Hardware-Optimized Retrieval-Augmented Generation Using Parallel Hybrid Retrieval and 4-Bit Quantization for Multi-Domain Q&A

Roman M. Richard * and Alonica R. Villanueva

Posted Date: 9 June 2026

doi: 10.20944/preprints202606.0596.v1

Keywords: generative AI; large language models; small language models; sustainable computing

 doi:10.20944/preprints202606.0596.v1

Article

Hardware-Optimized Retrieval-Augmented Generation Using Parallel Hybrid Retrieval and 4-Bit Quantization for Multi-Domain Q&A

Roman M. Richard [1,2,*] and Alonica R. Villanueva [1]

[1] Graduate Programs, Technological Institute of the Philippines, Quezon City, Philippines

[2] Computer Engineering Department, Technological Institute of the Philippines, Quezon City, Philippines

* Correspondence: romanrichard@ieee.org

Abstract

Modern QA systems pair Large Language Models (LLMs) with Retrieval-Augmented Generation (RAG) to boost factuality, but the pipeline is GPU- and memory-intensive. In this paper, we contribute: (1) the implementation of RAG that utilizes an LLM quantized to 4-bit with a parallel hybrid-retrieval algorithm that uses a small language model (SLM)-based cross-encoder; and (2) the benchmarking of a full-precision (FP32) RAG, a 4-bit quantized LLM (Q-RAG), and parallelized and quantized RAG (QP-RAG) on a 300-query, multi-domain dataset with over 500 tokens per query. The FP32RAG baseline delivers the highest BLEU and F1 but incurs the heaviest memory and GPU load and erratic CPU use. In Q-RAG, replacing the LLM with its 4-bit equivalent cuts GPU and memory usage by similar margins while holding answer quality within 3% of baseline. Adding parallel hybrid retrieval, in QP-RAG, retains response quality similar to the quantization-only approach showing that up to 56% reduction in estimated GPU power index can be achieved with negligible quality degradation –highlighting its viability for sustainable AI deployments. Thus, under the tested configuration, 4-bit RAG offers the strongest observed quality-to-efficiency trade-off, while the inclusion of parallel computing may be ideal for high-throughput or edge scenarios; and the full precision when any quality loss is unacceptable.

Keywords: generative AI; large language models; small language models; sustainable computing

1. Introduction

Retrieval-Augmented Generation (RAG) is an emerging technology that aims to mitigate issues inherent in LLMs by integrating external, up-to-date information that isn't normally possible with conventional models. This technology increases the ability of the LLM to produce more factual responses with updated information [1]. However, using RAG in different applications poses some sustainability issues; specifically, in hardware utilization where there must be minimal penalty on the quality of the generated response. While existing research has individually advanced the fields of hybrid retrieval [2], model quantization [3], and multi-domain adaptability [4], there is a clear gap in integrating these aspects into a cohesive, hardware-optimized RAG system.

In this paper, we implement an advanced RAG as a baseline (FP32RAG), along with two variants: one that integrates a 4-bit quantized LLM (Q-RAG), and another that includes quantization with a Parallelized Hybrid Retrieval Algorithm (QP-RAG). The implemented hardware-optimized RAG is evaluated to determine the hardware optimization and assess the relevance and accuracy of the generated answers using metrics grouped into hardware efficiency and quality of response; aiming to determine the trade-off between optimization and quality of responses. The main contribution of this study is an empirical benchmark of different RAG variants, specifically, full-precision, 4-bit quantized, and parallelized, under common multi-domain QA and hardware-efficiency evaluation.

 doi:10.20944/preprints202606.0596.v1

The remaining sections of this paper are as follows: Section II. Related Work dives into existing literature and gaps this study addresses; Section III. Methodology; Section IV. Results & Discussion; Section V. Concludes with recommendations.

2. Related Work

There have been many advances in the field of NLP-driven AI, such as RAG, specifically in hybrid retrieval, model quantization, and domain-specific applications. However, there has not been much work that aims to integrate these many emerging innovations for a hardware-optimized multi-domain implementation. HybGRAG [5] stands out as it addresses hybrid question-answering by integrating textual and relational data through a retriever bank and a critique module. It emphasizes adaptability and interpretability in handling semi-structured knowledge bases. While effective in combining data types, HybGRAG does not explore hardware optimization techniques like quantization, nor does it focus on multi-domain scalability. Meanwhile, DO-RAG [6] enhances domain-specific QA by integrating knowledge graphs with semantic vector retrieval, employing an agentic chain-of-thought architecture for improved reasoning. The framework is tailored for specific domains but does not address hardware efficiency or the challenges of multi-domain question answering.

Regarding multi-domain QA, diverse benchmarks [4] explore tuning strategies to improve out-of-domain generalization in RAG systems. Although it tackles multi-domain challenges, it does not incorporate hardware optimization strategies like quantization or parallel retrieval mechanisms. Similarly, studies [7] propose using 4-bit quantization for embedding vectors in RAG systems to reduce memory requirements and accelerate search processes but still fail to address the multi-domain aspect. In contrast, Qrazor [3] introduces a 4-bit quantization scheme for LLMs, achieving near full-precision accuracy without additional fine-tuning, thus enhancing deployment on resource-constrained hardware.

