Flash Powering the adoption of LLMs on Edge

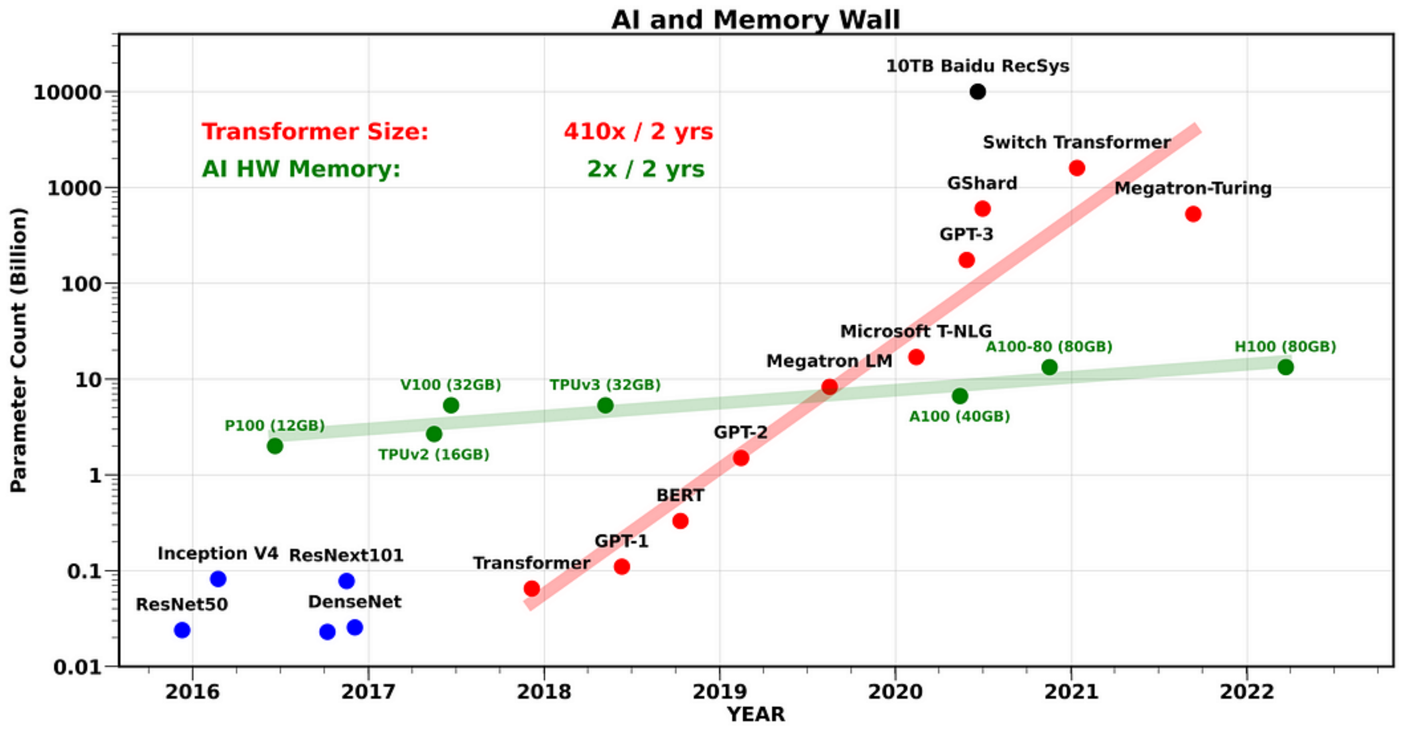
Vishwas Saxena

