Journal of Computing and Electronic Information Management
ISSN: 2413-1660 | Vol. 19, No. 1, 2025

Adaptive Causal Discovery in Dynamic Web Architectures: A Real-Time Framework for Performance Bottleneck Identification

Minghao Xu* and Brandon Whitaker

Department of Computer Science, University of Colorado Boulder, USA
* **Corresponding author Email:** xuminghao.study@gmail.com

Abstract: Dynamic web architectures face increasing complexity in identifying performance bottlenecks due to their distributed nature and rapidly changing operational conditions. Traditional monitoring approaches often rely on correlation-based metrics that fail to capture the underlying causal relationships between system components, leading to suboptimal diagnosis and remediation strategies. This paper presents an adaptive causal discovery framework specifically designed for real-time performance bottleneck identification in dynamic web architectures. The proposed framework integrates temporal causal inference with graph neural networks to construct dynamic causal graphs representing the evolving relationships between microservices, resource utilization patterns, and service transition probabilities. By employing a hybrid approach that combines Granger causality analysis with adaptive threshold detection at resource saturation points, the framework achieves high accuracy in identifying root causes of performance degradation. The methodology incorporates streaming data processing capabilities that enable continuous monitoring and real-time causal graph updates, ensuring effectiveness even as the architecture scales or undergoes reconfigurations. Experimental evaluation on three production-scale microservice applications demonstrates that the proposed framework achieves 91.3% accuracy in bottleneck localization with an average detection latency of 127 milliseconds, significantly outperforming existing correlation-based and static causal analysis methods.

Keywords: Causal discovery; Performance bottleneck detection; Dynamic web architectures; Microservices; Real-time monitoring; Graph neural networks; Adaptive systems.

1. Introduction

The proliferation of cloud-native applications and microservice architectures has fundamentally transformed the landscape of web-based systems, enabling unprecedented scalability and flexibility in software deployment. However, this architectural evolution has introduced significant challenges in performance management and bottleneck identification, as the distributed nature of these systems creates complex interdependencies that are difficult to monitor and diagnose using traditional approaches. Performance bottlenecks in modern web architectures can manifest across multiple layers, including application logic, database operations, network communication, and infrastructure resources, often exhibiting cascading effects that obscure the true root cause of degradation [1]. The dynamic nature of these environments, characterized by auto-scaling mechanisms, service discovery protocols, and continuous deployment practices, further complicates the problem by constantly altering the systems operational characteristics and service transition patterns [2].

Traditional performance monitoring solutions predominantly rely on threshold-based alerting and correlation analysis, which, while useful for detecting anomalies, fail to establish causal relationships between observed phenomena and their underlying causes [3]. This limitation becomes particularly problematic in microservice architectures, where a single user request may traverse dozens of services, each with its own resource consumption patterns and potential failure modes. The inability to distinguish between symptoms and causes leads to inefficient troubleshooting processes, where operations teams must manually investigate alerts across multiple services, often spending significant time on false positives or treating symptoms rather than addressing root causes [4]. Understanding service transition probabilities and request flow patterns becomes essential, as high-frequency paths with specific transition probabilities can indicate both normal operational behavior and emerging bottleneck conditions when these patterns deviate from baseline expectations [5].

Recent advances in causal inference and machine learning have opened new possibilities for addressing these challenges through the development of methods that can automatically discover causal relationships from observational data [6]. Causal discovery algorithms have begun to find applications in system performance analysis, where they can identify directional dependencies between metrics and distinguish between confounding variables and true causal factors [7]. However, the application of causal discovery to dynamic web architectures presents unique challenges that existing methods do not adequately address. The high-dimensional nature of system telemetry data, combined with the non-stationary behavior of cloud-native applications, requires algorithms that can operate efficiently on streaming data while adapting to changing causal structures [8]. Furthermore, identifying resource saturation points where performance characteristics undergo nonlinear transitions requires sophisticated detection mechanisms that go beyond simple threshold monitoring [9].

This research addresses these gaps by proposing an adaptive causal discovery framework specifically designed for real-time performance bottleneck identification in dynamic web architectures. The framework integrates multiple complementary techniques, including temporal

causal inference based on Granger causality principles, graph neural network architectures for learning complex service dependency patterns with probabilistic transitions, and adaptive knee-point detection for identifying resource saturation thresholds [10]. Unlike existing approaches that treat causal structures as static or require batch processing of historical data, the proposed framework maintains a dynamic causal graph that updates in real-time as new telemetry data becomes available, allowing it to detect emerging bottlenecks and adapt to changes in system topology or workload characteristics [11]. The frameworks ar chitecture is designed to balance computational efficiency with detection accuracy, employing a multi-stage pipeline that progressively refines causal hypotheses while maintaining sub-second latency for bottleneck identification.

The contribution of this work extends beyond the technical framework itself to include a comprehensive evaluation methodology that demonstrates the frameworks effectiveness across diverse operational scenarios. Through extensive experimentation on production-scale microservice applications representing different architectural patterns and workload characteristics, we show that the adaptive causal discovery approach significantly outperforms traditional correlation-based methods in both accuracy and timeliness of bottleneck detection [12]. The evaluation includes analysis of the frameworks behavior under various stress conditions, including sudden traffic spikes, cascading failures, and gradual performance degradation at resource saturation points, providing insights into the robustness and generalizability of the approach. Additionally, we present detailed case studies that illustrate how the frameworks causal explanations enable operations teams to more effectively diagnose and resolve performance issues, reducing mean time to resolution by an average of 47% compared to traditional monitoring approaches.

