Locality-Aware Beam Scheduling for Efficient Test-Time Compute with a Consumer-grade GPU

Hsing-Ti Wang, Hung-Tso Shiao, Chia-Lin Yang

Speaker: **Hung-Tso Shiao**

國立臺灣大學
National Taiwan University

Outline

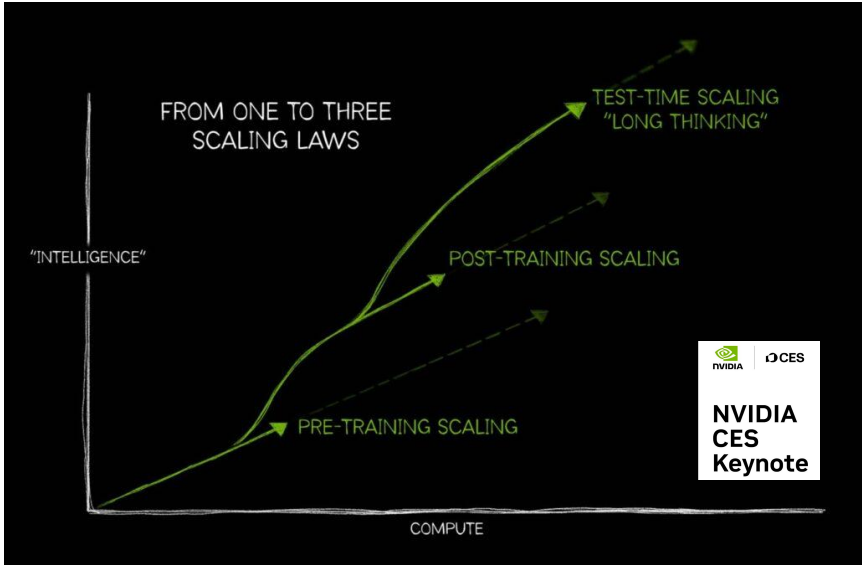
- Background & Motivation
- Proposed Method
- Evaluation
- Conclusion

Outline

- Background & Motivation
- Proposed Method
- Evaluation
- Conclusion

Test-Time Compute

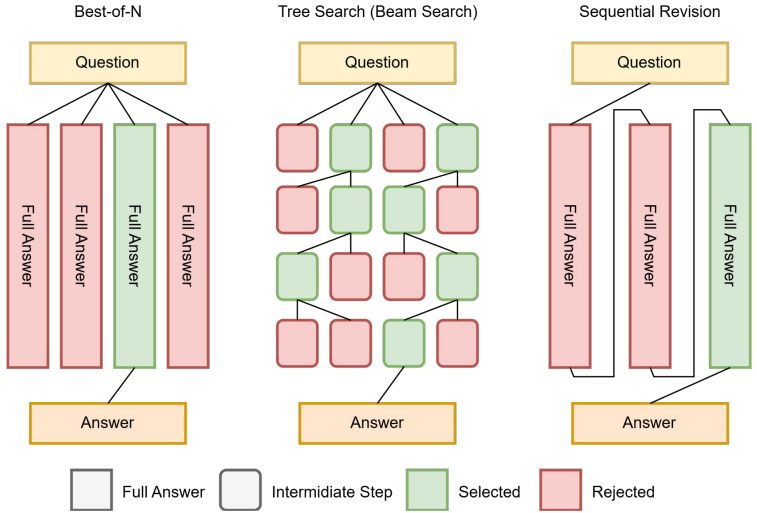
- For a long time, inference in language models was designed to be fixed-cost. But recent breakthroughs show that giving models more compute at test time can significantly improve reasoning.
- This approach—known as Test-Time Compute—is emerging as a key innovation, already adopted by leading labs.



