In-Depth Analysis of Vector Post-Training Quantization (VPTQ) for Large Language Models

October 23, 2024

Abstract

Vector Post-Training Quantization (VPTQ) is an innovative method of quantizing large language models (LLMs), allowing for a significant reduction in model size to extremely low bitness without substantial loss in accuracy. This document examines the working principle of VPTQ, its advantages and disadvantages compared to existing methods, its impact on the field of artificial intelligence, potential business applications, and directions for future development.

1 Introduction

As the size of large language models (LLMs) increases, significant challenges arise in their deployment and inference, related to high memory consumption and computational resources. Weight quantization is one of the key optimization methods that allows reducing model size and accelerating its operation. Vector Post-Training Quantization (VPTQ) represents an advanced approach in the field of quantization, achieving extremely low bitness (down to 2 bits) without the need for additional model training.

2 VPTQ Working Principle

VPTQ uses vector quantization to represent model weights using indices and codebooks (lookup tables). The main idea is as follows:

1. **Splitting weights into vectors:** The weight matrix of the model is split into vectors of fixed length.
2. **Clustering:** Each vector is compared with predefined centroids (cluster centers) in the codebook, and it is assigned the nearest index.
3. **Using second-order optimization:** To minimize quantization errors, second-order optimization is applied, which allows for precise selection of centroids and minimizes the impact of quantization on model performance.
4. **Updating quantization errors:** After quantization, individual vectors are adjusted using the remaining errors, which allows for higher accuracy.

3 Comparative Performance Analysis

We conducted additional tests comparing VPTQ and Ollama implementations of the Qwen2.5 14B model on an A2 16GB GPU. The results show a significant difference in token generation speed between VPTQ and Ollama models.

3.1 VPTQ Results

Model	Tokens	Tokens/second
v8-k256-256-woft	111	5.22
v8-k65536-256-woft	111	5.21
v8-k65536-65536-woft	111	2.10
v16-k65536-65536-woft	111	2.02
v8-k65536-0-woft	111	3.77

Table 1: VPTQ Performance Results

3.2 Ollama Results

Model	Quantization	Tokens/second
qwen2.5:14b-instruct	q6_K	10.06
qwen2.5:14b-instruct	q5_0	14.30

Table 2: Ollama Performance Results

3.3 Analysis of Performance Discrepancy

The slower inference in VPTQ models compared to Ollama can be attributed to several factors:

1. **Kernel and Quantization Optimization**: Ollama utilizes more optimized kernels for inference, significantly increasing token processing speed (up to 205.29 tokens/s in some cases). VPTQ models, on the other hand, use less optimized quantization algorithms, reducing their efficiency in token generation.

2. **Complex VPTQ Architecture**: Models with higher levels of quantization, such as `v8-k65536-65536-woft`, show a noticeable decrease in speed (down to 2 tokens/s). This is because lower bitrates and a larger number of centroids require more computational resources for dequantization and codebook lookup, increasing latency.

3. **Differences in Parallelization Approach**: Ollama may better manage multi-processor computations and parallel processing, which could also explain higher inference speeds. VPTQ is currently not optimized for efficient use of multi-processor systems.

4. **Quantization Model Differences**: VPTQ models show varying results depending on the chosen codebook parameters (e.g., `k256` vs `k65536`), indicating that model configuration can significantly affect performance. Models with larger codebooks may generate higher latencies due to a more complex index table lookup process.

5. **Suboptimal Kernels for Low Bitrates**: VPTQ's kernels for inference at lower bitrates (e.g., 2 or 3 bits) are not optimized, leading to more computations during dequantization and codebook usage. This substantially increases latencies and reduces performance. 4-bit models have more optimized kernels, explaining their higher speed compared to lower bits.

6. **CUDA Configuration Issues**: Some users have encountered problems with CUDA configuration where GPU kernels were not found, leading to a switch to a slower PyTorch-based implementation, further slowing down inference.

7. **Lack of Multi-GPU and KV Cache Optimizations**: The current VPTQ implementation lacks key optimizations for working with multiple GPUs and using KV cache, which could significantly improve performance, especially on larger models.

These findings suggest that the main reasons for VPTQ's slower performance compared to Ollama are unoptimized kernels, a more complex quantization architecture, and fewer parallel computing resources, which increase token generation time. To improve VPTQ's performance, optimizing kernels for lower bitrates, enhancing support for multi-processor systems, and addressing CUDA configuration issues would be beneficial.

4 Comparative Analysis: VPTQ vs Other Quantization Methods

VPTQ demonstrates significant advantages over other quantization methods, including GPTQ, QuIP, and AQLM. Let's compare these methods across key metrics:

4.1 Accuracy at Low Bitwidth

VPTQ excels in maintaining high accuracy even at extremely low bitwidths (down to 2 bits):

- **VPTQ vs GPTQ**: On LLaMA-2 7B at 2-bit quantization, VPTQ achieves a WikiText-2 perplexity of 6.13, while GPTQ results in an unusable model with perplexity over 50.
- **VPTQ vs QuIP**: For LLaMA-2 13B at 2-bit, VPTQ achieves a WikiText-2 perplexity of 5.32, compared to QuIP's 5.35.
- **VPTQ vs AQLM**: On LLaMA-2 70B at 2-bit, VPTQ and AQLM achieve similar perplexities (3.93 vs 3.94), but VPTQ shows better QA accuracy (68.6

4.2 Quantization Time

VPTQ is more efficient in the quantization process:

- **VPTQ vs AQLM**: For LLaMA-2 7B, VPTQ takes 2 hours compared to AQLM's 11.07 hours.
- **VPTQ vs GPTVQ**: For LLaMA-2 13B, VPTQ requires 3.2 hours, while GPTVQ needs 3.7 hours.

