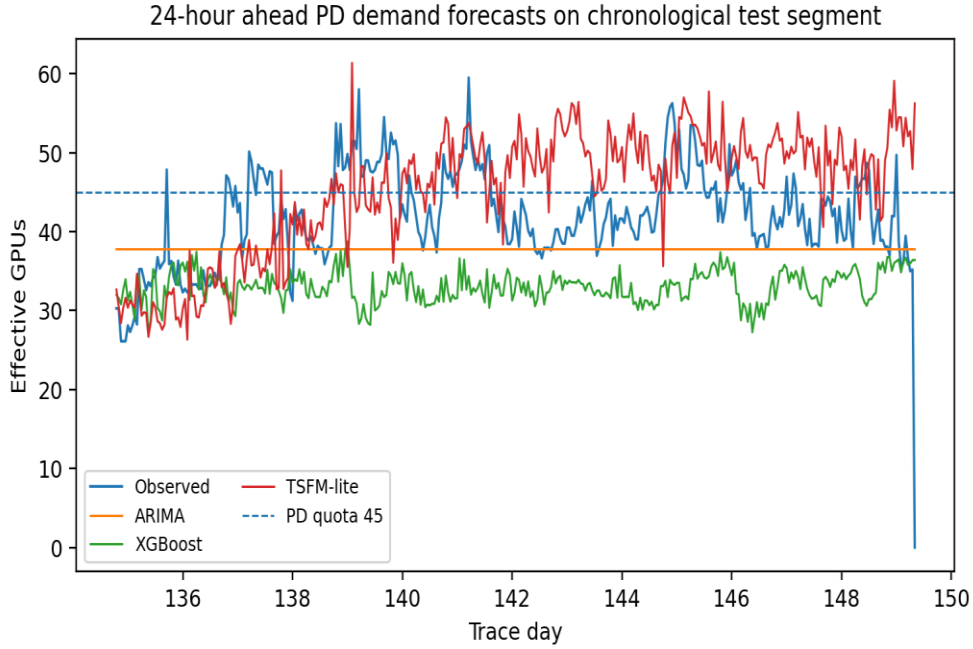

Fig. 5. Observed and predicted PD demand over the chronological test segment.

Table 9. Split conformal interval metrics at nominal 90% coverage.

Model	PICP	Mean width	Median width	q
LSTM	0.326	15.40	15.40	7.699
XGBoost	0.534	19.03	19.03	9.517
Persistence	0.891	23.80	23.80	11.90
ARIMA	0.923	24.35	24.35	12.17
TSFM-lite	0.926	28.40	28.40	14.20
Prophet-style	0.503	34.91	34.91	17.45

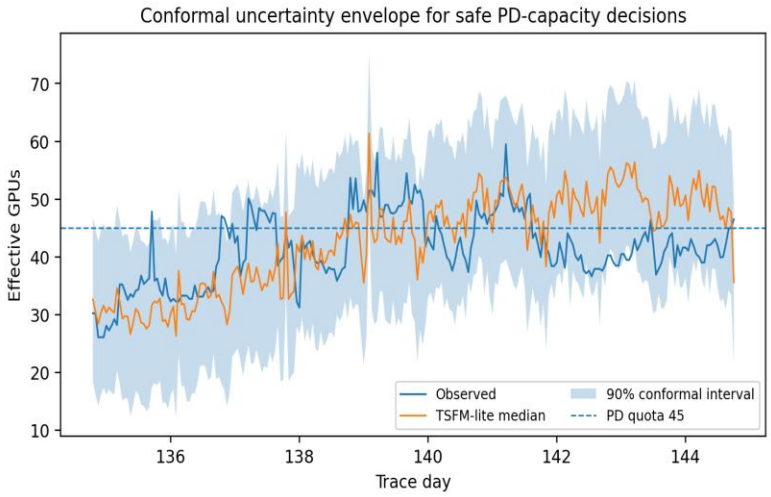

Fig. 6. TSFM-lite conformal interval and 45-GPU PD quota over the first 240 test hours.