Jensen Huang (NVIDIA), CES 2025 Keynote

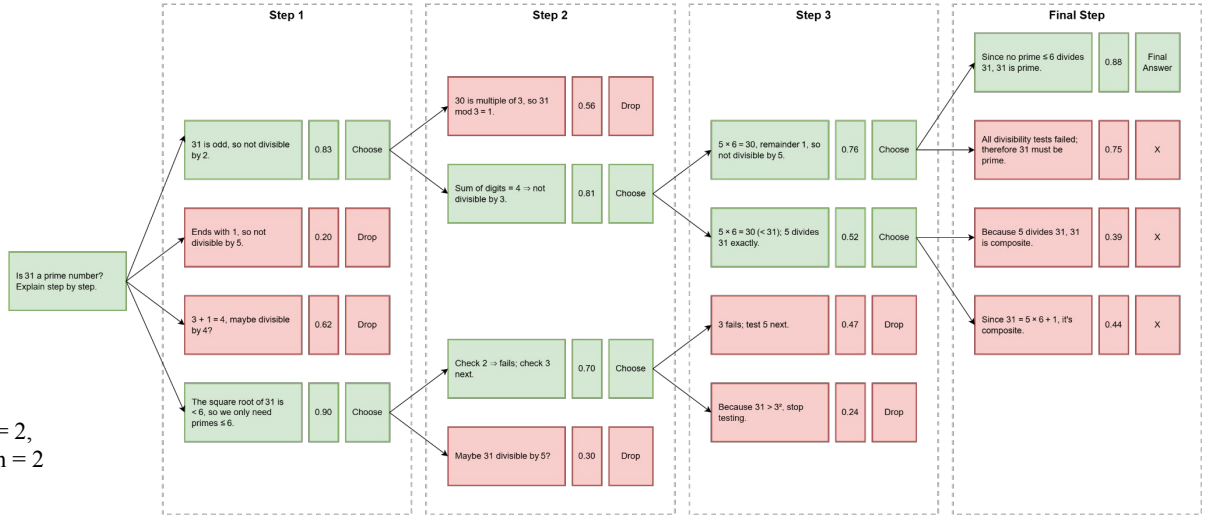
Overview of Test-Time Compute

- Test-Time Compute Strategies:
  - **Best-of-N Sampling**: Independently generates N full answers, and selects the best one.
  - **Step-wise Tree Search**: Builds reasoning step by step by exploring and picking the best branch.
  - **Sequential Revision**: Produces an initial draft and refines it through multiple rounds of self-correction.
- Among these methods, Tree Search provides a good balance between effectiveness and practicality, and therefore the focus of our work."



Overview of Step-Wise Beam Search Algorithm

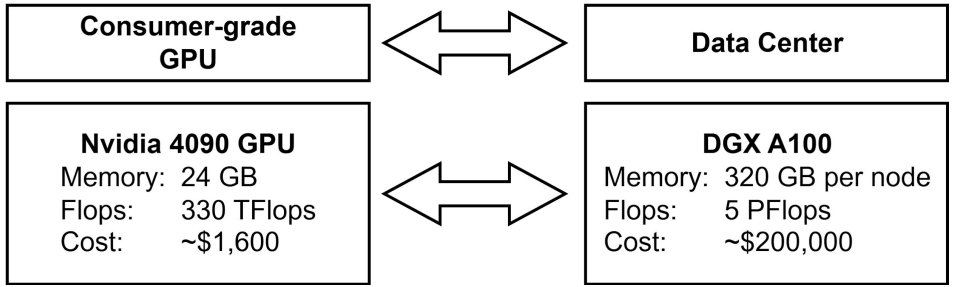
- Each step in Step-wise Beam Search consists of three phases:
  - **Expansion Phase**: Generate `beam_size × beam_width` candidates from the current beam.
  - **Rating Phase**: Score all the candidate outputs.
  - **Selection Phase**: Select `beam_size` candidates to form the next beam.
- While TTC opens new opportunities for better inference quality, it also introduces new challenges for consumer-grade GPUs



beam_size = 2,
beam_width = 2

LLM Inference with Consumer-grade GPU

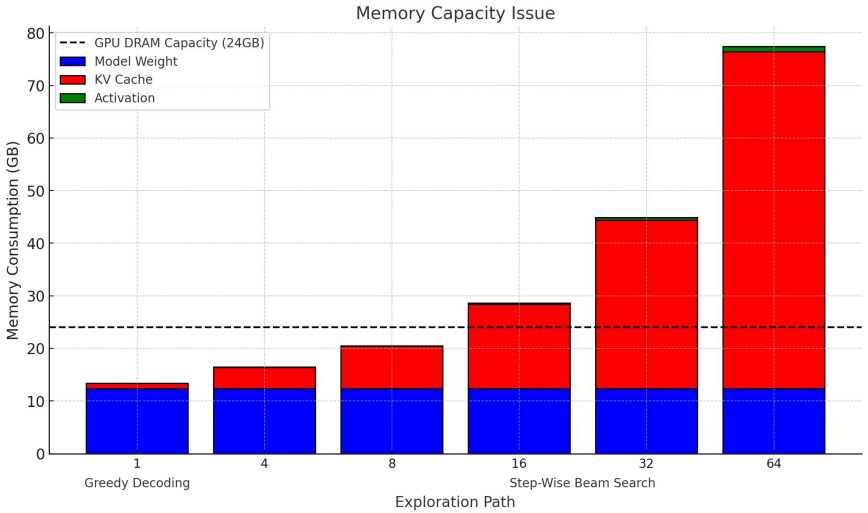
- As LLMs gain wider adoption, there is growing interest in running them on consumer-grade GPUs. Local inference offers three key advantages: better data privacy, customization, and reduced hardware cost.
- However, consumer-grade GPUs face two main constraints when running large language models: limited memory capacity and comparatively lower compute power.



NVIDIA DGX A100 Datasheet

Test-Time Compute Memory Issue

- Existing discussions of LLM inference on consumer-grade GPUs have primarily focused on single-batch scenarios, assuming a relatively small KV cache memory compare to model weights.
- However, under test-time compute, the KV cache grows linearly with the exploration space, introducing substantial memory footprint as a new and critical issue.

