Flash-Powered Mixture of Language Models Inference on Edge Devices

[Confidential]

08.05.2025

VISHWAS SAXENA, SENIOR TECHNOLOGIST
Deepankar Kansal, senior engineer

CONTENTS

(→)

Challenges in LLM Inference on Edge

Model Size & Complexity

- State-of-the-art LLMs have billions of parameters and require extensive memory and compute power only available on high-end GPUs.

Resource Constraints on Edge Devices

- Edge devices have limited GPU VRAM (< 16GB) and computing power, making local LLM inference infeasible without accuracy-reducing compression.

Limits of Multi-Model Deployment on Edge

- Multi-model setups demand far more resources than edge devices can provide.

Expensive GPU Hardware

Tier	GPU VRAM Range	Approx. Price (USD)
Entry/Mid	8 – 12GB	$700 - $1200
Mid/High	12 – 16GB	$1200 - $1600
Premium	16GB+	$2500+

Increasing GPU VRAM for inference on the edge devices is economically infeasible

Architecture Leveraging Flash

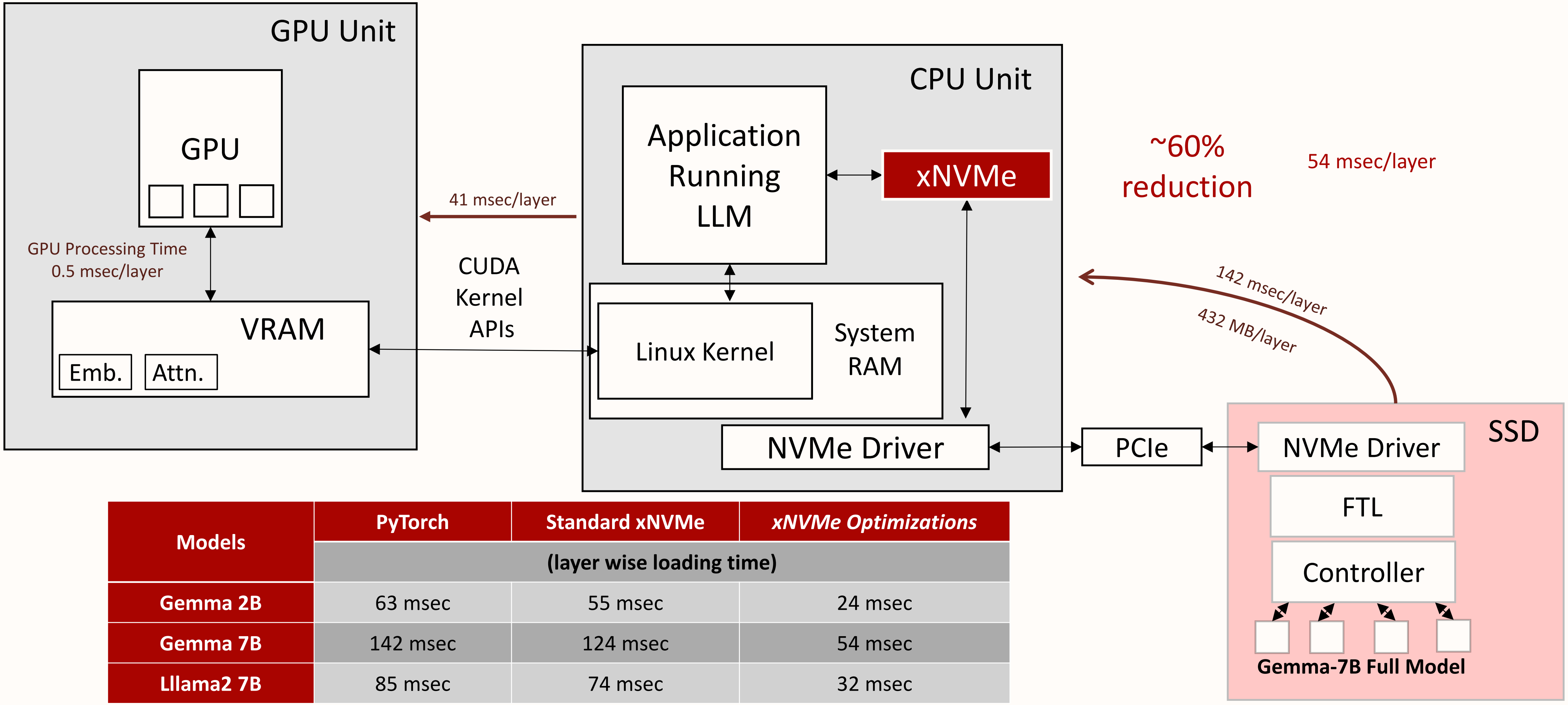
Can we stream parameters from flash memory while achieving acceptable inference?

