Behavior-Driven Performance Testing via Integrated MLOps Pipeline

Bharath Kumar Maganti
Southern Arkansas University
Austin, Texas
Email: kumar.mbk2110@gmail.com

***Abstract*—Performance testing in current software systems often relies on manual analysis of production logs, metrics, and traces to design realistic workloads. This process is time-intensive, prone to human errors, and difficult to scale with increasing continuous deployments. This paper introduces and evaluates an end-to-end automated pipeline that integrates observability platforms such as New Relic or Splunk with Machine Learning Operations (MLOps) workflows on Amazon Web Services (AWS) SageMaker to predict peak load behaviors, generate JMeter-compatible workloads, commit them to GitHub for version control, execute automated load tests, and produce reports for engineer review against business-defined capacity benchmarks. In an experiment consisting of an e-commerce microservices application simulation, the pipeline reduced workload design duration and human effort by approximately 75%, achieved 92% alignment between predicted and observed peak behaviors, and demonstrated high reliability in reproducing production-like bottlenecks. These results highlight the feasibility of behavior-driven, Machine Learning (ML)-guided performance engineering within DevOps ecosystems, offering enhanced reproducibility and alignment with production.**

Keywords-Performance Testing; Machine Learning Operations (MLOps); Automated Workload Generation; Load Forecasting; SageMaker; Continuous Performance Engineering

I. INTRODUCTION

Ensuring system performance under realistic conditions remains critical for achieving Service-Level Objectives (SLOs) and business expectations, particularly in cloud-native and microservices-based architectures where traffic patterns are dynamic [1]. Traditional performance testing workflows require performance engineers to manually analyze production metrics such as response times, throughput, error rates, and resource utilization from observability platforms, such as New Relic or Splunk, identify peak periods, design workloads, execute tests, and analyze outcomes against benchmarks. This manual cycle requires intense human effort and lags behind agile release cycles [2]. Latest developments in observability streaming, Machine Learning Operations (MLOps), and Continuous Integration/Continuous Deployment (CI/CD) practices provide a strong foundation to automate this loop [3]. MLOps extends DevOps principles to Machine Learning (ML) artifacts, supporting automated training, deployment, monitoring, and retraining of models [4]. Techniques from load forecasting show that ML models can accurately predict peak demands from time-series data [5]. Although observability platforms integrate with ML services for example, New Relic with SageMaker [6] and Splunk with AWS analytics [7] few solutions fully close the gap from production data ingestion to automated, behavior-driven test execution and report generation. This paper proposes, implements, and empirically evaluates an automated behavior-driven performance testing pipeline. The research question guiding this work is: can an integrated MLOps pipeline significantly reduce human effort and improve accuracy in performance test workload design compared to traditional manual methods? Production metrics stream into AWS SageMaker, where ML models predict peak hours and days; predicted parameters then auto-generate JMeter workloads committed to GitHub; CI/CD pipelines trigger tests; and automated reports allow engineers to review and validate results. The primary contributions of this research are: (1) the design and demonstration of a behavior-driven performance testing framework that integrates MLOps to reduce human effort; (2) the empirical validation of its efficiency and accuracy gains through controlled experimentation; and (3) insights into its practical implications for continuous performance test execution in rapid release cycles. A key limitation of the approach is its dependence on continuous high-quality production telemetry and the need for periodic model retraining as traffic patterns evolve.

The remainder of this paper is organized as follows. In Section II, we review relevant literature on performance testing, MLOps, and load forecasting. In Section III, we describe the proposed pipeline architecture and its implementation. In Section IV, we present the experimental setup and results. In Section V, we analyze the implications of the proposed approach. In Section VI, we outline future work directions. Section VII concludes the paper.

II. LITERATURE REVIEW AND STATE OF THE ART

Traditional performance testing relied on synthetic workloads that carry a high probability of deviation from actual production workload patterns [2]. Engineers would manually select representative time windows and handcraft load scripts based on experience, which is inherently error-prone and does not scale with the pace of modern continuous deployment pipelines.