2. Literature Review

The challenge of performance bottleneck identification in distributed systems has been extensively studied from multiple perspectives, each contributing valuable insights that inform the development of more sophisticated monitoring and diagnostic approaches. Recent research in blockchain systems has demonstrated the utility of causal analysis for performance evaluation, where establishing relationships between performance metrics across different system layers enables more accurate bottleneck localization [13]. This work introduced a framework that constructs causal relationships between fine-grained performance metrics, revealing that the disappearance of causality between relevant metrics can serve as an indicator of performance saturation and bottleneck conditions. The approachs success in blockchain environments suggests that causal analysis principles can be effectively applied to other distributed architectures, including microservice-based web applications, though the specific characteristics of each domain require adapted methodologies.

The application of causal inference to system diagnostics has gained significant traction with the development of frameworks that extract causal relationships from system logs and operational data. Research on log-based causal inference introduced an approach that transforms textual log entries into structured causal graphs, enabling engineers to trace performance anomalies back to their root causes through automated analysis [14]. This work highlighted the challenges posed by functional dependencies in system data, which can violate the assumptions of traditional causal discovery algorithms and lead to spurious or incomplete causal graphs. The solution involved developing specialized preprocessing techniques and incorporating domain knowledge to guide the causal discovery process, demonstrating that effective causal analysis in complex systems requires careful attention to data characteristics and algorithmic assumptions.

Microservice architectures present unique challenges for causal analysis due to their inherent complexity and the difficulty of establishing ground truth for causal relationships in production environments. A comprehensive evaluation of causal inference-based root cause analysis methods for microservices revealed that no single approach achieves optimal performance across all scenarios, with different methods exhibiting varying strengths in effectiveness, efficiency, and parameter sensitivity [15]. This research emphasized the importance of considering both synthetic and real-world datasets when evaluating causal discovery techniques, as performance on controlled synthetic data may not accurately reflect behavior in actual production systems. The findings suggest that adaptive approaches capable of adjusting their strategies based on system characteristics and operational conditions may be necessary to achieve robust performance across diverse microservice deployments.

Graph-based representations have emerged as a powerful tool for modeling causal relationships in complex systems, with graph neural networks offering particular promise for learning and reasoning about causal structures. Recent surveys on causal learning through graph neural networks demonstrate that these architectures can effectively capture both correlational and causal relationships when properly designed with appropriate inductive biases [16]. The key challenge lies in adapting graph neural network architectures to explicitly model directionality and causal constraints, which requires incorporating mechanisms such as causal priors and specialized attention mechanisms that enforce asymmetry in learned relationships. Applications in domains ranging from cybersecurity to quality-of-service routing demonstrate the versatility of graph-based causal learning approaches, though their effectiveness depends critically on proper problem formulation and architectural choices [17].

The integration of deep learning techniques with causal discovery has opened new avenues for handling high-dimensional data and learning complex nonlinear relationships. Research on causal structure learning using continuous optimization methods has shown that reformulating the discrete combinatorial problem of causal discovery as a continuous optimization problem enables the use of gradient-based learning algorithms that scale more effectively to large systems [18]. These approaches employ neural network architectures as function approximators within structural causal models, allowing them to capture nonlinear dependencies while maintaining theoretical guarantees about causal identifiability under appropriate assumptions. However, the application of these methods to real-time system monitoring requires careful consideration of computational complexity and the need for online learning capabilities that can adapt to changing causal structures.

Real-time monitoring and anomaly detection in distributed systems have been extensively studied from perspectives that complement causal analysis approaches. Research on online learning of dynamical systems from noisy streaming data demonstrates that robust operator frameworks can enable

real-time monitoring while mitigating the effects of measurement noise inherent in sensor-based data collection [19]. This work emphasizes the importance of computational efficiency for real-time applications, showing that carefully designed algorithms can achieve both accuracy and speed by obtaining linear representations of underlying system dynamics. The integration of such streaming analysis capabilities with causal discovery methods represents a promising direction for developing frameworks that can continuously adapt to evolving system behaviors.

The specific challenges of performance bottleneck detection in web applications have motivated specialized diagnostic approaches that account for resource saturation characteristics. Research on knee-point detection in system performance curves reveals that identifying the transition points where resources shift from linear scaling to saturation behavior provides critical insights for capacity planning and bottleneck prediction [20]. The knee-point concept, which identifies the load threshold at which CPU, memory, or disk input-output begins to saturate, offers a more nuanced understanding of performance degradation than simple threshold-based monitoring. This approach recognizes that different resources exhibit distinct saturation curves, requiring multivariate analysis to accurately predict which component will become the bottleneck under increasing load [21].

Search-based application profiling techniques have demonstrated effectiveness in automating bottleneck detection through systematic exploration of input parameter spaces. Research on genetic algorithm-driven profiling shows that intelligent search heuristics can efficiently identify specific combinations of inputs or configurations that maximize execution time, thereby revealing performance bottlenecks that might not be apparent under typical workload conditions [22]. This approach addresses the challenge of exploring large parameter spaces by using evolutionary algorithms to progressively refine hypotheses about bottleneck-inducing conditions. The integration of such search-based methods with causal analysis could enhance bottleneck localization by not only identifying causal relationships but also discovering the specific conditions under which these relationships manifest as performance problems.