There exist benchmarks [4] that investigate the use of tuning strategies to improve how RAG systems respond to queries out of their usual domain. While the aforementioned study explores different challenges in multi-domain QA, there have been no integration of strategies in hardware optimization, like quantization or parallelization. On the other hand, some studies [7] have addressed this need by proposing the use of quantization techniques in RAG systems with the goal of reducing the utilization of memory and also accelerating the search process. While this does tackle issues on hardware efficiency, it lacks focus on its use in multi-domain QA. The last notable study in this regard proposed the use of Qrazor [3] that utilizes 4-bit quantization for LLMs. The bright side about Qrazor's approach is that even without fine-tuning, it was able to achieve almost full-precision accuracy – this suggested the ideal use of the technique for scenarios where hardware usage is heavily constrained. However, even if this technique used hardware-efficiency centric methods like quantization, there was no optimization in terms of retrieval nor in the aspect of multi-domain QA.

3. Methodology

In this study, the general methodology (as depicted in Figure 1) begins by the selection of publicly available, open-source, and anonymized datasets; this ensures that any privacy and ethical issues are avoided. This section discusses the dataset and sampling techniques, hybrid retrieval, RAG variant pipelines, and the methods for evaluation.

 doi:10.20944/preprints202606.0596.v1

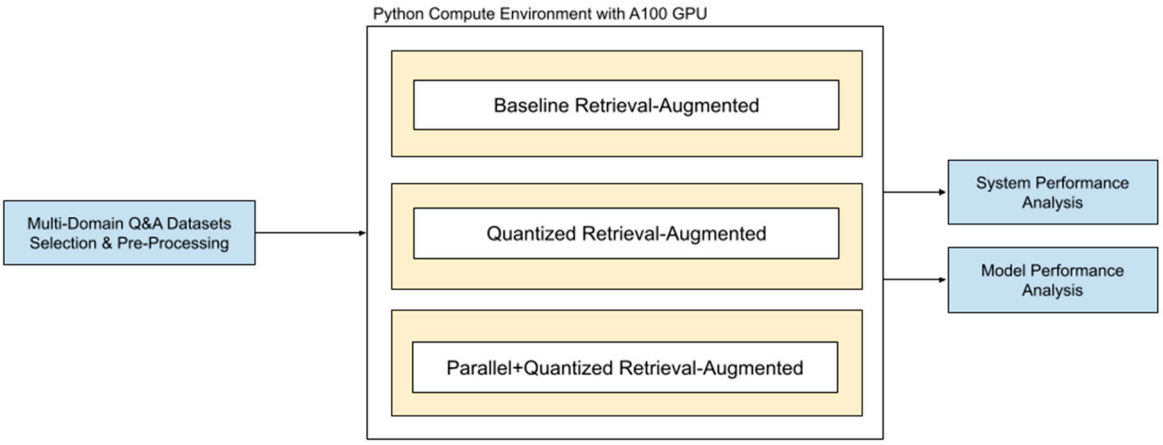

Figure 1. The General Methodology for the Study.

3.1. Development Environment

All related processes from development, testing, to evaluation were performed on individual compute instances equipped with an Nvidia A100 GPU with 40GB RAM. This development environment ran using Python 3.x, with compatible libraries including PyTorch, Transformers, BitsAndBytes, FAISS CPU, CUDA Toolkit, psutil, and GPUtil.

3.2. Dataset & Sampling

Three datasets containing Question-Answer pairs from different domains were utilized. Specifically, (1) FinQA 10-K dataset [8] which tackles real-world financial documents, aiming to measure the RAG's ability to demonstrate complex numerical reasoning; (2) PrivacyQA [9] which includes legal and regulatory jargon, requiring the RAG to perform comprehension of related policies; and (3) LogicInference_QA [10] that includes tasks aimed at the logical reasoning.

All three QA datasets were randomly sampled for 100 QA pairs each. As seen in Figure 2, the query lengths of the sampled dataset, measured in number of tokens, show that diverse query lengths are represented. This aims to simulate how the RAG would have to encounter different context windows as per the various sizes of user's queries.

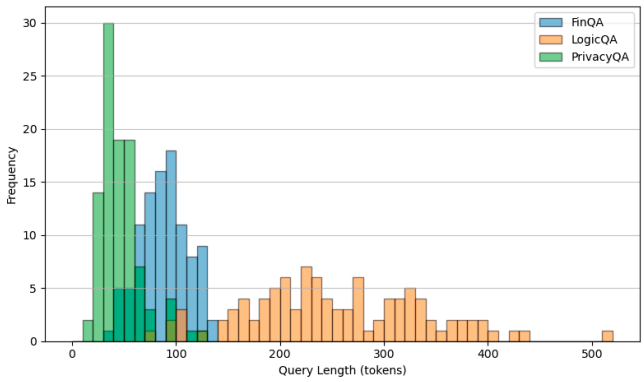

Figure 2 Query Length Distribution by Category.

A secondary dataset was utilized in this study for the hybrid retrieval – the Open Super-large Crawled Aggregated coRpus (OSCAR) dataset. It is a massive multilingual corpus derived from the Common Crawl web archive through automated language classification and filtering pipelines (Brack et al., 2024). Due to its scale, it is used to simulate conditions that resemble realistic uses of RAG.

 doi:10.20944/preprints202606.0596.v1

Once the datasets have been chosen, the study continues with the development of different RAG implementations: a full-precision RAG (FP32RAG), a RAG with 4-bit Quantized LLM (Q-RAG), and a RAG with 4-bit Quantized LLM and Parallelized Hybrid Retrieval (QP-RAG).

