International Journal of Artificial Intelligence, Data Science, and Machine Learning
Grace Horizon Publication | Volume 7, Issue 1, 111-115, 2026
ISSN: 3050-9262 | https://doi.org/10.63282/3050-9262.IJAIDSML-V7I1P120

Original Article

AI-Assisted Forecasting and Capacity Planning in Multi-Tenant Environments

Nikhita Kataria
Manager, Software Engineering.

Received On: 28/12/2025 ***Revised On: 30/01/2026*** ***Accepted On: 02/02/2026*** ***Published On: 04/02/2026***

Abstract *- Capacity planning and infrastructure automation are still some of the hardest problems in modern distributed systems, with a direct impact on overall costs. However, they are often underestimated, especially as hardware keeps improving and each CPU core becomes faster over time. As infrastructure scales in size and complexity, historical statistics and operational models based on historical data fail to provide reliable forecasting metrics and cost projections. Building on prior research that examines operational ownership, infrastructure reliability, and deployment correctness, this paper investigates the application of Artificial Intelligence (AI) to capacity planning and Infrastructure as Code (IaC). We demonstrate how learning-based optimization, predictive analysis and automation can improve planning accuracy, reduce operational toil, and help achieving reliability and efficiency objectives. The paper positions AI not as a replacement for capacity management systems but presents a force multiplier to ensure the accuracy of such pieces of software is high ensuring forecasts and reality do not differ by a large amount.*

Keywords *- Artificial Intelligence, Capacity Planning, Infrastructure as Code, Site Reliability Engineering, Machine Learning, Reliability Engineering.*

1. Introduction

Modern infrastructure management has shifted over time where it no longer entails managing individual servers only, it also requires positioning of the hardware according to the services that are running in the system and is very much aware of the software parameters as well. One of the most modern orchestration engines that exemplifies this is Kubernetes where large AI platforms, compute, storage, GPUs, networking infrastructure are all pooled together with some logical constraints however still shared across organizations. In such setup, infrastructure being correct and performance is not just about if resources were created successfully, it mostly is focused on if systems is highly available, capable of performing consistently and keeps working the same as changes happen i.e. is reliable. This requirement emerges from the fact that in modern distributed systems, changes are constant with new workloads, upgrades, scaling events, failures, downtimes happening all the time and older mechanisms to predict capacity requirements with buffers calculated on the basis of organic growth are no longer applicable and any misses are slower as capacity backfill is a long supply chain process. Modern approaches to capacity require understanding of how workloads pack together, configurations of GPU's such as shared or dedicated or split, isolation strategies, requirements of confidentiality etc. With this approach to manging infrastructure via code has also evolved as it does not only look at provisioning machines and focusses on understanding cluster layout, device parameters, stacking rules and how platform evolves over time. AI helps by constantly learning from these systems, simulating behaviors based on known constraints, pro-actively take actions on predicted failures and adjust ahead of time avoiding potential service downtimes. The goal of this paper is not to replace human judgement, but to understand the context and constraints better and apply the same consistently at scale.

2. Background and Motivation

Often capacity planning is treated as a cost-optimization strategy to improve overall efficiency. However, in practice it ties closely to the reliability of services running on the actual hardware. Insufficient capacity manifests itself as latency regressions, deployment failures, on-call fatigue, delayed scale ups and recover of applications during unavoidable mass failures leading to cascading impact. Prior research on on-call ownership highlights how poor capacity planning also increases cognitive load during incidents. Infrastructure as a code has now matured from just being a provisioning mechanism to a primary control plane for allocations of hardware in the right way. Current work on infrastructure deployment workflows demonstrates that misconfigurations on hardware level are a dominant cause of outages in some cases. Static IaC definitions lack situational awareness and motivation of using AI into capacity planning stems for the fact that we want it to be environment aware and operate at scale. As infrastructure complexity grows, operational intent must be encoded in systems capable of learning, adaptation, and continuously improving.