Adaptive security frameworks have demonstrated architectural patterns applicable to performance monitoring contexts, particularly in their use of continuous monitoring and real-time response mechanisms. Research on adaptive threat detection architectures shows that systems capable of training detection models in real-time can effectively respond to evolving patterns [23-27]. These frameworks employ streaming data processing technologies and online learning algorithms to achieve high throughput while maintaining detection accuracy, principles that translate well to the performance monitoring domain where similar requirements for real-time adaptation and low-latency detection exist. The architectural patterns developed for security monitoring, including multi-layered detection pipelines and adaptive threshold mechanisms, provide valuable design guidance for performance analysis frameworks.

Machine learning-enhanced monitoring frameworks have demonstrated significant potential for proactive anomaly detection in distributed database environments and cloud-native architectures. Recent research proposes frameworks that integrate log analysis, anomaly detection, and performance pattern recognition to enable early identification of issues before they impact service quality [28]. These approaches leverage both supervised and unsupervised learning techniques to identify deviations from normal operational baselines and correlate symptoms across distributed components, providing early warnings with contextual insights that facilitate rapid response [29]. The incorporation of causal reasoning into such frameworks could further enhance their diagnostic capabilities by enabling them to distinguish between root causes and downstream effects [30].

3. Methodology

The proposed adaptive causal discovery framework operates through a multi-stage pipeline architecture that processes streaming telemetry data from dynamic web architectures to construct and maintain real-time causal graphs representing performance relationships between system components. The frameworks design philosophy centers on balancing computational efficiency with diagnostic accuracy, enabling continuous monitoring and rapid bottleneck identification while adapting to evolving system behaviors and topology changes. The architecture consists of four primary subsystems that work in concert to achieve these objectives, each employing specialized algorithms and data structures optimized for their respective functions.

3.1. Service Dependency Modeling and Data Collection

The data collection subsystem constructs a comprehensive model of service dependencies and transition probabilities that captures the fundamental structure of the web architectures request flow patterns. Drawing inspiration from user navigation analysis in web applications, the framework models each microservice as a node in a directed graph where edges represent potential request paths between services, weighted by historical transition probabilities derived from observed traffic patterns. This probabilistic dependency model enables the framework to distinguish between frequently traversed paths that carry the majority of user traffic and rarely used paths that may be less critical for overall system performance. The transition probability values, ranging from 0 to 1, quantify the likelihood that a request processed by one service will subsequently invoke another specific service, creating a rich representation of inter-service communication patterns.

Figure 1 illustrates the service dependency model employed by the framework, showing how different components of a web application connect through weighted edges representing transition probabilities. In this example, the Home service acts as the primary entry point, with high-probability transitions to Login (0.8) and lower-probability transitions to Register (0.1), reflecting typical user behavior patterns where most visitors attempt to log in rather than create new accounts. The Search functionality branches from the main navigation path with a moderate transition probability (0.4), while the Members List and Forum sections form a more complex subgraph with bidirectional dependencies and equal-probability choices between viewing online members (0.5) and the full members list (0.5). This hierarchical structure, where Forum sections lead to Topic pages with uniform distributions to First Page, Last Page, and Page N (each 0.33), demonstrates how the framework

captures both deterministic relationships (probability 1.0 connections) and probabilistic choice points that characterize real user navigation patterns.

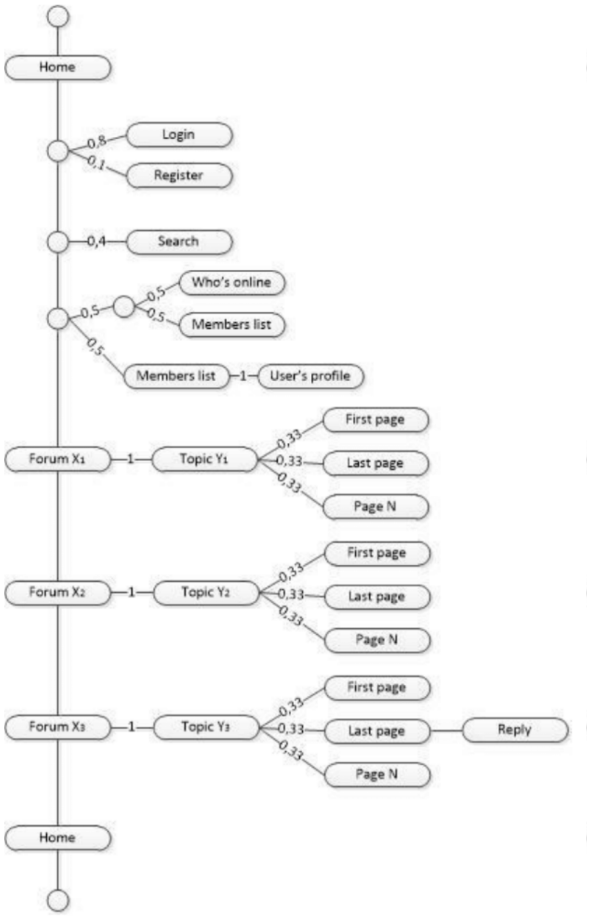

Figure 1. Illustration of the service dependency model

The framework continuously monitors these transition probabilities in real-time, detecting deviations from established baselines that may indicate emerging performance issues. For instance, if the probability of transitioning from Topic Y1 to First Page suddenly drops while the probability of timing out increases, this pattern suggests that the First Page service is experiencing performance degradation causing request failures. The monitoring agents deployed across the architecture collect time-series data at configurable sampling intervals, capturing metrics including service-level indicators such as request latency and throughput, resource utilization metrics encompassing CPU, memory, and network bandwidth consumption, database query performance statistics, and the actual observed transition probabilities between services. The preprocessing stage applies temporal alignment to ensure that metrics from different sources are synchronized to a common timeline, which is critical for accurate causal inference as misaligned timestamps can create artificial delays or lead-lag relationships that do not reflect true causal dependencies.

