Paper Review – ORCA: A Distributed Serving System for Transformer-Based Generative Models

Summary: The paper presents ORCA, a distributed serving system designed to make large Transformer-based generative models (like GPT-3) faster and more efficient during inference. Modern language models generate one token at a time across many iterations, but most existing systems, such as NVIDIA's FasterTransformer or Triton, treat each request as a single atomic unit, batching and running it to completion. This rigid approach leads to wasted computation when requests finish early and long waiting times for those that arrive late. ORCA reimagines the serving pipeline by breaking the strict request boundary. It introduces iteration-level scheduling, where the scheduler executes the model one decoding step at a time, allowing the system to continuously mix new and ongoing requests. Along with this, ORCA uses selective batching, a clever compromise that applies batching only to layers with large, shared parameters (like linear and normalization layers), while executing the Attention module individually for each request to avoid shape conflicts. The algorithm coordinating this process ensures that each iteration uses available GPU slots efficiently while keeping per-request key–value states persistent across iterations. Architecturally, ORCA separates its control plane (for scheduling decisions) from the data plane (for GPU computation) and scales across nodes using both inter-layer and intra-layer parallelism. This hybrid design allows the system to handle models with hundreds of billions of parameters. Implemented in C++ using CUDA, gRPC, and NCCL, ORCA was evaluated on Azure NDv4 VMs with 8xA100 GPUs (1.6 Tb/s interconnect) using GPT models from 13B to 341B parameters, showing up to 36.9x higher throughput than FasterTransformer at similar latency, demonstrating its strong scalability and efficiency for large-scale inference.

Evaluation: This paper targets one of the most important bottlenecks in large-scale model deployment: efficient inference for autoregressive Transformers. The authors clearly address why current request-level batching schemes are not suited for iterative decoding workloads and offer a principled redesign of both the computation model and scheduling algorithm. The core ideas, iteration-level scheduling and selective batching, are novel and practical, and the implementation demonstrates solid engineering judgment, particularly in how it balances batching flexibility with GPU utilization. The experimental evaluation is extensive, covering both controlled microbenchmarks and full end-to-end serving workloads under dynamic traffic, comparing latency, throughput, and scalability against NVIDIA's FasterTransformer across models (13B-341B) under varying batch sizes and GPU counts. The use of models ranging from tens to hundreds of billions of parameters makes the evidence credible and scalable. Although the reliance on synthetic traces slightly limits real-world validation, the evaluation convincingly shows that ORCA consistently outperforms FasterTransformer in throughput and latency, especially under high-load and multi-GPU configurations. In terms of clarity and structure, the paper is clearly written and well organized.

Main Takeaways

1. Iteration-level control is essential for autoregressive workloads: handling each decoding step separately prevents idle GPUs and slashes latency.
2. Selective batching is a clean trade-off, keeping most of the performance benefits of batching while enabling flexible scheduling across variable-length requests.
3. Tighter scheduler-engine coupling can unlock new efficiencies, suggesting that the traditional separation between serving frameworks and execution engines may need to be rethought for next-generation generative models.

Strengths:

1. Robust distributed design supporting inter/intra-layer parallelism for >100B-scale models.
2. Strong experimental methodology spanning microbenchmarks and end-to-end serving.

Weaknesses:

1. Selective batching leaves potential efficiency untapped for Attention-heavy workloads.
2. Manual tuning of max_bs introduces operational complexity for practitioners.

Discussion:

1. Would iteration-level scheduling generalize cleanly to other autoregressive domains like image generation or audio synthesis?