Robust Intelligent Computing for HPC Capacity Management: A Prediction-to-Operations Framework under Distribution Shift

Hongyi Zhou[1][0009-0001-7392-1498](✉) and Shouqiang Liu[2][0000-0002-2795-0685](✉)

[1] Aberdeen Institute of Data Science and Artificial Intelligence, South China Normal University, Foshan, 528225, Guangdong, China.
[2] School of Artificial Intelligence, South China Normal University, Nanhai, China.
h.zhou.23@abdn.ac.uk; liusq@m.scnu.edu.cn

Abstract. Accurate runtime information is important for high-performance computing (HPC) capacity management, but user-provided walltime limits are often unreliable and prediction errors have asymmetric operational costs. This paper presents a compact prediction-to-operations framework that links submit-time runtime prediction, probability recalibration, and discrete-event simulation (DES) under cross-trace distribution shift. Using public Parallel Workloads Archive traces, we train gradient-boosted models on deployable submit-time metadata, evaluate them with chronological internal testing and strict external validation, and apply post-hoc recalibration for operational reliability. The classifier reaches AUC 0.817 on the internal test split and 0.863 on an external trace. Platt scaling further improves held-out external calibration, reducing ECE to 0.066. The calibrated predictions are then connected to a DES capacity policy. In a stylized single-resource DES scenario with C = 8 abstract capacity slots, the prediction-driven policy reduces simulated overflow probability from 0.081 to 0.065, while increasing elective deferrals by 447 jobs over 90 simulated days. The framework makes the safety–throughput trade-off explicit and provides a reproducible path from prediction quality to capacity-management KPIs.

Keywords: Intelligent computing, machine learning, HPC workload traces, job runtime prediction, external validation, uncertainty quantification, distribution shift, capacity management

1 Introduction

High-performance computing (HPC) systems support scientific and industrial workloads whose resource demands vary sharply across users, applications, and time. Schedulers and capacity planners therefore depend on job runtime information, yet production systems often receive walltime limits that are inaccurate or strategically padded [1,2]. Learning-based runtime prediction can help, but a deployable model must be evaluated

within an operational decision process, with accuracy treated as only one part of the assessment.

Existing work has improved runtime prediction and studied how estimates interact with scheduling primitives such as backfilling [1,2,3,4,5]. However, practical deployment raises four coupled issues: public traces differ substantially across systems, probabilities may be miscalibrated under shift, under-estimation and over-estimation have different costs, and operators ultimately care about decision-level KPIs such as overflow, deferrals, occupancy, and idle capacity. These issues motivate a prediction-to-operations view that connects external validation, recalibration, and simulation-based policy evaluation in one reproducible chain.

We study this setting with public Parallel Workloads Archive (PWA) traces. The development trace and external trace differ markedly in runtime and processor distributions, so external validation is not a cosmetic check. Our framework uses submit-time features, validates predictors chronologically and across traces, recalibrates probabilities on held-out external data, and feeds calibrated predictions into a DES capacity policy.