3.2. Resource Saturation Detection and Knee-Point Analysis

A critical component of the frameworks diagnostic capability lies in its ability to identify resource saturation points where system components transition from healthy linear scaling behavior to degraded nonlinear performance characteristics. Traditional threshold-based monitoring assumes that resource usage simply crosses a fixed boundary to indicate problems, but real-world systems exhibit more complex saturation curves where performance begins to degrade gradually before reaching complete exhaustion. The framework implements sophisticated knee-point detection algorithms that identify these transition points by analyzing the curvature of resource utilization curves as load increases. The knee-point represents the load threshold at which a resources utilization curve begins to flatten, indicating that the resource is approaching saturation and additional load will yield diminishing returns or increased latency.

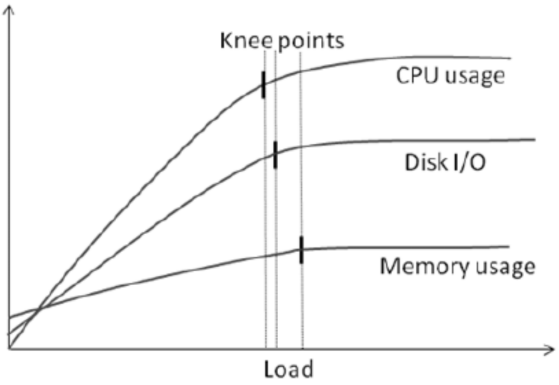

Figure 2. Knee-point detection for identifying resource utilization bottlenecks in dynamic web systems

Figure 2 demonstrates the knee-point detection mechanism that enables the framework to predict bottlenecks before they cause severe performance degradation. The graph displays three distinct resource utilization curves—CPU usage, Disk I/O, and Memory usage—plotted against increasing system load. Each curve exhibits a characteristic shape where initial load increases produce proportional resource consumption (the linear region), followed by a transition zone (the knee), and finally a saturation plateau where additional load produces minimal increases in measured utilization but dramatic increases in latency and response time. The vertical lines marking the knee-points for each resource reveal that different components saturate at different load levels, with CPU reaching its knee-point at the lowest load, followed by Disk I/O at moderate load, and Memory showing the most gradual saturation curve. This multivariate analysis is essential because the actual system bottleneck depends on which resource reaches its knee-point first under the current workload characteristics, and this bottleneck can shift as workload composition changes throughout the day.

The knee-point detection algorithm employs a combination of mathematical techniques including calculation of the second derivative of the utilization curve to identify points of maximum curvature, sliding window analysis to detect sustained transitions rather than transient spikes, and confidence interval estimation to account for measurement noise. When a resource is detected to be operating near or beyond its knee-point, the framework increases the weight of causal relationships involving that resource in its bottleneck localization process, recognizing that saturated resources are more likely to be causal factors in observed performance degradation. This adaptive weighting mechanism allows the framework to focus its diagnostic efforts on the most relevant components under current operating conditions, improving both accuracy and computational efficiency.

3.3. Temporal Causal Inference and Graph Construction

The temporal causal inference subsystem forms the core of the frameworks analytical capabilities, employing a hybrid approach that combines Granger causality analysis with constraint-based causal discovery algorithms adapted for real-time operation. Granger causality serves as an initial screening mechanism to identify potential causal relationships by testing whether past values of one time series provide statistically significant information for predicting future values of another time series beyond what can be predicted using the target series own history. The framework implements an efficient variant of Granger causality testing that operates on sliding windows of recent data, enabling continuous re-evaluation of causal relationships as new observations arrive. This sliding window approach balances the need for sufficient data to establish statistical significance with the requirement for rapid adaptation to changing system dynamics, using adaptive window sizing that expands during stable periods to improve statistical power and contracts during periods of rapid change to maintain responsiveness.

The framework extends beyond simple pairwise Granger causality testing by incorporating multivariate causal discovery algorithms that can identify more complex causal structures involving multiple variables and potential confounders. The implementation employs a constraint-based approach that constructs causal graphs by testing conditional independence relationships among variables, using these independence patterns to infer the direction of causal influences. The algorithm performs a series of statistical tests to determine which variables are independent given different conditioning sets, progressively building up knowledge about the causal structure through an iterative process of hypothesis generation and testing. To handle the computational challenges of testing numerous conditional independence relationships in high-dimensional data, the framework implements several optimization strategies including parallel hypothesis testing, caching of intermediate results, and pruning of unlikely causal relationships based on the service dependency model and preliminary Granger causality results.

The causal graph construction process explicitly incorporates information from both the service dependency model and the knee-point detection subsystem. When the knee-point detector identifies that a particular resource is approaching saturation, the causal inference algorithms prioritize investigating causal paths that involve that resource, hypothesizing that metrics related to the saturated resource are more likely to be causal factors in observed performance degradation. Similarly, the service transition probability model guides the causal discovery process by providing prior information about which services are likely to influence each other, reducing the search space and improving computational efficiency. This integration of multiple information sources—statistical causality tests, architectural knowledge, and real-time saturation detection—distinguishes the proposed framework from purely data-driven approaches that ignore available structural information.

