Offloading and Quantizing LLaMA-3.1-8B for Resource-Constrained Inference on NVIDIA RTX 3060 Ti

111550165 吳宗樺, 111550087 林炫廷

Abstract

Large language models (LLMs) like **LLaMA-3.1-8B** present substantial memory constraints, limiting deployment on resource-constrained hardware. We explore strategies to enable efficient inference of LLaMA-3.1-8B on an **NVIDIA RTX 3060 Ti GPU**, which lacks sufficient VRAM for standard model execution.

We begin by attempting to load the model in full precision, which immediately triggers **Out-of-Memory (OOM)** errors. To address this, we apply:

- Post-training quantization: Compressing the model by reducing weight precision (e.g., from FP32 to INT8 or INT4) without retraining.
- Mixed-precision quantization: Using different bit-widths (e.g., 8-bit for critical layers, 4-bit elsewhere) to balance efficiency and accuracy.
- Dynamic offloading: Actively transferring non-critical tensors (e.g., key-value cache) between GPU VRAM and CPU RAM/disk to fit large models on limited-memory GPUs.

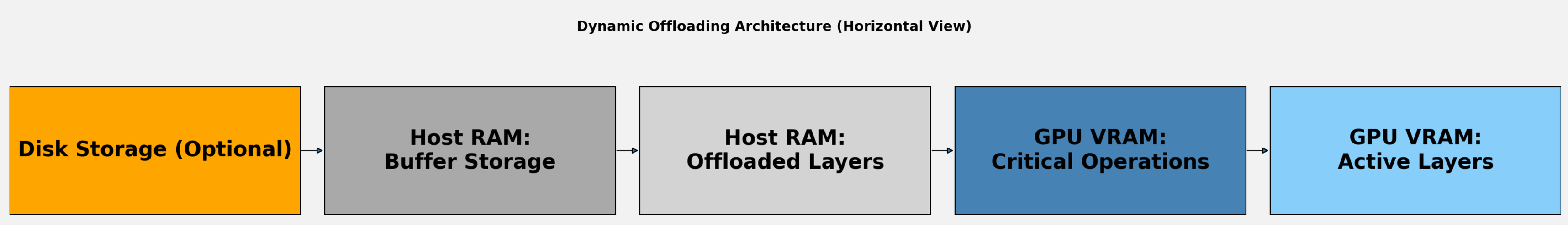
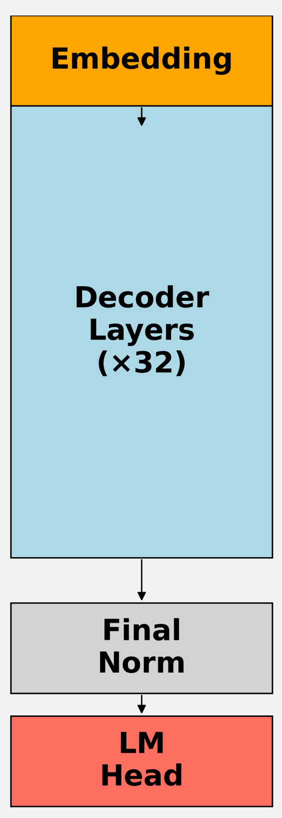


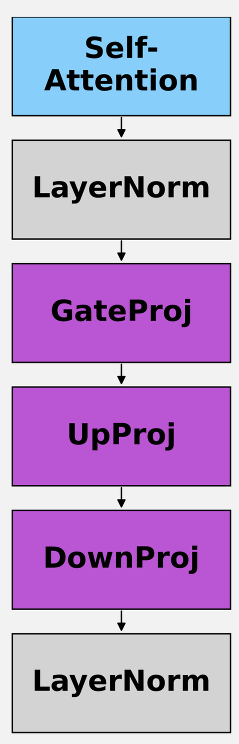
Figure 1: Illustration of dynamic offloading across memory hierarchies during inference.

