

Fine-Grained Energy Accounting in Production LLM Serving

Xianyi Yuan¹, Hanlong Liao², Kunming Zhang¹,
Deke Guo^{3,*}, Guoming Tang^{1,*}

¹The Hong Kong University of Science and Technology (Guangzhou)

²National University of Defense Technology

³Sun Yat-sen University

* Corresponding author

Aug. 7, 2026

① Introduction

② Motivation

③ Methods

④ Results

1 Introduction

2 Motivation

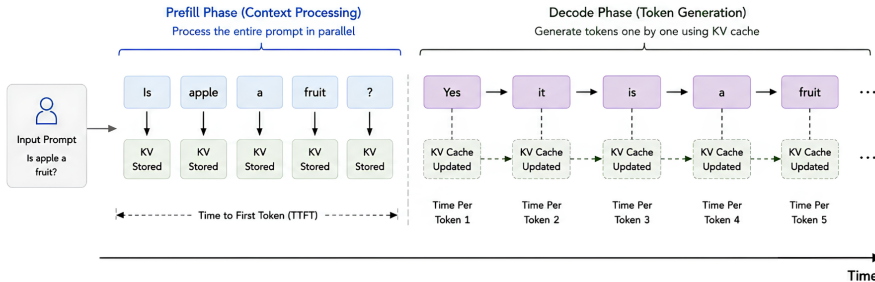
3 Methods

4 Results

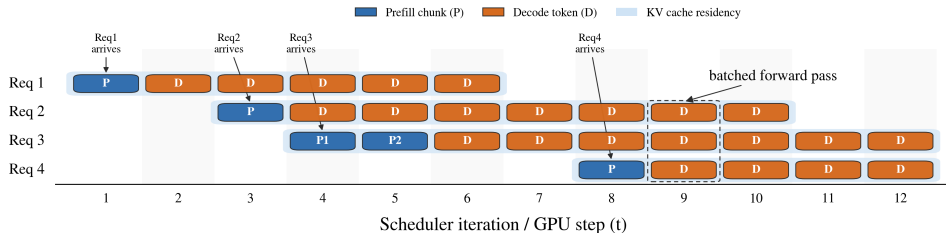
Preliminary of LLM Serving

Two phases in LLM inference

- **Prefill phase:** processes all input tokens in parallel to compute hidden states and build the KV cache. It is highly parallel and typically computation-intensive.
- **Decode phase:** generates output tokens sequentially by reusing the KV cache. It involves iterative execution and is often memory-access intensive.



Production LLM Serving



- **Prefill chunking:** splits long input prompts into smaller chunks and processes them across multiple iterations. This reduces long prefill blocking and enables better scheduling flexibility.
- **Continuous batching (ORCA'22):** schedules requests at iteration-level by dynamically merging prefill chunks and decode steps into the same batch. It improves GPU utilization and reduces serving latency under dynamic workloads.

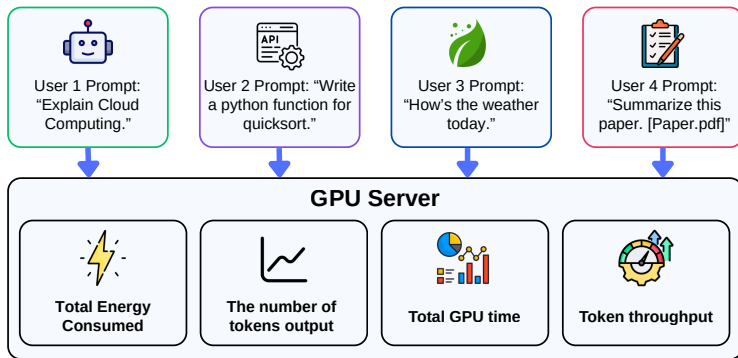
① Introduction

② Motivation

③ Methods

④ Results

Energy Attribution Problem in Production LLM Serving



Think about it, based on these indicators. How can we fairly and accurately allocate the cost to specific users? Is it fair to allocate energy consumption using tokens?

