Behemoth: A Flash-centric Training Accelerator for Extreme-scale DNNs

Kim, Shine., Yunho, Jin., Gina, Sohn., Jonghyun, Bae., Taejun, Ham., Jaewook, Lee. **USENIX FAST'21**
Seoul National University and Samsung Electronics

2024. 08. 21

Presentation by Nakyeong Kim
nkkim@dankook.ac.kr

Contents

1. Introduction

Extremely Large Model Era

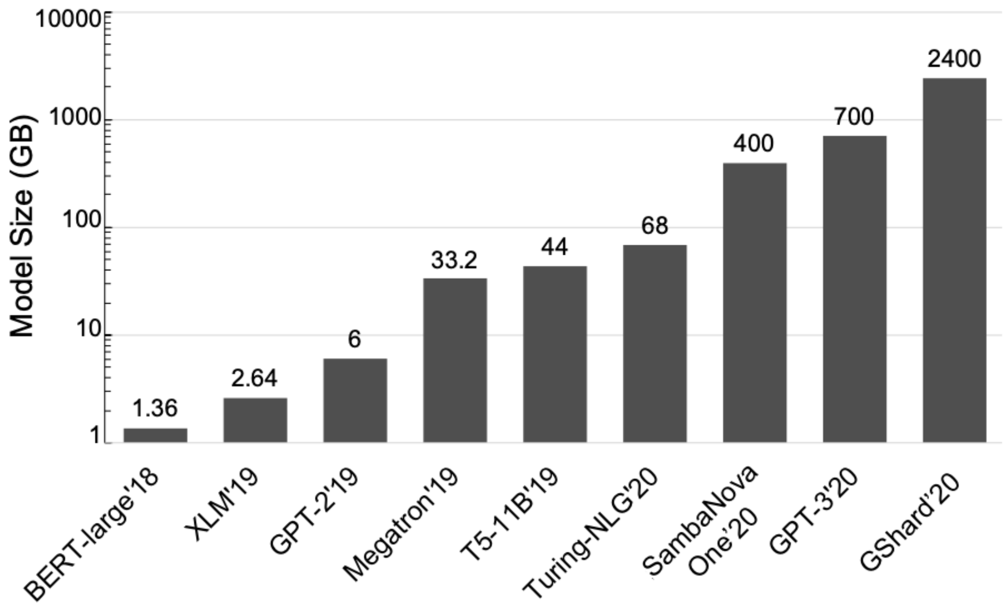


Figure 1: Trends of model size scaling with large NLP models

1. Introduction

Extremely Large Model Era

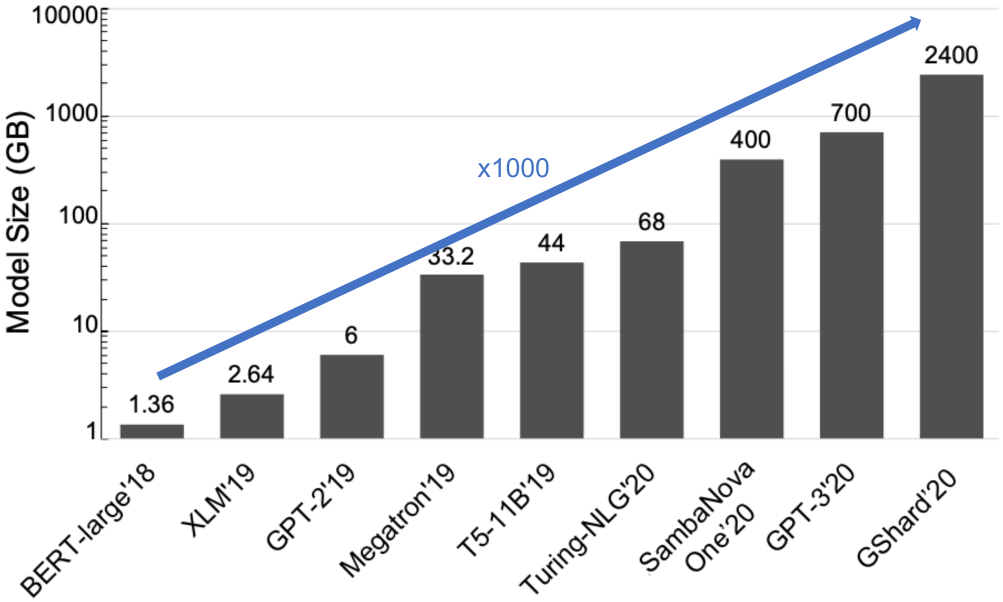


Figure 1: Trends of model size scaling with large NLP models

1. Introduction

Extremely Large Model Era

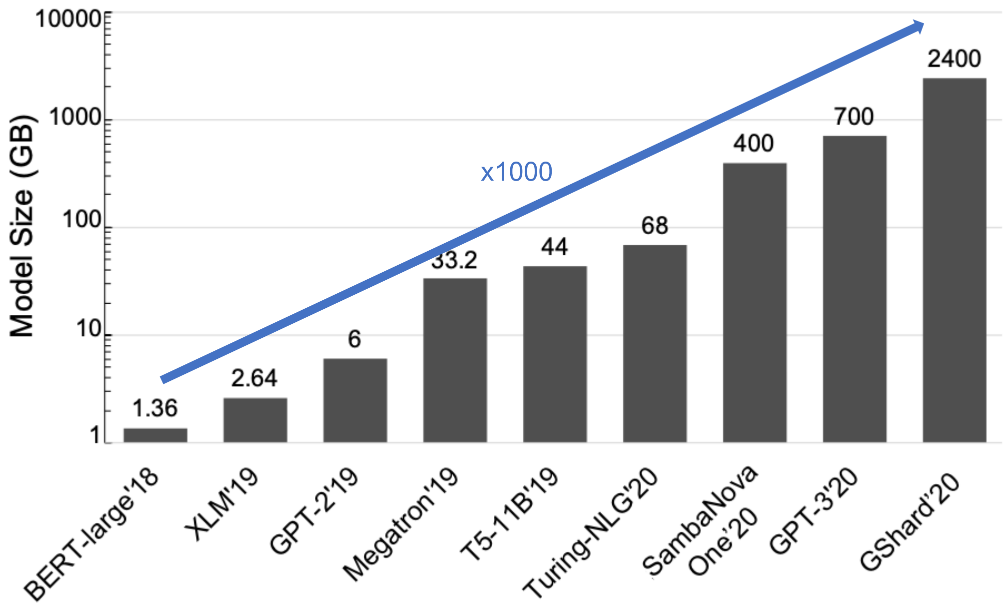
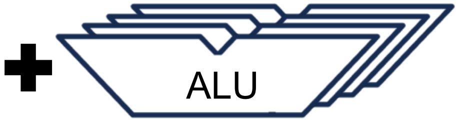