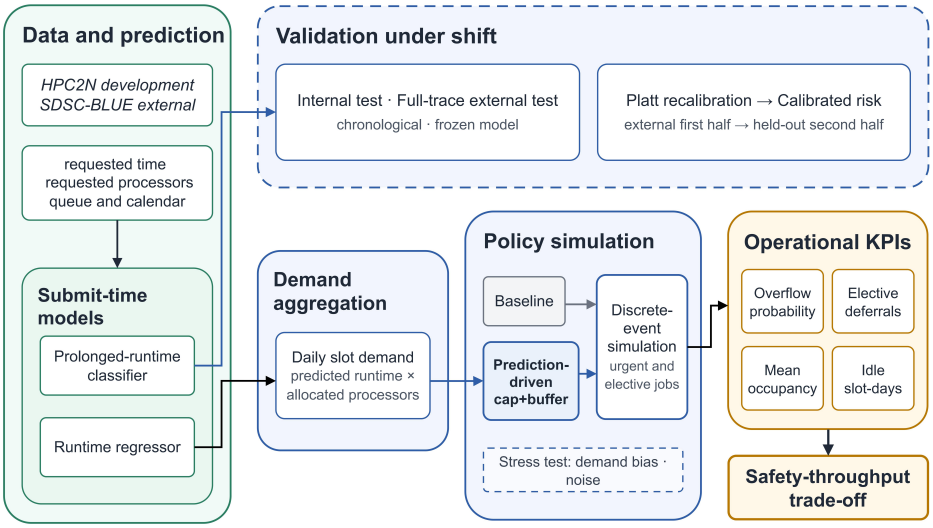

Fig. 1. Schematic overview of the prediction-to-operations framework under distribution shift. Public workload traces and submit-time features produce runtime estimates and calibrated risk scores, which are aggregated into daily demand and evaluated through DES-based capacity KPIs and reliability checks.

Our contributions can be summarized as follows:

- We formulate HPC runtime prediction as a prediction-to-operations problem under cross-trace distribution shift.
- We evaluate submit-time predictors with chronological internal testing, strict external validation, and post-hoc probability recalibration.
- We connect calibrated predictions to a DES-based capacity policy and quantify safety–throughput trade-offs and error sensitivity.

Figure 1 summarizes the prediction-to-operations translation. The rest of the paper presents related work, data and labels, methodology, experimental protocol, results, discussion, and conclusions.

2 Related Work

HPC workload traces and runtime prediction. The PWA provides a common source of workload traces in Standard Workload Format (SWF), but trace quality, missing fields, and policy changes can affect conclusions if they are not handled consistently [6]. Runtime-prediction work has shown that system-generated estimates can improve scheduling relative to raw user estimates [1,2]. Recent methods use workload-aware clustering, online classification, and cross-system features to improve deployability [3,4,5]. Surveys of job-characteristic prediction further emphasize that runtime estimates are a key input for intelligent HPC resource allocation [7].

Error asymmetry, calibration, and distribution shift. Scheduling decisions are sensitive to estimation error, and over- and under-estimation do not have symmetric consequences [8]. Learning-based schedulers also face temporal drift and trace-to-trace distribution changes [7,9]. For deployment, discrimination alone is insufficient: probability calibration determines whether risk scores can be used as reliable decision inputs [10]. We therefore report both predictive accuracy and calibration behavior before and after Platt scaling [11].

Resource-aware and robust operations. Reliability-aware platforms and uncertainty-aware scheduling studies provide complementary tools for decision-making under uncertain workloads [12]. Robust optimization and resource-allocation methods usually begin with explicit uncertainty sets or allocation objectives, whereas our work begins with empirical cross-trace predictive uncertainty and asks how calibrated prediction errors propagate into capacity KPIs. Thus, the contribution is not a new robust optimizer, but a compact evaluation chain that bridges prediction, recalibration, and operations.

3 Data and Problem Setup

We use two cleaned SWF traces from the PWA. HPC2N-2002-2.2-cln [6] is used for training, validation, and internal chronological testing. SDSC-BLUE-2000-4.2-cln [6] is used as a strict external-validation trace. Each job is represented by submit time, wait time, runtime, requested time, and processor counts when available. SWF sentinel values are treated as missing, and the cohort requires positive runtime, at least one allocated processor, parseable timestamps, and unique job identifiers.

The main classification task is prolonged-runtime risk, defined as runtime at least 3600 seconds. The regression task predicts runtime in hours through a log-runtime model. All deployable features are available at or near submission: requested time, requested processors, queue or partition identifiers, and calendar features derived from submission time. Features such as user identity, application name, wait time, and post-

start telemetry are excluded because they are not consistently available across traces or are not known at submission.

For the operations layer, jobs are assigned urgent or elective status by a fixed rule. Jobs with requested time at most 3600 seconds are urgent; when requested time is missing, jobs below the training-trace 25th percentile runtime are treated as urgent. The rule is fixed on the training trace and then reused on the external trace. Table 1 shows the resulting cohort and confirms substantial trace-to-trace shift.

Table 1. Cohort and shift summary for the development and external traces.

