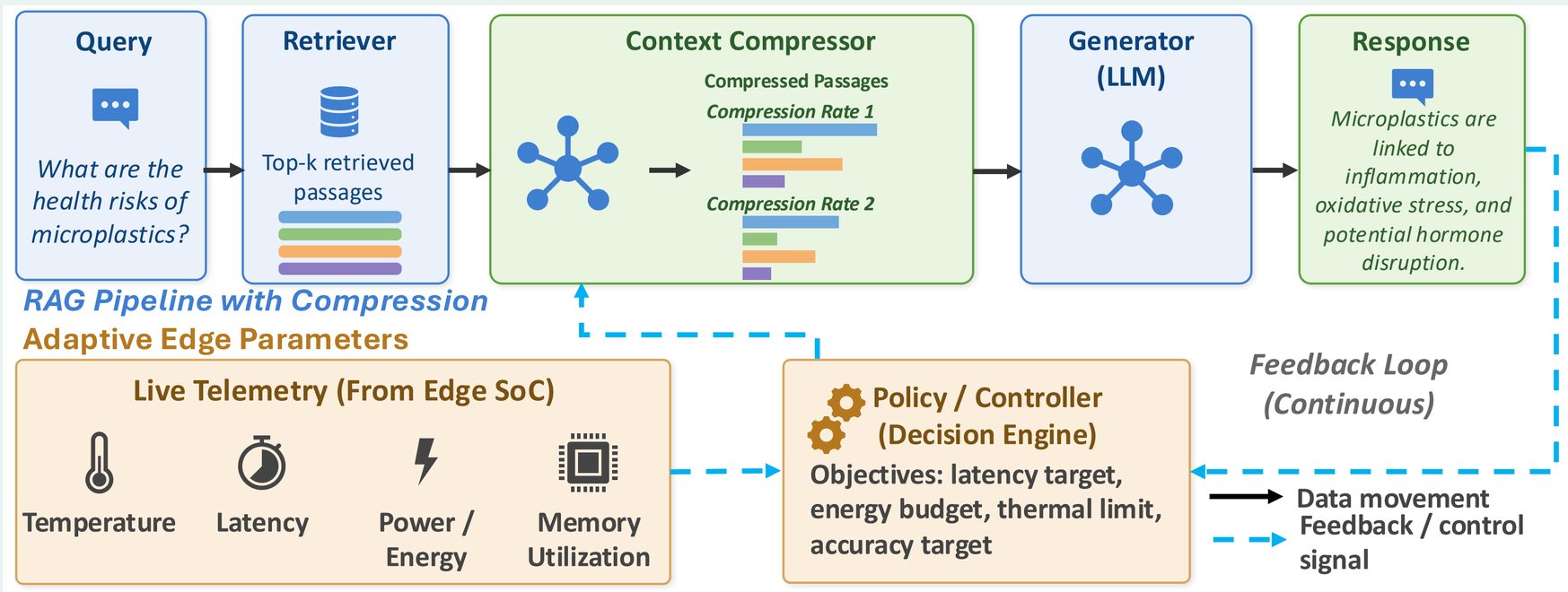
From Retrieved Context to Runtime Control: Adaptive Compression for Edge-based RAG

Zlatan Feric* Amir Taherin* Yanzhi Wang David Kaeli

Northeastern University, Boston, MA · ACM AI Leadership Summit 2026 · *Equal contribution

