Time Series Analysis of Server Performance Simulation for Large Language Model Deployment

Master Thesis of Kevin Bernardo

Supervisors: Prof. Dr. Markus Döhring[1], Prof. Dr. Sebastian Döhler[2]

h_da HOCHSCHULE DARMSTADT UNIVERSITY OF APPLIED SCIENCES

Introduction

- In Germany, there are demands to host large language models (LLMs) on internal servers by banks to satisfy the related **data protection requirements on confidential information**.
- For deploying an LLM on an internal server, **monitoring** the system and **recognizing** if a **high rate of request failures** occurs are important to prevent high server failure rates, which can be done by **forecasting** with a machine learning model.
- Before bringing the system into production, it is important to test whether the **time series features** generated from the LLM server are relevant for training the model for forecasting and find some insights from it.
- This study used the **BurstGPT** [1] simulation that reflects a real-world situation to test whether the features extracted from an LLM deployment server are relevant for LLM server performance forecasting using **XGBoost** [2] and find the key features for the forecasting process using **SHAP** [3].

Methodology

The experiment was based on an example use case of a **chatbot project for English conversations**, with **OASST2** [4] as the chosen prompt dataset. The project used the **vLLM** [5] serving framework to run the **Mistral-7B-Instruct-v0.2** [6] model and **Prometheus** [7] for server monitoring. The experiment was divided into two main parts: **simulation** and **forecasting**.

- *Simulation:* First, the **training, validation, and test** dataset was created using **BurstGPT** [1] method to simulate burst situations of requests sent to the server, as shown by the **workload generator** component in Figure 1. Next, the generated requests were used for the **simulation run**. Finally, the **time series data** regarding the **requests** and **server performance metrics** were extracted from the **LLM inference server** after the simulation run was finished.

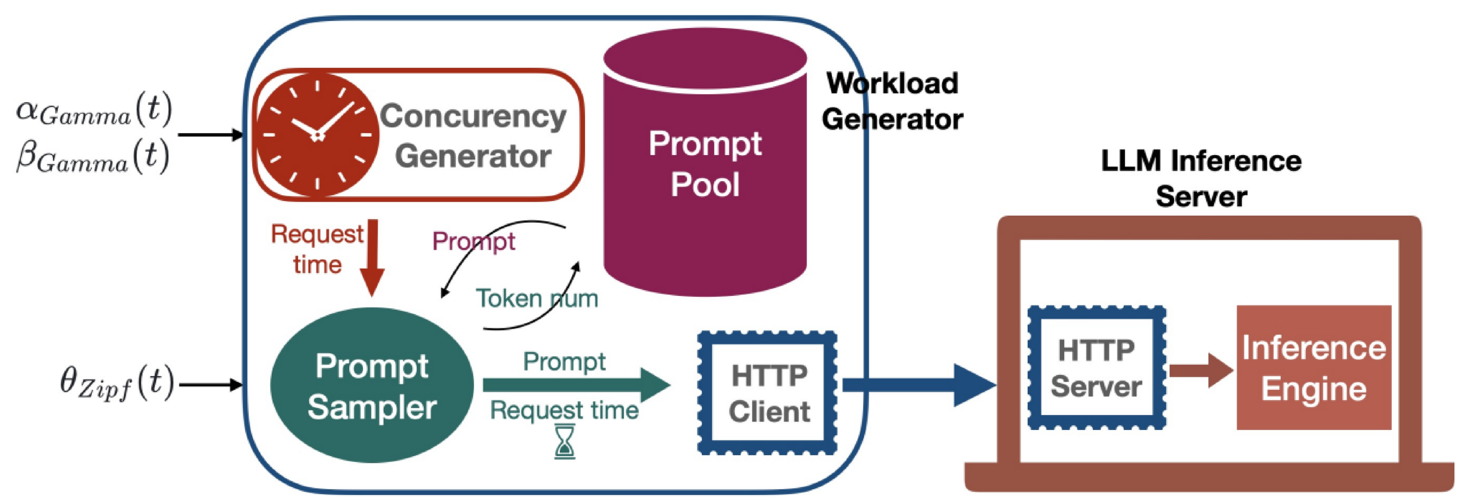

Figure 1. Diagram of simulation with BurstGPT [1]. The **concurrency generator** created random times to send the request to the LLM server based on **Gamma distribution**. The **prompt pool** contained all possible prompts for prompt sampling. The **prompt sampler** sampled the prompts from the prompt pool based on **Zipf distribution** of the prompt's token length. The generated requests were sent to the **LLM inference server** for testing.

