HOTCARBON '26

Evaluating LLM Knowledge Placement for Carbon-Efficiency

RAG · Prefix Caching · LoRA Fine-tuning
Carbon emission, Model accuracy, Response latency

Rankyung Hong
Abhishek Chandra

Univ. of Minnesota, Twin Cities

INTRODUCTION

What LLMs Are Used For

💬 Conversational AI

Chatbots, virtual assistants, and Q&A that hold natural, multi-turn dialogue.

💻 Code Generation

Writing, completing, and debugging code from natural-language prompts.

✍️ Content Creation

Drafting articles, emails, marketing copy, and summaries at scale.

📊 Data Analysis

Interpreting tables, extracting insights, and answering questions over data.

BACKGROUND

LLM Training vs Inference

Training

- Large-scale compute across many GPUs
- Optimizes (learns) the model weights
- One-time, upfront expensive process

Compute: massive | Timeframe: one-time | Goal: build the weights

Inference

- Uses fixed, already-trained weights
- Generates outputs token by token
- Continuous, real-time at every query

Compute: per-query | Timeframe: continuous | Goal: serve answers

Train once to set the weights — then pay compute again on every inference query.

BACKGROUND

Where Does Knowledge Live?

Parametric Knowledge

Baked into model weights

- Stored directly in the trained weights
- Always available — no lookup step
- Grows model size; costly to update
- Can become stale over time

Non-Parametric Knowledge

External storage via RAG

- Kept in an external document store
- Retrieved on demand per query
- Keeps the base model smaller
- Easy to refresh; adds retrieval cost

Retrieval-Augmented Generation (RAG): fetch relevant text from an external store at query time and add it to the prompt — instead of relying only on knowledge baked into the model's weights.

Tradeoff: instant availability vs. retrieval cost, model size, and freshness.

MOTIVATION

The Problem: Carbon vs. Quality vs. Cost

Carbon Footprint

- Energy scales with model size
- Bigger weights → more gCO_2/query
- Datacenter power dominates

Model Quality

- Static weights grow stale
- RAG injects fresh knowledge
- Scored by token-F1 accuracy

Latency & Cost

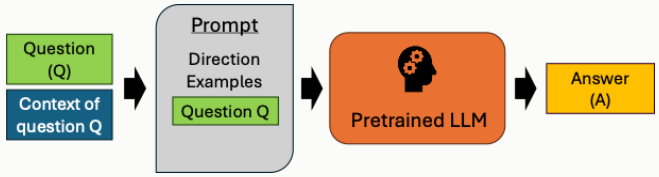
- Smaller models run faster
- Retrieval adds lookup overhead
- Measured in sec & $ per query

Which knowledge-placement method is most carbon-efficient — without losing quality or speed?

