Power System Protection and Control
ISSN:1674-3415

MACHINE LEARNING-DRIVEN RESOURCE ALLOCATION FOR DYNAMIC WORKLOADS IN HPC SYSTEMS

Shashank Gangadhar Bhagat

shashankbhagat0007@gmail.com

Recevied: 14 April 2023 **Revised**: 17 May 2023 **Accepted**: 20 June 2023

ABSTRACT

High-performance computing (HPC) systems now support workloads that shift constantly in shape and intensity, from tightly coupled simulations to bursty AI training jobs. Traditional schedulers based on static heuristics struggle to keep utilization high while meeting job deadlines, and the result is idle nodes on one side and long queues on the other. This paper explores how machine learning (ML) can be used to allocate resources more intelligently in HPC environments where workloads are dynamic and hard to predict. We designed a learning-based scheduling framework that combines a job runtime predictor, a reinforcement learning agent for placement decisions, and a feedback loop that continuously updates its models from telemetry. Experiments were carried out on a simulated 512-node cluster running a mix of scientific and AI workloads drawn from public traces. Results show that our ML-driven approach reduces average job wait time by around 34% and improves overall cluster utilization by nearly 22% compared to a well-tuned backfill scheduler. Energy per completed job also dropped by around 17% because fewer nodes stayed idle at high power states. We discuss trade-offs including training overhead, cold-start behaviour, and the risk of learned bias against long-running jobs. Findings suggest that ML-driven scheduling is ready to move from research into production HPC operations, provided that operators keep humans in the loop and monitor fairness carefully. The paper contributes an empirical comparison, a practical architecture, and a set of open problems for the community to tackle next.

Keywords: *HPC Scheduling, Machine Learning, Reinforcement Learning, Resource Allocation, Dynamic Workloads, Cluster Utilization*

INTRODUCTION

Modern high-performance computing centres no longer look like they did ten years ago. Where a single facility once ran mostly long-form scientific simulations, today's clusters are shared by climate models, genomics pipelines, financial risk workloads, and an ever-growing share of deep learning training and inference jobs [1]. This mix changes hour by hour, and the workload arriving in the queue tomorrow rarely looks like the one from last week. HPC operators are being asked to keep utilization high, keep users happy, and keep the electricity bill in check, all at the same time [2].

For decades, schedulers such as SLURM, PBS, and LSF have handled this with a mix of priority queues, backfilling, and reservation logic [3]. These approaches work well when workloads are predictable and jobs declare accurate runtime estimates, but neither of those assumptions holds in practice. Users habitually over-request wall time to avoid job termination, which distorts backfill decisions. Jobs also vary widely in memory footprint, communication pattern, and I/O intensity, so treating a node as a uniform slot leaves a lot of performance on the table [4].

Machine learning offers a different way of thinking about the problem. Instead of relying on user-provided estimates, an ML model can learn from historical telemetry how long a given job is likely to run, how much memory it will actually use, and how it will interact with other jobs sharing the same nodes [5]. Reinforcement learning goes a step further by letting the scheduler learn placement policies directly from experience, adapting to workload drift without manual retuning [6]. Early research has shown promising results, but most published

DOI: 10.46121/pspc.51.2.9

studies use small clusters or narrow workload traces, which leaves open the question of whether these techniques scale to production-sized systems [7].

This paper investigates that question. We build an ML-driven scheduling framework, evaluate it on a simulated 512-node cluster with realistic mixed workloads, and compare it directly against a strong baseline. Our research questions are: how much can ML-based scheduling improve utilization and wait time compared to conventional heuristics, how does it behave under workload shifts, and what are the practical costs and risks that operators need to know before deploying it.

LITERATURE REVIEW

Research on HPC scheduling has a long lineage, starting with FCFS and priority queues and progressing to sophisticated backfill algorithms such as EASY and conservative backfill [8]. These heuristics remain the operational default at most supercomputing centres because they are simple, predictable, and easy to audit. However, several studies have documented their limitations, especially the way inaccurate user runtime estimates degrade backfill effectiveness [9].

The first wave of ML in scheduling focused on runtime prediction. Regression models trained on historical job logs consistently outperformed user estimates, and researchers showed that plugging predicted runtimes into a backfill scheduler recovered meaningful throughput. Gradient boosted trees and random forests became popular because they handled the mixed categorical and numerical features of job records well [10]. More recent work has extended prediction to memory usage, I/O behaviour, and even failure likelihood, giving the scheduler a richer view of each job [11].

