

Mercury Token Economy and Internal Routing Cost

Chen, Ho Yiing (norika)¹

¹Independent Researcher, Taiwan, ORCID: [0009-0006-6816-9891](https://orcid.org/0009-0006-6816-9891), norika at
charenix.com

Draft: June 18, 2026

Abstract

Two language models can answer the same question with different numbers of output tokens, different latency, and different internal routes. Standard benchmarks usually record the outside: accuracy, throughput, or length. Mercury adds the inside. This paper defines token economy as the relation between answer quality, output token count, and measured internal routing cost.

The central claim is practical: a long answer is not always deeper reasoning, and a short answer is not always efficient. A model can waste tokens at the surface, burn hidden computation before each token, or reach a stable answer through a compact route. I define output token cost, tokenizer burden, internal activation mass, route dispersion, useful-information efficiency, and per-token internal load. These equations make it possible to ask why one model answers quickly while another drags the same question through more internal streets.

Author Note

This paper starts from a very ordinary frustration. Ask the same question to several models and the answers do not only differ in quality. They differ in length, speed, hesitation, and waste. Some models write too much. Some answer too quickly and miss the point. Some look concise while burning a huge internal route before each word.

Mercury lets us stop guessing where that cost went.

The question is small enough to measure and large enough to matter. A local model that wastes five seconds or fifty extra tokens on every ordinary request does not merely feel worse; it changes the economics of the product that wraps around it.

1 The Same Prompt Is Not The Same Work

Let x be a prompt and m a model. The generated answer is

$$y_m(x) = (y_1, \dots, y_{T_m(x)}),$$

where $T_m(x)$ is the output token count. If two models answer the same question with different $T_m(x)$, the difference may come from tokenizer segmentation, generation style, instruction tuning, uncertainty, or internal route length.

The outside number is easy to count. The reason is not.

That gap is where this paper lives.

2 Tokenizer Burden

Let $b_m(x)$ be the number of input tokens produced by model m 's tokenizer for the same text x . Define relative tokenizer burden:

$$\text{TB}_m(x) = \frac{b_m(x)}{\text{chars}(x) + \epsilon}.$$

For Chinese and Japanese prompts this term matters. A model that splits a sentence into more pieces pays more attention and cache cost before it even starts answering. The model may look slower, but part of the slowness was born at tokenization.

3 Internal Activation Mass

Let $a_{\ell,u,d}^{(m)}(x)$ be the measured activation or importance at layer ℓ , module u , and dimension d . For a threshold τ , define

$$I_m(x) = \sum_{\ell,u,d} \mathbf{1} \left[a_{\ell,u,d}^{(m)}(x) > \tau \right].$$

This is not a claim that every active site is meaningful. It is a cost measurement. A prompt that lights up many places asks the model to move through more of its body.

Mercury can draw that body instead of treating it as a hidden scalar.

The drawing is not decoration here. It is the accounting sheet.

4 Route Dispersion

A compact route and a scattered route can have the same mass. Let

$$p_i = \frac{w_i}{\sum_j w_j + \epsilon}$$

be normalized activation weight across observed route sites. Route dispersion is

$$D_m(x) = - \sum_i p_i \log(p_i + \epsilon).$$

Low dispersion means the answer traveled through a concentrated path. High dispersion means the model touched many weakly coordinated regions.

This is where token economy becomes more than a length statistic.

Short can still be expensive.

5 Per-Token Internal Load

Define per-token internal load:

$$L_m(x) = \frac{I_m(x)}{T_m(x) + \epsilon}.$$

A high $L_m(x)$ means each output token required a large observed internal route. A low $L_m(x)$ means the route was compact per generated token.

This number should be interpreted with quality. A low load with a bad answer is not efficiency; it is premature stopping.

That distinction is what keeps the metric honest.