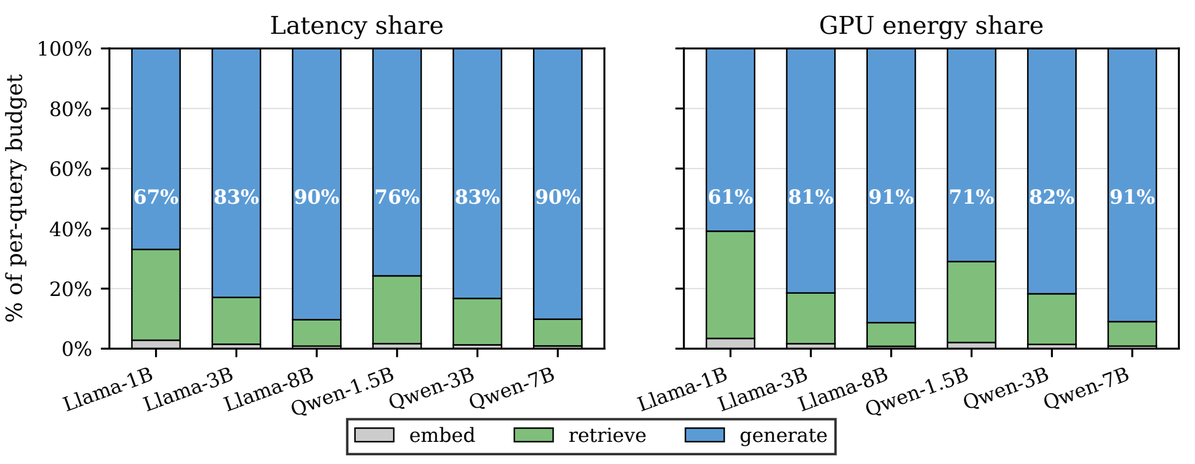
arXiv:2608.19535

Our Vision: Telemetry-Informed Adaptive Compression for Edge RAG



- Every retrieved token costs prefill time, KV-cache, memory traffic, and energy.
- The compressor runs **on the same SoC** — compression is not free.
- The right metric is **net benefit**: generation savings − compressor overhead.
- A fixed offline rate is blind to workload and device state.
- **Vision:** a controller watches workload features + live telemetry and picks the rate **per query**.

Observation 1: Generation Dominates the Budget

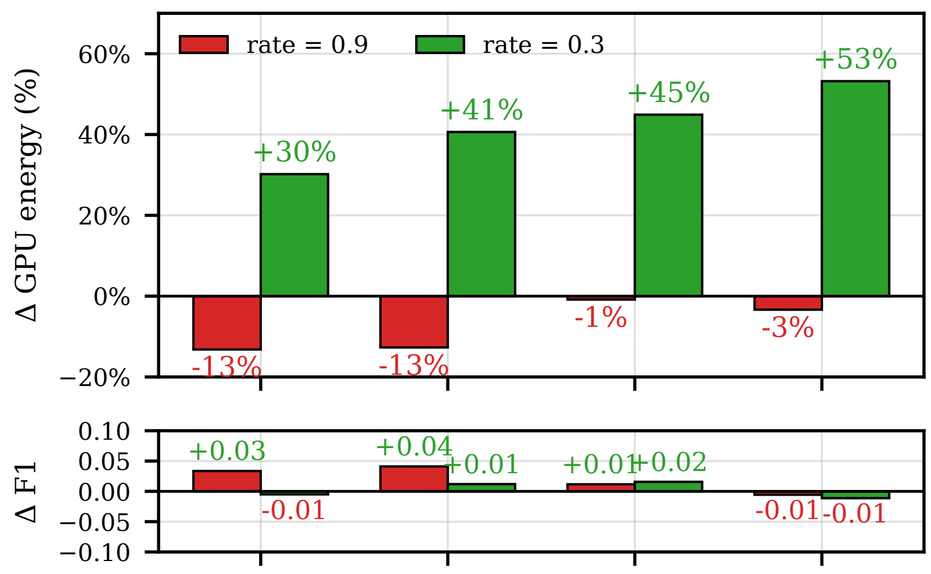


90 % / 91 %
of per-query latency / GPU energy is the generator alone at 7–8 B

33 % / 39 %
of time / energy goes to embed+retrieve at 1 B — little headroom for compression

Above 3 B: shortening the prompt is the highest-leverage knob.

