Breakthrough low-latency, high-energy-efficiency LLM inference performance using NorthPole

Rathinakumar Appuswamy†, Michael V. Debole†, Brian Taba, Steven K. Esser, Andrew S. Cassidy, Arnon Amir, Alexander Andreopoulos, Deepika Bablani, Pallab Datta, Jeffrey A. Kusnitz, Nathaniel J. McClatchey, Neil McGlohon, Jeffrey L. McKinstry, Tapan K. Nayak, Daniel F. Smith, Rafael Sousa, Ignacio Terrizzano, Filipp Akopyan, Peter J. Carlson, Rajamohan Gandhasri, Guillaume J. Garreau, Nelson M. Gonzalez, Megumi Ito, Jennifer L. Klamo, Yutaka Nakamura, Carlos Ortega Otero, William P. Risk, Jun Sawada, Kai Schleupen, Jay Sivagnaname, Matthew Stallone, Takanori Ueda, Myron D. Flickner, John V. Arthur, Rameswar Panda, David D. Cox, Dharmendra S. Modha*
IBM Research, *dmodha@us.ibm.com, †Contributed equally.

***Abstract*—For a 3-billion-parameter LLM, a research prototype inference appliance with 16 IBM AIU NorthPole processors delivers a massive 28,356 tokens/second of system throughput and sub-1 ms/token (per-user) latency while consuming merely 672 W for 16 NorthPole cards in a compact 2U form factor.**

With a focus on low latency and high energy efficiency, when NorthPole (in 12 nm) is compared to a suite of GPUs (in 7/5/4 nm) at various power consumptions, at the lowest GPU latency, NorthPole provides 72.7× better energy metric (tokens/second/W) while providing better latency.

***Index Terms*—AI accelerators, large language model**

I. Introduction

Large language models (LLMs) [1] have achieved significant performance benchmarks across diverse AI tasks [2], such as assisting in programming by offering code suggestions [3], [4], excelling in standardized tests [5], and aiding in content creation of articles, blogs, images, and videos [6].

There are two major and conflicting challenges emerging in wide-scale deployment of LLMs in particular, and AI in general, namely: energy consumption and latency of response.

First, because LLMs demand substantial energy resources for both training and inference [7], [8], a sustainable future computational infrastructure is needed to enable their efficient and widespread deployment. Energy efficiency of data centers is becoming critical as their carbon footprints expand, and as they become increasingly energy-constrained. According to the World Economic Forum,

> "At present, the environmental footprint is split, with training responsible for about 20% and inference taking up the lion's share at 80%. As AI models gain traction across diverse sectors, the need for inference and its environmental footprint will escalate." [9]

Second, many applications such as interactive dialog and agentic workflows require very low latencies. Decreasing latency, within a given computer architecture, can be achieved by decreasing throughput, however, that leads to decreasing energy efficiency. To paraphrase a classic systems maxim,

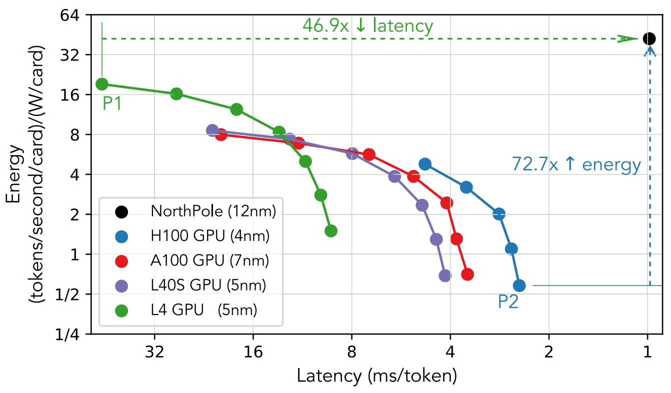
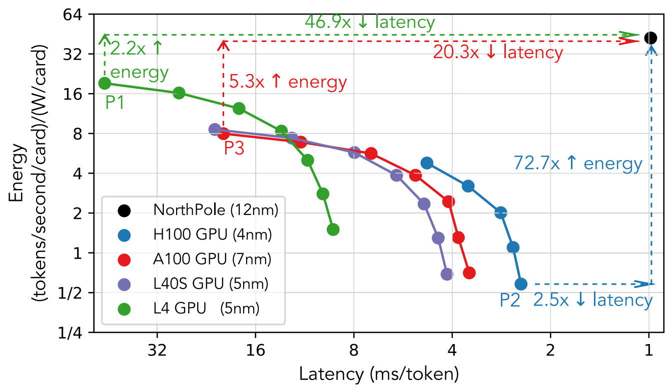


Fig. 1: NorthPole (12 nm) performance relative to current state-of-the-art GPUs (7/5/4 nm) on energy and system latency metrics, where system latency is the total latency experienced by each user. At the lowest GPU latency (H100, point P2), NorthPole provides 72.7× better energy metric (tokens/second/W). At the best GPU energy metric (L4, point P1), NorthPole provides 46.9× lower latency.