METHODS

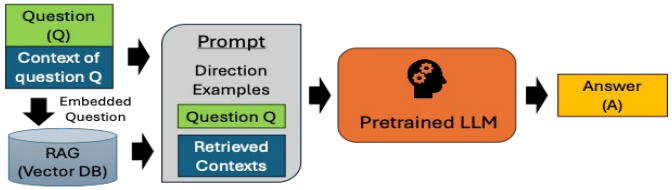
Three Ways to Place Knowledge

Base



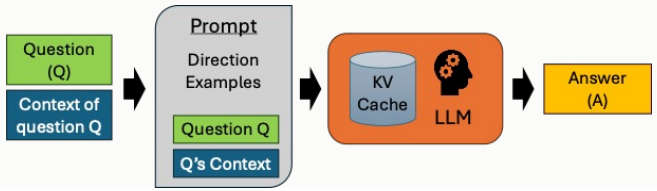
No augmentation, baseline

RAG



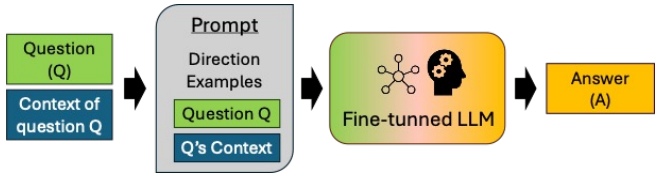
Retrieved context, ***transient***

Cache



Reused KV state, ***intermediate***

Fine-tune



Updated weights, ***persistent***

METHODS

Experimental Design

LLM Models

TinyLlama 1.1 B
Small model

Llama 3.1 8B
Large model

Datasets

SQuAD
short context

TriviaQA
long context

Hardware

Lambda Cloud
Single GPU - A100 40GB

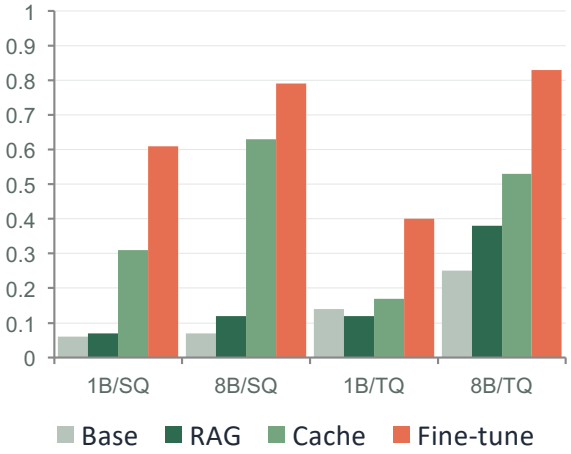
Carbon = Eq × CI × PUE + Tq × εGPU

operational: energy × grid CI × overhead + embodied: GPU-hours × amortized build cost

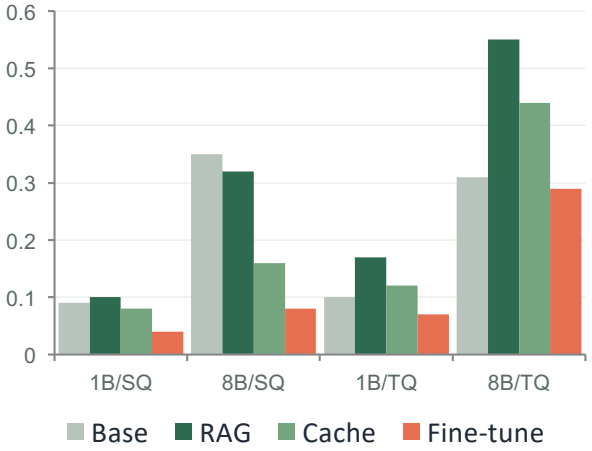
RESULTS

Which method wins?

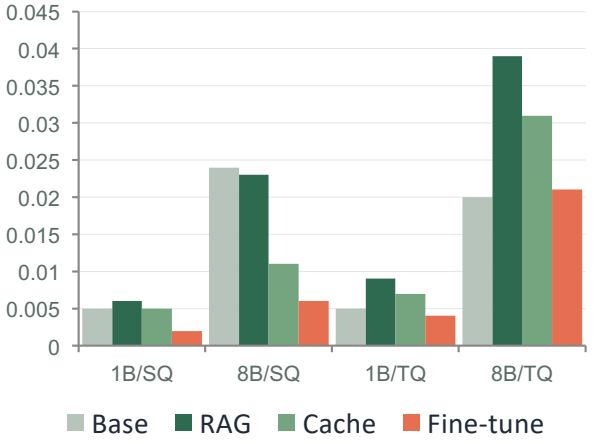
Quality (Token F1)
Higher Better



Latency (sec / query)
Lower Better



Carbon (g CO_2 / query)
Lower Better



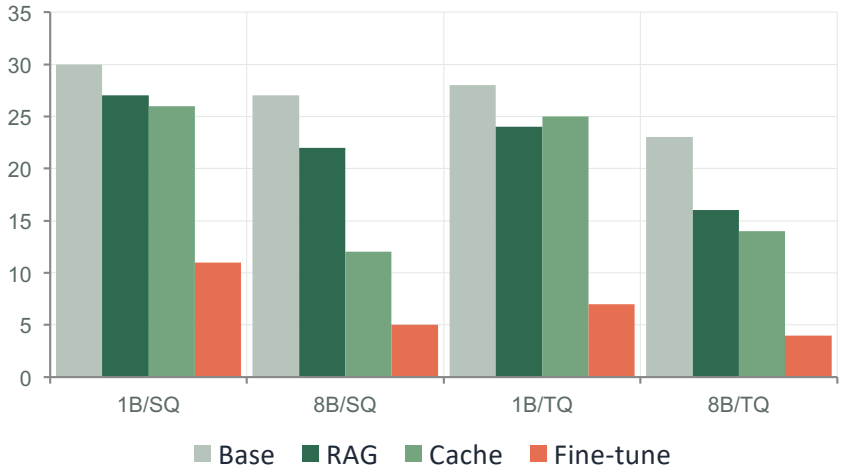
1B: Small Model · 8B: Large Model · SQ: Small Context · TQ: Large Context

