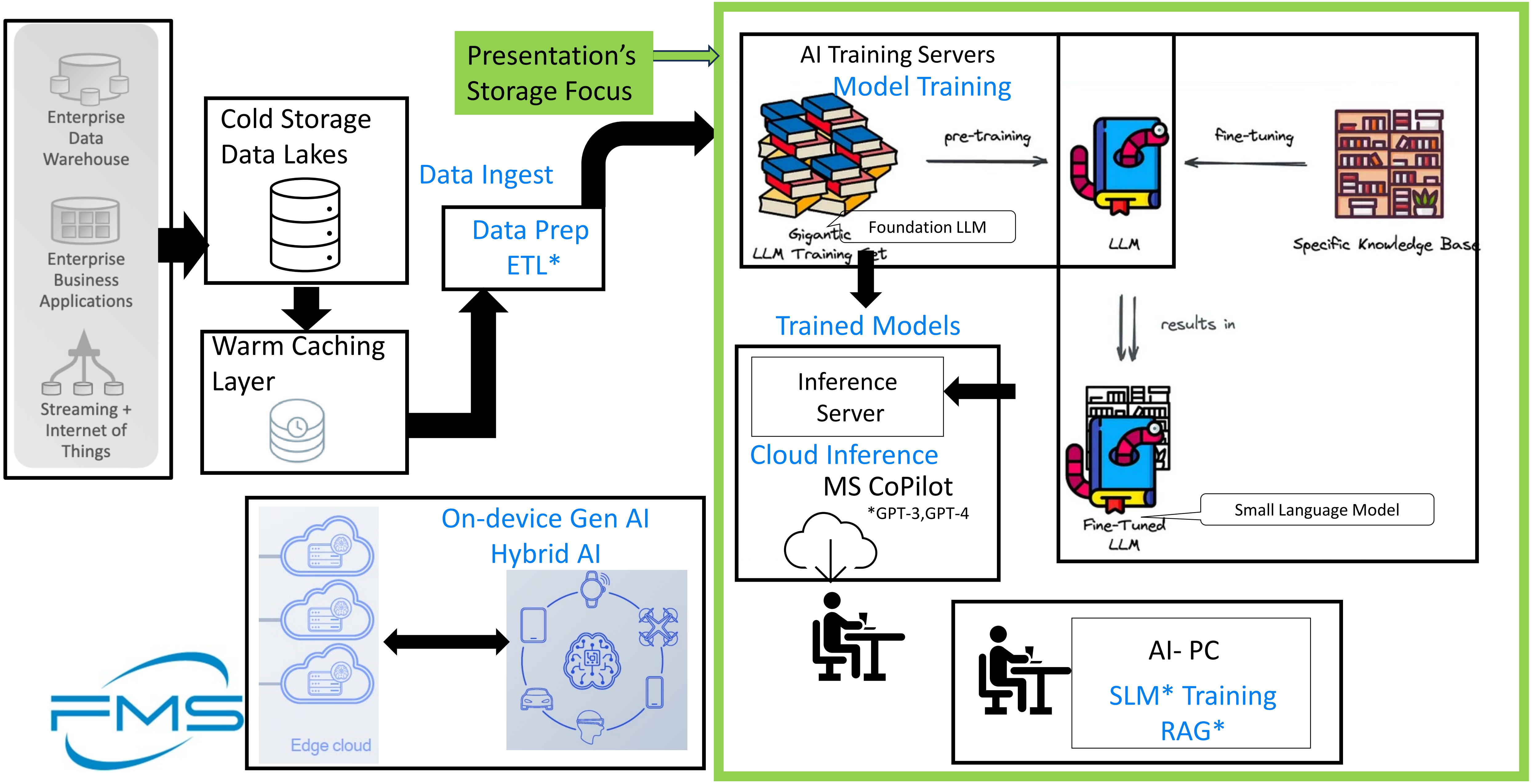
What can Storage do for AI?

Presenter: Suresh Rajgopal

Distinguished Member of Technical Staff (Micron Technology)

AI/ML pipeline and Storage Use cases

*ETL- Extract, Transform, Load
*SLM- Small Language Models
*RAG – Retrieval Augmented Generation
(“Smart Search and Retrieval in PC”)
*GPT - Generative Pre-trained Transformer



Outline

- Motivation
  - Why do we need (NVMe) Flash Storage to play a larger role in Training and Inference?
- Opportunities
  - Where can Flash storage contribute?
- Illustrated Example
  - What did we learn about flash storage in AI Training/Inference from our testing?