Figure 1: Trends of model size scaling with large NLP models

1. Introduction

Extremely Large Model Era



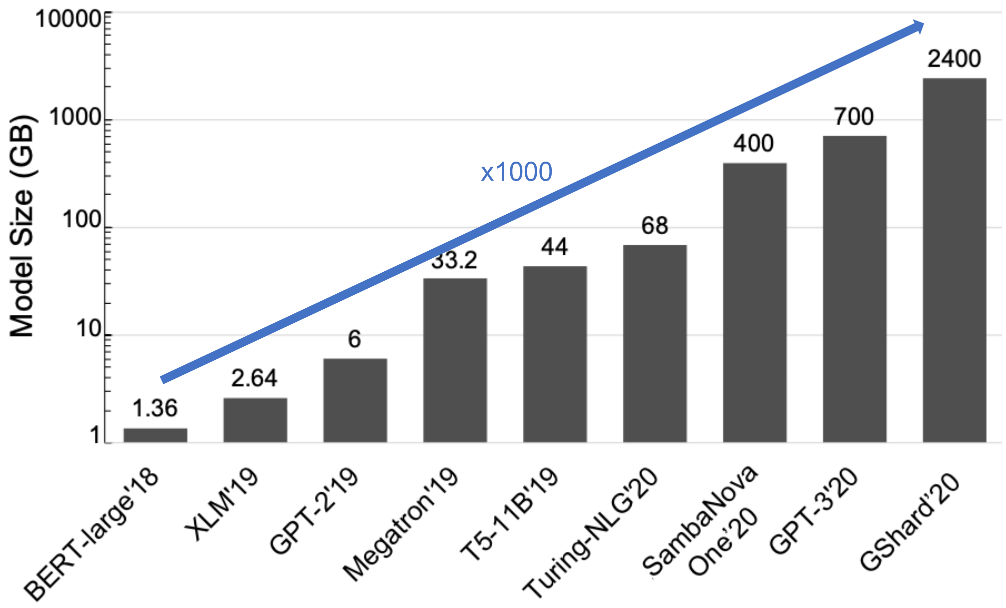


Figure 1: Trends of model size scaling with large NLP models

1. Introduction

Memory Capacity Problem

- Two Solution
  1. Discard some computation results and recalculate
  2. Utilize model parallelism (HBM-based memory)

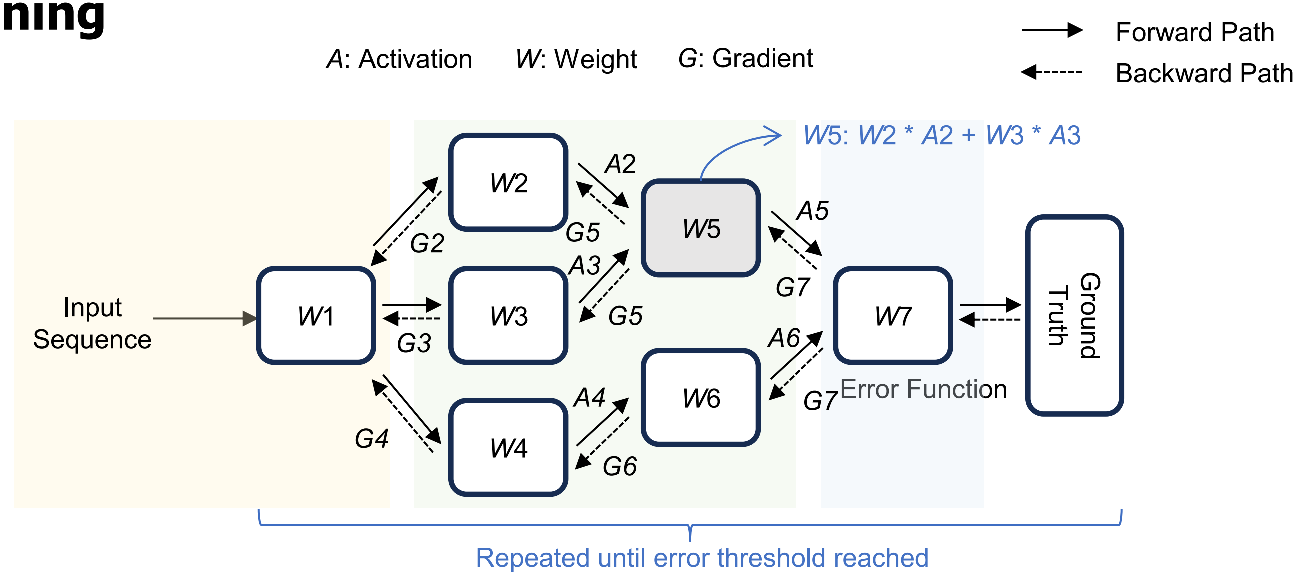
1. Introduction

Memory Capacity Problem

- Two Solution
  1. Discard some computation results and recalculate
     - Incur amount of extra computation
  2. Utilize model parallelism (HBM-based memory)
     - Require careful load balancing and stall arise (by dependency)

