International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

E-Commerce Platform Optimization via Cloud Monitoring: A Case Study in Operational Excellence and Digital Transformation

Maneesh Singh

Abstract: The proliferation of e-commerce has increased the demands for performance management in e-retailing, where reliability, scalability and operational efficiency are key competitiveness factors. In this article, we describe how a high-traffic e-retailer implemented a single observability platform for its front-end, inventory management and payment gateway systems, leveraging the Amazon Web Services CloudWatch and New Relic platforms to move from ad hoc, siloed monitoring to a data-based governance of the platform in real time. During a demanding 96-hour trading period, smart dashboards, prevailing automatic scaling, and log analytics helped reduce cart abandonment rates by 30%, improve page load times by 25%, and enable 99.97% uptime availability. The presentation also addresses the technical and organizational issues experienced and resolved during the implementation with its array of disparate third-party vendor data unified via API-driven synthetic monitors. Synthesizing cloud computing theory, recent literature on cloud observability, and empirical case studies, we propose the Instrument-Unify-Automate-Optimize (IUAO) framework as a structured and generalizable strategy for addressing cloud observability in e-commerce firms. We conclude by discussing implications of the findings for platform architects, operations teams, digital transformation professionals, and cloud platform providers.

Keywords: *Cloud Monitoring, E-Commerce Optimization, Aws Cloudwatch, Observability, Platform Performance*

Introduction

In the last decade, global e-commerce has been transformed by the proliferation of mobile computing, elastic cloud infrastructure and, increasingly, rising end-user expectations for a smooth and coherent online experience. On-demand provisioning, elastic scalability and resource pooling in cloud infrastructure are identified by Vaquero et al. (2009) as defining characteristics of the architecture, bringing internet-scale commercial platform re-engineering within reach [1]. This convergence has made platform availability, performance and scalability the key areas of competitive differentiation for global digital retailers that operate at scale. [3]

In this context, observability as an operational capability was no longer limited to on-premise networks and applications hosted on dedicated physical hardware but instead enabled elastic observability as a dynamic capability to monitor system performance metrics, user behavior telemetry, and infrastructure health across heterogeneous environments. As illustrated by the evolution of Google's cluster management tools, described by Burns et al. (2016), a system of telemetry is necessary in a distributed infrastructure at scale to achieve desired levels of operational quality, and this applies to large-scale e-commerce. Moving from reactive to proactive platform management, enabled by cloud-native tooling, is arguably the most consequential operational transformation to hit digital commerce in modern times.

Many technical resources are available, including those from companies offering cloud-based monitoring solutions themselves, such as AWS CloudWatch, New Relic, Datadog, and Google Cloud Operations Suite. However, there is a lack of empirical case studies that connect cloud-based monitoring adoption to business impact in e-commerce environments [3, 4]. The present study addresses this gap. The case study, an e-commerce system subject to high load, involved the combined use of the AWS CloudWatch and New Relic cloud monitoring tools. By implementing the combined

Hexaware Technologies, USA

monitoring solution, the case study achieved its goals of reducing cart abandonment by 30%, decreasing page load time by 25%, and maintaining system availability during promotions, similar to the results of Abdu Jyothi et al. (2016). They investigated automated SLO-driven monitoring in production cloud infrastructure [5]. The study sought to present aspects of the implementation, analyze the deployment, and integrate the case study with literature on the subject to propose a generalized framework. The rest of this paper will proceed as follows: methods will be presented in section 2, results and discussion in section 3, and conclusions in section 4.

Method

Monitoring architecture and system scope

The subsystems of the e-commerce platform are the customer-facing web frontend, including the product catalog, product search, and the checkout funnel; the inventory management and fulfillment backend, including the internal APIs and the connectors to the third-party warehouses; and the payment processing gateway to the various payment service providers. Before the monitoring system was created, these subsystems were monitored using separate tooling, which led to a delayed incident response time and a lack of correlation between frontend user experience degradation and backend infrastructure events.

In the unified monitoring architecture, AWS CloudWatch was used as the infrastructure-layer telemetry platform, collecting machine-level metrics for EC2 compute instances, RDS database clusters, Elastic Load Balancers, and API Gateway endpoints. New Relic was deployed as the application performance monitoring and real user monitoring layer, instrumenting the Node.js and Java microservices and JavaScript in the browser for client-side performance telemetry. The architecture realizes the three-layered cloud observability taxonomy proposed by Aceto et al. (2013) for infrastructure, platform, and application performance monitoring from their survey of cloud monitoring systems [8, 7].