3. AI-Driven Capacity Planning

In Kubernetes based platforms, capacity planning extends beyond just aggregation of resource utilization and

capacity planning must consider pod placement constraints which varies depending on the type of workloads and their latency requirements. Some commonly known constraints are GPU partitioning, NUMA locality, average utilization threshold, locality concerns and core scheduler behavior. Machine learning models trained on Kubernetes native metrics such as pod churn, pending pods, allocation pressure, node conditions, application classification provide significantly higher prediction accuracy that just host level utilization averages that are historically the only key parameter that was fed into capacity planning. In GPU centric deployments like enterprise AI factories forecasting mechanism help anticipate fragmentation, MIG saturation, cross tenant contention as well. This further exemplifies that capacity planning aligns very closely with scheduler realities rather than theoretical mathematics. Operational failures in Kubernetes environments often get cascaded due to control plane failures such as scheduling backlogs, admission rejections, or degraded daemon sets which require special capacity planning and disaster recovery strategy than just application plane.

Learning based anomaly detection with these signals enables early intervention before a workload is impacted and hence AI driven optimizations must respect key Kubernetes primitives of disruption budgets, maintenance windows, and fault domains. Reinforcement learning approaches can learn scaling actions that minimize disruption while preserving service level objectives and reliability goals. Infrastructure as code defines cloud resources as well as cluster composition such as node pools, accelerators, CNI configurations, storage classes, device plugins and security classifications. AI assisted metadata generation helps express intent at a lower level to represent isolation boundaries, disruption requirements, upgrade strategies, security or ACL dependencies rather than low level resource counts. Large Kubernetes platforms evolve continuously, and AI models trained on prior rollouts can estimate disruption risk, helping operators stage changes through canaries and phased rollouts which a must have for ephemeral infrastructure where drift accumulation poses a major reliability risk. Learning based drift detection transforms IaC into a continuously reconciled contract between desired and actual cluster state.

3. System Architecture

Figure 1 illustrates a proposed system architecture for AI-driven capacity planning in hyperscale Kubernetes environments integrating observability, forecasting, predictive modeling, rules embedding, policy enforcement and automated IaC execution. At the lowest level, observability layer collects real-time metrics, events and logs from control plane, node pools, pods and hardware utilization. Telemetry sources include Prometheus metrics, event streams and any custom scheduler annotations specific to the deployment setup. This layer provides the necessary automation for real time understanding of cluster state, resource utilization and workload behavior. Next, feature engineering layer transforms raw telemetry into actionable signals capturing pending pods, fragmentation, node health and utilization, maintenance windows, workloads priorities that feed into machine learning models that perform predictive forecasting and possible optimizations by simulating these signals.

This enables the system to anticipate resource demand and recommend scaling actions across CPU, GPU, and confidential workloads. Key innovation of this architecture is the inclusion of a rules engine, which acts as a decision gate between AI recommendations and execution controlled by human intervention. While predictive model provides proactive scaling suggestions the rules engine embeds constraint requirements known to software engineers as well as validates them while generating recommendations. These constraints may include isolation requirements for confidential workloads, GPU MIG partitioning limits, disruption budgets during rolling upgrades, anti-affinity rules and cost or budget thresholds.

Observability is built into rules engine as well to flag any recommendations that violate these rules for human review after any automated adjustments that can be done thereby reducing the noise for humans as well. Once recommendations pass all validations, the next layer of policy and decision engine generates final instructions for cluster scaling, pod migration or resource bin packing. These decision pipe into actual IaC and automation pipelines via tools such as Helm, Terraform or any other CI/CD workloads. This ensures that the desired AI predicted cluster state is applied reliably to the infrastructure and maintains consistency across all node pools and workloads The architecture supports continuous closed-loop feedback and enabled hyperscale clusters to operate efficiently while respecting operational constraints which ensures efficient utilization and strict isolation for sensitive workloads. Additionally, the system is designed to support federated learning across multiple clusters allowing predictive models to generalize insights from one environment to another such as learning from experiments in a control setup and then replicating the same in production.

