The Hidden Business Costs of Ignoring Performance Testing - The Silent Budget Killer

Sowmya Chintakindi
Independent Researcher and Sr. Performance Engineer, USA
e-mail: sowmyar909@gmail.com

***Abstract*—In this era of artificial intelligence and the digital world, the speed and responsiveness of the application have become a critical factor in maintaining a competitive edge. With increasing user expectations of a seamless and high-performance application, even minor delays can lead to customer dissatisfaction and may drive them to competitors. This increase in user expectations made performance testing one of the crucial aspects of the software development life cycle to evaluate the application's speed, reliability, and responsiveness under varying load conditions. Despite its critical role, organizations often overlook performance testing until failure strikes, users leave, and revenue is lost. This paper aims to raise awareness of the importance of performance testing and the consequences of ignoring it. Through real-world case studies and industry insights from practical experience, this paper highlights the impact of inadequate performance testing on the business. Also, it explores best practices to make applications scalable, reliable, and efficient. In a world where every milliseconds matters, performance testing shouldn't be an option - it's a necessity.**

Keywords-Performance testing; Reliability; User experience; Hidden costs.

I. INTRODUCTION

Performance testing is typically positioned at the final stage after development and functional testing. Due to this, it frequently receives limited time and attention as teams spend most of the time in development and validation. The common assumption is that bypassing performance testing can save time and accelerate deployment if no significant performance-related changes are made. However, this will be done at a hidden cost. The actual cost of this may not be immediate. Still, the risks accumulate beneath the surface in unexpected outages, slowness, business loss, and frustrated customers who may never return. This paper explores the hidden costs of bypassing performance testing that organizations cannot afford to ignore and provides some strategies to have seamless and resilient applications.

This paper starts with research methodology in section 2 and then provides background on performance testing, how it is performed, and how it helps businesses in section 3. Section 4 provides some understanding of IT outages, their causes, effects, and preventive measures. This is followed by some case studies on applications that were affected by bypassing performance testing and the loss incurred in section 5. Conclusions are drawn in section 6.

II. RESEARCH METHODOLOGY

This study captured some of the real-time case studies to analyze the importance of performance testing. This research highlights that even minor modifications can impact overall performance and potentially lead to revenue loss. This study demonstrated that continuous and thorough performance validation is essential for maintaining system reliability and business outcomes.

A. Description and purpose of the paper

This study highlights the importance of implementing performance tests and active monitoring practices from early development to production deployment. It is designed to raise awareness, guide organizations, and advocate for a performance-first mindset to avoid unanticipated business losses [1].

B. Research questions

1) RQ1: What happens when performance testing is ignored before a release?

Rationale: When performance testing is ignored before a release, applications will be at risk in production with slow response times, system crashes under load, and loss of business. This study aims to bring awareness to how important it is to have performance tests for the changes made.

2) RQ2: Why do some teams bypass performance testing?

Rationale: Software teams often overlook performance testing. Assuming that there were no changes related to performance, the quality assurance team functionally tested the software, and no performance-related issues would arise. This study highlights the most common reason for performance bottlenecks: bypassing performance testing.

3) RQ3: How do performance-related failures affect customer trust and brand reputation? Rationale: When performance issues surface, it can cause poor customer experience and unpredicted revenue loss. Ultimately, this can damage the company's reputation and reduce customer trust. This study identified some industries that were affected due to performance issues.

4) RQ4: What strategies can be implemented to evaluate business costs of performance issues? Rationale: Since this paper highlighted some of the performance issues that affected some industries, it also mentions the strategies to implement performance testing for resilient systems.

5) RQ5: What are the gaps between state-of-the-art and state-of-practice in performance testing? Rationale: This study helps identify the importance of performance testing and the impact of ignoring it, which is not found in other scholarly articles [2][3][4].

C. Limitation of the approach

Though performance testing is critical, it has some limitations. Setting up realistic test environments can be complex.

Performance testing can sometimes produce negative results if the application's configuration or capacity is not equivalent to production. Proper planning and analysis may lead to misinformation and misguided optimizations .