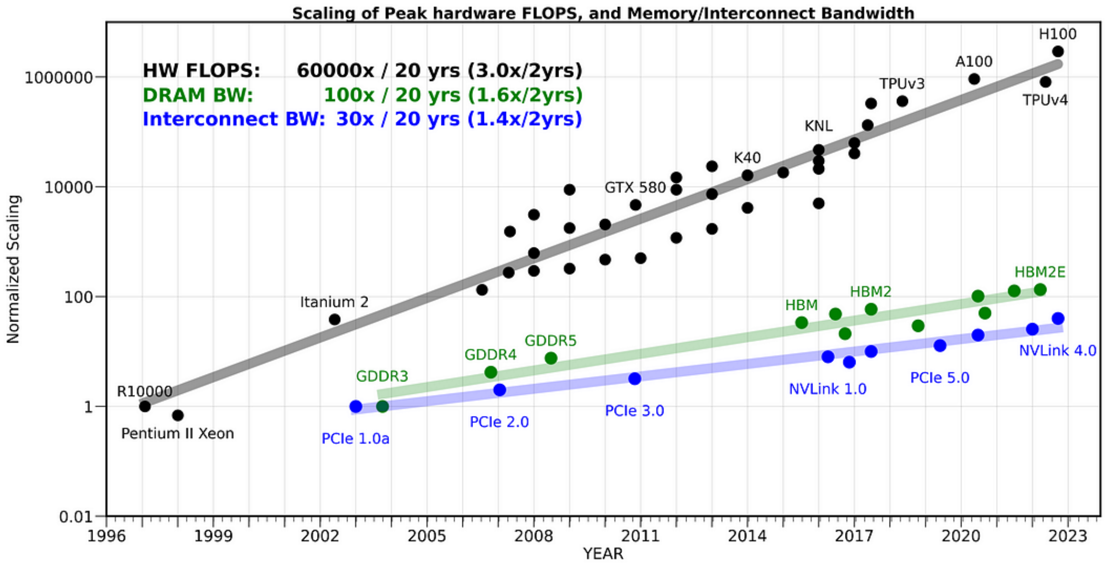
Senior Technologist, Western Digital

Aadharsh Kannan,