FMS

Cost, Power and Time impacts of Training [0]

Cost

- Each training run of GPT-3 cost 5M$[1]
- Cost of foundational model training is over 100M$[1]
- Largest models can cost **>1B$** to train by 2027[2]

Time

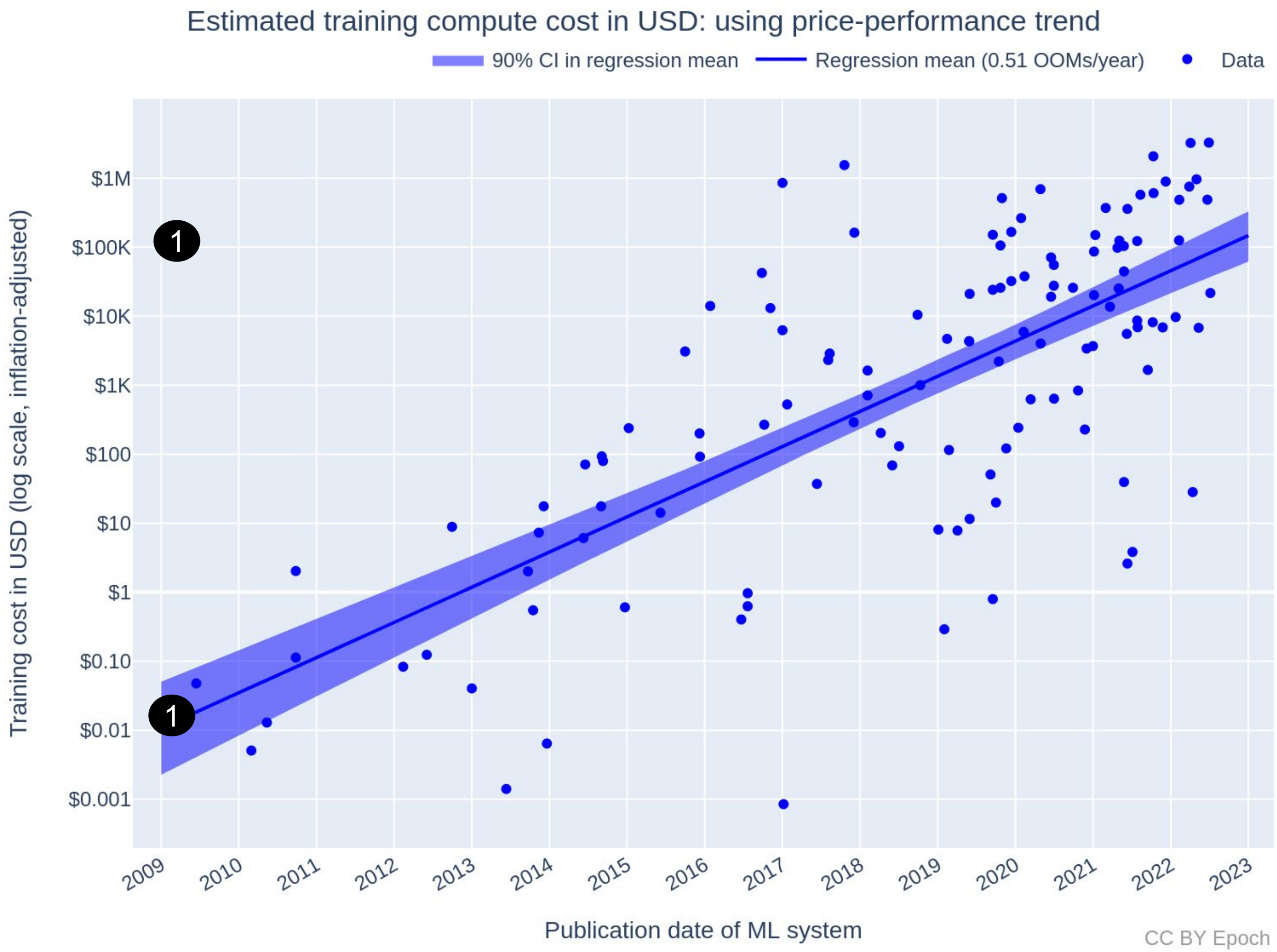
- Meta’s Llama2 70B model took 1.7Mhrs[3]
- Palm-540B model took 8.8Mhrs[3]
- Training GPT-3 - **36yr**s with 8V100 GPUs/ or 7months with 512 GPUs[4]
- GPUs utilization is best-case **50%** usually much lower [0]