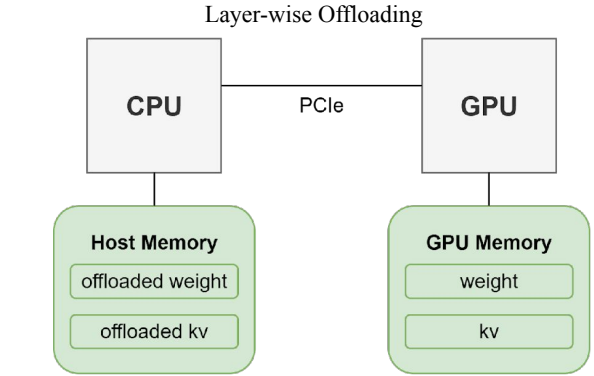


$$\mathrm{KV}_{\mathrm{total}} = \mathrm{KV}_{\mathrm{per\ path}} \times \mathrm{N}_{\mathrm{Beam}}$$

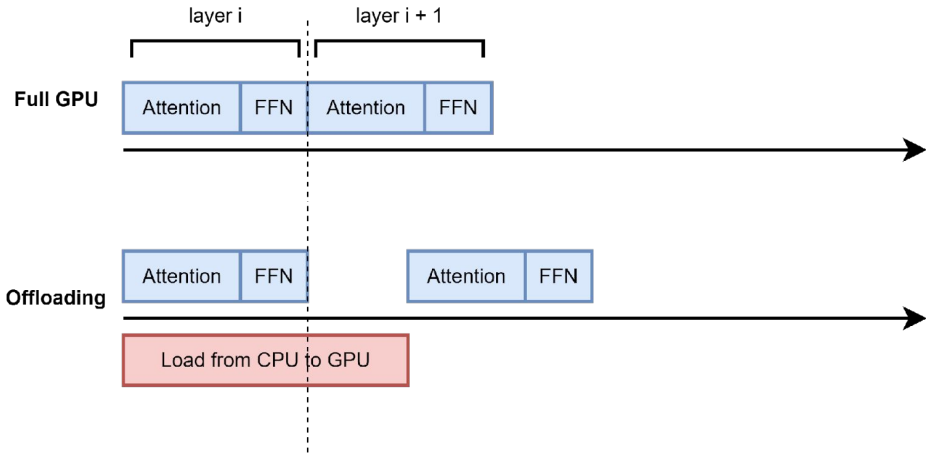
KV cache emerges as the dominant memory cost—an often overlooked bottleneck in prior consumer-grade GPU LLM inference work.

Layer-wise Offloading

- Modern LLM inference systems such as DeepSpeed and FlexGen support offloading model weights or KV cache to CPU memory.



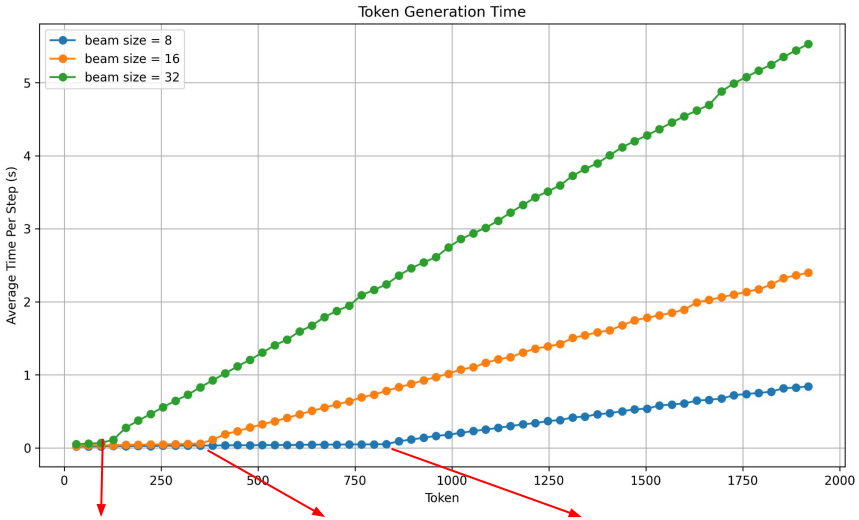
Both weights and KV cache are measured at the per-layer level.



DeepSpeed-inference: Enabling efficient inference of transformer models at unprecedented scale, 2022, SC
FlexGen: High-Throughput Generative Inference of Large Language Models with a Single GPU, 2023, ICML

Performance Impact

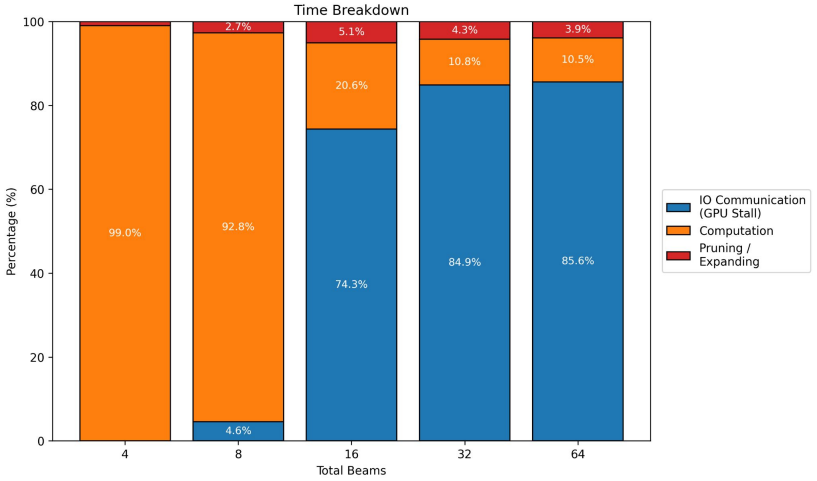
- Offloading KV cache causes severe performance degradation due to high I/O overhead.
- As the number of beam paths increases, I/O overhead—unavoidable GPU stalls due to data transfers—gradually dominates the runtime. At large beam counts, it accounts for ~85% of total latency.