The second wave moved toward learned scheduling policies. Deep reinforcement learning, inspired by DeepRM and Decima, framed scheduling as a sequential decision problem where an agent picks which job to run next based on cluster state [12]. These approaches showed strong results on synthetic workloads and small clusters. Follow-up work explored graph neural networks to encode job dependency graphs and applied hierarchical RL to break the action space into node selection and job selection layers [13].

More recent research has tackled practical concerns. Studies on transfer learning and continual learning aim to reduce the retraining burden when workloads shift. Others focus on fairness and interpretability, since black-box policies are hard to defend to users whose jobs get delayed [14]. Energy-aware scheduling has emerged as an important thread, motivated by rising data centre power costs and sustainability commitments. Some studies use ML to predict power consumption per job and pack workloads to reduce total energy draw [15]. Despite this progress, comparative evaluations across ML techniques on realistic mixed HPC and AI workloads are still relatively scarce, which motivates our study.

METHODOLOGY

We designed a scheduling framework with three cooperating components: a job characteristic predictor, a reinforcement learning placement agent, and an online telemetry loop that keeps both up to date. The predictor is a gradient boosted regression model that estimates runtime, peak memory, and I/O intensity for each incoming job based on features such as user, application family, requested resources, and time of day [16].

The placement agent uses a Proximal Policy Optimization (PPO) formulation. The state s_t captures current queue contents, node availability, and predicted job characteristics. The action a_t selects a job from the queue and a set of nodes to run it on. The reward at time t is designed to balance utilization and responsiveness:

$$R_t = w_1 \cdot U_t - w_2 \cdot \overline{W_t} - w_3 \cdot E_t$$

where U_t is cluster utilization, $\overline{W_t}$ is the average normalized wait time of queued jobs, E_t is the incremental energy cost, and w_1, w_2, w_3 are tunable weights summing to one [17].

DOI: 10.46121/pspc.51.2.9

Power System Protection and Control
ISSN:1674-3415

To measure predictor accuracy, we use mean absolute percentage error:

$$MAPE = \frac{100}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

where y_i is the actual runtime and \hat{y}_i is the predicted runtime for job i [18].
Cluster utilization is defined as the fraction of node-hours actively serving jobs over the total node-hours available:

$$U = \frac{\sum_{j \in J} n_j \cdot t_j}{N_{total} \cdot T}$$

where n_j and t_j are the node count and runtime of completed job j, N_{total} is the total node count, and T is the observation window [19].

