19th USENIX Conference on File and Storage Technologies (FAST '21)

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

Shine Kim^{1,2,*} Yunho Jin^{1,*} Gina Sohn¹ Jonghyun Bae¹ Tae Jun Ham¹ Jae W. Lee¹



SAMSUNG

¹ Seoul National University

² Samsung Electronics

*Equal Contributions

Explosive expansion of DNNs

- Deep Neural Networks have become widespread in various application domains
 - Natural language processing, computer vision, recommendation, and so on
- Increasing the model size is crucial to improve accuracy of DNNs
 - Extreme-scale models demand a tremendous amount of computation and memory capacity



DNN training is a repetitive process of matrix operation



- DNN training is a repetitive process of matrix operation
 - Forward path: multiply activation and weights to generate expected value
 - Calculate the difference (loss) between expected value and ground truth
 - Backward path: propagate the loss in backward order and update weights

A: Activation W: Weight G: Gradient



- DNN training is a repetitive process of matrix operation
 - Forward path: multiply activations and weights to generate expected value
 - Calculate the difference (loss) between expected value and ground truth
 - Backward path: propagate the loss in backward order and update weights



- DNN training is a repetitive process of matrix operation
 - Forward path: multiply activations and weights to generate expected value
 - Calculate the difference (loss) between expected value and ground truth
 - Backward path: propagate the loss in backward order and update weights



- DNN training is a repetitive process of matrix operation
 - Forward path: multiply activation and weights to generate expected value
 - Calculate the difference (loss) between expected value and ground truth
 - Backward path: propagate the loss in backward order and update weights



- Memory capacity wall
 - DNN model size exceeds memory capacity of a single GPU
 - Forces users to partition the model and distribute to HBM DRAM on GPU (Model Parallelism)



*figure borrowed from http://jalammar.github.io/how-gpt3-works-visualizations-animations/

- Memory capacity wall
 - DNN model size exceeds memory capacity of a single GPU
 - Forces users to partition the model and distribute to HBM DRAM on GPU (Model Parallelism)



- Memory B/W underutilization
 - As a DNN model (matrix) size increased, each value in the matrix is reused more often
 - The memory B/W requirement does not increase as the computation amount increases



1) Training with batch size of 16 on 840 TFLOPs compute core

- Memory B/W underutilization
 - As a DNN model (matrix) size increased, each value in the matrix is reused more often
 - The memory B/W requirement does not increase as the computation amount increases



50GB/s¹⁾ (required B/W)

Total 28TB/s (HBM B/W)

Significant memory B/W underutilization occurs!

1) Training with batch size of 16 on 840 TFLOPs compute core

Scaling of DNNs necessitates *a new memory system with high-capacity and low-cost* (replacing low-capacity, high-cost HBM)









• Compute Core: compute tensors and transfer data between Tensor Buffer and NANDs



- Compute Core: compute tensors and transfer data between Tensor Buffer and NANDs
- Tensor Buffer: keep tensors in DDR DRAM serving as a staging (prefetching/offloading) area for NANDs



- Compute Core: compute tensors and transfer data between Tensor Buffer and NANDs
- Tensor Buffer: keep tensors in DDR DRAM serving as a staging (prefetching/offloading) area for NANDs
- NANDs: store tensors like HBMs in conventional training system

Behemoth adopts a **two-level** memory architecture using DDR DRAM and NAND flash to reduce the DNN training cost



- Compute Core: compute tensors and transfer data between Tensor Buffer and NANDs
- Tensor Buffer: keep tensors in DDR DRAM serving as a staging (prefetching/offloading) area for NANDs
- NANDs: store tensors like HBMs in conventional training system





• Activation Node: compute and store activations



- Activation Node: compute and store activations
- Weight Node: update and store weights



- Activation Node: compute and store activations
- Weight Node: update and store weights
- Host system: transfer training command sequence to Behemoth



- Activation Node: compute and store activations
- Weight Node: update and store weights



- Activation Node: compute and store activations
- Weight Node: update and store weights



- Activation Node: compute and store activations
- Weight Node: update and store weights



- Activation Node: compute and store activations
- Weight Node: update and store weights



- Activation Node: compute and store activations
- Weight Node: update and store weights



- Activation Node: compute and store activations
- Weight Node: update and store weights



- Activation Node: compute and store activations
- Weight Node: update and store weights



- Activation Node: compute and store activations
- Weight Node: update and store weights

Flash Memory System (FMS) is the main storage in Behemoth to meet the bandwidth and endurance requirements of extreme scale DNN training



Flash Memory System (FMS) is the main storage in Behemoth to meet the bandwidth and endurance requirements of extreme scale DNN training