2. Background

DNN Training



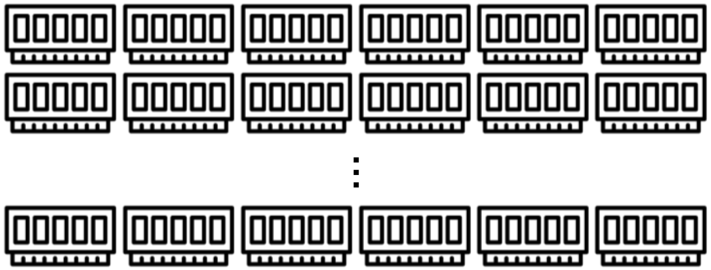
- Training process is deterministic
- Training require extra storage to buffer each layers' output and weight

2. Background

DNN Training – Challenge

- TPU have inefficient memory systems that unnecessarily expensive
  - Each value in the matrix is reused many times, requiring small number of memory access

GPT-3 (2.1TB)



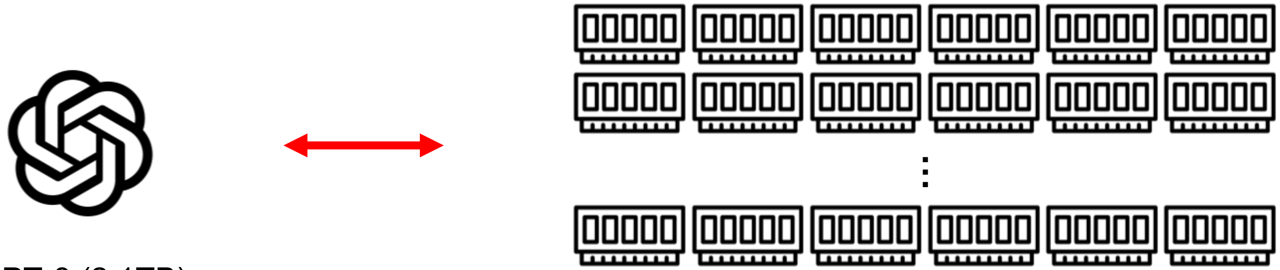
TPU with HBM (32GB) * 66

	Computation (TFLOPS)	I/O Bandwidth (GB/s)
GPT	73.7	9.26
TPU with HBM	105	600

2. Background

DNN Training – Challenge

- TPU have inefficient memory systems that unnecessarily expensive
  - Each value in the matrix is reused many times, requiring small number of memory access



GPT-3 (2.1TB)

TPU with HBM (32GB) * 66

	Computation (TFLOPS)	I/O Bandwidth (GB/s)
GPT	73.7	9.26
TPU with HBM	105	600

Under-utilized!

