Approaches to Comprehensive Performance Evaluation of Software Applications: A Systematic Literature Review

Yauhen Unuchak, Tatyana Unuchak
University of Maribor, Faculty of Organizational Sciences, Kranj, Slovenia
yauhen.unuchak@student.um.si, tatyana.unuchak@student.um.si

Performance evaluation of modern software applications is necessary to optimize resources and improve the reliability of software applications, whose main problems are related to low throughput, long response times, and high support costs. Software Performance Engineering methodology provides early prediction and optimization using simulation, benchmarks and profiling. Analysis of publications revealed problems with data processing, lack of standardization of metrics and lack of effective application of analytical models, which complicate the process of evaluating the performance of information systems. Addressing these issues and standardizing approaches will help improve the processes for evaluating the performance and reliability of information systems that meet user and customer requirements.

DOI
https://doi.org/10.18690/um.fov.2.2025.75

ISBN
978-961-286-963-2

Keywords:
software performance, performance evaluation, application monitoring, performance modeling, performance metrics, performance testing, performance prediction, benchmarks

University of Maribor Press

1 Introduction

Choosing the optimal configuration of computing resources for an application becomes a non-trivial task even for experienced developers (Cunha et al., 2013). To resolve these issues, software performance analysis and management techniques are actively used to identify bottlenecks, optimize resource utilization and reduce costs (Hsu & Devetsikiotis, 2007; Litoiu & Barna, 2013).

The performance of software applications depends on many factors, including component implementation, third-party services used, deployment platform, usage profile, and competition for resources(Danciu et al., 2015). An important aspect is to consider performance metrics not only at the infrastructure level, but also at the level of algorithms, applications, third-party components and services. This allows to comply with service level agreements (SLAs) and optimize the parameters of software applications (Streitz et al., 2018). Since the early 2000s, Software Performance Engineering (SPE) methodology has become an important tool for early prediction and management of software application performance throughout the entire development life cycle (Evangelin Geetha et al., 2011). However, the results of such modeling often do not correlate with real-world performance in the long term, making regular performance evaluation and testing an important part of the development process (Tsuji et al., 2017).

2 Methodology

We have performed a systematic literature review to explore the main directions and findings of previous research in the area of software application performance. The literature review procedure followed the basic steps of the Prisma 2020 methodology(Page et al., 2021).

To select appropriate keywords, we first defined a research question by clarifying the boundaries of our survey. The question was defined as follows: **RQ**: What metrics and monitoring techniques are used to evaluate the performance of software applications? The constraints were specified as:

- publication Years: 1990–2025,
- language: English,

- document Types/Source type: article, proceedings paper, review, early access, book chapter, editorial material, book review, book, discussion,
- categories/subject area: computer science theory methods, computer science information systems, computer science software engineering, computer science artificial intelligence, computer science interdisciplinary applications.

The search was performed using 34 combinations of search queries. Examples of search query combinations are as follows: "performance evaluation" AND "software", "performance evaluation" AND "application", "software performance" AND "evaluation", "performance assessment" AND "software systems".
The results of the initial search without cleaning are shown in Table 1.

Table 1: Literature search results

Web of Science	Scopus	ProQuest Dissertations & Theses	Total
works	works	works	works
9667	34619	1688	45974

The obtained results were merged into one data source, where duplicates were removed for records having the same DOI and title. In the next step, publications were selected where at least one of the key phrase combinations among the title or abstract was found: "performance assessment", "performance model", "performance evaluation", "performance testing", "performance measurement", "performance metrics". The final data source was reduced to 339 records. The primary manual analysis of the abstracts resulted in the removal of an additional 176 records (e.g., addressing hardware and network equipment performance issues). The text of 23 publications could not be accessed. The text of the remaining 134 publications was downloaded and analyzed.

3 Results

In the early 2000s, the process of evaluating the performance of software applications was conducted in three steps. First, components were described using Unified Modeling Language (UML) or Architecture Description Language (ADL) diagrams, then the project was transformed into an analytical model such as queueing networks (QN) or Stochastic Rendezvous Network (SRN) model. In the

final stage, the experimental results were used to refine the design (Yuan et al., 2006). Further development of techniques such as benchmark-based testing, profiling and prototyping have greatly simplified and accelerated the performance evaluation process (Hsu & Devetsikiotis, 2007).

