Research Track: Applied ML and AI for Data Management II

GPEmu: A GPU Emulator for Faster and Cheaper Prototyping and Evaluation of Deep Learning System Research

Meng Wang, Gus Waldspurger, Naufal Ananda, Yuyang Huang, Kemas Wiharja, John Bent, Swaminathan Sundararaman, Vijay Chidambaram, Haryadi S. Gunawi

THE UNIVERSITY OF CHICAGO

Telkom University

Los Alamos NATIONAL LABORATORY

IBM

TEXAS The University of Texas at Austin

DL System Research is Hot

- Deep Learning (DL) is widely used
- Training is time-consuming and inference is latency-sensitive
  - It’s important to optimize deep learning system performance
- Lots of deep learning system research
  - Data loading
  - Preprocessing
  - Job Scheduling
  - Network communication
  - …

But GPUs Are Expensive

- GPUs have limited availability in research cloud
  - GPU machines on *Chameleon cloud* require reservations weeks in advance.
- GPUs on commercial clouds are expensive
  - *Cloudbank* provides up to $5,000 per 6-months for system-focused research
  - Can reserve 4 AWS p3.16xlarge instances *only for 51 hours*

Meng, how is your research progress?

Not bad, I will have access to a GPU machine in 1 week.

… so what’s your plan while you are waiting?

GPU is NOT Always Needed

GPU is NOT NEEDED when…

Working on **the layers above the GPU** to increase GPU utilization

Job Scheduling

Data Preprocessing

Data loading

DL **System** Stack

App1

App2

GPU Emulator

CPU Preprocessing

Storage Systems

Only **GPU Performance** is needed

GPEmu Overview

A GPU Emulator for Faster and **Cheaper** prototyping and evaluation of DL system research.

- ✚ Without GPUs
- ✚ 30+ models and 6 GPUs
- ✚ Easy to use
- ✚ Support both Single node & Distributed setups

DL **System** Stack

App1 | App2

GPEmu

CPU Preprocessing

Storage Systems

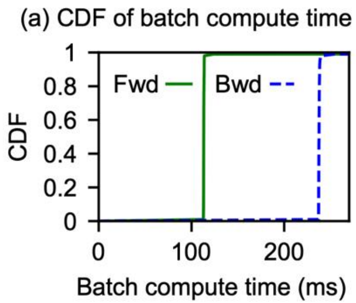
- **Time Emulation**
- **Memory Emulation**
- **Distributed Support**
- **Sharing Support**

- **Effectively reveal** DL system **bottlenecks** during emulations
- **Quickly show** the **benefits** of new system optimizations across a comprehensive spectrum of research areas

GPEmu Features

Time Emulation

Sleep-based approach



Profile its GPU time
$T_{compute}$

Consistent with the same setup

model.compute() → sleep($T_{compute}$)