2. Motivation

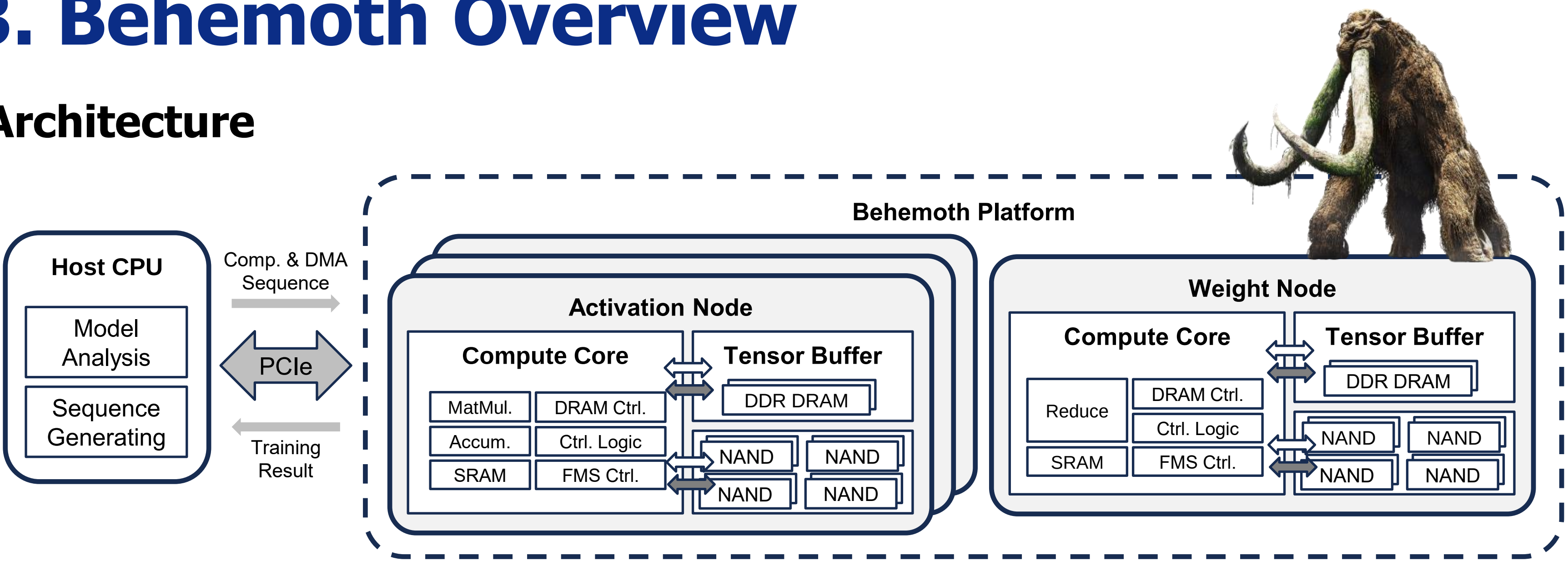
Flash Memory System

- Cost-effective large-scale language model training platform
  - Replace HBM to flash memory
  - Architect for language model

- Need to address
  - Extremely-low bandwidth
  - Endurance

3. Behemoth Overview

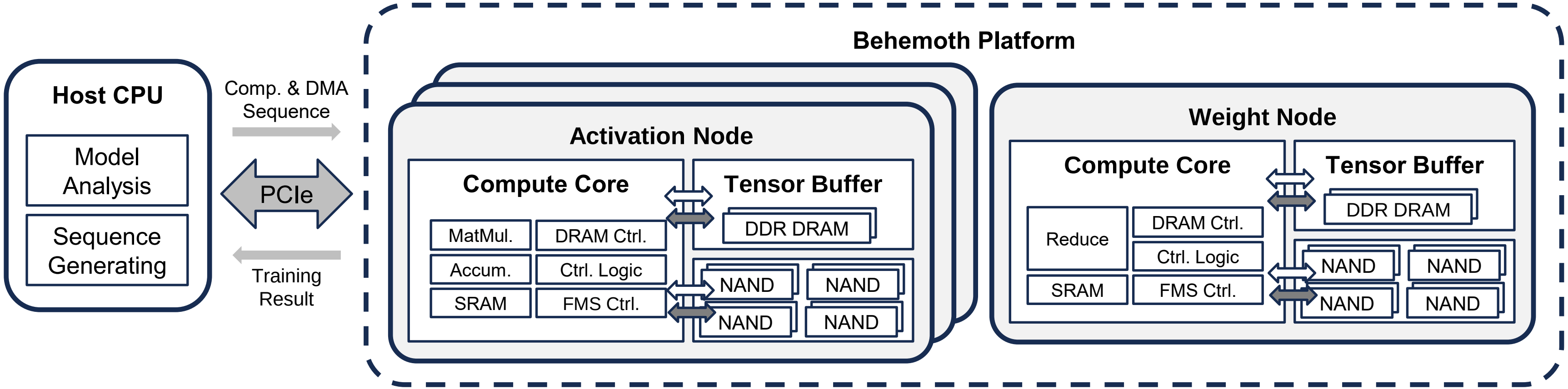
Architecture



- Data Parallelism
  - Training dataset is partitioned across multiple devices
  - To satisfy computation, memory, and bandwidth requirements, integrate one Weight Node with multiple Activation Nodes

3. Behemoth Overview

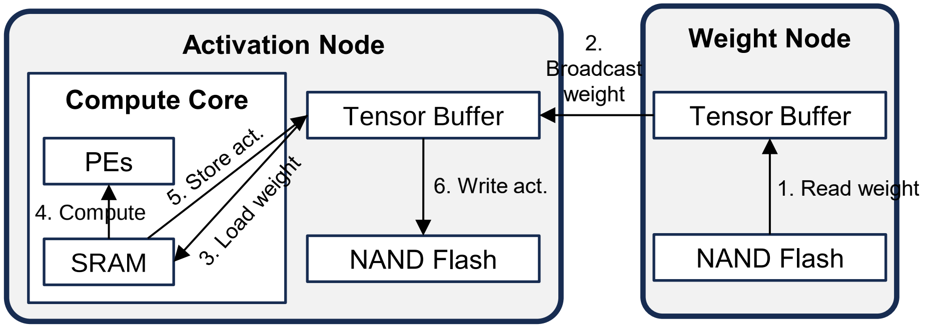
Architecture



- Model Analysis
- Sequence Generating
  - DMA command sequence: control data transfer between Tensor Buffer and NAND flash devices
  - Computation command sequence: list operation commands to perform on Compute Core

3. Behemoth Overview

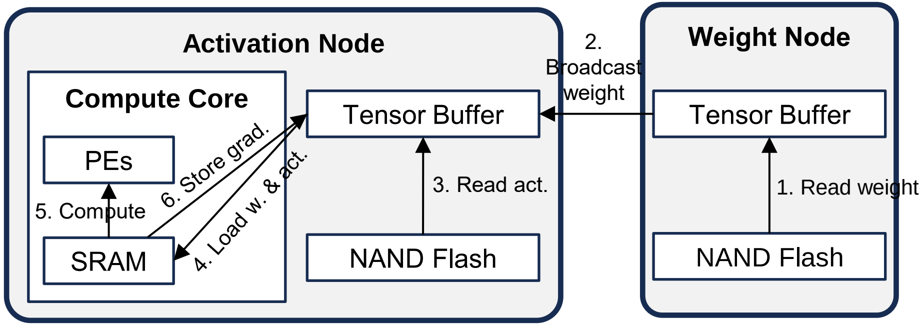
Execution Walk-Through – Forward Path



- Each steps can be overlapped
  While computation is performed on Activation Node, weights on Weight Node are prefetched
- Activation tensor in Tensor Buffer is written to NAND flash for reuse during backward propagation

