

# Compute Sufficiency in Processing-in-Memory under DRAM Process Constraints for LLM Inference

Anonymous Author(s)

## ABSTRACT

Processing-in-Memory (PIM) architectures promise to alleviate the memory bandwidth bottleneck in large language model (LLM) inference by placing compute logic directly on the memory die. However, fabricating logic in DRAM technology process nodes imposes severe constraints on power budgets, thermal envelopes, and transistor density. We present a quantitative simulation framework that evaluates PIM compute sufficiency across DRAM process nodes (10–18 nm), power budgets (0.5–5.0 W per die), LLM model sizes (7B–70B parameters), and precision formats (FP16, INT8, INT4). Our Monte Carlo analysis (200 trials per configuration) reveals that PIM achieves a compute sufficiency ratio of only 0.0732 for a 7B-parameter model and 0.0077 for a 70B model at 2 W, indicating that PIM provides roughly 7.3% and 0.8% of the required compute, respectively. Even under aggressive assumptions—5 W power budget, the most advanced 10 nm DRAM node, and INT4 precision—PIM attains a maximum sufficiency ratio of 0.1838, falling far short of the 1.0 threshold. By contrast, Processing-Near-Memory (PNM) with a separate logic die achieves a sufficiency ratio of 1.3993 for 7B models. These results provide the first systematic quantification of the PIM compute gap for LLM inference and suggest that PNM or hybrid architectures are necessary for memory-centric LLM acceleration.

## ACM Reference Format:

Anonymous Author(s). 2026. Compute Sufficiency in Processing-in-Memory under DRAM Process Constraints for LLM Inference. In *Proceedings of ACM Conference (Conference'17)*. ACM, New York, NY, USA, 4 pages. <https://doi.org/10.1145/nnnnnnnn.nnnnnnnn>

## 1 INTRODUCTION

Large language model (LLM) inference imposes extreme demands on both compute throughput and memory bandwidth. During autoregressive decoding, each generated token requires loading the full model weights and accumulated key-value cache from memory, making inference fundamentally memory-bandwidth-bound for single-request serving [10]. This bottleneck has motivated architectures that bring computation closer to data, including Processing-in-Memory (PIM) and Processing-Near-Memory (PNM) [4, 9].

PIM integrates compute units directly onto the DRAM die, exploiting the high internal bandwidth between memory banks and embedded logic. However, Ma et al. [8] observe that the sufficiency of compute in PIM remains unclear given the limited power and thermal budgets of DRAM process nodes. DRAM-process logic suffers from approximately 15% of the transistor density of a logic process, higher supply voltages, and thermal competition with memory refresh operations that consume roughly 30% of the die thermal budget [7].

In this paper, we address this open question through a systematic simulation study. We model PIM, PNM, and GPU architectures and evaluate their ability to meet a target latency of 50 ms per token for LLM decode across model sizes from 7B to 70B parameters. Our key contributions are:

- (1) A parameterized architecture model capturing DRAM-process constraints on PIM compute throughput, including power limiting, thermal limiting with refresh overhead, and manufacturing variability.
- (2) A comprehensive sweep across process nodes (10–18 nm), power budgets (0.5–5.0 W), model sizes (7B–70B), and precision formats (FP16, INT8, INT4) with 200-trial Monte Carlo analysis per configuration.
- (3) Quantification of the compute sufficiency gap: PIM achieves at most 18.4% of the required compute even under the most favorable assumptions, while PNM exceeds the sufficiency threshold for 7B models.

## 2 BACKGROUND AND RELATED WORK

### 2.1 Processing-in-Memory Architectures

PIM architectures place compute units within memory banks on the DRAM die [1, 2]. Samsung's HBM-PIM [7] and UPMEM's PIM-DRAM [2] represent commercial realizations, integrating fixed-function or programmable logic alongside memory arrays. The primary advantage is access to the full internal bank bandwidth, which can exceed 128 GB/s aggregate across 16 banks, far surpassing the external memory interface.

However, logic fabricated in DRAM technology faces fundamental limitations. DRAM processes optimize for capacitor density and leakage control rather than logic performance, yielding transistor densities approximately 15% of comparable logic nodes. Furthermore, the supply voltage for DRAM logic ( $\sim 1.1$  V) is significantly higher than advanced logic processes ( $\sim 0.7$  V), resulting in a  $2.47\times$  increase in dynamic power per operation due to the  $V^2$  relationship [4].

