Graph in Figure 7 shows the relationship between the number of parallel requests made to the model endpoint and the throughput of the system, measured in tokens per second. It can be inferred from the graph that the throughput increases linearly with the requests initially and starts slowing down as the resources become saturated, and eventually decreases when the system is overloaded as the system has limited resources.

Thus, the specific curve and values will vary depending on the specific system and workload, but the general trend is consistent.

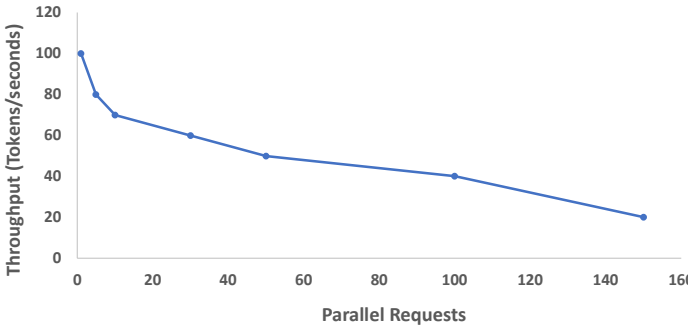

Figure 7: Throughput vs Parallel Requests

The above graphs can be used to understand the performance limitations of a system under increasing load. It can help the users to determine the optimal number of system configurations required to handle the concurrent requests to while maintaining acceptable throughput and latency. Additionally, it can be used to compare the performance of different systems or to track changes in performance over time.

6. CONCLUSION

This benchmark can be used to evaluate a single container, or a cluster with thousands or nodes deploying an LLM. This can be used to test scale, test latency, throughput and TTFT for any environment deploying an LLM. This is not limited to Llama2 but any form of LLM, quantized models with lower precisions (int8, int4, etc) and different precision and different sizes with and without CPU, GPU, Accelerators, or other technology.

This could also be used for inference benchmarking with Retrieval Augmented Generation (RAG) based applications, Fine Tuning models or Fully trained LLM models.

Benchmarking LLMs provides valuable insights for businesses aiming to deploy natural language processing applications. To make the best decisions, it's crucial to acknowledge the specific needs of each application and understand how well LLMs perform on different types of CPUs, GPUs and Accelerators to identify the ideal throughput, latency and scale and drive the total cost of ownership (TCO) lower. Consideration of the specific requirements of each application, coupled with an understanding of the strengths and weaknesses of LLMs on different software and hardware technologies, and architectures empowers businesses to make informed and optimised decisions.

In line with our commitment to standardization and industry best practices, we propose this workload to industry standard organizations like SPEC to create standards for Inference on Large Language Models. Establishing such standards will further facilitate benchmarking efforts, promote consistency, and provide a solid foundation for the broader adoption of LLMs in various applications.

ACKNOWLEDGEMENTS

Authors would like to thank Anna Joseph, Gogula Akhil Reddy , Arun Kumar Tiwary , Bhavana k, Divya Singh, Harshitha T, Vadla Sai Charitha, Sarthak Dwivedi, Kammara Prasad Achari, Arunima Divya, who are engineers from InfobellIT who helped test and develop this benchmark.

REFERENCES

[1] https://spec.org/
[2] https://tpc.org/
[3] Raghunath Nambiar, Tilmann Rabl, Karthik Kulkarni, Michael Frank:
Enhancing Data Generation in TPCx-HS with a Non-uniform Random Distribution. TPCTC: 2015: 94-129
[4] Meikel Poess, Raghunath Nambiar, Karthik Kulkarni, Chinmayi Narasimhadevara, Tilmann Rabl, Hans-Arno Jacobsen: Analysis of TPCx-IoT: The First Industry Standard Benchmark for IoT Gateway Systems. ICDE 2018: 1519-1530
[5] https://www.intel.com/content/www/us/en/developer/articles/technical/accelerate-llama2-ai-hardware-sw-optimizations.html
[6] https://huggingface.co/NousResearch/Llama-2-7b-hf?ref=blog.truefoundry.com
[7] https://github.com/huggingface/text-generation-inference
[8] https://locust.io/
[9] Llama 2: Open Foundation and Fine-Tuned Chat Models - https://arxiv.org/pdf/2307.09288.pdf
[10] https://www.anyscale.com/blog/reproducible-performance-metrics-for-llm-inference
[11] EchoSwift: https://github.com/Infobellit-Solutions-Pvt-Ltd/EchoSwift