MACIEJ BESTA, LORENZO PALEARI

06.08.2025

Graphs & LLMs: Synergy

Recent advances

- AAAI submissions (2x)
- NeurIPS rebuttals (3x)
- TPAMI success
- TPAMI submission

KV Caching: System Prompt Compression

- Compress the KV cache of the system prompt
  - System prompt may be huge (10k+ tokens for Claude)
  - This would give many advantages
    - Use heavy slow compression algorithm
    - It could make the pipeline much faster, while also saving lots of space at runtime
  - However: Problematic when adding user parameters, like tools or other dymanic variable, they will change mainly the positional bedding of part of the system prompt

KV Caching: Other Various Attempts

- Attempts at tuning various mechanisms. Passed over.
  - Not enough performance premises
  - Not too novel
- Asynchronous Compression: Compress older input from user at run-time in an asynchronous way.
  - It is similar to how people tend to forget older things in memory
  - Beginning to investigate when Lorenzo is back.

KV Caching: Clarification

- Asking for clarification on why MLA is not too great
  - MLA seems to be faster than other methods.
  - This is highlighted is the fact that with MLA (after certain conditions, i.e., harnessing Experts and Compression) becomes compute bound and not memory bound.
  - Compute power is rising faster than bandwidth and memory capacity - even if it slows down a little but, we should be ok?

- Discussion

High-Performance RLMs: Sharing Weights

Problem

- In Reinforcement Learning with Human Feedback (RLHF), training and generation (rollout) GPUs operate in separate phases.
- After each training step, updated weights must be transferred from training GPUs to generation GPUs.
- This leads to **full-model weight transfers** (hundreds of MBs to GBs) over PCIe, NVLink, or network, introducing significant latency.

Goal

- Allow generation GPUs to **directly access model weights in shared memory** maintained by training GPUs, eliminating explicit weight transfers.

High-Performance RLMs: Design Overview

Architecture Design

- **Shared Memory Pool**: Training GPUs maintain model parameters in a designated memory region accessible to generation GPUs
- **Version Control**: Implement atomic version counters to track weight updates and ensure consistency
- **Access Patterns**: Generation GPUs perform read-only access while training GPUs have exclusive write access

High-Performance RLMs: Design Overview

Architecture Design

- **Shared Memory Pool**: Training GPUs maintain model parameters in a designated memory region accessible to generation GPUs
- **Version Control**: Implement atomic version counters to track weight updates and ensure consistency
- **Access Patterns**: Generation GPUs perform read-only access while training GPUs have exclusive write access