3.3. Hybrid Retrieval

The RAG pipelines utilize a common hybrid retrieval technique to obtain relevant data chunks from an external document corpus – this is where the answers get the additional context. In this retrieval step, a Small Language Model (SLM)-based method is implemented to judge the relevance of the retrieved document to the query, the top-K documents are then included as context for the final prompt provided to the LLM for response generation.

The documents sampled from the external dataset, are divided into N-sized chunks (N=1000), with a chunk overlap size of 100, and then passed to a sentence transformer (all-MiniLM-L6-v2) to generate sentence embeddings that are then indexed; with 384-dimensional embeddings. The indexing techniques utilized for the vector database is a vector store with Facebook AI Similarity Search (FAISS) with the Hierarchical Navigable Small World (HNSW) algorithm for approximating nearest neighbors with the following parameters: M = 64, this defines the maximum number of graph edges per vector; efConstruction=200, the amount of candidate queues per search used during the index construction; efSearch = 128, the size of candidate queues per search. The SLM-based hybrid retriever takes the user's query and the number of top documents (Top-K) and the number of candidate documents for re-ranking, these were set as Top-K = 5, and candidates = 100. The FAISS HNSW index is then queried to retrieve the set of candidate document IDs based on the nearest neighbour and returns the related document chunks which are then re-ranked. The document re-ranking used a cross-encoder (cross-encoder/ms-marco-MiniLM-L-6-v2) that assigns a precise relevance score to every document and query combination; the top-5 highest-scoring documents are then selected.

3.4. Retrieval-Augmented Generation (RAG) Variants

As seen in Figure 3, the FP32RAG variant integrates hybrid retrieval that combine dense vectors with SLM-based retrieval to capture broad and precise information during retrieval better. As a result, the context to a prompt now includes the initial query and the top-K related documents; this final prompt is then sent to the full-precision (FP32) LLM, a Mistral-7B-Instruct v0.3 model (mistralai/Mistral-7B-Instruct-v0.3), to generate a response. This variant is dedicated as the baseline for comparison of the other variants and is based on research by Y. Gao et al. (2023) [12].

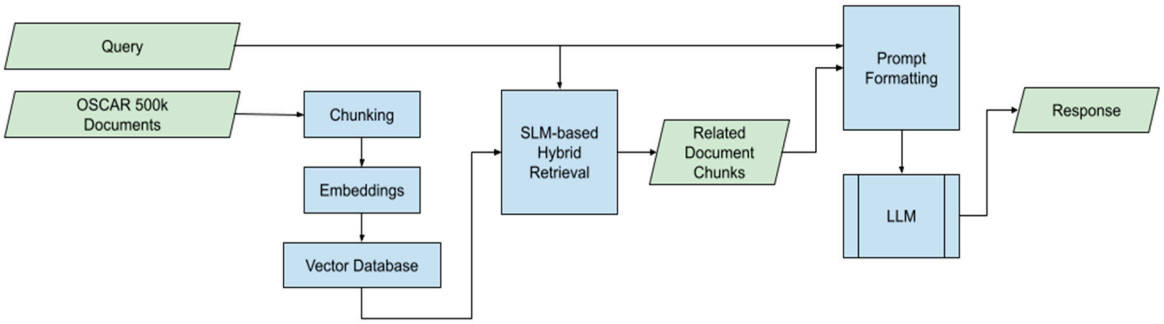

Figure 3. Advanced RAG Pipeline based on [12] for FP32RAG.

Quantization is a model optimization technique that reduces the weights and/or activations of a model, in this scenario, it is applied to the LLM. Typically, we compress 32-bit floating point (FP32) models into 8-bit or 4-bit integers [13], which studies have shown to help reduce memory footprint, accelerate inference, and lower hardware requirements. Figure 4 shows a pre-trained Mistral LLM

 doi:10.20944/preprints202606.0596.v1

with 7-Billion Parameters (unsloth/mistral-7b-instruct-v0.3-bnb-4bit), quantized using the bnb (BitsAndBytes) 4-bit format, is utilized in the architecture. All processes remain the same, but with a 4-bit quantized LLM.

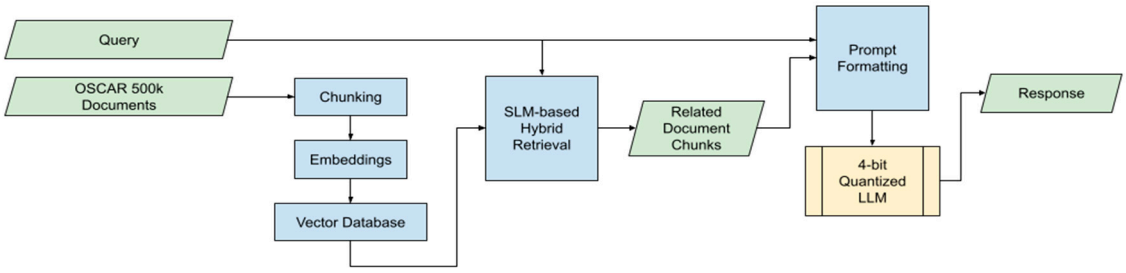

Figure 4. Advanced RAG Pipeline based on [12] for Q-RAG.

