2024년 09월 12일

AUTHOR

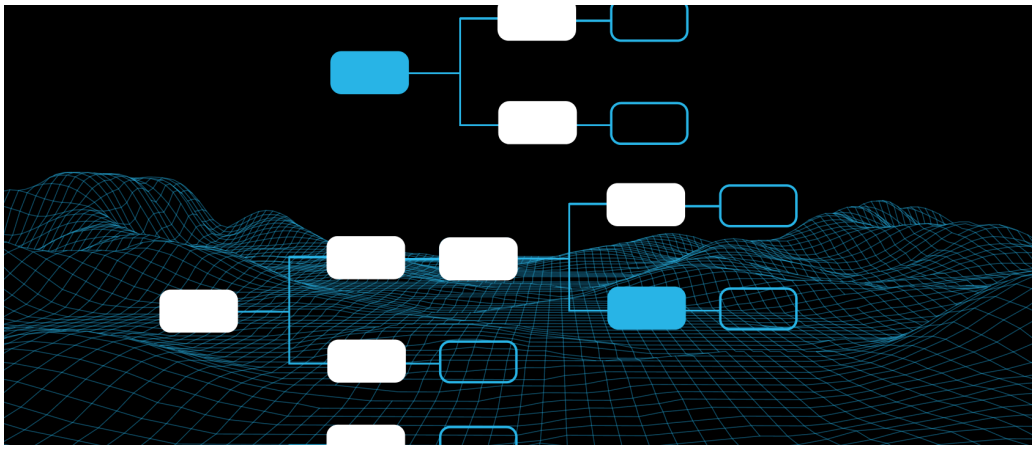
Vincent Chan

Hao Zhang

Flex Wang

LLM Interactive Workloads: Optimizing GPU Capacity for Interactive and Batch Workloads

Gen AI



At Snowflake we offer a wide variety of LLM-powered features in Cortex AI, including Cortex LLM Functions, Snowflake Copilot and our recently released Cortex Analyst, now in public preview. While some products, like Cortex LLM Functions, rely on high throughput and are less latency-constrained, products like Snowflake Copilot and Cortex Analyst emphasize low latency. In this article, we will explore how to utilize our GPU machines to efficiently serve both types of workloads.

The Need for Interactive Workloads

In our previous system, inference was optimized for throughput by continuously batching as many sequences together as could fit in GPU memory for execution. By doing this, we can optimize throughput by saturating the GPU memory.

However, just optimizing throughput is not feasible when serving interactive workloads that require low latency due to high latencies at the request level. For interactive LLM applications, two key metrics are the time to first token (TTFT) and time per output token (TPOT). Low time to first token is vital for interactive workloads because it reduces the

end user's idle time. Time per output token should be at least as fast as most humans can read, which is about 10 tokens per second.

Handling interactive workloads

When optimizing the batch size of requests, fully using available compute, as measured by Teraflops (TFLOPS), is a strong proxy for maximized throughput. By running larger batch sizes, there is less data movement required, and ultimately yields higher throughput due to better GPU utilization.

There is no such thing as a free lunch, and AI inference is no different. Larger batches increase utilized TFLOPS, which means higher GPU utilization, but this will also increase average TPOT. Larger batches lead to larger matrix computations, which increase the time to the next output token at the request level.

AUTHOR

Vincent Chan

Hao Zhang

Flex Wang

Figure 1. Chart that showcases the relationship between batch sizes, throughput and latency. These benchmarks were run with Llama 3 70B on 8xA100 GPUs with vLLM, with approximately 1,024 input tokens and 144 output tokens.

As we can observe in Figure 1, running requests with lower batch sizes decreases the average request latency to just a few seconds, but it consequently reduces the throughput and thus, the utilization of the GPU. On the other hand, increasing the batch size maximizes throughput, but it also drastically increases the average request latency.

One solution here is to limit the maximum batch size to a smaller number, as this achieves the low latency needed to satisfy interactive requests.

Dedicated machines to interactive or offline batch inference?

Simply stated, the needs of batch and interactive systems are at odds. Better overall throughput via large batches are incompatible with interactive workloads, while fully interactive workloads lead to poor utilization of GPU resources. To handle the different types of workloads, we can choose to dedicate some machines purely for interactive requests and other machines only for offline batch inference. This solution, however, is an inefficient use of GPU resources since interactive workloads receive bursts of traffic throughout the day instead of a consistent stream of work that needs to be completed.

AUTHOR

Vincent Chan

Hao Zhang

Flex Wang

To make better use of our GPU resources, we can utilize the same machine to handle both interactive and offline batch inference. We know that limiting the max batch size will give us the desired latency we want for interactive requests, and we can dynamically limit the batch size for interactive requests. Although the max batch size is limited due to the interactive requests, interactive requests don't always meet that max batch size. This effectively means we have free 'slots' that can be filled by requests from our offline batch inference requests to fulfill the remaining batch and not entirely degrade throughput for noninteractive requests.

Improving TTFT and TPOT

To improve TTFT, we assign a higher priority to interactive requests over batch inference workloads that are less latency-constrained. Interactive requests preempt noninteractive requests in the scheduling queue, moving them to the front for execution. Prioritizing interactive requests in this way shortens the time they need to wait for TTFT.

Once an inference engine receives an interactive request, noninteractive requests are evicted from execution until the number of concurrent running requests is equal to or below the maximum batch size specified for the model during interactive requests. The evicted noninteractive requests are re-queued for execution on another inference engine, if available. By evicting running noninteractive requests, we ensure that there is space for the execution of the interactive request, thereby reducing the TTFT. Additionally, since the batch size is restricted during runtime, TPOT is also improved.

