An Introductory Guide to Developing GenAI Services for Higher Education

Sarah Rodenbeck
Rosen Center for Advanced Computing
Purdue University
West Lafayette, IN, USA
srodenb@purdue.edu

Erik Gough
Rosen Center for Advanced Computing
Purdue University
West Lafayette, IN, USA
goughes@purdue.edu

Ashish
Rosen Center for Advanced Computing
Purdue University
West Lafayette, IN, USA
ashish@purdue.edu

Sathvika Kotha
Rosen Center for Advanced Computing
Purdue University
West Lafayette, IN, USA
kotha8@purdue.edu

K. Meher Hasanth
Rosen Center for Advanced Computing
Purdue University
West Lafayette, IN, USA
kmeherha@purdue.edu

Durga Dash
Rosen Center for Advanced Computing
Purdue University
West Lafayette, IN, USA
dashd@purdue.edu

***Abstract*—This paper reports on the lessons learned from developing and deploying campus-wide large language model (LLM) services at Purdue University for generative AI (GenAI) applications in education and research. We present a framework for identifying an LLM solution suite and identify key considerations related to developing custom solutions. While the GenAI ecosystem continues to evolve, the framework is intended to provide a tool- and organization-agnostic approach to guide leaders in conversations and strategy for future work and collaboration in this emerging field.**

***Index Terms*—Large Language Model Deployment, GenAI, Cloud Computing, Software Design, Requirements Analysis, Information Retrieval, Language Models, Software Management**

I. Introduction

The release of recent Large Language Models (LLMs) such as ChatGPT, LLaMA, Claude, and Gemini has catalyzed significant interest in Generative AI (GenAI) across various fields, including education [1]. The University of Michigan has been at the forefront in this area, notably creating and deploying the U-M GPT and U-M Maizey tools, designed for general use and fine-tuning, respectively [2].

Over the past six months, our team at Purdue University has sought to assemble a suite of LLM services to satisfy a range of use cases within the campus and ACCESS communities. Through our exploration, we have realized that there is little agreement over best practices and approaches to embarking on similar projects as new tools are released so frequently [3], [4]. Additionally, contrary to traditional software projects, a key element of GenAI projects is adapting to constant changes in the landscape (both internal GenAI-related priorities/appetite and external services available). Therefore, planning to pivot is a top-level consideration. This paper shares our experiences and learnings in evaluating, adapting, and deploying a suite of LLM services to enable AI-powered research. We have sought to leverage this into a general framework with guidance for other groups embarking on similar projects.

II. Methods

We present our approach in the context of a decision framework, which includes requirement analysis, suite selection, and custom service development considerations as top-level issues, which are discussed in turn.

A. Requirement Analysis

One pitfall that early adopters may fall into is to assume that finding a single approach that works for all the needs at a university is the single high-priority first step. On the contrary, a suite of services is the likely outcome of a substantially developed GenAI initiative. To note, the outcome might still converge to one or a few solutions depending on the mission or size of an institution; however, by assuming a multi-pronged approach from the start, the development effort will likely be more adaptable as requirements change. Therefore, much of the early effort is best spent on formalizing 1) overall project constraints, 2) user profiles, and 3) intra-project prioritization rather than searching for a single solution. While a general requirement analysis approach is still valid, there is a subtle difference in the benefits posed by conducting this type of analysis within the GenAI space. Rather than only being used for initial planning, such an analysis is used as a basis for assessing field developments. For example, suppose a new model or tool comes out. What is the relative benefit of adding support for this into the original suite versus shifting to support this instead of something in the original suite versus doing nothing and keeping the original approach? This assessment depends on fully understanding the vision and resources behind the project and the prioritized user profiles but allows more focused discussions.

B. Tool Selection

This part of the framework primarily concerns identifying a solution suite that satisfies the prioritized use cases from section 2A. It explores the balance between commercial and

custom services, addressing not just a simple build vs. buy decision but the optimal mix for usage and guidance.

Control is an overarching theme in this analysis, focusing on the control provided by various solutions as well as the level of control required. For instance, freely available tools such as the public version of ChatGPT pose a trade-off with data security, with chat histories being used by default for further fine-tuning and offering users very little control [5]. Our community's regular interactions with proprietary data and the need for privacy and IP considerations made such solutions generally inappropriate for organizational GenAI use in our case.