III. Performance Testing

As internet users are increasing, so does the load on applications. We need performance testing to maintain the applications to perform efficiently and effectively with minimum infrastructure.

A. What is performance testing?

UptimeIntelligenceIt is one of the critical processes in the Software development life cycle that ensures the system is stable, reliable, and scalable under various load conditions. This testing simulates user load from routine traffic to surviving extreme stress. This uncovers how the system truly performs, whether it's measuring response times or testing under peak load conditions; performance testing uncovers hidden bottlenecks, fine-tuning the application, and guarantees system stability to provide exceptional user experience at every turn.

B. Evolution of performance testing

In the early 1990s, as the Internet began to gain attention, performance testing was purely a manual endeavor. Testers relied on manual approaches to measure application performance. In 1991, Mercury introduced WinRunner, an automated GUI testing tool that allows users to record and replay user activities. This reduced significant reliance on manual testing.

The demand for faster applications grew as the Internet boomed, making performance testing more essential. In this momentum, Mercury developed the first performance testing tool, LoadRunner, in 1993. This tool helped testers assess application performance under heavy loads. Since then, performance testing has continuously evolved, with many tools emerging into the market. The rise of open-source load testing tools enabled organizations to execute performance testing more efficiently and cost-effectively.

Performance testing has evolved to integrate seamlessly with Agile methodology and DevOps to emulate continuous integration and deployment models. The advent of cloud platforms has further enabled performance testing to evolve to provide scalable environments. Today, integration with AI has enhanced this process to be quicker and more proactive.

C. Performance testing process

Performance testing is a structured process, and its life cycle includes various phases. Starting with nonfunctional requirements gathering, test planning, test case creation, test script creation, execution, result analysis, and dashboard generation.

1) Non-functional requirements gathering: This process begins with gathering non-functional requirements such as expected throughput, critical business transactions, response times, and anticipated resource utilization. Multiple meetings are necessary to collect these requirements. Understanding these from a business and technical perspective helps plan effective testing activities.

2) Test Plan Creation: The next step in this process is creating a test plan and test cases. A test plan is a comprehensive document summarizing all the requirements gathered in the first step of the process. Creating these test plans ensures the effective execution of performance testing. Test case documents list all the scenarios that need to be tested.

3) Test Script Creation: Tests are created using testing tools like LoadRunner, JMeter where different test scenarios are created to simulate user actions virtually and real-time load conditions.

4) Test Execution: Next is a test environment to execute performance tests in the lab. This is the crucial step, as this lab needs to be a production replica to ensure accurate test results. The execution phase starts once the test scripts are ready. This step requires active monitoring of applications with monitoring tools during the test to find bottlenecks or areas of performance improvement. This is where the test plan will effectively plan the number of tests and duration of the execution phase.

5) Results Analysis and Report generation: Once the tests are completed, all the test results are gathered, and a summary report is generated. In some cases, tests are executed again after performance improvements.

D. Key performance metrics

Key performance metrics are used to evaluate the application's performance and efficiency during performance testing. The following are some of the key performance indicators that are captured during performance testing to assess application performance and find bottlenecks to enhance the system.

1) Response times: It is the measure of time taken for a system to respond to a user request. It is calculated by averaging the response times of all the requests sent during the test. In some cases, the 90th percentile of the response times was measured. The 90th percentile response time is calculated as the average response time corresponding to the fastest 90% of the requests.

2) User load: It refers to the number of virtual users simulated with a load-testing tool to access the system under various real-time conditions. These conditions can include average load, peak load, stress load, and concurrent user load.

3) System utilization of the server: This refers to the percentage of system resources like CPU, memory, disk, and network used during the performance test. It provides information about how well the system handles the load during the test.

4) Latency: This refers to the delay between sending a request to a server and receiving a response from the server. This latency can be in the system, network, disk, or application. This metric is measured as time to first buffer, network latency, or round-trip time.

