SI:Contemporary Industry Products

# Composition of Experts on the SN40L Reconfigurable Dataflow Unit

Raghu Prabhakar, Ram Sivaramakrishnan, Darshan Gandhi, Yun Du, Mingran Wang, Xiangyu Song, Kejie Zhang, Tianren Gao, Angela Wang, Karen Li, Joshua Brot, Calvin Leung, Tuowen Zhao, Mark Gottscho, Edison Chen, Kaizhao Liang, Swayambhoo Jain, Urmish Thakker, Kevin J. Brown, Kunle Olukotun,

***SambaNova Systems Inc**, Palo Alto, CA, 94303, USA*

*Abstract—Monolithic large language models (LLMs) pose significant challenges in training and serving during an active deployment. In contrast, Composition of Experts (CoE) is a modular approach that lowers the cost and complexity of training and serving. In this article, we explore unique hardware challenges for CoE models, such as lower operational intensity and the cost of switching between models. We describe the Sambanova SN40L Reconfigurable Dataflow Unit (RDU) that combines streaming dataflow and a new three-tier memory system with SRAM, HBM, and DDR DRAM. A single 8-socket SN40L Node achieves speedups between* $2\times$ *to* $13\times$ *due to aggressive operator fusion over an optimized baseline. The SN40L Node deploys Samba-CoE – a 1T (trillion) parameter CoE – with* $19\times$ *smaller machine footprint, speeds up model switching time by* $15\times$ *to* $31\times$*, and achieves an overall speedup of* $3.7\times$ *over a DGX H100 and* $6.6\times$ *over a DGX A100.*

Recent advances in the training and inference of large language models (LLMs) has taken the world by storm. State-of-the-art generative AI/ML applications include ChatGPT and Gemini, which are *monolithic LLMs* consisting *Trillions* of parameters and trained with curated datasets comprising *trillions* of tokens. Training such models costs millions of dollars. For instance, compute costs to train OpenAI's GPT-4 is estimated to be $78 million USD, and Google's Gemini Ultra to be $191 million USD [1], which limits the power of AI to a select few hyperscalers.

The ML research community has responded with ecosystems of much smaller, open source, modular models that are just as capable, but are cheaper and easier to train and serve [2]. For example, Flan-T5-XL only has 3B parameters, but it surpasses the 175B-parameter GPT-3's MMLU score by nearly 10% [3]. While Parameter-efficient Fine-tuning Techniques (PEFT) Techniques like LoRA are used to shrink expert weights, PEFT techniques do not achieve the same level of quality as Supervised Fine-Tuning (SFT) under several scenarios [4]. Consequently, the smaller expert models are often entire models that are specialized using additional training or SFT (there are over 9000 variants of Llama 2 on HuggingFace at the time of this writing). Proof points like these have bolstered community activity in building and training smaller models by specializing base models to a domain, by fine-tuning base models to a specific task or group of tasks, and by distilling or compressing larger models into smaller models. Furthermore, *compositions* of such smaller models have been shown to demonstrate emergent behavior that matches large monolithic models [5]. We believe that such modular systems (referred here as ***Composition of Experts (CoE)***) would play a pivotal role in advancing the AI frontier [6]

In this article, we first describe a composition of experts system and its unique hardware requirements. We also present the **SambaNova SN40L Reconfigurable Dataflow Unit (RDU)**, a commercial dataflow accelerator that combines **streaming dataflow parallelism** with a novel **three-tier memory system** containing large on-chip SRAM, HBM, and DDR DRAM that is directly attached to the accelerator. Later, we

XXXX-XXX © 2024 IEEE
Digital Object Identifier 10.1109/XXX.0000.0000000

quantify and discuss the impact of streaming dataflow parallelism on several real-world benchmarks, showing speedups ranging from **2**× to **13**× over an optimized baseline. We also show that for CoE inference deployments, the SN40L reduces machine footprint by up to **19**×, speeds up model switching time by **15**× to **31**×, and achieves an overall speedup of **3.7**× to **6.6**× over DGX H100 and DGX A100, respectively

## COMPOSITION OF EXPERTS

In this section, we describe one instance of a CoE built and deployed on the SN40L, called ***Samba-CoE***. Figure 1 shows the Samba-CoE pipeline from prompt to response.