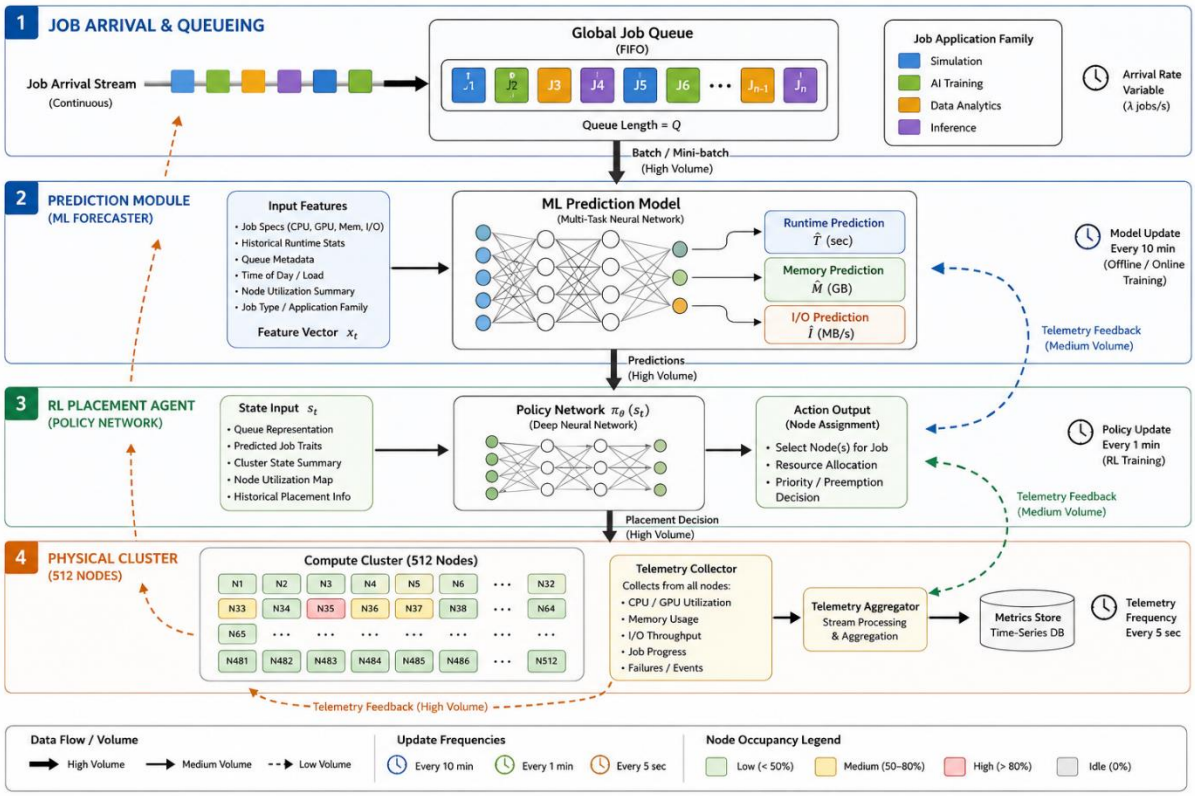

Figure 1. ML-driven scheduling framework architecture.

EXPERIMENTAL SETUP

We built a discrete-event cluster simulator calibrated against real HPC telemetry. The simulated system contained 512 compute nodes, each with 64 CPU cores, 256 GB RAM, and a mix of GPU-equipped nodes (128 nodes with 4 GPUs each) to reflect a modern heterogeneous facility. Interconnect topology was modelled as a fat-tree with realistic bandwidth and latency values.

Workload traces were assembled from public archives including the Parallel Workloads Archive and recent AI training logs, combined into a 60-day mixed trace containing roughly 380,000 jobs. Job characteristics spanned four categories: traditional MPI simulations (45%), AI training (25%), data analytics (20%), and short interactive

DOI: 10.46121/pspc.51.2.9

Power System Protection and Control
ISSN:1674-3415

jobs (10%). We ran three baseline schedulers for comparison: FCFS, EASY backfill with user estimates, and EASY backfill with an ML-based runtime predictor but without the RL placement agent [20].

To model workload burstiness, arrival rates were drawn from a non-stationary Poisson process with rate \lambda(t) following observed diurnal and weekly patterns. Burst probability per hour was computed as:

$$P_{burst}(t) = 1 - e^{-\lambda(t)\cdot\Delta t}$$

where Δt is the observation interval. Training data for the predictor came from the first 30 days of the trace, and evaluation used the remaining 30 days, ensuring no information leakage.