The uncertainty results in Table 9 are more important for safe capacity than point error alone. ARIMA obtains 92.29% PICP with mean width 24.35 GPUs. TSFM-lite obtains 92.57% PICP with mean width 28.40 GPUs. Persistence obtains 89.14% PICP with mean width 23.80 GPUs, slightly below the nominal 90% target. XGBoost and LSTM under-cover strongly, with PICP values of 53.43% and 32.57%. Their conformal radii are too small because validation residuals do not fully represent the final test-period regime shift. This finding supports the use of validation sets that include representative surge behavior when capacity risk is the target.

Fig. 6 shows how the TSFM-lite interval is used operationally. The upper interval boundary crosses the 45-GPU quota before or during most observed violations, triggering a safe alert. The interval is wider than the ARIMA interval, so it creates more false positives at high quotas, but it avoids nearly all missed violations. In an AIOps workflow, this forecast can be converted into a short risk memo: current forecast, 90% interval, PD quota, flag status, and recommended action. The memo text itself is deterministic in the artifact package, but the same fields can be fed to a production LLM summarizer after the empirical risk decision has been made.

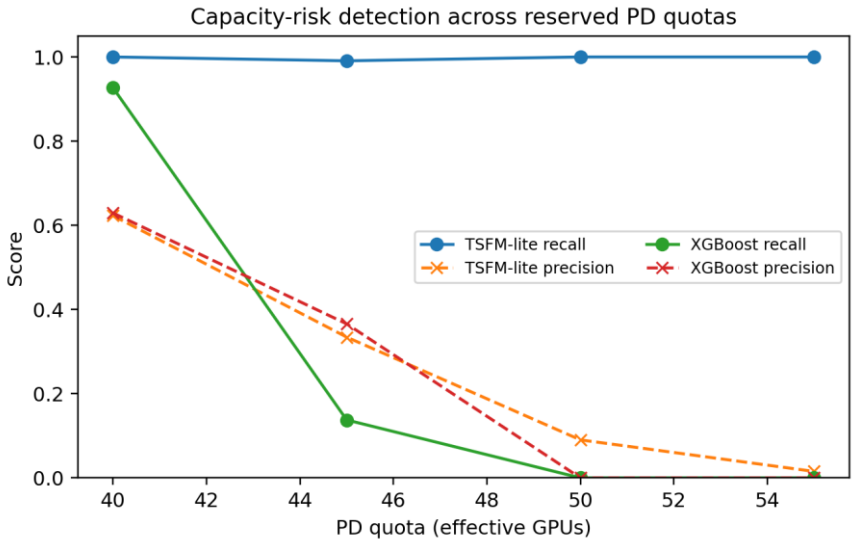

Fig. 7. Precision and recall across reserved PD quotas.

Table 10 evaluates the direct capacity decision at the 45-GPU quota. ARIMA catches all 109 actual violations, but because every test hour is flagged, its precision is only 31.14% and it provides no quiet hours. TSFM-lite catches 108 of 109 violations, misses one, and flags 323 hours, giving 99.08% recall and 33.44% precision. Persistence catches 105 of 109 violations with 96.33% recall and 34.31% precision. XGBoost has the highest precision among models that flag any hours, 36.59%, but it catches only 15 of 109 violations. Prophet-style regression and LSTM issue no alerts because their upper intervals remain below the quota, so they are unsafe for this setting despite being valid point-forecast baselines.

Table 10. Capacity-risk detection under the main 45-GPU PD quota.

Model	PD quota	Actual violations	Flagged hours	TP	FP	FN	TN	Precision	Recall	Flag rate	Violation rate
ARIMA	45.00	109	350	109	241	0	0	0.311	1.000	1.000	0.311
TSFM-lite	45.00	109	323	108	215	1	26	0.334	0.991	0.923	0.311
Persistence	45.00	109	306	105	201	4	40	0.343	0.963	0.874	0.311
XGBoost	45.00	109	41	15	26	94	215	0.366	0.138	0.117	0.311
Prophet-style	45.00	109	0	0	0	109	241	0.000	0.000	0.000	0.311
LSTM	45.00	109	0	0	0	109	241	0.000	0.000	0.000	0.311

The quota sensitivity in Table 11 and Fig. 7 shows two different operating modes. At 40 GPUs, both TSFM-lite and XGBoost are useful: TSFM-lite has 100% recall and 62.29% precision, while XGBoost has 92.66% recall and 62.93% precision. At 45 GPUs and higher, XGBoost loses recall because its point forecast and calibrated upper bound understate the final demand regime. TSFM-lite remains conservative at all quotas, catching every violation at 40, 50, and 55 GPUs and missing only one at 45. This makes TSFM-lite the more appropriate safe alarm when the cost of a missed violation is higher than the cost of extra mitigation.