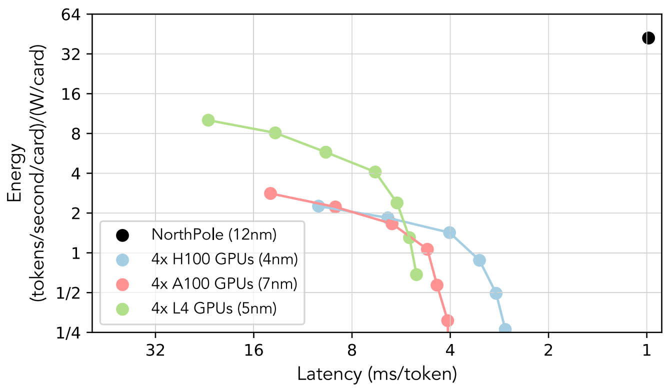
> "Throughput problems can be cured with money. Latency problems are harder because the speed of light is fixed." (Paraphrased from [10] by replacing "bandwidth" with "throughput.")

GPUs can achieve lower latency by using smaller batch size, at the expense of decreased throughput and decreased energy efficiency. Also, GPU sharding, which uses data parallelism across multiple GPUs, may reduce latency but only at the cost of energy efficiency. Further, with or without sharding, GPUs seem to hit a hard wall of latency lower bound. The trade-off between energy efficiency and latency for GPUs can be seen in Fig 1.

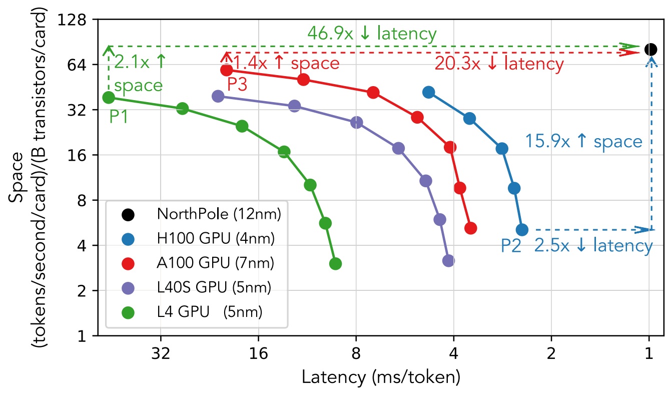
Therefore, a critical research question addressed in this paper is to simultaneously achieve the dual and conflicting objectives of low latency at high energy efficiency.



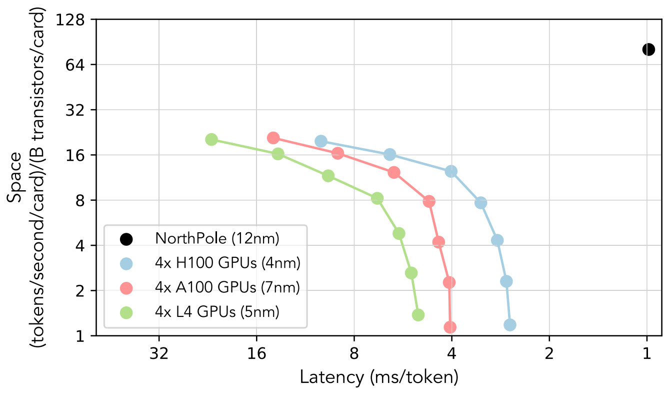
(a) 1 GPU card



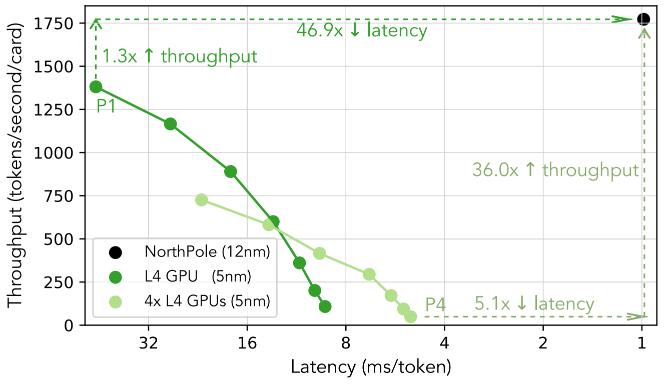
(b) 4 GPU cards



(c) 1 GPU card



(d) 4 GPU cards



(e) L4 and NorthPole

Fig. 2: Panels (a)–(d) show NorthPole (12 nm) performance relative to current state-of-the-art GPUs (7 / 5 / 4 nm) on energy, space, and system latency metrics, where system latency is the total latency experienced by each user. Panel (a) is the same as Fig. 1 with notations added for point P3. Panels (a) and (c) use a single GPU and panels (b) and (d) use sharding, which may decrease latency but only at the cost of energy and space efficiencies. At the lowest GPU latency (H100, point P2), NorthPole provides 72.7× better energy metric (tokens / second / W) and 15.9× better space metric (tokens / second / transistor) while still providing 2.5× lower latency. At the best GPU energy metric (L4, point P1), NorthPole provides 46.9× lower latency and 2.1× better space metric while still providing 2.2× better energy metric. At the best GPU space metric (A100, point P3), NorthPole provides 20.3× lower latency and 5.3× better energy metric while still providing 1.4× better space metric. Panel (e) shows NorthPole (12 nm) performance relative to the L4 GPU (5 nm) on throughput (tokens / second / card) and system latency metrics. At the lowest L4 latency (point P4), NorthPole provides 36.0× higher throughput. At the highest L4 throughput with a sub-50 ms / token latency (point P1), NorthPole provides 46.9× lower latency. GPU power used to compute each energy metric is shown in Table I. Because there is no instrumentation available to measure the actual power for different batch sizes, the same power is used for all batch sizes and this may underestimate the energy metric but the qualitative picture still holds.