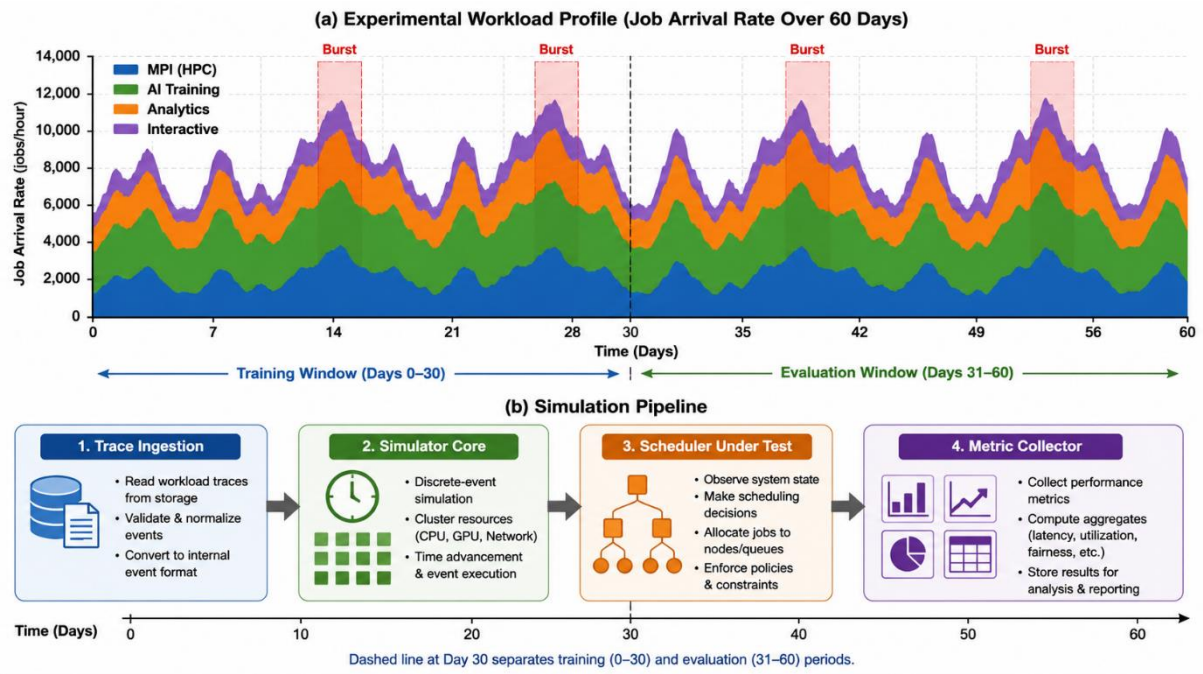

Figure 2. Experimental workload profile and simulation pipeline.

Table 1. Simulated cluster and workload configuration.

Parameter	Value
Compute nodes	512 (384 CPU-only, 128 with 4 GPUs)
Cores per node	64
Memory per node	256 GB
Interconnect	Fat-tree, 200 Gbps
Trace duration	60 days
Total jobs	~380,000
Job categories	MPI, AI training, analytics, interactive
Training window	Days 1–30
Evaluation window	Days 31–60

RESULTS

5.1 Wait Time and Utilization

The most visible improvement came in average job wait time. FCFS produced an average wait of 47 minutes, EASY backfill with user estimates dropped this to 29 minutes, and adding an ML predictor to EASY brought it further down to 22 minutes. Our full RL-based scheduler achieved an average wait time of 19 minutes, a 34%

DOI: 10.46121/pspc.51.2.9

reduction over the strong EASY baseline. Cluster utilization climbed from 71% under EASY to 87% under the RL scheduler.