Samba-CoE consists of several expert models and a router model. Each expert is fine-tuned in a specific domain. We leveraged several excellent expert models fine-tuned on domains like coding, math, and language translation from the open source community. The router is another specialist model that dynamically assigns each input prompt to the most relevant expert. For instance, a math-related query would be routed to the math expert, while a coding question would go to the code expert.

The Samba-CoE is inspired by the Mixture-of-Experts (MoE) architecture, but has key differences. While both MoEs and CoEs are more sparsely activated than traditional models, CoEs are more flexible and capable. Prior work has shown that CoEs can outperform both MoEs and large models like GPT-3.5 and GPT-4 [7]. Samba-CoE outperforms GPT-4 on the public AlpacaEval community leaderboard at the time of this writing. We note that CoEs and MoEs are orthogonal techniques that can be easily combined. A CoE can leverage expert models that are implemented internally as MoEs.

Here we use a simplified Samba-CoE with 150 Llama2 7B expert models, or 1 trillion total parameters, to expound on the system-level challenges. The CoE concept and the Samba-CoE system are not limited to Llama2.

## HARDWARE REQUIREMENTS

CoE consists of several small expert models working in tandem on a task. Outputs from one expert determine which expert(s) to execute next. Running an expert involves loading model parameter weights to the accelerator's main memory, and then executing the model. An efficient CoE system can be thought to have following capabilities:

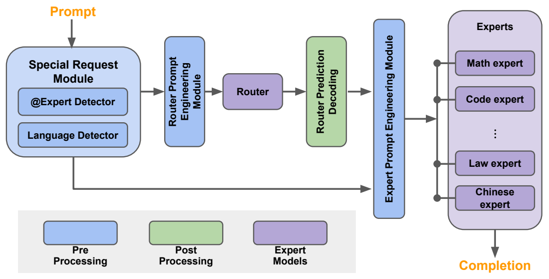
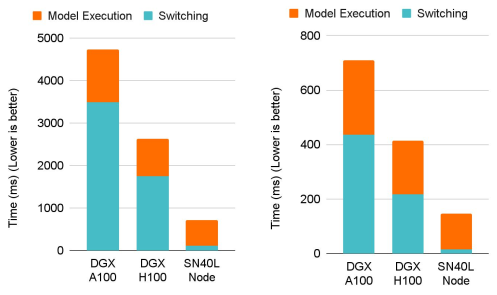
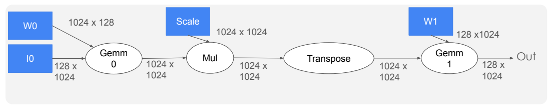


FIGURE 1: Samba-CoE Pipeline from prompt to completion.

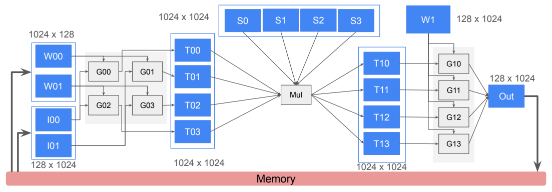


(a) BS=8, TP=8 (b) BS=1, TP=8

FIGURE 2: CoE latency breakdown between model switching and model execution to generate 20 output tokens from a Llama2-7B expert. The SN40L RDU executes CoEs efficiently by combining streaming dataflow and a novel three-tier memory hierarchy of SRAM, HBM, and DDR.



(a) An example dataflow graph showing a simplified Monarch FFT decomposition [8].



(b) A streaming dataflow implementation of Figure 3a.

FIGURE 3: FlashFFTConv operator and streaming dataflow implementation.

1) ***Aggressive* Operator Fusion and Pipeline Parallelism** to efficiently execute small, low operationally intensive models.
2) **High-Bandwidth Memory** to exploit temporal and spatial locality in weights and intermediate results during generative inference, and
3) **High-Capacity Memory** to minimize switching costs and store the parameters of many expert models

CoE execution time is broken down into model execution time and model switching time, as seen in Figure 2. Minimizing CoE execution time can be used to either reduce the machine footprint per user or increase the number of users supported under a given footprint. To reduce model execution time, we show the advantages of streaming dataflow over conventional operator fusion. To minimize model switching time, we motivate the need for both a high-capacity accelerator-local DDR interface and HBM.