II. Results

NorthPole is an inference accelerator chip and software ecosystem co-designed from first principles to offer exceptional efficiency for neural network inference [11], [12]. Despite NorthPole not being specifically designed for LLMs, surprisingly, this paper demonstrates that the novel NorthPole architecture enables low-latency, high-energy-efficiency LLM inference (Fig. 1, Fig. 2, and Table I).

The main results of this paper are as follows.

For a 3-billion-parameter LLM whose model structure is derived from IBM Granite-8B-Code-Base model [4] and is consistent with Llama 3 8B [13] as well as Mistral 7B [14], the paper demonstrates a research prototype inference appliance with 16 NorthPole processors.

In absolute terms, the appliance delivers $28,356$ tokens/sec of system throughput and sub-1 ms / token (per-user) latency while consuming 672 W for 16 NorthPole cards in a 2U form factor.

In relative terms, when comparing NorthPole (in 12 nm) to a suite of GPUs (A100 / L4 / L40S / H100, respectively, in 7 / 5 / 5 / 4 nm) at various power consumptions, it can be seen from Fig. 2(a) and Fig. 2(c) that at the lowest GPU latency (point P2), NorthPole provides $72.7\times$ better energy metric (tokens / second / W) and $15.9\times$ better space metric (tokens / second / transistor) while still providing $2.5\times$ lower latency; at the best GPU energy metric (point P1), NorthPole provides $46.9\times$ lower latency and $2.1\times$ better space metric while still providing $2.2\times$ better energy metric; and at the best GPU space metric (point P3), NorthPole provides $20.3\times$ lower latency and $5.3\times$ better energy metric while still providing $1.4\times$ better space metric.

In particular, when comparing NorthPole (in 12 nm) to L4 GPU (in 5 nm) with comparable power, it can be seen from Fig. 2(e) that at the highest (sub-50 ms / token) L4 throughput (point P1), NorthPole provides $46.9\times$ lower latency while still providing $1.3\times$ higher throughput and, at the lowest L4 latency (point P4), NorthPole provides $36.0\times$ higher throughput in tokens / second / card while still providing $5.1\times$ lower latency.

III. NorthPole Architecture

The NorthPole processor, shown in Fig. 3, is fabricated in a 12 nm process technology, with 22 billion transistors in a 795 mm^2 area. Inspired by the brain and optimized for silicon, its architecture is derived from ten complementary design axioms spanning computation, memory, communication, and control that collectively enable NorthPole to vastly outperform other architectures on standard AI inference tasks, even compared to processors manufactured in more advanced process technologies [11].

For details of the axiomatic NorthPole architecture, see [11], [12]. In brief, NorthPole tiles 256 modular cores in a 16×16 two-dimensional array. Each core contains a vector-matrix multiplier (VMM) that performs $2,048$, $4,096$, and $8,192$ operations per core per cycle at INT8, INT4 and INT2 precision, respectively. The core compute also includes a 4-way, 32-slice, FP16 vector unit and a 32-slice activation

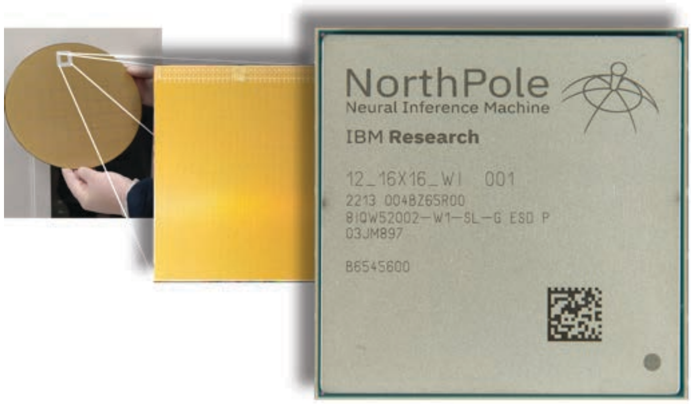


Fig. 3: NorthPole processor: Silicon wafer (left), bare die (center), packaged module (right).

function unit. The core array has a total of 192 MB of SRAM, with 0.75 MB in each core. On-chip memory is tightly coupled to the compute units and control logic, with an aggregate bandwidth of 13 TB / s between core memory and compute. In addition, $4,096$ wires cross each core in both horizontal and vertical directions to communicate parameters, instructions, activations, and partial sums over four dedicated networks-on-chip (NoCs). To prevent stalls, an on-chip frame buffer with 32 MB of SRAM decouples off-chip communication of input and output data from on-chip computation by the core array.

IV. Appliance

NorthPole has been prototyped in a PCIe Gen3 $\times$ 8 card, shown in Fig. 4, of which 16 were installed in an off-the-shelf 2U server to assemble a research prototype inference appliance, shown in Fig. 5. The server comprises $2\times$ Intel Xeon Gold 6438M processors, each with 32 cores and 60 MB cache, running at 2.2 GHz. The system also contains 512 GB of 4800 MHz DDR5 memory. Two PCIe Gen5 $\times$ 16 buses are attached to each of the two server processors, providing a total of 256 GB / s of PCIe bandwidth across the four buses (per direction). These four buses each fan out $4\times$ via PCIe bridges, to the system's 16 PCIe slots, each populated with a NorthPole card. These 16 NorthPole cards use a maximum of $1/2$ the aggregated 256 GB / s PCIe bandwidth available. The system runs Red Hat Enterprise 8.9 and NorthPole uses the built-in VFIO kernel driver so that user-space software can manage the hardware. The system uses the IOMMU to manage address translation, as well as to enable security features such as device isolation and virtualization to run applications using virtual machines or container technologies.