Commercial services like Azure AI Studio offer access to proprietary models while offering additional data privacy [6] but remain bound to the provider's policies and pricing structures. These services necessitate the use of commercial cloud resources and do not allow for the extraction of fine-tuned models. However, they support a broader range of models and include low-code tools for creating personalized "Assistants" with custom data. These services' features have been converging, with most providing APIs, model playgrounds, assistant-building tools, and user data support. However, different services offer access to different proprietary models (e.g., Gemini is only available through Google). Thus, partner selection hinges on existing organizational partnerships, desired flexibility/model access, and cost considerations. Cost estimations consider factors like deterministic seat-based versus undefined usage-based fees, intended usage, and the size and characteristics of the user base (e.g., API calls vs interface) [7], [8]. To serve all researcher needs (e.g., if access to many different proprietary models is needed), forming partnerships with multiple commercial services may be necessary.

Custom services, built from open-source tools/models and hosted with on-premises resources, require more resources to deploy and maintain but offer greater control. This makes them particularly appealing to power users or those with stringent data protection needs. Despite the growth in available open-source tools, most resources still lack complete functionality out of the box. Additionally, for the goals of our project, building a custom service was crucial to enabling lower-cost or free access to Generative AI tools to users who may not have the funding to pay for commercial services. Rather than having usage-based costs, custom services will incur significantly more development and infrastructure costs, which are discussed in more depth in section 2C. Still, when such resources are already available, this can essentially subsidize the raw cost [9].

Mapping use cases to the necessary level of control, balanced against costs and priorities, forms a strategic execution roadmap. It is crucial to understand both the short-term and long-term solution landscapes. Figure 1 illustrates a sample mapping of user types to services and feature usage, highlighting the flexibility in these mappings. While enterprise cloud services may require less startup time, the end state of custom services can be comparable, mainly differing in the models available (which may be a key consideration, depending on the user base). Initially, a GenAI solution suite might focus more on enterprise services, but users may be guided toward custom services as new features are added. Additionally, some scenarios may necessitate hybrid solutions where custom features are developed on top of commercial APIs.

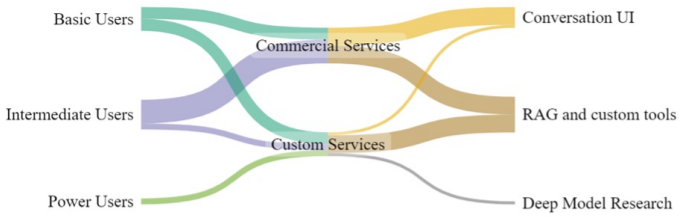

Fig. 1. Mapping of users to services and feature usage.

C. Custom Service Development

Assuming that a custom service has been identified as part of the solution suite, as we did, understanding what goes into a custom service is also critical for planning. We conducted a comprehensive investigation into the process of deploying an on-prem LLM, specifically the LLaMA LLMs, onto the Anvil composable subsystem—a Kubernetes-based private cloud for deploying and managing persistent services [10].

1) Approach and Deployment: The massive size of LLMs, such as LLaMA3 with 70 billion parameters [11] and GPT 3.5, the most recent OpenAI model for which details are available, with 175 billion parameters [12], makes deployment more complex than with smaller services. Initial issues included errors related to insufficient CUDA memory–internal testing showed that hosting the full LLaMA3 model required 4x 80GB GPUs. Employing quantization was paramount to navigating this issue. Quantization involves some information loss, in this case reducing the precision of numerical values, but allows the model to fit into smaller hardware than the cluster it was trained on [13]. Significant work has been done on these topics, and tools like Ollama automatically employ quantization methods, even enabling LLMs to run on local machines [14]. However, running quantized models on a cluster with GPU resources improves performance. Initially, we used a more manual approach for quantization, but the development ecosystem surrounding Ollama presented a very attractive option.