## Streaming Dataflow

**Conventional Operator Fusion is Insufficient:** Operator fusion is a common optimization technique to increase operational intensity and improve hardware utilization [9]. However, expert models often contain operators with low operational intensity coupled with complex access patterns involving shuffles and transposes [8]. Complex access patterns severely restricts the efficacy of fusion on GPUs. Frameworks like PyTorch2 and TensorRT have documented restrictions on patterns that are explicitly not supported for fusion. Consequently, many complex fused kernels are still handwritten for GPUs.

Figure 3a depicts a simplified Monarch FFT decomposition [8] with tensor shapes annotated on the edges. Without fusion, this kernel is memory-bound with a low operational intensity of 39.5. If fully fused, however, the same kernel is compute-bound with a high operational intensity of 410.4. Higher operation intensity allows applications to achieve roofline performance for a given target accelerator. For instance, an A100 GPU has a TFLOPS/TBps ratio of approximately $300/2 = 150$, meaning kernels with operation intensities less than 150 FLOPs/byte are memory-bound on the A100.

However, GPUs cannot fuse all of Figure 3a for the following reasons:

1) Rigid memory hierarchy and programming model creates data movement bottlenecks: A GPU kernel is launched with a grid of thread blocks. The grid structure is fixed for the duration of the kernel. Fusing *Gemm0* and *Mul* would be trivial. However, *Transpose* forces threads to access values from threads in other SMs, triggering a data exchange across SMs via the shared cache and HBM. As there is no other means to transfer data between SMs, this lack of flexibility creates a bottleneck at the shared cache and HBM
2) Insufficient on-chip SRAM capacity forces materialization of the output of *Transpose* to HBM, preventing a fusion opportunity.
3) No pipeline parallelism exploited between operators: Higher order Monarch FFT decompositions (studied in Section ) create many small matrix multiplies that are $32 \times 32 \times 32$ or smaller, which do not utilize all SMs efficiently. However, there is abundant pipeline-level parallelism across all the matrix multiplies and element-wise operators. The GPU SIMT programming model does not provide a straightforward way to execute the operators in Figure 3a as a pipeline.

**Streaming Dataflow enables Pipelining and Automatic Fusion with Arbitrary Access Patterns:** Unlike conventional fusion, *streaming dataflow* executes operators as a coarse-grained pipeline. Tensors are tiled and streamed through this pipeline. Tiles can have *any arbitrary read and write access patterns* between operators. Operators are expressed at Tensor-level granularity and automatically fused by an optimizing compiler.

Figure 3b depicts the spatially fused implementation. Blue boxes represent on-chip buffer units, and gray boxes represent on-chip compute units. The operators `Gemm0`, `Mul`, and `Gemm1` are executed as stages in a coarse-grained pipeline. The blue memory units in between serve as decoupling stage buffers that hold intermediate results. More compute units are assigned to `Gemm0` and `Gemm1` as they account for a larger fraction of the total operations. Input and output bandwidths to and from stage buffers are matched to their respective stages by using the appropriate number of memory units. For instance, logical stage buffer `I0` is partitioned into two memory units `I00` and `I01` to match the required input bandwidth to `Gemm0`. Buffer `S0 - S3` is partitioned into four memory units for capacity reasons. The transpose operation is fused into buffers `T0*` and `T1*` as an access pattern.

We distill the observations from above into the following list of on-chip architecture features to enable streaming dataflow:

1) **Composable memory units**: Hardware should support programmable interleaving of logical addresses across multiple physical memory units.
2) **Address generation bandwidth and flexibility**: High data bandwidth requires high address gen-

eration bandwidth. Address generation hardware should allow generating multiple concurrent addresses at high throughput for arbitrarily complex address expressions

3) **Systolic and streaming compute**: ML accelerators implement systolic arrays to increase compute density for GEMM-like operations. However, in many ML models, GEMMs are frequently followed by element-wise operators and reductions which require high throughput streaming compute capability.
4) **One-to-many, many-to-one, and data reordering**: Disparities between the number of producer and consumer units create one-producer-to-many-consumers and many-producers-to-one-consumer traffic streams that also require flow control. For one-to-many, hardware support is required to create fan-out paths in the interconnect from the source to a program-decided set of destinations. For many-to-one traffic, hardware must provide a protocol to reorder data from multiple streams at the destination.