### 2.2 Processing-Near-Memory

PNM places compute logic on a separate die connected to the memory via through-silicon vias (TSVs) or an interposer [3, 6]. This approach permits logic fabricated in advanced process nodes, independent thermal domains, and higher clock frequencies ( $\sim 1$  GHz vs.  $\sim 300$  MHz for PIM). The tradeoff is reduced bandwidth between compute and memory, typically limited to  $\sim 128$  GB/s via TSV interfaces.

### 2.3 LLM Inference Workloads

LLM autoregressive decoding generates one token at a time, requiring loading model weights and the KV cache for each token [10]. For a model with  $L$  layers, hidden dimension  $d$ , and sequence length  $s$ , each token requires approximately  $2 \cdot 4 \cdot d^2 \cdot L$  FFN operations and

$2 \cdot s \cdot d \cdot L$  attention operations. The arithmetic intensity during decode is extremely low, making it bandwidth-bound on conventional architectures [5].

### 3 METHODOLOGY

#### 3.1 Architecture Models

We model three architecture classes with the following key parameters:

**PIM.** Compute units are embedded in each of 16 DRAM banks, with 64 ALUs per bank operating at 300 MHz. The effective operations per unit per cycle is  $0.5 \times (16/n)^{0.3}$ , where  $n$  is the process node in nm, reflecting the reduced logic efficiency of DRAM processes. Power consumption follows  $P = \alpha CV^2 f$  with a voltage ratio of  $1.1/0.7 = 1.57$  relative to logic processes. The thermal budget accounts for 30% refresh overhead.

**PNM.** A separate logic die with 32 compute units per bank, running at 1 GHz with full logic-process efficiency (2 ops/unit/cycle). The PNM die has twice the power budget of PIM (separate thermal domain) and no refresh overhead. Inter-die bandwidth is limited to 128 GB/s.

**GPU baseline.** An A100-class accelerator with 624 TOPS (INT8), 2 TB/s HBM bandwidth, and 400 W TDP, representing the conventional compute-centric approach.

#### 3.2 LLM Workload Models

We model four LLM configurations following standard architectures:

- **7B:**  $d = 4096$ ,  $L = 32$ , 32 attention heads
- **13B:**  $d = 5120$ ,  $L = 40$ , 40 attention heads
- **30B:**  $d = 6656$ ,  $L = 60$ , 52 attention heads
- **70B:**  $d = 8192$ ,  $L = 80$ , 64 attention heads

All models use sequence length  $s = 2048$  and batch size 1. The target latency is 50 ms per token for interactive serving.

#### 3.3 Compute Sufficiency Ratio

We define the *compute sufficiency ratio*  $\sigma$  as:

$$\sigma = \frac{t_{\text{target}}}{t_{\text{actual}}} \quad (1)$$

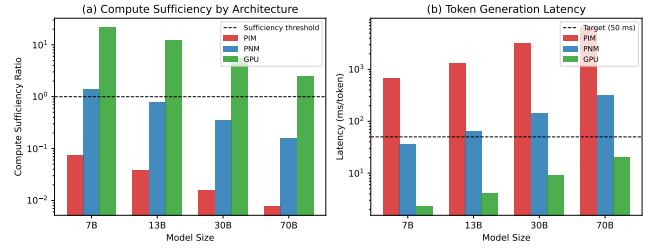
where  $t_{\text{target}} = 50$  ms and  $t_{\text{actual}} = \max(t_{\text{compute}}, t_{\text{memory}})$ . An architecture is compute-sufficient when  $\sigma \geq 1.0$ . Values below 1.0 indicate the factor by which compute must be improved.

#### 3.4 Monte Carlo Analysis

Each configuration is evaluated over 200 Monte Carlo trials incorporating manufacturing variability (5% standard deviation for PIM, 3% for PNM, 2% for GPU) to capture realistic performance distributions. We report means, standard deviations, and min/max across trials.