Case Study: Existing Methods Fail



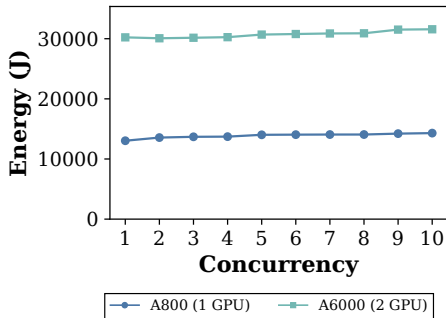
Why did the traditional methods fail here?

- **Token-based attribution:** assigns energy cost according to the number of generated tokens.

It ignores batching effects and phase-level differences, causing unfair attribution when requests have different prefill/decode behaviors.

- **Leave-one-out (LOO) attribution:** measures each request's marginal contribution by removing it from the batch.

It fails under continuous batching because interactions among concurrent requests make individual contributions non-additive.



Setup: Each request has 1 prefill token and 1024 decode tokens. The figure reports the total GPU energy consumption at different concurrency levels.

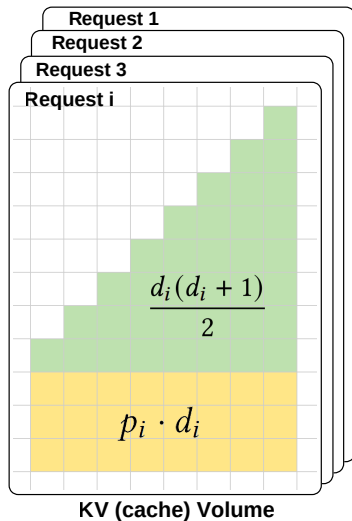
① Introduction

② Motivation

③ Methods

④ Results

KV Volume & Energy Model



KV Volume Definition

During decoding, each generated token attends to all previously stored KV cache.

We define **KV Volume** as the accumulated KV cache occupancy over decode steps:

$$Vol_i = \sum_{k=1}^{d_i} (p_i + k) = p_i \cdot d_i + \frac{d_i(d_i + 1)}{2}$$

where p_i and d_i denote prompt and decode length.

Intuition

KV Volume measures the **area under the KV growth curve**, capturing both cache size and lifetime.

A larger KV cache leads to higher attention computation and memory access cost, resulting in increased inference energy.

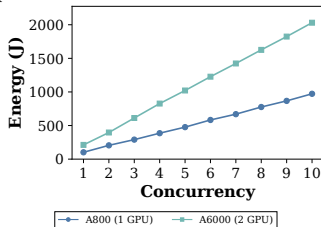
Phase-aware Energy Model



Prefill Energy

$$E_{\text{prefill}} = \sum_i \alpha_p p_i$$

- Compute-bound and highly parallel.
- Energy scales nearly linearly with prompt length.
- Prefill costs are approximately additive across requests.



Decode Energy

$$E_{\text{decode}} = \sum_{\tau} U_{\text{KV}}(\tau) \cdot \text{EPV}(U_{\text{KV}}(\tau))$$

- Memory-bound due to repeated KV-cache access.
- Energy depends on the dynamic KV occupancy.
- KV Volume captures cache size and lifetime.
- EPV models nonlinear sharing among concurrent requests.

$U_{\text{KV}}(\tau)$: total KV-cache occupancy at decode step τ .

Shapley-based Online Energy Attribution

Shapley Fairness Target

$$\phi_i^{dec} = \mathbb{E}_{\pi} [\nu(P_i^{\pi} \cup \{i\}) - \nu(P_i^{\pi})]$$

- Decode requests form a cooperative energy game.
- Coalition value $\nu(S)$ is estimated by the KV-volume energy model.
- Each request pays its average marginal contribution.
- Provides fairness under nonlinear KV-cache sharing.

P_i^{π} : requests before i in permutation π .