Power

- GPT-2 model training consumed 28MWhrs[5]
- GPT-3 consumed **10X** more 284MWhrs. [> 500 refrigerators running annually!!]
- Google just reported a 48% greenhouse gas increase due to AI in datacenters[6]

Foundational Model Training will be accessible to only a very few

The Need to Democratize Training



Training Cost (EpochAI.org)

- 0.5 order of magnitude cost increase ($10^{0.5}$) every year ~ 3X
- Cost = Hardware Cost + Energy Cost
  – Upfront HW Cost and %age time spent on training?
  – Energy Cost = Power x training time x Energy Rate
- 124 ML systems (not just LLMs)

Making SLM Training accessible to more data scientists is a growing challenge

NVMe Storage Offload in AI Training

- AI Training relies on keeping all training related data close to the GPU
  - Type of Data
    - Model Parameters (Weights and Biases)
    - Optimizer States (between training batches) and Gradients (parameter adjustments)
    - Checkpointing data (intermediate states)
    - Working Memory (during forward/backward passes)
  - For a 1T model, GPU requires ~30TB of operational training data – "Memory Wall"
  - Grows with model size and context size
- Today
  - Model scaling relies on aggregating GPU Memory (across several 100 GPUs)
  - 3D Parallelism – Data, Tensor or Pipeline parallelism
- Offload
  - Leverage heterogeneity in AI Servers – distribute training data in CPU/CXL/NVMe Flash

Effective Offload can provide a significant cost and power benefit

*SLM=Small Language Models

Opportunities for Offload to NVMe Storage

1. Foundational Model Training, Democratizing Training of SLMs*

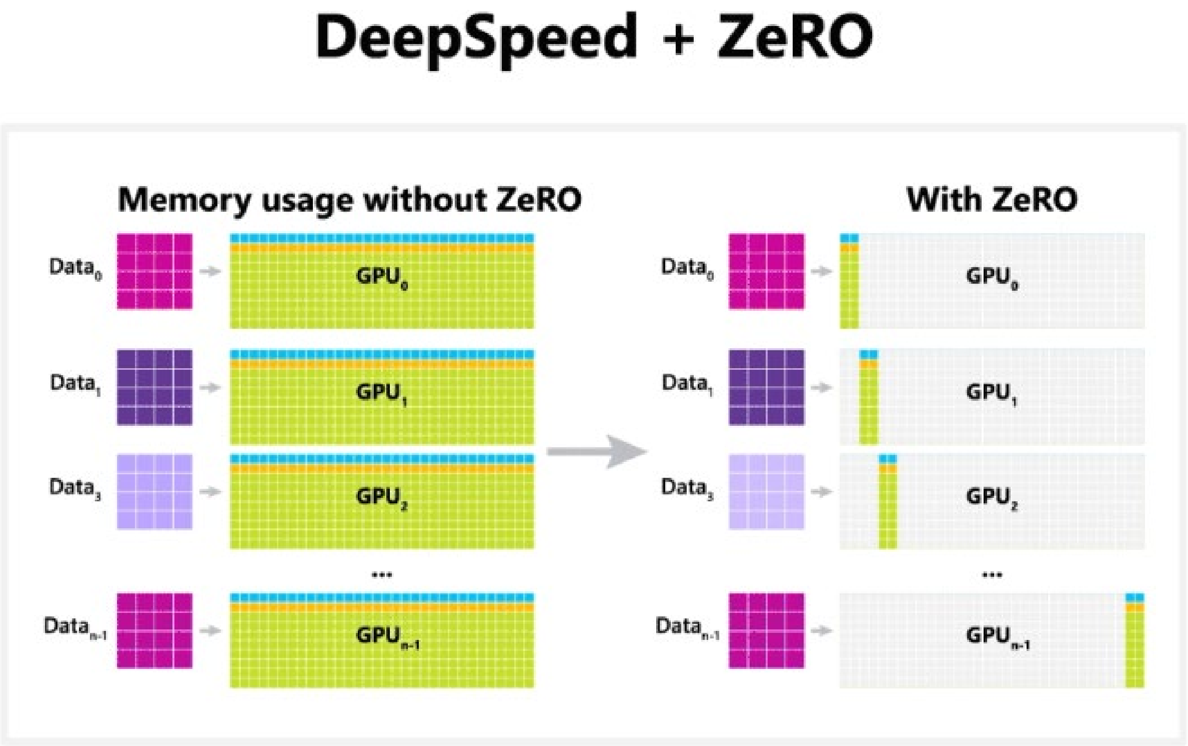
2. Inference on the Cloud

3. Enabling Inference on PCs, Mobile Devices and the Edge

Microsoft Deep Speed- important contributor to enabling Offload during Training and Inference