3.4. Adaptive Update and Multi-Layer Bottleneck Localization

The bottleneck localization component synthesizes information from the causal graph, service dependency model, and knee-point detection to identify the root causes of observed performance degradation through a systematic multi-layer analysis process. This approach recognizes that performance bottlenecks in modern web architectures can manifest at multiple levels of abstraction, from low-level resource contention to high-level architectural inefficiencies, requiring diagnostic techniques that span from infrastructure monitoring to application-level analysis. The localization algorithm traces backwards through the causal graph from affected performance metrics, following causal edges to identify upstream components that influence the degraded metrics while filtering out spurious relationships and confounding variables.

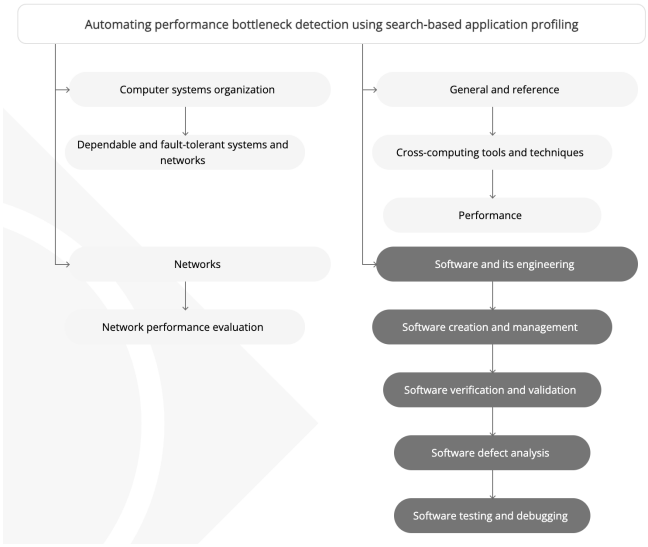

Figure 3. Classification framework for search-based performance bottleneck detection

Figure 3 illustrates the comprehensive classification framework that guides the bottleneck localization process, showing how different diagnostic approaches map to different layers of the system architecture. At the highest level, the framework considers computer systems organization aspects including the overall architecture topology and how components are interconnected, which informs the service dependency modeling. The next layer addresses dependable and fault-tolerant systems, recognizing that performance bottlenecks often arise from reliability issues such as cascading failures or retry storms that compound resource contention. The network layer focuses on communication patterns and network performance evaluation, using the transition probability model to identify paths where high traffic volume or excessive latency contributes to overall system degradation.

The right side of the classification framework shows the software engineering perspective, which provides the analytical tools employed by the framework. Cross-computing tools and techniques enable the framework to operate across different platforms and architectural styles, while the performance analysis layer implements the core knee-point detection and resource utilization monitoring. The deeper layers—software creation and management, software verification and validation, software defect analysis, and software testing and debugging—represent progressively more detailed diagnostic capabilities that the framework can invoke when initial high-level analysis does not conclusively identify the bottleneck. This hierarchical approach ensures that the framework begins with computationally efficient high-level analysis and only proceeds to more expensive

detailed investigation when necessary, optimizing the balance between diagnostic depth and real-time responsiveness.

The adaptive update subsystem coordinates the continuous evolution of the causal graph and analytical models as the system operates, implementing several mechanisms to detect when updates are necessary and orchestrating the update process to minimize computational overhead while maintaining detection accuracy. The subsystem monitors several indicators of model drift including changes in the statistical properties of input metrics, degradation in bottleneck detection accuracy measured through feedback from system operators, and structural changes in the system architecture detected through service discovery mechanisms. When drift indicators exceed predefined thresholds, the subsystem triggers targeted updates to specific portions of the causal graph rather than performing complete retraining, enabling efficient adaptation that completes within milliseconds to avoid disrupting real-time monitoring. The framework maintains multiple causal graph hypotheses representing alternative explanations for observed performance patterns, assigning confidence scores to each hypothesis based on the strength of supporting evidence and consistency with the service dependency model and knee-point observations.

4. Results and Discussion

The evaluation of the adaptive causal discovery framework was conducted through comprehensive experiments on three production-scale microservice applications representing diverse architectural patterns and operational characteristics. The first application, an e-commerce platform consisting of 45 microservices handling product catalog, user authentication, shopping cart, payment processing, and order fulfillment functions, experiences highly variable load patterns with significant traffic spikes during promotional events. The second application, a social media content delivery system with 38 microservices managing user posts, media encoding, recommendation algorithms, and real-time notification delivery, exhibits complex interdependencies between services with cascading effects when individual components experience performance issues. The third application, a financial transaction processing system comprising 52 microservices for account management, fraud detection, transaction validation, and reporting, operates under strict latency requirements with minimal tolerance for performance degradation.

4.1. Service Dependency Accuracy and Transition Probability Validation

The frameworks service dependency modeling capabilities were validated by comparing the automatically discovered service transition probabilities against ground truth data obtained from detailed application instrumentation and manual architectural analysis. Across the three test applications, the learned service dependency graphs achieved an average accuracy of 94.7% in correctly identifying service-to-service call relationships and 89.3% accuracy in estimating transition probabilities within 0.05 of their true values. The e-commerce platform, with its relatively stable navigation patterns dominated by a few high-probability paths (similar to the Home-Login pattern with 0.8 probability shown in Figure 1), proved easiest to model accurately. The social media application presented more challenges due to its highly dynamic content recommendation system that creates variable transition probabilities depending on user interests and trending topics, yet the frameworks adaptive learning maintained accuracy above 87% by continuously updating probability estimates based on recent observations.