Each NorthPole card receives and transmits data via DMA engines resident on each card. Operating independently, these DMA engines can simultaneously receive and transmit tensors in multiple ways. The first method is a standard PCIe endpoint model, where the host programs the DMA engines to read inputs from host memory and write tensors back to host memory after computation is completed. The second method uses additional hardware features on each card to let

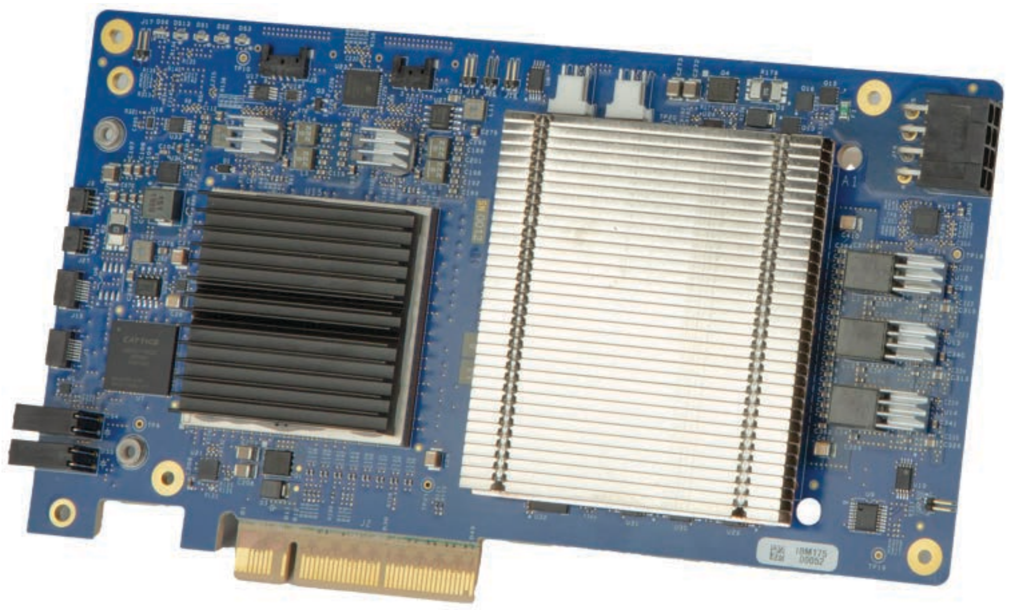


Fig. 4: NorthPole PCIe card.

NorthPole cards communicate directly with one another, via PCIe, without requiring transfers to and from host memory or additional software management at runtime. Using direct NorthPole-to-NorthPole communication enables larger models to span multiple NorthPole chips while reducing communication latencies and overheads from a purely software-managed system.

V. Mapping LLMs to a NorthPole Appliance

The strategy for mapping LLMs, shown in Fig. 6, was inspired by three key observations. First, for models that are large enough to be useful, an entire transformer layer can fit in the memory of a single NorthPole chip using INT4 for weights, activations, and KV cache ("w4a4"), and the output layer can fit in 2 chips. Second, if weights and KV cache reside entirely on-chip, only the small embedding tensor needs to be communicated between layers at run time, which is well within the bandwidth of PCIe Gen3×8. Third, a prototype NorthPole Appliance can readily be assembled by populating an off-the-shelf server with 16 NorthPole PCIe cards.

This suggests a strategy that maps each transformer layer onto its own NorthPole card, using pipeline parallelism in the style of GPipe [15], and splits the output layer across 2 NorthPole cards, using tensor parallelism [16], [17], with layers sending embedding tensors to each other via PCIe Gen3×8. During inference, a mini-batch of, say, N user requests is divided into, say, M equal micro-batches and the micro-batches are pipelined through the 16 NorthPole cards.

While pipeline parallelism has been exploited for training LLMs (with no latency constraints), its use during inference is limited by the large mini-batch required to minimize the idle time per pipeline stage, or pipeline bubbles. For example, it was found [15] that efficient training required the number of micro-batches M to be about four times the number of pipeline stages. The mini-batch size N is limited by (a) the desired per-token latency of the system, and (b) the memory available to store the KV cache for the entire mini-batch. The low-latency compute and 13 TB / sec of on-chip memory bandwidth enable NorthPole to achieve extremely low per-token latency, so the the limiting factor in choosing N is the memory available to store the entire KV cache on-chip. In addition, we found that a number of micro-batches M equal to the number of pipeline stages was sufficient to keep the pipeline idle time negligible.

For the experiments reported in this paper, we chose a mini-batch size of $N = 28$, divided into $M = 14$ equal micro-batches, resulting in a micro-batch of size 2 being computed by each NorthPole card. Our architectural design choices to perform efficient computations at such a small batch size were key to enabling the efficiencies demonstrated in Fig. 1 and Table I.

