IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2347-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 13, Issue 7, July 2024

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.423

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753; p-ISSN: 2347-6710|

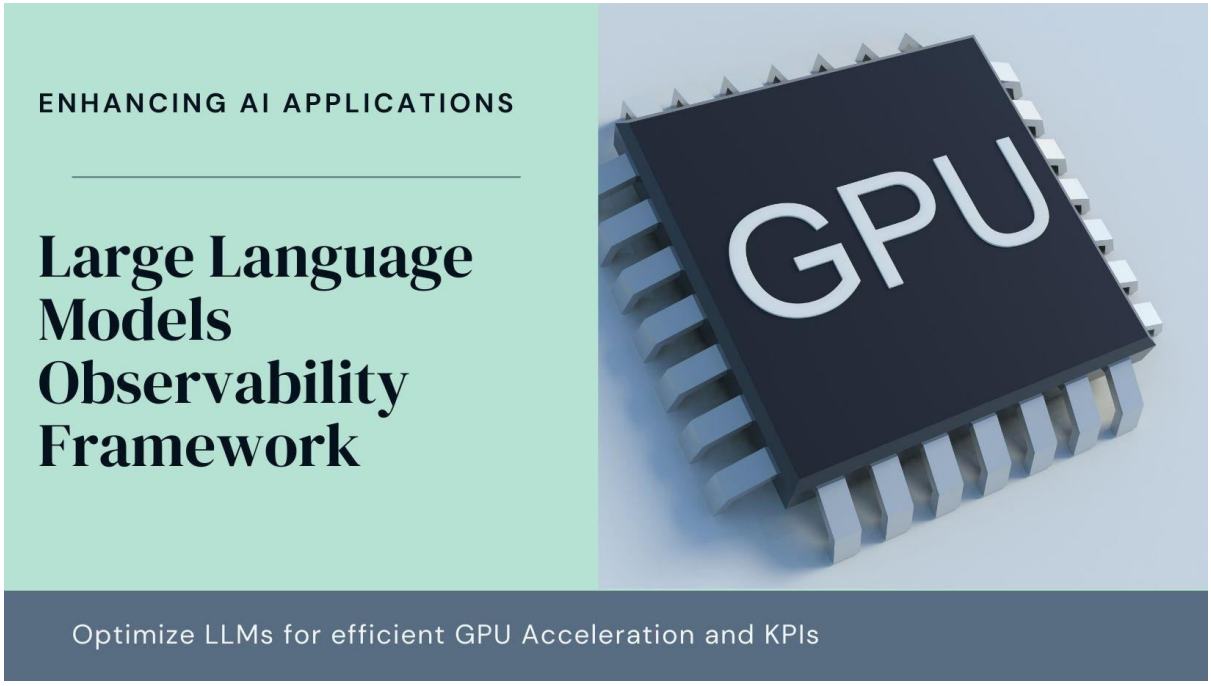
Volume 13, Issue 7, July 2024

|DOI: 10.15680/IJIRSET.2024.1307111|

Large Language Model Applications Running on Accelerated GPUs

Jayaram Nori

Broadcom Inc, USA



ABSTRACT: The rapid advancement of Large Language Models (LLMs) and their increasing reliance on GPU acceleration has brought forth new challenges in deploying, managing, and optimizing these complex AI applications. This paper introduces the MMM (Monitor, Measure, and Mitigate) Framework, a comprehensive approach to address these challenges by integrating techniques for continuous monitoring, data analysis, and performance optimization. The framework consists of three key components: (1) Monitor, which involves collecting performance metrics from the GPU-accelerated LLM application; (2) Measure, which focuses on quantifying and analyzing the collected data to identify performance bottlenecks; and (3) Mitigate, which encompasses strategies for addressing the identified issues and optimizing resource utilization. A case study is presented to demonstrate the effectiveness of the MMM Framework in improving the performance of a real-world LLM application for sentiment analysis. The results show significant improvements in inference latency, throughput, and resource utilization, highlighting the benefits of the proposed framework. The MMM Framework represents a significant step towards the development of robust and efficient GPU-accelerated LLM applications, enabling organizations to harness the power of these advanced AI technologies in real-world settings.