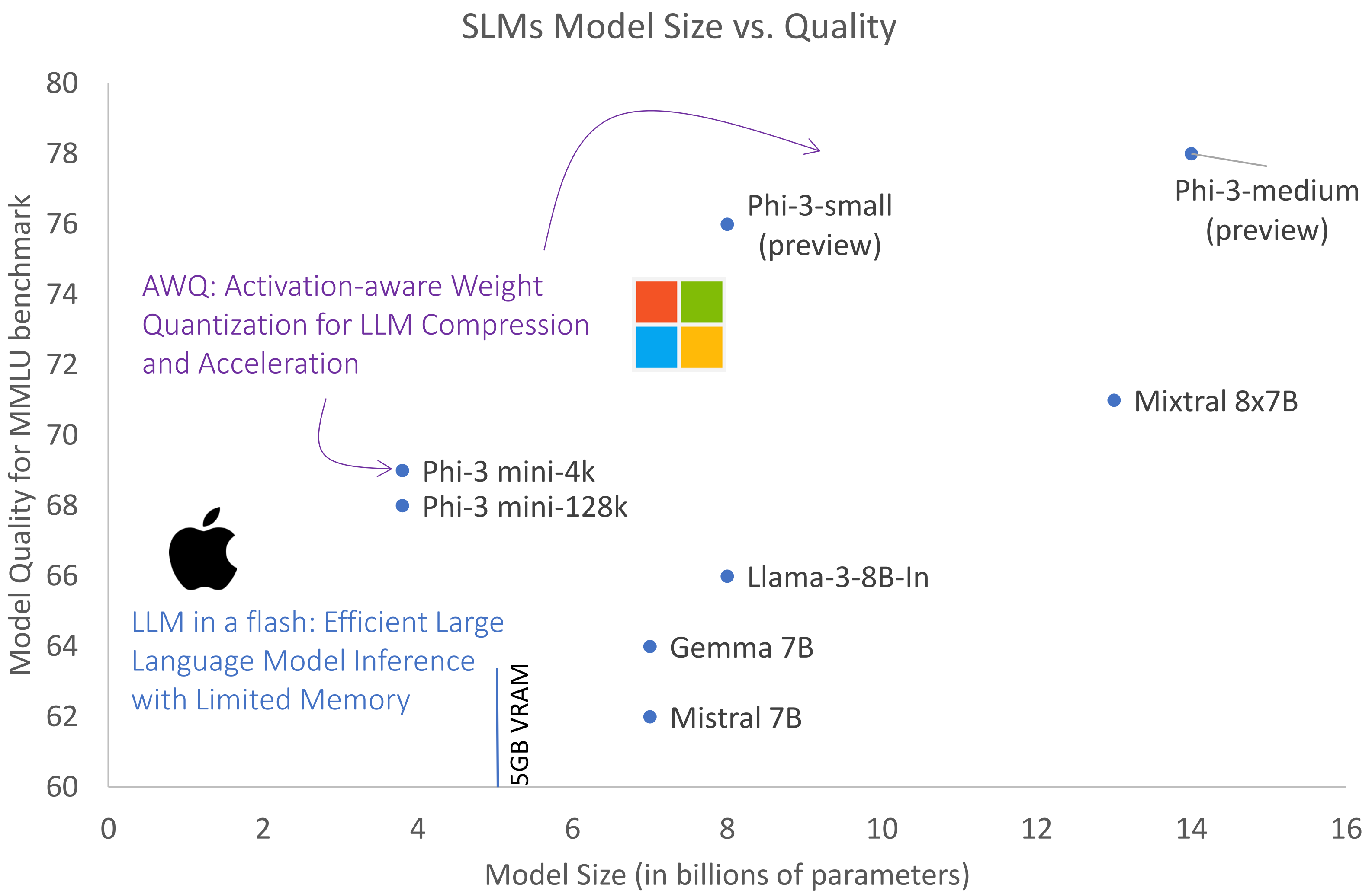
Vice President, Economics and Data Science, Western Digital