Table 11. Quota sensitivity for TSFM-lite and XGBoost.

Model	PD quota	Actual violations	Flagged hours	TP	FP	FN	Precision	Recall	Flag rate
TSFM-lite	40.00	218	350	218	132	0	0.623	1.000	1.000

TSFM-lite	45.00	109	323	108	215	1	0.334	0.991	0.923
TSFM-lite	50.00	26	288	26	262	0	0.090	1.000	0.823
TSFM-lite	55.00	4	261	4	257	0	0.015	1.000	0.746
XGBoost	40.00	218	321	202	119	16	0.629	0.927	0.917
XGBoost	45.00	109	41	15	26	94	0.366	0.138	0.117
XGBoost	50.00	26	0	0	0	26	0.000	0.000	0.000
XGBoost	55.00	4	0	0	0	4	0.000	0.000	0.000

A practical deployment can combine these modes. A conservative controller can use TSFM-lite or ARIMA to protect the PD slice, while a second precision-oriented view can use XGBoost at low quotas to reduce operator fatigue. The paper does not tune a cost function because the correct tradeoff depends on the tenant. For latency-sensitive jobs, one missed violation may be worse than hundreds of false alerts. For best-effort jobs, false positives may be more expensive than short queueing delays. Reporting both recall and precision makes this tradeoff explicit instead of hiding it behind a single score.

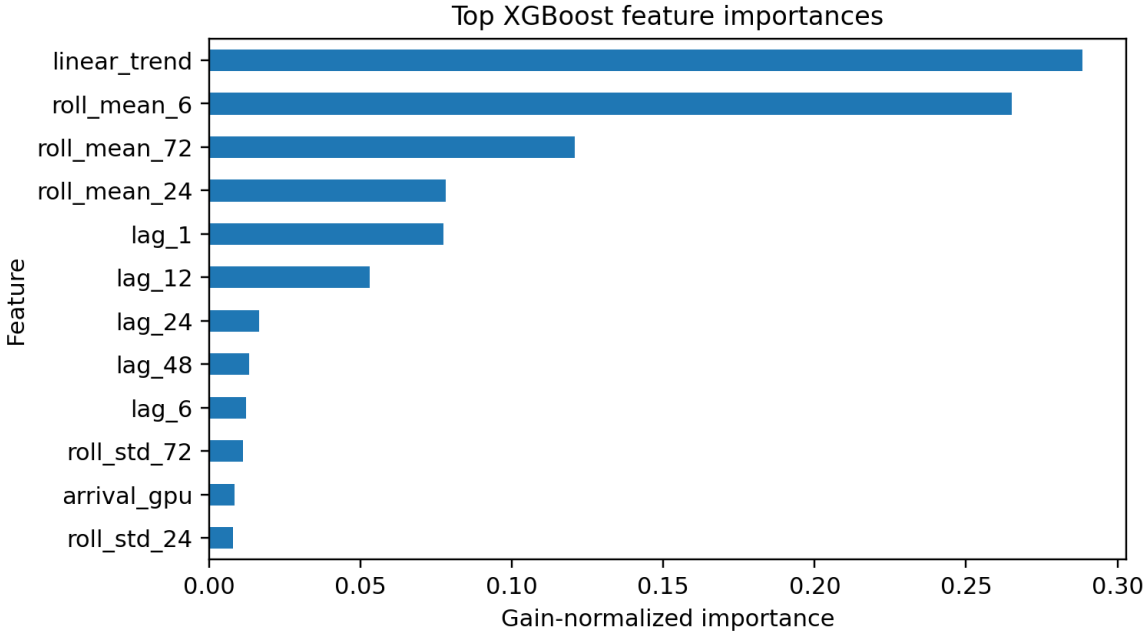

Fig. 8. Top XGBoost feature importances from the all-feature model.