3. Behemoth Overview

Execution Walk-Through – Backward Path



- Like forward propagation, all steps are pipelined and operated in parallel
- After calculation of all layers is completed, final weight gradient tensor is transferred from Activation Node to Tensor Buffer of Weight Buffer
- If all weight gradients have been received, Weight Node update training results to weights

4. Flash Management System

Problems

1. Limited bandwidth
2. Endurance

4. Flash Management System

Problems

1. Limited bandwidth
2. Endurance

To fully utilize high peak bandwidth of NAND device

- Make writing sequential as much as
- Prevent slow NAND firmware running from being bottleneck

4. Flash Management System

Data Access Pattern

Written by host before training start
Discarded once training finished

Written by Compute Core during forward path of training
Consumed during backward path of training

Table 1: DNN training data types and multi-stream support

#: Stream name (Act. Node / Weight Node)	Persistency	Retention	Access permission	
			Host	Behemoth
1: NV-Stream (Training inputs / –)	Non-volatile	Years	Append-only seq. write	Read only
2: V-Stream (Activations / Interm. weights)	Volatile	Minutes	N/A	Read & Append-only seq. write
3: NV-Stream (– / Trained weights)	Non-volatile	Years	Read only	Read & Append-only seq. write

4. Flash Management System

Data Access Pattern

Updated at end of each iteration

Only updated at end of training
Later read by host CPU

Table 1: DNN training data types and multi-stream support

#: Stream name (Act. Node / Weight Node)	Persistency	Retention	Access permission	
			Host	Behemoth
1: NV-Stream (Training inputs / –)	Non-volatile	Years	Append-only seq. write	Read only
2: V-Stream (Activations / Interm. weights)	Volatile	Minutes	N/A	Read & Append-only seq. write
3: NV-Stream (– / Trained weights)	Non-volatile	Years	Read only	Read & Append-only seq. write

4. Flash Management System

Data Access Pattern

- Each two data types housed in same device
- It is beneficial to separate different types of data to logically isolated spaces

Table 2: NAND block layout for a chip and multi-stream attributes of Activation Node

	NAND Block Layout					Stream attributes	
	PBN \ Plane	0	1	. . .	7	Capacity	P/E cycle/ Retention
	0 9	FTL Metadata (LBN2PBN map, PB metadata, etc)					
83	10 92	1: NV-Stream (training input)				249 GiB	50K / 1 year
578	93 671	2: V-Stream (activation data)				1737 GiB	2M / 1 day
	672 682	Reserved blocks for bad block replacement					

4. Flash Management System

Data Access Pattern

- Each two data types housed in same device
- It is beneficial to separate different types of data to logically isolated spaces

→ Sequential Write

Each single stream can have their own
logical address space,
access permission,
allowed P/E cycle

Table 2: NAND block layout for a chip and multi-stream attributes of Activation Node

	NAND Block Layout					Stream attributes	
	PBN \ Plane	0	1	. . .	7	Capacity	P/E cycle/ Retention
	0	FTL Metadata					
	9	(LBN2PBN map, PB metadata, etc)					
83	10	1: NV-Stream (training input)				249 GiB	50K / 1 year
	92						
578	93	2: V-Stream (activation data)				1737 GiB	2M / 1 day
	671						
	672	Reserved blocks for bad block replacement					
	682						

4. Flash Management System

Seperation via Data Types

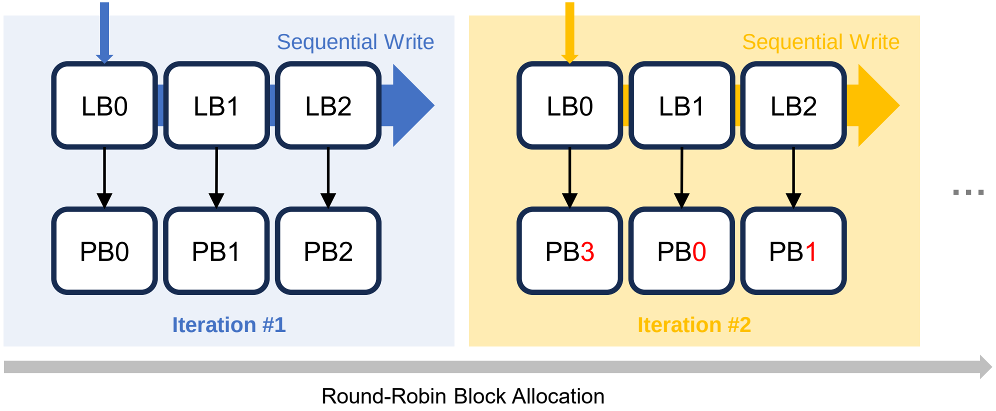
- Enable optimizations
  - Lightweight FTL

    Only sequential writes, complicated GC and wear-leveling is unnecessary

    ➔ Remove GC functionality

    ➔ Replace wear-leveling block allocator with simple round-robin block allocator

  - Hardware automation of write path



4. Flash Management System

Seperation via Data Types

- Enable optimizations
  - Lightweight FTL

  - Hardware automation of write path

    Modern SSD controllers adopt read automation feature by exploiting specialized hardware

    Write path is much more complex than read path (still rely on firmware)