Deep Speed – Microsoft "AI at-Scale Initiative"

- Open-source optimization library for Distributed Training and Inference
- ZeRO optimization technique (ZeRO-1, ZeRO-2, ZeRO-3)
  - Eliminates memory redundancies during training optimizations
  - Partitions model states (parameters, optimizer states, gradients) across multiple devices



ZeRO Benefits: Scale model size, No model code refactoring needed

ZeRO-Offload- Democratizes model training

- Extend ZeRO to offload model states from GPU memory to CPU/NVMe
- Key Optimizations
  - Partition model parameters, optimizer states and gradients across different memory tiers
  - Overlap slower tier (memory/storage) access with computation
- ZeRO-Infinity – ZeRO Offload to scale model training
- ZeRO-Inference – Adapts ZeRO-Infinity for inference (offload: model parameters, KV Cache)
- Inference Requirements
  - Latency – How quickly can a model process an input prompt and produce outputs?
  - Throughput – How many inferences can the model handle per unit time? "batch size"
- Inference Types
  - Online Inference – latency is important, e.g. Chatbot
  - Offline/Batch Inference-Thruput is important along with HW utilization, e.g. User Recommendations, Caption generation, Batched article summaries

NVMe Offload is a good candidate for Offline Inference

*SLM=Small Language Models

Other opportunities for Offload to NVMe Storage

1. Training Large Foundational Models, Democratizing Training of SLMs*
    - Zero, Zero-Infinity (MSFT Deep Speed)
    - Benefits- Scale model size, No model code refactoring needed, improved GPU utilization
    - Requires- SW optimizations – partition parameters, overlap compute with storage access

2. Inference on the Cloud
    - DLRM Inference (Meta)
    - Benefits -Large batch sizes, ML Ops (multiple models), power and cost benefits (less GPU memory)
    - Requires-Choosing what to offload (capacity-bound vs compute/BW intensive),

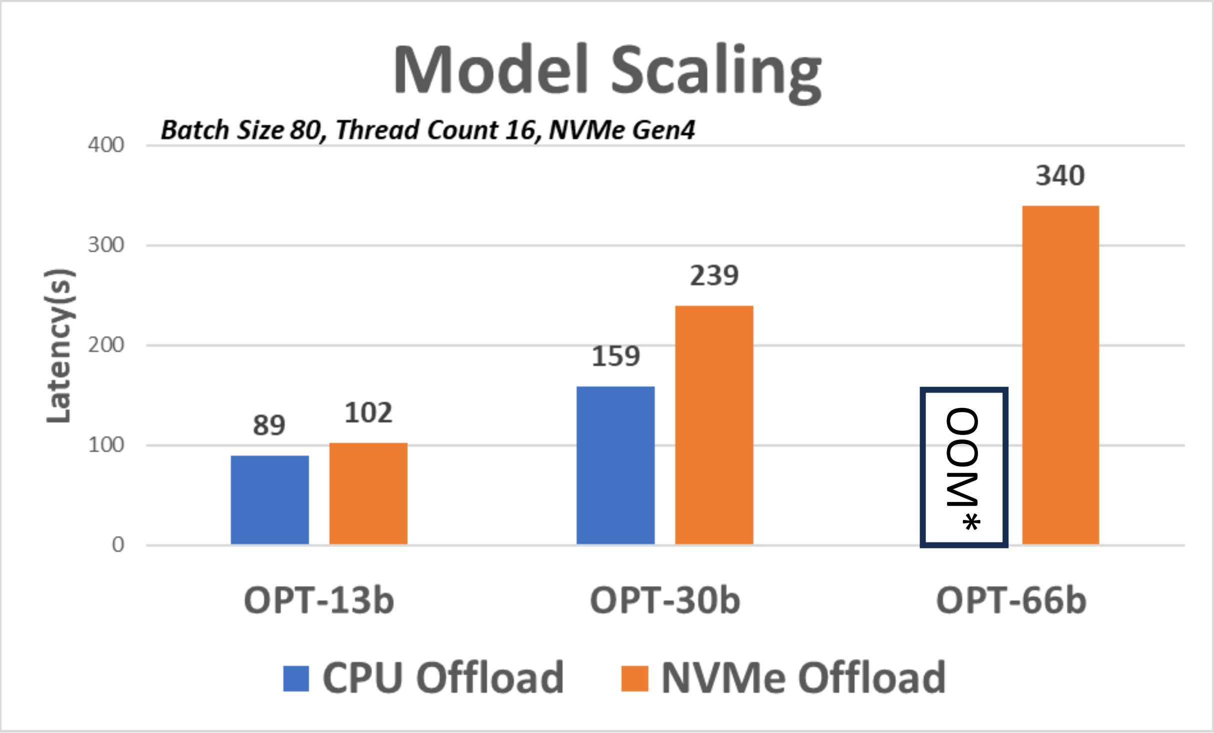
3. Enabling Inference on the Edge
    - LLM inference on resource-constrained devices (Apple)
    - Benefits –Cost, power savings, Latency to first token
    - Requires-Flash aware optimizations (32K or larger – "read and discard" over "reading small")

