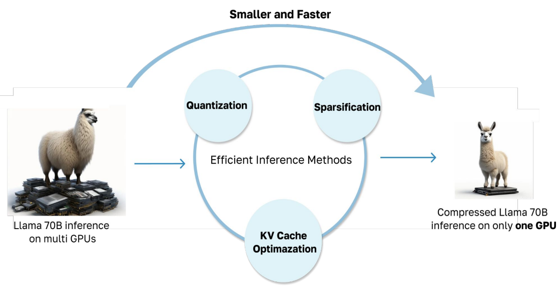
Efficient Inference of Large Language Models on a Single GPU

STUDENTS: MIHIR PATHAK, ANNIE HUANG, YIFEI SHEN, KATHERINE TANG, ARVIND RAMAN, TIANYI LI

Introduction

- **Problems Statement:** Large Language Models (LLMs), like LLaMA 3-70B, require over 140 GB of memory in FP16, far exceeding the capacity of cost-effective GPUs like NVIDIA A40 (48 GB). Long-context inference further amplifies memory and compute demands, resulting in high latency and low throughput[1].
- **Objective:** Enable efficient inference of LLMs on a single A40 GPU by:
  - Supporting ≥10k-token inputs
  - Ensuring ≤5% accuracy degradation
  - Achieving ≥10 tokens/sec throughput

Methodology



KV CACHE OPTIMIZATION

- Supports sparse caching, using only the most relevant KV pairs during decoding to reduce compute.
- Enables KV quantization to compress memory footprint.
- Save memory usage and improve inference throughput, especially for long-context autoregressive tasks.

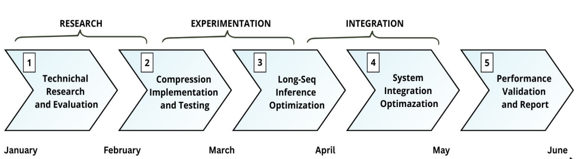
Post-Training Quantization

- Converts weights/activations to low-precision (e.g., FP16 → INT4) post-training.
- Improves inference speed and reduces memory/bandwidth usage.
- Preserves accuracy using calibration data.

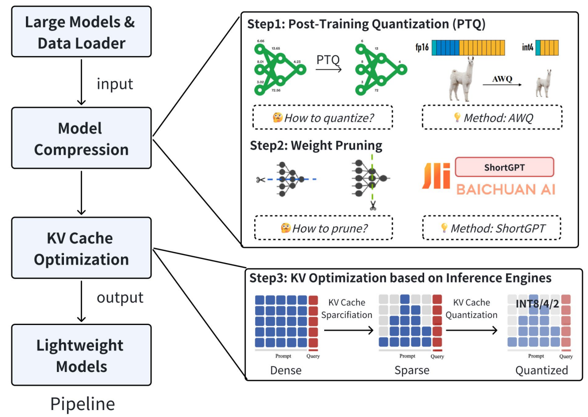
Weight Pruning

- Removes redundant or low-importance weights.
- Enables sparse computation for faster and lighter inference.
- Can be structured (e.g., channel or block pruning) or unstructured, with trade-offs between accuracy and hardware compatibility.

Milestones



Optimization Pipeline



Technology Combinations

We experimented with a variety of compression techniques:

Quantization:

- **QTIP** SOTA int2 weight-quantization method using incoherent processing and Trellis-based codebooks. In our tests, it **reduces 70B model size 132GB to 20GB**, with WikiTexts PPL 3.59 to 7.19. General-task accuracy drop remains under 20%.
- **OmniQuant** Introduces loss-aware weight clipping to selectively constrain critical weights, enhancing quantization robustness.
- **AWQ** / **GPTQ** Widely adopted for inference due to ease of integration and strong compatibility with LLaMA-3 models.

Weight Pruning / Sparsification:

- **Wanda** Prunes low-importance weights per neuron using activation scores, supporting N:M sparsity for acceleration. PPL(8B, 50% sparsity):8.28 to 11.97. Throughput: 35 tokens/s.
- **SliceGPT** Uses singular value pruning to remove weight matrix rows/columns, boosting efficiency but adding structural complexity that hinders integration. PPL (8B, 50% sparsity): 8.28 to 99.76. Throughput(4000 in, 256 out): 9.49 to 29.18 tokens/s.
- **ShortGPT** Prunes less critical attention blocks with controllable accuracy trade-offs, validated through empirical analysis. Our evaluation results shown in Fig 2.