Building on these predictions, the Capacity Gap Estimation Module computes the delta between forecasted demand and available capacity. Rather than producing abstract utilization forecasts this component quantifies actionable capacity requirements, including the number of additional CPU and GPU nodes (with MIG profile awareness). This module incorporates suggested placements and then makes informed decisions in real time on capacity needs. Its output represents a precise capacity shortfall or surplus to enable deterministic planning decisions based on real time cluster snapshot. In summary, the architecture provides a hybrid framework that combines AI-driven foresight with rule-based safety, closed-loop automation, and declarative enforcement, enabling Kubernetes-based infrastructure at massive scale to operate predictively, safely, and efficiently.

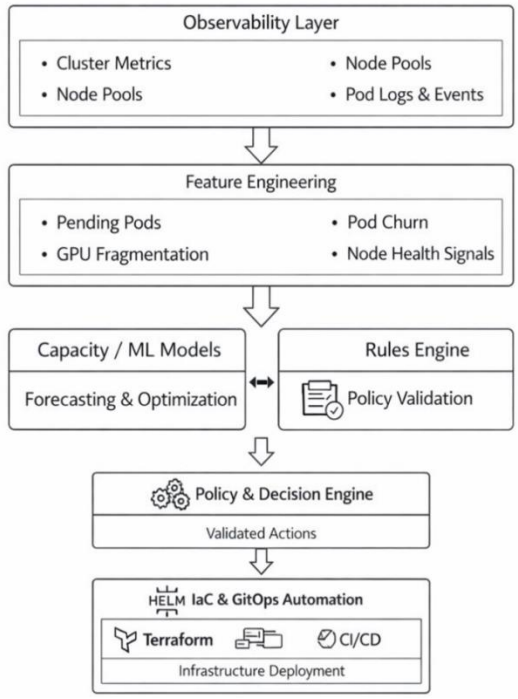

Fig 1: System Architecture

4. AI-Driven Kubernets Cluster Optimization

The motivation behind AI driven capacity planning is to directly enable Kubernetes cluster optimization by strongly aligning resource supply with scheduler behavior and workload dynamics. This is a missing piece often known to cause discrepancies between capacity planned and capacity needed as schedulers are often not fed information on hardware capabilities and vice versa achieving the following goals:

1. Scheduler Aware Capacity Planning: historically cluster sizing has often led to stranded capacity either due to bin packing inefficiencies, conservative disruption budgets or constraints that cannot be avoided due to company policies. AI based forecasting attempts to model scheduler signals to recommend optimal node pool sizes and shapes. This approach helps in reduced scale up latency during demand spikes.
2. Predictive Right Sizing of Node Pools: By combining AI forecasting for workload mix and growth forecasting AI models enable proactive right sizing of the node pools as well as get ability to predict what happens when new capacity is added and if the overall cluster efficiency is improved or degraded. Instead of uniformly scaling the clusters, this system can selectively scale specific pools and target applications of same nature (memory bound, latency bound or compute bound) accordingly resulting in reduced cost while preserving isolation.
3. Maintenance Aware Optimization: AI driven planning incorporates maintenance schedules, disruption budgets, and fault domain constraints. This allows specific clusters to maintain appropriate headroom to sustain any maintenance operations without causing service level degradation.

5. Confidential Workload Constraints

Not all workloads in a multi-tenant environment can be considered fungible or safe to co-locate with other workloads. Applications processing highly confidential information require strict isolation guarantees to comply with confidentiality or data access requirements. Examples include proprietary models, financial workloads or inference services processing users' private information. In such scenarios, traditional bin packing or capacity assumptions do not hold valid. AI driven capacity planning needs to incorporate such non stackable constraints such as pods cannot share nodes with other tenants, GPU partitioning is restricted, fault domain and security isolation requirements need to be met over utilization efficiency. AI models adapt to these constraints very effectively as these can be easily modeled during simulations and validated. Forecasting recommendations then focus on exclusive capacity demand per workload classification and preserve higher headroom to maintain security requirements. This shifts capacity planning objectives from maximum isolation to predictable isolation and cost efficiency is balanced against compliance risk. In this paper we assume that such workloads receive dedicated, forecast backed capacity while benefiting from predictive scaling and maintenance-aware planning.