Start offloading at token 115.

Start offloading at token 357.

Start offloading at token 841.



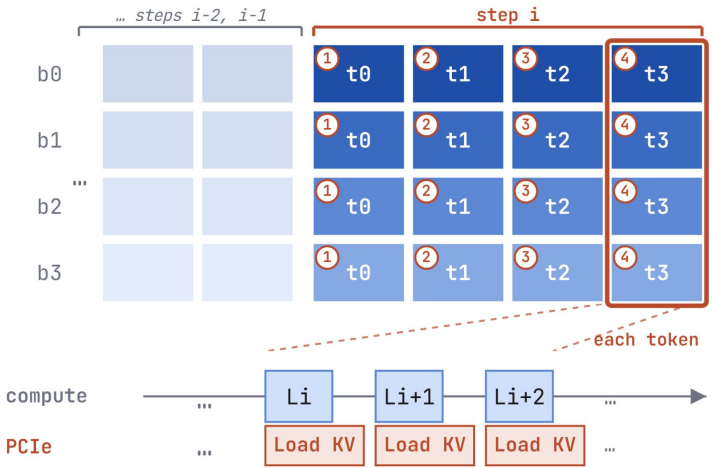
Outline

- Introduction & Motivation
- Proposed Method
- Evaluation
- Conclusion

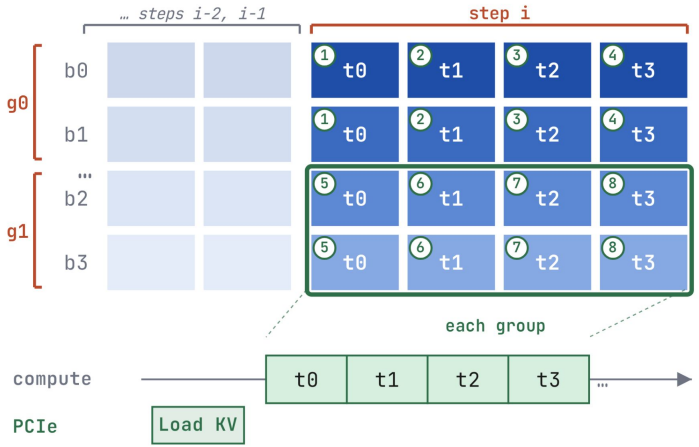
Insight 1 - Inter-token locality

- Generating token i+1 needs the KV of all earlier tokens in the sequence.

Token-by-Token (Baseline)

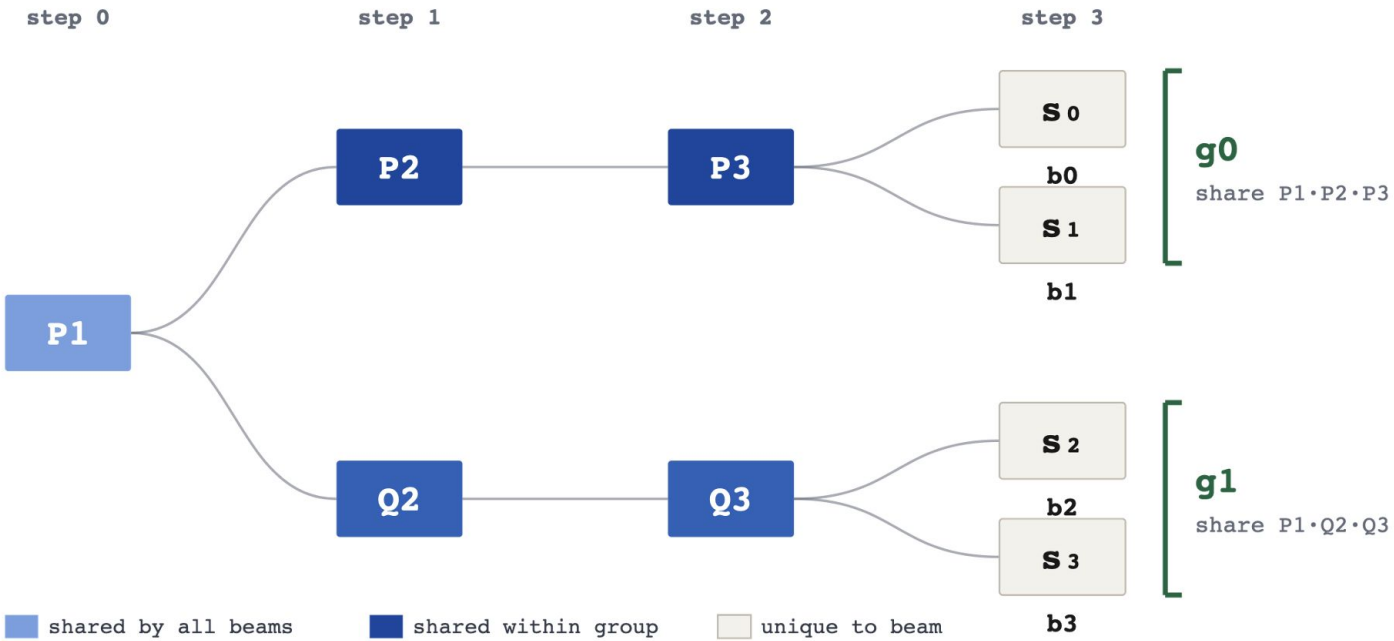


Group-by-Group (Ours)



Insight 2 - Inter-beam locality

- Beam search produces a tree — sibling beams share long prefixes from their common ancestor.
- By grouping siblings means the shared KV is loaded once per group, not once per beam.



