Provisioned Throughput Benchmarking

Performance for LLMs is often measured in terms of tokens/second. When configuring production endpoints, it's important to ensure your endpoint is configured to scale to support all the requests your application will send to the endpoint without impacting concurrency. When configuring the scale-out ranges for endpoints deployed with Provisioned Throughput, we've found it easier to reason about the inputs going into your system (which are tokens!).

What are tokens?

Large Language Models (LLMs) read and generate text in terms of what's called a "token". Tokens can be words or sub-words, and the exact rules for splitting text into tokens vary from model to model. For instance, you can use online tools to see how Llama's tokenizer converts words to tokens.

Why measure LLM performance in terms of tokens/second?

Traditionally, model serving endpoints have been configured based on the number of concurrent requests per second (RPS). However, a given LLM inference actually takes different amounts of time based on how many tokens are passed in and how many it generates, which can be imbalanced across requests. Therefore, accurately measuring your workload scaleout requires measuring in terms of the content of your request - tokens. These can be different depending on your use case:

- **Varying lengths of input contexts:** While some requests may involve only a few tokens (e.g., a short question), others may involve hundreds or even thousands of tokens (e.g., a long document for summarization). This variability makes it challenging to configure a serving endpoint based on RPS alone, as it does not account for the varying processing demands of the different requests.
- **Varying lengths of output depending on use case:** Different use cases for LLMs can lead to vastly different output lengths. Generating output tokens is actually the most time intensive part of LLM inference, so this can dramatically impact throughput. For example, summarization involves shorter, pithier responses, but text generation (like writing articles or product descriptions) can generate much longer answers.

How should you interpret tokens/second ranges in Provisioned Throughput?

Provisioned throughput endpoints are configured in terms of a range of tokens/second that you can send to the endpoint. The endpoint will scale up and down to handle the load of your production application. You will be charged per hour based on the range of tokens/second your endpoint is scaled to. There are 2 important things to consider:

1. **How we measure tokens/second performance of the LLM**
   We benchmarked against a workload representing summarization tasks that are common for **retrieval-augmented generation style use cases**. Specifically, the workload we benchmarked to consists of:
   - 2048 input tokens
   - 256 output tokens

- The token ranges displayed *combine* input and output token throughput and optimize for balancing throughput and latency.

  We benchmark that users can send that many tokens/second *concurrently* to the endpoint (i.e multiple requests hitting the endpoint at the same time), which more accurately represents how one would actually use the endpoint in production.

2. **How does autoscaling work?**

Model Serving features a rapid autoscaling system that will scale the underlying compute to meet the tokens/second demand of your application. We scale provisioned throughput up in chunks of tokens/second, so you will be charged for additional units of provisioned throughput only when you're using them.

The best way to know what tokens/second range on your Provisioned Throughput endpoint works for your use case is to perform a load test with a representative dataset, and Databricks support is happy to help run this.

LLM Inference Details

LLMs perform inference in a two-step process: "prefill", where the tokens in the input prompt are processed in parallel, and "decoding", where text is generated one 'token' at a time in an autoregressive manner. Each generated token is appended to the input and fed back into the model to generate the next token. Generation stops when the LLM outputs a special stop token or when a user-defined condition is met.

Measuring performance for LLMs is nuanced, and we detail our philosophy on benchmarking in this blog post.

If you have overall inference latency targets, here are some useful heuristics for evaluating models:

- Output length dominates overall response latency: For average latency, you can usually just take your expected/max output token length and multiply it by an overall average time per output token for the model.
- Input length has a substantial impact on the required memory to process requests - the addition of 512 input tokens increases latency less than the production of 8 additional output tokens in the MPT models.

We detail our thoughts about LLM performance benchmarking in much more detail here