# LLM Inference Performance on Chiplet-based Architectures and Systems

Surim Oh[1]
soh31@ucsc.edu
UC Santa Cruz
Santa Cruz, CA, USA

Eric Qin
ecqin@meta.com
Meta
Sunnvale, CA, USA

Yang Yang
yyn@meta.com
Meta
Sunnvale, CA, USA

Mengchi Zhang
mengchi@meta.com
Meta
Menlo Park, CA, USA

Raj Parihar
parihar@meta.com
Meta
Sunnvale, CA, USA

Ashish Pandya
ashishpandya@meta.com
Meta
Sunnvale, CA, USA

## Abstract

Large Language Models (LLMs) have become increasingly prevalent, enabling a wide range of tasks across various platforms, from handheld devices and wearables to large-scale datacenters. Many of these applications, such as co-pilot and chatbot, rely on decoder-only style LLMs with multi-billion parameters, which require significant computational resources to achieve desired performance metrics. As LLM workloads continue to evolve and demand more substantial computational resources, it is essential to explore innovative approaches to improve their performance.

One promising approach is the Multi-Chip-Module (MCM) architecture, which offers high-performance computing, storage, and network capabilities. However, the performance characteristics of LLMs on MCM architectures are not yet fully understood. To address this knowledge gap, we conducted a series of carefully designed experiments to investigate LLM inference performance on various MCM architectures. Our study provides detailed sensitivity analyses of die-to-die bandwidth, cache policies, and chiplet configurations, offering valuable insights into optimizing LLM performance on MCM architectures.

***Keywords*** – LLM, Chiplet, Multi-Chip-Module (MCM), GPUs, Prefill, Decode

## 1 Introduction

Increased deployment of LLMs has led to an unprecedented demand for compute and memory bandwidth at all levels. Legacy systems were not designed to handle the massive computational requirements of trillion-parameter models, and system designers are now facing the challenge of finding novel solutions to overcome the "memory wall" problem for LLMs. Recent advancements in data center accelerators have led to the integration of multiple dies (chiplets[2]) into a single package, enabling the maintenance of performance scaling [2, 8, 10, 15]. The NVIDIA Blackwell [4] and d-Matrix Corsair [5] architecture exemplifies this trend by merging multiple dies into a single unified GPU, thereby incorporating a substantial amount of computing power. The NVIDIA DGX B200, configured with eight Blackwell GPUs where two dies are merged into each single GPU, delivers unparalleled generative AI performance.

One of the apparent advantages of a chiplet-based approach is its ability to improve yield issues when chips approach the reticle limit. However, we have observed that chiplet-based computing, connected with die-to-die interfaces, can also enable more efficient sharing of data, thereby reducing off-chip memory bandwidth requirements or even eliminating them altogether. Our objective is to investigate the impact of the die-to-die interface (bandwidth, latency, and/or power) on LLM inference use cases for various MCM architectures.

## 2 Background

### 2.1 Large Language Model (LLM) Inference

Today's LLM [17] is trained on vast datasets and consists of multiple layers, including embedding, attention and feed forward layers. In this work, we focus on Decoder-only Transformer models [14] which are adopted by most of the LLMs today such as LLaMA [9] and GPTs [3]. Parallelizing LLM inference across multiple devices can be beneficial due to the high volume of compute and memory operations involved.

In the process of LLM inference, there are two primary phases. During the prefill phase, the LLM processes the input tokens to generate the context for subsequent token generation. The prefill latency being measured as the Time To First Token [1] (TTFT) where it is usually bounded by compute. Subsequently, the output tokens are generated auto-regressively one at a time during the decode phase. The Time Per Output Token [1] (TPOT) serving as a measure of the decode latency is usually bounded by memory – reading parameters, Key and Value (KV) cache.

[1]Work done as an intern at Meta.

[2]In this paper, chiplet and multi-chip-module are used interchangeably.

