Summary of “Optimizing Generative Inference”

Xueshen Liu (liuxs), Juechu Dong (joydong)

Problem and Motivation

The deployment of large generative models faces critical challenges in computational demand and cost, which limit their broader application. Efficient serving systems are essential for leveraging these models' capabilities without incurring prohibitive expenses or performance compromises. This lecture we discussed three systematic optimization ideas in different aspects: computation scheduling, memory management and request caching.

- ORCA: This paper specifically addresses the inefficiency in serving transformer-based models due to the high computational cost and latency, emphasizing the need for optimized **task scheduling** to enhance throughput and reduce operational costs.
- PagedAttention: Focuses on optimizing GPU **memory management** for large language models, where traditional memory management techniques fall short in handling dynamic memory requirements efficiently.
- NIRVANA: Targets the latency challenges in diffusion models, proposing **caching** strategies to balance the trade-off between speed and quality by exploiting the commonality of prompts across sessions.

Related Works

Scheduling Granularity: Request-level scheduling and batching, often led to unbalanced workloads and increased latency. Techniques like dynamic batching and model parallelism have been explored, yet they fail to fully address the nuanced challenges of transformer model serving, such as the alignment of queries for efficient batching and the variable computational demands of different model layers.
Memory Management: This approach addresses the challenges of efficiently handling the key-value (KV) cache memory which dynamically grows and shrinks as the model generates new tokens, a common issue in existing systems that results in significant memory waste due to internal/external fragmentation and reservation.
Caching for Diffusion Models: The denoising process of image generation is computation intensive and suffers from long latency, but over 50% of iterations are used to refine the image. To reduce latency by reusing the generated images, previous works retrieve directly based on input prompt similarity, which suffers in quality and flexibility.

Solution Overview

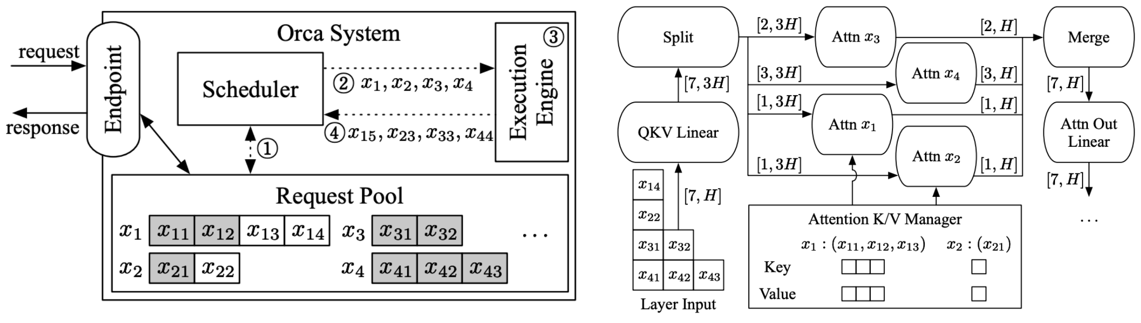
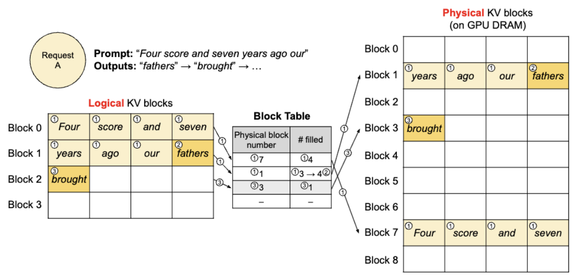


Fig1. Architecture of Iteration-level Scheduling & Selective Batching

ORCA proposes a novel distributed serving system specifically tailored for transformer-based generative models. The key insight guiding their solution is the adoption of iteration-level scheduling utilizing a request pool to manage incoming queries. To address the challenge of batching requests in a per-iteration manner—which requires aligning queries of the same phase and length—ORCA introduces selective batching. This method involves flattening data at the linear layer and then reformatting it for the attention operation on a per-query basis. This technique allows for more efficient utilization of computational resources and reduces overhead.


