International Journal of Artificial Intelligence, Data Science, and Machine Learning
Grace Horizon Publication | Volume 7, Issue 2, 1-3, 2026
ISSN: 3050-9262 | https://doi.org/10.63282/3050-9262.IJAIDSML-V7I2P101

Original Article

Hybrid Accelerator Selection for Generative AI Workloads: A Cost-Effective Approach Based on Model Type and Pipeline Stage

Rajalakshmi Srinivasaraghavan
Independent Researcher, USA.

Received On: 16/02/2026 ***Revised On: 13/03/2026*** ***Accepted On: 22/03/2026*** ***Published On: 02/04/2026***

Abstract *- With the increasing adoption of machine learning, AI, and large language models in production environments, the requirement for computational acceleration continues to intensify. However, defaulting to expensive off-chip accelerators, such as GPUs and TPUs for all workloads leads to unnecessary cost inefficiencies. This paper presents a framework for hybrid accelerator selection that considers model characteristics, pipeline stages, and workload requirements. We demonstrate that strategic use of on-chip acceleration (CPU-based SIMD) for smaller models and latency-insensitive workloads, combined with selective deployment of off-chip accelerators for compute-intensive tasks, can significantly reduce infrastructure costs while maintaining performance requirements.*

Keywords *- AI, Performance, CPU, Machine Learning, Deep Learning, LLM.*

1. Introduction

1.1. The Acceleration Dilemma

Modern machine learning deployments face a critical resource allocation challenge. While specialized accelerators like NVIDIA GPUs and Google TPUs chips offer substantial performance improvements for certain workloads, their cost and availability constraints make universal deployment impractical. With cloud instances ranging from $2-4 per hour, making them 10-20x more expensive than CPU-based instances.

1.2. The Case for Hybrid Acceleration

Not all ML models require the same computational resources. A Retrieval-Augmented Generation (RAG) pipeline, for example, consists of multiple stages with vastly different computational profiles: embedding generation, vector search, reranking, and text generation. Treating all stages uniformly leads to resource waste and unnecessary costs.

This paper argues for a model-aware, stage-aware hybrid acceleration strategy that matches computational resources to workload characteristics, considering:

- Model size and architecture
- Computational intensity
- Latency sensitivity
- Throughput requirements
- Cost constraints

2. Acceleration Technologies: A Comparative Analysis

2.1. On-Chip Acceleration

Modern CPUs incorporate specialized instruction sets for AI/ML workloads:

Intel Advanced Vector Extensions (AVX-512): Provides 512-bit SIMD operations, offering significant speedup for matrix operations compared to scalar execution. Particularly effective for small batch inference where GPU memory transfer overhead dominates computation time.

Intel Advanced Matrix Extensions (AMX): Introduced in Sapphire Rapids processors, AMX provides dedicated matrix multiplication units to accelerate low precision operations.

ARM Scalable Vector Extension (SVE): Offers flexible vector lengths and efficient ML operations on ARM-based servers.

IBM Power10 processors feature built-in Matrix-Multiply Assist (MMA) technology, accelerating AI inferencing and machine learning workloads.

2.2. Off-Chip Acceleration

GPUs: NVIDIA’s CUDA ecosystem remains the industry standard, with A100 and H100 devices offering higher TFLOPS of FP16 compute capacity. Such accelerators are indispensable for large-language-model workloads exceeding 7B parameters and for scenarios requiring high throughput.

The IBM Spyre Accelerator is a specialized, PCIe-attached AI accelerator designed for high-performance, low-

latency inferencing, particularly within IBM Z, LinuxONE, and Power systems.

AMD Instinct MI300X/MI325X/MI350X GPUs: These are specialized GPU accelerators designed for high-performance computing (HPC) and AI training, particularly focusing on accelerating single and double-precision workloads in data centers.