### 2.2 Multi-Chip-Module (MCM) Architecture

MCM Architectures [2] are gaining traction due to reticle limit of dies, costs, and yield. Recent advancement in silicon interposers allow high bandwdith communication between chiplets [11] to minimize performance loss from on-chip wires to package-level wires. MI300A APU has a mix of three CCD (CPU) and six XCD (GPU) chiplets on top of four I/O Dies (IODs). Between the IODs are two N/S links of 3.0 TB/s/direction and two E/W links of 2.4 TB/s/direction. NVIDIA B200 GPU has two I/O dies connected by a link of 5TB/s/direction [13] as shown in Figure 1. Recent studies have shown that enabling reuse across chiplets can help mitigate these effects[10].

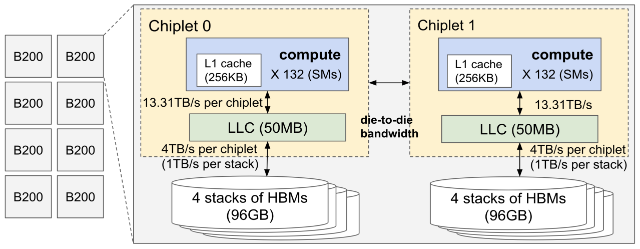

**Figure 1.** Simulated system: B200 x 8

#### 2.2.1 LLC Caching Policies.

Due to the large tensor sizes in state-of-the-art LLMs, tensor data is distributed across HBM stacks [3, 9]. To reduce memory access latency and improve performance, Last-Level Cache (LLC) are utilized in state-of-the-art MCM-based GPUs [12] to cache the data from HBM. There are two commonly used caching policies: memory-side caching and compute-side caching, as illustrated in Figure 2. With memory-side caching, the LLC only caches data from the HBM attached to the same chiplet, which we refer to in this paper as *caching@local*. In contrast, compute-side caching allows the LLC to cache data from HBM attached to other chiplets as well, which we denote as *caching@local+remote*. While compute-side caching can reduce inter-chiplet traffic for operations with high LLC data reuse, it also introduces increased complexity due to the need for cache coherence. In this paper, we evaluate the performance impact of these two caching policies.

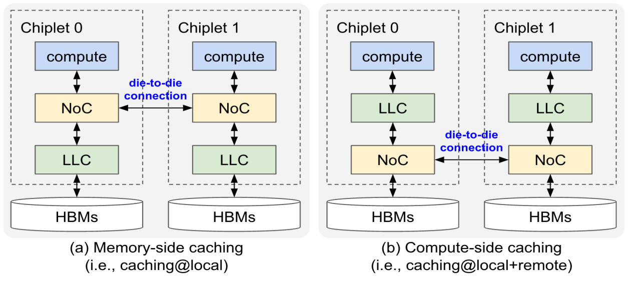
Chiplet 0
Chiplet 1
compute
NoC
LLC
HBMs
die-to-die connection
(a) Memory-side caching (i.e., caching@local)
(b) Compute-side caching (i.e., caching@local+remote)

**Figure 2.** Caching policies on an MCM-based GPU

## 3 Experimental Setup

### 3.1 LLM models

We investigate three distinct LLM models, each with a varying number of model parameters. Table 1 provides a summary of the key characteristics of each model.

| Feature | Llama3-70b [9] | GPT3-175b [3] | Llama3-400b [9] |
|---|---|---|---|
| #Params | 70 billions | 175 billions | 400 billions |
| #Layers | 80 | 96 | 126 |
| #Heads | 64 | 96 | 128 |
| Dimension | 8192 | 12288 | 16384 |

**Table 1.** Target LLM models.

### 3.2 Simulation Environment

To explore the performance of LLM inference workloads on different MCM architectures, we leverage the open-source hardware evaluation framework, LLMCompass [16]. We enhance LLMCompass by incorporating the MCM architecture, implementing various caching policies across multiple chiplets, and introducing different LLM models. Figure 1 shows the 8-device system we simulated, where all devices are fully connected via NVLink5 in an all-to-all configuration. To maximize device utilization, we employ eight-way tensor parallelism. We then sweep the die-to-die bandwidth to assess its effect on die-to-die communication. All experimentation in this study utilized FP16 numerics.

## 4 Performance Analysis

The computational graph of a Transformer is comprised of a stack of decoder layers which are composed of a sequence of operators, including matrix multiplication (Matmul), Softmax, layer normalization (LayerNorm), and activation function (GELU [3, 7]). We investigate the impact of die-to-die bandwidth on the key operators for three different LLM models, different MCM architectures, and LLC caching policies. For this performance analysis, we use the batch size of 32 and the input sequence length of 8192.