Metric	Development	External
Jobs, N	202,870	223,637
Runtime median (h)	0.798	0.064
Runtime P75/P90 (h)	4.769 / 10.528	0.507 / 2.448
Wait median (h)	0.444	0.023
Allocated procs median/P75	2 / 8	8 / 32
Requested-time missing (%)	0.0	0.0
Urgent jobs (%)	34.9	51.8
Prolonged label prevalence (%)	43.3	17.9
KS: runtime / procs / wait	0.313 / 0.747 / 0.202	-

The large differences in runtime, parallelism, and label prevalence motivate treating the external trace as a genuine shift test, not merely as a second random split.

4 Methodology

4.1 Submit-Time Prediction and Calibration

Numerical features are median-imputed, categorical features are one-hot encoded with unknown handling, and the preprocessing pipeline is serialized with each fitted model.

For prolonged-runtime classification, we compare logistic regression with gradient-boosted decision trees implemented with LightGBM [13]. For runtime regression, we compare ridge regression and a gradient-boosted regressor trained on log(1+runtime), with results reported back in runtime-hours. Requested time captures user expectations about job duration, requested processors represent resource scale, queue identifiers encode system routing, and calendar features capture workload seasonality.

External validation applies the trained development-trace model directly to the external trace without retraining. Where recalibration is reported, Platt scaling is fit on the first half of the external trace by time order and evaluated on the held-out second half. This protocol avoids reporting in-sample recalibration gains while showing whether a lightweight local calibration step can improve operational reliability under shift.

4.2 Demand Aggregation and Capacity Policy

Job-level runtime predictions are aggregated into a single-resource daily demand signal expressed in abstract capacity-slot units. The trace-reported allocated processor count p_j is used to weight runtime demand, but the resulting DES capacity should be interpreted as a stylized policy-evaluation scale rather than the physical processor capacity of the source systems. For jobs in set J_t arriving on day t, predicted runtime $\hat{r}_j$, and allocated processors p_j, day-averaged demand is

$$d_t = \frac{\sum_{j \in J_t} \hat{r}_j p_j}{24} \quad (1)$$

Let C denote the scenario capacity in the same abstract slot units as d_t, and let b denote a reserved safety buffer..

The prediction-driven policy sets the elective slot cap to

$$S_t^{\text{pred}} = \max(0, C - b - \lceil d_t \rceil) \quad (2)$$

The baseline uses no additional elective restriction beyond total capacity.

This simple cap+buffer rule makes the safety, throughput trade-off explicit without fitting policy parameters after seeing the DES results.

4.3 Discrete-Event Simulation and Scalability

Each DES replication simulates arrivals over a 90-day policy-evaluation horizon built from the aggregated daily demand signal. Replications use the same demand sequence but independent stochastic DES draws. The simulator tracks total occupancy and elective occupancy. Urgent jobs are admitted whenever total capacity is available. Elective jobs are admitted only when total capacity and the prediction-driven elective cap are both satisfied; otherwise they are counted as deferred or cancelled. We record overflow probability (urgent overflow events divided by total simulated arrivals), elective deferrals, mean occupancy, and idle slot-days. The primary scenario uses C = 8 abstract capacity slots, and {8, 12, 16} is used as a predefined stylized capacity grid for policy stress testing. These capacity values define comparable DES scenarios under the same demand-construction rule. Error sensitivity is assessed by applying systematic bias and variance perturbations to prediction-derived demand.

The online portion consists only of tree-model inference and daily aggregation. For T trees of depth d, per-job inference is O(Td), while aggregation for n_t daily arrivals is $O(n_t)$. The DES is used offline for policy evaluation and is not on the scheduling critical path. This deployment assumption requires submit-time features, a transparent single-resource capacity abstraction that maps processor-weighted runtime demand into comparable slot units, and enough recent local outcomes for recalibration.

5 Experimental Protocol

Internal evaluation uses chronological train/validation/test fractions of 0.70/0.15/0.15 by job start time to mimic prospective deployment and avoid time leakage. Unrecalibrated external validation evaluates the development-trace model on the full external trace. Hyperparameters are fixed a priori, including 300 boosting iterations, learning rate 0.05, and 31 leaves for the LightGBM models. All experiments use a fixed random seed and the same preprocessing pipeline.