Since noninteractive requests are evicted from an engine during this process, we cache the previously output tokens to save the work done before running the evicted request on another engine. Since the noninteractive request is a little more latency-relaxed, we eat the cost of prefilling the input and output tokens from the evicted request on another engine.

AUTHOR

Vincent Chan

Hao Zhang

Flex Wang

Figure 2. Initial state of the inference engine when loaded with noninteractive requests, then upon receiving two interactive requests, running noninteractive requests are evicted to make room for the interactive requests where the batch size is limited.

For noninteractive workloads that are evicted from execution, we attempt to find another inference engine replica to handle this workload. Interactive requests are monitored and are taken into account when determining whether to either swap/scale in more replicas to handle the given requests, such that throughput is not heavily degraded during interactive requests.

Maintaining consistent TPOT with continuous batching

As requests batched with interactive workloads finish, new incoming requests take their place as space becomes available for another request to run concurrently. Currently, prefill is prioritized to batch in as many requests as possible during the decoding step. Since both prefill and decoding occur on the same GPU machines, decoding is blocked by the prefill step. This can result in choppy token streaming for interactive requests instead of the smooth experience expected from chat applications.

To mitigate this issue, we enabled chunked prefill (see papers: DeepSpeed-FastGen: High-throughput Text Generation for LLMs via MII and DeepSpeed-Inference and SARATHI: Efficient LLM Inference by

Piggybacking Decodes with Chunked Prefills) at the inference engine layer. Instead of prefilling requests entirely before performing the decoding step, only a chunk of the request is prefilled up to a cap, reducing the wait time for requests in the decoding queue. The trade-off is a slight degradation in prefilling, due to chunked prefill capping the prefill sequence at every step. Still, ultimately, we can expect TPOT to have a much smaller variance, resulting in a better end-user experience for streaming.

AUTHOR

Vincent Chan

Hao Zhang

Flex Wang

Figure 3. From SARATHI: Efficient LLM Inference by Piggybacking Decodes with Chunked Prefills, the arithmetic intensity is shown for both prefill and decode. We can observe that prefill is more susceptible to be compute-bound due to heavy matrix computations, compared to decoding.

Alternatively, other methods can keep TPOT consistent with continuous batching, such as prefill-decode disaggregation. The gist of this method is that the prefill and decoding steps are split onto different machines since prefilling is compute-bound and decoding is memory bandwidth-bound. With prefill-decode disaggregation, TPOT and TTFT can be optimized for "goodput" to serve both interactive and offline batch workloads.

To enhance the end-user experience for streaming applications, implementing chunked prefill can significantly improve the consistency of TPOT. These strategies optimize GPU resource usage and offer a flexible solution that adapts to the varying needs of both interactive and offline batch workloads. By reducing the variance in TPOT and minimizing the blocking of decoding processes, these methods help ensure smoother, real-time interactions in chat applications. Moreover, their ability to efficiently handle diverse workloads makes them well-suited for a wide range of applications, contributing to a more efficient and user-friendly inference system.

Summary

At Snowflake, our customers expect an uncompromised mix of efficiency and low latency, and their LLM workloads are no different. Unbounded

by the traditional distinction between interactive and batch workloads, our shared approach lets us handle both high throughput and low latency tasks efficiently on our GPU machines. By dynamically capping the batch size for interactive tasks based on the workload, we keep latency low without heavily degrading throughput for other tasks. Giving priority to interactive tasks in the scheduling queue and kicking out noninteractive tasks when needed also improves the time to first token and token output rate.

This is essential for Snowflake as it allows us to deliver a seamless user experience across our suite of LLM products in Cortex AI. Whether it's fast responses in Snowflake Copilot, real-time insights with Cortex Analyst, or handling heavy data processing with Cortex LLM Functions, we demonstrate how we can serve products with different use cases while optimizing GPU resource utilization.

AUTHOR

Vincent Chan

Hao Zhang

Flex Wang

SHARE

RELATED CONTENT

SHARE

2024년 09월 05일

Model Hotswapping: Optimizing AI Infrastructure and Enhancing LLM Efficiency

At Snowflake, we support customer AI workloads by offering a diverse range of open source and proprietary large language models (LLMs). As we expand our first-party product offerings in Snowflake…

2024년 07월 23일

Achieve Low-Latency and High-Throughput Inference with Meta's Llama 3.1 405B using Snowflake's

2024년 07월 11일

Snowflake Arctic Cookbook Series: A Deep Dive into LLM Evaluation Standards

What level of astronomy

Full Details

Optimized AI Stack

Meta's Llama 3.1 405B represents a groundbreaking milestone for open-weight large language models (LLMs), pushing...

Expand your knowledge

knowledge should an accountant have? Today's evaluations of LLMs assess their...

Find Out How

Vincent Chan

Hao Zhang

Flex Wang

START YOUR
30-DAY FREE TRIAL

START NOW

SHARE

플랫폼 개요

아키텍처

데이터 애플리케이션

데이터 마켓플레이스

SNOWFLAKE 파트너 네트워크

지원 및 서비스

회사

문의하기

Sign up for Snowflake Communications

diana.shaw@snow

United States

By submitting this form, I understand Snowflake will process my personal information in accordance with their Privacy Notice. Additionally, I consent to my information being shared with Event Partners in accordance with Snowflake's Event Privacy Notice. I understand I may withdraw my consent or update my preferences here at any time.

SUBSCRIBE NOW

AUTHOR

Vincent Chan

Hao Zhang

Flex Wang

SHARE