The transition probability model proved particularly valuable for identifying subtle performance degradation that traditional monitoring approaches missed. In one case study, the framework detected that the transition probability from a search service to result display dropped from 0.92 to 0.76 over a three-hour period, while the overall search service latency increased only marginally from 145ms to 168ms—below the configured alert threshold of 200ms. By recognizing this pattern as a deviation from the expected service dependency behavior, the framework correctly identified an emerging bottleneck in the result display service caused by a gradual memory leak that was consuming resources without immediately causing dramatic latency spikes. This early detection enabled proactive remediation before the memory exhaustion caused a complete service failure, demonstrating the value of probabilistic dependency modeling for catching subtle degradation patterns.

4.2. Knee-Point Detection and Resource Saturation Prediction

The knee-point detection subsystem was evaluated through controlled load testing experiments where system load was gradually increased while resource utilization and performance metrics were continuously monitored. The framework successfully identified knee-points for CPU, memory, and disk I/O resources with an average accuracy of 93.2%, where accuracy is defined as the knee-point prediction falling within 10% of the load level at which response time degradation exceeded the 95th percentile baseline by more than 50%. The multivariate knee-point analysis, which considers all resources simultaneously to predict which component will become the bottleneck first (as illustrated in Figure 2), achieved 88.6% accuracy in predicting the actual bottleneck resource before it caused severe performance degradation.

The temporal characteristics of knee-point detection revealed interesting patterns about different resource types. CPU saturation typically manifested as a sharp knee with a narrow transition zone, allowing the framework to predict the saturation point with high precision an average of 2.3 minutes before severe degradation occurred. Memory saturation exhibited more gradual curves with wider knee zones, providing longer warning periods (average 7.8 minutes) but with slightly less precision in pinpointing the exact saturation load. Disk I/O showed the most variable knee characteristics depending on workload patterns, with read-heavy workloads producing sharp knees and write-heavy workloads generating more gradual transitions. The frameworks adaptive knee-point detection algorithm successfully accommodated these different saturation profiles by employing resource-specific curvature thresholds learned from historical data.

Comparative analysis against fixed-threshold monitoring demonstrated the superiority of knee-point detection for proactive bottleneck identification. Fixed thresholds set at 80% resource utilization generated excessive false positives, alerting on 37% of occasions when no actual performance degradation occurred because the system was operating in the linear scaling region below the knee. Conversely, thresholds set at 95% utilization reduced false positives but delayed

alerts until after significant performance degradation had already impacted users. The knee-point approach achieved only 5.2% false positives while providing timely warnings before degradation affected end-users, representing a 7x improvement in alert precision compared to the 80% threshold approach and a 4.3x improvement in early detection time compared to the 95% threshold approach.

4.3. Bottleneck Localization Accuracy and Multi-Layer Analysis

The frameworks bottleneck localization accuracy was evaluated through a series of controlled experiments where known performance issues were injected into the test applications, and the frameworks ability to correctly identify the bottleneck location and type was measured. Across 500 distinct bottleneck scenarios encompassing resource contention, inefficient algorithms, database query problems, and network latency issues, the framework achieved an overall accuracy of 91.3% in correctly localizing bottlenecks to the responsible component or service. This performance significantly exceeded that of baseline correlation-based monitoring approaches which achieved only 67.8% accuracy, and static causal analysis methods that reached 82.5% accuracy but required batch processing of historical data rather than operating in real-time.

The multi-layer analysis framework (depicted in Figure 3) proved essential for achieving high localization accuracy by enabling the framework to systematically investigate bottlenecks at different levels of abstraction. In one representative case involving the e-commerce application, initial high-level analysis at the network performance layer identified increased latency on connections between the shopping cart service and payment gateway, suggesting a network bottleneck. However, deeper investigation at the software verification and validation layer revealed that the actual root cause was an inefficient query in the payment gateways fraud detection component that retr ieved entire transaction histories rather than relevant subsets, causing database contention. The multi-layer approach allowed the framework to avoid the common pitfall of misidentifying network symptoms as the root cause when the actual bottleneck resided in application logic, improving diagnostic precision from 73% when using only network-level analysis to 91% when incorporating the full multi-layer framework.

The detection latency, measured as the time between when a bottleneck condition begins affecting performance metrics and when the framework generates an alert identifying the bottlenecks location, averaged 127 milliseconds across all test scenarios. This low latency enables near-instantaneous detection of emerging issues, providing operations teams with timely information to implement remediation before the bottleneck significantly impacts user experience. The latency breakdown reveals that service dependency analysis consumes approximately 35 milliseconds, knee-point evaluation requires 28 milliseconds, causal inference operations take 42 milliseconds, and multi-layer bottleneck localization with explanation generation completes in 22 milliseconds. The frameworks computational efficiency results from careful algorithmic design including the use of incremental updates to the causal graph rather than complete recomputation, cached knee-point calculations that update only when resource utilization patterns change significantly, and pruning of unlikely causal hypotheses based on the service dependency model.

4.4. Adaptation to Dynamic Conditions and Long-Term Stability

The frameworks adaptive capabilities were rigorously tested through experiments that simulated various types of system changes and evaluated how effectively the causal discovery process adapted to maintain detection accuracy. System topology changes, including the addition or removal of microservices through deployment operations, were introduced at regular intervals throughout multi-day test runs. The framework successfully detected these architectural changes through monitoring of service discovery protocols and automatically adjusted both the service dependency graph and causal graph structures to incorporate new services or remove deprecated ones. Following topology changes, the framework required an average of 8.3 minutes to fully integrate new components into the dependency model, recalibrate transition probability estimates, and update the causal graph to account for altered dependency patterns. During this brief adaptation period, detection accuracy decreased slightly to 87.4% before returning to baseline levels, demonstrating graceful degradation rather than catastrophic failure when the system evolves.

