HW8

Tguo67

Question(s):
Read the related materials, and answer the following questions:
1. What benefits and limits does DistServe propose for its design?
2. By comparing the evaluation results in DistServe and TaiChi, what differences are there? What can be the potential reasons?
3. How does *Throughput is Not All You Need* argue for its idea? Do you think the arguments hold in real serving scenarios? What are your reasons?

Related materials:
DistServe: Disaggregating Prefill and Decoding for Goodput-optimized Large Language Model ServingLinks to an external site.
Prefill-Decode Aggregation or Disaggregation? Unifying Both for Goodput-Optimized LLM ServingLinks to an external site.
Throughput is Not All You Need: Maximizing Goodput in LLM Serving using Prefill-Decode DisaggregationLinks to an external site.

Answers:

1. What benefits and limits does DistServe propose for its design?

Benefits:

- **Goodput optimization**: DistServe disaggregates *prefill* (large, parallelizable batch operations) and *decode* (sequential, latency-sensitive operations) across specialized servers. This prevents resource underutilization caused by coupling the two, thus improving effective throughput per latency constraint (goodput).
- **Resource efficiency:** Prefill servers are compute-intensive, while decode servers are memory/latency-intensive; separating them allows heterogeneous resource allocation and higher overall utilization.
- **Scalability & flexibility:** It supports dynamic routing and scaling for each stage independently, adapting to varying request loads.

Limits:

- **Network overhead:** Disaggregation introduces inter-server communication latency and potential bandwidth bottlenecks for transferring KV caches or intermediate results.
- **Complex scheduling and coordination:** Requires fine-grained routing to balance loads and minimize end-to-end latency.
- **Prefill–decode coupling for short prompts:** For very small inputs, disaggregation overhead may outweigh its benefits, so performance gains are workload-dependent.

2. By comparing the evaluation results in DistServe and TaiChi, what differences are there? What can be the potential reasons?

- DistServe shows larger throughput and goodput improvements in high-load scenarios, while TaiChi (which unifies prefill and decode dynamically) achieves better average latency and tail-latency stability under mixed workloads.
- DistServe uses strict disaggregation, optimizing for large batch prefill efficiency but paying extra inter-stage communication cost. But TaiChi instead unifies both paths under one scheduler and dynamically adjusts the allocation between prefill and decode. This reduces network overhead and idle time when requests are short or bursty.
- TaiChi benefits more from adaptive aggregation in realistic, heterogeneous query mixes.
- DistServe's advantage appears mainly under heavy, long-context, high-throughput loads, where compute saturation dominates and disaggregation helps.

3. How does *Throughput is Not All You Need* argue for its idea? Do you think the arguments hold in real serving scenarios? What are your reasons?

The paper argues that goodput (timely completions within SLA) matters more than raw throughput and throughput can be high even when many requests miss latency targets.

In real world, I think it is true for interactive or multi-tenant serving, maybe not that relevant for batch/offline inference.