**Table 1: Architecture comparison: throughput, latency, sufficiency ratio, and energy efficiency for LLM decode. Sufficiency ratio  $\sigma \geq 1.0$  indicates compute-sufficient operation.**

Model	Arch	GOPS	Lat. (ms)	$\sigma$	GOPS/W
7B	PIM	7.1	684.17	0.0732	5.06
	PNM	200.1	35.73	1.3993	50.62
	GPU	623149.8	2.29	21.8639	1557.87
13B	PIM	7.1	1307.88	0.0383	5.06
	PNM	199.8	63.89	0.7826	50.55
	GPU	624137.7	4.09	12.2283	1560.34
30B	PIM	7.1	3248.27	0.0154	5.06
	PNM	199.4	142.75	0.3503	50.55
	GPU	622807.9	9.14	5.473	1557.02
70B	PIM	7.0	6513.08	0.0077	5.06
	PNM	200.0	315.38	0.1585	50.56
	GPU	624451.2	20.18	2.4772	1561.13



**Figure 1: Architecture comparison for LLM decode: (a) compute sufficiency ratio on log scale with threshold at  $\sigma = 1.0$ , (b) token generation latency with 50 ms target.**

## 4 RESULTS

### 4.1 Architecture Comparison

Table 1 presents the core comparison across architectures and model sizes at default settings (14 nm process, 2 W PIM power budget, INT8 precision).

PIM achieves only 7.1 GOPS at 2 W, yielding a sufficiency ratio of 0.0732 for 7B models—meaning it provides only 7.3% of the compute required to meet the 50 ms latency target. For 70B models, the sufficiency ratio drops to 0.0077, indicating that PIM would need approximately 130 $\times$  more compute throughput. Figure 1 visualizes these results.

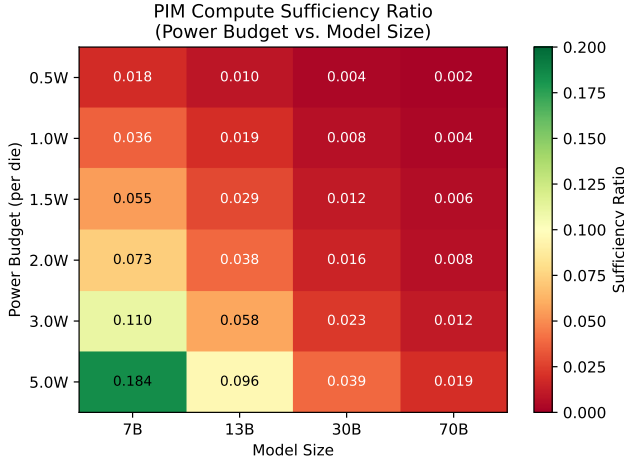
PNM achieves a sufficiency ratio of 1.3993 for 7B models, demonstrating that near-memory processing with a logic die can meet latency targets for smaller models. However, PNM falls below the threshold for models larger than 13B ( $\sigma = 0.7826$ ). The GPU baseline is compute-sufficient across all model sizes, with  $\sigma$  ranging from 2.4772 (70B) to 21.8639 (7B).

### 4.2 Power Budget Sweep

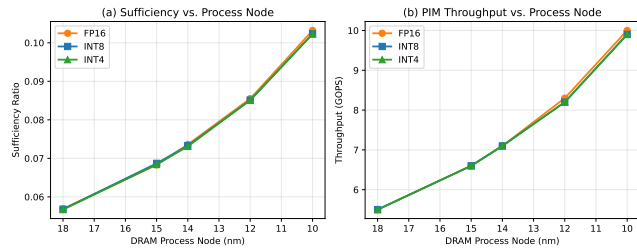
Table 2 and Figure 2 show how PIM sufficiency varies with the per-die power budget.