- *Forecasting:* Both **time series data** from the simulation were aggregated into **regular intervals** of **one minute** and combined. The target variable for the forecasting was the **server's failure rate of processing requests at specific time intervals**. The stationarity of the time series data was tested with the **augmented Dickey-Fuller test**. An **XGBoost** [2] model with early stopping was compared with **ARMA**, **VAR**, and **univariate XGBoost** model for forecasting. The performance was determined based on **MAE** and **RMSE** metrics on the test data with **rolling and expanding evaluation setups** [8]. **RMSE** was specifically used for **choosing the lag values** and the **hyperparameter tuning**. The trained XGBoost model was interpreted using **SHAP** [3] to find the **important features** of the forecasting process.

Result

The simulation was run for **8 hours and 5 minutes**, sending **90,322 requests**, with around **35.11%** being **failed requests** mainly due to **timeout**. Table 1 shows the **forecasting quality** of the server's failure rate for each model using the rolling and expanding window evaluation setups. The bolded numbers indicate the **lowest value** of the evaluation metric in each column, which implies the **best performance** of all models. **XGBoost** achieved the **lowest MAE and RMSE** metrics value, especially when using the **rolling window evaluation setup**. Figure 2 shows the **SHAP bar plot** from the XGBoost model for interpretation, sorted in descending order based on the **mean of the absolute SHAP values**, which indicate the **average influence of a feature** for forecasting the data points in test data. Notice that the XGBoost model was trained with **lag 1** and **lag 2** features of each predictor. The feature '**request_duration**' had by far the **highest value** of the mean of the absolute SHAP value.

Model	Rolling		Expanding	
	MAE	RMSE	MAE	RMSE
XGBoost (multivariate)	**0.07900**	**0.17835**	**0.08792**	**0.18668**
ARMA	0.17644	0.37356	0.17004	0.36475
VAR	0.11336	0.28558	0.11261	0.28718
XGBoost (univariate)	0.17223	0.29040	0.16625	0.28541

Table 1. The performance of the models on the test data.