3.1 Benchmarks

Benchmarks are a key tool for evaluating the performance of software applications and configurations. Testing is performed in isolated environments to accurately analyze resources including CPU, memory, and networking (Chhetri et al., 2014; Peng et al., 2004; Podzimek & Chen, 2013). For example, the IBM Trade Performance Benchmark tool is designed to analyze all layers of enterprise application architecture, including client, server, and storage layers (Dube et al., 2007). In high performance computing (HPC), NASA uses specialized HPC benchmarks that enable deep analysis of systems (Mehrotra et al., 2012). Microbenchmark approaches evaluate individual components such as CPU and memory, while Macrobenchmarks measure system performance in real-world scenarios (Scheuner, 2022).

3.2 Performance monitoring of software applications

Real application monitoring is an approach to performance evaluation based on analyzing system behavior through real or simplified versions of applications (Tsuji et al., 2021). Applications are instrumented to collect data on parameters such as resource utilization, task execution time, and access to shared resources, allowing workloads to be modeled for the target platform (Ittershagen et al., 2015). Basic monitoring techniques include tracing, logging, and the use of specialized tools that help collect data on function calls and other aspects of system behavior (Becker et al., 2008; Guo et al., 2015; Meyer et al., 2020; Saastamoinen & Kreku, 2011; Yao et al., 2018). Certain solutions use user agents to collect data, followed by processing and visualization of the results (Willnecker et al., 2015).

3.3 Key performance indicators

Approaches to software application performance evaluation cover a wide range of metrics and techniques. Taking into account the interests of various stakeholders, including end users, testers, and developers, requires the selection of appropriate

techniques and metrics for performance evaluation. Users emphasize response time, while testers focus on throughput, reliability, maximum number of users, and fault tolerance (Devaraj et al., 2008; Guan et al., 2019; Li et al., 2019). Key metrics such as average response time and throughput are complemented by specific metrics such as Equivalent Instruction Count (EIC) and Normalized Equivalent Instruction Count (NEIC), which take into account "wait time", CPU, memory and network usage (Meyer et al., 2020; Rupnow et al., 2010; Wang & Ying, 2018; Weyuker & Vokolos, 2001). Metrics of successful/failed requests and task timing characteristics are important for analyzing systems under load to improve scaling algorithms (Cholomskis et al., 2018).

3.4 Server hardware data collection tools

Software application monitoring technologies have evolved from simple profiling tools to sophisticated solutions. The first tools, such as prof and gprof, captured function timestamps and became a standard for UNIX systems (Malony, 1990). Application performance management (APM) tools such as Dynatrace and AppDynamics monitor response times, resource usage and failure rates, with centralized data storage for analysis (Rabl et al., 2012). Cloud providers, including Amazon CloudWatch, provide CPU and memory utilization data (Podzimek & Chen, 2013), while the Ganglia and Nagios tools provide online access to server performance information (Yao et al., 2018).

3.5 Building performance models

Building performance models is a key analysis step that allows us to predict and evaluate the behavior of systems early in the design process. Architectural models are used to plan capacity and automate resource management by considering metrics such as resource utilization, response time, and throughput (Eismann et al., 2018). Advanced approaches include analytical modeling, measurement, and simulation, where analytical models use mathematical expressions, measurements are applied to existing systems, and simulations create models for preliminary evaluation (Evangelin Geetha et al., 2011; Huang, 2004; Rupnow et al., 2010).

Automatic performance model generation from UML diagrams enables the use of Layered Queueing Networks (LQNs) and Petri nets for quantitative analysis (Campos & Merseguer, 2006; D'Ambrogio & Iazeolla, 2005; Gómez-Martínez & Merseguer, 2006; Pham & Nguyen, 2013; Tang et al., 2018). Markov models, which describe probabilistic transitions between system states, integrate with UML and stochastic process algebra, simplifying the prediction and analysis of complex systems (Sbeity & Dbouk, 2014; Tribastone & Gilmore, 2008).

