LLM-Assisted Microservice Performance Modeling

Maximilian Hummel, Nathan Hagel,
Minakshi Kaushik, Jan Keim, Erik Burger
Karlsruhe Institute of Technology, Germany
{firstname.lastname}@kit.edu

Heiko Koziolek
ABB Corporate Research Center, Germany
Heiko.Koziolek@de.abb.com

Abstract

Performance models guide resource-aware design and analysis of microservices, for example, through resource demands in Palladio Service Effect Specifications. Existing derivation methods for microservice performance models rely either on extensive measurements or on manual expert estimates. We present a workflow that leverages large language models (LLMs) to synthesize parametric performance models from textual artifacts such as source code or documentation, as well as from structured user input. We evaluate the approach on an OPC Unified Architecture (UA) server microservice, comparing LLM-generated performance models against measurement baselines on fixed hardware. LLM-generated performance models capture CPU core utilization as a linear function, consistent with measurements. Although not entirely accurate, these performance models provide first-order approximations for early capacity planning and design decisions.

Index Terms— Microservices, Performance Modeling, Large Language Models, Palladio, OPC UA

1 Introduction

Modern software systems are increasingly built as microservices. While this architecture enables scalability and independent development and deployment, it also complicates reasoning about performance. A single service's resource demand on CPU, memory, or network can become a bottleneck that scales nonlinearly under load and strains entire deployments. Anticipating such behavior early is important to avoid costly redesigns and overprovisioning. Performance models offer a way forward: they abstract how services consume resources under workload variation, allowing for early prediction of scalability and capacity. In frameworks such as the Palladio Component Model, resource demands are defined for Service Effect Specifications (SEFFs), making performance an explicit part of architectural design [2]. However, deriving such models remains challenging. Measurement-based approaches need executable prototypes and realistic testbeds. Expert estimates must account for multiple dimensions like realistic usage profiles, external dependencies and deployment [1].

LLMs open a new path. They can extract performance-relevant knowledge from artifacts such as code, documentation, or structured questionnaires. Prior efforts have used LLMs to predict execution times [8] or infer dependency graphs [7], but these typically lack parameterized performance models. Parameterized performance models enable what-if analysis (e.g., varying workload size or payload characteristics). They can also be used in Palladio as resource demands for SEFFs.

This paper presents a workflow that leverages LLMs to synthesize parametric performance models for microservices. Our approach combines artifact-based prefill of a performance questionnaire with user-guided completion, producing interpretable performance equations for resource demands. We evaluate the method on an OPC UA [5] server microservice, comparing LLM-generated models against measurement baselines, and demonstrate that even without measurements, the models capture realistic trends in CPU core utilization. As a result, LLMs can provide first-order approximations for early capacity planning and design support.

2 Related Work

Performance modeling of microservices has been addressed through different techniques. Measurement-based approaches derive models from controlled experiments. For example, Jindal et al. [4] introduced MicroService Capacity (MSC) in the Terminus tool, using sandboxed services, load tests, and regression models to estimate capacity. While accurate, such methods require running systems and extensive measurements.

Model-driven techniques aim to provide earlier insights. Pinciroli et al. [6] analyzed how microservice design patterns affect performance, underscoring the value of architectural abstractions. However, parameterizing these models remains challenging in dynamic, heterogeneous environments.

Recently, LLMs have been explored for automating performance analysis. Nguyen-Nhat et al. [8] proposed LLMPerf, which augments LLMs with regression heads to predict execution times from artifacts,

but without interpretable models. Zhang et al. [9] applied LLMs to estimate the performance of embedded software on RISC-V, while Hu et al. [7] introduced LLM4MDG to generate dependency graphs from documentation. These works demonstrate that LLMs can extract performance-relevant properties; however, they typically focus on raw numerical predictions or structural dependencies.

In contrast, our approach synthesizes parametric, resource-centric performance models that remain interpretable and structurally aligned with established measurement-based baselines.

3 Methodology

We present an LLM-based workflow to derive resource-centric performance models for a single microservice. The concept applies to CPU, memory, and network. The resulting performance models could be used, e.g., to define resource demands for SEFFs in Palladio.