Observation 2: The Value of Compressing Varies; the Safe Rate Does Not



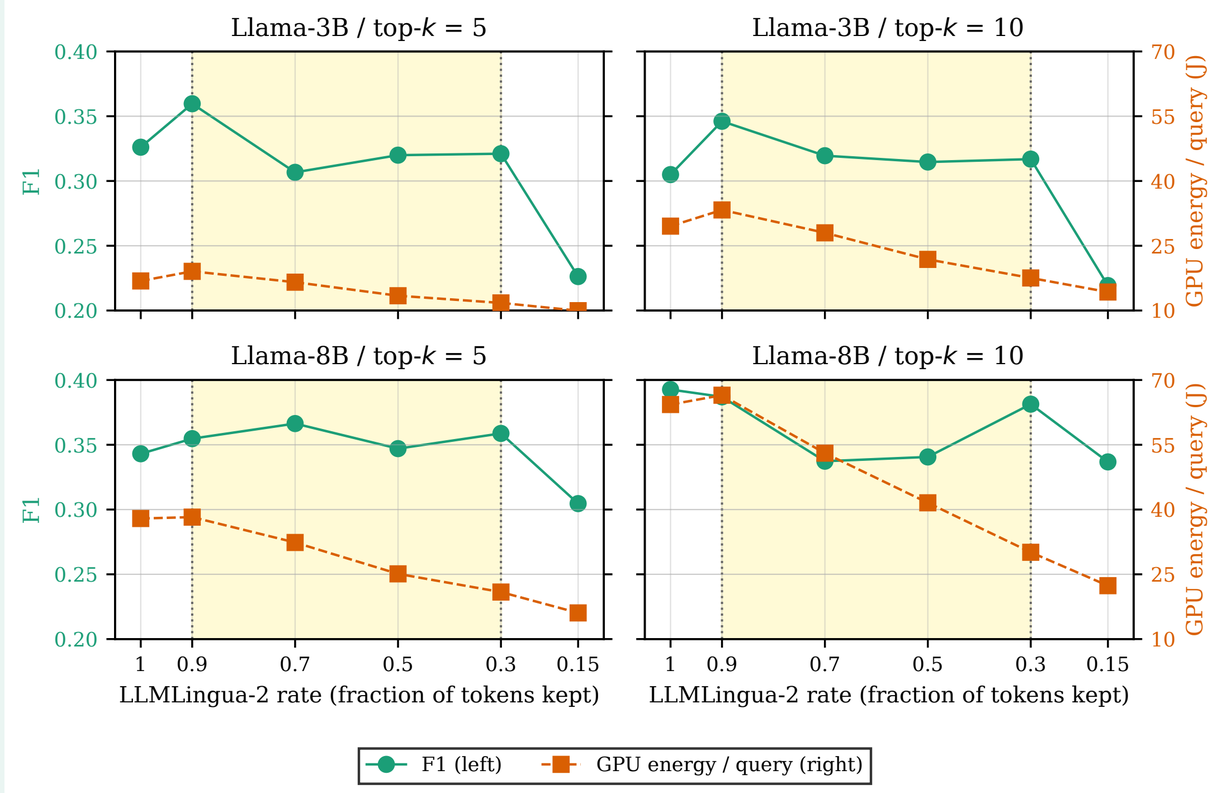
Net savings at rate 0.3 vs. uncompressed:

Model	k	Δlat.	ΔE_{GPU}	ΔF1
Llama-3B	5	+8.5%	+30.2%	−0.005
Llama-3B	10	+15.6%	+40.6%	+0.012
Llama-8B	5	+25.2%	+44.9%	+0.016
Llama-8B	**10**	**+32.6%**	**+53.2%**	−0.011

+53 % GPU
+48 % SoC energy saved per query at rate 0.3 (Llama-8B, k=10) — negligible quality loss

1.8×
spread in the win across workloads — the *payoff* is workload-dependent, the setpoint is stable

Observation 3: Two Knees Frame an Adaptive Operating Region



Answer F1 (left axis) and per-query GPU energy (right axis) vs. LLMLingua-2 rate on HotpotQA. Shaded band: adaptive operating region between the two knees.

- **Quality knee:** F1 flat from rate 1.0 down to 0.3; collapses at 0.15.
- **Energy knee:** rate 0.9 is a net *loss* — overhead not amortized.
- Between the knees: an ≈**0.6-wide operating region** for a controller.

Recommendation: Utilize Compression as a Runtime Knob

- **Controller:** workload features + live telemetry → per-query rate.
- **Cost prediction:** train cost models on this characterization data.
- **Cross-platform:** extend the analysis across Xavier, Orin, Thor.

Experimental Setup (AGX Thor)

Platform	AGX Thor: Blackwell GPU, 128 GB LPDDR5x, 273 GB/s
Corpus	Wikipedia 2018, ∼9.4 M passages (FlashRAG); e5-base-v2 + FAISS GPU index (∼28 GB)
Datasets	Natural Questions, HotpotQA; 100 seed-paired queries per configuration
Generators	Llama-3.2 1B/3B, Llama-3.1 8B; Qwen-2.5 1.5B/3B/7B (fp16)
Compressor	LLMLingua-2; rate ∈ {1.0, 0.9, 0.7, 0.5, 0.3, 0.15} (fraction of tokens kept)
Telemetry	GPU/SoC power, energy, memory, thermals via `tegrastats` (100 ms), per RAG stage

Built on **RAGMark** (our stage-level RAG benchmarking framework, IISWC'26) with telemetry collection from **Hydra** (our edge-LLM characterization framework, IISWC'26).
RAGMark: `github.com/zferic/RAGMark`
Hydra: `github.com/amirtaherin/hydra`, arXiv:2608.25053

Takeaway: the compression rate is a *runtime system knob* — adjust it per query, based on telemetry – let compression pay for its own overhead.

Paper: arXiv:2608.19535 · To appear in the Proceedings of the ACM AI Leadership Summit 2026 · Presented by Amir Taherin — amirtaherin.github.io