Memory Wall Problem in the Client





Memory Wall Problem in the Client



AI On Device: LLM Inference on Client Devices

Size (Llama2 7B model size is 14GB)

In today's architecture, significant portion of weights need to reside on GPU VRAM

On client devices, GPUs have limited VRAM capacity (typically 4 to 16 GB)

Increasing VRAM for inference on the client is economically infeasible

$1,500.99

Intel 13th Generation Core i9
Processor Model

Budget | Low | Medium | **High**

NVIDIA GeForce RTX 4060
Graphics

8 GB GDDR6 VRAM

Entry-Level | Mid-Range | **High-End**

32 gigabytes
System Memory (RAM)

Low | Medium | High | **Very High**

$625.00

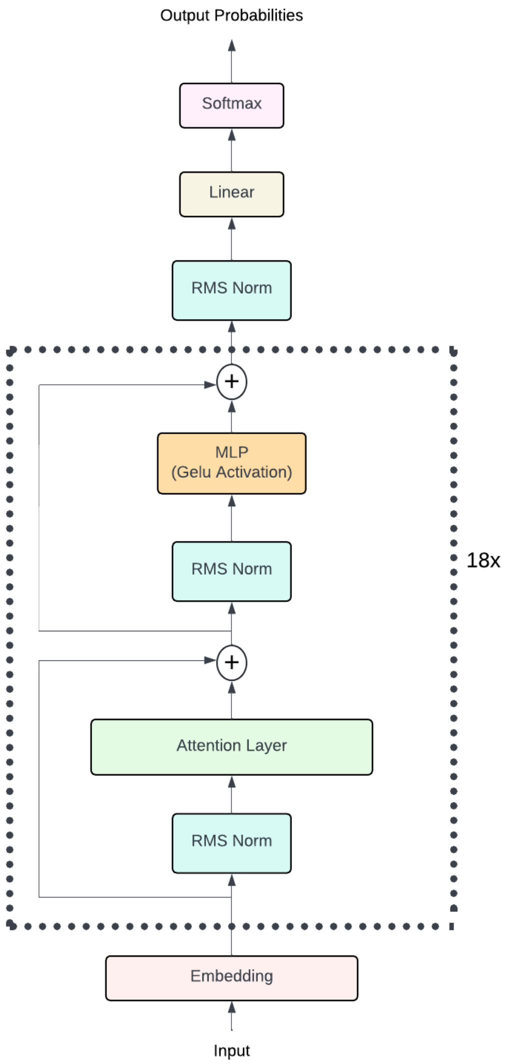
NVIDIA RTX 2000 Ada Generation

MPN: 900-5G192-0041-000

$625.00

- GPU Memory Size: 16 GB GDDR6 with ECC
- Form Factor: 2.7"(H) x 6.6"(L), dual slot, half height.
- Thermal Solution: Blower Active Fan

LLM Architecture



		Sparse
Embedding layer	Attention Layer	MLP Layer
20%	8%	72%

Contribution of Layers towards the size of Model

The number of parameters in Gemma 2B : 2.51 billion.

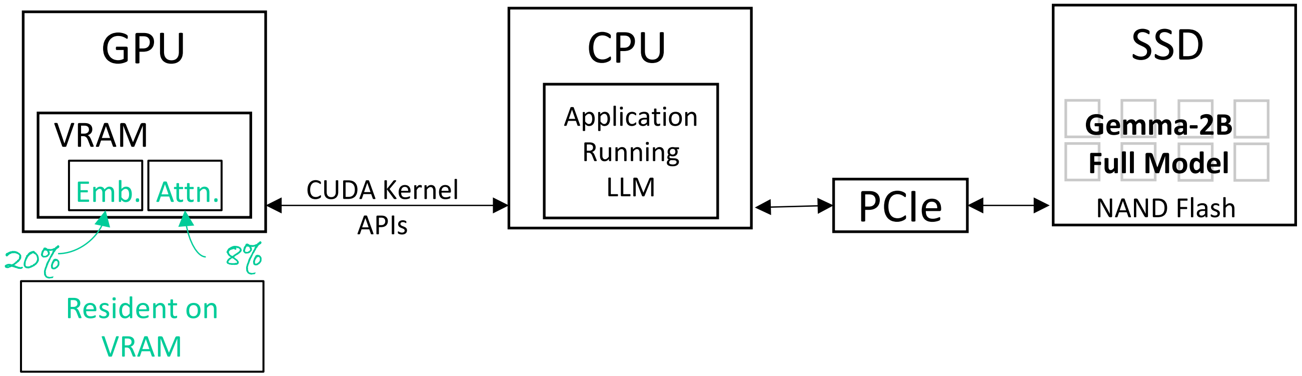
Each parameter is of the type FP16 (half – precision).