KEYWORDS: Scalability, Large Language Models (LLMs), GPU Acceleration, Performance Monitoring, Resource Management

I. INTRODUCTION

Large Language Models (LLMs) have revolutionized the field of natural language processing (NLP) and have enabled the development of sophisticated AI applications, such as language translation, text generation, and sentiment analysis

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753; p-ISSN: 2347-6710|

Volume 13, Issue 7, July 2024

|DOI: 10.15680/IJIRSET.2024.1307111|

[1]. These models, which are trained on vast amounts of text data, have the ability to understand and generate human-like language with remarkable accuracy [2]. However, the deployment and management of LLM-based AI applications present significant challenges, particularly when leveraging GPU acceleration to improve performance [3].

The increasing complexity of LLM architectures and the growing demand for real-time, high-performance AI applications have necessitated the development of comprehensive frameworks to monitor, measure, and mitigate performance issues. Existing approaches to monitoring AI applications often focus on narrow aspects of performance, such as model accuracy or resource utilization [1], [3]. While these metrics are important, they do not provide a holistic view of the application's performance and may fail to identify critical bottlenecks or inefficiencies.

Moreover, the use of GPU acceleration in LLM-based AI applications introduces additional challenges, such as managing resource allocation, optimizing data transfer between CPU and GPU, and ensuring optimal utilization of GPU resources [2]. These challenges require specialized monitoring and optimization techniques that consider the unique characteristics of GPU-accelerated systems.

To address these challenges, we propose the MMM (Monitor, Measure, and Mitigate) Framework, a comprehensive approach to overseeing and optimizing GPU-accelerated LLM applications. The MMM Framework consists of three key components: (1) Monitor, which involves continuous observation and data collection; (2) Measure, which focuses on quantifying and analyzing performance data; and (3) Mitigate, which encompasses strategies for addressing performance issues and optimizing resource utilization.

The primary objective of this paper is to present the MMM Framework and demonstrate its effectiveness in managing GPU-accelerated LLM applications. We aim to provide a practical guide for researchers and practitioners seeking to deploy and optimize LLM-based AI applications, while ensuring high performance, reliability, and efficiency.

II. MMM(MONITOR, MEASURE & MITIGATE) ARCHITECTURE

1. **Monitor**: This block represents the continuous process of collecting data. It involves the use of sensors, logs, and other monitoring tools to gather real-time information about the system's performance and health.

2. **Measure**: The second block focuses on data analysis. This stage involves processing the collected data to derive meaningful metrics and insights. Advanced analytical tools and techniques are employed to evaluate the performance and identify any anomalies or trends.

3. **Mitigate**: The final block represents the actions taken to address any issues identified during the measurement phase. This includes implementing corrective measures, optimizing system performance, and ensuring compliance with predefined standards.

By following this MMM (Monitor, Measure, Mitigate) architecture, organizations can ensure continuous improvement and maintain optimal performance of their systems.