Analytical Model

- Traditional decoding processes all beams layer-by-layer and token-by-token. This results in frequent KV cache swapping, as layers are repeatedly reloaded for each token.
- Our method processes each beam group through the entire decoding step before switching, avoiding frequent KV swaps across tokens.

Token-by-Token Strategy.

$$\mathrm{L}_{\mathrm{inGPU}}(s) = \min\left(\mathrm{N}_{\mathrm{layer}}, \left\lfloor \frac{\mathrm{Mem}_{\mathrm{GPU}}}{\mathrm{KV}_{\mathrm{layer}}(s)} \right\rfloor\right) \quad (1)$$

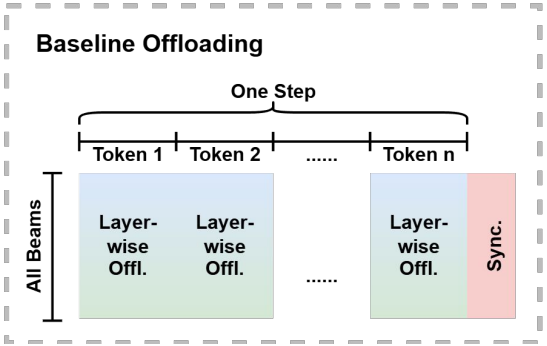
$$\mathrm{DM}_{\mathrm{token}}(s) = (\mathrm{N}_{\mathrm{layer}} - \mathrm{L}_{\mathrm{inGPU}}(s)) \times \mathrm{KV}_{\mathrm{layer}}(s) \quad (2)$$

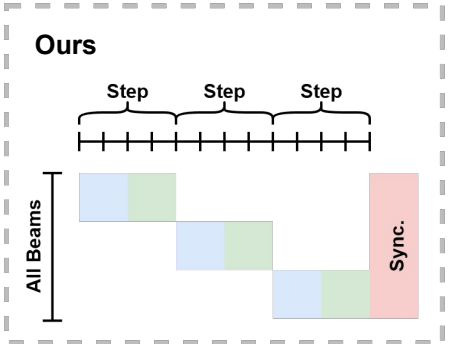
$$\mathrm{DM}_{\mathrm{req}} = \sum_{i=0}^{\mathrm{N}_{\mathrm{gen}}-1} \mathrm{DM}_{\mathrm{token}}(i + \mathrm{S}_{\mathrm{prompt}}) \quad (3)$$

Beam Scheduling Strategy.

$$\mathrm{DM}_{\mathrm{step}}(s) = \mathrm{N}_{\mathrm{beam}} \times \mathrm{KV}_{\mathrm{beam}}(s) \quad (4)$$

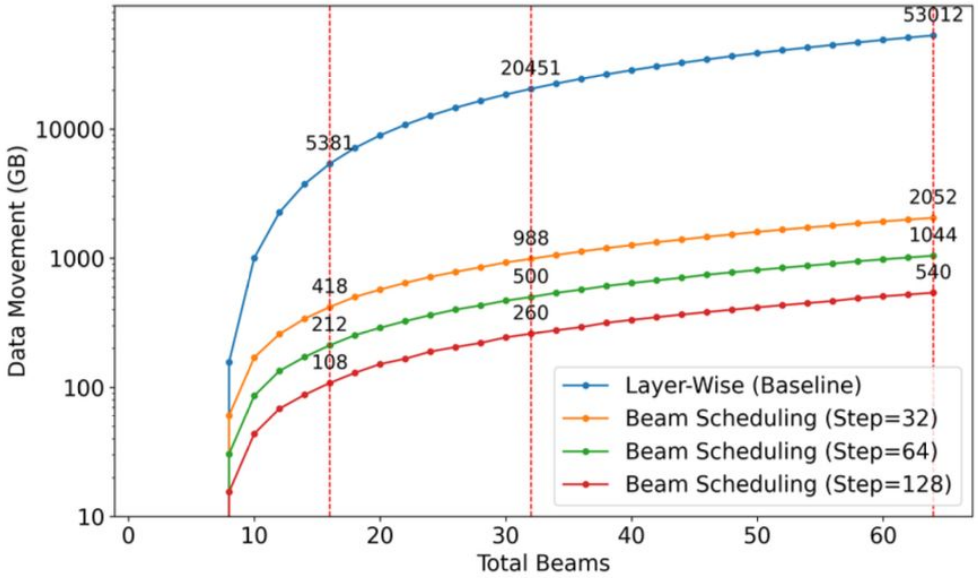
$$\mathrm{DM}_{\mathrm{req}} = \sum_{i=0}^{\left\lceil \frac{\mathrm{N}_{\mathrm{gen}}}{\mathrm{N}_{\mathrm{step}}} \right\rceil - 1} \mathrm{DM}_{\mathrm{step}}\left(i \times \mathrm{N}_{\mathrm{step}} + \mathrm{S}_{\mathrm{prompt}}\right) \quad (5)$$