- GPU Compute Time
- Host-to-GPU Transfer Time
- GPU-driven Preprocessing Time

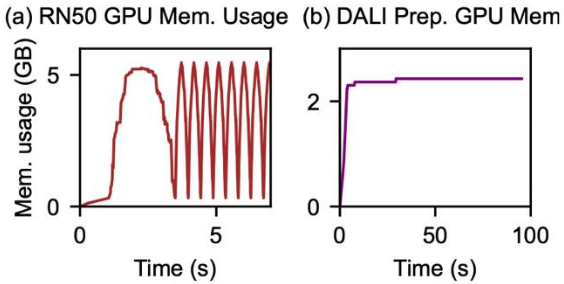
- 30+ models
- 6 GPUs

GPEmu Features

Memory Emulation

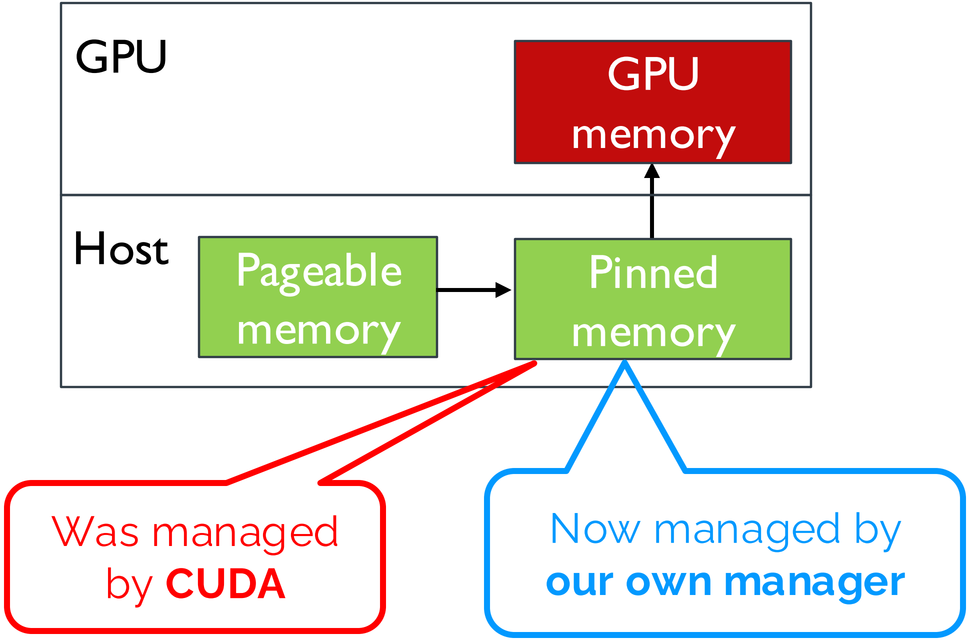
1. Emulate GPU Memory Usage

- By GPU Compute
- By GPU-driven preprocessing



Emulate whether a DL workload can fit into a GPU's memory

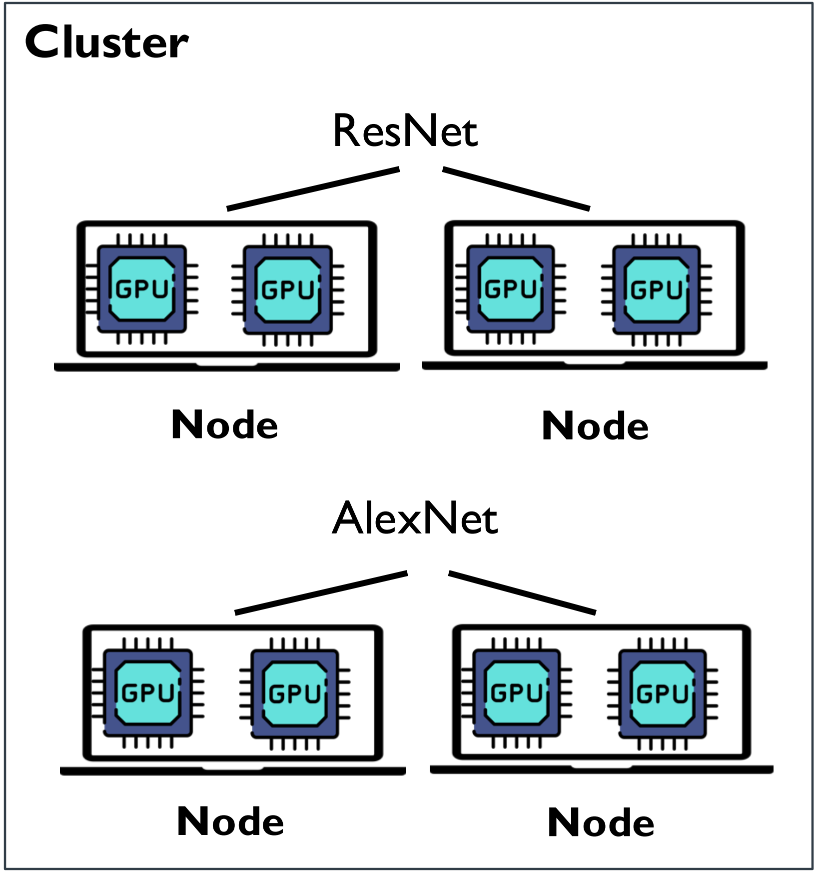
2. Emulate Pinned Memory



GPEmu Features

Distributed Support

- **Multi-GPU** Training
- **Multi-Node** Training
  - Support Distributed data-parallel training
- **Multi-Job** Scheduling
  - emuGPU device plugin
  - Evaluate scheduling jobs wrapped in GPEmu containers