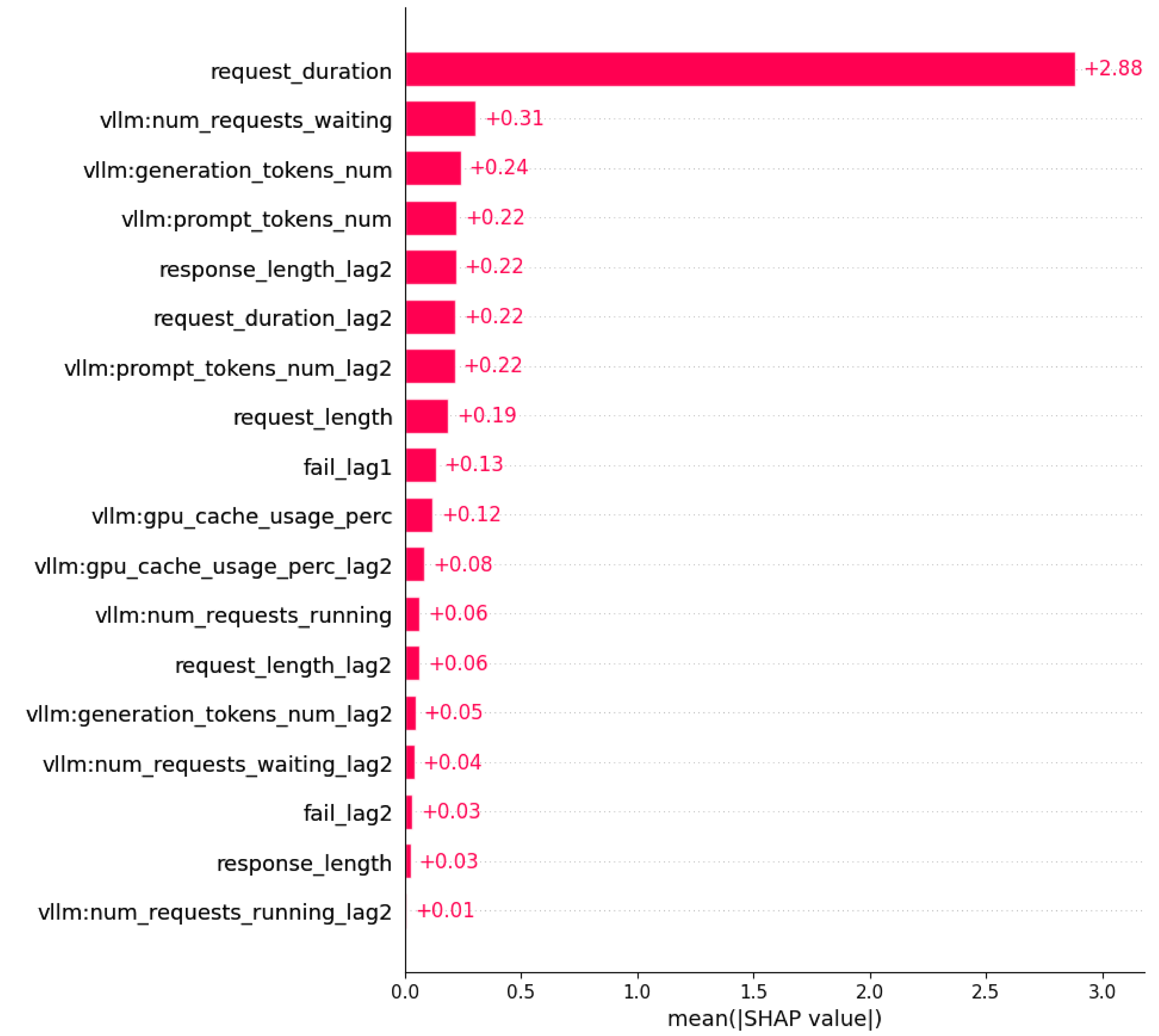

Figure 2. SHAP bar plot.

Discussion

The experiment result can be outlined into **three main findings**.

Meaningful forecast with LLM simulation using XGBoost and BurstGPT

The time series data generated from the LLM simulation using **BurstGPT** was useful for the use case to create **a meaningful forecast** of the LLM server performance, i.e., the **server's failure rate of processing requests at specific time intervals**, with the **XGBoost** model. With the rolling and expanding window evaluation setup, the **XGBoost** model outperformed the ARMA, VAR, and univariate XGBoost model. The **multivariate models** (XGBoost & VAR) also performed better in general than the **univariate models** (ARMA & univariate XGBoost).

Successful essential key features search for forecasting using SHAP

The model interpretation method with **SHAP** values was able to find the **comprehensible key features** essential for forecasting the LLM server performance based on the use case. The interpretation of the XGBoost model with SHAP revealed that **request duration** was the most important feature of the forecasting process. This result was consistent with the **XGBoost feature importance method** with **weight**, **gain**, and **coverage**.

Empirical prompt length distribution vs. Zipf distribution for prompt sampling

Evaluating the **prompt length distribution** assumption before choosing the prompt sampling distribution is important, as it might offer a **different result** on the simulation, forecasting, and interpretation. By **substituting** the Zipf distribution with the **empirical prompt length distribution** of the chosen dataset for prompt sampling, this change made the simulation run **23 minutes** longer, with **more failed requests** of around **42.95%** and some **differences** in terms of the **feature distributions** after data preparation based on the **Kolmogorov-Smirnov test**.

Future Works

In the future, **other use cases** with **different LLM serving frameworks** should be tested to test the generality of the approach. The possibility of **testing the result** from this experiment in a **production environment** should also be explored to check if the result and understanding from the simulation can be translated into production.

References

[1] Y. Wang, Y. Chen, Z. Li, Z. Tang, R. Guo, X. Wang, Q. Wang, A. C. Zhou, and X. Chu, "Towards efficient and reliable llm serving: A real-world workload study," 2024. Preprint.

[2] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, (New York, NY, USA), p. 785–794, Association for Computing Machinery, 2016.

[3] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[4] OpenAssistant, "OASST2 Huggingface model card," [Online; accessed 27 November 2024].

[5] vLLM, "vLLM documentation v0.5.3.post1," [Online; accessed 27 November 2024].

[6] Mistral, "Mistral-7B-Instruct-v0.2 Huggingface model card," [Online; accessed 27 November 2024].

[7] Prometheus, "Prometheus documentation," [Online; accessed 27 November 2024].

[8] H. Hewamalage, K. Ackermann, and C. Bergmeir, "Forecast evaluation for data scientists: common pitfalls and best practices," *Data Mining and Knowledge Discovery*, vol. 37, pp. 788–832, Mar 2023.