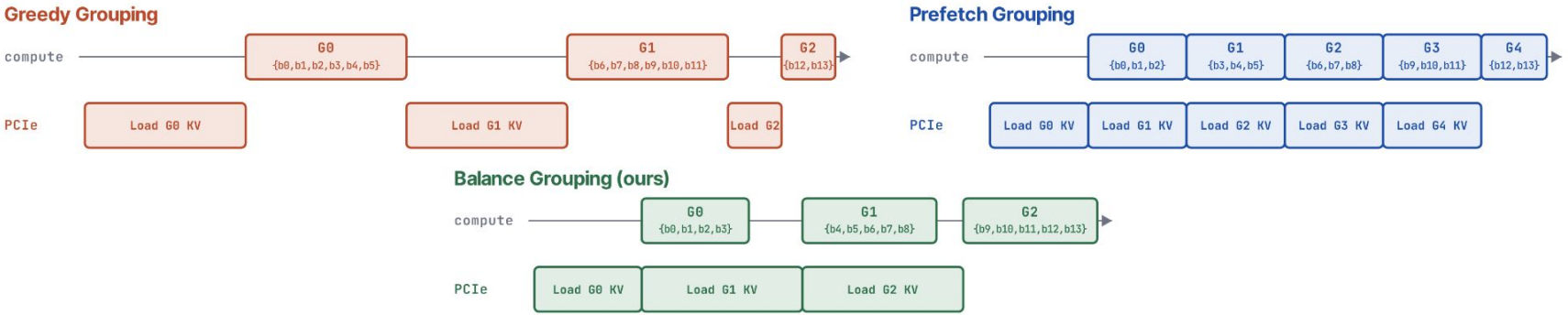
Theoretical Modeling of Data Movement

- Our method significantly reduces KV cache transfer overhead, with greater savings observed under larger beam sizes and longer decoding steps.



Beam Grouping and Prefetching

- Compute rounds vs. Prefetch overlap (GPU capacity = 6, total beams = 14)
  - Greedy — Pack to the ceiling. {6, 6, 2} runs in 3 rounds, but puts loads on the critical path.
  - Prefetch — Reserve half the GPU. {3, 3, 3, 3, 2} hides loads, but doubles the rounds.
- We propose Balanced grouping — {4, 5, 5}: 3 rounds like Greedy, with 1–2 beams of headroom per round to overlap.



Outline

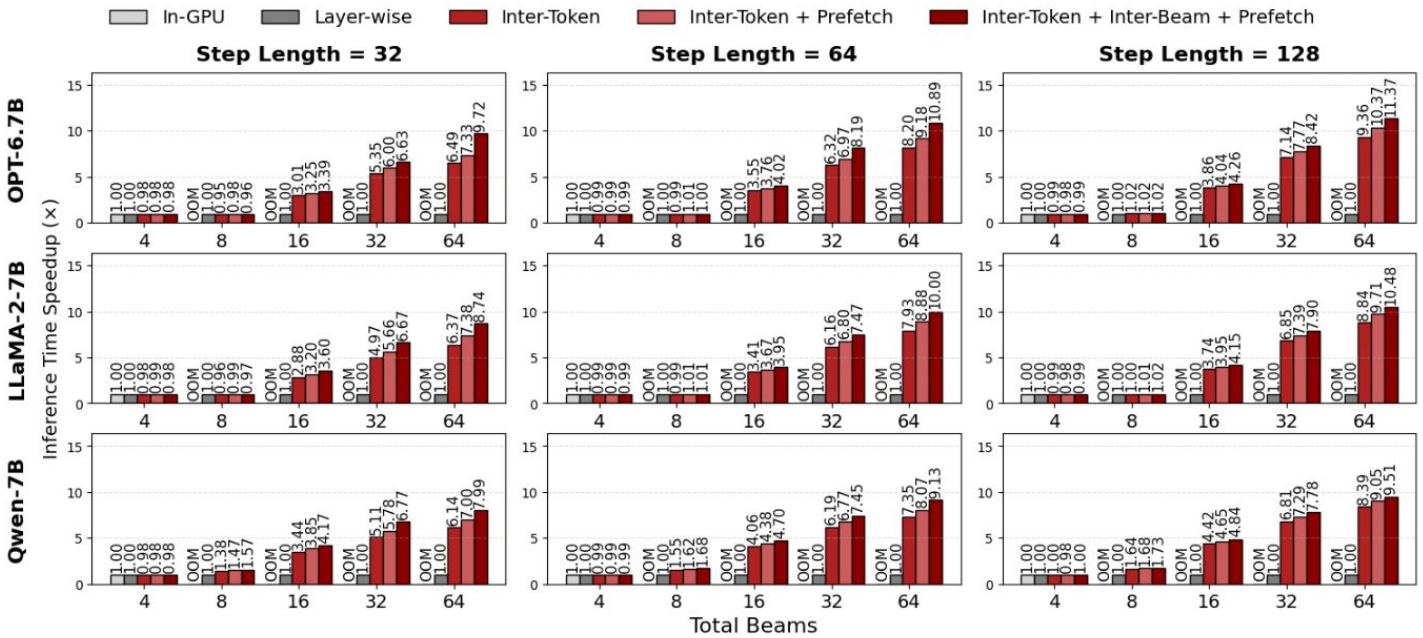
- Introduction & Motivation
- Proposed Method
- Evaluation
- Conclusion