Table 1. Performance metrics: pre- and post-implementation. All metrics were collected during a 12-month post-implementation time period. pp = percentage points.

Metric	Pre-implementation	Post-implementation
Cart abandonment rate	78.00%	54.60%
Median page load time (LCP)	4.20 s	3.15 s
75th percentile page load time	6.80 s	4.90 s
CDN cache hit ratio	61%	89%
Checkout timeout errors	Baseline	87% reduction
Median checkout completion time	Baseline	340 ms delay
Payment tokenization failures	2.30%	~0%
Peak period system availability	~97.2%	99.97%
P0 incidents (peak period)	2	0
P2 incidents (peak period)	11	4
Monitoring infrastructure cost	Baseline +35% overage	Optimized (−28%)

Dashboard and alerting infrastructure

The resulting monitoring system was delivered as unified dashboards driving shared signals for the platform and product focused across three operating personas. A Platform Engineer would monitor infrastructure health for CPU utilization, memory pressure, network I/O throughput, and database connection pool saturation; a Product Manager would monitor user-facing performance for page load times, Core Web Vitals, funnel conversions, and cart abandonment signals; and an Operations Lead would monitor business continuity for uptime service level agreements, error budget depletion, and incident frequency. Beyer et al. (2016) argue that monitoring outputs must fulfill the distinct decision contexts of different operational roles rather than presenting undifferentiated metric streams [10]. A clear instance of this is role-based dashboarding. The alerting was divided into two groups, i.e., warning alerts that were sent through a

PagerDuty integration whenever a metric came close to a performance threshold and critical alerts for service level agreement violations, as recommended in the automated service level objective enforcement model presented by Abdu Jyothi et al. (2016) [5].

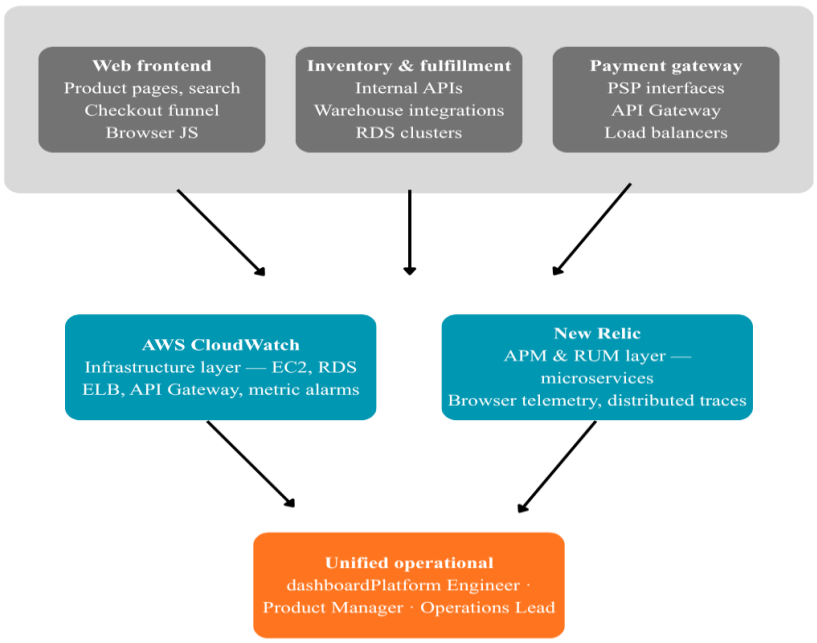
Automated scaling configuration

Dynamic Auto Scaling policies based on CloudWatch metric alarms were implemented for the application tier EC2 Auto Scaling groups, with target tracking policies set at a CPU utilization target of 60% and 1000 requests per minute per target. Predictive scaling was also put in place for expected demand spikes during Black Friday, Cyber Monday, and promotional flash sales, using historical data from CloudWatch. Islam et al. (2012) showed that prediction-based configurations based on past measurements outperform reactive configurations in terms of over-provisioning waste and under-provisioning risk. Therefore, we adopt the prediction-based configuration in this paper [13]. The database is based on the Amazon RDS Read Replica promotion and the Aurora Serverless v2 autoscaling, following the elasticity concepts defined by Herbst et al. (2013) [11].