### 4.1 Die-to-Die Impact on LLM Models

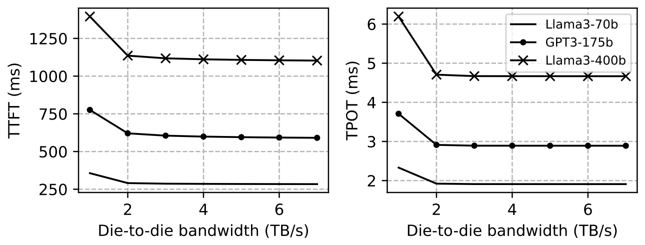

**Figure 3.** Latency of 70b/175b/400b LLMs on an 8-devices 2-chiplets B200 system with a sweep of die-to-die bandwidth.

Figure 3 shows the prefill and decode latency of three LLM models. Larger models have longer latency, but the speed ratio does not match the parameter count ratio. Specifically, Llama3-400b's prefill latency is ~2.5x that of Llama3-70b, while its decode latency is ~1.6x. Low die-to-die bandwidth negatively impacts latency for all models. As bandwidth increases, latency decreases and eventually plateaus at around 4 TB/s. However, larger models require slightly more bandwidth to achieve saturation.

### 4.2 Die-to-die Impact on Operators

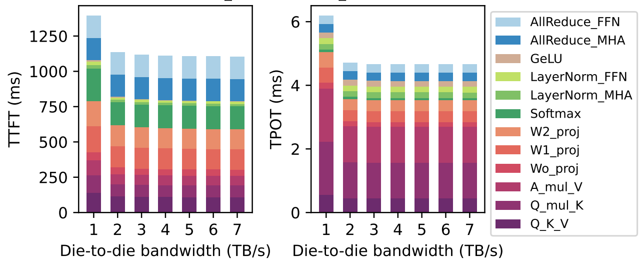

**Figure 4.** Latency breakdown on 2-chiplets system for Llama3-400b.

Figure 4 shows prefill and decode latency on a two-chiplets system across die-to-die bandwidth. The performance gain begins to saturate at 4 TB/s, when the bidirectional die-to-die bandwidth matches the HBM bandwidth of 8 TB/s. Increasing die-to-die bandwidth from 1 TB/s to 2 TB/s results in a ~1.2x speedup for decode latency, but further increases have diminishing performance improvement. With *caching@local+remote*, fusion, and KV cache for reuse across operators, there is little die-to-die traffic. Matrix multiplication is highly sensitive to die-to-die bandwidth due to the substantial amount of tensor reads and writes required from the last level cache (LLC) or high-bandwidth memory (HBM). During the prefill phase, all operators except AllReduce exhibit decreasing latency as die-to-die bandwidth increases. AllReduce is a collective communication operation that takes place across multiple devices.

We examine how different MCM architectures affect LLM inference, keeping the peak PFLOPS of a single device and per-device HBM bandwidth constant. Figure 5 illustrates the impact of die-to-die bandwidth on various MCM designs: 8, 4, 2-chiplet, and 1 chiplet, all with a peak throughput of 2250 TFLOPS. The 2-chiplet design mirrors NVIDIA's B200 architecture, while the 8-chiplet and 4-chiplet designs use a ring topology for die-to-die transfer, where each chiplet has two neighbors for communication.

One chiplet (monolithic) does not have any die-to-die links, and is the baseline. Figure 5 shows that increasing die-to-die bandwidth will converge to monolithic performance for both prefill and decode stages. The performance scales with increasing die-to-die bandwidth until the bidirectional die-to-die bandwidth reaches the HBM bandwidth. At this point, the HBM bandwidth becomes a bottleneck and further increases in die-to-die bandwidth do not result in improved performance. The performance degradation for a die-to-die bandwidth of 1 TB/s is greater for two chiplets than for eight chiplets, because the total die-to-die bandwidth is higher in the latter case.