The method is based on a structured performance questionnaire. If a textual artifact (e.g., code, documentation) is available, an LLM extracts candidate answers. Otherwise, the user fills it manually. The questionnaire covers: (1) service and functional scope, (2) deployment and configuration, and (3) workload characteristics for the particular microservice.

Workflow We propose a two-step workflow to generate performance models (see Figure 1): (1) Prefill: The LLM extracts candidate answers from artifacts, leaving unknowns blank; (2) Completion and synthesis: The user finalizes missing inputs for the questionnaire, and the LLM generates a parametric performance model. A shared state maintains artifact content, questionnaire answers, and resource selection across steps.

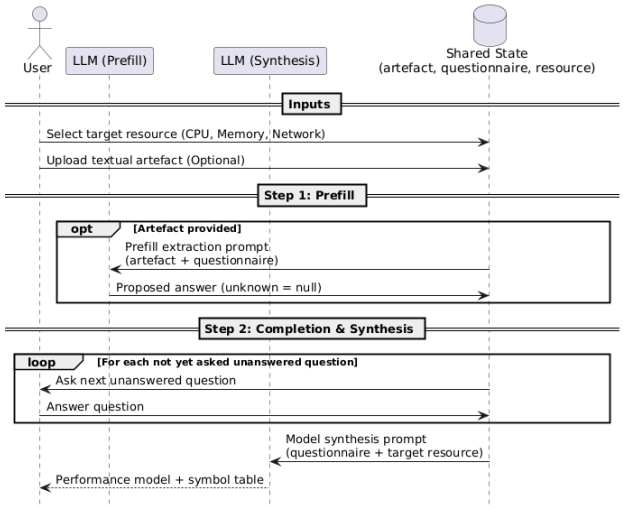

Figure 1: LLM-based workflow for deriving resource-centric performance models for a microservice.

Prompting Setup We use two prompt templates: (1) Prefill, which is provided an artifact and unanswered questions to elicit answers for those questions. (2) Synthesis, which is provided a completed questionnaire to elicit a performance model with a symbol table, assumptions and rationale. The synthesis prompt assigns the LLM the role of an experienced performance engineer.

Performance Model The output of the workflow is a resource-specific (CPU, memory, network) performance model plus a symbol table for transparency. For example, CPU core utilization could be modeled as

$$\text{CPU Core Util.\%} = \alpha_0 \cdot \text{req\_rate} + \alpha_1 \cdot \text{payload\_size} + \alpha_2,$$

where the coefficients are estimated by the LLM using the questionnaire as input. *req_rate* is requests per second, and *payload_size* is bytes per request.

4 Evaluation

We evaluate the presented approach by comparing LLM-generated performance models against measurement data for an OPC UA server microservice. The goal is to assess how well LLMs can estimate resource demands without measurements.

Use Case OPC UA is an industrial machine-to-machine communication protocol designed for interoperability, security, and scalability. It is widely used in automation and IoT environments, often on constrained hardware. Performance modeling of OPC UA is therefore relevant for guiding hardware sizing and avoiding overprovisioning or failures [3]. In our evaluation, we target CPU core utilization of an OPC UA server as the performance metric.

Setup and Data The measurements were conducted on an Intel Core Ultra 7 165H with 32 GB RAM and Windows 11. We used the Python OPC UA library (version 0.98.13). The server updated signals every 0.5 s, while one client read a varying number of signals (100, 1000, 3000, 5000, 10000) at the same rate. CPU core utilization of the server process was measured over fixed durations. The OPC UA server process can utilize multiple OS threads across several cores. We report CPU core utilization as the sum over logical cores (100% per core). The complete measurement setup, results, source code, and all prompt templates used in this study are openly available[1].

The resulting measurements were compared against models generated by the proposed LLM-based approach. GPT-5 was used to prefill the performance questionnaire and to synthesize the performance models. We used source code, documentation and nothing as artifact to prefill the questionnaire. Questions not

[1] https://doi.org/10.5281/zenodo.17310391

answered by the prefilling step are answered through a predefined manually filled questionnaire for the OPC UA use case. The questionnaire was manually completed by a master-level computer scientist working in industrial research on OPC UA, ensuring domain familiarity and practical performance knowledge. We compared the measured mean and maximum CPU core utilization % in the steady state to the LLM-generated performance models.