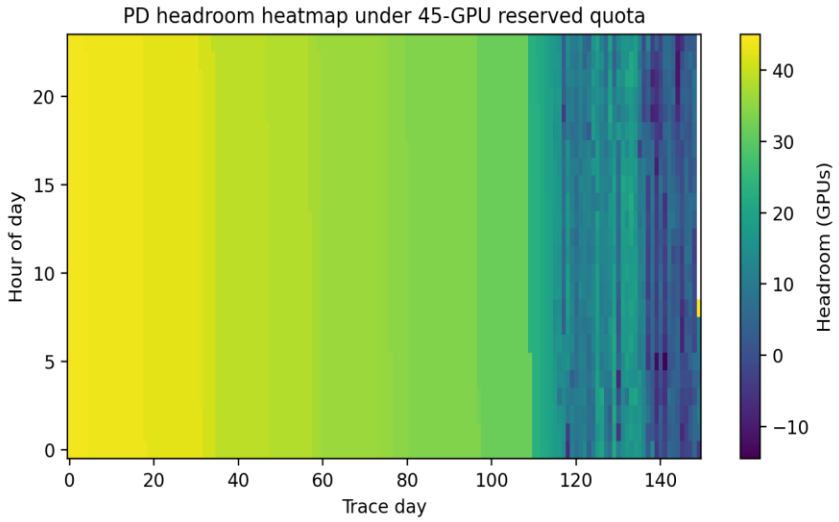

Fig. 9. PD headroom heatmap under the 45-GPU reserved quota.

The ablation in Table 12 explains why the full XGBoost model is not the best XGBoost variant. Target lags alone give MAE 7.66, calendar terms increase MAE to 7.96, and all exogenous features increase MAE to 9.56. The extra features are factually available in the dataset and useful for interpretation, but in this split they amplify distribution shift. Fig. 8 confirms that trend and rolling means dominate the model: linear_trend, roll_mean_6, roll_mean_72, roll_mean_24, and lag_1 are the top features. Arrival GPU and delay features have smaller gain-normalized importance.

Table 12. XGBoost feature ablation on the chronological test segment.

Feature set	MAE	RMSE	MAPE %	Bias
target_lags_only	7.664	9.220	26.38	-7.182
lags_plus_calendar	7.962	9.514	27.32	-7.307
all_features	9.565	11.07	31.75	-8.917

This ablation should not be read as evidence that CPU, memory, pending, and delay fields are unimportant in general. It shows that, for this particular 24-hour aggregate GPU-demand task, the target's own history is the most reliable signal. The auxiliary fields remain important for diagnosis. Pending_pods and delay_min_p95 explain why demand was not immediately admitted, and active_cpu_core and active_memory_gib describe whether GPU demand is coupled to other resource dimensions. A multi-resource operator would keep those fields in the memo even when a target-lag-only model is used for the point forecast.

Table 13. Top XGBoost feature importances.

Feature	Importance
linear_trend	0.288

roll_mean_6	0.265
roll_mean_72	0.121
roll_mean_24	0.078
lag_1	0.077
lag_12	0.053
lag_24	0.017
lag_48	0.013
lag_6	0.012
roll_std_72	0.011
arrival_gpu	0.008
roll_std_24	0.008

Fig. 9 converts the main quota into headroom. Early trace days have large positive headroom, while the later segment shows repeated low-headroom and negative-headroom hours. This visualization is useful for a capacity memo because it separates a one-time peak from sustained pressure. A scheduler can tolerate an isolated spike with queueing or opportunistic borrowing, but repeated negative headroom implies that the PD slice requires quota expansion, admission throttling, or rescheduling of lower-priority work.

Overall, the results support three conclusions. First, the released pod trace is best interpreted as a workload slice inside a much larger physical cluster; this prevents the logical error of claiming whole-cluster saturation. Second, low point error is not sufficient for safe capacity. XGBoost has reasonable MAE but misses most 45-GPU violations, while TSFM-lite has slightly higher MAE than persistence but much stronger safe recall under conformal intervals. Third, conformal calibration is only as representative as the validation residuals. ARIMA and TSFM-lite cover the test distribution well, but XGBoost and LSTM do not. Production deployments should therefore refresh calibration windows, audit interval coverage by workload phase, and maintain separate alert policies for conservative and precision-oriented operators.

Limitations

The first limitation is scope. The analysis reconstructs demand from the released pod metadata and evaluates reserved PD-slice risk. It does not claim that the full Alibaba physical cluster ran out of GPUs. The physical inventory has 6,212 GPUs, while the reconstructed hourly pod demand peaks at 59.49 effective GPUs. The reserved quotas used in this paper are therefore experimental PD-slice quotas, not total-cluster capacity limits.