Answer: Yes, and it opens up important opportunities for flash memory.

Speed up

- ***Row Column Bundling*** : Clustering the up and down projection neurons. This will help in reducing number of reads from the SSD.
- ***xNVMe Optimizations*** : SSD read I/O latency reduced by approximate 60%.
- ***Parallel Reads*** : Read from SSD to CPU (one layer ahead) and CPU to GPU in parallel.
- ***Streaming Approach*** : A portion of MLP layers stays resident on the GPU while subsequent layers are loaded one at a time.
- ***Reside Embeddings on CPU*** : This approach helps reduce GPU VRAM usage but may increase latency for certain models, depending on their architecture.

LLM loading from NVMe device

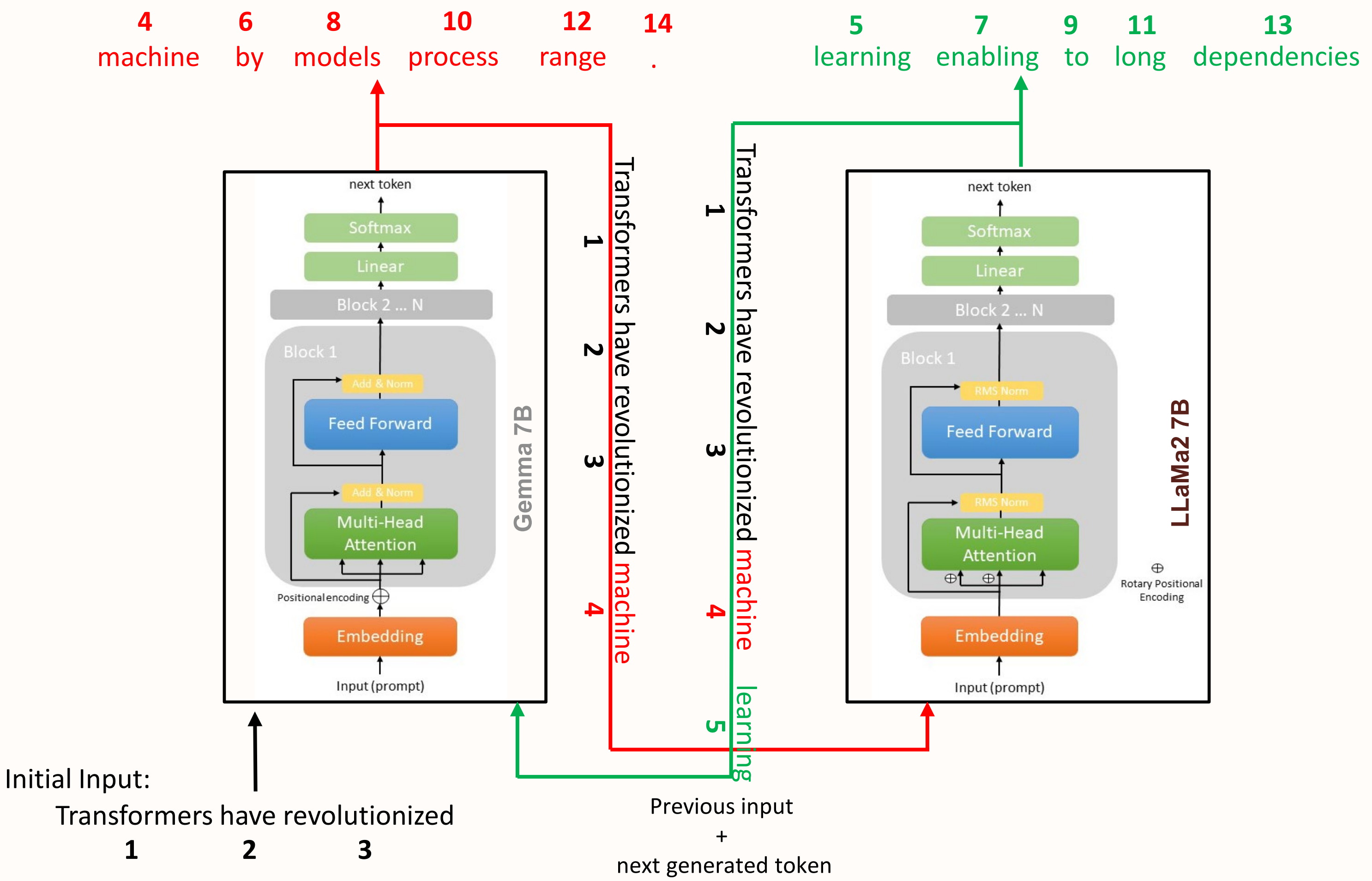


Models	PyTorch	Standard xNVMe	*xNVMe Optimizations*
	(layer wise loading time)		
Gemma 2B	63 msec	55 msec	24 msec
Gemma 7B	142 msec	124 msec	54 msec
Lllama2 7B	85 msec	74 msec	32 msec

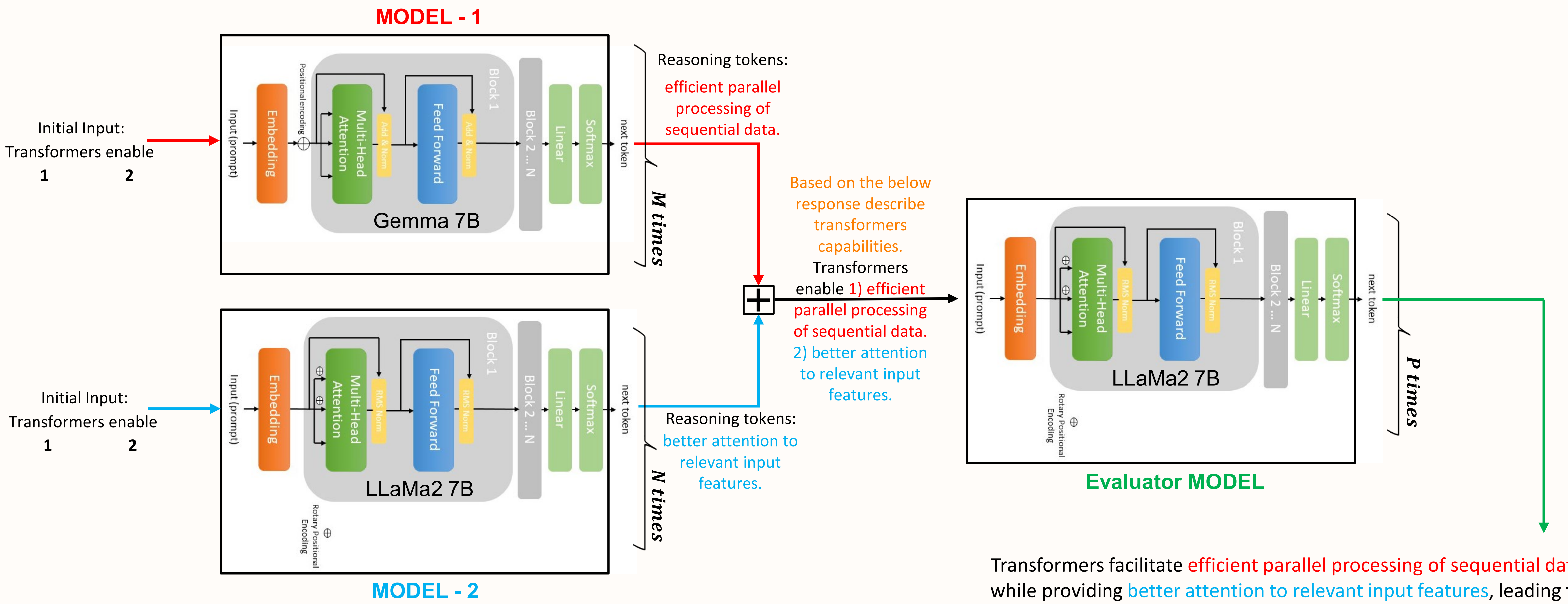
Interleaving Language Models

- **Inference from combination of models** on a single GPU.
- **Enhances response quality** by incorporating multiple expert perspectives.
- Encourages **diversity of thought**, increasing the reliability of the final input.
- Optimized setup to get the best performance from the GPU.

Final output:
Transformers have revolutionized machine learning by enabling models to process long range dependencies.