Monte Carlo Approximation

$$T = \left\lceil \frac{2^k}{m} \right\rceil$$

$$\hat{\phi}_i^{dec} = \frac{1}{T} \sum_{t=1}^T \Delta_i(\pi_t)$$

- Exact Shapley requires $O(m!)$ permutations.
- Random sampling estimates marginal contributions.
- Budget k controls accuracy-overhead trade-off.
- Enables online request-level accounting.

① Introduction

② Motivation

③ Methods

④ Results

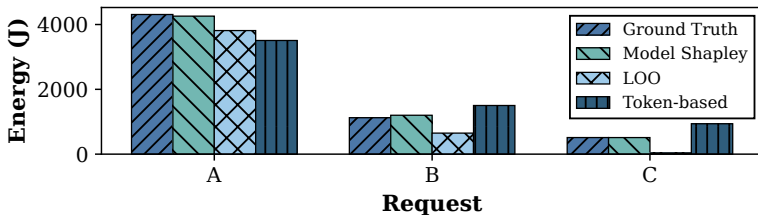
Accurate and Fair Energy Attribution



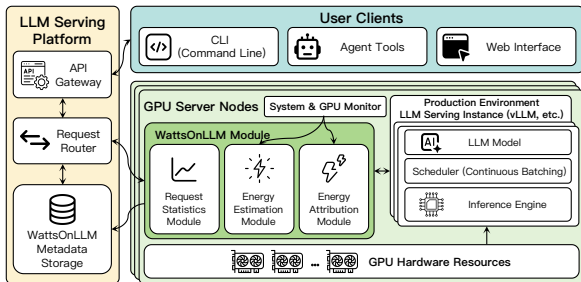
Key Observation

Model Shapley closely matches the physically measured ground truth, while token-based and LOO methods exhibit clear attribution bias.

- Compare attribution results with real GPU energy measurements.
- WattsOnLLM closely follows the ground-truth Shapley allocation.
- Token-based and LOO methods fail to capture request interactions.
- Achieves fair attribution under continuous batching.



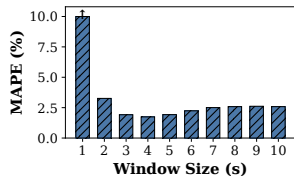
WattsOnLLM: Production Energy Accounting System



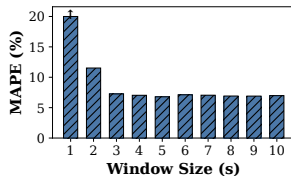
End-to-end Attribution Pipeline

- Integrates energy accounting into production LLM serving.
- Collects runtime request information and KV-cache behaviors.
- Phase-aware model estimates workload energy consumption.
- MC-Shapley provides online request-level attribution.

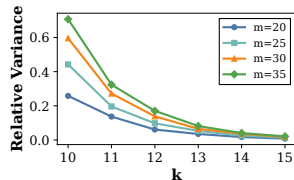
WattsOnLLM: Accurate and Efficient Online Accounting



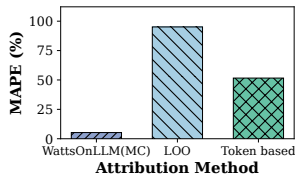
(a) Estimation error on A800



(b) Estimation error on Dual A6000



(c) MC-Shapley convergence



(d) Attribution accuracy

Key Results

- The phase-aware model achieves only **1.9%** minimum error on A800.
- Estimation error remains within **6.9%** on Dual A6000.
- MC-Shapley steadily converges as the sampling budget increases.
- WattsOnLLM reaches **5.25%** attribution error, far below LOO and token-based methods.

Acknowledgements



Xianyi Yuan

HKUST(GZ)



Hanlong Liao

NUDT



Kunming Zhang

HKUST(GZ)



Deke Guo

SYSU



Guoming Tang

HKUST(GZ)

Thank you to all collaborators.

We thank the SusCom Lab of HKUST(GZ) and all collaborators for valuable discussions, experiment support, and feedback on fine-grained energy accounting in production LLM serving.

Thank you for listening !

Question?

Xianyi Yuan, Guoming Tang*

Email: yuanxianyi1018@outlook.com, guomingtang@hkust-gz.edu.cn