### Model Hosting and Switching Costs

HBM's limited capacity limits the number of experts that can be in a CoE when hosted on a GPU or TPU. With HBM alone, running large CoEs requires either (a) using more machines for HBM capacity, which increases costs, complicates deployment, and introduces load balancing challenges, or (b) using the host's memory, which increases switching latency, as shown in Figure 2. Higher capacity DDR memory that is attached directly to the accelerator reduces both model hosting and model switching costs. Furthermore, CoEs exhibit temporal locality in expert parameters, as they are used multiple times (during autoregressive decoding, for instance). HBM plays a key role in exploiting this temporal data locality by acting as a software-managed caching tier between DDR and SRAM.

Consequently, we conclude that systems to execute composition of smaller models need two types of off-chip memories: (1) **high-bandwidth** memory to exploit temporal locality of expert parameters, and (2) **high-capacity** memory to store expert parameters in a small footprint.

In the next section, we describe the SN40L Reconfigurable Dataflow Unit, which was designed to satisfy these requirements.

## SN40L ARCHITECTURE

The SN40L dataflow accelerator is fabricated using TSMC's 5FF process and packaged as a dual die socket using Chip-on-Wafer-on-Substrate (CoWoS) multi-chip packaging technology. Figure 4a shows the salient components of the SN40L, which are described below.

**RDU Tile**: A coarse-grained reconfigurable array of dataflow cores. Consists of Pattern Compute Units (PCUs), Pattern Memory Units (PMUs), and Address Generation and Coalescing Units (AGCUs) that are connected together in a two-dimensional mesh interconnect called the Reconfigurable Dataflow Network (RDN).

**Memory Interfaces**: The SN40L interfaces with two tiers of off-chip memories – HBM and DDR. Both memory spaces are software managed. The DDR tier can have a peak memory capacity of 1.5 TiB at a peak bandwidth of over 200 GB/s. The HBM tier has 64 GiB of capacity with a peak bandwidth of about 2 TB/s per socket.

**Die-to-Die (D2D) Interface**: SN40L tile components can stream data between two dies directly without going through off-chip memory.

**Host Interface**: SN40L interfaces with a host x86 CPU using a PCIe link. This interface supports DMA between host and device off-chip memory as well as direct communication between the host and the tile.

**Peer-to-Peer (P2P) Interfaces**: Connects an SN40L to other SN40L RDUs. A peer-to-peer protocol described in section  provides primitives to implement collective communication primitives.

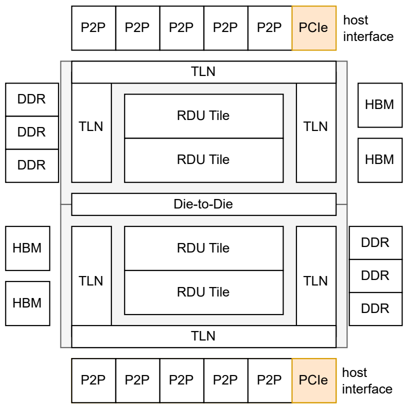
**Top Level Network (TLN)**: This network connects an SN40L tile to the host, memory, and peer-to-peer interfaces.

Figure 4b illustrates an SN40L tile with the key dataflow components: PCUs, PMUs, RDN switches, and AGCUs. The following subsections describe them in more detail.

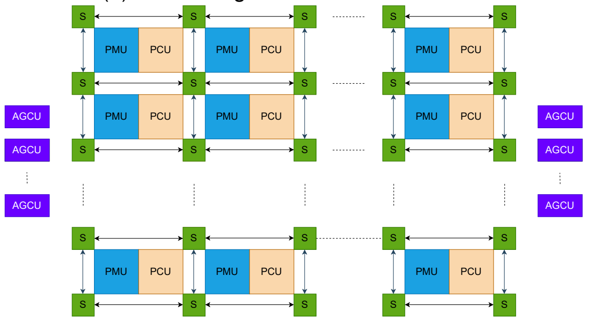
### Pattern Compute Unit (PCU)

The PCU provides systolic and streaming compute capabilities in the SN40L.