5) Error rate: It is the measure of the percentage of failed requests out of the total requests sent during the test. It is measured as transaction error rate, HTTP error codes, and Network error rate. The higher the error rate, the more unreliable the system is.

6) Page loading time: This is the metric measured for web-based applications. It measures the time taken to load the page, all the image files, DNS lookup, connection time, and server processing time during the test.

7) Page size: This is the metric used for web-based applications. It measures the size of the page, including image files, HTML, and non-HTML resources.

8) database metrics: These are the metrics obtained from the database, such as long-running queries, top SQL, deadlocks, IO, and many more.

E. Behind the scenes of performance testing

Relying on the same number of physical computers to generate hundreds or thousands of users seems impractical. Instead, load testing tools can enable this user load simulation virtually with just a few machines. Figure 1 is the architecture of a typical load testing environment where virtual users are simulated and executed performance tests like real-world scenarios.

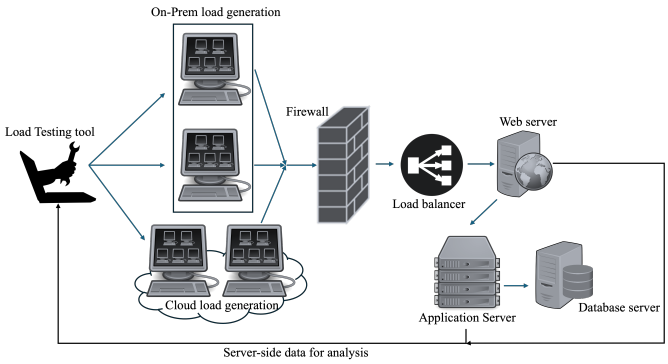

Figure 1. Architecture of a typical load testing environment

The load testing tool uses load generator resources to simulate virtual users to send traffic over the firewall to the web server or application servers like real users and receive responses and metrics for further analysis.

F. Business outcome of performance testing

Performance testing not only improves performance of the application but directly impacts business success.

- Faster response and smooth display, keep users to stay.
- Seamless application increases customer satisfaction and attracts new users.
- As user engagement increases, business growth accelerates and thus higher revenue.
- Organizations with high performance applications gain a competitive edge.
- It helps organization to identify bottlenecks early and prepare for any unplanned application failures.
- Performance testing prevents outages and failures that damage customer trust.
- It contributes to sustainability by optimizing resource utilization and energy consumption to create eco-friendly and cost effective digital solutions.

IV. Understanding IT outages: Causes, Effects and Preventive measures

As technology continues to evolve, the reliability on software is rapidly growing and organizations continue to face significant challenges in maintaining up time of the systems.

A. Causes of IT outages

A recent data from Uptime Intelligence, on average, there are 10 to 20 high-profile IT outages every year that cause serious or severe financial loss [5].

Another study from Magnita reveals that 68% of the organizations conduct performance testing, and 55% of them encounter difficulties due to the unavailability of test environments. This indicates that over half of the organizations may deploy software into production without proper testing and mainly performance testing to assess the system reliability under real-world conditions [6].

IT outages can occur from a variety of factors like hardware failures, cyberattacks, software faults, and capacity or congestion-related issues, which contribute to 22% of IT outages, according to respondents from Statista. This reveals that nearly a quarter of the IT outages occurred due to poor handling of demand, resulting in performance degradation [7].

B. Effects of IT outages

According to survey conducted by uptime, In 2022 alone, a quarter of respondents reported that their outages are costing over $1 million, while 45% reported their cost of outages are between $100,000 and $1 million. This marks a clear trend that cost of IT outages are steadily increasing, making investments in IT reliability more critical than ever [5].

According to Splunk, Global 2000 companies lose $400B annually due to application failures or slower. This includes direct financial losses from suspended operations and indirect losses like reputational damage, loosing customers [8]

In February 2017, Google released a report by analyzing over 900,000 mobile pages to assess mobile page speed performance across various industry sectors. The analysis revealed that for 70% of the pages examined, it took nearly 7 seconds for the visual content above the fold to display, and more than 10 seconds to fully load all visual content [9].