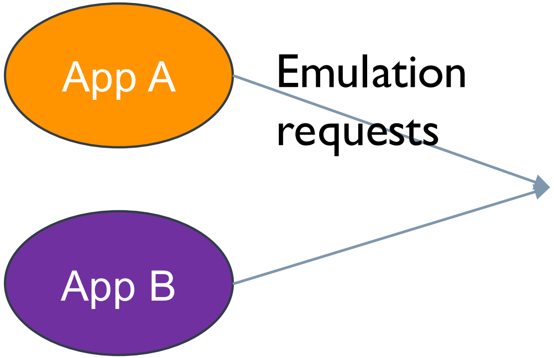


GPEmu Features

Sharing Support

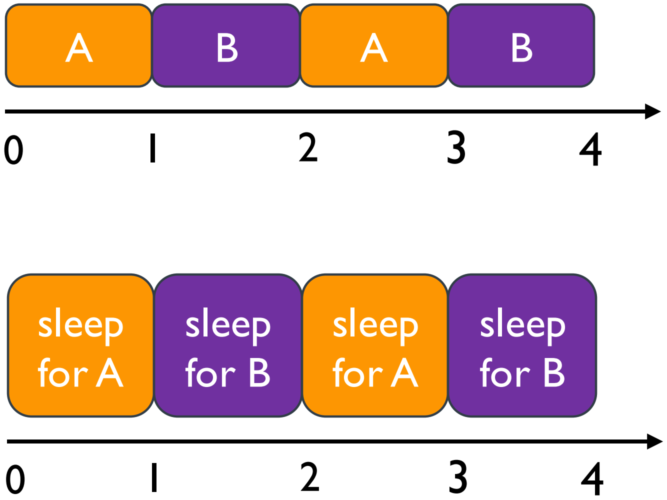
- Support single-node
- Support K8S

GPU Time Sharing



Model on GPU

Our **Emulation Sharing Manager**



Supported Research

- Data stall analysis
- Preprocessing disaggregation
- Data loader optimization
- Distributed training optimization
- GPU scheduling
- GPU sharing

Reproduced 9 papers

DataStall [49], VLDB '21
TF-DS [22], SoCC '23
FastFlow [60], VLDB '23
FFCV [42], CVPR '23
LADL [66], HiPC '19
Synergy [48], OSDI '22
Allox [41], EuroSys '20
Salus [67], MLSys '20
Muri [74], SIGCOMM '22