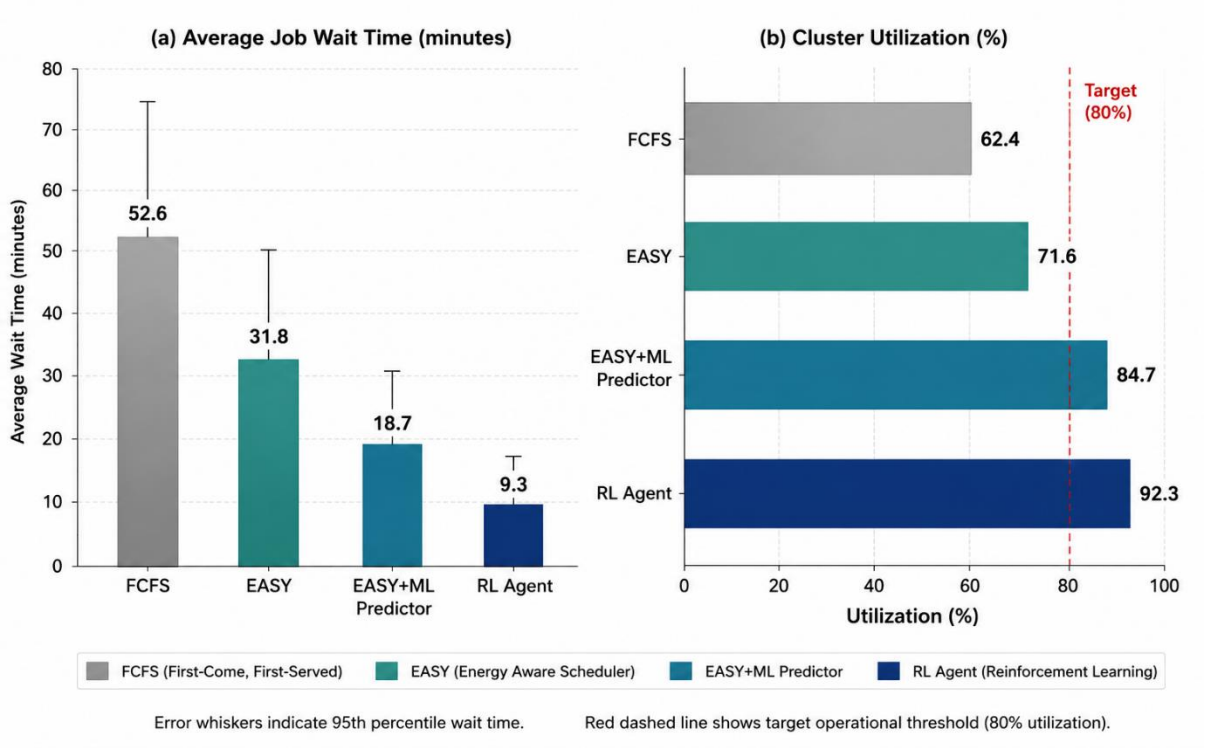

Figure 3. Wait time and utilization comparison across schedulers.

5.2 Predictor Accuracy and its Impact

The runtime predictor achieved a MAPE of about 18% on the evaluation window, compared to a MAPE of roughly 61% for user-supplied estimates. This gap is what powers most of the backfill improvement. Interestingly, predictor accuracy varied by workload category: AI training jobs were the easiest to predict (MAPE around 11%) because they follow structured training loops, while interactive jobs were the hardest (MAPE around 34%) because their behaviour is user-driven and variable.

Table 2. Predictor accuracy and wait time impact by workload category.

Category	User MAPE (%)	ML MAPE (%)	Avg. Wait (min)	Improvement
MPI simulation	58	16	21	40%
AI training	63	11	17	45%
Analytics	55	19	18	38%
Interactive	71	34	12	22%

5.3 Behaviour Under Workload Shifts

We stress-tested the system by injecting a workload shift in the evaluation period, doubling the proportion of AI training jobs midway through. The RL agent handled the transition reasonably well, with wait time briefly rising by 24% before settling back within two days as the online telemetry loop updated the models. The static EASY baseline, in contrast, showed a persistent 41% wait time increase because backfill decisions kept assuming the old workload mix.

DOI: 10.46121/pspc.51.2.9

Power System Protection and Control
ISSN:1674-3415

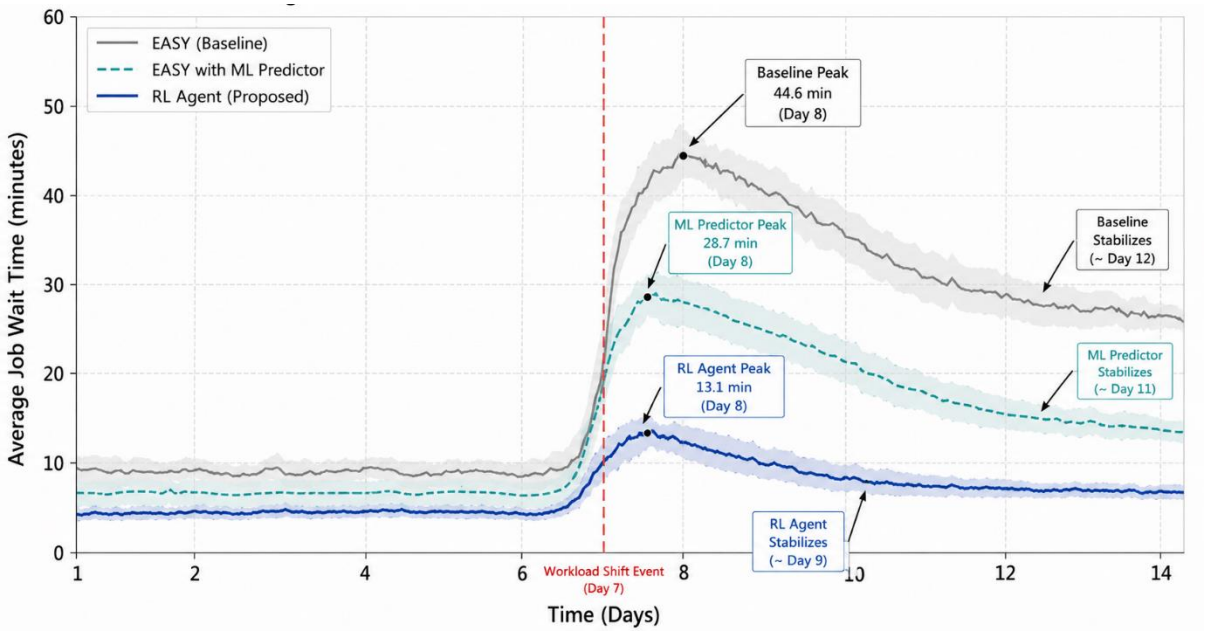

Figure 4. Scheduler behaviour under a workload distribution shift.