VI. The LLM model and the training approach

A. The LLM model

The model employed for testing our system was based on the open-source IBM Granite-8B-Code-Base model [4], an 8-billion-parameter transformer decoder with 36 transformer layers, 4096 hidden size, $14,336$ FFN intermediate size, 32 attention heads, 8 key-value heads using grouped-query attention (GQA), and $49,152$ vocabulary size. To fit into a single server with 16 NorthPole cards, a 3-billion-parameter version of this model was used with 14 transformer layers and one output layer, quantized to w4a4 precision but otherwise preserving the original architecture.

Notably, this choice of model configuration matches that of Llama 3 8B [13] as well as Mistral 7B [14] on a per-layer

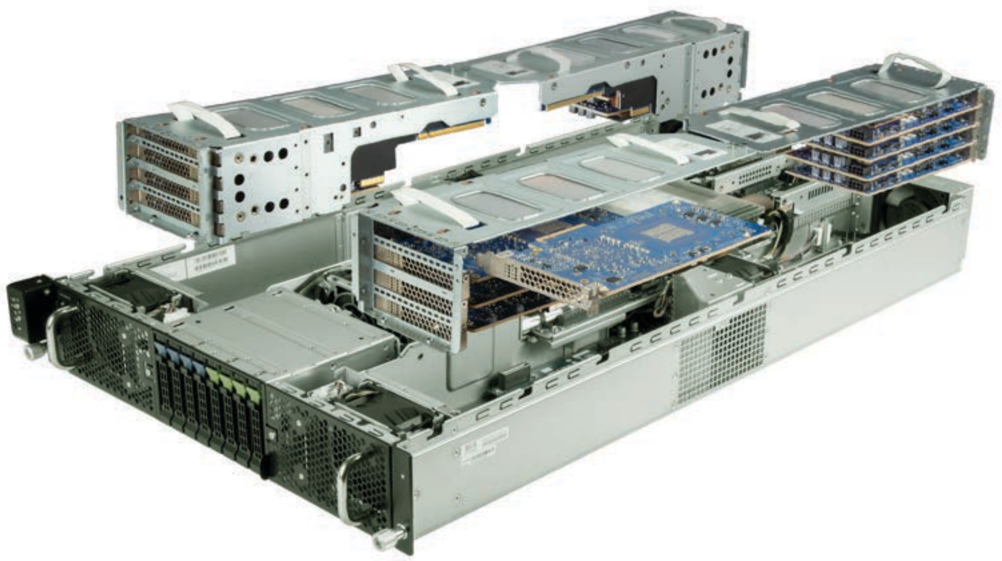


Fig. 5: Exploded view of the research prototype appliance showing installation of the 16 NorthPole PCIe cards. NorthPole cards can communicate via the standard PCIe endpoint model through the host or directly, and more efficiently, with one another via additional hardware features on each card.