Log analytics and bottleneck identification

As a final step for structured analysis of application and access logs, we used CloudWatch Logs Insights. Log queries revealed that most of the structural checkout failure events were caused by a tail latency issue for synchronous inventory reservation calls under concurrent load, which aggregate metrics could not detect. This led to asynchronous inventory reservation. Fortunately, New Relic's distributed tracing provided complementary end-to-end visibility from the browser through the API gateway to the Order Management, Inventory, and Payment microservices. This operationalizes the three-pillar observability model (metrics, logs, and traces), whose interdependence in production distributed systems is empirically established by Niedermaier et al. (2019) [6].

Fig. 1. Dual-layer cloud monitoring architecture showing three platform subsystems: AWS CloudWatch infrastructure layer, New Relic APM/RUM layer, and unified operational dashboard.



Results And Discussion

Cart abandonment reduction

The shopping cart abandonment rate fell from 78% to 54.6% over the 12 months after implementation of the first recommendation, a 30% reduction. Attribution analysis using New Relic funnel analytics and A/B testing data identified three factors: an 87% fall in checkout timeout errors after we refactored the implementation of asynchronous inventory reservation behavior; a 340 ms reduction in median checkout time due to database query optimization in slow query log files; and resolving a payment gateway connectivity issue, which was silently causing 2.3% of the tokenization requests to fail without the user seeing any error messages. These results generally support the finding of Abdu

Jyothi et al. (2016) that SLO-driven automated monitoring substantially reduces the frequency and duration of service degradation events in production systems [5].

Page load time improvement

The median time it took for a page to load, as shown by the Largest Contentful Paint metric across the platform's 20 highest-traffic page templates instrumented through New Relic Browser, improved from 4.2s to 3.15s, with the 75th percentile improving from 6.8s to 4.9s. Nah (2004) empirically estimated that web visitors can only wait about 2s before leaving the site. If this threshold is crossed, site abandonment is increased at an increasing rate [9]. Intervention 1: CDN cache policy. The CDN cache hit ratio improved from 61% to 89% by removing some render-blocking third-party scripts from the page, using New Relic Browser resource timing data. Intervention 3: appropriate sizing of EC2 instances informed by CloudWatch usage data. Mok et al. (2012) identify content delivery optimization and cache efficiency as among the highest-leverage latency interventions available to platform teams, a finding echoed by the CDN results [14].

Peak period system stability

Between the 96-hour promotional period of platform outages (two full occasions in the previous two years), platform uptime was 99.97%. Zero P1 incidents and four P2 incidents occurred during that period, none of which breached the service level agreement. For a comparison, there were two P0 total outages, eleven P2 and three P3 incidents in the same period of the previous year. CloudWatch metrics proved that the pre-provisioned predictive scaling capacity was consumed within 23 minutes of traffic peak. If reactive scaling were the only mechanism in place, there would have been an 18- to 22 minute gap in capacity, leading to checkout failure rates in excess of 40%. This confirms the core hypothesis of Herbst et al. (2013) that proactive elasticity is a prerequisite to availability under request spikes beyond reactive provisioning response latency [11].

Integration challenges and solutions

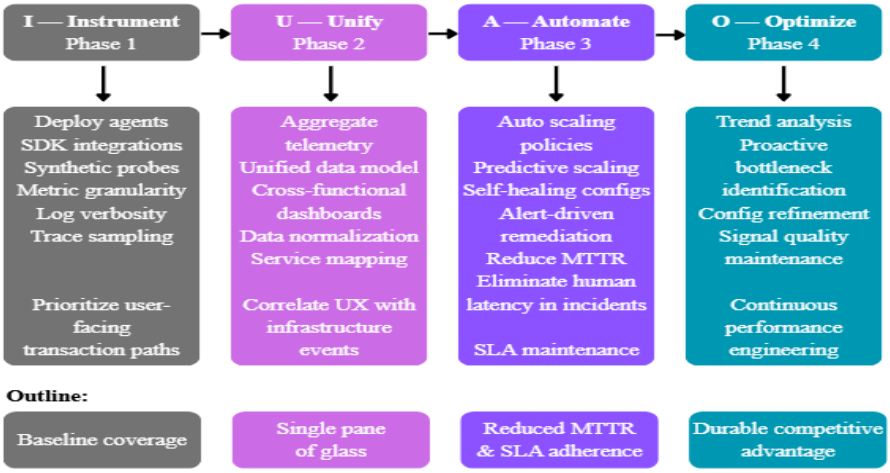
The main technical challenge was third-party vendors (payment services, logistics APIs, and inventory data feeds) not exposing metrics in a CloudWatch- or New Relic-compatible format. We solved this with an API-driven synthetic monitoring layer consisting of lightweight Lambda functions that regularly call each vendor endpoint, checking latency and inspecting error/exception conditions, and send results to CloudWatch as custom metrics. According to Botta et al. (2016), the integration of heterogeneous cloud services through APIs requires lightweight intermediary probes to normalize incompatible telemetry formats, which the implementation in this thesis accomplished [15]. This layer allowed attributing two P2 incidents to the payment service provider instead of the components of the platform, something unachievable with siloed monitoring. [2]