Classification is summarized with AUC, AUPRC, Brier score, and expected calibration error (ECE) [10]. Regression is summarized with MAE and RMSE in runtime-hours. DES KPIs are reported as means and 95% bootstrap confidence intervals across 30 replications, with bootstrap-based two-sided p-values for policy deltas. The ECE uses ten equal-width probability bins; lower values indicate closer agreement between predicted probabilities and observed frequencies. All reported tables and figures are generated from deterministic scripts and run configurations. The experiment manifest records input checksums, preprocessing settings, model hyperparameters, recalibration parameters, DES seeds, and simulation settings. This provenance record is used to keep the prediction and operations layers reproducible.

6 Results

6.1 Distribution Shift

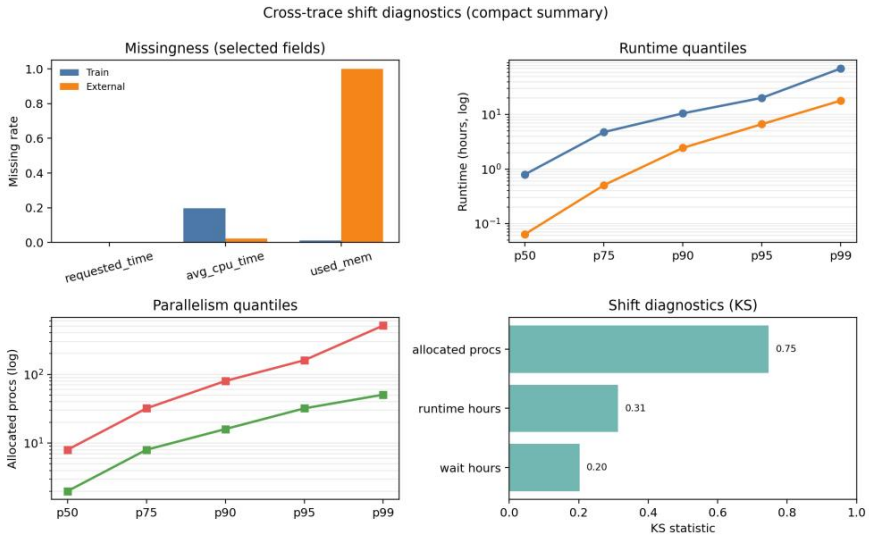

Fig. 2. Cross-trace shift diagnostics. The compact summary compares selected missingness, runtime and parallelism quantiles, and KS statistics between the development and external traces.

Table 1 and Figure 2 show that the external trace is not a random replicate of the development trace. The external jobs have shorter runtimes but larger processor allocations, and the urgent-job proportion increases from 34.9% to 51.8%. The KS statistics for runtime, allocated processors, and wait time are 0.313, 0.747, and 0.202, respectively. This supports treating external validation as a distribution-shift test, not merely as a second internal split.

6.2 Prediction and Calibration

Tables 2 and 3 and Figure 3 report the main prediction results. With submit-time features only, the gradient-boosted classifier improves over logistic regression on the internal chronological test split, reaching AUC 0.817 and AUPRC 0.668. Under unrecalibrated full-trace external validation, discrimination remains useful with AUC 0.863 and AUPRC 0.552. The higher external AUC should not be read as universal improvement under distribution shift; it indicates that the selected one-hour threshold creates a different discrimination problem on the external trace, so AUC should be interpreted jointly with AUPRC, calibration, and the shift diagnostics in Table 1.

Table 2. Prolonged-runtime classification performance. AUC, AUPRC, Brier score, and ECE summarize discrimination and probabilistic reliability.

Model	Split	AUC	AUPRC	Brier	ECE
Logistic regression	Internal	0.555	0.489	0.316	0.244
LightGBM	Internal	0.817	0.668	0.171	0.098
LightGBM	External	0.863	0.552	0.116	0.075
LightGBM + Platt	External held-out	0.863	0.552	0.113	0.066

Calibration drift remains important for operational use. After fitting Platt scaling on the first half of the external trace, the recalibrated model achieves ECE 0.066 on the held-out external half, supporting the use of local recalibration before operational deployment (Figure 4). For runtime regression, the LightGBM regressor slightly improves over ridge regression internally and achieves lower absolute error on the external trace, consistent with the shorter external runtime distribution.