KV Cache Optimization:

- **PyramidKV**: Reduces memory via layer-wise shrinking of KV cache, preserving accuracy by retaining key cache information.
- **KIVI** / **KVQuant**: Compress KV cache to reduce memory with minimal impact on performance. Our results shown in Fig 3.

Fig.1

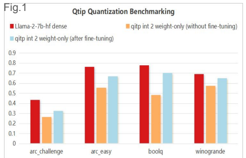


Fig.2

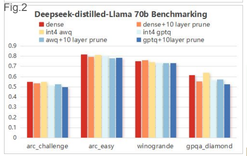
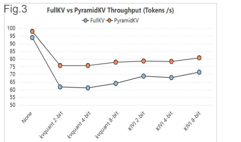


Fig.3



Final Results & Conclusion

Method Selection & Integration: We investigated various compression techniques to enable efficient inference on a single A40 GPU. After evaluating several options, we converged on a unified pipeline focused on compression effectiveness and ease of integration.

Quantization - AWQ: Selected over QTIP, SmoothQuant, GPTQ for seamless framework support, better performance-efficiency trade-off, and superior pruning synergy.

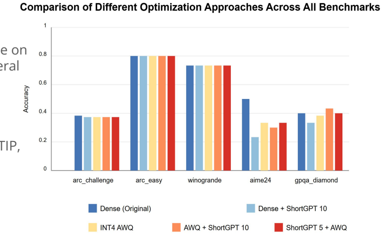
Pruning - ShortGPT: Outperformed Wanda and SliceGPT with full attention block pruning, minimal structural change, and negligible degradation (≤10 layers).

KV Cache - PyramidKV: Enables 10K+ token inference via layer-wise importance decay, retaining critical entries vs. uniform truncation.

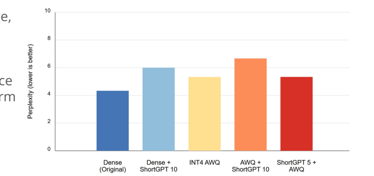
Why This Combination? AWQ enables efficient low-bit inference; ShortGPT enhances efficiency with minimal disruption and easy integration; PyramidKV further addresses kv cache memory bottlenecks. The complementary design enables real-world deployment on constrained hardware.

Key Takeaways:
1) Seamless integration outweighs standalone performance.
2) Cross-technique compatibility enables unified pipelines.
3) Practical methods outperform complex alternatives.

Comparison of Different Optimization Approaches Across All Benchmarks

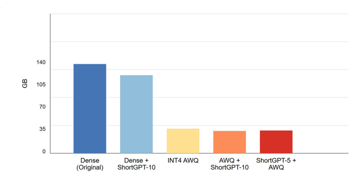


Comparison of Different Optimization Approaches - Perplexity



Lower PPL values indicate better language model performance

Comparison of Model Size Reduction Techniques



Up to 75% Size Reduction with AWQ + ShortGPT-10

Model Size	Throughput	Input Context	Accuracy
75% ↓	**98.02 tok/s**	**10K+**	**Preserved**
132GB → 33GB	~10× our goal	token inputs	benchmark scores
AWQ int4 + shortGPT-10	*PyramidKV (no quant)*	*Long-context support*	*≤5% accuracy loss*

Future Work

- Investigate performance with ultra-long contexts (>50k tokens)
- Compress Larger Models (405B) & Other model architectures (e.g.MOE)
- Extend optimizations to consumer GPUs (NVIDIA RTX 4090 24GB)
- Integrate KV cache optimization methods into famous inference engines, e.g. SGLang/Dynamo
- Explore INT1/INT2 quantization with minimal accuracy loss
- Integrate structured sparsity like Wanda on quantized model to further improve throughput
- Develop methods to increase accuracy recovery for heavily pruned models (20+ layers)
- Explore combined optimization techniques to maintain quality while improving speed

ELECTRICAL & COMPUTER ENGINEERING
UNIVERSITY of WASHINGTON

ADVISORS: Radha Poovendran, Yan Li, Hongyu Yu

SPONSOR: Lenovo Research