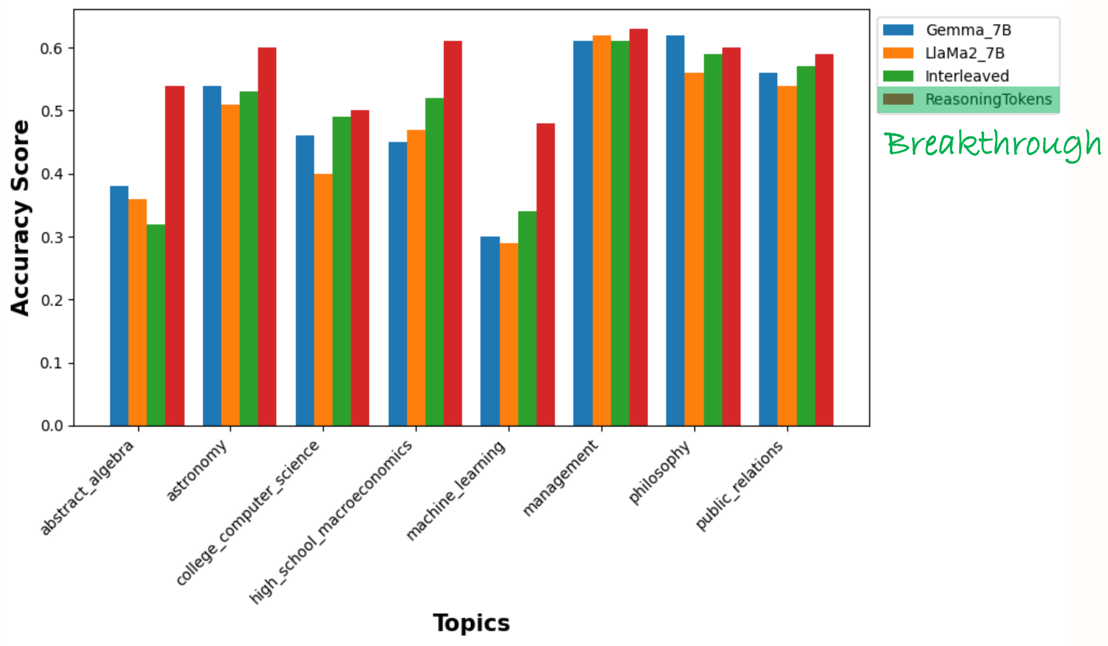


Enhancing Responses Through Reasoning Tokens



Transformers facilitate efficient parallel processing of sequential data while providing better attention to relevant input features, leading to improved performance in various natural language processing tasks.

Results



Accuracy scores of inferencing techniques for different topics from MMLU dataset

Memory footprint

Model	Traditional Method	**SHARAG**
Gemma 7B	17 GB	**5.1 GB**
Llama2 7B	15.5 GB	**5.8 GB**
Gemma 7B + Llama2 7B (Interleaved/Reasoning)	32.5 GB	**11 GB**

Minimum memory requirements of SHARAG over traditional inference methods

Summary

- **Efficient Model Loading:** Leveraging flash memory and an optimized xNVMe module enables direct streaming of language model weights to GPU VRAM, supporting inference on memory-constrained devices.
- **Maximized GPU Utilization:** The streaming technique maximizes GPU VRAM utilization by consistently loading the GPU to full capacity, ensuring optimal performance.
- **Advanced Inference Techniques:** Interleaving and Reasoning Tokens outperform traditional single-model inferencing.
- **Enhanced Edge Performance:** Dynamic loading from flash and inferencing techniques, when combined, provide a way to achieve improved LLM results on edge devices.

Upcoming Explorations – Optimization with Different Backends

Optimize backends provides specialized tools that compile and optimize how models run by planning data use ahead of time, making inference faster and smoother.

How ML Backends Work

Model

↓

Create Model Graph

↓

Analyze Execution Flow & Dependencies

↓

Build Execution Plan

↓

Identify Needed Weights & Tensors per stage

↓

Pre-Fetch Weights Before Execution

↓

Inference

Benefits For Our Project

01 With backend we can pre-fetch weights for each block, reducing I/O bottleneck and latency.

02 Developing a module tightly knit to backends; allowing efficient memory management without major changes to the inference source code.

Hence, we are looking forward to exploring open-source backends.

THANKS!

SANDISK™