A significant portion of mobile pages were found to be excessively large, with 70% over 1MB, 36% over 2MB, and 12% exceeding 4MB. The study also indicated that as page load time increases from 1 second to 7 seconds, the probability of a mobile site visitor bouncing increases by 113%[9].

C. Preventive measures with performance testing

Addressing the causes of IT outages necessitates a need to understand the root causes of outages and the implementation of robust performance testing. The findings of effects of IT outages highlights the critical importance of performance testing in today's fast pacing digital world. Slower applications not only frustrate users but also impact organization financially. Having a proper test environment and regular performance testing helps identify root causes and reduce its effects by helping businesses enhance user engagement, reduce abandonment rates, and ultimately drive better financial outcomes.

Here are some of the strategies to adapt anticipated and unforeseen challenges that can achieved with performance testing.

- Execute performance tests early in development.
  - Steps to achieve it.
    - Have a clear expectations and success criteria such as acceptable response times.
    - Choose right test environment which is realistic and isolated from other Development, QA environments.
    - Simulate real world scenarios using performance testing tools like LoadRunner or JMeter.
    - Conduct early profiling in development stage to catch resource intensive code paths.
    - Enable continuous monitoring using application monioring tools like Dynatrace or Prometheus.
    - Perform regression testing for every code release.
  - Outcomes.
    - Identify potential performance issues before they occur in production affecting customers.
    - Reduce the chances of inefficient coding.
    - Prepare the teams for the unexpected.
- Monitor system utilization.
  - Steps to achieve it.
    - Instrument monitoring tools like Dynatrace, Grafana, Cloud Watch to monitor system performance metrics like CPU, memory, and disk utilization.
    - Enable continuous monitoring and visualize key performance metrics on dashboard for real time analysis.
  - Outcomes.
    - Detects performance anomalies and resource bottlenecks.
    - Reduces unplanned downtime through proactive monitoring.
    - Improves observability and reliability.
- Conduct different types of testing based on the load.
  - Steps to achieve it.
    - Identify load patterns in production.
    - Set performance benchmark goals based on business requirements.
    - Identify the type of test required like stress, endurance, spike, and negative-scenario tests.
    - Execute the load tests using performance testing tools.
    - Identify bottlenecks, optimize, and retest until goals are achieved.
  - Outcomes.
    - Identify weakest components.
    - Identify the causes of system crashes.
    - Identify configuration related issues that happen only under load.
- Execute Chaos testing.
  - Steps to achieve it.
    - Identify mission critical and vulnerable components.
    - Choose right tool like Gremlin to induce performance bottlenecks.
    - Plan for chaos experiments that align with real-world failure scenarios.
    - Enable continuous monitoring during the test to identify the problem pattern.
    - Fix vulnerabilities and revalidate the fixes through multiple tests.
  - Outcomes.
    - Identify hidden weaknesses in the system.
    - Ensures system remain resilient.
- Introduce disaster recovery testing.
  - Steps to achieve it.
    - Identify the mission-critical systems.
    - Choose disaster recovery test type like full interruption.
    - Create and activate the DR plan.
    - Monitor during disaster recovery, analyze the outcomes, and improve to reduce the gaps.
  - Outcomes.
    - Ensures that critical systems can be restored during crashes.
    - Create readiness during disasters.
    - Refines recovery strategies based on test results.
- Cloud auto scaling.
  - Steps to achieve it.
    - Define scaling strategy based on usage metrics like CPU > 70%.
    - Enable real-time performance metrics monitoring.
    - Configure auto scale policies in cloud platforms.
    - Test scaling up/down based on the usage and optimize thresholds.
  - Outcomes.
    - Increase uptime during traffic surges.
    - Uses infrastructure efficiently and reduces costs
    - Ensure seamless user experience during traffic surges.
- Automate testing process in continuous delivery.