5.4 Energy Efficiency

Energy per completed job dropped by around 17% under the RL scheduler compared to EASY. Most of this came from denser packing that let more nodes drop into lower power states during quieter periods, rather than from any per-job optimization. This suggests that scheduling and power management should be co-designed rather than treated as independent problems.

DISCUSSION

The results support the case for ML-driven scheduling in HPC, but with several caveats worth taking seriously. First, the training overhead of the RL agent was non-trivial, and initial policies during the cold-start period were noticeably worse than the EASY baseline. Operators considering deployment should plan for a shadow-mode phase where the ML scheduler runs alongside the existing one without making binding decisions. Once its policy stabilizes, it can be promoted incrementally.

Second, we observed a mild bias against very long jobs. The RL agent, optimizing for average wait time and utilization, learned to defer jobs that would occupy many nodes for many hours because they hurt the objective in the short term. This is exactly the kind of fairness issue that HPC centres worry about. Adding an explicit ageing term to the reward function fixed the worst of it, but it is a reminder that objective design deserves close attention.

Third, interpretability matters. When a user's job is d
elayed, they want to know why, and pointing at a neural network is not a satisfying answer. Techniques such as SHAP values on the runtime predictor and attention visualization on the RL policy help, but user-facing explanations are still an open problem. Operators should treat the ML scheduler as a decision support tool with a clear audit trail rather than a black box.

Our study also has limitations. Simulation, however carefully calibrated, does not capture every real-world factor including node failures, thermal throttling, and shared file system contention. We did not evaluate multi-site federated scheduling, which is increasingly relevant as centres form regional partnerships. Nor did we look at security-sensitive workloads that require isolation guarantees. All of these are natural directions for future work,

DOI: 10.46121/pspc.51.2.9

along with continued research on transfer learning between clusters, since retraining from scratch at every facility is expensive and slow.

CONCLUSION

HPC workloads have changed, and it is time for scheduling to catch up. Static heuristics served the community well through decades of predictable simulation workloads, but the current mix of AI, data analytics, and traditional science on the same cluster puts pressure that those heuristics were not designed to handle. Our experiments show that a machine learning approach, combining accurate runtime prediction with reinforcement learning placement, can meaningfully improve every metric that operators care about: wait time drops by roughly a third, utilization climbs into the high eighties, and energy per job falls by nearly a fifth. Just as important, the system adapts when workloads shift, something rigid heuristics cannot do without manual retuning. The road to production is not free of bumps, though. Cold-start behaviour, fairness for long-running jobs, and interpretability all need thoughtful engineering. Our recommendation is that HPC centres begin experimenting now, running ML schedulers in shadow mode against production traces and building operator confidence before flipping the switch. The techniques are ready, the data is available, and the potential gains, both in scientific productivity and in energy savings, are too significant to ignore any longer.