6. Experimental Evaluation

In this paper we target evaluation of proposed system architecture in a Kubernetes deployment of 100,000 nodes and behavior under extreme load. The simulation accounts for API server rate limits, controller throughputs and *etcd* performance characteristics. GPU nodes are modeled with MIG partitioning and cross-tenant allocation logic. Confidential workloads are simulated via strict node isolation and mandatory anti affinity policies. Analysis is conducted by simulating pod arrival rates, replication requirements, and bursty workloads. Across all scenarios, the AI driven model is observed to maintain robust performance and reactive scaling is observed to have significant degradation under sudden workload spikes. These results demonstrate that predictive capacity planning is essential to maintain a highly scalable setup.

6.1. Simulation Setup

To get realistic results that are applicable directly in the software industry we model real-world large-scale deployments by simulating a data center with 100,000 nodes comprised of 60,000 CPU only nodes, 30,000 GPU nodes and 10,000 dedicated nodes for confidential workloads spread across multiple Kubernetes clusters which are further divided into multiple node pools reflecting this composition. Workloads running in these pools are comprised of CPU intensive pods with 3 replicas each, GPU heavy ML jobs with 2 replicas and MIG-aware allocation, and confidential applications requiring 1 replica and strict isolation. Pod placement constraints, fault domain isolation, multi-tenant separation are also modeled to emulate realistic scheduling at scale. Metrics, events and logs from these simulated clusters feeds into feature engineering pipeline producing required signals for predictive AI models and rules-based validation enabling policy compliant recommendations.

6.2. Methodology

As per the described architecture, time series forecasting using LSTM models estimated pod demand across different node pools in the clusters and reinforcement learning recommends scaling actions for allocation of nodes and pool resizes based on the simulated traffic patterns. Rules engineer accurately validated the recommendations and violations were adjusted in real time automatically or quickly escalated for human review. We evaluate 3 strategies in the experimental setup: static provisioning with conservative fixed buffers and manual allocation of nodes, reactive auto-scaling which triggers scale ups based on utilization and AI driven predictive planning via validations done by the rules engine. Key metrics include total utilization, scheduling latency, pending pods, fragmentation, SLA and rules violations.

6.3. Simulation Results

Table I. captures the results across different workloads, and we observe that with AI driven approach pending pods decreased dramatically, GPU fragmentation dropped by 30% and average node utilization improved by 22% compared to static provisioning. These reductions are observed with confidential workloads still maintaining strict isolation. The rules engine blocked close to 300 unsafe AI suggested actions thereby automatically preventing SLA violations or policy breaches that could have occurred at the expense of minor reductions in utilization.

Table 1: Simulation Results Summary

Metric	Static Provisioning	Reactive Autoscaling	Proposed Architecture
CPU Node Utilization	50%	62%	72%
GPU Node Utilization	48%	60%	78%
Confidential Node Utilization	80%	80%	91%
Pending Pods	12,500	6,000	1,800
GPU Fragmentation	High	Medium	Low
SLA Violations	4,000	2,100	500
Rule Violations	0	N/A	~320

7. Challenges and Limitations

While software industry is moving heavily use of AI, AI based capacity planning also poses several challenges that must be addressed:

- Data Volume and Velocity: Processing metrics, events and logs at scale and in real time introduces fast streaming requirements and heavy use of storage. This requires investment in state-of-the-art solutions using model training to preprocess and infer signals on the fly to ensure timely decisions.
- Generalized solutions: Predictive models that are trained on one cluster with a certain workload pattern may need to be re-trained on other clusters if there is a lot of heterogeneity in clusters. Often, in such cases companies move towards creating homogenous clusters that can be operated with some predictability.
- Security and Confidentiality: Deployment of workloads with strict confidentiality requirements limits bin packing and effective use of hardware and creates overhead to ensure resource isolation.
- Building Trust on AI decisions: AI decisions can be difficult to interpret specially in incident situations where reliable recovery takes most importance, and every predicted decision needs to be critically explained and understood. This is tackled by associating a confidence score on the predicted actions.
- Seamless integration: Often outputs of prediction models do not directly tie into CI/CD pipeline actions and need to be translated which introduces one more layer of delay and operational complexity.