We repeated the evaluation with another LLM (GPT-4.1) and different numbers of OPC UA clients (1, 2, 5, 10), and observed results consistent with those reported.

4.1 Results

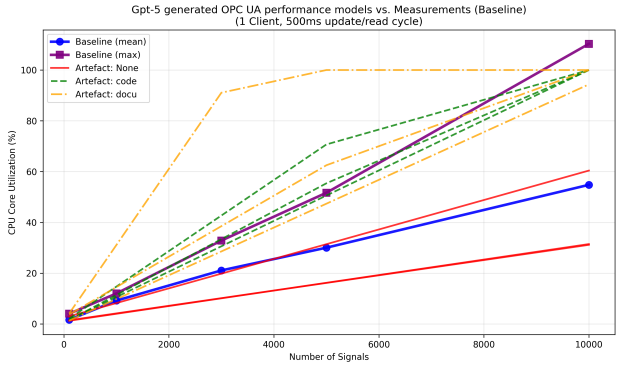

Figure 2: Comparison of LLM-generated performance models with measured mean and maximum baseline CPU core utilization of the OPC UA server as the number of signals increases.

Figure 2 shows the comparison between the generated performance models and the measured baseline. Nine performance models were generated with three artifact types: None (manually answering questionnaire), documentation, and code, with three generated performance model instances each.

The models generally reproduce the increasing trend of CPU core utilization with the number of signals. However, accuracy differs by artifact type. Performance models generated without artifacts consistently underestimate the maximum CPU core utilization. Models based on code follow the slope more closely and approximate the maximum values better. Documentation and code artifacts both yield models that are closer to the measured maximum, while models generated without artifacts align more with the measured mean utilization.

Overall, the results show that LLM-generated models can provide a reasonable first estimation of performance behavior. The choice of artifact influences quality, with code artifacts producing the most realistic approximations. While such models cannot replace measurements, they can serve as an early indication of expected system performance.

5 Conclusion

We presented an LLM-based workflow for deriving microservice performance models and demonstrated its feasibility on an OPC UA server use case. While the results indicate that LLMs can provide reasonable first-order approximations, the evaluation is limited to a single microservice, hardware platform, and resource type. Future work will extend the study to a broader range of microservices and resources to assess generality. To increase accuracy, we plan to integrate the LLM with deployment, load generation, and measurement tools so that empirical data can be used to fit model coefficients and produce more precise performance predictions.

References

[1] H. Koziolek. “Performance evaluation of component-based software systems: A survey”. In: *Performance evaluation* 67.8 (2010), pp. 634–658.

[2] R. H. Reussner et al. *Modeling and simulating software architectures: The Palladio approach*. MIT Press, 2016.

[3] A. Burger et al. “Bottleneck Identification and Performance Modeling of OPC UA Communication Models”. In: *Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering*. ICPE ’19. Mumbai, India: Association for Computing Machinery, 2019, pp. 231–242.

[4] A. Jindal, V. Podolskiy, and M. Gerndt. “Performance Modeling for Cloud Microservice Applications”. In: *Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering*. ICPE ’19. Mumbai, India: Association for Computing Machinery, 2019, pp. 25–32.

[5] OPC Foundation. *OPC Unified Architecture – Part 1: Overview and Concepts*. Specification Part 1, Release 1.05. OPC Foundation, 2021.

[6] R. Pinciroli, A. Aleti, and C. Trubiani. “Performance Modeling and Analysis of Design Patterns for Microservice Systems”. In: *2023 IEEE 20th International Conference on Software Architecture (ICSA)*. 2023, pp. 35–46.

[7] J. Hu et al. “LLM4MDG: Leveraging Large Language Model to Construct Microservices Dependency Graph”. In: *2024 IEEE 23rd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. 2024, pp. 859–869.

[8] M.-K. Nguyen-Nhat et al. “LLMPerf: GPU Performance Modeling meets Large Language Models”. In: *2024 32nd International Conference on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*. 2024, pp. 1–8.

[9] W. Zhang, M. Hassan, and R. Drechsler. “LLM-assisted Performance Estimation of Embedded Software on RISC-V Processors”. In: *2025 IEEE 28th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*. 2025, pp. 7–12.