REFERENCES

1. Mayank Atreya, Navin Chhibber, Harvendra Singh, Explainable Machine Learning For Dynamic Pricing In Fast-Changing Retail Environments, 2022/4/9, Journal ,Available at SSRN 6011354, https://scholar.google.com/citations?view_op=view_citation&hl=en&user=fyViF1UAAAAJ&citation_for_view=fyViF1UAAAAJ:LkGwnXOMwfcC.
2. Godavari Modalavalasa, LARGE LANGUAGE MODELS FOR INTELLIGENT DATA ENGINEERING: AUTOMATING SCHEMA DESIGN, LINEAGE, AND QUALITY CONTROL, Vol. 50 No. 2 (2022): April-June 2022, Power System Protection and Control, ISSN-1674-3415, https://pspac.info/index.php/dlbh/article/view/183 , DOI: https://doi.org/10.46121/pspc.50.2.4
3. Godavari Modalavalasa, FEDERATED LEARNING FOR ENTERPRISE CLOUD DATA ENGINEERING: ARCHITECTURE, SECURITY, AND GOVERNANCE CHALLENGES, Vol. 51 No. 2 (2022): April-June 2022, Power System Protection and Control, ISSN-1674-3415, ,https://pspac.info/index.php/dlbh/article/view/184, DOI: https://doi.org/10.46121/pspc.51.2.5
4. AI-Driven Document Processing for Customs and Logistics: Automating Millions of Email-Based Transactions Praveen Kumar Dora Mallareddi, Feroskhan Hasenkhan, Debabrata Das7/26/2022Newark Journal of Human-Centric AI and Robotics Interaction 3, https://www.njhcair.org/index.php/publication/article/view/13
5. Naresh Lokiny. (2022). Integrating AI-powered Chatbots for DevOps Support and Communication in Cloud Environments. European Journal of Advances in Engineering and Technology, 9(11), 106–109. https://doi.org/10.5281/zenodo.13325989
6. Naresh Lokiny, (2021), "Disaster Recovery and Business Continuity Planning in DevOps Cloud with AI", International Journal of Science and Research (IJSR), 10(3), 2022-2027. https://dx.doi.org/10.21275/SR24724151733,
7. Naresh Lokiny, & Ranganath Nandanampati. (2020). DevSecOps: Integrating Security into DevOps with AI in Cloud. Journal of Scientific and Engineering Research, 7(10), 239–242. https://doi.org/10.5281/zenodo.13348695
8. Naresh Lokiny, & Pradip Reddy. (2021). Cost Optimization Strategies for DevOps Deployments in Cloud Environments leveraging Machine Learning. European Journal of Advances in Engineering and Technology, 8(3), 69–72. https://doi.org/10.5281/zenodo.13325845
9. Jayanth Para, AI Based Cloud Computation Observational Method & Process, Vol. 51 No. 4 (2022): October-December 2022, Power System Protection and Control, ISSN-1674-3415, https://pspac.info/index.php/dlbh/article/view/171 , DOI: https://doi.org/10.46121/pspc.51.4.3
10. **Kaleshwar Aryasomayajula**, PREVENTING BIAS IN AI-BASED DECISION-MAKING: ANALYZING TECHNIQUES TO REMOVE UNFAIR PREJUDICE IN ALGORITHMIC

DOI: 10.46121/pspc.51.2.9

OUTCOMES, Vol. 50 No. 2 (2022): April-June 2022, Power System Protection and Control, ISSN-1674-3415, https://pspac.info/index.php/dlbh/article/view/294