- Steps to achieve it.
  - Integrate the process using automation tools to trigger tests automatically when build is triggered.
  - Trigger the tests when code deployment job is triggered.
  - Monitor the test results and improve the process based on the trends.
- Outcomes.
  - Performance degradation is detected automatically for every release.

V. Case Studies: The cost of downtime

Downtime of service unavailability can be due to maintenance or unexpected failures. Even a few minutes of downtime can lead to revenue loss and customer dissatisfaction. Here are some of the real world outages to understand the importance of robust performance testing in mitigating these outages [10].

A. Case Study 1: Azure Resource Manager exhausts capacity

Azure Resource Manager is the central tool that is used to deploy, manage and control Azure based resources.

1) Date of the incident: January 21, 2024.

2) Issue: Azure Resource Manager nodes failed on startup and more resources were consumed by the failed nodes, exhausting capacity.

3) Root cause: A configuration change gave preview access to new feature in June 2020 that has a code defect. This made nodes fail to startup.

4) Effect: Impacted downstream Azure services that rely on Azure Resource Manager to be unavailable.

5) Downtime: 7 hours.

6) Implications: A configuration change may not seem like affecting performance. In some cases, these changes can still have unexpected effects that impacts the performance which highlights the importance of performance testing in every stage of development.

7) Strategy to avoid this issue: Implementing negative scenarios as part of performance testing can help avoid these issues.

B. Case Study 2: Jira users seeing 503 service unavailable

Atlassian Jira is a tool that provides teams to plan and track work across different stages of the project.

1) Date of the incident: January 18, 2024.

2) Issue: Users of Atlassian Jira unable to track the status of their work as they saw 503 service unavailable errors.

3) Root cause: A scheduled database upgrade degraded the performance.

4) Effect: Caused an increase in back pressure which made requests to timeout.

5) Downtime: 3.5 hours.

6) Implications: Database upgrades require rigorous performance testing with higher load than expected due to potential changes in the database structure and indexing mechanisms. These changes, if not thoroughly tested can affect system performance and sometimes causes system outages.

7) Strategy to avoid this issue: Executing rigorous performance testing with all possible critical scenarios can help in avoiding these issues. This can be achieved with proper requirements gathering and test plan.

C. Case Study 3: Microsoft 365 outage

Microsoft 365 is a personal or business subscription service that provides services and apps for personal and business purposes.

1) Date of the incident: November 25, 2024.

2) Issue: Users saw 503 service unavailable errors while using Microsoft services.

3) Root cause: A change that surged number of requests being routed through servers, thereby affecting system performance.

4) Effect: Impacted processing capabilities of the infrastructure.

5) Downtime: It is not complete downtime but affected services for 7 hours.

6) Implications: A small change can lead to surge in incoming traffic, stressing the systems and causing service disruptions. This incident underscores the importance of load testing and continuous monitoring to detect bottlenecks from the traffic patterns. Although it is not complete downtime, even partial outages can significantly affect user experience.

7) Strategy to avoid this issue: Executing Spike testing as part of performance testing which replicates these sudden surge in requests can avoid these issues.

D. Case Study 4: Netflix broadcast disruptions

Netflix faced issues while broadcasting the live streaming of Jake Paul vs. Mike Tyson boxing event. Although this wasn't its first live streaming attempt, it was reported that its the most streamed event.

1) Date of the incident: December 20, 2024.

2) Issue: Netflix users reported that the service was not available a head of the live boxing event

3) Root cause: Netflix uses Open Access appliances (OCA) to store and deliver video content. These OCAs are preloaded with content during non-peak hours, while live streams happened in real time. These OCAs could not keep up with surge in traffic.

4) Effect: Received 500,000 reports that users were having problems streaming the match[11].

5) Downtime: 6 hours.

6) Implications: This disruption emphasize the importance of performance testing to ensure systems can handle peak load, quick response times and scale efficiently based on demand.

7) Strategy to avoid this issue: Measuring current production load and evaluating the performance of the system with 20% more than peak production volume.

