Workload Characterization of CPU-Based Medical RAG Systems on ARM Architecture

1. Problem and Motivation

Medical AI systems increasingly require on-premise deployment due to privacy regulations (e.g., HIPAA) and data sovereignty constraints. While GPU-based LLM inference has been extensively studied, clinical and edge environments often depend on CPU-only deployments on ARM architectures such as Apple Silicon.

Existing workload characterization studies primarily target x86 CPUs or GPU-accelerated systems, leaving ARM CPU-only inference uncharacterized. This gap limits optimization for OS-level resource allocation and scheduling.

This project conducts the first systematic workload characterization of CPU-only medical RAG on Apple M2 Pro. By profiling CPU utilization, memory footprint, and latency across pipeline stages, we establish a performance baseline that informs ARM-specific system design and deployment strategies.

2. Related Work

Recent work in RAG optimization has focused on GPU and x86 environments. Jiang et al. [1] proposed RAGO, a framework for profiling RAG workloads to enhance throughput and resource use, while Izacard et al. [2] introduced Atlas, demonstrating efficiency through retrieval integration. However, these systems presume GPU availability.

For LLM inference efficiency, Alizadeh et al. [3] (LLM in a Flash) and Xu et al. [4] (SpecEE) achieved performance gains through speculative or flash-based approaches, but both assume high-performance hardware. Na et al. [5] analyzed CPU-based inference, yet focused on x86 with matrix accelerators, not ARM.

In the medical domain, Li et al. [6] developed BiomedRAG for biomedical retrieval and summarization, highlighting domain-specific adaptation benefits. Yet, no study explores OS-level workload patterns on ARM-based CPU deployments, a critical gap given their growing presence in healthcare environments.

3. Methodology

Research Question:
What are the system-level performance characteristics of CPU-only medical RAG workloads on ARM architecture?

Experimental Setup:
A medical RAG pipeline is deployed on an Apple MacBook Pro M2 Pro (10-core CPU, 16GB

memory) using Ollama with Llama-3.2-3B (CPU-only mode). The pipeline includes: (1) input validation, (2) embedding generation (sentence-transformers), (3) vector retrieval (Annoy index), (4) result validation, and (5) LLM generation. We evaluate 100 medical queries (50 emergency, 50 treatment).

Measurement & Control:
Background processes (Spotlight, iCloud) are disabled. Execution is verified as CPU-only via Activity Monitor. Metrics collected:
- End-to-end latency and per-layer timing (time.perf_counter())
- CPU and memory usage (psutil)
- Query length and type

Analysis:
Latency distributions (p50, p95), CPU utilization patterns, and memory trends are compared between emergency and treatment queries. Statistical correlations reveal bottlenecks and inform deployment recommendations for ARM-based inference.

Data Presentation:
Results will be presented through: (1) time breakdown charts showing per-layer contribution, (2) latency distribution plots (p50/p95) for query types, (3) CPU/memory usage timelines, and (4) summary tables comparing emergency vs treatment workloads. Visualizations will be generated using matplotlib/seaborn.

4. Timeline and Deliverables

Timeline:
- Oct 14–27: Setup & instrumentation
- Oct 28–Nov 10: Primary data collection
- Nov 11–17: Validation & reruns
- Nov 20: CODE FREEZE
- Nov 21–Dec 9: Analysis, visualization, report

Deliverables:
1. Dataset with per-layer profiling metrics
2. Statistical analysis and charts
3. Deployment recommendations for ARM-based medical AI
4. Experimental scripts and documentation
5. Final presentation and report

Expected Contribution:
This work delivers the first OS-level characterization of CPU-only medical RAG workloads on ARM. Findings guide efficient resource scheduling and model deployment in privacy-sensitive environments.

IEEE Bibliography

1. [1] W. Jiang et al., “RAGO: Systematic performance optimization for retrieval-augmented generation serving,” in Proc. 52nd Annu. Int. Symp. Comput. Architecture (ISCA), Tokyo, Japan, 2025. [Online]. Available: https://dl.acm.org/doi/10.1145/3695053.3731093
2. [2] G. Izacard et al., “Atlas: Few-shot learning with retrieval-augmented language models,” J. Mach. Learn. Res., vol. 24, pp. 1–43, 2023. [Online]. Available: https://dl.acm.org/doi/10.5555/3648699.3648950
3. [3] K. Alizadeh et al., “LLM in a flash: Efficient large language model inference with limited memory,” in Proc. 62nd Annu. Meeting Assoc. Comput. Linguistics (ACL), Bangkok, Thailand, 2024, pp. 12562–12584. [Online]. Available: https://aclanthology.org/2024.acl-long.678.pdf
4. [4] J. Xu et al., “SpecEE: Accelerating large language model inference with speculative early exiting,” in Proc. 52nd Annu. Int. Symp. Comput. Architecture (ISCA), Tokyo, Japan, 2025. [Online]. Available: https://dl.acm.org/doi/10.1145/3695053.3730996
5. [5] S. Na et al., “Understanding performance implications of LLM inference on CPUs,” in Proc. IEEE Int. Symp. Workload Characterization (IISWC), Vancouver, BC, Canada, 2024, pp. 169–180. [Online]. Available: https://ieeexplore.ieee.org/document/10763564
6. [6] M. Li et al., “BiomedRAG: A retrieval-augmented large language model for biomedicine,” J. Biomed. Inform., vol. 162, Art. no. 104769, 2025. [Online]. Available: https://dl.acm.org/doi/abs/10.1016/j.jbi.2024.104769