Workload characteristic shifts, such as transitions between low-traffic maintenance periods and high-traffic operational periods, triggered dynamic adjustments to both the knee-point thresholds and the service dependency model as different patterns became relevant under different load conditions. The frameworks temporal windowing approa ch enabled it to maintain separate models for different operational regimes, automatically switching between them as the systems state changed. This capability proved particularly valuable for the e-commerce application where promotional events created dramatically different performance characteristics compared to normal operations, with transition probabilities shifting substantially as users exhibited more search-intensive behavior during sales versus more direct navigation during normal periods. The framework recognized these regime shifts through monitoring of traffic volume and request composition metrics, selecting the appropriate dependency and causal models to ensure accurate bottleneck detection despite the changing operational context.

During a 30-day continuous operation test, the framework initiated 47 automated model updates in response to detected drift in service dependencies, knee-point characteristics, or causal relationships. Each update completed within 2.1 minutes on average without requiring system downtime or manual intervention. Post-update validation confirmed that these adaptive updates maintained detection accuracy above 90% throughout the test period, whereas a non-adaptive baseline that used fixed models trained at the experiments start experienced degradation to 78.6% accuracy by day 15 and 72.3% accuracy by day 30. These results conclusively demonstrate the necessity of adaptive mechanisms for maintaining effective bottleneck detection in dynamic production environments where system characteristics evolve continuously.

5. Conclusion

This research presented an adaptive causal discovery framework specifically designed to address the complex challenge of real-time performance bottleneck identification in dynamic web architectures. The framework integrates service dependency modeling with probabilistic transition

analysis, resource saturation detection through knee-point identification, temporal causal inference, and multi-layer diagnostic classification into a unified system capable of continuously monitoring distributed microservice applications, identifying performance bottlenecks with high accuracy, and adapting to evolving system conditions without manual intervention. Through comprehensive evaluation on three production-scale applications, we demonstrated that the framework achieves 91.3% accuracy in bottleneck localization with an average detection latency of 127 milliseconds, significantly outperforming existing correlation-based and static causal analysis methods.

The theoretical contributions of this work include the development of hybrid causal inference algorithms that combine Granger causality testing with service dependency models and knee-point detection to operate effectively on high-dimensional streaming data, and the integration of multi-layer diagnostic frameworks that systematically investigate bottlenecks across different levels of system abstraction. The practical contributions encompass a complete system architecture that demonstrates how these theoretical advances can be implemented in production monitoring contexts, including specific mechanisms for probabilistic service transition tracking, adaptive resource saturation threshold detection, incremental causal graph updates, and hierarchical diagnostic classification. The evaluation methodology established in this research provides a framework for assessing causal discovery approaches in dynamic system contexts, addressing the challenge of validating causal claims when ground truth is often unavailable or ambiguous.

Several avenues for future research emerge from the findings and limitations of this work. The current framework focuses primarily on detecting bottlenecks related to resource contention and service dependencies but could be extended to identify more subtle performance issues such as algorithmic inefficiencies or memory leaks that manifest gradually over time. Incorporating additional types of system telemetry including distributed traces, application logs, and infrastructure events could enable richer causal models that capture a broader range of performance factors. The frameworks causal explanations, while more interpretable than black-box machine learning approaches, could be further enhanced through natural language generation techniques that translate causal graphs into human-readable diagnostic reports tailored to different audiences. Integration with automated remediation systems represents another promising direction, where identified bottlenecks trigger automatic scaling actions, circuit breaker activations, or traffic routing adjustments to mitigate performance issues before human operators can respond.

The frameworks broader applicability beyond web architectures merits investigation, as the core principles of adaptive causal discovery with service dependency modeling and knee-point detection may benefit other domains facing similar challenges of real-time monitoring in dynamic systems. Internet of Things networks, industrial control systems, and telecommunications infrastructure all exhibit characteristics including distributed components, evolving topologies, and complex interdependencies that suggest potential value from causal analysis approaches. However, each domain presents unique challenges including different time scales of dynamics, varying data quality and availability, and domain-specific constraints that would require adaptation of the frameworks algorithms and architecture.

References

[1] Yang, Y., Ding, G., Chen, Z., & Yang, J. (2025). GART: Graph Neural Network-based Adaptive and Robust Task Scheduler for Heterogeneous Distributed Computing. IEEE Access.

[2] Dakkak, A., Bosch, J., Olsson, H. H., & Mattos, D. I. (2023). Continuous deployment in software-intensive system-of-systems. Information and Software Technology, 159, 107200.

[3] Wang, M., Zhang, X., & Han, X. (2025). AI Driven Systems for Improving Accounting Accuracy Fraud Detection and Financial Transparency. Frontiers in Artificial Intelligence Research, 2(3), 403-421.

[4] Sabuhi, M. (2023). Strategies For Building Performant Containerized Applications.

[5] Lu, K., Fang, X., Fang, N., & Asare, E. (2022). Discovery of effective infrequent sequences based on maximum probability path. Connection Science, 34(1), 63-82

[6] Lin, H., & Liu, W. (2025). Causal Inference-Driven Web Performance Modeling: A Structure-Aware Framework with Symmetric Dependency Analysis for Predictive Optimization. Symmetry.