    1. Garbage collecting
    2. Wear-leveling
    3. Guaranteeing data consistency
    4. Managing metadata for recovery
    5. Handling exceptions for P/E failures

4. Flash Management System

Seperation via Data Types

- Enable optimizations
  - Lightweight FTL

  - Hardware automation of write path

    Modern SSD controllers adopt read automation feature by exploiting specialized hardware

    Write path is much more complex than read path (still rely on firmware)

    1. ~~Garbage collecting~~
    2. ~~Wear-leveling~~ → Minimized
    3. Guaranteeing data consistency
    4. ~~Managing metadata for recovery~~ → unnecessary for temporary data
    5. ~~Handling exceptions for P/E failures~~ → Rare

    ➔ Prevent firmware from being bottleneck

4. Flash Management System

Endurance

- FMS use flash as temporary buffer for activations and intermediate weights
- Frequently reprogrammed values → affect SSD lifespan? (no)
- If stored data only a few minutes, enough to keep data until guaranteed retention time
- Reduced retention reduce need for hardware resources
  (e.g., complex ECC engines or extra over-provisioning space)

5. Evaluation

Sequence: 2048 tokens
Batch: 1
Activation: 1 batch
Size: *Width* x *Dimension*

Environment

- Comparison
  - TPU-based DNN training system (for Behemoth) 1)
  - Conventional SSD (for FMS) 2)
- Metric
  - Memory cost (1)
  - Throughput (2)
  - Tensor lifespan (2)
- Model
  - Compute Core by MAESTRO
  - FMS by MQSim

- Workloads (12)

Table 3: DNN models evaluated with Behemoth. We use a sequence length of 2048 (tokens) for each model.

Model	Size	Total act. (GB)	Total weight (GB)	PFLOP
BERT/GPT3-like [5, 18]	1×1	44	350	2.15
	1×2	88	698	4.42
	1×4	175	1393	8.56
	2×1	88	1395	8.56
	2×2	175	2786	17.12
	2×4	349	5569	34.21
T5-like [54]	1×1	40	305	0.62
	1×2	80	609	1.25
	1×4	160	1218	2.49
	2×1	80	1218	2.49
	2×2	160	2436	4.99
	2×4	319	4871	9.97

5. Evaluation

Memory Cost – Platform

- Model parallelism is difficult to load-balance
  - GPT-3 with 24-stage pipeline
- Data parallelism enable complete model to be trained on single device

Table 4: Platform configurations for the cost evaluation of Behemoth.

NPU Parameters		
Number of cores	16 cores (52.5 TFLOPs per core)	
Number of PEs	524,288	
Peak throughput	840 TFLOPs	
Host I/F conf.	PCIe Gen4 × 32 lane [51]	
Memory Parameters		
	Resembled TPU [27]	**Behemoth**
Buffer conf.	16GB HBM	16GB DDR4 DRAM + 2TB NAND flash
Peak bandwidth	300GB/s	50GB/s
Compute Parameters		
Parallel comp. method	Model parallelism	Data parallelism

5. Evaluation

Memory Cost – Platform

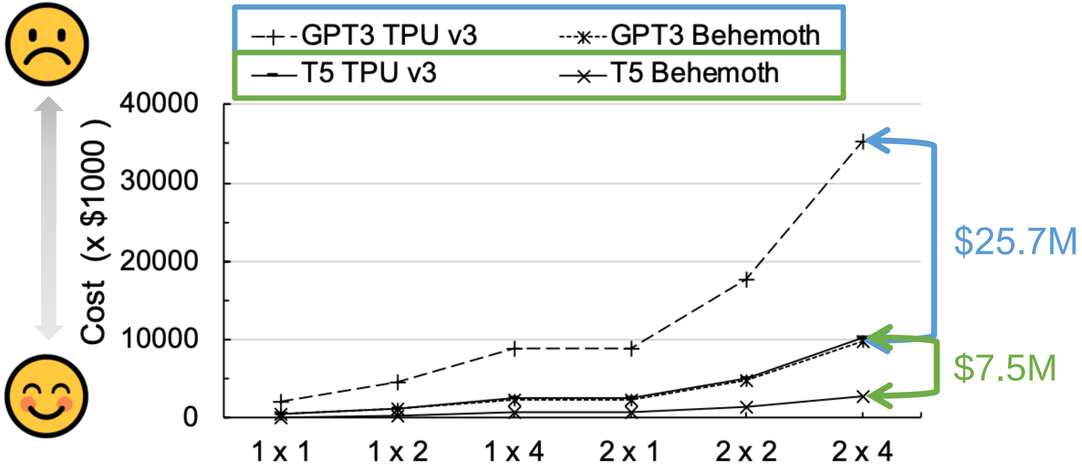


Figure 8: Memory cost comparison between TPU v3 [27] and Behemoth. $W \times D$ in the figure illustrates that the dimension of each layer is increased by W times and the number of layers is increased by D times.