Figure 5a illustrates the PCU as both 2D systolic array and as a SIMD core. The systolic array accelerates matrix multiplications like *Gemm0* and *Gemm1* in Figure 3a. Inputs to the systolic array are streamed left-to-right and top-to-bottom through a structure called *broadcast buffer*. Accumulated results are drained left-to-right to output FIFOs through the tail unit. Matrix multiplication can be parallelized further across multiple PCUs, similar to the depiction in Figure 3b. As a SIMD core, the PCU executes a parallel multidimensional tensor operation in a fully pipelined fashion, like *Mul* in Figure 3b. Each SIMD stage supports common

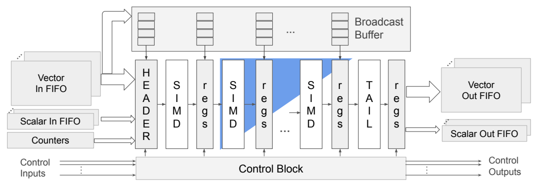


(a) Block Diagram of the 2-die SN40L

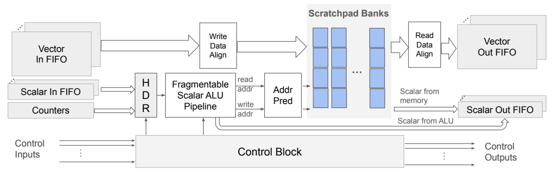


(b) SN40L Tile: PCUs, PMUs, RDN switches, AGCUs

FIGURE 4: SN40L Die and Tile Diagrams.



(a) PCU diagram showing systolic and SIMD operation with cross-lane reduction.



(b) PMU ALUs, predication, and data alignment units.

FIGURE 5: PCU and PMU Architecture

arithmetic, logical, and bit-wise operations in FP32, BF16, and INT32 formats. The PCU can be configured to implement an optional cross-lane reduction network, shown as the blue triangle in the figure. Lane-wise reductions are also supported, in pure SIMD fashion. Counters track loop iterations and generate control events on the completion of a loop.

An operation can be parallelized across multiple PCUs in a data parallel, tensor parallel, or pipeline parallel fashion. Data parallelism is achieved by partitioning inputs and outputs to create multiple independent data streams that are processed by different PCUs.

Tensor parallelism is achieved by forking into data parallel streams, then joining them. Pipeline parallelism is achieved by chaining multiple PCUs together to fuse operations and increase operational intensity.

## Pattern Memory Unit (PMU)

The PMU in SN40L provides the on-chip memory capacity, bandwidth, and addressing flexibility for efficient operator fusion. Figure 5b shows the high-level block diagram of the PMU. PMUs are used to store on-chip tensors like inputs, parameters, metadata, and intermediate results. For example, all blue blocks in Figure 3b correspond to PMUs.

The key components of a PMU include the following:

- **Scratchpad Memory:** A highly banked single-ported memory that allows for concurrent read and write accesses and supports high throughput tensor transformations such as transpose.
- **Fragmented Scalar ALU Pipeline:** A multi-stage ALU pipeline that can be fragmented to generate multiple addresses per cycle for independent, flexible read and write access patterns. Address predication support in the ALUs allows a tensor to be stored into and accessed from multiple PMU instances.
- **Data Alignment Unit:** Data alignment units enable transformations like gather, scatter, transpose, type-casting, and vector lane permutation.

## Reconfigurable Dataflow Network (RDN)

The RDN is the on-chip programmable interconnect that facilitates communication between PCUs, PMUs, and AGCUs. It consists of three physical fabrics - vector, scalar, and control. The vector and scalar fabrics are packet-switched. .The control fabric is circuit-switched. The vector fabric is the primary conduit for tensor data. The scalar fabric is mainly used to transport metadata such as addresses and scalar data. The control fabric is used to carry tokens for distributed coarse-grained flow control and, to orchestrate streaming dataflow.

The RDN is implemented using a mesh of non-blocking switches. Transmissions on the vector and scalar fabric are subject to end-to-end flow control through either fine-grained hardware credits or coarse-grained software tokens.

### Address Generation and Coalescing Unit (AGCU)

The AGCU is a reconfigurable dataflow bridge for the RDU tile to access local device memory (HBM/DDR), host memory, remote RDU device memory, and remote RDU tiles via the TLN. On the tile-side, it acts like a dataflow core by exposing RDN vector, scalar, and control ports. On the TLN-side, it generates read and write requests and coalesces the responses. It is equipped with a scalar address generation pipeline and counters, bearing some similarities to the PMU logic (sans SRAM).