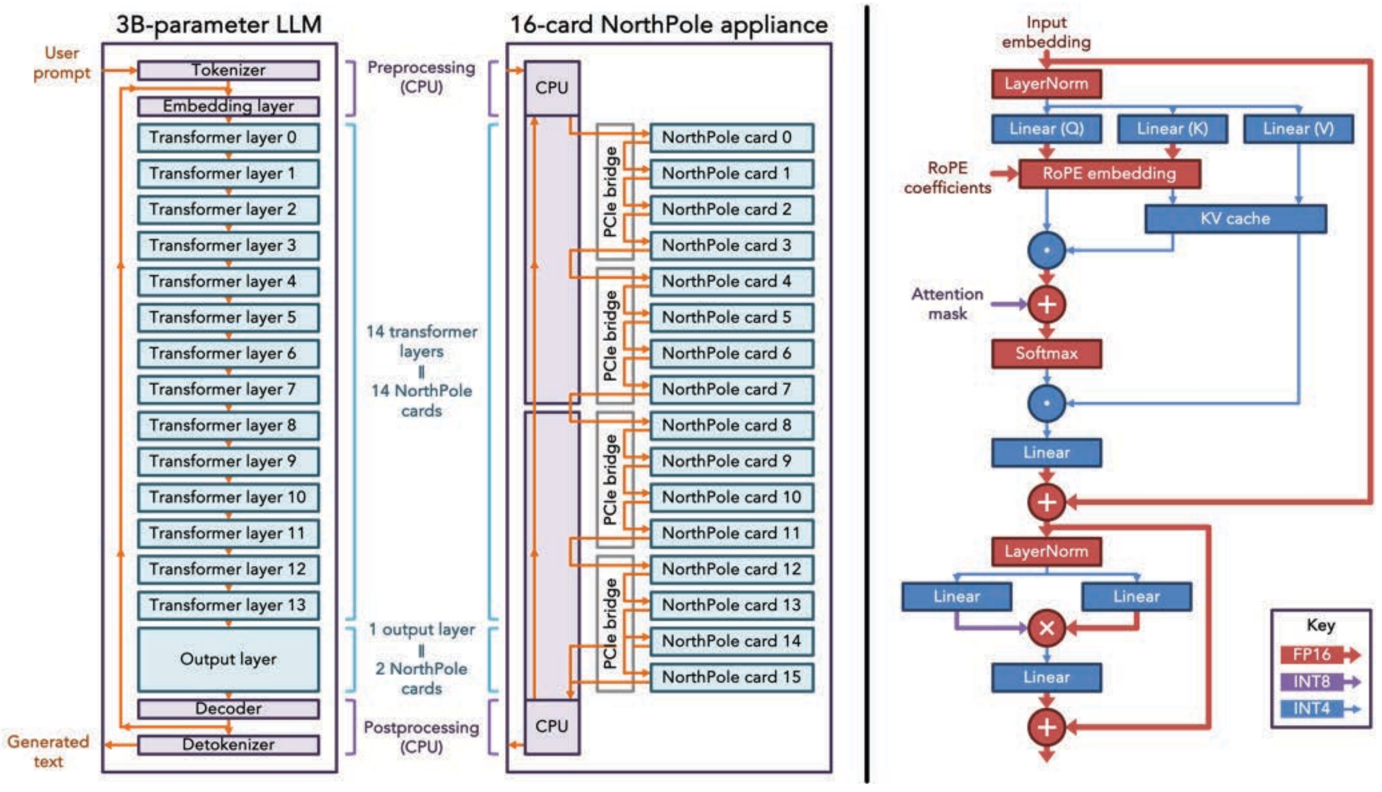


Fig. 6: Strategy for mapping the 3-billion-parameter LLM to the 16-card NorthPole appliance. Each transformer layer is mapped to one NorthPole card and the output layer is mapped to two cards (left). For each layer, all weights and KV cache are stored on-chip, so only the small embedding tensor produced by each card's layer must be forwarded to the next card over low-bandwidth PCIe when generating a token. Within each transformer layer (right), weights and KV cache are stored at INT4 precision. Activations are also INT4 except when higher dynamic range is needed for accumulations.

basis, which only differ from the model we benchmark in layer count, model vocabulary size, and the training data used.

B. Training for Full-precision Accuracy

To recover the task accuracy of the original model after quantization, the following procedure was used to create model weights. First, a baseline model was trained from scratch at full FP16 precision on 1 trillion tokens of code from 116 languages, following the recipe of [4]. Next, the baseline model output layer weights and inputs, and the SiLU activations were quantized to INT8, and all other weights, linear layer inputs, and matmul inputs were quantized to INT4. Finally, post-quantization accuracy was recovered using quantization-aware training for a further 8.5 billion tokens from the Python-language subset of the training data, with learning rate 8×10^{-5} and batch size 128, employing the LSQ algorithm [18]. The activation quantizer step sizes were trained using a hot start that boosted their learning rate $200\times$ for the first 250 steps of training to help quickly adapt to the data.

The baseline FP16 model running on GPU and quantized model running on NorthPole showed pass@10 accuracy on HumanEvalSynthesize-Python within 0.01 of one another (0.3001 GPU vs. 0.2922 NorthPole) [19]. Rather than push the bounds of task accuracy, overall training was abbreviated compared to the Granite-8B-Code-Base model to focus on hardware performance characterization.

VII. Runtime Application

During inference, as shown in Fig. 6, tokens are generated by a highly pipelined user application running on the host CPUs that preprocesses text into input tensors by using the tokenizer and embedding layer, puts input tensors into the first NorthPole card in the appliance, receives the resulting output tensors from the last NorthPole cards in the appliance, post-processes output tensors using the decoder and detokenizer, and recirculates the generated token as the next input. The user application is also responsible for the user interface and any higher-level optimizations like prompt prefilling.

To offload neural network workloads to NorthPole, the user application calls a user-space runtime library with a simple API to configure the NorthPole cards with layer weights and KV caches at initialization time, and send and receive input and output tensors at run time. Weights and KV cache stay resident in on-chip memory after configuration, and do not need to be streamed from off-chip at run time. The runtime library also manages the on-chip frame buffer to prevent the NorthPole cores from being stalled by lack of input data or a recipient for output data. Intermediate tensors are passed from card to card without host intervention, as described in Section IV.

VIII. Performance Results

The NorthPole 16-card appliance achieved throughput of 28,356 tokens / sec on the 3-billion-parameter LLM. The LLM was configured for a sequence length of 2048 (1024 prompt length, 1024 tokens generated) and the decoder used greedy sampling.

To compare against GPUs, we measured the single-card performance of two GPUs targeting low-power inference (L4 [20] and L40S [20]) and two GPUs targeting high-throughput training (A100 [21] and H100 [22]). All systems ran the same LLM model and configuration, except that NorthPole operated at w4a4 precision and the GPUs at the best possible precision of w4a16 since, to the best of our knowledge, no w4a4 CUDA kernels were available. In our GPU experiments, we utilized GPTQ quantized models and tested using vLLM (version 0.5.4) Marlin kernels for benchmarking against NorthPole. The use of GPTQ quantization provided the best available model inference performance for GPUs by reducing the precision of weights while maintaining acceptable accuracy. Additionally, the Marlin kernels were employed to optimize matrix operations, particularly in handling sparse and dense matrix multiplications in GPUs. Benchmarking with the vLLM runtime enabled us to assess both throughput and latency, ensuring optimal model performance on the given hardware configuration. For experiments with more than one GPU card, Tensor Parallelism equal to the available number of cards was used to effectively obtain the minimal possible latency over NVLink. From our experiments, sharding decreases latency but leads to degraded throughput / card for GPUs. Importantly, note that NorthPole's outstanding performance is primarily a result of massive on-chip memory bandwidth and only secondarily a result of lower precision.

Table I shows the measured performance results for the NorthPole and the GPU systems on a per-card basis. The fundamental metrics are Throughput, Latency, Space, and Energy metrics, which are defined as follows [11].