>50GB/s Read and Write Bandwidth

5-year Endurance

USENIX FAST'21, Behemoth: A Flash-centric Training Accelerator for Extreme-scale DNNs

Improving Bandwidth of FMS

SSD firmware has become a bottleneck for scalable performance



Improving Bandwidth of FMS

- SSD firmware has become a bottleneck for scalable performance
- H/W implemented (automated) data-path can be a solution



Improving Bandwidth of FMS

- SSD firmware has become a bottleneck for scalable performance
- H/W implemented (automated) data-path can be a solution
- Complex functions of FTL make data-path automation difficult
 - Garbage Collection (GC), Wear-leveling (WL), Metadata management for persistency, and so on



FMS separates data types and adopts lightweight FTL to implement H/W automated data path



36
FMS separates data types and adopts lightweight FTL to implement H/W automated data path

#: Stream name			Access pe	ermission
(Act. Node / Weight Node)	Persistency	Retention	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

Multi-stream support for data seperation

FMS separates data types and adopts lightweight FTL to implement H/W automated data path

#: Stream name			Access pe	ermission
(Act. Node / Weight Node)	Persistency	Retention	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

Multi-stream support for data seperation



Strict append-only seq. write for lightweight FTL

Improving Bandwidth of FMS



H/W automated write data path of FMS

(a) write command pipeline: transfers data from TSB to an SRAM buffer in the FMS controller

(b) NAND program pipeline: programs data in the SRAM to NANDs

Improving Endurance of FMS

Endurance of SSD relies on the Program/Erase (P/E) cycles for NAND block



Improving Endurance of FMS

- Endurance of SSD relies on the Program/Erase (P/E) cycles for NAND block
- DNN training workloads cause frequent P/E operation



Behemoth reduces the data retention time and maintains very low WAF (~1)

USENIX FAST'21, Behemoth: A Flash-centric Training Accelerator for Extreme-scale DNNs



42

Behemoth reduces the data retention time and maintains very low WAF (~1)



- Max. data lifespan: 41 sec.
- 1 year retention → 3 days
 - P/E cycle can be increased by at least 40x ^{1, 2)}
 - e.g., 50K P/E cycle \rightarrow 200K P/E cycle

Tensor lifespan for a training iteration of GTP-3 on Behemoth

1) Yu cai et al, ICCD'12, Flash correct-and-refresh: Retention-aware error management for increased flash memory lifetime 2) Ren-Shuo Liu et al, FAST'12, Optimizing NAND flash-based SSDs via retention relaxation



Behemoth reduces the data retention time and maintains very low WAF (~1)

- Only performs monotonic sequential writes and reads
 - No garbage collection \rightarrow WAF 1



Evaluation Methodology

- We evaluate our platform's effectiveness by
 - 1) Comparing the memory cost of Behemoth against the conventional TPU-based DNN training system
 - 2) Comparing the training throughput of FMS against conventional SSDs



Evaluation Methodology

- We evaluate our platform's effectiveness by
 - 1) Comparing the memory cost of Behemoth against the conventional TPU-based DNN training system
 - 2) Comparing the training throughput of FMS against conventional SSDs