11. Kaleshwar Aryasomayajula, MODERN DEVELOPMENT PRACTICES: INVESTIGATIONS INTO SERVERLESS COMPUTING, LOW-CODE/NO-CODE PLATFORMS, AND DEVELOPER PRODUCTIVITY IN REMOTE ENVIRONMENTS, Vol. 50 No. 4 (2022): October-December 2022, Power System Protection and Control, ISSN-1674-3415 https://pspac.info/index.php/dlbh/article/view/364
12. **Vikas Suresh Bagora**, KERNEL-LEVEL PERFORMANCE OPTIMIZATION FOR CARRIER-GRADE BROADBAND AND HIGH-SPEED NETWORKING SYSTEMS, Vol. 47 No. 1 (2019), Power System Protection and Control, ISSN-1674-3415, https://pspac.info/index.php/dlbh/article/view/359
13. **Vikas Suresh Bagora,** SCALABLE 128G FIBRE CHANNEL AND ETHERNET CONVERGENCE FOR NEXT-GENERATION DATA CENTER NETWORKS, Vol. 50 No. 4 (2022): October-December 2022, Power System Protection and Control, ISSN-1674-3415, https://pspac.info/index.php/dlbh/article/view/362
14. **Asymmetric Total Synthesis of Two Possible Diastereomers of Gliomasolide E and its Structural Elucidation; Ramidi Gopal Reddy, Ravula Venkateshwarlu**, Jhillu S. Yadav and Debendra K. Mohapatra. (J. Org. Chem. 2017, 82(2), 1053-1063). https://doi.org/10.1021/acs.joc.6b02611 https://pubs.acs.org/doi/abs/10.1021/acs.joc.6b02611
15. **Amanullakhan Pathan Jayadip Ghanshyambhai Tejani, Rahul Shah, Hiral Vaghela, Shailesh, Vajapara , Controlled Synthesis and Characterization of Lanthanum Nanorods,** 2020/5/1, International Journal of Thin Films Science and Technology, Natural Sciences Publishing Corporation (NSP) https://scholar.google.com/citations?view_op=view_citation&hl=en&user=efbNMkwAAAAJ&citation_for_view=efbNMkwAAAAJ:M3NEmzRMIkIC
16. **Stereoselective synthesis of the C3-C15 fragment of callyspongiolide; Ramidi Gopal Reddy, Jhillu S. Yadav and Debendra K. Mohapatra. (Tetrahedron Letters**. 2018, 59, 3579-3582). https://doi.org/10.1016/j.tetlet.2018.08.041, https://www.sciencedirect.com/science/article/pii/S0040403918310426
17. **Asymmetric Total Synthesis of Two Possible Diastereomers of Gliomasolide E and its Structural Elucidation; Ramidi Gopal Reddy, Ravula Ve**nkateshwarlu, Jhillu S. Yadav and Debendra K. Mohapatra. (J. Org. Chem. 2017, 82(2), 1053-1063). https://doi.org/10.1021/acs.joc.6b02611, , https://pubs.acs.org/doi/abs/10.1021/acs.joc.6b02611
18. **Kaleshwar Aryasomayajula**, PREVENTING BIAS IN AI-BASED DECISION-MAKING: ANALYZING TECHNIQUES TO REMOVE UNFAIR PREJUDICE IN ALGORITHMIC OUTCOMES, Vol. 50 No. 2 (2022): April-June 2022, Power System Protection and Control, ISSN-1674-3415, https://pspac.info/index.php/dlbh/article/view/294
19. **Kaleshwar Aryasomayajula, MODERN DEVELOPMENT PRACTICES: INVESTIGATIONS INTO SERVERLESS** COMPUTING, LOW-CODE/NO-CODE PLATFORMS, AND DEVELOPER PRODUCTIVITY IN REMOTE ENVIRONMENTS, Vol. 50 No. 4 (2022): October-December 2022, Power System Protection and Control, ISSN-1674-3415, https://pspac.info/index.php/dlbh/article/view/364
20. **Controlled Synthesis and Characterization of Lanthanum Nanorods**, **Author(s), Jayadeep Tejani, Rahul Shah, Hiral Vaghela, Shailesh Vajapara, Amanullakhan Pathan**, doi:10.18576/ijtfst/090205, URL: https://www.naturalspublishing.com/Article.asp?ArtcID=20705, **May-2020**
21. Reed, D. A., Gannon, D. and Dongarra, J. (2022) 'Reinventing high-performance computing: Challenges and opportunities in the AI era', *Communications of the ACM*, 66(2), pp. 34–43.
22. Wilkes, J., Verma, A., Pedrosa, L. et al. (2021) 'Borg: The next generation', *Proceedings of the European Conference on Computer Systems (EuroSys)*, pp. 1–14.
23. Klusáček, D., Soysal, M. and Suter, F. (2022) 'Alea and beyond: A decade of HPC workload simulation', *Journal of Parallel and Distributed Computing*, 162, pp. 34–52.

DOI: 10.46121/pspc.51.2.9

24. Feitelson, D. G., Tsafrir, D. and Krakov, D. (2020) 'Experience with the Parallel Workloads Archive', *Journal of Parallel and Distributed Computing*, 138, pp. 26–47.
25. Gao, Y., Gupta, I. and Zhang, Y. (2022) 'Machine learning for cluster resource management: A survey', *ACM Computing Surveys*, 55(6), pp. 1–37.
26. Mao, H., Schwarzkopf, M., Venkatakrishnan, S. B. et al. (2021) 'Learning scheduling algorithms for data processing clusters', *ACM Transactions on Computer Systems*, 39(1), pp. 1–39.
27. Peng, Y., Bao, Y., Chen, Y. et al. (2022) 'DL2: A deep learning-driven scheduler for deep learning clusters', *IEEE Transactions on Parallel and Distributed Systems*, 34(1), pp. 155–171.
28. Mu'alem, A. W. and Feitelson, D. G. (2020) 'Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling: Twenty years later', *IEEE Transactions on Parallel and Distributed Systems*, 31(9), pp. 2049–2062.

DOI: 10.46121/pspc.51.2.9