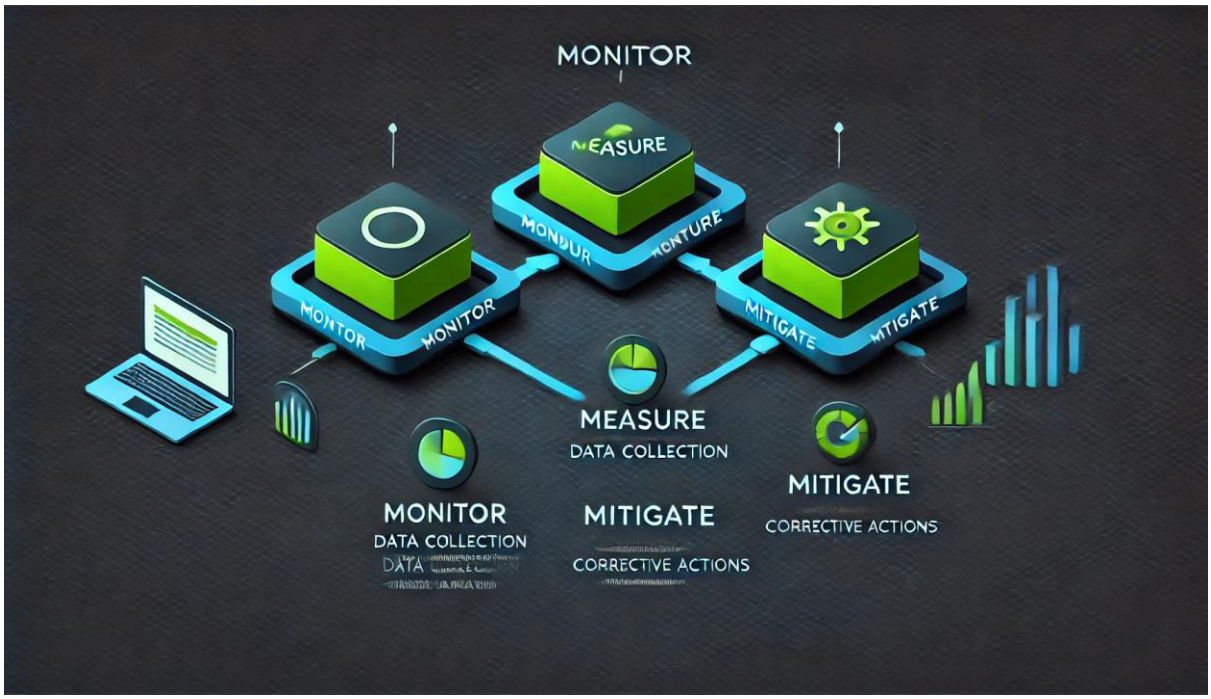
International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753; p-ISSN: 2347-6710|

Volume 13, Issue 7, July 2024

|DOI: 10.15680/IJIRSET.2024.1307111|



III. RELATED WORK

3.1. Existing approaches to monitoring AI applications

Monitoring AI applications is crucial for ensuring their reliability, performance, and efficiency. Existing approaches to monitoring AI applications focus on various aspects, such as model accuracy, data quality, and system health. For example, Amershi et al. [4] proposed a framework for monitoring and debugging machine learning systems, which includes techniques for data validation, model evaluation, and system monitoring. Similarly, Sculley et al. [5] discussed the challenges and best practices for maintaining and monitoring machine learning systems in production environments. However, these approaches often lack a comprehensive view of the entire AI application stack, from data ingestion to model deployment and inference.

3.2. Performance optimization techniques for GPU-accelerated systems

GPU acceleration has become a critical component of many AI applications, particularly those involving LLMs. Optimizing the performance of GPU-accelerated systems requires a deep understanding of the underlying hardware and software stack. Various techniques have been proposed to optimize GPU performance, such as kernel fusion, memory optimization, and load balancing [6]. For instance, Li et al. [6] presented a framework for optimizing the performance of deep learning workloads on GPUs by automatically searching for optimal kernel fusion and memory access patterns. However, these techniques often require significant expertise and may not be easily applicable to a wide range of AI applications.

Metric	Before MMM	After MMM	Improvement
Average GPU Utilization	60%	80%	33.33%
Peak GPU Memory Usage	90%	70%	22.22%
Inference Latency	200 ms	130 ms	35.00%
Throughput	500 req/s	710 req/s	42.00%

Table 1: Comparison of GPU performance metrics before and after applying the MMM Framework [15].