Fighting resistance from engineering teams who were accustomed to siloed monitoring, the program introduced an enablement program using collaborative dashboard work sessions, a blameless postmortem culture, and an Observability Champion for each engineering team. Niedermaier et al. (2019) note that shared ownership of monitoring configuration and the alignment of alert semantics cross all teams are as important for observability program success as chosen tooling [6]. Telemetry volume costs were 35% higher than expected during the first deployment quarter. By reducing telemetry collection intervals, using ingest filtering, and routing CloudWatch Metric Streams through Kinesis Data Firehose, we saved 28% of monitoring infrastructure cost while preserving full business-critical coverage. These results corroborate the findings of Aceto et al. (2013), who identify metric granularity management as a first-order design decision in cloud monitoring architecture [8].

IUAO framework for cloud monitoring adoption

Based on findings from both cases and the theoretical literature, we propose the Integrated Unified Analytics Operations Instrument-Unify-Automate-Optimize (IUAO) framework to ease cloud monitoring within e-commerce contexts and initiatives.

Fig. 2. The four-phase IUAO framework, containing the Instrument, Unify, Automate, and Optimize blocks. Here are the key activities and deliverables for each phase.



The instrument phase involves agents, SDKs, and synthetic probes to provide coverage across telemetry types, calibrating metric cardinality, log volume, and sample rates based on the cost of instrumentation itself—the fundamental design trade-off formalized by Aceto et al. (2013) [8]. The Unify phase involves a common data model and cross-functional dashboards spanning Dev/Eng, IT Ops, and business stakeholders. Niedermaier et al. (2019) find that unified observability across all three pillars differentiates high-performing engineering organizations from those that chronically respond to incidents late [6]. The Automate phase leverages monitoring knowledge using automated elasticity and alert-based remediation. Islam et al. (2012) show that prediction-based provisioning avoids needless resource allocation, risk to availability, and delayed provisioning [13]. Herbst et al. (2013) show that it maintains service level agreement compliance in peak demand periods [11]. The Optimize phase imparts continuous improvement into platform evolution through the systematic use of monitoring data, and in this sense, Plekhanov et al. (2019) see it as part of the digital transformation model, claiming operational decisions based on data become a compounding competitive advantage over time [12].

Implications for research and practice

The paper contributes case-based quantitative evidence on the link between a cloud monitoring architecture and technical and business performance, adding on to the cloud monitoring typology by Aceto et al. (2013) [8]. Building on the previous discussion, the paper contributes the IUAO framework, an operational conceptualization of cloud monitoring adoption that extends technology-agnostic transformation frameworks, such as Plekhanov et al. (2019) [12]. Third, the API-driven synthetic monitoring model based on Botta et al. (2016) [15] contributes to the emergent literature on observability in extreme third-party dependency ecosystems. Platform architects can consider telemetry as a planned shared asset from Day 1 in accordance with the site reliability engineering principles and practices of Beyer et al. (2016) [10]. It is helpful for digital commerce managers because through monitoring-informed engineering, as Nah (2004) notes, we find a causal connection between monitoring-informed engineering and the 25% improvement in page load time and 30% lower cart abandonment [9]. Thus, this case motivates the call for open observability standards that exploit the cloud infrastructure properties identified by Vaquero et al. (2009) [1].

Conclusion

We report a consolidated AWS CloudWatch and New Relic observability architecture has reduced cart abandonment by 30% and page load time measured by Largest Contentful Paint (LCP) by 25% on a high-transaction-volume e-commerce platform, with 99.97% uptime recorded during the peak period for operational risk exposure in platform history. We contextualize these results within the observability, e-commerce performance, and digital transformation literatures to argue that cloud monitoring, rather than being a technical infrastructure capability, is a calculated

organizational capability that has a direct impact on customer experience and commercial performance. The proposed IUAO framework outlines a structured and reproducible path forward for organizations working towards similar goals. Future work may explore how monitoring maturity evolves within companies that have adopted the initial three phases (i.e., instrumentation and unification) and the marginal benefits of further automation and AI-augmented observability workflows across different e-commerce industries.