**Kernel Launch Orchestration:** Running a model involves executing a schedule of kernel launches, which can be *software-orchestrated* or *hardware-orchestrated*. Software orchestration allows more flexible scheduling of kernels and provides more host software visibility into model execution. Hardware orchestration offloads a static kernel schedule to the dedicated hardware in the AGCUs, which significantly reduces the overheads of software orchestration.

## CASE STUDIES

In this section, we quantify and discuss the impact of operator fusion and *Samba-CoE*.

### Impact of Operator Fusion

Table 1 describes the set of language model benchmarks used to quantify the impact of operator fusion. Autoregressive decoding steps take advantage of the KV cache, and has much lower compute and operational intensity compared to the prefill phase which constructs the KV cache. The compute graph structure of *FlashFFTConv* [8] is a complex version of the motivating example described in Section .

All experiments other than *FlashFFTConv* are evaluated on a system containing eight SN40L sockets and one host, witht the exception of *FlashFFTConv*, which is a smaller kernel that we evaluate on a single SN40L. We measure and compare the performance of each benchmark in three configurations:

- **Unfused**: Every PyTorch operator in the model is parallelized and executed as one or more kernels on the SN40L, with intermediate results materialized to DDR or HBM.
- **Fused + Software Orchestrated (SO)**: Operators are fused into fewer kernels using a combination of automatic compiler optimizations and programmer hints. Kernel scheduling is performed from software running on the host.
- **Fused + Hardware Orchestrated (HO)**: Same fused kernels as above, but kernel scheduling is offloaded to hardware using the feature described in Section .

| **Model** | **Size** | **Sequence Length** | **Configurations** |
|---|---|---|---|
| llama2 | 7B | 4K | prefill, decode, train |
| sparseGPT | 13B | 2K | train (87.5% sparse) |
| llama2 | 70B | 4K | prefill, decode |
| bloom | 176B | 8K | prefill, decode |
| mistral | 7B | 2K, 4K | prefill, decode |
| falcon | 40B | 2K | prefill, decode |
| llava1.5-multimodal | 7B | 4K | prefill, decode |
| FlashFFTConv | N/A | 1M | FFT Convolution for long sequence models |

TABLE 1: Benchmarks and their descriptions. Here, ‘prefill’ = First token generation, ‘decode’ = Autoregressive decoding token generation with KV cache, ‘train’ = training.

Figure 6 shows the impact of operator fusion on all the benchmarks.

**Operator Fusion Speedup** With fusion on the SN40L, the entire *FlashFFTConv* benchmark is executed with a single kernel launch, in a manner similar to the simplified diagram shown in 3b. The increased operation intensity provides a speedup of 13× over the unfused baseline.

Inference prefill, training, and sparseGPT benchmarks achieve speedups in the 1.5× to 3× range. With spatial fusion, higher operation intensity is achieved with sufficient coarse-grained pipeline parallelism to keep the on-chip units occupied.

In spite of having low operational intensity, autoregressive decoding inference benchmarks gain from fusion by eliminating the overheads of kernel launch and unnecessary HBM traffic. We observe speedups from 1× to 13×, with Mistral achieving the highest speedup.

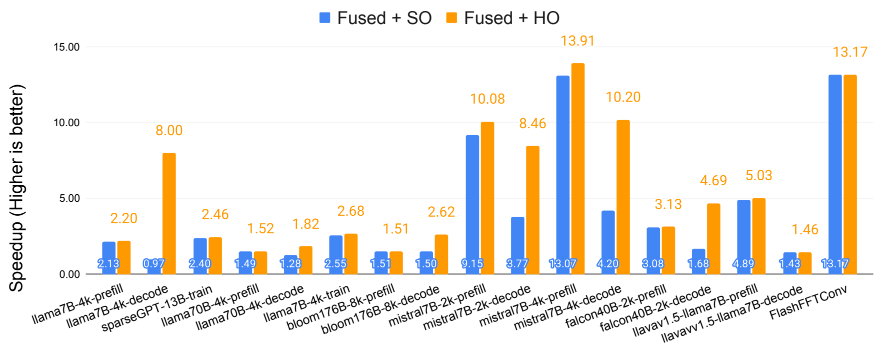


FIGURE 6: Measured benchmark speedups over an unfused baseline running on 8 SN40L sockets. SO = Software-Orchestrated, HO = Hardware-Orchestrated.