Figure 5 shows that QP-RAG uses the advanced RAG architecture with a 4-bit quantized LLM and includes parallelization applied to the cross-encoder re-ranking of the SLM-based hybrid retrieval method. Specifically, the scoring workload is distributed across 4 threads and chunked using a ceiling-division workload split, then parallel scoring is performed using the cross-encoder re-ranker model (cross-encoder/ms-marco-MiniLM-L-6-v2). Once all candidate scores are collected, the top-5 most relevant documents are selected. The final prompt includes the top-5 retrieved documents as context to answer the user query, and then the 4-bit quantized LLM (unsloth/mistral-7b-instruct-v0.3-bnb-4bit) generated the response.

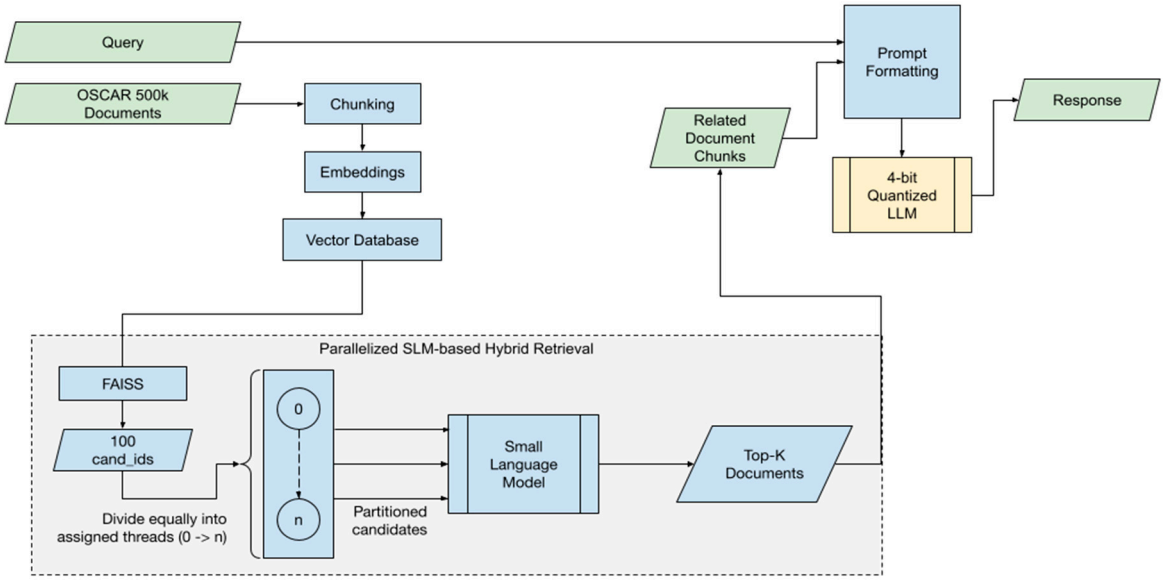

Figure 5. Advanced RAG Pipeline based on [12] for QP-RAG.

3.5. Evaluation

In the evaluation, we compare the performance of the FP32RAG, treated as baseline, to the other RAG variants on 300 queries of varying query length in tokens. We sample the instantaneous CPU, memory, and GPU utilization using psutil and GPUtil tracing tools in Python before and after each inference – psutil is used to monitor CPU and RAM usage, whereas GPUtil retrieves information about GPU usage.

$$U_{ave} = \frac{u_b + u_a}{2} \quad (1)$$

 doi:10.20944/preprints202606.0596.v1

We express this as u_b and u_a for the utilization before and after each inference. Therefore, Equation 1 shows the utilization of the metrics per query. Table 1 summarizes the evaluation metrics for hardware efficiency and response quality dimensions of the three RAG variants (FP32RAG, Q-RAG, and QP-RAG). Particularly, the estimated GPU Power Index is derived from GPU utilization and computed using the GPU's Thermal Design Power (TDP). The Nvidia A100 GPUs used have an estimated TDP of 250W. A linear utilization-based model to compute for the estimated GPU Power Index (P_{est}) for each query as seen in Equation 2.

Table 1. Summary of Evaluation Metrics.

Type	Metric
Hardware Efficiency	CPU Utilization
	Memory Utilization
	GPU Utilization
	Estimated GPU Power Index
Response Quality	Token-F1
	BLEU
	ROUGE-1
	Precision
	Recall
	F1-Score
	Bert Exact

$$P_{est(q)} = P_{idle} + (P_{full} - P_{idle}) \times \left(\frac{GPU\ Util(q)}{100}\right) \quad (2)$$

Equation 2 assumes the linear scaling of estimated GPU Power Index with utilization, such that when the GPU is idle, it consumes P_{idle} (0.10 TDP); at full load, it consumes P_{full} (0.75 TDP). It uses approximate empirical measurements, and the estimated GPU Power Index is linear for values in between. However, the approximation is limited to a single server instance on the computational environment with GPU utilization as sole predictor. Other scalability predictors are not considered in the linear relationship due to the GPU-centric nature of the RAG variants; and does not consider the effect of other workload beyond the NLP-driven application. Since each run query utilizes the GPU, sustainability is measured here by observing whether there are significant differences in the baseline compared to the other variants; this is computed statistically in terms of Mean Power, Median, Standard Deviation, Min-Max, and difference in estimated GPU Power Index for all 300 queries.