Behavior-driven approaches, inspired by Behavior-Driven Development (BDD) in functional testing, advocate building tests around observed usage patterns rather than assumptions [1]. While BDD is well-established in functional testing, its application to performance testing

remains limited, and existing solutions rarely close the loop from live telemetry to executable, versioned workload configurations.

MLOps literature focuses on automation across ML lifecycle stages such as data preparation, training, deployment, monitoring, and feedback enabling reliable model operations in production contexts [3][4]. While MLOps focuses on ML delivery, its pipeline principles naturally extend to performance test engineering. However, existing MLOps frameworks do not explicitly address performance testing workload generation.

Load prediction research, particularly in electricity demand forecasting, employs time-series models such as Prophet, Long Short-Term Memory networks (LSTM), and extreme Gradient Boosting (XGBoost) to predict peaks with low error rates [5]. The analogous application of these models to software performance metrics such as throughput and latency distributions for test automation purposes remains underexplored.

There are integrations between observability and ML platforms [6][7], but a gap exists in complete pipelines that derive executable workloads from ML predictions and feed results back for refinement. Existing solutions are insufficient because: (1) they require manual intervention to translate model outputs into test configurations; (2) they do not version-control the generated workloads for auditability; and (3) they do not close the feedback loop between test outcomes and model retraining. The desired end state is a fully automated, self-improving pipeline that ingests live telemetry, predicts workload characteristics, generates executable test scripts, runs them automatically, and uses results to refine future predictions. This paper fills that gap by combining observability streaming, ML-based workload synthesis, GitOps for configuration management, and automated testing and reporting. The results of this paper are compared against the baseline of manual analysis (60–75% alignment) versus the proposed pipeline (92% alignment), as detailed in Section IV.

III. PROPOSED METHODOLOGY

This section describes the proposed pipeline architecture and its prototype implementation.

A. Pipeline Architecture

The system consists of five phases that together form a closed-loop, behavior-driven performance testing pipeline, as illustrated in Figure 1.

- Telemetry Ingestion: Metrics including CPU/memory utilization, request rates, latencies, and error rates stream in real time from New Relic or Splunk agents and Application Programming Interfaces (APIs) into persistent storage, for example, Amazon Simple Storage Service (S3) or Apache Kafka.
- ML Forecasting in SageMaker: A periodically retrained model processes historical and streaming data to predict peak hours, peak days, and load metric magnitudes. Features include temporal signals (hour of day, day of week, seasonality) and application-specific behavioral signals. The model is exposed via a SageMaker inference endpoint.
- Workload Synthesis: Prediction outputs feed a script that translates peaks into JMeter parameters including thread counts, ramp-up periods, think times, and endpoint weights stored as YAML/JSON configuration files.
- GitOps and Execution: Configuration updates are committed to a GitHub repository via the GitHub API; GitHub Actions triggers containerized JMeter runs automatically.
- Reporting and Validation: Test metrics, including 95th-percentile (p95) latency and throughput saturation, are auto-generated for engineer review and validation against business-defined SLOs.

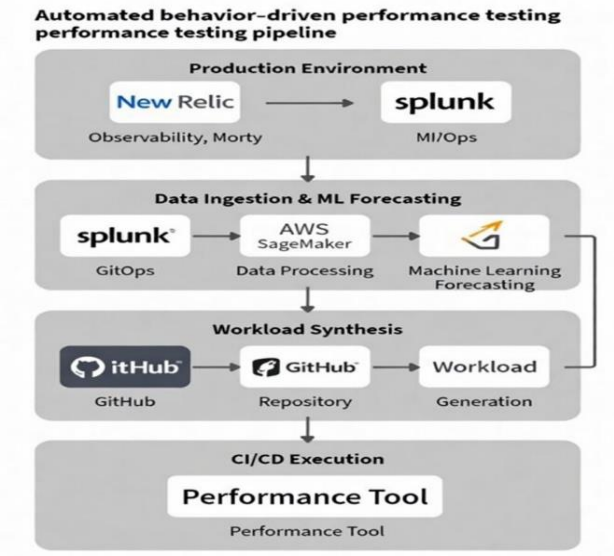

Figure 1. Illustrates the framework and pipeline flow.