E. Case Study 5: J. Crew website availability dropped

J. Crew is an American clothing retailer that sells clothing, shoes and accessories.

1) Date of the incident: November 23, 2018.

2) Issue: Shoppers were not able to make purchases and frequently bumped with "hang on a sec" message.

3) Root cause: Application servers couldn't keep up with the load.

4) Effect: J.Crew lost $775,000 due to unsold inventory

5) Downtime: 5 hours.

6) Implications: This case study emphasizes the importance of executing performance testing to prepare for peak-season to ensure the system is ready to handle peak load or scale based on the demand.

7) Strategy to avoid this issue: Environment setup to execute performance tests before peak season in regular intervals like holiday readiness tasks with increased load than previous year can help the systems perform the best during peak season.

VI. Conclusion and future work

In conclusion, since outages can occur anytime when we least expect, implementing thorough performance testing can help in minimizing the risk. By implementing performance testing early in the development, and helps to identify bottleneck early. Some of the outages can happen even with rigorous performance testing, the efforts mentioned in the paper can help identify the bottlenecks and solutions faster to ensure uninterrupted services to customers. Performance testing isn't just about avoiding downtime, it can ensure systems can perform flawlessly even under heavy load.

Future work will explore case studies that benefited from performance testing. More details on types of performance testing implemented in the case studies, and strategies to maintain system reliability to 99% will also be researched.

References

[1] E. Klotins, T. Gorschek, K. Sundelin, and R. Berntsson Svensson, "Towards cost-benefit evaluation for continuous software engineering activities.," *Empirical Software Engineering*, vol. 27, p. 157, 2022. DOI: 10.1007/s10664-022-10191-w.

[2] X. Han and T. Yu, "An empirical study on performance bugs for highly configurable software systems," ser. ESEM '16, New York, NY, USA: Association for Computing Machinery, 2016, ISBN: 9781450344272. DOI: 10.1145/2961111.2962602.

[3] M. R. Woodward and M. A. Hennell, "Strategic benefits of software test management: A case study," *Journal of Engineering and Technology Management*, vol. 22, no. 1, pp. 113–140, 2005, Research on Social Networks and the Organization of Research and Development, ISSN: 0923-4748. DOI: https://doi.org/10.1016/j.jengtecman.2004.11.006.

[4] S. Zaman, B. Adams, and A. E. Hassan, "A qualitative study on performance bugs," in *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*, 2012, pp. 199–208. DOI: 10.1109/MSR.2012.6224281.

[5] U. institute, "Annual outages analysis 2023," Last accessed: February, 2025, 2023, [Online]. Available: https://datacenter.uptimeinstitute.com/rs/711-RIA-145/images/AnnualOutageAnalysis2023.03092023.pdf.

[6] Magnitia, "Software testing statistics – 2023," Last accessed: February 20,2025, 2023, [Online]. Available: https://magnitia.com/blog/software-testing-statistics-2023.

[7] A. Petrosyan, "Most common root causes of it system and software-related outages worldwide," Last accessed: February, 2025, 2023, [Online]. Available: https://www.statista.com/statistics/1482105/it-system-software-related-outages-root-cause/.

[8] Splunk, "The hidden costs of downtime strike below the surface," Last accessed: February, 2025, 2024, [Online]. Available: https://www.splunk.com/en_us/campaigns/the-hidden-costs-of-downtime.html.

[9] Google, "Find out how you stack up to new industry benchmarks for mobile page speed," Last accessed: February, 2025, 2017, [Online]. Available: https://think.storage.googleapis.com/docs/mobile-page-speed-new-industry-benchmarks.pdf.

[10] C. thousand eyes - internt research team, "Internet and cloud intelligence blog," Last accessed: February, 2025, 2024, [Online]. Available: https://www.thousandeyes.com/blog/?cat=outage-analyses.

[11] J. Yoon, "Thousands report netflix livestream crashes during mike tyson-jake paul fight," Last accessed: February, 2025, 2024, [Online]. Available: https://www.nytimes.com/2024/11/16/business/media/netflix-outage-crash-boxing.html.