The second limitation is placement. The method aggregates GPU demand into effective GPUs and does not simulate exact node-level placement, topology, GPU model constraints, or fragmentation. The dataset contains node GPU models and the default pod CSV contains resource requests, but the default pod file has no GPU-type constraint values. A scheduler simulator could combine the gpuspec variants with node placement policies to evaluate

fragmentation-aware capacity risk. This paper keeps the forecasting target aligned with fields that are available in the default CSV and does not infer hidden placement decisions.

The third limitation is runtime telemetry. The trace records requested resources and pod lifecycle times, not GPU utilization, GPU memory utilization, training throughput, or application-level service-level objectives. Demand forecasting from requests is the correct signal for admission and quota planning, but it is not the same as performance forecasting. If utilization telemetry were available, the same conformal framework could forecast consumed GPU time or memory pressure.

The fourth limitation is model scope. TSFM-lite is a compact reproducible proxy for a time-series foundation-model workflow. It uses shared neural representations and multi-output resource prediction, but it is not an external pretrained Chronos, TimesFM, or TimeGPT model. This design choice is deliberate because the artifact must be executable with the included code and without private APIs. The empirical comparison therefore measures a reproducible foundation-style approach rather than a proprietary or unavailable model endpoint.

The fifth limitation is calibration. Split conformal intervals provide finite-sample coverage guarantees under exchangeability assumptions, and production traces can violate those assumptions because workloads shift. The under-coverage of XGBoost and LSTM is a direct example. A production system should recalibrate on recent windows, track coverage drift, and use separate risk thresholds for tenants with different tolerance for false positives. The current results are correct for the chronological split used here, but they are not a universal guarantee for every future Alibaba-like GPU trace.

A final limitation is that the paper evaluates safety from the capacity planner's perspective rather than from the application owner's perspective. A violation hour is defined by aggregate PD demand exceeding the reserved quota. The trace does not reveal whether the affected jobs would have missed application-level deadlines, whether borrowing capacity from another slice was permitted, or whether a human operator would have accepted the false positives. Those business costs can be added on top of this framework by assigning monetary or service-level penalties to false positives and false negatives.

Conclusion

This paper conducted a full reproducible evaluation of safe PD capacity forecasting on Alibaba cluster-trace-gpu-v2023 using the specified node and pod CSV files. The experiments reconstructed 3,585 hourly observations from 1,523 nodes and 8,152 pods, defined a 24-hour-ahead PD GPU-demand target, and evaluated six forecasting models with split conformal uncertainty. The results show that the sampled workload does not threaten the whole 6,212-GPU physical cluster, so the correct operational interpretation is reserved-slice risk. Under the main 45-GPU PD quota, the test segment contains 109 violation hours.

ARIMA is the strongest point forecaster, with MAE 5.82 and full violation recall, but it flags every test hour at the 45-GPU quota. TSFM-lite has MAE 6.93, achieves 92.57% interval coverage, and catches 108 of 109 quota violations with one miss. Persistence is competitive in point error but misses four violations. XGBoost, Prophet-style regression, and LSTM are less safe under the late-trace regime shift. The empirical finding is that calibrated uncertainty determines whether a forecast can be used safely: a model should be judged not only by average error but also by whether its upper interval protects the PD slice when demand moves toward the quota.

The final artifact package includes the raw CSVs used for the experiments, processed hourly features, predictions, conformal intervals, tables, figures, and source code. All numerical claims in the manuscript are measured from these artifacts. The study provides a concise baseline for uncertainty-aware GPU capacity planning and a reproducible template for extending the analysis to fragmentation-aware scheduling, GPU model constraints, and live AIOps memo generation.

References

[1] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, "Large-scale cluster management at Google with Borg," in Proc. 10th European Conf. Computer Systems (EuroSys), 2015, pp. 1-17.

[2] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, Omega, and Kubernetes," Commun. ACM, vol. 59, no. 5, pp. 50-57, 2016.

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

[4] Y. Cheng, Z. Chai, and A. Anwar, "Characterizing co-located datacenter workloads: An Alibaba case study," in Proc. Asia-Pacific Workshop on Systems (APSys), 2018, pp. 1-7.