¢.	♦ → Ø a gittub.com/CMU-SATAR/MQSm			
MAE&TRO Home GitHub Docs Resources	Why GitHub? ~ Team Em	terprise Explore - Marketplace Pricing -	Search	/ Signin Sig
그는 것은 것이다.	CMU-SAFARI / MQSim			⊗ Wetch 24 🗘 Star
1. 그는 신문은 관련하다. 그는 그 10 P 전 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	O Code 💿 Issues 🛞 🛛 🗅 Pull requests 🛞	🕑 Actions 🔄 Projects 🕕 Security 🖂 Insi	ghts	
MAFOTRO	P master + P t branch 🛇	¢ tags	Go to file 🚊 Code +	About
MAESTRO:	😵 arashta Merge puli request 40	38 from civita/master 🔤 1d	1a217 on May 16, 2020 3147 commits	MQSim is a fast and accurate simulator modeling the performance
An Open-source Infrastructure for	test18	Merge remote-tracking branch 'origin/master'	3 years ago	of modern multi-queue (MQ) SSDs well as traditional SATA based SSDs
An open-source initiastructure for	M MC	Correct minor typos	9 months ago	MQSim faithfully models new high- bandwidth protocol implementation
Modeling Dataflows within Deep	traces	Code refactored	3 years ago	steady-state SSD conditions, and t
viouening Datanows within Deep	D .gitignore	gitignore	2 years ago	full end-to-end latency of request modern SSDs. It is described in de
earning Accelerators	D ucense D M95man	First source commit Windows compilation files	3 years ago	in the FAST 2018 paper by A
carriing Accelerators	D MQSimuciprol	Added a separate OOO scheduler that supports	3 years ago NVMe priorities 9 months ago	Property and cryonuts public D Beating
1988년 1978년 1979년 2월 1988년 1979년 1978년 1	D MQSimvcoproi.fiters	Added a separate OCO scheduler that supports		MIT License
t Storted Workh Demo	MQSimucoprojuser	Default execution arguments in Visual Studio	3 years ago	
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	D Maketile	updated the makefile to compile main.cpp from	arc/ 3 years ago	Releases
YARA YARAH YARAH MURANA KUNA YARA	D READARE.md	Added new option for transaction scheduling	9 months ago	No releases published
승규가에 있어? 집에 가지 않으니 이 것이 좋는 것이?	B sedconfig.xml	Added a separate OOO scheduler that supports		Packages
A BENTLE SUBTRY SPACE AND SP	D workload ami	Added a separate OOO scheduler that supports	NVMe priorities 9 months ago	No anchore published
el sont state est produktion (* 1977) and the Ma	README.red			
지마는 김 사람이 아파 수가 사망했다. 또는 그는 것 같아요. 그 것 같아.	MOGINE A GIN	ulator for Modern NVMe ar		Contributors (2)
	MQSIM: A SIM	ulator for Modern NVMe ar	10 SATA SSDS	🍀 📵 🕸 🕲 🛞 🤅
	Usage in Linux			
rerview	Run following commands:			Languages
	5 make			C++ 88.0%  HTML 10.2% Diter 0.7%
learning techniques, especially convolutional neural networks (CNN), have pervaded vision applications across image classification, face vition, videa processing, and so an due to the high degree of accuracy they provide. Both industry and academia are exploring specialized		guration File> -w <workload definition="" file=""></workload>		
rdware accelerator ASICs as a solution to provide low-latency and high-throughput for CNN workloads.				

NPU Simulator: MAESTRO¹⁾ SSD Simulator: MQ-Sim²⁾

1) https://maestro.ece.gatech.edu/ 2) https://github.com/CMU-SAFARI/MQSim



#### **Evaluation Methodology**

- We evaluate our platform's effectiveness by
  - 1) Comparing the memory cost of Behemoth against the conventional TPU-based DNN training system

**N T 1** 1

2) Comparing the training throughput of FMS against conventional SSDs

SSD Simulator:

MQ-Sim²⁾

ee galethedu S	♦ ⇒ Ø a github.com/CMU-SATAR(MQSIm)			
MAE&TRO Home Github Docs Resources	Why GitHub? ~ Team Enter	prise Explore - Marketplace Pricing -		
	CMU-SAFARI / MQSim			⊕ Mech 24
	O Gode 💿 Issues (1) 11 Pull requests (8) 💿	Actions 🗄 Projects 💿 Security 🖂 Insights		
MAFOTRO	P master + P thranch © 0	tags	Go to file 👲 Code +	About
MAESTRO:	😵 anashta Merge puli request #38 f	from civita/master i= 3d1a2r7 on M	ay 16, 2020 💿 147 commits	MQSim is a fast and accurate simulator modeling the perform
An Open-source Infrastructure for	In fast18	Merge remote-tracking branch 'origin/master'	3 years ago	of modern multi-queue (MQ) 1 well as traditional SATA based
	B 975	Connect minor typos	9 months ago	MQSim faithfully models new bandwidth protocol implement
Modeling Dataflows within Deep	traces	Code refectored	3 years ago	steady-state SSD conditions, full end-to-end latency of reg
	D .gtigsore D Locinski	gitignore First source commit	2 years ago 3 years ago	modern SSDs. It is described in the FAST 2018 paper by A.
Learning Accelerators	D MgSmah	Windows compliation files	3 years ago	P people.int.ethz.ch/omutiaj
YOR 아버지는 FUELOW 관계에서 가장한 것을 다 가지 않는 것이 없다.	D MQSimucoproj	Added a separate OCO scheduler that supports NMMe prior	ties 9 months ago	D Readme
<u> 전문의 입지 방법은 방법에 해야 한다. 또 하는 것</u> 같이 많이 있다.	D MQSimvooprej.fitters	Added a separate QCO scheduler that supports NMMe prior	ties 9 months ago	MIT License MIT License
Get Started Work Demo	MQSimuciprejuser	Default execution arguments in Visual Studio	3 years ago	
~ 영상(영상) / 이상 영상 이상 / 이상 / 이상 이상 / 이상 / 이상 / 이상	D Makefie D Blackel end	updated the makefile to compile main.cpp from src/ Added new option for transaction scheduling	3 years ago 9 months ago	Releases No minutes published
	D sedeenfaunt	Added a securate QCO scheduler that supports NMMe prior		
승규님, 다 그렇는 것은 것 같은 동안에 있는 것이 같이 많다.	D workload.ami	Added a separate OOO scheduler that supports NMMe prior		Packages
에의 그렇는 도로가 봐야 했는 것이 가지 못했다.	README.md			No packages published
	MQSim: A Simul	lator for Modern NVMe and SA	TA SSDs	Contributors 🔋
Overview	Usage in Linux			
Overview	Run following commands:			Languages
Deep learning techniques, especially convolutional neural networks (CNR), have pervoded vision applications across image classification, face	5 make 5 JMPSim -1 +550 Footfoor	ation File> -w -deurkload Definition File>		C++ 08.6% MTML 10.2% Other 0.2%