Fine-Tune Wins Every Axis

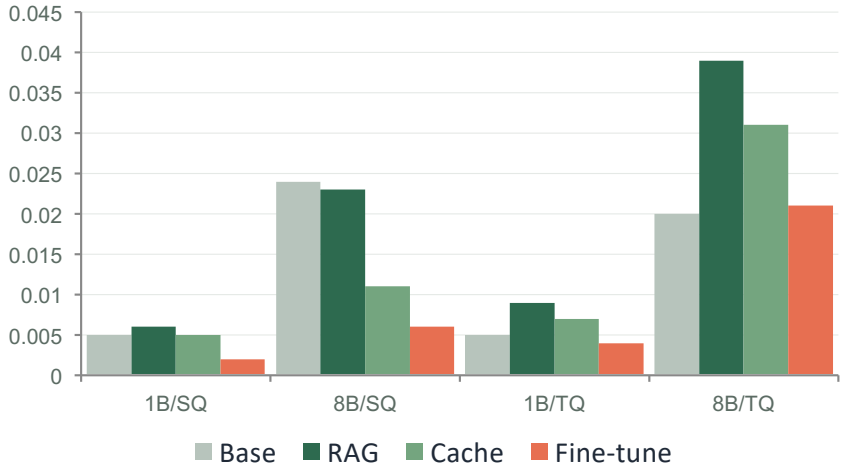
RESULTS

Why Fine-tune Wins?

Output Tokens / Query



Carbon (g CO_2 / query) Lower Better



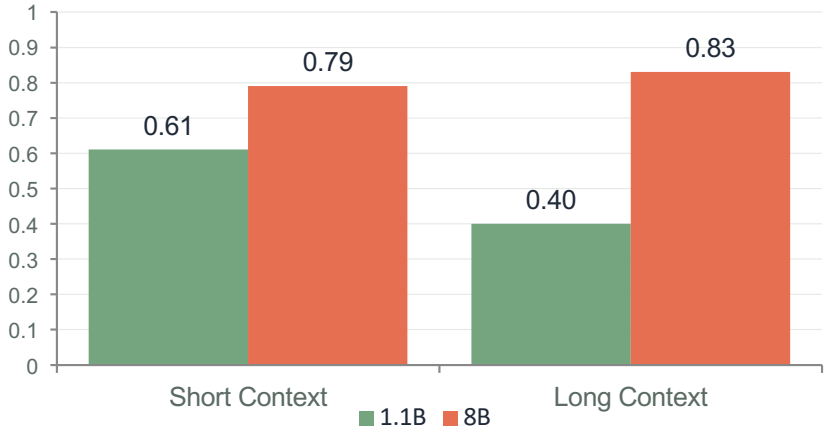
1B: Small Model · 8B: Large Model · SQ: Small Context · TQ: Large Context