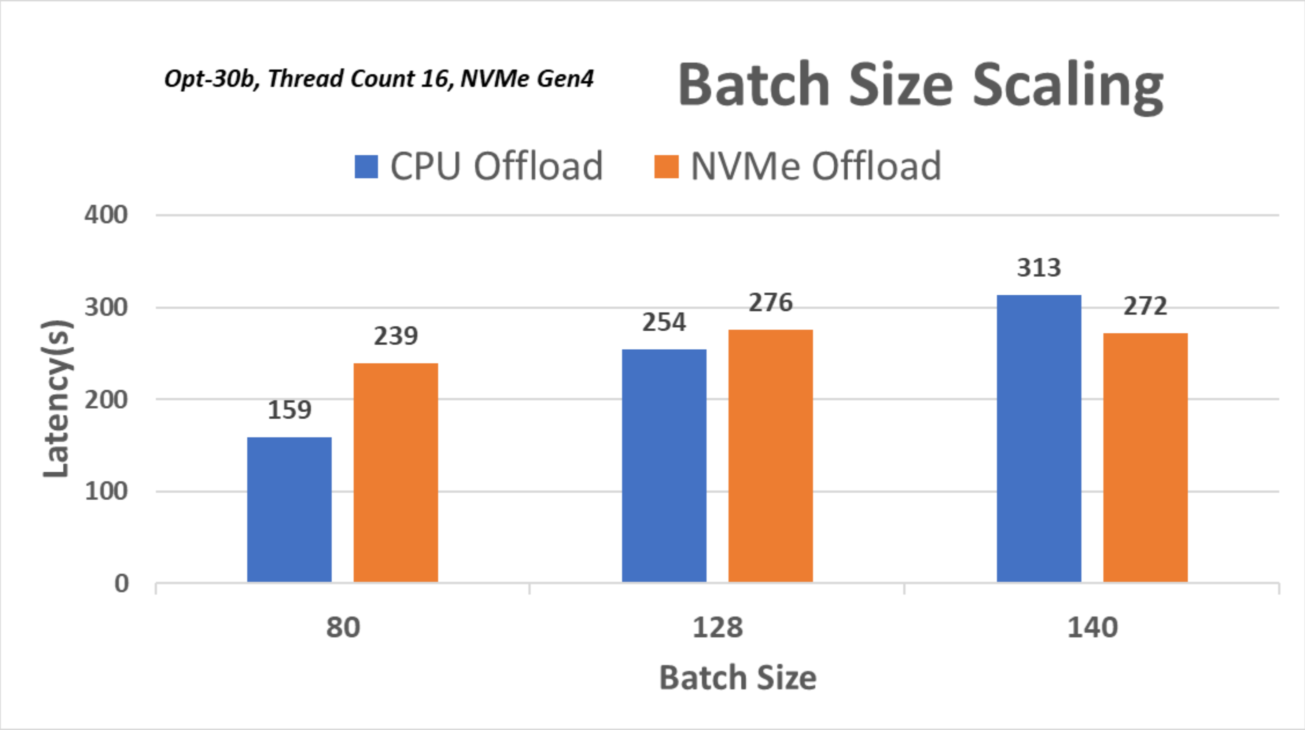
NVMe Offload with ZeRO Inference

- System: (representative resource constrained system)
  - Dell PowerEdge XE8545(Gen4, Gen3) SuperMicro SYS-512GE-TNRT (Gen5)
  - CPU: AMD EPYC 7413 24-Core Processor
  - Memory: constrained to 256 GiB
  - GPUs: A100-40GB (single GPU), L40S (single GPU)
  - Micron NVMe: 4TB SSD
- Deep Speed ZeRO-Inference Configuration
  - 4b Weight quantization, KV Cache Offload, 512B Prompt Length
  - NVMe Block Size: 2MB, Thread Count 16 (Not the library default)
- Options explored
  - Models: Hugging Face OPT models (13b, 30b, 66b)
  - Batch sizes: 80, 128,140,
  - Offload to CPU vs Offload to NVMe SSD
  - PCIe Generations: Gen3, Gen4, Gen5
- Metrics
  - Prefill Latency (Input token processing - Compute bound can be parallelized)
  - Decoder Latency (Output token generation-not easily parallelizable)
  - Run-time (Total Latency)