However, the model must still be paired with sufficient GPU memory and resources to work well, especially for a large community of users. On average, model performance correlates with model size, but smaller models may still be sufficient for particular tasks [15]. Assuming that anticipated users will want a choice of models, though, sizing the deployment to account for the largest model size will result in the best performance. While LLaMA 3 70B, for example, does work on GPUs with smaller memory, response times

suffer. For satisfactory performance with multiple models, we recommend the minimum resource request includes 16 CPUs, 32GB of RAM, and a GPU with 40GB of memory. 40 GB of GPU memory allows for loading multiple small models simultaneously as well as larger models like LLaMA 3 70B with 4 bit quantization. Deploying the solution in an auto-scaling platform like Kubernetes is also desired to handle the highly variable GenAI resource demand. Ongoing computing costs must be factored into budgetary discussions.

2) Performance Benchmarking: Beyond our basic assessment of minimum resources, benchmarking performance was an essential aspect of our work to validate the scalability and limits of our approach. While the actual models and tools continue to shift, we anticipate the benchmarking approach to be more transferrable [16].

Our approach was based on LLMPerf [17], a library designed to validate and benchmark the throughput and latency performance of LLMs [18]. The library supports multiple backend frameworks and provides customizable scenarios to simulate real-world usage. Key features include scalable testing, detailed performance metrics, and extensibility for integration with existing testing infrastructures. LLMPerf's design also allows users to reproduce results reliably, ensuring that performance assessments are accurate and repeatable. We found it particularly valuable for optimizing LLMs for specific applications, as it offers pre-configured and user-defined benchmarking capabilities. Building on LLMPerf's framework and what we have found to be standard metrics in this domain, we chose to assess performance based on the following metrics.

- Inter-Token Latency: Displays variability in processing time between tokens, assessing fluctuating token processing speeds [19].
- End-to-End Latency: Total time from initiating to completing a request, pointing to variability in total response times.
- Time to First Token (TTFT): Latency from starting a request to generating the first token, where differences can indicate potential variations due to system load or complexity.
- Request Output Throughput: Measures tokens generated per second, which helps understand the system's efficiency in producing output.

We used a containerized benchmarking environment to ensure a stable and replicable platform [20]. Benchmarking was performed on an Ollama v0.2.4 container instance running in Kubernetes that was allocated 32 CPUs, 64GB of RAM and one A100 80GB GPU. The instance was configured to process up to 64 parallel requests using the OLLAMA_NUM_PARALLEL configuration option.

While LLMPerf did not have native support for an Ollama model backend, we were able to use LiteLLM, a library that standardizes API interactions across various LLM providers, to facilitate interaction [21]. Notably, LiteLLM's design is optimized for low-resource environments, making it particularly beneficial for our containerized setup, where resource efficiency is paramount.

We used a dedicated node on Purdue's Negishi cluster with 128 cores and 256 GB RAM as our benchmarking client and validated no client bottlenecks were present. LLMPerf load tests were conducted from inside a conda virtual environment for an increasing number of parallel requests.

While results will be highly dependent on the allocated resources for an LLM deployment as well as parameters like number of input and output tokens, a sample of our results can be found in Figure 2. Unsurprisingly, latency reliably increases with the number of concurrent requests. The relationship between Inter-Token latency and End-to-End latency is also easily visualized. Results for token throughput show that throughput drops quickly as more parallel requests are handled, eventually reaching a point where responses are less than 10 tokens/s, a threshold that would be considered too slow for users of the system. Using this benchmarking data, we plan to do future work to determine an optimal auto-scaling configuration for Ollama instances.