3. Pipeline-Based Acceleration Strategy

3.1. RAG Pipeline Analysis

3.1.1. A Typical RAG Pipeline Consists Of Four Stages [9]

- Query Embedding (e.g., sentence-transformers/all-MiniLM-L6-v2, 22M parameters)
- Vector Search (algorithmic, not ML)
- Reranking (e.g., cross-encoder/ms-marco-MiniLM, 33M parameters)
- Generation (e.g., Llama-2-7B, 7B parameters)

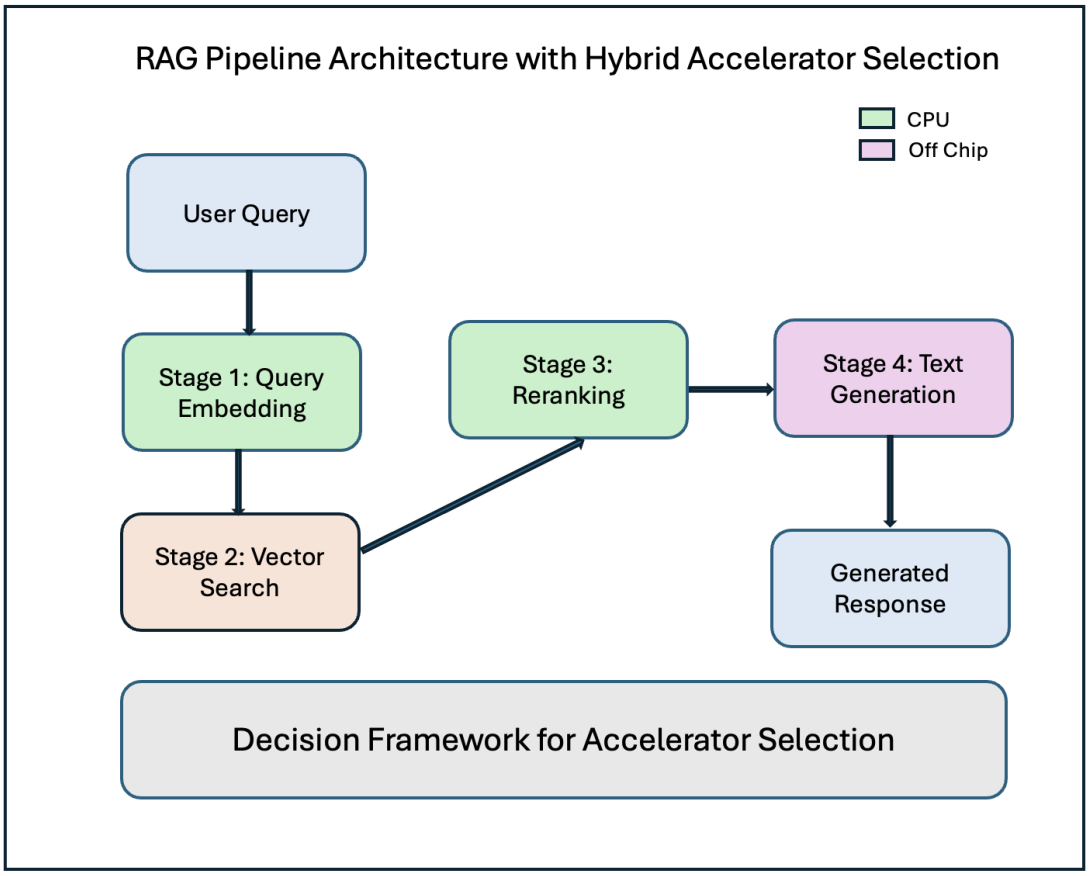

Fig 1: Hybrid-Accelerated Retrieval-Augmented Generation (RAG) Pipeline Architecture

3.2. Stage-Specific Acceleration Decisions

3.2.1. Embedding Models (22-110M parameters)

- Recommendation: On-chip acceleration (CPU)
- Rationale: Small model size means GPU memory transfer overhead exceeds computation time on CPU
- Performance: CPU achieves 50-200 embeddings/second, sufficient for most applications

3.2.2. Reranking Models (33-340M parameters)

- Recommendation: On-chip acceleration for latency-tolerant workloads; GPU for high-throughput
- Rationale: Cross-encoder reranking is compute-intensive but operates on small candidate sets (10-100 documents)

3.2.3. Generation Models (>7B parameters)

- Recommendation: Off-chip acceleration (GPU/TPU)
- Rationale: Autoregressive generation requires high memory bandwidth and compute; CPU performance degrades significantly
- Performance gap: Off chip accelerators achieves higher throughput for models >7B parameters.