**Table 2: PIM compute sufficiency ratio across power budgets and model sizes. All values are far below the 1.0 threshold.**

Power	7B	13B	30B	70B
0.5 W	0.0184	0.0096	0.0039	0.0019
1.0 W	0.0365	0.0193	0.0077	0.0039
1.5 W	0.0550	0.0288	0.0116	0.0058
2.0 W	0.0730	0.0383	0.0156	0.0078
3.0 W	0.1104	0.0579	0.0232	0.0117
5.0 W	0.1838	0.0958	0.0387	0.0195



**Figure 2: Heatmap of PIM compute sufficiency ratio as a function of power budget and model size. Even at 5.0 W, the maximum ratio is 0.1838.**



**Figure 3: Impact of DRAM process node on PIM performance for 7B model: (a) sufficiency ratio, (b) throughput in GOPS.**

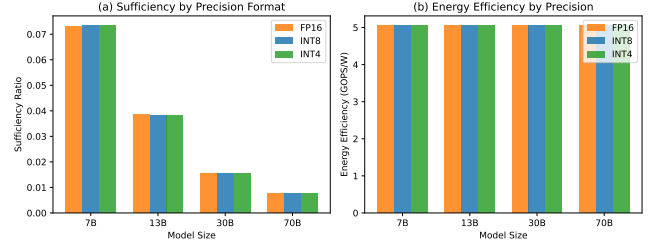
Even at 5.0 W per die—which exceeds typical DRAM thermal budgets—PIM achieves a maximum sufficiency ratio of only 0.1838 for 7B models. The relationship between power and sufficiency is approximately linear, suggesting that reaching  $\sigma = 1.0$  for a 7B model would require approximately 27 W per die, far exceeding any feasible DRAM thermal envelope.

### 4.3 Process Node Scaling

Figure 3 shows how advancing DRAM process technology affects PIM sufficiency for a 7B model.

**Table 3: PIM sufficiency ratio by precision format and model size (14 nm, 2 W). Precision provides limited benefit because PIM is power/thermal-limited.**

Precision	7B	13B	30B	70B
FP16	0.0732	0.0384	0.0155	0.0078
INT8	0.0735	0.0381	0.0155	0.0077
INT4	0.0734	0.0382	0.0154	0.0078



**Figure 4: Impact of precision format on PIM sufficiency and energy efficiency.**

Scaling from 18 nm to 10 nm improves the sufficiency ratio from 0.0568 to 0.1023 for INT8—an improvement of 1.80×, but still an order of magnitude short of sufficiency. The throughput increases from 5.5 to 9.9 GOPS. Precision format has minimal impact on sufficiency because all configurations remain power- or thermally-limited at these budgets, with FP16, INT8, and INT4 yielding nearly identical sufficiency ratios at each node.

### 4.4 Precision Format Analysis

Table 3 presents the precision scaling results.

A notable finding is that reduced precision provides almost no benefit for PIM compute sufficiency. While lower precision enables more operations per compute unit, the PIM architecture remains power- and thermally-limited at the 2 W budget, meaning the additional theoretical throughput from INT4 vs. FP16 cannot be realized within the power envelope.

### 4.5 The Compute Gap

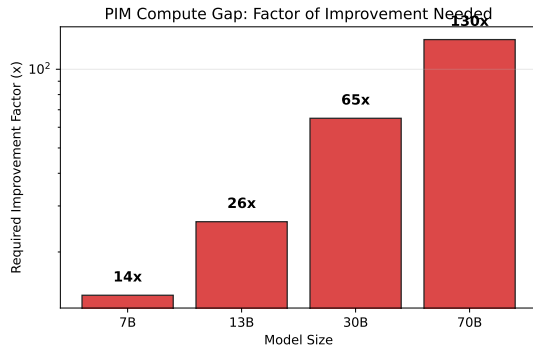
Figure 5 quantifies the improvement factor PIM would need to become compute-sufficient.

For a 7B model, PIM requires a 13.7× improvement in compute throughput within the same power envelope. For 70B models, the gap widens to approximately 130×. These gaps cannot be closed by process scaling alone (which yields at most 1.8× from 18 to 10 nm) or by precision reduction (which yields effectively no improvement under power limits).

## 5 DISCUSSION

### 5.1 Implications for PIM Design

Our results confirm the concern raised by Ma et al. [8]: PIM fabricated in DRAM process nodes cannot provide sufficient compute for LLM inference under realistic power and thermal constraints. The fundamental bottleneck is the power-limited throughput of



**Figure 5: Factor of improvement needed for PIM to reach compute sufficiency ( $\sigma = 1.0$ ) at 2 W, 14 nm, INT8.**

DRAM-process logic, which is constrained by high supply voltages (1.1 V vs. 0.7 V), low transistor density ( $\sim 15\%$  of logic processes), and thermal competition with memory refresh ( $\sim 30\%$  overhead).