Experimental Setup

- **Hardware configuration**
  - CPU: Intel(R) Core(TM) i9-14900KF
  - GPU: NVIDIA GeForce RTX 4090 24GB
  - Host Memory: 128 GB DDR4 RAM
  - PCIe Interface: Gen4.0 x8
- **Model**
  - OPT-6.7B
  - Llama-2-7B
  - Qwen-7B

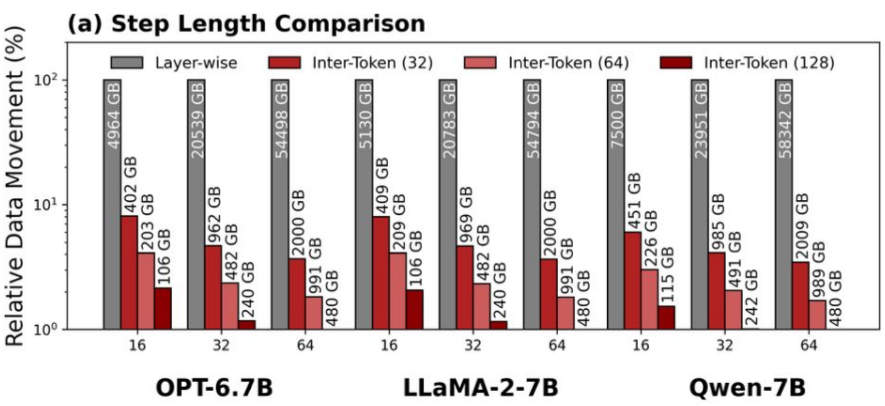
End-to-End Inference Performance

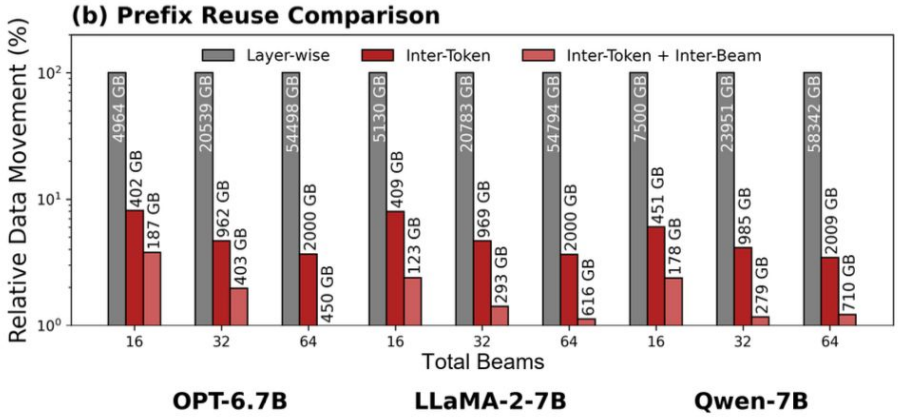
- Our method achieves speedup of 3.39×–9.72× on OPT-6.7B , 3.60×–8.74× on LLaMA-2-7B, and 4.17×–7.99× on Qwen-7B.



KV Cache Movement Analysis

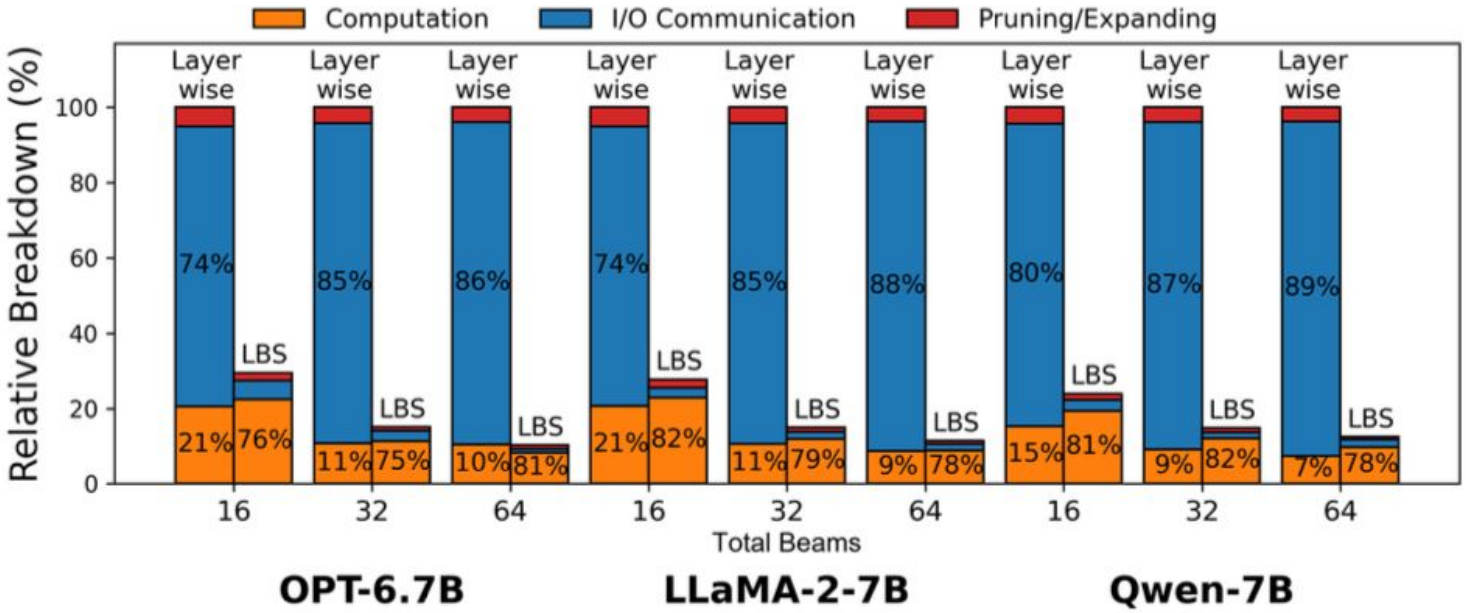
- Compared with the layer-wise baseline, exploiting inter-token locality cuts transfers to below 5% of the original volume, which match our analytical model.
- Enabling inter-beam locality yields more than a 2× reduction in transfer volume.