3.3. Decision Framework

- IF model_size < 100M AND latency_requirement > 100ms: USE on-chip acceleration (CPU)
- ELIF model_size < 1B AND batch_size < 8: USE on-chip acceleration (CPU)
- ELIF model_size < 7B AND throughput_requirement < 10 req/sec: USE entry-level off-chip acceleration
- ELSE: USE high-end off-chip acceleration

4. Workload Characteristics and Acceleration Selection

4.1. Latency-Sensitive vs. Latency-Tolerant Workloads

4.1.1. Real-Time Inference (Latency < 100ms)

- User-facing applications require consistent low latency
- GPU acceleration justified even for smaller models when p99 latency matters
- Example: Chatbot responses, real-time recommendations

4.1.2. Batch Processing (latency tolerance > 1s)

- Offline workloads can leverage CPU parallelism effectively
- Cost optimization prioritized over latency

Examples: Document embedding for vector databases, bulk reranking, data preprocessing.

5. Implementation Considerations

5.1. Software Optimization

5.1.1. CPU Optimization

- Use optimized libraries: Intel oneDNN, OpenBLAS, ONNX Runtime
- Enable quantization: INT8 inference provides 2-4x speedup with minimal accuracy loss
- Batch processing: Accumulate requests to improve CPU utilization

5.1.2. GPU Optimization

- Use TensorRT, vLLM for optimized inference
- Implement dynamic batching to maximize GPU utilization
- Consider multi-instance GPU for workload isolation

5.2. Orchestration and Routing

5.2.1. Implement Intelligent Routing Based On

- Model size detection
- Current system load
- Latency requirements
- Cost budgets

Tools like Ray Serve, KServe, and Triton Inference Server support heterogeneous deployment.

6. Future Directions

6.1. Adaptive Selection

6.1.1. AI/Machine Learning-Based Meta-Models Could Predict Optimal Accelerator Selection Based On

- Historical performance data
- Real-time system metrics
- Cost constraints
- SLA requirements

7. Conclusion

Hybrid accelerator selection represents a pragmatic approach to ML infrastructure optimization. By matching computational resources to workload characteristics—considering model size, pipeline stage, and latency requirements—organizations can achieve significant cost savings (20-60%) without sacrificing performance.

The key insights are:

- Small models (<100M parameters) benefit from on-chip acceleration, especially for latency-tolerant workloads
- Pipeline stages have different computational profiles requiring different acceleration strategies
- Offline workloads should default to CPU-based acceleration
- Large models (>1B parameters) and high-throughput scenarios justify GPU investment

As ML deployment scales, intelligent resource allocation becomes not just an optimization but a necessity for sustainable operations.

References

[1] Lewis, P., et al. (2020). "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." Advances in Neural Information Processing Systems, 33, 9459-9474.

[2] Intel Corporation. (2023). "Intel Advanced Vector Extensions 512 (Intel AVX-512) Overview." Intel Developer Zone.

[3] Intel Corporation. (2022). "Intel Advanced Matrix Extensions (Intel AMX) Overview and Programming Guide." Intel Architecture Instruction Set Extensions Programming Reference.

[4] ARM Limited. (2023). "ARM Scalable Vector Extension: A Vector Length Agnostic Architecture." ARM Architecture Reference Manual Supplement.

[5] NVIDIA Corporation. (2023). "NVIDIA H100 Tensor Core GPU Architecture." NVIDIA Technical Whitepaper.

[6] Google Cloud. (2023). "Cloud TPU System Architecture." Google Cloud Documentation.

[7] Intel Corporation. (2023). "Intel oneAPI Deep Neural Network Library (oneDNN) Performance Benchmarks." Intel oneAPI Documentation.

[8] IBM Spyre accelerator. https://www.ibm.com/docs/en/ibm-spyre-for-power?topic=spyre-accelerator-power

[9] IBM Matrix Multiply Accelerator: https://www.ibm.com/support/pages/introduction-mma-matrix-math-accelerator-component-power10-systems