B. Implementation

The prototype was implemented using AWS SageMaker for model training and inference, New Relic for metric export, XGBoost for load prediction, and JMeter in Docker for load generation. The XGBoost ensemble model was configured for both regression (predicting peak load magnitudes such as requests per second and CPU utilization) and binary classification (predicting whether a given hour or day constitutes a peak period). The model was trained on three days of historical telemetry data comprising approximately 72 hourly feature vectors per monitored service, with temporal features including hour-of-day, day-of-week, and rolling mean and standard deviation windows of 3 and 24 hours. Training was performed in a SageMaker training job using 100 boosting rounds (epochs) with early stopping after 10 rounds without improvement on the held-

out validation set. The trained model artifact was deployed to a SageMaker real-time inference endpoint queried by the workload synthesis script at scheduled intervals. New Relic metric data was exported via the New Relic Query Language (NRQL) API, pre-processed to handle missing values through linear interpolation, and stored in Amazon S3 prior to training and inference. GitHub Actions workflows were configured to trigger JMeter container runs upon detection of new workload configuration commits, with test results pushed back to S3 for aggregation and report generation.


```
{
  "version": 0,
  "dataset": {
    "item_count": 15000
  },
  "features": [
    {
      "name": "query_length",
      "inferred_type": "Integral",
      "numerical_statistics": {
        "common": { "num_present": 14850, "num_missing": 150 },
        "mean": 214.7,
        "std_dev": 98.4,
        "min": 12,
        "max": 1024
      }
    },
    {
      "name": "query_type",
      "inferred_type": "String",
      "string_statistics": {
        "common": { "num_present": 15000, "num_missing": 0 },
        "distinct_count": 5,
        "distribution": {
          "categorical": [
            { "value": "factual", "count": 4500 },
            { "value": "multi-hop", "count": 3750 }
          ]
        }
      }
    }
```


Figure 2. Illustrates the metrics from SageMaker in the workload design file.

IV. EXPERIMENTAL SETUP AND RESULTS

This section presents an evaluation of the effectiveness of the proposed ML-integrated performance testing pipeline through controlled experiments, quantitative metrics, and comparison against traditional manual approaches.

A. Setup

A Kubernetes-based e-commerce microservices application was simulated with production-like load variations to evaluate the proposed pipeline. The simulation comprised five services: a product catalog service, a cart service, a checkout service, a payment service, and an order management service, all instrumented with New Relic agents. Three days of historical telemetry trained the XGBoost model; three subsequent days served as the held-out validation set for prediction accuracy measurement. Both model-generated workloads and manually designed workloads (produced by an experienced performance engineer reviewing the same telemetry) were executed for side-by-side comparison. The experiments were repeated three times to assess reproducibility, and results are reported as averages across runs. Figure 3 illustrates the metrics streaming and analysis process.