6 Useful Information Efficiency

Let $Q_m(x) \in [0, 1]$ be task quality, judged by exact match, rubric score, or human evaluation. Output efficiency is

$$E_m^{\text{out}}(x) = \frac{Q_m(x)}{T_m(x) + \epsilon}.$$

Internal efficiency is

$$E_m^{\text{int}}(x) = \frac{Q_m(x)}{I_m(x) + \epsilon}.$$

The interesting model is not necessarily the shortest one. It is the one that keeps $Q_m(x)$ high while avoiding unnecessary output and unnecessary internal wandering.

For product work, this is the number that matters. The user does not buy a beautiful route. The user buys a good answer that arrives without wasting time, money, battery, or patience.

7 Four Failure Modes

Mercury separates four cases:

case	$T_m(x)$	$I_m(x)$
surface verbosity	high	low or medium
hidden struggle	low or medium	high
waste	high	high
compact competence	low or medium	low

Only the last case is the dream. The other three can all look acceptable if we only read the answer text.

That is why the internal route matters.

8 Latency

Let $R_m(x)$ be wall-clock response time in an environment suitable for model m . A simplified latency decomposition is

$$R_m(x) = r_{\text{tok}}b_m(x) + r_{\text{prefill}}P_m(x) + r_{\text{decode}}T_m(x) + r_{\text{route}}I_m(x) + \epsilon_m,$$

where $P_m(x)$ is prefill work and ϵ_m covers hardware/runtime noise. This equation is deliberately not pretending to be a perfect systems model. It is a checklist: tokenization, prefill, decoding, and internal route cost can all contribute.

Fast and slow are not personalities. They are measurements waiting to be separated.

In deployment, this decomposition also explains why two systems that look similar in a benchmark table can feel completely different in the hand: one spends its delay in prefill, one spends it in decoding, and one spends it in a wide internal route that was invisible until the route itself was counted.

9 Relation To Existing Work

Scaling laws relate model size, data, and compute to loss [1, 2]. Tokenization work shows that subword choices shape sequence length and language handling [3, 4]. Efficient transformer work measures runtime and attention cost [5]. Mechanistic interpretability studies circuits and internal components [6].

Mercury connects these concerns at the prompt level. It asks how many tokens were used, how much internal route mass was observed, and whether the answer was worth the cost.

10 Limits

Internal activation mass is not a perfect proxy for FLOPs. Route dispersion is not a proof of reasoning. Quality scoring can be noisy. A careful paper must report hardware, runtime, tokenizer, sampling parameters, and prompt set.

Still, the framework is useful because it prevents a lazy comparison. Bigger, slower, longer, and smarter are different words, and product decisions become dangerous when those words are treated as interchangeable. Mercury keeps them separate by forcing each answer to carry both an outside cost and an inside route receipt.

11 Conclusion

Mercury Token Economy and Internal Routing Cost turns a simple complaint into a measurable object. One model may use more tokens because its tokenizer fractured the input, another may answer slowly because prefill or route mass became expensive, and a short answer may still feel costly because each token dragged a large hidden route behind it. The answer lives in the joint space of tokenization, output length, route mass, dispersion, and quality. Once those terms are visible, token cost stops being a mystery and becomes a map.

References

- [1] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling Laws for Neural Language Models. arXiv:2001.08361, 2020. <https://arxiv.org/abs/2001.08361>
- [2] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training Compute-Optimal Large Language Models. arXiv:2203.15556, 2022. <https://arxiv.org/abs/2203.15556>
- [3] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. *Proceedings of ACL*, 2016. <https://aclanthology.org/P16-1162/>
- [4] Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. *Proceedings of EMNLP: System Demonstrations*, 2018. <https://aclanthology.org/D18-2012/>
- [5] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Re. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. arXiv:2205.14135, 2022. <https://arxiv.org/abs/2205.14135>
- [6] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A Mathematical Framework for Transformer Circuits. *Transformer Circuits*, 2021. <https://transformer-circuits.pub/2021/framework/index.html>