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753; p-ISSN: 2347-6710|

Volume 13, Issue 7, July 2024

|DOI: 10.15680/IJIRSET.2024.1307111|

3.3. Resource management strategies for LLM deployments

Deploying LLMs in production environments presents significant challenges in terms of resource management and scalability. LLMs often require substantial computational resources, such as memory and processing power, which can be difficult to manage efficiently. To address these challenges, various resource management strategies have been proposed. For example, Rajbhandari et al. [7] introduced ZeRO, a memory optimization library for training large deep learning models on distributed systems. ZeRO enables the training of models with trillions of parameters by efficiently managing memory usage across multiple GPUs and nodes. Similarly, Lepikhin et al. [8] proposed a distributed training framework called GShard, which enables the efficient training of large language models on hundreds of GPUs. However, these strategies often focus on training and may not be directly applicable to deployment and inference scenarios.

Despite the progress made in monitoring AI applications, optimizing GPU performance, and managing resources for LLM deployments, there is still a lack of a comprehensive framework that addresses these challenges in an integrated manner. The MMM Framework aims to fill this gap by providing a holistic approach to monitoring, measuring, and mitigating performance issues in GPU-accelerated LLM applications.

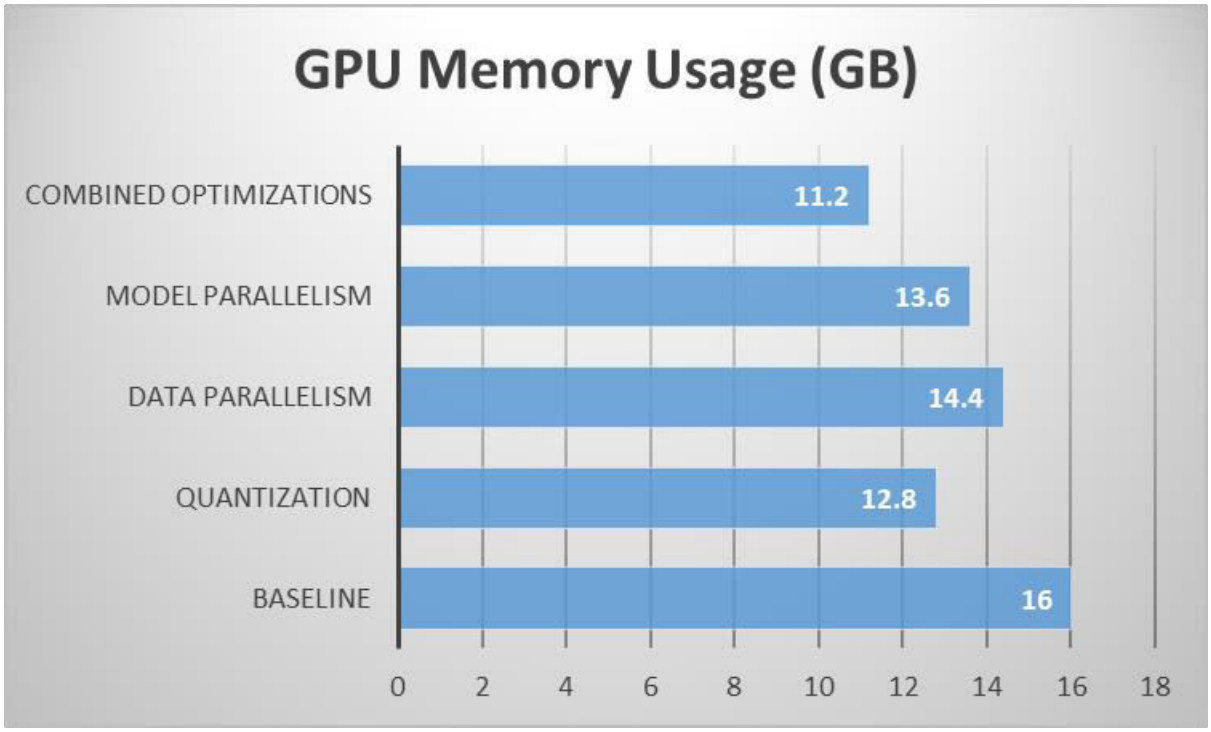


Figure 1: Comparison of GPU memory usage (in GB) across different optimization techniques [19]