After running the 300 queries and collecting the results, each RAG variant's response to the queries is paired with the original intended result to compute the response quality. Figure 6 shows the query, ground truth answer, and RAG response. For each comparison of the ground truth answer to the per RAG response, we compute the following: BLEU with smoothing (to avoid zero scores on short answers) and a ROUGE scorer for ROUGE-1 and ROUGE-L. We compare the RAG-generated response to the Ground Truth Answer and compute five quality signals: (1) a simple set-based Token-level F1 that captures overlapping words regardless of order, (2) BLEU for n-gram precision with smoothing, and (3-4) ROUGE-1 and ROUGE-L F-measures for unigram overlap and longest common subsequence similarity. Additionally, we compute the BERTScore (precision, recall, F1) in a single batched call across all ground truth-RAG response pairs to capture semantic similarity that tolerates paraphrasing. It is important to note that in this analysis, the length of the generated answers was not directly isolated for the comparison of performance metrics.

 doi:10.20944/preprints202606.0596.v1

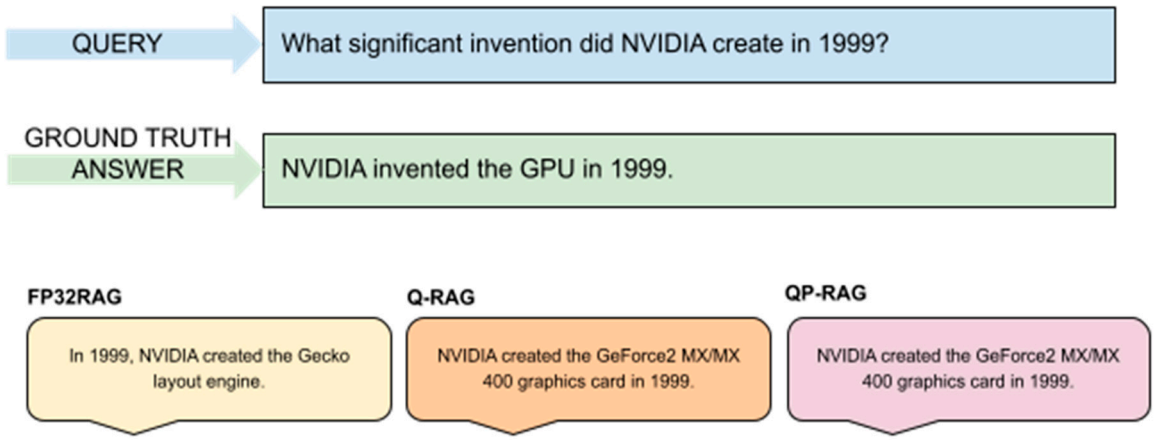

Figure 6. Sample Query, Expected Response, and Response per RAG Variant.

4. Results and Discussion

CPU, memory, and GPU utilization are metrics used to measure hardware efficiency per RAG. Based on expected trends, the quantized LLM should have generally lower CPU utilization than the baseline RAG performance. However, introducing the thread-level parallelism should show increased hardware utilization for all 300 data points with varying query (token) lengths. As seen in Figure 7, the performance of the RAG system utilizing a quantized model generally sits above the baseline for small-to-mid-length queries, showing the likelihood that the CPU is being driven harder with the quantized model. Beyond ~200 tokens, the Q-RAG curve occasionally dips slightly below baseline, reflecting the additional per-token unpacking or dequantization overhead. The parallel hybrid RAG configuration performs almost consistently the highest of all three curves across the full token range, with typical utilization in the 11-12% band and peaks up to ~15.3% (around 380 tokens). The smoother, elevated profile suggests that implementing the multi-threaded parallelism effectively saturates compute units, ensuring more uniform CPU usage even as query length grows. Regarding memory savings performed against the baseline values, Figure 8 shows that the Q-RAG sits above zero across the entire token range, implying that it consistently reduces memory footprint relative to the full-precision baseline. Its typical savings are in the 0.8% to 1.3% range for around 50- to 300-token queries, peaking at around 1.4% near the 240-token bin.

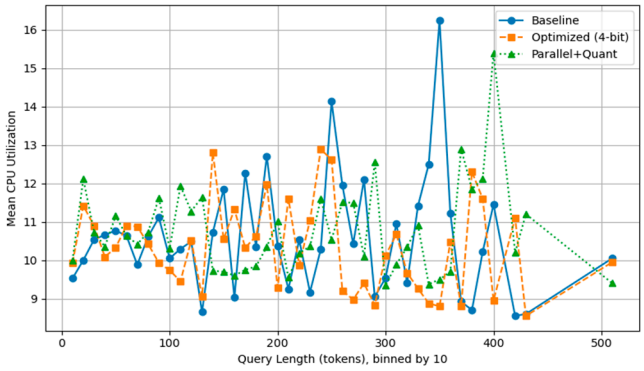

Figure 7. Overall CPU Utilization vs. Query Length.