| **Metric** | **vs. DGX A100** | **vs. DGX H100** |
|---|---|---|
| Overall Speedup, BS = 8, 20 output tokens | 6.6× | 3.7× |
| Overall Speedup, BS = 1, 20 output tokens | 4.8× | 2.8× |
| Expert Speedup, BS=1, 20 output tokens | 2.0× | 1.5× |
| Overall Speedup, BS = 8, 200 output tokens | 4.2× | 2.7× |
| Overall Speedup, BS = 1, 200 output tokens | 3.9× | 2.6× |
| Expert Speedup, BS=1, 200 output tokens | 3.2× | 2.3× |
| Model Switching Time | 31× | 15× |
| > 150 Experts | DGX OOM | DGX OOM |

TABLE 2: Samba-CoE Performance Comparison between SN40L Node, DGX A100, and DGX H100.

**Hardware-Orchestrated Kernel Launch Speedup:** Autoregressive decoding inference benchmarks achieve a noticeable speedup of 1.4× to 8×. Kernels have very short execution times in these benchmarks, making kernel launch overheads significant. Offloading kernel scheduling to the SN40L cuts out the overheads of software scheduling to provide a speedup.

## Composition of Experts

Table 2 summarizes the results vs. DGX A100 and DGX H100. We study two scenarios with increasing expert counts: **Latency Impact** on a single node, and **System Footprint Impact** to sustain the same TP8 latency.

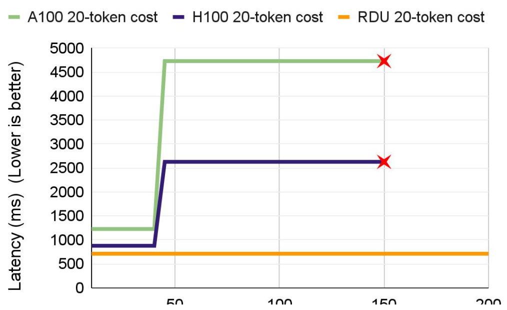
**Latency Impact:** We model two use cases: a chatbot conversation use case to produce 20 output tokens per input prompt, and a translation use case to produce 200 output tokens [10] The models and router are mapped as tensor-parallel over eight sockets (TP8) fashion on all platforms. The router and KV-cache is always in HBM. The SN40L Node numbers are measured on real hardware. As Samba-CoE is not deployed on DGX, we estimate latencies using published model latency numbers [10] and optimistic model switching estimates based on DGX specs [11], [12]. The total latency includes the time to run the router, copy the required expert weights, and running the expert.

We report latencies for batch size (BS) = 1 and BS=8 cases. Note that "batch" applies to the Samba-CoE model as a whole, and not to individual experts.

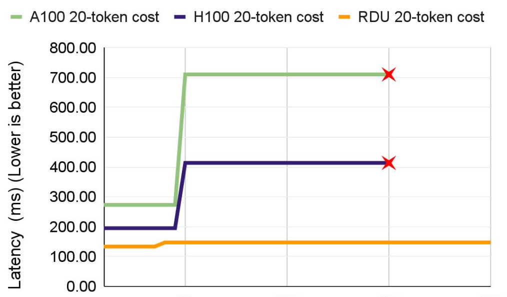
Figure 7 compares latencies across the three platforms. We analyze these results in two broad categories:

**Under 50 experts:** The SN40L Node is 2× faster than the DGX A100 and 1.5× faster than the DGX H100 to generate 20 tokens. For 200 tokens, the speedup numbers are 3.2× and 2.3×, respectively. Note that generative inference is memory-bound, and the SN40L Node has comparable HBM bandwidth to A100 (and lower than H100). The speedups clearly demonstrate the benefits of streaming dataflow: the *entire decoder* layer is fused into a single kernel call, using almost 90% of the PCUs and PMUs, and saturating close to 85% of HBM bandwidth. Furthermore, as the model mostly contains multiple identical decoder layers, SN40L sees virtually zero kernel launch overheads.