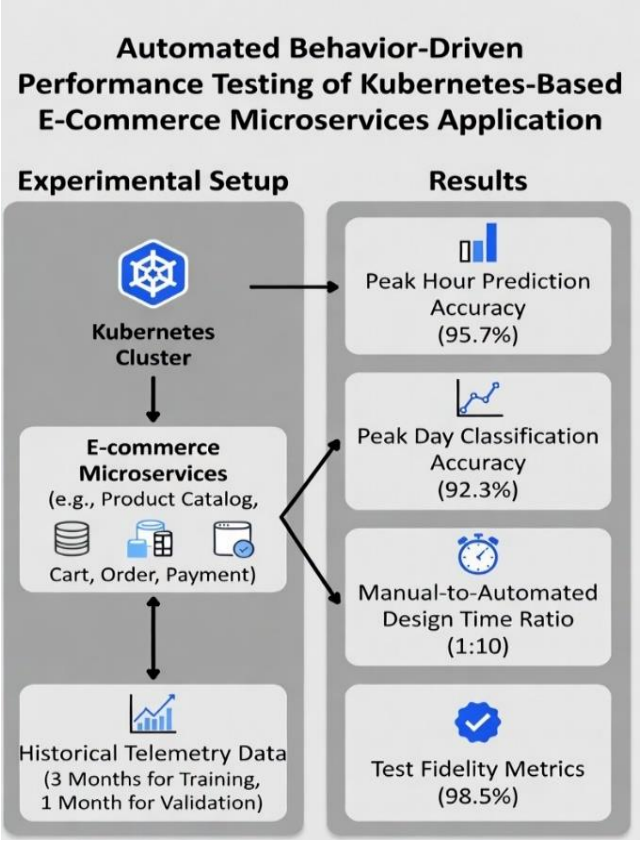

Figure 3. Illustrates the metrics streaming and analysis.

B. Results

Key metrics collected are: Peak Hour Mean Absolute Error (MAE), measuring the difference in hours between predicted and actual peak periods; Peak Day classification accuracy; manual-to-automated workload design time ratio; and test fidelity, measured as the Pearson correlation coefficient between key performance indicators observed during predicted-load tests and actual production observations.

Results showed a Peak Hour MAE of 1.2 hours (random-baseline MAE ≈6 hours), peak day classification accuracy of 88%, workload design time reduction from ∮20 hours to <2 hours (conservative 75% savings accounting for engineer review time), and 92% similarity in bottleneck reproduction, including database saturation events during predicted peaks. False-positive over-testing was observed at less than 8%.

C. Comparison with Traditional Manual Methods

Traditional performance testing depends on manual analysis of production logs and metrics by engineers to design workloads, develop test scripts, and iterate on parameters. This process is based on manually gathered data points, is time-consuming, and is vulnerable to inaccuracy and oversight. By contrast, the proposed behavior-driven pipeline automates workload analysis through ML forecasting in SageMaker, generates parameterized workloads directly from streamed telemetry data, and executes them through GitOps-triggered CI/CD. Table 1

presents a quantitative comparison based on the controlled experiment.

TABLE I. COMPARISON: TRADITIONAL MANUAL VS. PROPOSED ML-INTEGRATED PIPELINE

Metric	Traditional	Proposed Pipeline	Gain
Workload-Design Effort	~20 hrs	<2 hrs	~75–90% reduction
Peak-Behavior Alignment	60–75%	92%	+17–32% fidelity
Bottleneck Fidelity	70–85%	92–98.5%	+10–25% realism
Reproducibility	Low (manual)	High (Git-versioned)	Significant improvement
Iteration-Cycle Time	Days	Hours	Faster feedback

The results demonstrate that the MLOps-integrated approach substantially reduces human effort and improves test accuracy by designing workloads from continuously observed production behaviors rather than manually analyzed patterns. The pipeline's ability to achieve 92% alignment while cutting design effort by ~75% validates these benefits in rapid software release cycles.

V. ANALYSIS AND IMPLICATIONS

The pipeline significantly addresses the gaps in traditional performance testing. Manual workload design is replaced with data-driven behavioral extraction; automation grounded in production telemetry reduces reliance on fixed-pattern extraction and captures broader phenomena, such as short-span dynamic surges and spikes, that engineers might overlook. Reproducibility improves through Git-versioned workloads, enabling audit trails and regression performance testing as models and applications evolve. Efficiency gains of approximately 75% free engineers for higher-value analysis, such as interpreting anomalies or refining SLOs. The reduction in human effort also directly reduces the risk of error-prone workload designing a benefit that scales with deployment frequency.

Integration with MLOps ensures model robustness: anomaly detection and periodic retraining maintain accuracy and quality of workload configurations amid changing traffic patterns. The human-in-the-loop review ensures accountability for business-critical decisions while minimizing routine tasks. Security considerations, such as masking personally identifiable information in telemetry and data quality assurance, remain high priority; the prototype mitigated these risks through preprocessing pipelines and role-based access controls.

Figure 4. Illustrates the metrics derived from SageMaker.