[5] J. Guo, Z. Chang, S. Wang, H. Ding, Y. Feng, L. Mao, and Y. Bao, "Who limits the resource efficiency of my datacenter: An analysis of Alibaba datacenter traces," in Proc. IEEE/ACM Int. Symp. Quality of Service (IWQoS), 2019, pp. 1-10.

[6] W. Xiao et al., "Gandiva: Introspective cluster scheduling for deep learning," in Proc. 13th USENIX Symp. Operating Systems Design and Implementation (OSDI), 2018, pp. 595-610.

[7] J. Gu et al., "Tiresias: A GPU cluster manager for distributed deep learning," in Proc. 16th USENIX Symp. Networked Systems Design and Implementation (NSDI), 2019, pp. 485-500.

[8] Yuanzheng Chen, Yitian Zhang, David Chau, & Matt Sherman. (2023). Credit Card Default Risk Tiering with Probability Calibration and Uncertainty-Driven Rejection: A Reproducible Study on the UCI Credit Card Clients Dataset. Journal of Advanced Computing Systems , 3(4), 31-47. https://doi.org/10.69987/JACS.2023.30403

[9] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, Time Series Analysis: Forecasting and Control, 5th ed. Hoboken, NJ, USA: Wiley, 2015.

[10] Meng-Ju Kuo, Boning Zhang, & Haozhe Wang. (2023). Tokenized Flow-Statistics Encrypted Traffic Analysis: Comparative Evaluation of 1D-CNN, BiLSTM, and Transformer on ISCX VPN-nonVPN 2016 (A1+A2, 60 s). Journal of Advanced Computing Systems , 3(8), 39-53. https://doi.org/10.69987/JACS.2023.30804

[11] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2016, pp. 785-794.

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput., vol. 9, no. 8, pp. 1735-1780, 1997.

[13] A. Vaswani et al., "Attention is all you need," in Advances in Neural Information Processing Systems, vol. 30, 2017, pp. 5998-6008.

[14] H. Zhou et al., "Informer: Beyond efficient transformer for long sequence time-series forecasting," in Proc. AAAI Conf. Artificial Intelligence, vol. 35, no. 12, 2021, pp. 11106-11115.

[15] B. Lim, S. O. Arik, N. Loeff, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," Int. J. Forecasting, vol. 37, no. 4, pp. 1748-1764, 2021.

[16] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting," in Proc. Int. Conf. Learning Representations (ICLR), 2020.

[17] V. Vovk, A. Gammerman, and G. Shafer, Algorithmic Learning in a Random World. New York, NY, USA: Springer, 2005.

[18] G. Shafer and V. Vovk, "A tutorial on conformal prediction," J. Mach. Learn. Res., vol. 9, pp. 371-421, 2008.

[19] J. Lei, M. G'Sell, A. Rinaldo, R. J. Tibshirani, and L. Wasserman, "Distribution-free predictive inference for regression," J. Amer. Stat. Assoc., vol. 113, no. 523, pp. 1094-1111, 2018.

[20] Y. Romano, E. Patterson, and E. Candes, "Conformalized quantile regression," in Advances in Neural Information Processing Systems, vol. 32, 2019, pp. 3543-3553.

[21] C. Delimitrou and C. Kozyrakis, "Paragon: QoS-aware scheduling for heterogeneous datacenters," in Proc. 18th Int. Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS), 2013, pp. 77-88.

[22] C. Delimitrou and C. Kozyrakis, "Quasar: Resource-efficient and QoS-aware cluster management," in Proc. 19th Int. Conf. Architectural Support for Programming

Languages and Operating Systems (ASPLOS), 2014, pp. 127-144.

[23] Siming Zhao, Hailin Zhou, & Daniel Martinez. (2023). LLM-Assisted Causal Attribution of Service Performance Upgrades on Churn and Tenure: Full Evaluation on the IBM Telco Customer Churn Dataset. Journal of Advanced Computing Systems , 3(2), 18-34. https://doi.org/10.69987/JACS.2023.30202

[24] M. Abadi et al., "TensorFlow: A system for large-scale machine learning," in Proc. 12th USENIX Symp. Operating Systems Design and Implementation (OSDI), 2016, pp. 265-283.