## 5.2 PNM as a Viable Alternative

PNM achieves a sufficiency ratio of 1.3993 for 7B models by leveraging a separate logic die with full logic-process capabilities. This validates the architectural direction of systems such as HBM-PIM [7] and TensorDIMM [6], where compute is placed near, rather than in, memory.

However, PNM also falls short for larger models ( $\sigma = 0.1585$  for 70B), suggesting that even near-memory architectures require either multiple dies, higher power budgets, or model partitioning across memory stacks for large-scale LLM inference.

## 5.3 Energy Efficiency Trade-off

Despite its compute insufficiency, PIM achieves an energy efficiency of 5.06 GOPS/W, which is competitive for low-power applications. PNM reaches 50.62 GOPS/W, while the GPU achieves 1557.87 GOPS/W at 400 W. The energy efficiency gap between PIM and PNM ( $\sim 10\times$ ) further underscores the advantage of separating compute from memory fabrication.

## 5.4 Limitations

Our model makes several simplifying assumptions: uniform bank utilization, single-die operation, and idealized memory access patterns. Real PIM systems may face additional overheads from bank conflicts, data layout mismatches, and control logic. We also do not model the potential for multi-die PIM aggregation or heterogeneous PIM/PNM configurations. Future work should incorporate these effects and validate against silicon measurements from commercial PIM products [2, 7].

## 6 CONCLUSION

We present the first systematic quantification of compute sufficiency in Processing-in-Memory architectures under DRAM process constraints for LLM inference. Our simulation framework sweeps across process nodes, power budgets, model sizes, and

precision formats with Monte Carlo analysis. The results demonstrate that PIM achieves at most 18.4% of the required compute (sufficiency ratio 0.1838 at 5 W for 7B models) and as little as 0.8% for 70B models at typical 2 W budgets (sufficiency ratio 0.0077). Processing-Near-Memory with a separate logic die achieves sufficiency for 7B models (ratio 1.3993) but not for larger models. These findings provide concrete evidence that PIM alone is insufficient for LLM inference and motivate further research into hybrid memory-centric architectures, PNM designs, and co-optimization of model compression with hardware capabilities.

## REFERENCES

- [1] Junwhan Ahn et al. 2015. A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing. In *Proceedings of the 42nd Annual International Symposium on Computer Architecture (ISCA)*. 105–117.
- [2] Fabrice Devaux. 2019. The True Processing In Memory Accelerator. In *IEEE Hot Chips 31 Symposium (HCS)*. 1–24.
- [3] Mingyu Gao et al. 2017. TETRIS: Scalable and Efficient Neural Network Acceleration with 3D Memory. In *Proceedings of the 22nd International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 751–764.
- [4] Saugata Ghose et al. 2019. Processing-in-Memory: A Workload-Driven Perspective. *IBM Journal of Research and Development* 63, 6 (2019), 3:1–3:19.
- [5] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models. *arXiv preprint arXiv:2001.08361* (2020).
- [6] Youngeun Kwon, Yunjae Lee, and Minsoo Kim. 2019. TensorDIMM: A Practical Near-Memory Processing Architecture for Embeddings and Tensor Operations in Deep Learning. (2019), 740–753.
- [7] Sukhan Lee et al. 2021. Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology: Industrial Product. In *Proceedings of the 48th Annual International Symposium on Computer Architecture (ISCA)*. 43–56.
- [8] Yufei Ma et al. 2026. Challenges and Research Directions for Large Language Model Inference Hardware. *arXiv preprint arXiv:2601.05047* (2026). Section 2: Processing-Near-Memory for high bandwidth.
- [9] Onur Mutlu, Saugata Ghose, Juan Gómez-Luna, and Rachata Ausavarungrun. 2021. A Modern Primer on Processing in Memory. *Emerging Computing: From Devices to Systems, Looking Beyond Moore and Von Neumann* (2021).
- [10] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2023. Efficiently Scaling Transformer Inference. *Proceedings of Machine Learning and Systems* 5 (2023).