Partially Reside LLM on VRAM

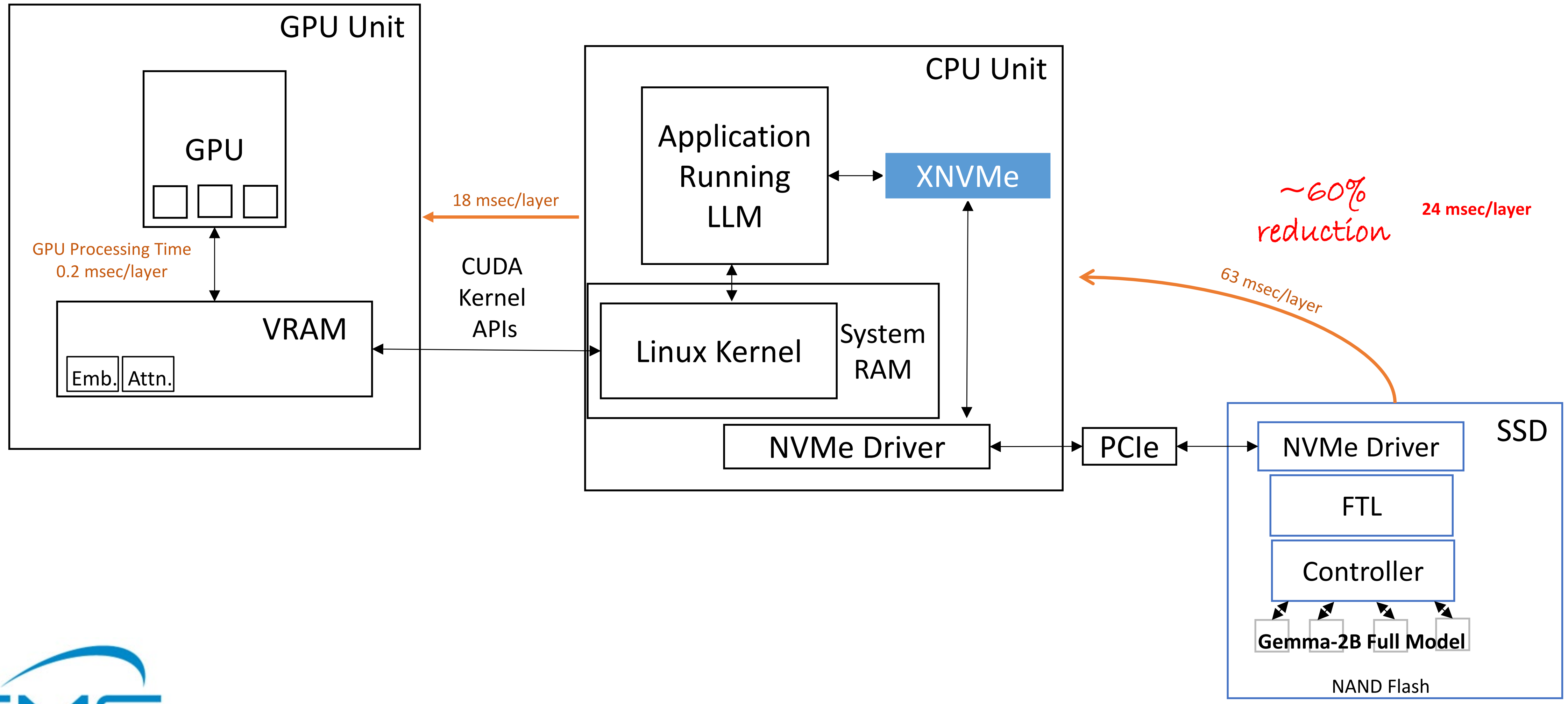
Each LLM varies in the percentage composition of its layer components.

Based on the layer composition, we can keep some part of the LLM resident on the GPU VRAM.

Example: In Gemma 2B parameter model, keep 28% (20% of embedding layer (Emb.) and 8% of attention layer (Attn.) resident on the GPU and load 72% of MLP layer (non-resident) on demand.



LLM loading from the NVMe device



Alternate architecture that leverages Flash

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

Several categories of LLMs exhibit a high degree of sparsity.
Can we leverage this to selectively load parameters to avoid redundant computations?

Squeeze

LRP (Low Rank Predictor): Predict which neurons will remain active and which ones will be zero; we then omit the zeroed-out neurons.

Speed up

Row Column Bundling: Clustering the up and down projection neurons. This will help in reducing number of reads from the SSD.

Can we co optimize the LLM and drive architecture?

Summary

Enabled Gemma to run on 4GB GPU VRAM machine by detecting sparsity using LRP.

Reduced data load time by a factor of 3 using XNVMe.

Integration of staging and prediction algorithm with XNVMe load/store.

Upcoming explorations

Train LRP on larger datasets to get enhanced accuracy.

Work on larger LLMs – as an example Llama2 7B model with ReLU activations with 90% sparsity.

Apply windowing techniques to load parameters only for recent tokens.

Thank You