Model Size and Batch Scaling Results

*OOM: Out of Memory





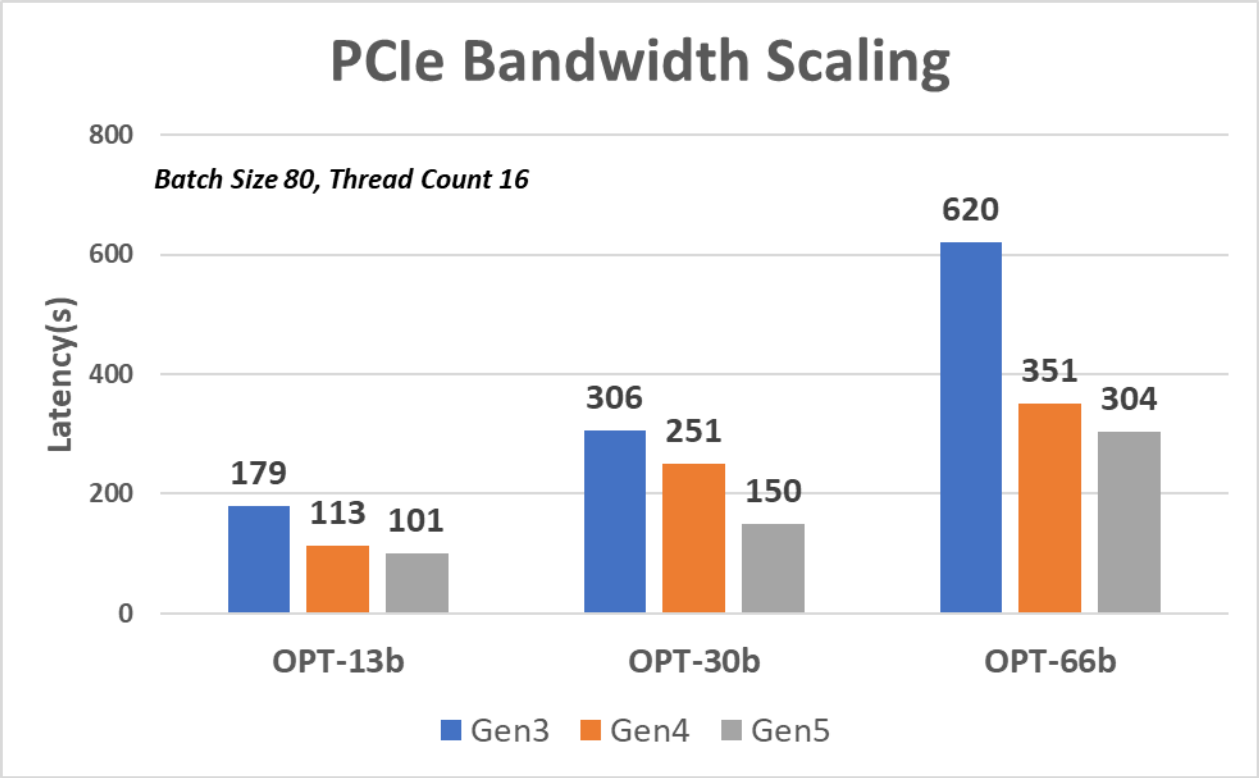
Data averaged across 5 inference runs

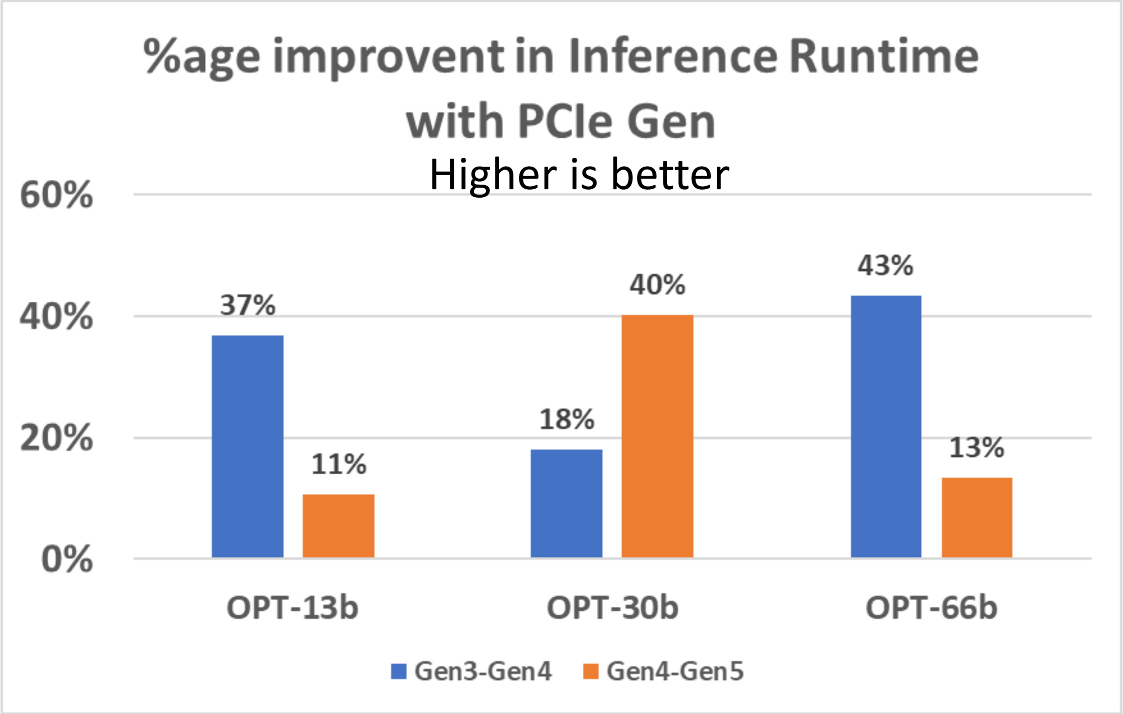
- Offload to CPU memory
  - Unable to support models > 30b (Out of Memory)
  - Does better for smaller models

- Model Scaling
  - As batch sizes increase NVMe offload run-times are better than CPU offload

Cost and Capacity of NVMe Storage enables model and batch scaling providing better perf/$

SSD PCIe Gen choice for NVMe Offload





- Larger models run an average of 40% faster with PCIe generation improvement
- Gen4- Gen5 improvements not as significant as Gen3-Gen4 – possible compute bottleneck that faster storage access cannot alleviate?

FMS

Observations

- NVMe Offload helps run larger models –better quality responses
- NVMe Offload can service larger batch sizes – more inference requests per unit time
- Offload libraries like ZeRO Inference should be leveraged
  - Will democratize training and inference and enable wider at-scale deployment of AI models
- Libraries can be further optimized
  - Larger Block Sizes
  - Greater Thread counts
- Faster NVMe SSDs will help further improve speed of inference
  - Average of 40% improvement in performance (PCIe Gen3 to Gen4)

Conclusions

- Power and Cost considerations for AI-at Scale deployment are real
- NVMe offload can be a cost and power efficient alternative to democratize training and cloud inference
- Benefits of high-quality responses from larger models can be leveraged by resource constrained mobile, client and edge devices
- Enabling NVMe Offload requires
  - Careful model optimizations to hide storage latency behind compute
  - Large blocks sizes and use of multiple threads further accelerate SSD performance
- Storage for AI – Call to Action
  - Move to faster PCIe interfaces on SSDs
  - Focus on Read performance, optimize bandwidth over latency(needs to be hidden)

Thank You!

- Questions?