Alternative approaches to software performance analysis and optimization cover UML diagram annotation, probabilistic methods, and specialized optimization tools. For example, the Software Performance MDA Framework methodology automates the transformation of UML diagrams into LQN models by extending the analysis of non-functional attributes (Di Marco & Mirandola, 2006). Other approaches, such as the AOP-based Performance Evaluation Framework, simplify the verification of performance requirements through aspects, and Response Surface Methodology optimizes the parameters of simulated applications in a cost-aware manner (Hsu & Devetsikiotis, 2007). Using Software Performance Curves and integrating workflows with UML through Directed Acyclic Graphs strengthens the connection between the design and analysis of software applications (Westermann & Momm, 2010; Zhang et al., 2011). Using UML-MARTE with LQN transformation and Genetic Search Algorithm allows to find optimal configurations of modeled applications (Amoozegar & Nezamabadi-pour, 2012). Polynomial Chaos Expansion methodology emphasize uncertainty accounting and self-adaptive performance (Aleti et al., 2018; Incerto et al., 2017). Another area is represented by analyzing dependencies between input parameters and performance characteristics using statistical models and machine learning (Aleti et al., 2018; Buneci, 2008; Happe et al., 2009; Moghadam, 2022; Yao et al., 2018).

3.6 Comprehensive application performance evaluation

Integrated performance evaluation of software applications combines application, infrastructure, and user interaction data to provide a comprehensive analysis. This approach, which began by describing systems as vector characteristics of basic operations such as I/O and processor commands (Xiaolong, 2001), has evolved with the introduction of the Apdex index, which converts response time data into a measure of user satisfaction (Chhetri et al., 2014). Methods such as Statistical

Learning Theory evaluate the average response time and its deviations to identify problems (Wang & Ying, 2018), while analyzing the optimal and maximum number of users determines the stability and reliability of the system (Li et al., 2019). Data integration, including response time, throughput, CPU and memory usage, not only allows you to optimize resources, but also to predict system behavior under changing loads (Tsuji et al., 2021).

Discussion

The results of this research emphasize the importance of comprehensive methodologies for evaluating the performance of software applications operating in complex ecosystems. A review of existing literature reveals persistent challenges in standardizing performance metrics, optimizing resource utilization, and developing effective modeling techniques. Addressing these challenges is critical to ensure the reliability and efficiency of software applications that play a key role in modern organizational processes and user satisfaction. The use of SPE methodology has proven to be effective in the early stages of performance prediction and optimization.

Benchmarking continues to play a key role in evaluating infrastructure configurations, providing valuable data on the capabilities of systems under different load conditions. However, there is a need to improve benchmarking practices by integrating micro- and macro-level analyses for more detailed and comprehensive assessments.

Modern software application monitoring tools provide accurate data collection, but challenges remain in aggregating and interpreting data from various system components to derive actionable insights. The use of user-friendly visualization and analytics platforms based on artificial intelligence can greatly enhance the utility of monitoring solutions.

The research emphasizes the importance of selecting appropriate performance metrics in line with stakeholder goals. To bring these perspectives together, composite performance indicators that integrate user experience metrics with infrastructure metrics need to be developed.

The use of statistical and machine learning techniques shows potential in improving prediction accuracy while reducing reliance on big data.

Integrated performance measurement systems that combine application, infrastructure, and user interaction data open the door to comprehensive systems analysis.

Conclusion

This research confirms the key role of performance evaluation in modern software development. Analysis of literature and practices demonstrates the need to standardize metrics, integrate monitoring data, and use modern models for performance analysis.

The future of software performance evaluation research involves the development of automation of monitoring processes, data collection and analysis, standardization of metrics and methodologies to unify approaches, and the use of hybrid modeling approaches that combine analytical and simulation techniques. These directions will ensure the creation of more reliable, scalable and efficient software solutions that meet user expectations and business requirements.