4.3 Flexibility and Scalability

VPTQ shows better performance across different model sizes:

- **LLaMA-3 Models**: On LLaMA-3 8B at 2-bit, VPTQ achieves a WikiText-2 perplexity of 9.29, significantly outperforming QuIP (85.1) and GPTQ (210.0).
- **Mistral-7B**: At 2-bit quantization, VPTQ achieves a WikiText-2 perplexity of 5.64, compared to QuIP's 6.02 and AQLM's 6.32.

4.4 Advanced Techniques

VPTQ incorporates several advanced techniques that contribute to its performance:

- **Channel-Independent Second-Order Optimization**: This allows for more granular quantization, reducing errors compared to methods like GPTVQ.
- **Residual Vector Quantization**: Enables better representation of weights, especially at low bitwidths.
- **Outlier Elimination**: Helps in dealing with extreme values, further improving accuracy.

4.5 High Accuracy at Low Bitness

One of the key advantages of VPTQ is its ability to maintain high model accuracy even at extremely low bitness (down to 2 bits). This is achieved through the use of second-order optimization and vector quantization.

4.6 Computational Efficiency

VPTQ significantly accelerates the inference process compared to traditional quantization methods. Due to model compression and optimization of the dequantization process, VPTQ provides an increase in throughput up to 1.6-1.8 times compared to current SOTA methods.

4.7 Reduction in Memory Requirements

VPTQ allows reducing the amount of memory required to store the model to 10-15% of the original volume. This makes it possible to run large models on devices with limited resources, such as modern GPUs with 24 GB of memory.

4.8 Ease of Integration

VPTQ can be easily integrated with existing deep learning frameworks such as PyTorch, which simplifies its application in various projects and research.

5 Disadvantages and Limitations of VPTQ

5.1 High Computational Costs at the Quantization Stage

Despite improved efficiency, the quantization process using VPTQ still requires significant computational resources. For example, quantizing the LLaMA-2 70B model requires 4 A100 GPUs with 80 GB of memory each for 19 hours.

5.2 Dependence on Model Architecture

The effectiveness of VPTQ may vary depending on the model architecture. For example, on LLaMA-3 70B with 2-bit quantization, VPTQ achieves a perplexity of 5.6, while on LLaMA-2 70B - 3.93. This indicates that results may vary depending on the specific model.

5.3 Limited Support for Multilingual Models

Current VPTQ testing has been conducted predominantly on English texts. Additional research is needed to confirm the method's effectiveness in multilingual scenarios.

5.4 Need for Specialized Tuning

To achieve optimal results, VPTQ may require fine-tuning of quantization parameters, such as vector length and codebook size, which can be challenging for users without deep knowledge in the field of quantization.

6 Impact of VPTQ on the Field of Artificial Intelligence

6.1 Resource Optimization

VPTQ helps reduce the size of AI models. This allows for saving memory and computational resources. As a result, more powerful models can be used on existing hardware.

6.2 Expanding Deployment Possibilities

Reducing model size opens up new possibilities for their use on devices with limited resources. This includes some mobile devices and Internet of Things devices. However, it's important to note that full-fledged application on smartphones is still limited.

6.3 Reduction in Infrastructure Costs

Companies providing AI services can reduce infrastructure costs. This is due to reduced memory requirements and increased system throughput.

6.4 Contribution to Sustainable Development

Reducing model size leads to lower energy consumption. This contributes to environmental sustainability and reduces the carbon footprint associated with large language models.

7 Business Applications and Use Cases of VPTQ

7.1 Optimization of AI Infrastructure

Companies providing cloud AI services can use VPTQ to reduce the costs of storing and processing large models. This allows offering more affordable solutions to clients.

7.2 Potential for Mobile AI Applications

Compressing models to 2-8 GB opens up prospects for future integration of powerful language models into mobile devices. However, at the moment, this remains an area of active research and development.

7.3 Internet of Things (IoT)

Devices with limited computational resources can use VPTQ to perform some AI tasks on-site. This can increase response speed and protect data privacy.

7.4 Education and Research

Reducing resource requirements makes VPTQ a useful tool for educational institutions and research centers. This allows them to work with large models without the need to invest in expensive infrastructure.

8 Future Work and Research Directions in VPTQ

8.1 Expanding Language Support

It is necessary to test VPTQ on models trained in various languages. This will help confirm its effectiveness in multilingual scenarios.

8.2 Integration with Hardware Accelerators

Work on optimizing VPTQ for special hardware accelerators, such as TPUs and FPGAs, can improve performance and reduce energy consumption.

8.3 Support for a Wider Range of Model Architectures

Expanding VPTQ support for different model architectures will allow the method to be used in more applications. This will increase its versatility and applicability.

9 Conclusion

Vector Post-Training Quantization (VPTQ) represents a significant advancement in model compression for large language models. By employing advanced techniques such as second-order optimization and residual vector quantization, VPTQ achieves extreme low-bitwidth quantization while maintaining high model performance. Key advantages of VPTQ include:

- Achieving 2-bit quantization with minimal accuracy loss
- Improved inference speed and memory efficiency
- Potential for deployment in resource-constrained environments

Despite challenges like high initial computational costs, VPTQ's impact on AI is substantial. It opens new possibilities for deploying powerful models on devices with limited resources. As research progresses, we can expect further refinements, including improved multilingual support and hardware optimization, contributing to more efficient and accessible AI systems in the future.