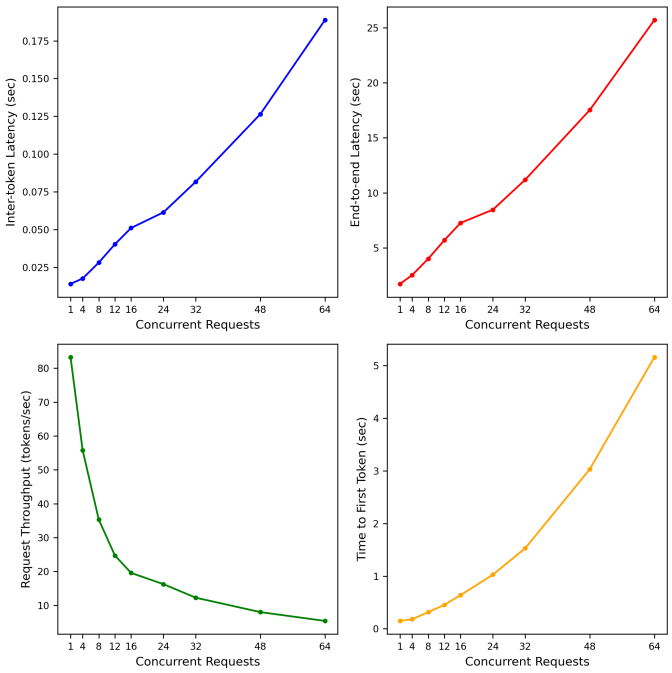

Fig. 2. Benchmarking Results

3) Additional Development Considerations: Beyond model deployment, numerous features are required to make custom services usable to a community. Pursuing a hybrid solution for building custom features on top of commercial model APIs may be possible but typically requires configuration as well [25]. However, solutions like Ollama are approaching the functionality offered by commercial solutions by automatically handling features such as streaming responses, not just providing a model backend [14]. Fortunately, many front-end tools have been designed to work with an Ollama backend, such as OpenWebUI and Chatbot Ollama [25],

[26]. However, no solution satisfied all our scaling needs, so having front-end expertise on the team was crucial for application modifications. Key features to consider include support for non-local backends, the ability to integrate with SSO, and the functionality to enable users to create custom retrieval augmented generation (RAG) systems [27], which also requires deploying a vector store like ChromaDB or PGVector. Figure 3 shows a sample deployment architecture. Finally, the pace of innovation means that while many frameworks are being released, there is not yet a consensus on the best approach to these tasks. Thus, building services modularly when possible and accounting for recurring costs to update the service to avoid technical debt as initial approaches become outdated are important. Support for RAG tools and agents for multi-hop reasoning has become very popular and should be considered a requirement for custom services in the long term.

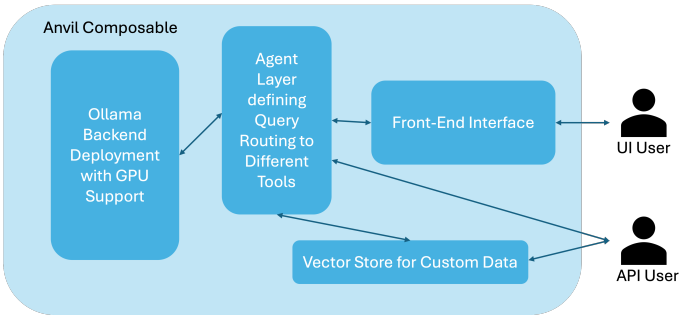

Fig. 3. Architecture of custom on-prem LLM service

III. Discussion

The ecosystem of GenAI tools evolves rapidly, and today's models and tools may differ significantly by the time Gateways '24 starts. Although our team at Purdue has thoroughly evaluated options for deploying a campus LLM, our specific results are less important than the broader insights gained. We believe other universities and research centers building similar gateways, and we hope our findings will guide their efforts. In this ever-changing environment, quickly adapting and assessing new developments is crucial for success.

IV. Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 2005632. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] Y. Liu et al., "Understanding LLMs: A Comprehensive Overview from Training to Inference," arXiv:2401.02038 [cs], Jan. 2024.

[2] T. Burns. "ITS debuts custom artificial intelligence services across U-M." University of Michigan. https://record.umich.edu/articles/its-debuts-customized-ai-services-to-u-m-community/ (accessed May 20, 2024).

[3] H. Crompton and D. Burke, "Artificial intelligence in higher education: The state of the field," Int J Educ Technol High Educ, vol. 20, no. 1, 22, Apr. 2023, doi:10.1186/s41239-023-00392-8.

[4] WhyLabs, "A Guide to Large Language Model Operations (LLMOps)," WhyLabs. https://whylabs.ai/blog/posts/guide-to-llmops (accessed May 26, 2024).

[5] OpenAI. "Data Controls FAQ." OpenAI. https://help.openai.com/en/articles/7730893-data-controls-faq (accessed May 20, 2024).