Table 3. Runtime regression performance. MAE and RMSE are reported in runtime-hours.

Model	Split	MAE	RMSE
Ridge regression	Internal	8.231	18.872
LightGBM	Internal	8.001	18.309
LightGBM	External	1.110	3.378

Tables 2 and 3 separate classification and regression results: classification metrics describe ranking and probabilistic reliability, whereas regression metrics quantify runtime error in hours. Two details are operationally important. First, the external AUC remains high, but AUPRC is lower than the internal LightGBM value, so the external trace should still be read as a shifted and differently imbalanced problem. Second, Platt

scaling improves ECE and Brier score on held-out external data, which supports using recalibrated probabilities when the downstream policy consumes risk scores.

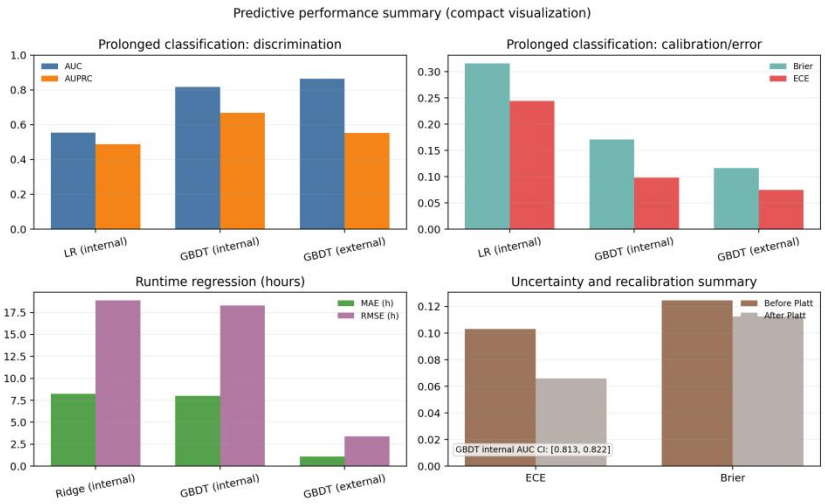

Fig. 3. Predictive performance summary. The figure compares classification discrimination, calibration/error metrics, runtime-regression error, and the effect of external Platt recalibration.

Figure 3 provides a compact visual summary of these results. It highlights that the deployable submit-time model is not only more discriminative than the linear baseline, but also usable as an input to operational control after recalibration.

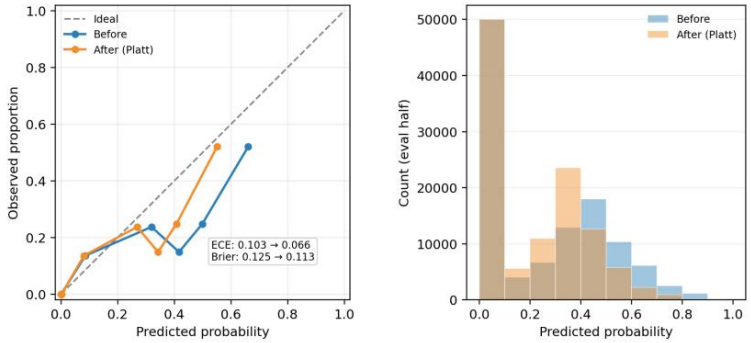

Fig. 4. External calibration before and after Platt scaling. The reliability curve and predicted-probability histogram show improved probabilistic reliability on held-out external data.

6.3 Operational Impact and Error Sensitivity

Table 4. Operational KPI comparison at capacity C = 8. Values are mean [95% bootstrap CI] across 30 DES replications.