The total number of tokens generated in response to a mini-batch of input prompts is:

$$tokens\_gen = micro\_batch\_size \times M \times tok\_seq\_len \quad (1)$$

where M is the number of micro-batches and tok_seq_len is the number of output tokens generated for a single user. The system throughput is the total number of tokens generated ($tokens\_gen$) in response to a mini-batch of input prompts divided by the total time to process the prompts, including both prompt prefill time ($prompt\_time$) and token generation time ($token\_gen\_time$):

$$sys\_throughput = \frac{tokens\_gen}{prompt\_time + token\_gen\_time} \quad (2)$$

Throughput is compared on a per-card basis by dividing the system throughput by the number of processing cards in the system:

$$throughput = \frac{sys\_throughput}{sys\_cards} = (tok/sec/card) \quad (3)$$

The latency is a measure of average time between generated output tokens for a particular user, and it is the sum of the time it takes for an embedding token to flow through the processing

TABLE I: Measured Performance Results

Measured performance of NorthPole and GPU systems, on a per-card basis. For each metric, ↓ indicates lower is better, while ↑ indicates higher is better. For the NorthPole 16-card appliance, power was measured per card, while total system throughput was divided by 16 cards. NorthPole latency is measured through all 16 cards. P1, P2, P3, P4, referencing the labeled points in Fig. 1 and Fig. 2, denote the highest GPU energy metric, the lowest overall GPU latency, the highest GPU space metric, and the lowest energy-efficient GPU latency, respectively.

Processor	**Transistors ↓** (billions / card)	**Power ↓** (W / card)	**Cards**	**Mini-batch**	**Throughput ↑** (tok / sec / card)	**Latency ↓** (ms / tok)	**Space Metric ↑** (tok / sec / card) / (billion transistors / card)	**Energy Metric ↑** (tok / sec / card) / (W / card)	
L4 GPU (5 nm) [20]	35.8	72	1×	1	108	9.26	3.02	1.50	-
			1×	64	1381	46.34	38.58	19.18	P1
			2×	1	76	6.58	2.12	1.06	-
			2×	64	1092	29.32	30.49	15.16	-
			4×	1	49	5.08	1.38	0.68	P4
			4×	64	726	22.05	20.27	10.08	-
L40S GPU (5 nm) [20]	76.3	350	1×	1	241	4.15	3.16	0.69	-
			1×	64	3002	21.32	39.34	8.58	-
			2×	1	128	3.89	1.68	0.37	-
			2×	64	1754	18.25	22.98	5.01	-
A100 GPU (7 nm) [21]	54.2	400	1×	1	282	3.55	5.20	0.70	-
			1×	64	3190	20.06	58.86	7.98	P3
			2×	1	132	3.79	2.44	0.33	-
			2×	64	1858	17.23	34.27	4.64	-
			4×	1	62	4.05	1.14	0.15	-
			4×	64	1126	14.22	20.77	2.81	-
H100 GPU (4 nm) [22]	80.0	700	1×	1	406	2.46	5.08	0.58	P2
			1×	16	3347	4.78	41.84	4.78	-
			2×	1	172	2.90	2.16	0.25	-
			2×	64	2204	14.52	27.55	3.15	-
			4×	1	94	2.65	1.18	0.14	-
			4×	64	1580	10.13	19.74	2.26	-
NorthPole (12 nm)	22.0	42	16×	28	1772	0.988	80.55	42.19	-

pipeline and the prompt prefill time amortized over the total number of generated tokens:

$$latency = \frac{prompt\_time + token\_gen\_time}{tok\_seq\_len} = (sec) \quad (4)$$

Equivalently, by combining equations 1, 2, and 4:

$$latency = \frac{mini\_batch\_size}{sys\_throughput} = (sec) \quad (5)$$

where $mini\_batch\_size = micro\_batch\_size \times M$. Note that this is the system latency that is seen by each user.

Normalizing by the number of cards in a system, we extend the space and energy metrics defined in [11] to be able to compare systems with varying number of cards. The resulting space and energy metrics are the throughput per card normalized by the number of processor transistors per card and the power per card, respectively:

$$\begin{matrix} space \\ metric \end{matrix} = \frac{sys\_throughput/card}{transistors/card} = \frac{(tok/sec)}{(transistor)} \quad (6)$$

$$\begin{matrix} energy \\ metric \end{matrix} = \frac{sys\_throughput/card}{power/card} = \frac{(tok/sec)}{(W)} \quad (7)$$

If the system throughput scales proportionally to the number of pipelined cards in the system, the normalization by cards cancels out, rendering the space and energy metrics invariant to the number of cards in the system. In general, the system throughput scales sub-linearly in the number of cards due to communication and synchronization overhead.

IX. Conclusions

We present the following contributions:

- We have demonstrated a research prototype multicard NorthPole appliance.
- We have demonstrated that large neural network models like LLMs can be effectively split across multiple NorthPole processors, extending our earlier work that showed single NorthPole processors outperforming other architectures on vision inference tasks (ResNet50, Yolo-v4) [11], [12].
- We have demonstrated that NorthPole's unique architecture is well-suited for LLM inference, leading NorthPole to dramatically outperform edge and datacenter GPUs in terms of dual objectives of low-latency and high-energy efficiency.

Because the NorthPole appliance must be used as a whole, it is most efficient for high-throughput applications.

This initial paper provides a springboard for research into further energy-efficiency optimizations, into mapping larger LLMs on correspondingly larger NorthPole appliances, into new LLM models co-optimized with NorthPole architecture, and into future system and chip architectures.