**Over 50 experts:** Latency spikes on DGXs (around 50 7B experts) happens when experts spill over to host DRAM. For BS=8, the SN40L Node achieves a speedup of 6.6× and 3.7× over DGX A100 and DGX H100, respectively. For BS=1, the SN40L Node



(a) BS=8, TP=8 Latency.



(b) BS=1, TP=8 Latency.

FIGURE 7: Samba-CoE latency comparison to generate 20 tokens with batch size=1 on the SN40L Node, DGX A100, and DGX H100.

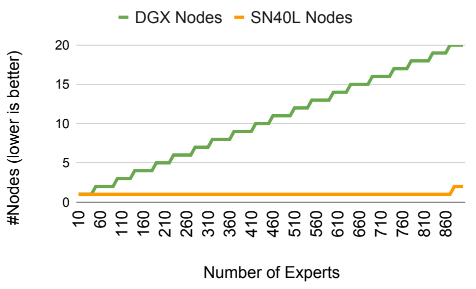


FIGURE 8: System footprint to sustain TP8 performance with increasing expert counts.

achieves speedups of 4.8× and 2.8× respectively. BS=8 requires copying a larger number of experts, and hence accounts for a larger fraction of the total time. The copy time on the SN40L Node is 31× faster than DGX A100 (which provides 32 GB/s host-to-GPU bandwidth) , and 16× faster than H100 (which provides 64 GB/s host-to-GPU bandwidth). DGXs run out of memory at 150 experts.

**System Footprint Impact:**

Figure 8 quantifies the system cost of increasing experts with the same performance on both platforms. A single SN40L Node can hold and serve a CoE of up to 850 experts at the TP8 latency. Achieving this with DGX would need 19 DGX nodes to hold all experts in HBM.

## ACKNOWLEDGMENTS

The author(s) would like to thank Apurv Vivek, Arjun Sabnis, David Koeplinger, David Jackson, Dawei Huang, Denis Sokolov, Jiayu Bai, Manish Shah, Mark Luttrell, Nathan Sheeley, Norris Liu, Sumti Jairath, and Yongning Sheng for their contributions to this paper.

## REFERENCES

1. N. Maslej, L. Fattorini, R. Perrault, et al, “The AI Index 2024 Annual Report,” Stanford University, 2024. [Online]. Available: https://aiindex.stanford.edu/wp-content/uploads/2024/04/HAI_AI-Index-Report-2024.pdf (URL)
2. C. Raffel, “Build An Ecosystem, Not A Monolith” [Online]. Available: https://colinraffel.com/talks/simons2023build.pdf (URL)
3. H. W. Chung, L. Hou, S. Longpre et al, “Scaling instruction-finetuned language models,” 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2210.11416 (URL)
4. D. Biderman, J. G. Ortiz, J. Portes et al, “LoRA Learns Less and Forgets Less,” 2024. [Online]. Available: https://arxiv.org/abs/2405.09673 (URL)
5. S. Gururangan, M. Li, M. Lewis et al, “Scaling expert language models with unsupervised domain discovery,” 2023. [Online]. Available: https://arxiv.org/abs/2303.14177 (URL)
6. M. Zaharia, O. Khattab, L. Chen et al, “The shift from models to compound ai systems,” 2024. [Online]. Available: https://bair.berkeley.edu/blog/2024/02/18/compound-ai-systems/ (URL)
7. SambaNova Systems, "Benchmarking Samba-1," 2024 https://sambanova.ai/blog/benchmarking-samba-1 (URL)
8. D. Y. Fu, H. Kumbong, E. Nguyen et al, “Flashfftconv: Efficient convolutions for long sequences with tensor cores,” 2023 https://arxiv.org/abs/2311.05908 (URL)
9. D. Zhang, S. Huda, E. Songhori et al, “A full-stack search technique for domain optimized deep learning accelerators,” ASPLOS 2022 https://arxiv.org/abs/2105.12842 (URL)
10. NVIDIA, “Llama-2 results, Nvidia nemo,” 2024 https://docs.nvidia.com/nemo-framework/user-guide/latest/performance/llama.html (URL)
11. NVIDIA, “NVIDIA DGX A100 datasheet,”

https://resources.nvidia.com/en-us-dgx-systems/dgx-ai (URL)

12. NVIDIA, "NVIDIA DGX A100 datasheet," https://resources.nvidia.com/en-us-dgx-systems/ai-enterprise-dgx (URL)