[6] OpenAI. "Introducing ChatGPT Enterprise." OpenAI. https://openai.com/blog/introducing-chatgpt-enterprise#OpenAI (accessed May 20, 2024).

[7] S. Heshmatisafa and M. Seppänen, "Exploring API-driven business models: Lessons learned from Amadeus's digital transformation." Digital Business, vol. 3, no. 1, 100055, Jan. 2023, doi:10.1016/j.digbus.2023.100055

[8] T. Hagendorff, "The Ethics of AI Ethics: An Evaluation of Guidelines," Minds and Machines, vol. 30, pp. 99-120, Feb. 2020, doi:10.1007/s11023-020-09517-8

[9] F. Kumeno, "Software engineering challenges for machine learning applications: A literature review," Intelligent Decision Technologies, vol. 13, no. 4, pp. 463-476, Feb 2020, doi:10.3233/IDT-190160

[10] X.C. Song et al. "Anvil - System Architecture and Experiences from Deployment and Early User Operations," in Practice and Experience in Advanced Research Computing (PEARC '22), 1–9.

[11] H. Touvron et al. "Llama 2: Open foundation and fine-tuned chat models," arXiv:2307.09288 [cs], July 2023.

[12] T. Brown et al., "Language Models are Few-Shot Learners," arXiv:2005.14165 [cs], May 2020.

[13] B. Jacob, et al. (2017). "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," arXiv:1712.05877 [cs], Dec 2017.

[14] J. Morgan. "Ollama." Ollama. https://ollama.com (accessed Apr. 16, 2024).

[15] T. Shnitzer et al., "Large Language Model Routing with Benchmark Datasets." arXiv:2309.15789 [cs], Sept. 2023.

[16] J. Dodge, S. Gururangan, D. Card, R. Schwartz and N.A. Smith, "Show Your Work: Improved Reporting of Experimental Results," arXiv:1909.03004 [cs], Sept. 2019.

[17] "LLMPerf: Large Language Model Performance Benchmarking," GitHub repository, Ray Project. [Online]. Available: https://github.com/ray-project/llmperf. (accessed Feb 4, 2024).

[18] J. Thiyagalingam, M. Shankar, G. Fox, and T. Hey, "Scientific Machine Learning Benchmarks." Nature Reviews Physics, vol. 4, pp. 413-420, 2022, doi:10.1038/s42254-022-00441-7.

[19] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ Questions for Machine Comprehension of Text," arXiv:1606.05250 [cs], June 2016.

[20] X. Zhou et al, "Benchmarking microservice systems for software engineering research." In ICSE '18 Companion (pp. 323-324), doi: 10.1145/3183440.3194991.

[21] "LiteLLM," GitHub repository, Berri AI. [Online]. Available: https://github.com/BerriAI/litellm (accessed Mar 5, 2024).

[22] K. Senjab, S. Abbas, N. Ahmed, A.u.R. Khan, "A survey of Kubernetes scheduling algorithms," Journal of Cloud Computing: Advances, Systems and Applications, vol. 12, no. 1, 87, June 2023, doi:10.1186/s13677-023-00471-1.

[23] Z. Sun, and A.V. Miceli-Barone, "Scaling Behavior of Machine Translation with Large Language Models under Prompt Injection Attacks," in Proceedings of the First edition of the Workshop on the Scaling Behavior of Large Language Models (SCALE-LLM 2024), pp. 9-23, Mar. 2024, doi:https://doi.org/10.48550/arXiv.2403.09832.

[24] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," Communications of the ACM, vol. 51, no. 1, pp. 107-113, Jan 2008, doi:10.1145/1327452.1327492.

[25] T.J. Baek. "Open WebUI." Open WebUI. https://docs.openwebui.com (accessed Apr. 14, 2024).

[26] I. Fioravanti. "ChatBot Ollama." GitHub repository. https://github.com/ivanfioravanti/chatbot-ollama. (accessed Feb. 23, 2024).

[27] P. Lewis, et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in Proceedings of Advances in Neural Information Processing Systems: 33, pp. 9459-9474, Dec. 2020, doi:10.5555/3495724.3496517.