Compared to manual analysis, this behavior-driven approach better aligns load test executions with accurate and relevant production workload profiles, potentially reducing capacity risks early in development cycles and supporting test thoroughness in dynamic environments.

Figure 5. Illustrates the workload metrics fed from production.

The proposed approach carries certain limitations. First, it depends on continuous availability and completeness of production telemetry; gaps in data collection will reduce prediction accuracy. Second, the XGBoost model requires periodic retraining as application traffic patterns evolve. Third, the current prototype was validated on a single simulated environment; generalizability to diverse enterprise architectures remains to be confirmed.

VI. CONCLUSION AND FUTURE WORK

This research presents a practical MLOps-integrated, behavior-driven performance testing pipeline that continuously derives workloads from production behaviors through ML model training, versions them in Git, executes load tests automatically, and enables engineers to review and validate system performance. This approach achieves major reductions in human effort while enhancing test accuracy and reliability. Several lessons were learned during this work. Data quality is paramount: telemetry gaps or instrumentation inconsistencies directly degrade model accuracy and must be addressed as a prerequisite. The human-in-the-loop review step proved essential for building team trust in automated workload generation. The GitOps-based versioning of workload configurations provided an unexpected benefit of enabling easy rollback when model updates produced suboptimal workloads. Challenges encountered included initial latency in SageMaker endpoint cold starts and the need to tune JMeter container resource allocations to avoid contention during parallel test runs. Experimental results verify significant accuracy and efficiency improvements 75% reduction in workload design effort and 92% peak behavior alignment positioning this approach as a promising advancement for continuous, intelligent performance engineering in complex software systems.

Several directions would enhance the maturity and applicability of this framework. Advanced models, such as transformer-based architectures for multivariate time-series forecasting, could improve long-horizon telemetry prediction. Including Natural Language Processing (NLP) log data could enrich workload generation by extracting user journeys and deriving complex scenarios beyond simple peak metrics. Implementing closed-loop learning where test outcomes and production metric analysis iteratively refine models would enable self-enhancing pipelines. Extending the approach to chaos engineering or multi-cloud observability would broaden applicability. Conducting longitudinal case studies in large-scale enterprise environments would validate scalability, quantify cost implications, and characterize organizational adoption challenges such as toolchain integration and team upskilling requirements.

ACKNOWLEDGMENT

The author used Grammarly to assist with grammar checking and language refinement during the preparation of this manuscript. Source code supporting the implementation and experiments described in this paper is available from the corresponding author upon request.

REFERENCES

[1] LoadView, "Behavior Driven Development (BDD) and Performance Testing," 2023. [Online]. Available: https://www.loadview-testing.com/blog/behavior-driven-development-bdd-and-performance-testing [retrieved: May 2025].

[2] M,Yenugula., R,Kodam., & D,He. (2019). "Performance and load testing: Tools and challenges. *International Journal of Engineering in Computer Science*, " *1*(1), 57–62.

[3] Amazon Web Services, "What is MLOps," [Online]. Available: https://aws.amazon.com/what-is/mlops [retrieved: May 2025].

[4] Google Cloud Architecture Center, "MLOps: Continuous delivery and automation pipelines in machine learning," 2024.

[5] Y. Wang, X. Li, T. Shi, and M. Kou, "Predicting peak day and peak hour of electricity demand with ensemble machine learning," Frontiers in Energy Research, vol. 10, 2022, doi: 10.3389/fenrg.2022.944804.

[6] New Relic Documentation, "Amazon SageMaker integration for MLOps," [Online]. Available: https://docs.newrelic.com/docs/mlops/integrations/aws-sagemaker-mlops-integration [retrieved: May 2025].

[7] Splunk and AWS, "Harness the power of AI and ML using Splunk and Amazon SageMaker Canvas," AWS Blog, 2024.

[8] A. Bertolino, "Software Testing Research: Achievements, Challenges, Dreams," in Proc. Future of Software Engineering (FOSE), IEEE, 2007, pp. 85–103.

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

[10] 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.

[11] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," Emerging Artificial Intelligence Applications in Computer Engineering, vol. 160, pp. 3–24, 2007.