References

Aleti, A., Trubiani, C., Van Hoorn, A., & Jamshidi, P. (2018). An efficient method for uncertainty propagation in robust software performance estimation. Journal of Systems and Software, 138, 222–235. https://doi.org/10.1016/j.jss.2018.01.010

Amoozegar, M., & Nezamabadi-pour, H. (2012). Software performance optimization based on constrained GSA. In AISP 2012—16th CSI International Symposium on Artificial Intelligence and Signal Processing (p. 139). https://doi.org/10.1109/AISP.2012.6313732

Becker, D., Frings, W., & Wolf, F. (2008). Performance Evaluation and Optimization of Parallel Grid Computing Applications. 16th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP 2008), 193–199. https://doi.org/10.1109/PDP.2008.27

Buneci, E. (2008). QUALITATIVE PERFORMANCE ANALYSIS FOR LARGE-SCALE SCIENTIFIC WORKFLOWS.

Campos, J., & Merseguer, J. (2006). On the Integration of UML and Petri Nets in Software Development (p. 36). https://doi.org/10.1007/11767589_2

Chhetri, M. B., Chichin, S., Vo, Q. B., & Kowalczyk, R. (2014). Smart CloudMonitor—Providing Visibility into Performance of Black-Box Clouds. 2014 IEEE 7th International Conference on Cloud Computing, 777–784. https://doi.org/10.1109/CLOUD.2014.108

Cholomskis, A., Pozdniakova, O., & Mažeika, D. (2018). Cloud Software Performance Metrics Collection and Aggregation for Auto-Scaling Module. In R. Damaševičius & G. Vasiljevienė (Eds.), Information and Software Technologies (Vol. 920, pp. 130–138). Springer International Publishing. https://doi.org/10.1007/978-3-319-99972-2_10

Cunha, M., Mendonça, N., & Sampaio, A. (2013). A Declarative Environment for Automatic Performance Evaluation in IaaS Clouds. https://doi.org/10.1109/CLOUD.2013.12

D'Ambrogio, A., & Iazeolla, G. (2005). Metadata-driven design of integrated environments for software performance validation. Journal of Systems and Software, 76(2), 127–146. https://doi.org/10.1016/j.jss.2004.04.014

Danciu, A., Chrusciel, A., Brunnert, A., & Krcmar, H. (2015). Performance Awareness in Java EE Development Environments. In M. Beltrán, W. Knottenbelt, & J. Bradley (Eds.), Computer Performance Engineering (Vol. 9272, pp. 146–160). Springer International Publishing. https://doi.org/10.1007/978-3-319-23267-6_10

Devaraj, E. G., Skumar, tv, & Kanth, K. (2008). Predicting performance of software systems during feasibility study of software project management (p. 5). https://doi.org/10.1109/ICICS.2007.4449845

Di Marco, A., & Mirandola, R. (2006). Model Transformation in Software Performance Engineering. In C. Hofmeister, I. Crnkovic, & R. Reussner (Eds.), Quality of Software Architectures (Vol. 4214, pp. 95–110). Springer Berlin Heidelberg. https://doi.org/10.1007/11921998_11

Dube, P., Yu, H., Zhang, L., & Moreira, J. E. (2007). Performance Evaluation of a Commercial Application, Trade, in Scale-out Environments. 2007 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 252–259. https://doi.org/10.1109/MASCOTS.2007.51

Eismann, S., Walter, J., von Kistowski, J., & Kounev, S. (2018). Modeling of Parametric Dependencies for Performance Prediction of Component-Based Software Systems at Run-Time. https://doi.org/10.1109/ICSA.2018.00023

Evangelin Geetha, D., Suresh Kumar, T. V., & Rajani Kanth, K. (2011). Predicting the software performance during feasibility study. IET Software, 5(2), 201–215. https://doi.org/10.1049/iet-sen.2010.0075

Gómez-Martínez, E., & Merseguer, J. (2006). ArgoSPE: Model-Based Software Performance Engineering (p. 410). https://doi.org/10.1007/11767589_23

Guan, X., Ma, Y., Shao, Z., & Cao, W. (2019). Design and Implementation of Mobile Application Performance Test Scheme Based on LoadRunner. 2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC), 90–93. https://doi.org/10.1109/ICEIEC.2019.8784620

Guo, J., Zhou, Z., & Zhang, H. (2015). Cocktail: A New Tool for Audit by Performance Evaluation on Local Machine. 2015 IEEE Symposium on Service-Oriented System Engineering, 241–246. https://doi.org/10.1109/SOSE.2015.40