KPI	Baseline	Prediction-driven	Δ	p
Overflow prob.	0.081 [0.068, 0.095]	0.065 [0.054, 0.077]	-0.016 [-0.033, 0.002]	0.084
Elective deferrals	8,976 [8,943, 9,012]	9,423 [9,391, 9,455]	447 [398, 493]	<0.001
Mean occupancy	6.47 [6.27, 6.65]	6.33 [6.14, 6.52]	-0.14 [-0.40, 0.14]	0.330
Idle slot-days	137.9 [121.4, 156.4]	150.0 [134.2, 168.1]	12.1 [-11.7, 36.7]	0.328

Table 4 compares the static baseline with the prediction-driven cap+buffer policy in the stylized DES scenario with C = 8 abstract capacity slots. The prediction-driven policy reduces mean overflow probability from 0.081 to 0.065, a relative reduction of 19.8%. The confidence interval for the delta is mostly negative but includes zero, indicating a directionally favorable safety gain without statistically definitive evidence. The same policy increases elective deferrals by 447 jobs over 90 days, which is statistically reliable and represents the throughput cost of the safety buffer. This pattern is the expected behavior of a conservative capacity policy: it buys safety by reserving capacity for urgent or uncertain demand. The magnitude of the trade-off is therefore more informative than a single significance test.

The results in Table 4 and Figure 5 make the operational trade-off explicit: reducing overflow risk requires reserving capacity and therefore deferring more elective work. Under the tested sensitivity settings, systematic demand bias mainly changes elective deferrals, with limited effect on overflow probability. Over-prediction makes the policy more conservative, while under-prediction admits more elective work and reduces the safety margin. Thus, downstream value depends on the relative cost of overflow and deferrals, not on prediction accuracy alone. If overflow events are operationally much more costly than elective deferrals, the prediction-driven policy can be justified even when the overflow reduction is modest; otherwise, the baseline may be preferable. Accordingly, the framework reports both predictive metrics and capacity KPIs.

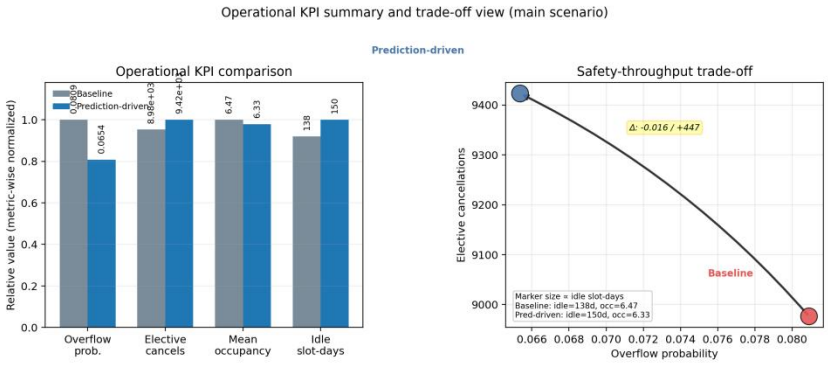

Fig. 5. Operational KPI summary and trade-off view at capacity C = 8. The prediction-driven policy lowers overflow probability while increasing elective deferrals and idle slot-days.

7 Discussion and Limitations

The main lesson is that runtime predictors should be evaluated through the decisions they support. External validation and recalibration expose reliability issues that accuracy metrics alone would hide. The DES then translates those issues into capacity KPIs. This framing is compatible with robust optimization and resource allocation: those methods can optimize decisions once uncertainty is represented, while our pipeline measures and calibrates prediction uncertainty before it enters the decision layer.

The framework is scalable under modest assumptions. Submit-time features must be available before scheduling, local outcomes must be collected for recalibration, and capacity must be abstracted into comparable slots for the DES policy. The online computation is lightweight because inference is tree evaluation and aggregation is linear in daily arrivals. The DES can be run offline during policy design or periodically during recalibration, so it does not delay individual job admission.

Several limitations remain. The DES simplifies heterogeneous computing resources into a stylized single-resource capacity abstraction. This limits its ability to reproduce the exact physical capacities of the source clusters, but allows the capacity grid to function as a controlled policy stress test for examining how scheduling outcomes respond to different levels of resource tightness. External validation uses one external trace, so broader multi-trace evaluation is needed before claiming universal transferability. The arrival process is simplified, and richer time-varying or bursty processes may change overflow estimates. The feature set is deliberately limited to submit-time metadata, which improves deployability but leaves out richer telemetry. Future work should extend the simulator to multi-resource capacity, evaluate more traces, study online recalibration, and examine fairness across user groups and job classes.