 doi:10.20944/preprints202606.0596.v1

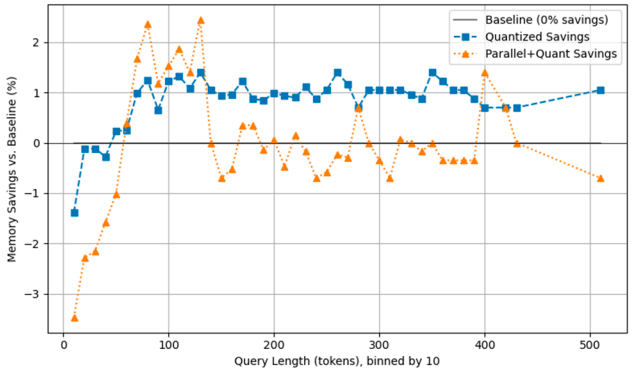

Figure 8. Overall Memory Savings vs. Query Length.

Additionally, for very short queries with fewer than 40 tokens, there is a negative dip likely due to the quantization-dequantization overhead. When implemented with thread-level parallelism, it starts with significant savings at the smallest bins, reflecting the efficiency of packing more work per thread. As query length grows, the parallel overhead of data replication erodes those early wins: by the 50 to 70 token range, the savings increase to 2.3%, then hover in the +1% to -1% range for most mid-range lengths. An analysis of the performance shown in Figure 8 suggests that parallelism incurs some penalty in the case of memory overhead as some of the savings show to dip regardless of the query length.

In Figure 9, the FP32RAG's performance in terms of GPU Utilization over all varying query sizes averages to around 55 to 58%. Meanwhile, Q-RAG and QP-RAG perform similarly with significantly lower utilization at 14 to 17%. Since FP32RAG uses the full model size, and Q-RAG and QP-RAG use a 4-bit quantized LLM, it can be inferred that quantization is responsible for the reduction in compute intensity. It is also worth noting that despite the similarity in GPU utilization, QP-RAG's inclusion of parallelized hybrid retrieval slightly increases the utilization to 15 to 18%. GPU Utilization is utilized to compute for the estimated GPU power index based on previously presented Equation 2. Table 2 shows the estimated GPU Power Index reported as the results of the linear utilization-based model; these do not reflect direct measurement of power consumption but are only values derived from the sampled GPU utilization from the Nvidia A100 TDP hardware specifications and the identified idle and full-load values. Due to the quantization present in both Q-RAG and QP-RAG, the estimated mean power has been reduced to 56%; from 115.31W mean power of FP32RAG to Q-RAG and QP-RAG's 50.21W and 50.54W, respectively.

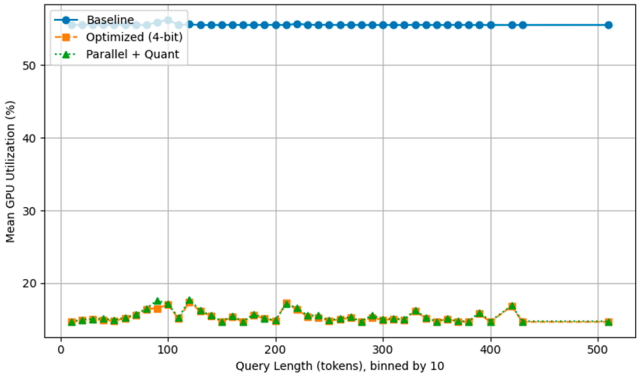

Figure 9. Overall GPU Utilization vs. Query Length.

 doi:10.20944/preprints202606.0596.v1

Table 2. Estimated GPU Power Index on Nvidia A100 TDP.

Metric	FP32RAG	Q-RAG	QP-RAG
Mean Power [W]	115.31	50.21	50.54
Median [W]	115.18	49.04	49.23
Std [W]	1.30	3.29	3.81
Min-Max [W]	115.18–131.76	48.85-82.50	48.98-82.37
Δ vs FP32RAG	-	56.45%	56.17%

Furthermore, reviewing the standard deviation value implies that varying query lengths also experience similar reduction in overall estimated GPU power index. All three variants perform below the A100's full 250W TDP; even FP32RAG's mean of 115W shows ~46% of the 250W TDP. Meanwhile, the other variants only utilize around 50W for most queries, suggesting more sustainable use of computing resources through changes in its architecture. While hardware efficiency metrics provide a glimpse at the performance of the RAG in its utilization of compute resources, it is unable to fully capture the nuance of RAG's capability. For this reason, it becomes necessary for us to also look at the response quality metrics.

The response quality metrics seen in Table 3 aim to further demonstrate whether the hardware efficiency achieved does not substantially compromise the quality of responses generated by the RAG. The low performance on all response quality metrics is inherent due to the external large corpus, OSCAR, containing web-scraped and potentially non-relevant documents to the different domains in the QA data. Due to the non-uniform difficulty in the QA domains, the quality metrics show lower performance in the LogicQA and PrivacyQA queries. While the FinancialQA (FinQA) can be answered with the external corpus; LogicQA and PrivacyQA also require specialized documents to answer. Specifically, LogicQA would have to be answered with logic and inference-focused data, while PrivacyQA would require highly situational information for any RAG variant to generate contextually relevant answers. While we observe the low performance of the RAG variants in terms of the different quality response metrics; we emphasize on the relative preservation of quality in the different variants as opposed to novel QA performance compared to the baseline variant – FP32RAG. In the comparative analysis of the variants' performance, we see that the different RAG variants for three different datasets yield identical profiles; while there are minimal losses in performance, none of them suggest a complete performance degradation due to the optimization performed.