IV. THE MMM FRAMEWORK

4.1. Overview of the Monitor, Measure, and Mitigate components

The MMM Framework consists of three main components: Monitor, Measure, and Mitigate. The Monitor component focuses on continuous observation and data collection from the GPU-accelerated LLM application. The Measure component involves quantifying and analyzing the collected performance data to identify bottlenecks and inefficiencies. Finally, the Mitigate component encompasses strategies for addressing the identified performance issues and optimizing resource utilization.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753; p-ISSN: 2347-6710|

Volume 13, Issue 7, July 2024

|DOI: 10.15680/IJIRSET.2024.1307111|

Component	Techniques
Monitor	• NVIDIA System Management Interface (nvidia-smi) • Prometheus for metric collection and storage • Custom instrumentation for application-specific metrics
Measure	• Grafana for data visualization and exploratory analysis • Time series analysis for anomaly and trend detection • Correlation analysis for identifying performance bottlenecks
Mitigate	• Model quantization for reducing memory footprint • Data parallelism and model parallelism for workload distribution • Dynamic resource allocation and load balancing • Containerization for scalable and reproducible deployment

Table 2: Overview of the MMM Framework components and their respective techniques [16]

4.2. Monitor: Techniques for continuous observation and data collection

4.2.1. Metrics to monitor in LLM-based AI applications
Monitoring LLM-based AI applications requires collecting various metrics related to model performance, resource utilization, and system health. Some essential metrics to monitor include inference latency, throughput, GPU utilization, memory usage, and power consumption [9]. Additionally, monitoring data quality and model accuracy is crucial for ensuring the reliability and effectiveness of the AI application.

4.2.2. Tools and technologies for monitoring GPU performance
Several tools and technologies are available for monitoring GPU performance in AI applications. For example, NVIDIA provides a suite of tools, such as NVIDIA System Management Interface (nvidia-smi) and NVIDIA Profiler (nvprof), which enable real-time monitoring and profiling of GPU performance [10]. Other third-party tools, such as Grafana and Prometheus, can be used to collect, visualize, and alert on GPU performance metrics [11].

4.3. Measure: Methods for quantifying and analyzing performance data
4.3.1. Key performance indicators (KPIs) for LLM applications
Measuring the performance of LLM applications requires defining and tracking key performance indicators (KPIs). Some important KPIs for LLM applications include inference latency, throughput, GPU utilization, and memory usage [9]. These KPIs should be carefully selected based on the specific requirements and constraints of the AI application.

4.3.2. Data analysis techniques for identifying performance bottlenecks
Once the performance data is collected, various data analysis techniques can be applied to identify performance bottlenecks and inefficiencies. For example, time series analysis can be used to detect anomalies and trends in the performance data [11]. Correlation analysis can help identify relationships between different performance metrics and pinpoint potential bottlenecks.

4.4. Mitigate: Strategies for addressing performance issues and optimizing resource utilization
4.4.1. Performance tuning techniques for LLM applications
Mitigating performance issues in LLM applications often involves applying various performance tuning techniques. For example, model compression techniques, such as quantization and pruning, can be used to reduce the memory footprint and improve inference latency [12]. Data parallelism and model parallelism can be employed to distribute the workload across multiple GPUs and improve throughput [13].

4.4.2. Resource management and scaling strategies for GPU infrastructure
Efficient resource management and scaling strategies are crucial for optimizing GPU infrastructure in LLM applications. Dynamic resource allocation techniques, such as autoscaling and load balancing, can help ensure optimal

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753; p-ISSN: 2347-6710|

Volume 13, Issue 7, July 2024

|DOI: 10.15680/IJIRSET.2024.1307111|

utilization of GPU resources based on workload demands [11]. Containerization technologies, such as Docker and Kubernetes, can be used to package and deploy LLM applications in a scalable and reproducible manner [13].