8. Threats to Validity

Experiments conducted in this paper are subject to a few threats to validity. As the evaluation relies on simulated environments, they do not incorporate real-world dynamics such as unexpected network downtime, operator interventions, unexpected scheduler behaviors, unexpected workload behaviors etc. The logs and events used to train and evaluate may not exactly represent various scenarios that may happen in a production environment. Generalization of models is also limited with forecasting and validations done one set of workloads that may under perform in new cluster topologies. Emergency overrides, policy changes and human decisions can alter the effectiveness of results seen in the experiments however can be easily modeled. Additionally, multi cluster interactions and heterogeneity of infrastructure may impact the results and must be considered while interpreting the experimental results.

9. Future considerations

With AI advancements happening at a speedy rate, future research has potential to expand AI driven capacity planning and IaC in multiple directions such as:

- Multi-agent re-enforcement learning can be used to manage heterogeneous node pools which dynamically adjust resources while respecting necessary constraints
- Understanding impact and effect of resource allocation and workload performance with predictive models can improve reliability and reduce operational risk.
- Models can be trained across multiple clusters with varying deployment patterns to further improve prediction accuracy and support multi-cloud orchestration.

- Integrating AI with chaos testing frameworks can help preemptively identify weak points in overall scheduling strategy.
- Developing models and visualization tools that allow SRE teams to understand the recommendations is crucial for adoption of solution proposed in production environments.

10. Conclusion

Enterprise platforms backed by Kubernetes constantly demand capacity planning and infrastructure automation. Static buffers baked into capacity planning cannot keep up with complexities introduced by schedulers operating in a continuously changing heterogenous environment. Integrating AI tooling, basing scaling decisions on real time telemetry and adapting Infrastructure as Code to apply such decisions can allow organizations to encode operational learning directly into the platform. AI driven forecasting enables infrastructure to evolve safely, scale predictably, and maintain reliability over time. Architecture proposed in this study does not replace operator expertise and rather allows them to amplify it to operate Kubernetes platforms at scale in a pro-active manner.

References

[1] Google SRE Team, *Site Reliability Engineering*, O'Reilly Media, 2016.

[2] B. Burns et al., *Designing Distributed Systems*, O'Reilly Media, 2018.

[3] D. G. Feitelson, *Workload Modeling for Computer Systems Performance Evaluation*, Cambridge University Press, 2019.

[4] M. Isard et al., "Autopilot: Automatic Data Center Management," *OSDI*, 2017.

[5] A. Verma et al., "Large-scale Cluster Management at Google with Borg," *OSDI*, 2015.

[6] M. Schwarzkopf et al., "Omega: Flexible, Scalable Scheduler for Large Clusters," *SOSP*, 2013.

[7] NVIDIA, "GPU Operator for Kubernetes," Technical Documentation, 2023.

[8] A. Gupta et al., "etcd: A Distributed, Reliable Key-Value Store for the Most Critical Data of a Distributed System," Cloud Native Computing Foundation, 2016.

[9] C. McClurg et al., "Rules Engines for Cloud Automation: Ensuring Policy Compliance in AI-Driven Systems," *Journal of Cloud Computing*, vol. 12, 2022.

[10] H. Lim et al., "Enhancing GPU Utilization in Multi-Tenant Kubernetes Clusters," SC Conference, 2021.

[11] P. Bodik et al., "Resource Management in Multi-Tenant Cloud Clusters: Lessons from Hyperscale Deployments," *SoCC*, 2012.

[12] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[13] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[14] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[15] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd Edition, MIT Press, 2018.

[16] C. Re et al., "Machine Learning for Systems: Review and Opportunities," HotCloud, 2018.

[17] J. Brownlee, *Deep Learning for Time Series Forecasting*, Machine Learning Mastery, 2018.

[18] T. Chen et al., "XGBoost: A Scalable Tree Boosting System," *KDD*, 2016.

[19] P. Domingos, "A Few Useful Things to Know about Machine Learning," *Communications of the ACM*, vol. 55, no. 10, pp. 78–87, 2012.

[20] L. Kleinrock, *Queueing Systems, Volume 2: Computer Applications*, Wiley, 1976.