Table 3. Performance of RAG Configurations on Multi-Domain Dataset Queries.

Metrics	FinQA Queries			LogicQA Queries			PrivacyQA Queries		
	FP32	Q	QP	FP32	Q	QP	FP32	Q	QP
Token F1	0.28	0.27	0.26	0.10	0.11	0.12	0.07	0.08	0.08
BLEU	0.13	0.13	0.13	0.05	0.05	0.05	0.03	0.03	0.04
ROUGE-1	0.33	0.32	0.32	0.32	0.31	0.30	0.10	0.09	0.09
Precision	0.17	0.16	0.16	0.04	0.03	0.03	0.02	0.01	0.01
Recall	0.33	0.32	0.33	0.27	0.26	0.25	0.01	0.01	0.01
F1 Score	0.26	0.25	0.25	0.15	0.14	0.13	0.02	0.01	0.01
bert_exact	0.17	0.16	0.16	0.02	0.01	0.01	0.02	0.02	0.02

A look at the metrics precision & bert_exact shows that the most significant relative drops are due to quantization slightly altering token embeddings on which downstream classifiers depend. Recall holds steady under QP-RAG, hinting that parallel retrieval pipelines may recover some of the recall lost to quantization. Performance in the logicQA queries shows a slight pull-back in ROUGE-1 (≈2 to 3% lower) and F1 Score (~1 to 2% drop) for the optimized runs versus Baseline. However, performance in the metrics Precision, Recall, and Token F1 remains within ~1% of Baseline. QP-RAG slightly recovers some of the decrease in Token F1 and Recall, but the overall performance is similar to Q-RAG. Additionally, we had determined that FinQA had a minimum, maximum, and average

 doi:10.20944/preprints202606.0596.v1

length of 36, 139, and 87.3, respectively. In comparison, LogicQA queries had a minimum, maximum, and average length of 75, 516, and 253.55, respectively; and PrivacyQA with 17, 120, and 45.67, respectively. When considering the query lengths and their effects on the metrics, LogicQA had the longest queries and also lower quality response scores, while FinQA with its mid-length queries had the best quality response scores.

In contrast, PrivacyQA had the shortest queries but still the lowest response scores. The evidence in these results suggests that query length could be indicative of task complexity, but it does not solely define the quality of generated responses. Furthermore, other factors, including task difficulty, domain-specific documents, and retrieval relevance, may be drivers for the RAG variant's response quality. In regard to the performance on the PrivacyQA queries, there is a noticeable uptick for QP-RAG in the BLEU metric, which suggests that the documents from the external corpus may be retrieved more appropriately leading to better context in the generation of responses. However, both Q-RAG and QP-RAG suffer from minimal loss which does not exceed 1% in precision under the PrivacyQA. By taking a broader look at all the different metrics across all configurations, every metric stays within a ±2 window of the Baseline for Q-RAG and QP-RAG.

Summarized results from evaluating different RAG variant's performance imply that Q-RAG can have efficient hardware use with little quality penalty; Q-RAG and QP-RAG provide a promising solution. Efficient hardware use is possible with a 1-2% loss in these response quality metrics through the Parallel with 4-bit Quantization configuration, as observed in QP-RAG; it often matches or even exceeds Baseline on metrics like BLEU or bert_exact while delivering the most significant efficiency gains.

Table 4. Summary of Average Values for Queries per RAG.

Average Metrics	FP32RAG	Q-RAG	QP-RAG
Token F1	0.16	0.15	0.15
BLEU	0.06	0.06	0.06
ROUGE-1	0.21	0.20	0.20
Precision	0.06	0.04	0.04
Recall	0.19	0.19	0.19
F1 Score	0.13	0.11	0.11
bert_exact	0.06	0.05	0.05

5. Conclusions

In this paper, multiple advanced RAGs have been implemented and compared based on multi-domain QA datasets to optimize hardware use through the implementation and benchmark analysis of Q-RAG and QP-RAG. The baseline model's performance suggests it remains the best choice for quality-focused implementations due to the lack of quantization artifacts, leveraging full-precision accuracy but at much greater hardware cost. Both implementations of the modified RAG utilize a 4-bit quantized LLM to generate responses; in this regard, comparing quantization alone to the baseline shows near-baseline response quality, where the performance against most metrics stays within ±3%, implying that it is an ideal option for retained performance in terms of response quality. However, it is noted that the tolerance of ±3% only serves as a comparison threshold for the study's present experimental setting and should be further validated in other application-specific deployments.

Beyond preserving response quality, utilizing a quantized model minimizes the use of the GPU and the RAG's memory footprint. The impact of this implementation is noticeable, as it provides the most considerable GPU and memory reductions with minimal quality degradation. QP-RAG proves its usefulness by responding with preserved quality, maximization of throughput, and the highest CPU utilization.