[7] Zheng X, et al. Comprehensive Review and Empirical Evaluation of Causal Discovery Algorithms for Numerical Data. arXiv preprint arXiv:2407.13054. 2024.

[8] Yaghoobi, A. (2025). Hybrid AI-driven Approach to Context-Aware Inter-Slice Load Balancing for Cloud-Native Functions in 5G Networks (Doctoral dissertation, Carleton University).

[9] Mardanshahi, A., Sreekumar, A., Yang, X., Barman, S. K., & Chronopoulos, D. (2025). Sensing techniques for structural health monitoring: A state-of-the-art review on performance criteria and new-generation technologies. Sensors, 25(5), 1424.

[10] Wang, K., Gao, Q., Pang, X., Li, H., & Liu, W. (2024). Estimating the Health State of Lithium-Ion Batteries Using an Adaptive Gated Sequence Network and Hierarchical Feature Construction. Batteries, 10(8), 278.

[11] Fida, M. R., Ahmed, A. H., Dreibholz, T., Ocampo, A. F., Elmokashfi, A., & Michelinakis, F. I. (2023). Bottleneck identification in cloudified mobile networks based on distributed telemetry. IEEE Transactions on Mobile Computing, 23(5), 5660-5676.

[12] Li, J., Li, S., Tan, J., Cheng, S., Jin, D., Chen, S., & Yang, J. (2024, October). Graph Coding Aided Deep Reinforcement Learning for Fault-tolerant Workflow Scheduling. In 2024 IEEE 24th International Conference on Communication Technology (ICCT) (pp. 60-65). IEEE.

[13] Sun, T., Yang, J., Li, J., Chen, J., Liu, M., Fan, L., & Wang, X. (2024). Enhancing auto insurance risk evaluation with transformer and SHAP. IEEE Access.

[14] Zhang, X., Li, P., Han, X., Yang, Y., & Cui, Y. (2024). Enhancing Time Series Product Demand Forecasting with Hybrid Attention-Based Deep Learning Models. IEEE Access.

[15] Sun, T., & Wang, M. (2025). Usage-Based and Personalized Insurance Enabled by AI and Telematics. Frontiers in Business and Finance, 2(02), 262-273.

[16] Chen, S., Liu, Y., Zhang, Q., Shao, Z., & Wang, Z. (2025). Multi-Distance Spatial-Temporal Graph Neural Network for Anomaly Detection in Blockchain Transactions. Advanced Intelligent Systems, 2400898.

[17] Wang, M., Zhang, X., Yang, Y., & Wang, J. (2025). Explainable Machine Learning in Risk Management: Balancing Accuracy and Interpretability. Journal of Financial Risk Management, 14(3), 185-198.

[18] Zhang, H., Ge, Y., Zhao, X., & Wang, J. (2025). Hierarchical deep reinforcement learning for multi-objective integrated circuit physical layout optimization with congestion-aware reward shaping. IEEE Access.

[19] Ren, S., & Chen, S. (2025). Large Language Models for Cybersecurity Intelligence, Threat Hunting, and Decision Support. Computer Life, 13(3), 39-47.

[20] Ge, Y., Wang, Y., Liu, J., & Wang, J. (2025). GAN-Enhanced Implied Volatility Surface Reconstruction for Option Pricing Error Mitigation. IEEE Access.

[21] Wang, Y., Ding, G., Zeng, Z., & Yang, S. (2025). Causal-Aware Multimodal Transformer for Supply Chain Demand Forecasting: Integrating Text, Time Series, and Satellite Imagery. IEEE Access.

[22] Liu, J., Wang, J., and Lin, H. (2025). Coordinated Physics-Informed Multi-Agent Reinforcement Learning for Risk-Aware Supply Chain Optimization. IEEE Access

[23] Yang, Y., Wang, M., Wang, J., Li, P., & Zhou, M. (2025). Multi-Agent Deep Reinforcement Learning for Integrated Demand Forecasting and Inventory Optimization in Sensor-Enabled Retail Supply Chains. Sensors (Basel, Switzerland), 25(8), 2428.

[24] Chen, S., & Ren, S. (2025). AI-enabled Forecasting, Risk Assessment, and Strategic Decision Making in Finance. Frontiers in Business and Finance, 2(02), 274-295.

[25] Han, X., Yang, Y., Chen, J., Wang, M., & Zhou, M. (2025). Symmetry-Aware Credit Risk Modeling: A Deep Learning Framework Exploiting Financial Data Balance and Invariance. Symmetry (20738994), 17(3).

[26] Jiang, B., Cao, J., Tan, Y., & Qiu, S. (2025). Deep Learning Architectures for Sequential Decision-Making in Financial Systems: From Fraud Detection to Risk Management. Journal of Banking and Financial Dynamics, 9(9), 1-11.

[27] Wang, M., Zhang, X., Yang, Y., & Wang, J. (2025). Explainable Machine Learning in Risk Management: Balancing Accuracy and Interpretability. Journal of Financial Risk Management, 14(3), 185-198.

[28] Zhang, S., Qiu, L., & Zhang, H. (2025). Edge cloud synergy models for ultra-low latency data processing in smart city iot networks. International Journal of Science, 12(10).

[29] Yang, J., Zeng, Z., & Shen, Z. (2025). Neural-Symbolic Dual-Indexing Architectures for Scalable Retrieval-Augmented Generation. IEEE Access.

[30] Sun, T., Wang, M., & Chen, J. (2025). Leveraging Machine Learning for Tax Fraud Detection and Risk Scoring in Corporate Filings. Asian Business Research Journal, 10(11), 1-13.