Request A
Prompt: "Four score and seven years ago our"
Outputs: "fathers" → "brought" → ...
Logical KV blocks
Block Table
Physical block number
# filled
Physical KV blocks (on GPU DRAM)
Four
score
and
seven
years
ago
our
fathers
brought

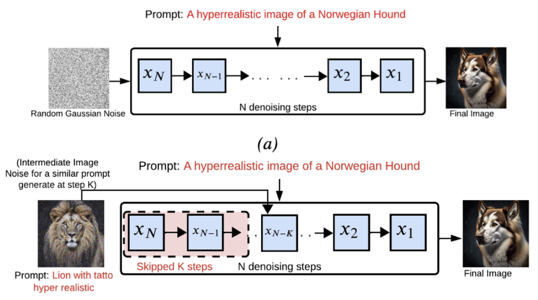

Prompt: A hyperrealistic image of a Norwegian Hound
Random Gaussian Noise
N denoising steps
Final Image
(a)
(Intermediate Image Noise for a similar prompt generate at step K)
Prompt: Lion with tatto hyper realistic
Skipped K steps

Fig2. Illustration of PagedAttention Fig3. Vanilla Diffusion vs. Approximate Caching

PagedAttention divides the cache into non-contiguous blocks, resembling pages in virtual memory, to mitigate issues of memory fragmentation and wastage. It dynamically allocates memory on demand, allowing for efficient utilization of GPU memory. The authors proposed vLLM based on pagedAttention using two optimizations / policies: 1) scheduling & preemption: PagedAttention implements a preemption strategy whereby entire blocks are preempted if needed. This preemptive approach frees up space for other requests that have run out of memory. The selection of which block to preempt typically follows a first-come-first-serve policy; 2) prefix sharing: Another technique introduced is prefix sharing, where memory blocks associated with the same service and pre-prompt are shared. This optimizes memory usage by reducing redundancy and maximizing sharing among similar requests. vLLM minimizes waste in KV cache utilization and supports various LLM architectures while efficiently managing variable sequence lengths.

NIRVANA includes several important components. 1) Embedding Generator: This generates embeddings for text prompts, grouping similar prompts in the embedding space and predicting cache hits.2) Cache Retrieval: Retrieves intermediate noisy states from specific embeddings. 3) Cache Maintenance: The cache is limited to 1TB and utilizes an LCBFU eviction policy to optimize compute savings by caching all intermediate states for a given prompt, reducing latency variance.

Limitations

Both Orca and vLLM are systematic optimization approaches by building schedulers on host, which will introduce additional complexity due to sophisticated management policy. Additionally, the system's effectiveness might vary depending on the specific characteristics of the workload and the transformer model being served, such as the length of input context, frequency of queries, and the amount of parameters of models, which may require fine-tuning of scheduling strategy (e.g. batch & block size) based on user data to maximize efficiency. On the other hand, vLLM is optimized for KV-cache-like workload and its granularity is fixed at block level, meaning that even if a residual part of the cache cannot fill up a block, the entire page is swapped when the system runs out of blocks. Things are similar for NIRVANA, where the effectiveness of the cache strategy may be specific to the model's pre-training, implying a lack of generalizability.

Future Research Directions

Future works for Orca and vLLM could be applying more adaptive scheduling algorithms that can accommodate varying workload patterns. Improve the scalability of the scheduler especially for long context with multiple GPU nodes. NIRVANA may be able to combine with other ML optimization strategies and similarity selection algorithms to improve quality and reduce latency. In general, people may also study more control strategies of operating systems to inspire the design of efficient and effective ML systems.

Summary of Class Discussion

Rate of Arrival Evaluation:
Typically follows a Poisson distribution, with the mean of the distribution altered to evaluate throughput and latency sensitivity of request-based systems.
Overhead of TLB Table Lookup/Virtual Context Switching:
While the overhead is negligible for relatively large models with longer iterations, it might become significant for smaller models with quicker iterations, such as those with small attention layers.
Performance under Extremely Long Context Length:
System performance degradation primarily stems from GPU memory depletion and the need for frequent swapping. The system does not account for scenarios where

the context length surpasses single-node memory capacity, leading to no swap space.

Sharing Prefix

Traditional assumptions of shared prefixes being exactly the same overlook practical scenarios where similar but slightly different prefixes exist, resulting in missed opportunities for efficient memory sharing.

Tolerance to Service Interruption in Cloud Services:

While the first-come-first-served policy partially manages interruptions, critical memory shortages inevitably degrade system performance.

Impact of Block Size on End-to-End Latency:

Small block sizes lead to a large mapping table, causing additional memory mapping overheads, diminishing system efficiency, while large block sizes result in increased internal fragmentation, also affecting efficiency.

Leverage of User Data

Leveraging user data, particularly in identifying similarities between user prompts across sessions, can help optimize systems. There's speculation about the competitive advantage this could offer large companies like OpenAI and Adobe, the latter being the entity behind the discussed paper. This highlights the broader implications of the research beyond technical improvements.