Happe, J., Li, H., & Theilmann, W. (2009). Black-box performance models: Prediction based on observation. Proceedings of the 1st International Workshop on Quality of Service-Oriented Software Systems, 19–24. https://doi.org/10.1145/1596473.1596479

Hsu, C.-C., & Devetsikiotis, M. (2007). An Automatic Framework for Efficient Software Performance Evaluation and Optimization. 40th Annual Simulation Symposium (ANSS'07), 99–105. https://doi.org/10.1109/ANSS.2007.12

Huang, T. (2004). Performance analysis of Web services-based systems with sensitivity analysis.

Incerto, E., Tribastone, M., & Trubiani, C. (2017). Software performance self-adaptation through efficient model predictive control. 2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE), 485–496. https://doi.org/10.1109/ASE.2017.8115660

Ittershagen, P., Hartmann, P. A., Grüttner, K., & Nebel, W. (2015). A workload extraction framework for software performance model generation. Proceedings of the 2015 Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools, 1–6. https://doi.org/10.1145/2693433.2693436

Li, H., Li, X., Wang, H., Zhang, J., & Jiang, Z. (2019). Research on Cloud Performance Testing Model (p. 183). https://doi.org/10.1109/HASE.2019.00035

Litoiu, M., & Barna, C. (2013). A performance evaluation framework for Web applications. Journal of Software: Evolution and Process, 25(8), 871–890. https://doi.org/10.1002/smr.1563

Malony, A. D. (1990). Performance observability.

Mehrotra, P., Djomehri, J., Heistand, S., Hood, R., Jin, H., Lazanoff, A., Saini, S., & Biswas, R. (2012). Performance evaluation of amazon EC2 for NASA HPC applications. ScienceCloud '12 - 3rd Workshop on Scientific Cloud Computing. https://doi.org/10.1145/2287036.2287045

Meyer, H., Odyurt, U., Pimentel, A. D., Paradas, E., & Alonso, I. G. (2020). An analytics-based method for performance anomaly classification in cyber-physical systems. Proceedings of the 35th Annual ACM Symposium on Applied Computing, 210–217. https://doi.org/10.1145/3341105.3373851

Moghadam, M. H. (2022). Intelligence-Driven Software Performance Assurance.

Page, M., Moher, D., Bossuyt, P., Boutron, I., Hoffmann, T., Mulrow, C., Shamseer, L., Tetzlaff, J., Akl, E., Brennan, S., Chou, R., Glanville, J., Grimshaw, J., Hróbjartsson, A., Lalu, M., Li, T., Loder, E., Mayo-Wilson, E., Mcdonald, S., & Mckenzie, J. (2021). PRISMA 2020 explanation and elaboration: Updated guidance and exemplars for reporting systematic reviews. BMJ, 372, n160. https://doi.org/10.1136/bmj.n160

Peng, L., See, S., Jiang, Y., Song, J., Stoelwinder, A., & Neo, H. (2004). Performance evaluation in computational grid environments (p. 62). https://doi.org/10.1109/HPCASIA.2004.1324016

Pham, H. V., & Nguyen, B. N. (2013). Class diagram based evaluation of software performance (Z. Zhu, Ed.; p. 876870). https://doi.org/10.1117/12.2008322

Podzimek, A., & Chen, L. Y. (2013). Transforming System Load to Throughput for Consolidated Applications. 2013 IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems, 288–292. https://doi.org/10.1109/MASCOTS.2013.37

Rabl, T., Sadoghi, M., Jacobsen, H., Gómez-Villamor, S., Muntés-Mulero, V., & Mankowskii, S. (2012). Solving Big Data Challenges for Enterprise Application Performance Management. Proc VLDB Endowment, 5. https://doi.org/10.14778/2367502.2367512

Rupnow, K., Adriaens, J., Fu, W., & Compton, K. (2010). Accurately evaluating application performance in simulated hybrid multi-tasking systems (p. 144). https://doi.org/10.1145/1723112.1723136

Saastamoinen, J., & Kreku, J. (2011). Application workload model generation methodologies for system-level design exploration. Proceedings of the 2011 Conference on Design & Architectures for Signal & Image Processing (DASIP), 1–7. https://doi.org/10.1109/DASIP.2011.6136888

Sbeity, I., & Dbouk, M. (2014). Software performance engineering using UML2SAN: Deadlock prediction of funds transfer. 2014 9th International Conference on Computer Engineering & Systems (ICCES), 318–323. https://doi.org/10.1109/ICCES.2014.7030978

Scheuner, J. (2022). Performance Evaluation of Serverless Applications and Infrastructures.