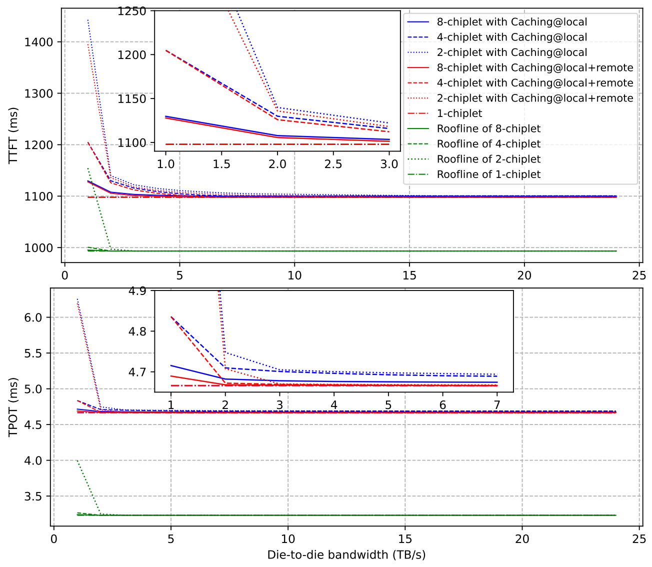

**Figure 5.** Latency w/ MCMs and caching policies (Llama-400b). The total die-to-die bandwidth is 8/4 times higher for 8/4-chiplet devices compared to 2-chiplet devices. For example, a 4-chiplet device has a total bandwidth of 20 TB/s (from two N/S and two E/W) when the die-to-die bandwidth is 5 TB/s, while an 8-chiplet device has a total bandwidth of 40 TB/s (from six N/S and two E/W).

### 4.3 LLC Caching Policies

The *caching@local+remote* policy exhibits a lower latency than that of the *caching@local* policy, owing to the optimal reuse of data from local LLC. Even without introducing more chiplets, the *caching@local+remote* policy achieves lower decode latency. For example, TPOT of 2-chiplet/4-chiplet with *caching@local+remote* is lower than that of 4-chiplet/8-chiplet with *caching@local* respectively even with a large die-to-die bandwidth. In the case of *caching@local*, half of the data required for a reuse must always come from a remote chiplet, where die-to-die bandwidth becomes a bottleneck.

## 5 Conclusion and Future Work

We assessed various MCM design alternatives, encompassing number of chiplets within a device, die-to-die bandwidth, and LLC caching methodologies. Our performance projections yield several findings. Firstly, the broader die-to-die bandwidth facilitates the transfer of substantial data volumes between chiplets. The incorporation of more chiplets within a single device results in an augmented cumulative die-to-die bandwidth, which in turn enhances the overall throughput. Lastly, caching at both local and remote chiplets increases data reuse on local and reduces die-to-die traffics and the need for additional chiplets. We leave exploring different interconnect topologies across chiplets, capacity-related modeling, FlashAttention [6] optimizations, and quantization (FP8, etc.) as future work to better understand their impact.

## References

[1] Amey Agrawal, Anmol Agarwal, Nitin Kedia, Jayashree Mohan, Souvik Kundu, Nipun Kwatra, Ramachandran Ramjee, and Alexey Tumanov. 2024. Metron: Holistic Performance Evaluation Framework for LLM Inference Systems. *arXiv preprint arXiv:2407.07000* (2024).

[2] Akhil Arunkumar, Evgeny Bolotin, Benjamin Cho, Ugljesa Milic, Eiman Ebrahimi, Oreste Villa, Aamer Jaleel, Carole-Jean Wu, and David Nellans. 2017. MCM-GPU: Multi-Chip-Module GPUs for Continued Performance Scalability. In *Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA'24)* (Toronto, ON, Canada) *(ISCA '17)*. Association for Computing Machinery, New York, NY, USA, 320–332. https://doi.org/10.1145/3079856.3080231

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 1877–1901. https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[4] NVIDIA Corporation. 2024. *NVIDIA Blackwell Architecture Technical Brief.* Retrieved August 20, 2024 from https://resources.nvidia.com/en-us-blackwell-architecture

[5] d Matrix. 2023. *d-Matrix TCO White Paper.* Retrieved August 22, 2024 from https://www.d-matrix.ai/wp-content/uploads/2023/10/d-Matrix-WhitePaper.pdf

[6] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems* 35 (2022), 16344–16359.