Model	Size	(GB)	(GB)	PFLOP	
BERT/GPT3-like	1×1	44	350	2.15	
	$1 \times 2$	88	698	4.42	
	$1 \times 4$	175	1393	8.56	
DEI(1/OI 10-IIKC	$2 \times 1$	88	1395	8.56	
	$2 \times 2$	175	2786	17.12	
	$2 \times 4$	349	5569	34.21	
T5-like	1×1	40	305	0.62	
	$1 \times 2$	80	609	1.25	
	$1 \times 4$	160	1218	2.49	
	$2 \times 1$	80	1218	2.49	
	$2 \times 2$	160	2436	4.99	
	$2 \times 4$	319	4871	9.97	

Total act.

**Evaluation workloads** 

https://maestro.ece.gatech.edu/
https://github.com/CMU-SAFARI/MQSim

**NPU Simulator:** 

MAESTRO¹⁾

2) https://github.com	n/CMU-SAFARI/MQSim
<b>Π</b> SENIX EΔST'21	Rehemoth: A Flash-centric Training Accelerator for Extreme-scale DNN

**Total weight** 



Platform configurations for the cost comparison

Memory cost¹⁾ comparison between TPU V3 and Behemoth

• To maintain the same training throughput, TPUv3-like platform costs up to 3.65x the memory cost

1) HBM: \$20/GB, SLC NAND: \$0.67/GB, DDR DRAM: \$4/GB

# **Training Throughput Evaluation**

- We compare Behemoth and baseline system utilizing the commodity SSDs
  - Behemoth with 2TB FMS
  - Behemoth core with 500GB of 4x SSDs (RAID 0)

	Storage Parameters				
	Behemoth FMS	Baseline SSD			
	2ТВ,	500GB,			
NAND	64 channels,	16 channels,			
Configurations	2 chips/channel,	2 chips/channel,			
	1 die/chip	1 die/chip			
Channel	1200	MT/s			
Speed Rate	(MT/s: Mega Transf	ers per Second [20])			
NAND	128Gb SLC / di	e: 8 planes / die,			
Structure	683 blocks / plane, 768 pages / block, 4KB page				
NAND	Read: 3µs, Program: 100µs, Block erase: 5ms				
Latency	Kead. $5\mu$ s, Flogram. IV	$J0\mu s$ , block erase: Jills			
	SRAM 16MB:	DRAM 512GB:			
Buffer	6MB for FTL metadata,	FTL metadata			
Configurations	10MB for I/O buffer	SRAM 8MB:			
	TOWE TO DO DUTE	I/O buffer, GC Buffer			
FTL	Block mapping	Page mapping,			
Schemes	Block mapping	Preemtible GC [38]			
OP ratio	N/A	7%			
Firmware	N/A	Write:			
Latency	11/21	1.45µs / a page (4KB)			
	Read:				
Contoller	1.93µs / an NVMe Cmd,	Read:			
Latency	Write:	1.93µs / an NVMe Cmd			
	1.18µs / an NVMe Cmd				

# **Training Throughput Evaluation**



- Behemoth is close to the ideal case
- Conventional SSDs show much lower training throughput (up-to 2.05x)
  - SSD firmware bottleneck is major cause for performance degradation

# Behemoth enables efficient data-parallel training of extreme-scale DNN models

- Analyze the memory capacity problem for extreme-scale DNN model training
- Identify new opportunities to leverage NAND flash devices to hold those models
- Present a novel flash-centric DNN training accelerator
- Show 3.65x memory cost savings over TPUv3 and 2.05x training throughput impr ovement over conventional SSDs

19th USENIX Conference on File and Storage Technologies (FAST '21)

# Thank you !

# Additional details in the paper:

- Analysis of Transformer: a key enabling primitive for extreme-scale DNNs
- Discussion of architectural decisions
- Coverage analysis for various DNN models
- Endurance evaluation