Model Architecture

- **Model**: LLaMA-3.1-8B
- **Architecture**: Transformer-based architecture with 8 billion parameters
- **Layers**: 32 transformer layers
- **Hidden Size**: 5120
- **Activation Function**: GeLU (Gaussian Error Linear Unit)



(a) The overview of LLaMA-3.1-8B architecture.



(b) Layer-wise parameter distribution in LLaMA-3.1-8B.

Figure 2: LLaMA-3.1-8B architecture and layer-wise parameter distribution.

References

[1] Zhihang Yuan et al., "LLM Inference Unveiled: Survey and Roofline Model Insights," arXiv:2402.16363, 2024.
[2] Amir Gholami et al., "A Survey of Quantization Methods for Efficient Neural Network Inference," arXiv:2103.13630, 2021.
[3] Ying Sheng et al., "FlexGen: High-Throughput Generative Inference of Large Language Models with a Single GPU," arXiv:2303.06865, 2023.
[4] Woosuk Kwon et al., "Efficient Memory Management for Large Language Model Serving with PagedAttention," arXiv:2309.06180, 2023.
[5] Wu et al., Prediction-Difference Quantization (PD-Quant), " arXiv:2212.07048, 2023.

Preliminary Attempt

We attempted to load the full-precision LLaMA-3.1-8B on RTX 3060 Ti immediately triggered CUDA OOM errors, confirming that direct inference is infeasible without optimization.

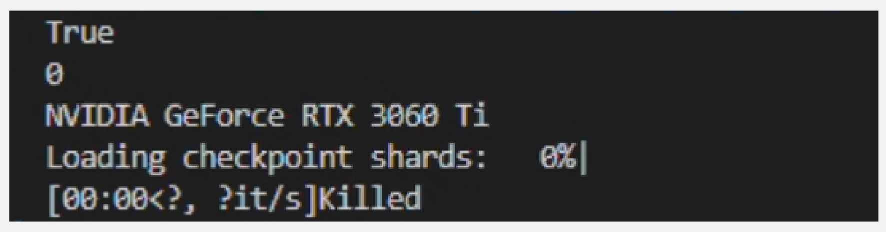


Figure 3: LLaMA-3.1-8B model architecture and parameter distribution.

Experiment

To evaluate the impact of quantization and offloading strategies on Llama-3.1-8B inference, we conduct experiments on an NVIDIA 3060 Ti GPU. We assess performance based on:

- **OOM(Out of memory) Error**: The system (usually CPU or GPU) runs out of available memory to store or compute data
- **Inference Time**: Time taken to process a batch of inputs (in seconds)
- **Perplexity**: Language model evaluation metric (lower is better)

Table 1. Performance metrics for LLaMA-3.1-8B with base configurations.

Configuration	OOM Error	Inference Speed (s)	Perplexity
Full-precision Model (FP32)	Yes	N/A	N/A
Baseline (PTQ with INT8)	No	33.320	7.293

Dynamic Offloading

Table 2. Performance metrics for LLaMA-3.1-8B with offloading configurations.

Configuration	OOM Error	Inference Speed (s)	Perplexity
HuggingFace's Offloading	No	205.508	9075.724
Our Offloading Strategy	No	159.728	7.234

Mixed-Precision Quantization With Offloading

Table 3. Performance metrics for LLaMA-3.1-8B with our configurations.

Configuration	OOM Error	Inference Speed (s)	Perplexity
Mixed Precision + Our Offloading Strategy	No	33.320	7.293
Mixed Precision (8b L4 & L7, 4b elsewhere)	No	13.141	7.291

Conclusion

We explored the feasibility of running the Llama-3.1-8B model on limited GPU resources using quantization and offloading techniques. These results highlight the practicality of combining quantization and offloading for resource-constrained environments.

Future Improvements

We aim to extend our work along several directions:

- Larger Model Experiments
  - Test Llama-3.1-13B model with quantization and offloading.
- Hardware Deployment
  - Re-run all experiments directly on other GPU-insufficient hardware, like embedded systems.
  - Take more different configure and compare their speed, memory usage, and performance.