Acknowledgment

This work builds on previously published research [11] that was supported by the United States Air Force under Contract No. FA8750-19-C-1518. The authors are grateful to Sam Skalicky for technical contributions.

References

[1] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[2] J. Yang, H. Jin, R. Tang, X. Han, Q. Feng, H. Jiang, B. Yin, and X. Hu, "Harnessing the power of LLMs in practice: A survey on ChatGPT and beyond," 2023. [Online]. Available: https://arxiv.org/abs/2304.13712

[3] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba, "Evaluating large language models trained on code," 2021. [Online]. Available: https://arxiv.org/abs/2107.03374

[4] M. Mishra, M. Stallone, G. Zhang, Y. Shen, A. Prasad, A. M. Soria, M. Merler, P. Selvam, S. Surendran, S. Singh *et al.*, "Granite code models: A family of open foundation models for code intelligence," *arXiv preprint arXiv:2405.04324*, 2024.

[5] O. (2023), "GPT-4 technical report," 2024. [Online]. Available: https://arxiv.org/abs/2303.08774

[6] D. McCandless, T. Evans, and P. Barton. (2024) The rise and rise of A.I. large language models (LLMs) & their associated bots like ChatGPT. [Online]. Available: https://informationisbeautiful.net/visualizations/the-rise-of-generative-ai-large-language-models-llms-like-chatgpt/

[7] B. Cottier, R. Rahman, L. Fattorini, N. Maslej, and D. Owen, "The rising costs of training frontier AI models," *arXiv preprint arXiv:2405.21015v1*, 2024.

[8] S. Samsi, D. Zhao, J. McDonald, B. Li, A. Michaleas, M. Jones, W. Bergeron, J. Kepner, D. Tiwari, and V. Gadepally, "From words to watts: Benchmarking the energy costs of large language model inference," 2023. [Online]. Available: https://arxiv.org/abs/2310.03003

[9] B. Ammanath, "How to manage AI's energy demand — today, tomorrow and in the future," 2024. [Online]. Available: https://www.weforum.org/agenda/2024/04/how-to-manage-ais-energy-demand-today-tomorrow-and-in-the-future/

[10] D. A. Patterson, "Latency lags bandwidth," *Commun. ACM*, vol. 47, no. 10, p. 71–75, Oct 2004. [Online]. Available: https://doi.org/10.1145/1022594.1022596

[11] D. S. Modha, F. Akopyan, A. Andreopoulos, R. Appuswamy, J. V. Arthur, A. S. Cassidy, P. Datta, M. V. DeBole, S. K. Esser, C. O. Otero *et al.*, "Neural inference at the frontier of energy, space, and time," *Science*, vol. 382, no. 6668, pp. 329–335, 2023.

[12] A. S. Cassidy, J. V. Arthur, F. Akopyan, A. Andreopoulos, R. Appuswamy, P. Datta, M. V. Debole, S. K. Esser, C. O. Otero, J. Sawada *et al.*, "11.4 IBM NorthPole: An Architecture for Neural Network Inference with a 12nm Chip," in *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 67. IEEE, 2024, pp. 214–215.

[13] AI@Meta, "Llama 3 model card," 2024. [Online]. Available: https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md

[14] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mistral 7B," 2023. [Online]. Available: https://arxiv.org/abs/2310.06825

[15] Y. Huang, Y. Cheng, A. Bapna, O. Firat, M. X. Chen, D. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu, and Z. Chen, "GPipe: Efficient training of giant neural networks using pipeline parallelism," 2019. [Online]. Available: https://arxiv.org/abs/1811.06965

[16] N. Shazeer, Y. Cheng, N. Parmar, D. Tran, A. Vaswani, P. Koanantakool, P. Hawkins, H. Lee, M. Hong, C. Young, R. Sepassi, and B. Hechtman, "Mesh-TensorFlow: Deep learning for supercomputers," 2018. [Online]. Available: https://arxiv.org/abs/1811.02084

[17] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-LM: Training multi-billion parameter language models using model parallelism," 2020. [Online]. Available: https://arxiv.org/abs/1909.08053

[18] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, "Learned step size quantization," in *International Conference on Learning Representations*, 2020.

[19] N. Muennighoff, Q. Liu, A. Zebaze, Q. Zheng, B. Hui, T. Y. Zhuo, S. Singh, X. Tang, L. Von Werra, and S. Longpre, "Octopack: Instruction tuning code large language models," *arXiv preprint arXiv:2308.07124*, 2023.

[20] NVIDIA Corporation, "NVIDIA ADA GPU Architecture (V2.01)," 2023. [Online]. Available: https://images.nvidia.com/aem-dam/Solutions/Data-Center/l4/nvidia-ada-gpu-architecture-whitepaper-v2.1.pdf

[21] ——, "NVIDIA Ampere GA102 GPU Architecture (V2.1)," 2021. [Online]. Available: https://images.nvidia.com/aem-dam/en-zz/Solutions/geforce/ampere/pdf/NVIDIA-ampere-GA102-GPU-Architecture-Whitepaper-V1.pdf

[22] ——, "NVIDIA H100 Tensor Core GPU Architecture (V1.04)," 2023. [Online]. Available: https://resources.nvidia.com/en-us-tensor-core/gtc22-whitepaper-hopper