5. Evaluation

Memory Cost – Platform

GPT3 have many parameters,
which need more memory capacity in training
It also support more long sequences

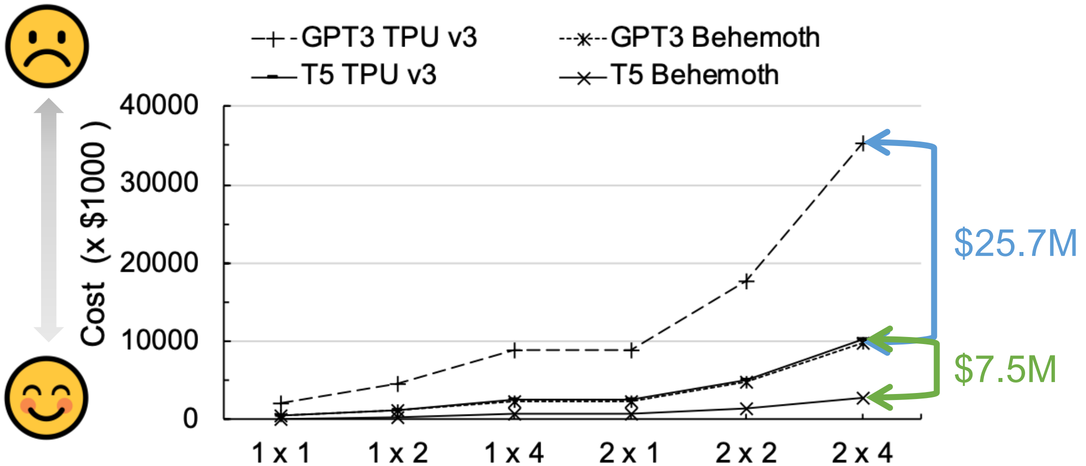


Figure 8: Memory cost comparison between TPU v3 [27] and Behemoth. $W \times D$ in the figure illustrates that the dimension of each layer is increased by W times and the number of layers is increased by D times.

5. Evaluation

Throughput – Storage

Table 5: FMS and conventional storage configuration.

Storage Parameters		
	Behemoth FMS	Baseline SSD x 4 (RAID0)
NAND Configurations	2TB, 64 channels, 2 chips/channel, 1 die/chip	500GB, 16 channels, 2 chips/channel, 1 die/chip
Channel Speed Rate	1200MT/s (MT/s: Mega Transfers per Second [20])	
NAND Structure	128Gb SLC / die: 8 planes / die, 683 blocks / plane, 768 pages / block, 4KB page	
NAND Latency	Read: 3μs, Program: 100μs, Block erase: 5ms	
Buffer Configurations	SRAM 16MB: 6MB for FTL metadata, 10MB for I/O buffer	DRAM 512GB: FTL metadata SRAM 8MB: I/O buffer, GC Buffer
FTL Schemes	Block mapping	Page mapping, Preemtible GC [38]
OP ratio	N/A	7%
Firmware Latency	N/A	Write: 1.45μs / a page (4KB)
Contoller Latency	Read: 1.93μs / an NVMe Cmd, Write: 1.18μs / an NVMe Cmd	Read: 1.93μs / an NVMe Cmd

Bandwidth: 2.75GB/s * 4 = 11.0GB/s

5. Evaluation

Throughput – Storage

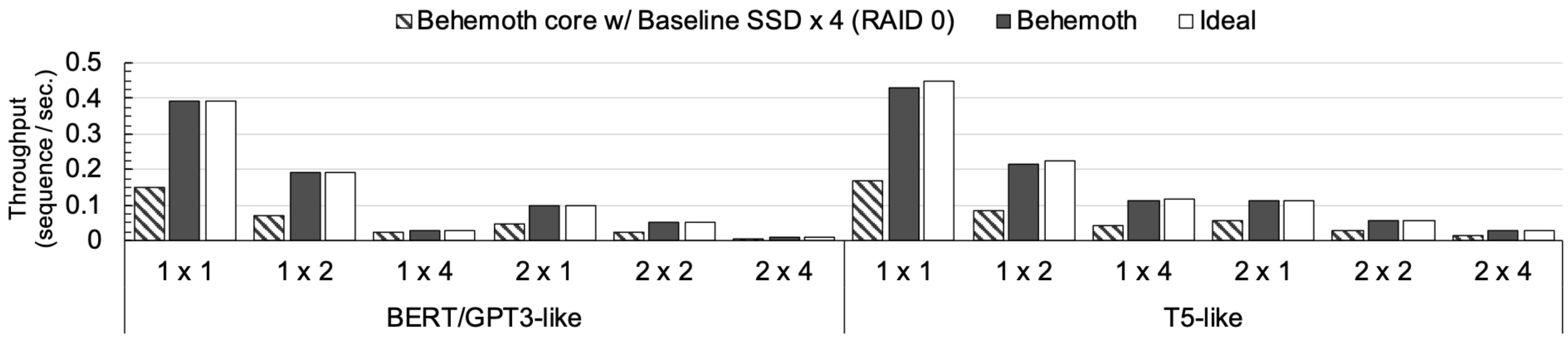


Figure 9: DNN training throughput of 432 Behemoths over various model sizes.

- Behemoth is close to ideal case (zero overhead from memory accesses)
- Baseline SSD achieve limited throughput bottlenecked by SSD firmware
- Lower speedup on wider models in BehemothFMS is because of higher data reuse (less bandwidth)