[7] Dan Hendrycks and Kevin Gimpel. 2023. Gaussian Error Linear Units (GELUs). arXiv:1606.08415 [cs.LG] https://arxiv.org/abs/1606.08415

[8] Mahmoud Khairy, Vadim Nikiforov, David Nellans, and Timothy G. Rogers. 2020. Locality-Centric Data and Threadblock Management for Massive GPUs. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'20)*. 1022–1036. https://doi.org/10.1109/MICRO50266.2020.00086

[9] AI @ Meta Llama Team. 2024. *The Llama 3 Herd of Models.* Retrieved August 26, 2024 from https://scontent-sjc3-1.xx.fbcdn.net/v/t39.2365-6/453304228_1160109801904614_7143520450792086005_n.pdf?_nc_cat=108&ccb=1-7&_nc_sid=3c67a6&_nc_ohc=Ckhh9hWw6wAQ7kNvgHEQ6vK&_nc_ht=scontent-sjc3-1.xx&oh=00_AYA9-lha48rvvpSyV-lxCOcd4_8FJMLQJ4cuRDq6MhRDDg&oe=66CD9E47

[10] Yakun Sophia Shao, Jason Clemons, Rangharajan Venkatesan, Brian Zimmer, Matthew Fojtik, Nan Jiang, Ben Keller, Alicia Klinefelter, Nathaniel Pinckney, Priyanka Raina, Stephen G. Tell, Yanqing Zhang, William J. Dally, Joel Emer, C. Thomas Gray, Brucek Khailany, and Stephen W. Keckler. 2019. Simba: Scaling Deep-Learning Inference with Multi-Chip-Module-Based Architecture. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'19)* (Columbus, OH, USA) *(MICRO'19)*. Association for Computing Machinery, New York, NY, USA, 14–27. https://doi.org/10.1145/3352460.3358302

[11] Alan Smith, Eric Chapman, Chintan Patel, Raja Swaminathan, John Wuu, Tyrone Huang, Wonjun Jung, Alexander Kaganov, Hugh McIntyre, and Ramon Mangaser. 2024. 11.1 AMD InstinctTM MI300 Series Modular Chiplet Package–HPC and AI Accelerator for Exa-Class Systems. In *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, Vol. 67. IEEE, 490–492.

[12] Alan Smith, Gabriel H Loh, Michael J Schulte, Mike Ignatowski, Samuel Naffziger, Mike Mantor, Mark Fowler Nathan Kalyanasundharam, Vamsi Alla, Nicholas Malaya, Joseph L Greathouse, et al. 2024. Realizing the AMD Exascale Heterogeneous Processor Vision: Industry Product. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 876–889.

[13] Ryan Smith. 2024. *NVIDIA Blackwell Architecture and B200/B100 Accelerators Announced: Going Bigger With Smaller Data.* Retrieved September 1, 2024 from https://www.anandtech.com/show/21310/nvidia-blackwell-architecture-and-b200b100-accelerators-announced-going-bigger-with-smaller-data

[14] Thomas Wang, Adam Roberts, Daniel Hesslow, Teven Le Scao, Hyung Won Chung, Iz Beltagy, Julien Launay, and Colin Raffel. 2022. What language model architecture and pretraining objective works best for zero-shot generalization?. In *International Conference on Machine Learning*. PMLR, 22964–22984.

[15] Jieming Yin, Zhifeng Lin, Onur Kayiran, Matthew Poremba, Muhammad Shoaib Bin Altaf, Natalie Enright Jerger, and Gabriel H. Loh. 2018. Modular Routing Design for Chiplet-Based Systems. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA'18)*. 726–738. https://doi.org/10.1109/ISCA.2018.00066

[16] Hengrui Zhang, August Ning, Rohan Baskar Prabhakar, and David Wentzlaff. 2024. LLMCompass: Enabling Efficient Hardware Design for Large Language Model Inference. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA'17)*. 1080–1096. https://doi.org/10.1109/ISCA59077.2024.00082

[17] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223* (2023).