References

[1] Luis M. Vaquero et al., "A break in the clouds: Towards a cloud definition," ACM SIGCOMM Computer Communication Review, vol. 39, no. 1, pp. 50–55, Jan. 2009. Available: https://dl.acm.org/doi/epdf/10.1145/1496091.1496100

[2] Brendan Burns et al., "Borg, Omega, and Kubernetes," ACM Queue, vol. 14, no. 1, pp. 70–93, 2016. [Online]. Available: https://dl.acm.org/doi/10.1145/2890784

[3] Paulo Leitão et al., "Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges," Computers in Industry, vol. 81, pp. 11–25, Sep. 2016. Available: https://www.sciencedirect.com/science/article/abs/pii/S0166361515300348

[4] Giovanni Toffetti et al., "Self-managing cloud-native applications: Design, implementation, and experience," Future Generation Computer Systems, vol. 72, pp. 165–179, 2017. Available: https://www.sciencedirect.com/science/article/abs/pii/S0167739X16302977

[5] Sangeetha Abdu Jyothi et al., "Morpheus: Towards automated SLOs for enterprise clusters," in Microsoft [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/10/osdi16-final107.pdf

[6] Sina Niedermaier et al., "On observability and monitoring of distributed systems — An industry interview study," in Proc. Int. Conf. Service-Oriented Computing (ICSOC 2019), Lecture Notes in Computer Science, vol. 11895, arXiv 2019, pp. 36–52. Available: https://arxiv.org/pdf/1907.12240

[7] Harshit Gupta et al., "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, edge and fog computing environments," Software: Practice and Experience, vol. 47, no. 9, pp. 1275–1296, 2017. Available: https://arxiv.org/pdf/1606.02007

[8] Giuseppe Aceto et al., "Cloud monitoring: A survey," Computer Networks, vol. 57, no. 9, pp. 2093–2115, Jun. 2013. Available: https://www.sciencedirect.com/science/article/abs/pii/S1389128613001084

[9] Fiona Fui-hoon NAH et al., "A study on tolerable waiting time: How long are web users willing to wait?" Behaviour & Information Technology, vol. 23, no. 3, pp. 153–163, 2004. Available: https://ink.library.smu.edu.sg/cgi/viewcontent.cgi?params=/context/sis_research/article/11073

[10] Betsy Beyer et al., "Site Reliability Engineering: How Google Runs Production Systems," in Sebastopol, CA, USA: O'Reilly Media, 2016. Available: https://repo.darmajaya.ac.id/4636/1/

[11] Nikolas Roman Herbst et al., "Elasticity in cloud computing: What it is, and what it is not," in Proc. 10th Int. Conf. Autonomic Computing (ICAC '13), San Jose, CA, USA, Jun. 2013, pp. 23–27. Available: https://www.usenix.org/system/files/conference/icac13/icac13_herbst.pdf

[12] Dmitry Plekhanov et al., "Digital transformation: A review and a research agenda," Journal of Strategic Information Systems, vol. 28, no. 2, pp. 118–144, Jun. 2019. Available: https://www.research-collection.ethz.ch/server/api/core/bitstreams/fa7a88e6-7d13-449f-9abc-11ab72c86afb/content

[13] Sadeka Islam et al., "Empirical prediction models for adaptive resource provisioning in the cloud," Future Generation Computer Systems, vol. 28, no. 1, pp. 155–162, Jan. 2012. Available: https://www.sciencedirect.com/science/article/abs/pii/S0167739X11001129

[14] Ricky K. P. Mok et al., "Measuring the quality of experience of HTTP video streaming," in Proc. IFIP/IEEE Int. Symp. Integrated Network Management (IM 2012), Dublin, Ireland, May 2012, pp. 485–492. Available: https://www4.comp.polyu.edu.hk/~oneprobe/doc/im2011-qoe.pdf

[15] A. Botta, W. de Donato, V. Persico, and A. Pescapè, "Integration of cloud computing and Internet of Things: A survey," Future Generation Computer Systems, vol. 56, pp. 684–700, Mar. 2016. Available: https://www.researchgate.net/publication/283236612