For most practical deployments, Q-RAG's configuration may offer the best balance by substantially reducing the GPU utilization, its estimated power index, and memory utilization while providing modest CPU usage reductions and minimal impact on most response quality metrics. A

 doi:10.20944/preprints202606.0596.v1

quality dip within positive/negative 3% is acceptable for 4-bit quantization for deployments similar to high-throughput or edge deployments where resource constraints dominate. The results suggest that QP-RAG's configuration should be used when the workload demands higher CPU utilization, due to having a negligible effect on hardware efficiency and response generation metrics. Adding CPU parallelism recovers ~0.7% of Token F1 and ~1% of BLEU. Utilizing this configuration allows the gain of more than 70% reductions in GPU load, 1% memory savings, and 5% to 20% CPU uplifts; metrics which could translate into meaningful cost, latency, or energy savings, where trading a few percent of quality may be a practical trade-off.

For production scenarios similar to the tested configurations and environment, especially where there is a need to reduce GPU and memory costs or boost serving throughput, losing up to 3% on the core response quality metrics is a reasonable price for the hardware efficiency unlocked with 4-bit quantization and parallelism. If any measurable drop is not tolerable, sticking with FP32 as used in the baseline RAG should be explored.

Future work should include repeated trials and paired statistical tests to further verify the observed differences and whether they remain significant across randomized variables, hardware available, sampling variations, and other domains.

Conflicts of Interest: The authors declare that there is no conflict of interest regarding the publication of this paper.

References

1. H. Soudani, E. Kanoulas, and F. Hasibi, "Fine Tuning vs. Retrieval Augmented Generation for Less Popular Knowledge," *Proceedings of the 2024 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, pp. 12–22, Dec. 2024, doi: 10.1145/3673791.3698415. [CrossRef] [Google Scholar] [Publisher Link]
2. K. Sawarkar, A. Mangal, and S. R. Solanki, *Blended RAG: Improving RAG (Retriever-Augmented Generation) Accuracy with Semantic Search and Hybrid Query-Based Retrievers*. 2024, pp. 155–161. doi: 10.1109/mipr62202.2024.00031. [CrossRef] [Google Scholar] [Publisher Link]
3. D. Lee, S. Choi, and I. J. Chang, "QRazor: Reliable and effortless 4-bit LLM quantization by significant data razoring," *ArXiv.org*, Jan. 2025, doi: 10.48550/arxiv.2501.13331. [CrossRef] [Google Scholar] [Publisher Link]
4. Misrahi, N. Chirkova, M. Louis, and V. Nikoulina, "Adapting large language models for Multi-Domain Retrieval-Augmented-Generation," *ArXiv.org*, Apr. 2025, doi: 10.48550/arxiv.2504.02411. [CrossRef] [Google Scholar] [Publisher Link]
5. M.-C. Lee et al., "HYBGRAG: Hybrid Retrieval-Augmented Generation on Textual and Relational Knowledge bases," *arXiv (Cornell University)*, Dec. 2024, doi: 10.48550/arxiv.2412.16311. [CrossRef] [Google Scholar] [Publisher Link]
6. D. O. Opoku, M. Sheng, and Y. Zhang, "DO-RAG: a Domain-Specific QA framework using knowledge Graph-Enhanced Retrieval-Augmented Generation," *arXiv (Cornell University)*, May 2025, doi: 10.48550/arxiv.2505.17058. [CrossRef] [Google Scholar] [Publisher Link]
7. T. Jeong, *4bit-Quantization in Vector-Embedding for RAG*. 2024, pp. 1037–1042. doi: 10.1109/icmla61862.2024.00156. [CrossRef] [Google Scholar] [Publisher Link]
8. virattt, "Financial Q&A - 10k." [Online]. Available: https://www.kaggle.com/datasets/yousefsaeedian/financial-q-and-a-10k
9. Ravichander, A. W. Black, S. Wilson, T. Norton, and N. Sadeh, *Question Answering for Privacy Policies: Combining Computational and Legal Perspectives*. 2019, pp. 4946–4957. doi: 10.18653/v1/d19-1500. [CrossRef] [Google Scholar] [Publisher Link]
10. S. Ontanon, J. Ainslie, V. Cvicek, and Z. Fisher, "LogicInference: A New Dataset for Teaching Logical Inference to seq2seq Models," *arXiv (Cornell University)*, Mar. 2022, doi: 10.48550/arxiv.2203.15099. [CrossRef] [Google Scholar] [Publisher Link]
11. M. Brack et al., *Community OSCAR: A Community Effort for Multilingual Web Data*. 2024, pp. 232–235. doi: 10.18653/v1/2024.mrl-1.19. [CrossRef] [Google Scholar] [Publisher Link]

 doi:10.20944/preprints202606.0596.v1

12. Y. Gao et al., "Retrieval-Augmented Generation for Large Language Models: A survey," *arXiv (Cornell University)*, Dec. 2023, doi: 10.48550/arxiv.2312.10997. [CrossRef] [Google Scholar] [Publisher Link]
13. Jacob et al., "Quantization and training of neural networks for efficient Integer-Arithmetic-Only inference," *arXiv.org*, Dec. 15, 2017. https://arxiv.org/abs/1712.05877. [CrossRef] [Google Scholar] [Publisher Link]