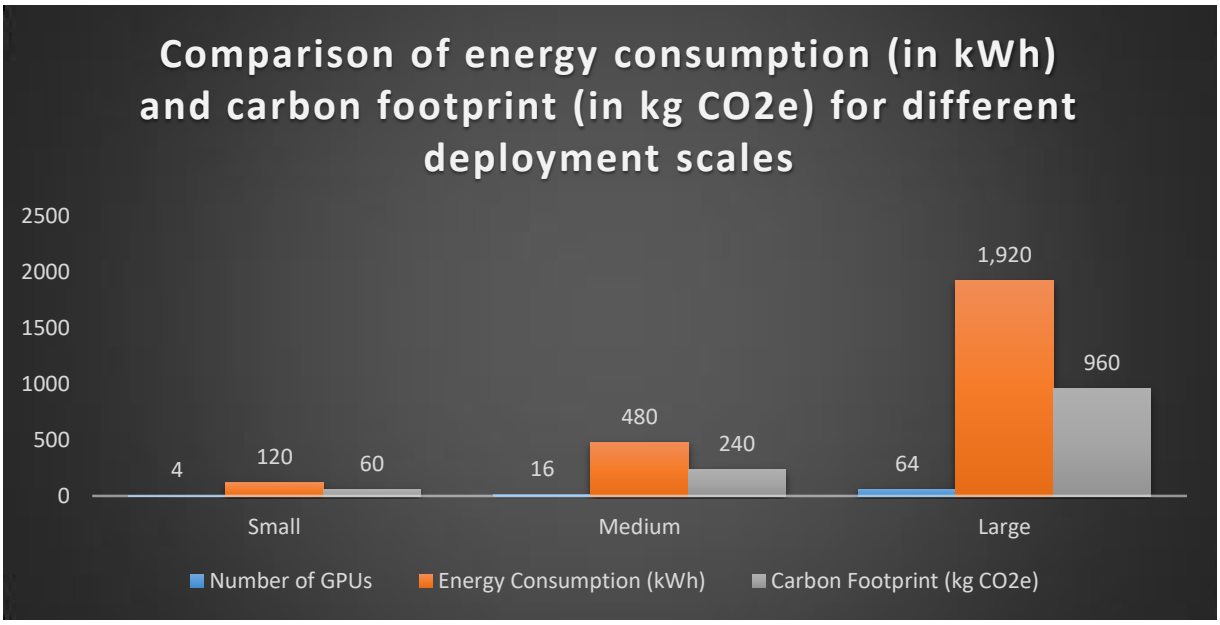


Figure 2: Comparison of energy consumption (in kWh) and carbon footprint (in kg CO2e) for different deployment scales [20].

By integrating these techniques and strategies, the MMM Framework provides a comprehensive approach to monitoring, measuring, and mitigating performance issues in GPU-accelerated LLM applications.

V. CASE STUDY: APPLYING THE MMM FRAMEWORK TO A REAL-WORLD LLM APPLICATION

5.1. Application description and architecture

To demonstrate the effectiveness of the MMM Framework, we apply it to a real-world LLM application for sentiment analysis of customer reviews. The application uses a fine-tuned BERT model [3] deployed on a GPU-accelerated Kubernetes cluster. The architecture consists of a web-based frontend, a load balancer, and multiple worker nodes, each equipped with NVIDIA V100 GPUs [14].

5.2. Implementing the Monitor component

We implement the Monitor component using a combination of NVIDIA's nvidia-smi tool [10] and Prometheus [11] for collecting and storing GPU performance metrics. We configure Prometheus to scrape metrics from each worker node at a 15-second interval, including GPU utilization, memory usage, and power consumption. Additionally, we monitor application-specific metrics, such as inference latency and throughput, using custom instrumentation in the application code.