8 Conclusion

This paper presented a compact prediction-to-operations framework for HPC capacity management under distribution shift. Using public PWA traces, we trained submit-time runtime predictors, evaluated them under chronological internal testing and strict external validation, and applied post-hoc recalibration to improve probabilistic reliability. The calibrated predictions were then connected to a DES-based capacity policy so that safety-throughput trade-offs could be measured directly.

The results show that submit-time gradient-boosted models can retain useful discrimination under cross-trace shift, while recalibration improves reliability for operational use. In the DES study, the prediction-driven policy reduced overflow probability at the cost of additional elective deferrals, making the operational trade-off explicit. Future work should extend the simulator to multi-resource constraints, evaluate more traces, and study online recalibration in production-like settings.

9 References

1. Tsafrir, D., Etsion, Y., Feitelson, D.G.: Backfilling using system-generated predictions rather than user runtime estimates. IEEE Transactions on Parallel and Distributed Systems 18(6), 789–803 (2007) https://doi.org/10.1109/TPDS.2007.70606.
2. Tang, W., Desai, N., Buettner, D., Lan, Z.: Job scheduling with adjusted runtime estimates on production supercomputers. Journal of Parallel and Distributed Computing 73(7), 926–938 (2013) https://doi.org/10.1016/j.jpdc.2013.02.006.
3. Park, J.-W., Kim, E.: Runtime prediction of parallel applications with workload-aware clustering. The Journal of Supercomputing 73(11), 4635–4651 (2017) https://doi.org/10.1007/s11227-017-2038-2.
4. Zrigui, S., de Camargo, R.Y., Legrand, A., Trystram, D.: Improving the performance of batch schedulers using online job runtime classification. Journal of Parallel and Distributed Computing 164, 83–95 (2022) https://doi.org/10.1016/j.jpdc.2022.01.003.
5. Yang, W., Liao, X., Dong, D., Yu, J.: Exploring job running path to predict runtime on multiple production supercomputers. Journal of Parallel and Distributed Computing 175, 109–120 (2023) https://doi.org/10.1016/j.jpdc.2023.01.001.
6. Feitelson, D.G., Tsafrir, D., Krakov, D.: Experience with using the Parallel Workloads Archive. Journal of Parallel and Distributed Computing 74(10), 2967-2982 (2014) https://doi.org/10.1016/j.jpdc.2014.06.013.
7. Hou, Z., Shen, H., Zhou, X., Gu, J., Wang, Y., Zhao, T.: Prediction of job characteristics for intelligent resource allocation in HPC systems: a survey and future directions. Frontiers of Computer Science 16(5), 165107 (2022) https://doi.org/10.1007/s11704-022-0625-8.
8. Dell'Amico, M., Carra, D., Pastorelli, M., Michiardi, P.: Revisiting size-based scheduling with estimated job sizes. In: 2014 IEEE 22nd International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 411 – 420. IEEE (2014). https://doi.org/10.1109/MASCOTS.2014.57.
9. Jajoo, A., Hu, Y.C., Lin, X., Deng, N.: A case for task sampling based learning for cluster job scheduling. In: 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22), pp. 19 – 33. USENIX Association, Renton, WA (2022). https://www.usenix.org/conference/nsdi22/presentation/jajoo.
10. Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On calibration of modern neural networks. In: Proceedings of the 34th International Conference on Machine Learning (ICML). Proceedings of Machine Learning Research, vol. 70, pp. 1321–1330. PMLR, Sydney, NSW, Australia (2017). http://proceedings.mlr.press/v70/guo17a.html.
11. Platt, J.C.: Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In: Smola, A.J., Bartlett, P., Schölkopf, B., Schuurmans, D. (eds.) Advances in Large Margin Classifiers, pp. 61–74. MIT Press, Cambridge, MA, USA (2000).
12. Naghshnejad, M., Singhal, M.: A hybrid scheduling platform: a runtime prediction reliability aware scheduling platform to improve HPC scheduling performance. The Journal of Supercomputing 76(1), 122–149 (2020) https://doi.org/10.1007/s11227-019-03004-3.
13. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.: LightGBM: A highly efficient gradient boosting decision tree. In: Advances in Neural Information Processing Systems 30 (NeurIPS), pp. 3146–3154 (2017).