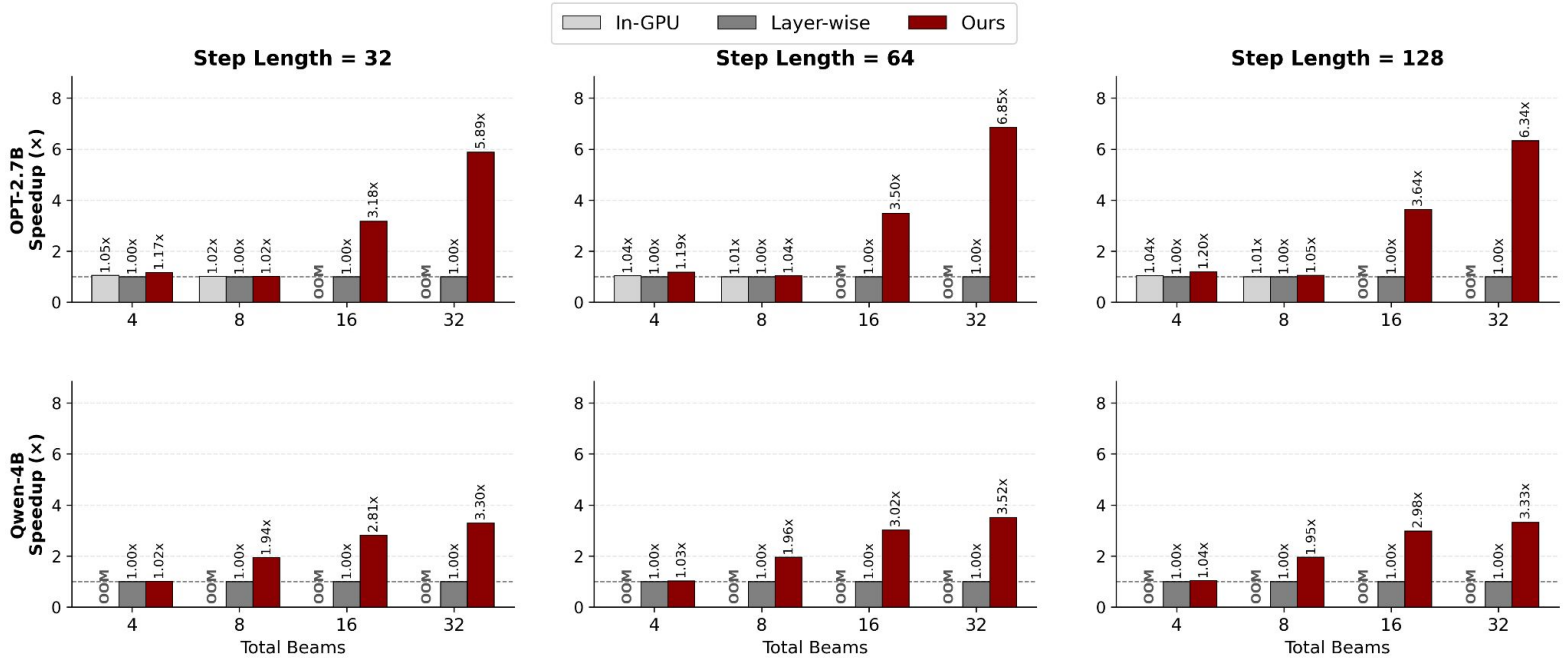
Decoding Time Breakdown and I/O Analysis

- our locality-aware beam scheduling reduces I/O overhead to below 15%, restoring computation as the dominant cost.



Evaluation on Other GPU Architectures

- The results on RTX 3080 is consistent to the results on the RTX 4090, our method achieving up to a 6.85×speedup on OPT-2.7B and 3.52× on Qwen-4B.



Outline

- Introduction & Motivation
- Proposed Method
- Evaluation
- Conclusion

Conclusion

- First comprehensive analysis of TTC workloads, showing KV cache as the major memory bottleneck (>70% memory at 32 beams, >80% at 64 beams), making layer-wise offloading inefficient.
- Identify two key locality patterns in step-wise beam search: inter-token locality and inter-beam locality, enabling efficient KV cache reuse.
- Propose Locality-aware Beam Scheduling with balanced grouping to overlap transfer and computation, reducing KV cache transfers by >95%.
- Achieve consistent end-to-end latency speedups across models: 3.39×–9.72× (OPT-6.7B), 3.60×–8.74× (LLaMA-2-7B), and 4.17×–7.99× (Qwen-7B).

Thank For Listening !