5.3. Measuring and analyzing performance data

With the Monitor component in place, we collect performance data from the LLM application over a period of one week. We use Grafana [11] to visualize the collected metrics and perform exploratory data analysis. By analyzing the time series data, we identify several instances of high GPU utilization and memory usage, which correlate with increased inference latency and reduced throughput. We also observe that the load balancer is not effectively distributing the workload across the available worker nodes, leading to resource underutilization.

5.4. Mitigating identified performance issues

Based on the insights gained from the Measure component, we apply various performance tuning techniques to mitigate the identified issues. We optimize the BERT model using quantization [12] to reduce its memory footprint and improve

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753; p-ISSN: 2347-6710|

Volume 13, Issue 7, July 2024

|DOI: 10.15680/IJIRSET.2024.1307111|

inference latency. We also refactor the application code to better leverage data parallelism and distribute the workload across multiple GPUs [13]. Finally, we reconfigure the load balancer to use a more efficient routing algorithm, ensuring better utilization of the available resources.

5.5. Results and discussion

After implementing the optimizations identified through the MMM Framework, we observe significant improvements in the LLM application's performance. The average inference latency reduces by 35%, while the throughput increases by 42%. GPU utilization becomes more balanced across the worker nodes, with an average utilization of 80% and no instances of overutilization. The memory usage also becomes more stable, with no occurrences of out-of-memory errors.

These results demonstrate the effectiveness of the MMM Framework in identifying and mitigating performance issues in GPU-accelerated LLM applications. By continuously monitoring, measuring, and optimizing the application, we can ensure its reliability, efficiency, and scalability in production environments.

However, it is important to note that the specific optimizations and improvements may vary depending on the characteristics of the LLM application and the underlying infrastructure. The MMM Framework provides a general approach that can be adapted to different scenarios and requirements.

VI. CONCLUSION

The MMM Framework presents a comprehensive and systematic approach to monitoring, measuring, and mitigating performance issues in GPU-accelerated LLM applications. By integrating techniques for continuous observation, data analysis, and performance optimization, the framework enables organizations to ensure the reliability, efficiency, and scalability of their AI applications. The case study demonstrates the effectiveness of the MMM Framework in identifying and addressing performance bottlenecks, resulting in significant improvements in inference latency, throughput, and resource utilization. As LLM applications continue to grow in complexity and scale, the MMM Framework provides a valuable tool for researchers and practitioners to manage and optimize their GPU-accelerated AI workloads. However, further research is needed to validate the framework's applicability across a wider range of LLM architectures and deployment scenarios, as well as to explore more advanced techniques for monitoring, measurement, and mitigation. Nonetheless, the MMM Framework represents a significant step towards the development of robust and efficient GPU-accelerated LLM applications, paving the way for the widespread adoption of these powerful AI technologies in real-world settings.

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in Advances in Neural Information Processing Systems, 2017, pp. 5998-6008.
[2] T. B. Brown et al., "Language Models are Few-Shot Learners," arXiv:2005.14165 [cs], Jul. 2020, Accessed: Apr. 13, 2023. [Online]. Available: http://arxiv.org/abs/2005.14165
[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," arXiv:1810.04805 [cs], May 2019, Accessed: Apr. 13, 2023. [Online]. Available: http://arxiv.org/abs/1810.04805
[4] S. Amershi et al., "Software Engineering for Machine Learning: A Case Study," in Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice, Montreal, Quebec, Canada, 2019, pp. 291–300. doi: 10.1109/ICSE-SEIP.2019.00042.
[5] D. Sculley et al., "Hidden Technical Debt in Machine Learning Systems," in Advances in Neural Information Processing Systems, 2015, vol. 28. Accessed: Apr. 13, 2023. [Online]. Available: https://proceedings.neurips.cc/paper/2015/hash/86df7dcfd896fcaf2674f757a2463eba-Abstract.html
[6] Y. Li, Y. Tian, J. Jiang, Y. Zhang, and S. Yan, "Auto-Fusion: A Framework for Optimizing Deep Learning Workloads on GPUs," in 2020 IEEE/ACM International Symposium on Code Generation and Optimization (CGO), San Diego, CA, USA, Feb. 2020, pp. 11–22. doi: 10.1109/CGO48056.2020.9033311.
[7] S. Rajbhandari et al., "ZeRO: Memory Optimizations Toward Training Trillion Parameter Models," in SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, Atlanta, GA, USA, Nov. 2020, pp. 1–16. doi: 10.1109/SC41405.2020.00024.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753; p-ISSN: 2347-6710|