Fewer Output Tokens → Less Decode Energy → Less Carbon

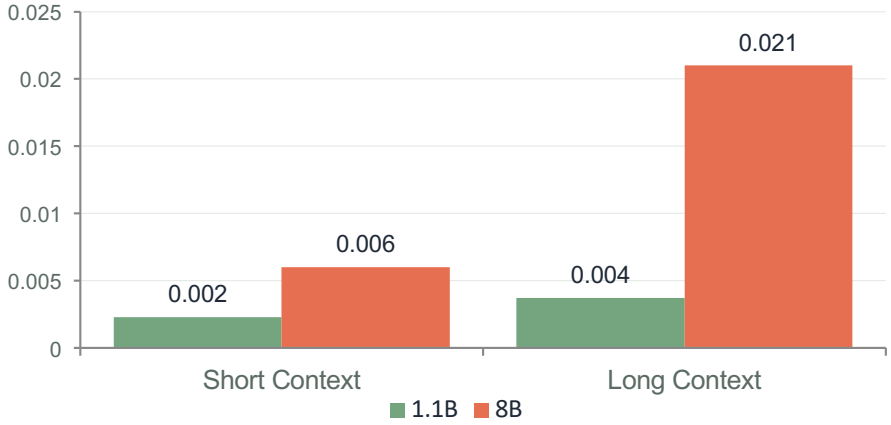
RESULTS

How Model Size Affects on Quality & Carbon

Quality (Token F1) Higher Better



Carbon (g CO_2 / query) Lower Better



1B: Small Model · 8B: Large Model · SQ: Small Context · TQ: Large Context

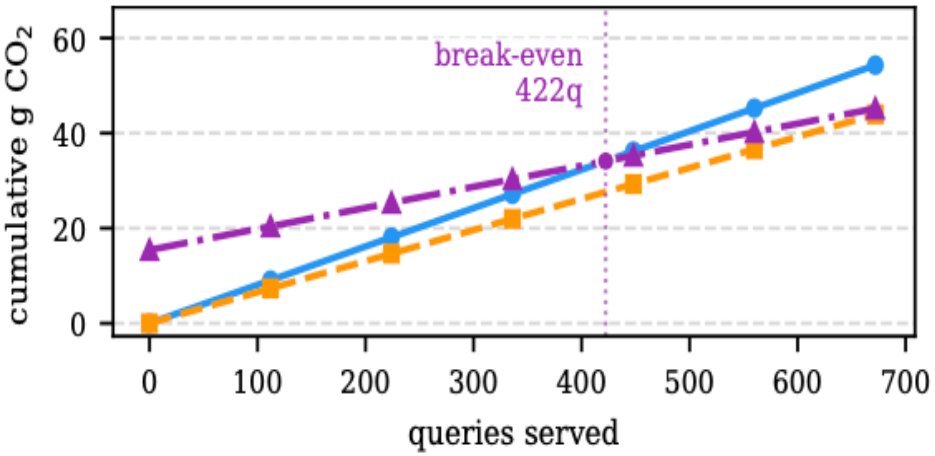
Bigger Model, Much Bigger Carbon

Smart Choice between Model Accuracy & Carbon

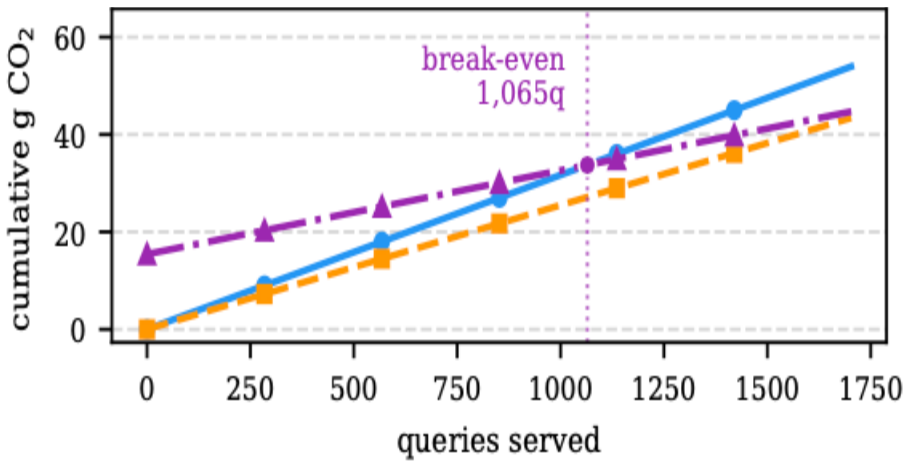
RESULTS

Cumulative Carbon vs Queries Served in Different Grids

Train in Clean Grid → Infer in Dirty Grid



Train in Clean Grid → Infer in Clean Grid



RAG Cache Fine-tune

Inference-Grid Placement Sets the Ceiling

GUIDANCE

Practitioner Playbook

1	**Fine-tune on stable knowledge**	One-time cost, ongoing savings
2	**Run inference on clean grid**	Inference cost multiplies every query
3	**Use smallest model that fits**	Bigger model = several-fold more carbon
4	**Use RAG for edge cases**	Fast-changing data, rare facts, source needed
5	**Layer caching on top**	Serving boost, not a replacement

DISCUSSION

Future Directions

Different Datasets

News summaries
Code generation

Hybrid LoRA + RAG

Fine-tune stable core
retrieve fresh facts
with carbon consideration

Distributed Systems

Multi-GPU cache reuse
Training sync cost
Inference worker placement

Thanks! Questions?

Rankyung Hong
hongx293@umn.edu

Abhishek Chandra
chandra@umn.edu

Univ. of Minnesota, Twin Cities · HotCarbon '26