Streitz, A., Barnert, M., Kienegger, H., & Krcmar, H. (2018). Performance Improvement Barriers for SAP Enterprise Applications: An Analysis of Expert Interviews (p. 228). https://doi.org/10.1145/3184407.3184434

Tang, X., Wang, Z., Li, X., Han, Z., He, Z., & Fu, Y. (2018). Performance analysis for multimedia communication systems with a multilayer queuing network model. China Communications, 15(8), 67–76. https://doi.org/10.1109/CC.2018.8438274

Tribastone, M., & Gilmore, S. (2008). Automatic Extraction of PEPA Performance Models from UML Activity Diagrams Annotated with the MARTE Profile. In WOSP'08: Proceedings of the 7th International Workshop on Software and Performance 2008 (p. 78). https://doi.org/10.1145/1383559.1383569

Tsuji, M., Kramer, W. T. C., & Sato, M. (2017). A Performance Projection of Mini-Applications onto Benchmarks Toward the Performance Projection of Real-Applications. 2017 IEEE International Conference on Cluster Computing (CLUSTER), 826–833. https://doi.org/10.1109/CLUSTER.2017.123

Tsuji, M., Kramer, W., Weill, J.-C., Nominé, J.-P., & Sato, M. (2021). A new sustained system performance metric for scientific performance evaluation. The Journal of Supercomputing, 77, 1–29. https://doi.org/10.1007/s11227-020-03545-y

Wang, R., & Ying, S. (2018). SaaS software performance issue identification using HMRF-MAP framework. Software: Practice and Experience, 48. https://doi.org/10.1002/spe.2607

Westermann, D., & Momm, C. (2010). Using software performance curves for dependable and cost-efficient service hosting. Proceedings of the 2nd International Workshop on the Quality of Service-Oriented Software Systems, 1–6. https://doi.org/10.1145/1858263.1858267

Weyuker, E., & Vokolos, F. (2001). Experience with performance testing of software systems: Issues, an approach, and case study. Software Engineering, IEEE Transactions On, 26, 1147–1156. https://doi.org/10.1109/32.888628

Willnecker, F., Brunnert, A., Gottesheim, W., & Krcmar, H. (2015). Using dynaTrace Monitoring Data for Generating Performance Models of Java EE Applications. In ICPE 2015—Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering. https://doi.org/10.1145/2668930.2688061

Xiaolong, Z. (2001). Application-specific_benchmark.

Yao, K., B. de Pádua, G., Shang, W., Sporea, S., Toma, A., & Sajedi, S. (2018). Log4Perf: Suggesting Logging Locations for Web-based Systems' Performance Monitoring (p. 138). https://doi.org/10.1145/3184407.3184416

Yuan, X., Duan, S., & Liu, Z. (2006). Exploring robust component-based software. Proceedings of the 2006 International Workshop on Software Quality, 75–80. https://doi.org/10.1145/1137702.1137717

Zhang, Z. H., Fei, T., & Chai, X. D. (2011). A framework for parallel simulation application performance evaluation and optimization. 2011 International Conference on Multimedia Technology, 5692–5695. https://doi.org/10.1109/ICMT.2011.6001687

About the authors

Yauhen Unuchak is a Master of Science in IT organisation and an expert in automated testing, software performance testing and machine learning. His research interests include automated testing, quality management systems, corporate information systems, big data and predictive analytics. He is one of the authors of the book "IT-Startup: 10 Tips for Beginners".

Tatyana Unuchak is a Master of Science in IT organisation and an expert in the field of web and mobile applications development. Her research interests are related to machine learning and project management process improvement.