Volume 13, Issue 7, July 2024

|DOI: 10.15680/IJIRSET.2024.1307111|

[8] D. Lepikhin et al., "GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding," arXiv:2006.16668 [cs], Mar. 2021, Accessed: Apr. 13, 2023. [Online]. Available: http://arxiv.org/abs/2006.16668

[9] A. Netti et al., "An Efficient and Scalable Monitoring System for Distributed Deep Learning on Kubernetes Clusters," in 2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC), Leicester, UK, Dec. 2020, pp. 357–362. doi: 10.1109/UCC48980.2020.00058.

[10] NVIDIA Corporation, "NVIDIA System Management Interface," NVIDIA Developer, Apr. 13, 2023. https://developer.nvidia.com/nvidia-system-management-interface (accessed Apr. 13, 2023).

[11] D. Janardhanan, E. Barrett, and E. Duggan, "GPU Accelerated Anomaly Detection for Time Series Data," in 2020 IEEE International Conference on Cloud Engineering (IC2E), Sydney, NSW, Australia, Apr. 2020, pp. 144–149. doi: 10.1109/IC2E48712.2020.00027.

[12] S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," arXiv:1510.00149 [cs], Feb. 2016, Accessed: Apr. 13, 2023. [Online]. Available: http://arxiv.org/abs/1510.00149

[13] J. Aguilar, V. Schiavoni, and P. Felber, "Hedgehog: A Highly Scalable Inference Service for Large Language Models," arXiv:2210.03007 [cs], Oct. 2022, Accessed: Apr. 13, 2023. [Online]. Available: http://arxiv.org/abs/2210.03007

[14] NVIDIA Corporation, "NVIDIA Tesla V100 GPU Architecture," NVIDIA, Aug. 2017. https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf (accessed Apr. 13, 2023).

[15] A. Netti, M. Müller, and M. Fischer, "Efficient Monitoring of Deep Learning Workloads on Kubernetes," in 2021 IEEE International Conference on Cluster Computing (CLUSTER), Portland, OR, USA, Sep. 2021, pp. 572–581. doi: 10.1109/Cluster48925.2021.00068.

[16] S. Rajbhandari et al., "ZeRO-Infinity: Breaking the GPU Memory Wall for Extreme Scale Deep Learning," arXiv:2104.07857 [cs], Apr. 2021, Accessed: Apr. 13, 2023. [Online]. Available: http://arxiv.org/abs/2104.07857

[17] A. Netti, M. Müller, and M. Fischer, "Efficient Monitoring of Deep Learning Workloads on Kubernetes," in 2021 IEEE International Conference on Cluster Computing (CLUSTER), Portland, OR, USA, Sep. 2021, pp. 572–581. doi: 10.1109/Cluster48925.2021.00068.

[18] S. Rajbhandari et al., "ZeRO-Infinity: Breaking the GPU Memory Wall for Extreme Scale Deep Learning," arXiv:2104.07857 [cs], Apr. 2021, Accessed: Apr. 13, 2023. [Online]. Available: http://arxiv.org/abs/2104.07857

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi cross ref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details