5. Evaluation

Throughput – Storage

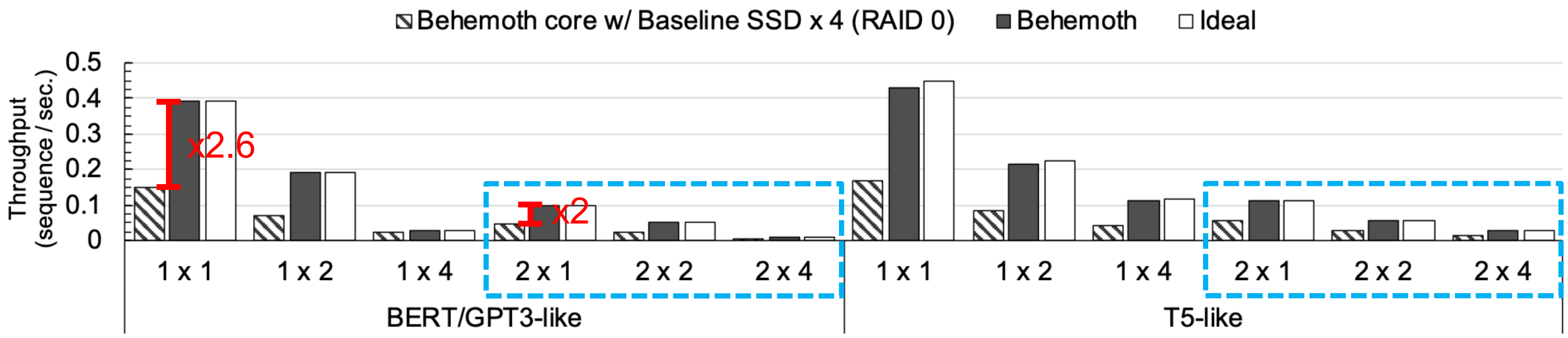


Figure 9: DNN training throughput of 432 Behemoths over various model sizes.

- Behemoth is close to ideal case (zero overhead from memory accesses)
- Baseline SSD achieve limited throughput bottlenecked by SSD firmware
- Lower speedup on wider models in BehemothFMS is because of higher data reuse (less bandwidth)

5. Evaluation

Tensor Lifespan – Storage

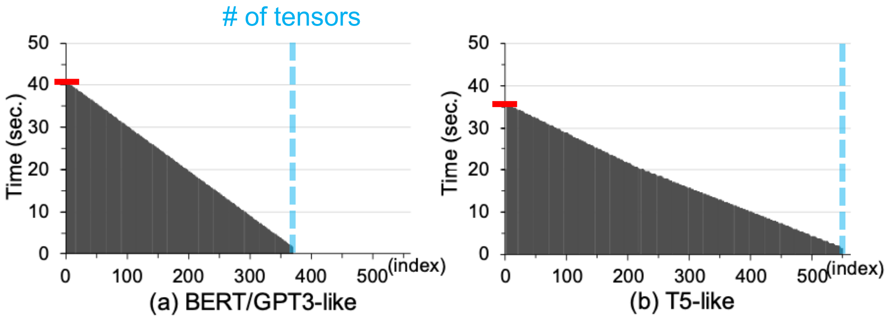


Figure 10: Tensor lifespan

- Longest lifespan of tensors is 41s
- Reducing retention time (1y → 3d) can increase P/E cycle by 40x~
- BehemothFMS guarantee to function 6.6 years with T5-like models
- We also assume that WAF is 1, because there no GC operations

5. Evaluation

Tensor Lifespan – Storage

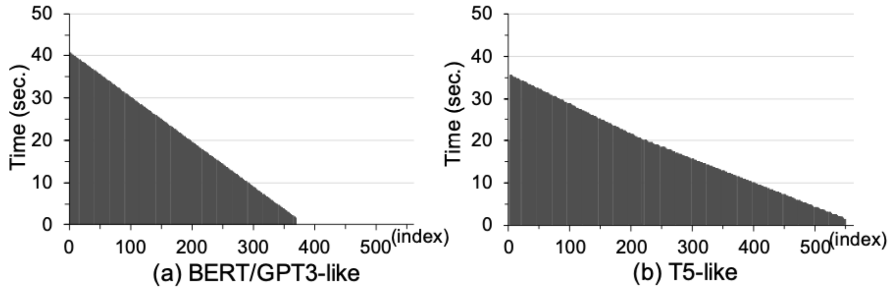


Figure 10: Tensor lifespan

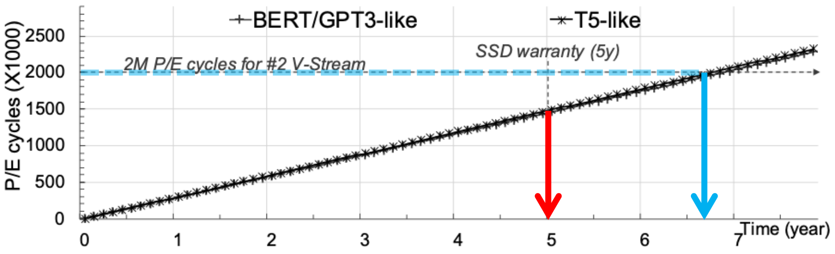


Figure 11: Behemoth FMS endurance

- Longest lifespan of tensors is 41s
- Reducing retention time (1y → 3d) can increase P/E cycle by 40x~
- BehemothFMS guarantee to function 6.6 years with T5-like models > 5y warranty
- We also assume that WAF is 1, because there no GC operations

6. Conclusion

- Recent DNN models require much more memory space for training as NLP grows exponentially. However, conventional DNN training platform(e.g., NVIDIA GPUs or Google TPUs) provide insufficient capacity, which leads to **excessive cost** and **memory bandwidth underutilization**.

- We propose Behemoth, a flash-based memory system for **cost-effective** training platform targeting extreme-scale DNN models. It overcomes low-bandwidth and endurance problem of SSDs by <u>separating data according to their characteristics</u>.

- Behemoth achieve much smaller memory system cost than conventional DNN training platform utilizing HBM devices.

Q&A

Thank you!

2024. 08. 21

Presentation by Nakyeong Kim
nkkim@dankook.ac.kr