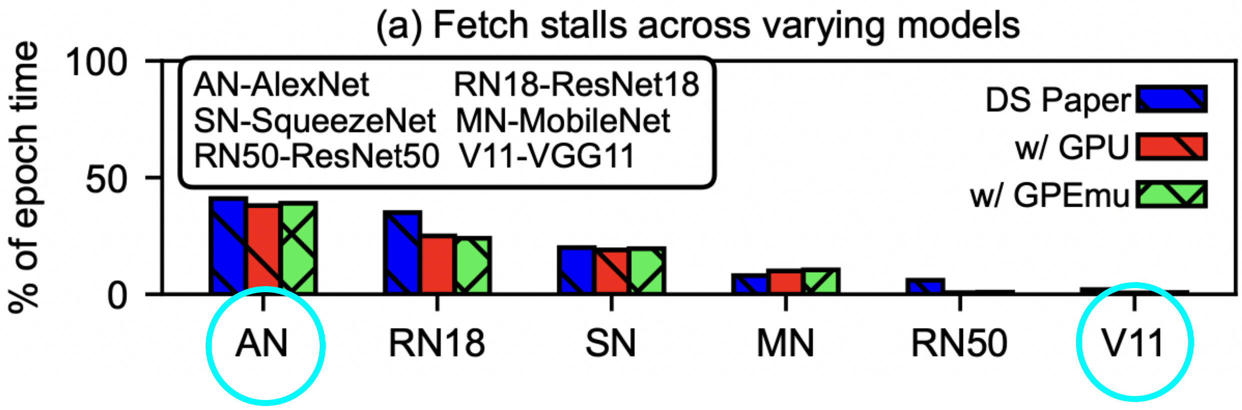
Prototyped 3 new optimizations

Small file first caching
Async data loading
File grouping

Case Study 1: Data Stall Analysis

- DataStall @ VLDB 2022
- Fetch Stall: time spent waiting for data fetching

Pattern is consistent: Different models have different fetch stall percentages

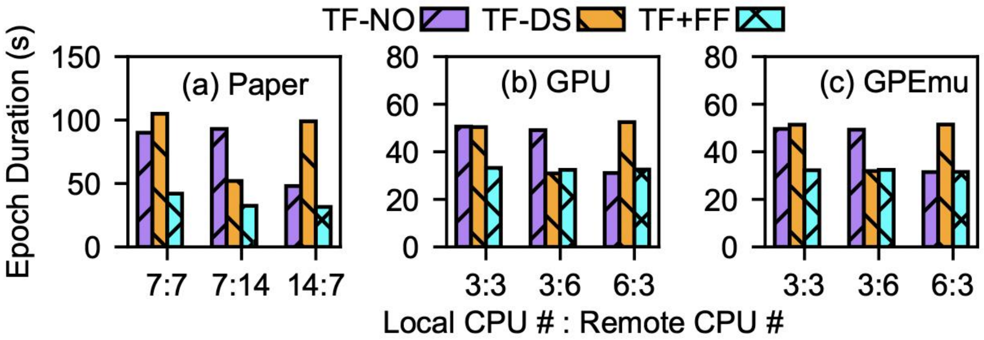


GPEmu values differ from DS paper but **align with GPU** results

Case Study 2: Preprocessing Disaggregation

- FastFlow @ VLDB 2023
- FastFlow: dynamically splits preprocessing pipeline between local and remote CPUs

Pattern is consistent: FastFlow outperforms others in all scenarios.



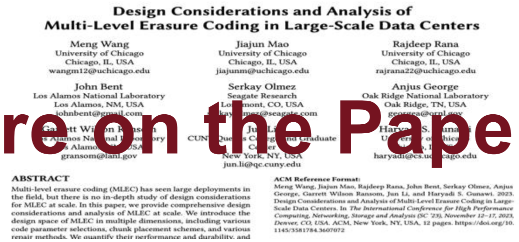
GPEmu values differ from FastFlow paper but **align with GPU** results

Supported Research

- Data stall analysis
- Preprocessing disaggregation
- Data loader optimization
- Distributed training optimization
- GPU scheduling
- GPU sharing

Reproduced 9 papers

DataStall [49], VLDB '21
TF-DS [22], SoCC '23
FastFlow [60], VLDB '23
FFCV [42], CVPR '23
LADL [66], HiPC '19
Synergy [48], OSDI '22
Allox [41], EuroSys '20
Salus [67], MLSys '20
Muri [74], SIGCOMM '22


Design Considerations and Analysis of Multi-Level Erasure Coding in Large-Scale Data Centers

Meng Wang, University of Chicago, Chicago, IL, USA, wangm12@uchicago.edu
Jiajun Mao, University of Chicago, Chicago, IL, USA, jiajunm@uchicago.edu
Rajdeep Rana, University of Chicago, Chicago, IL, USA, rajrana22@uchicago.edu
John Bent, Los Alamos National Laboratory, Los Alamos, NM, USA, johnbent@gmail.com
Serkay Olmez, Seagate Research
Anjus George, Oak Ridge National Laboratory, Oak Ridge, TN, USA

ABSTRACT
Multi-level erasure coding (MLEC) has seen large deployments in the field, but there is no in-depth study of design considerations for MLEC at scale. In this paper, we provide comprehensive design considerations and analysis of MLEC at scale. We introduce the design space of MLEC in multiple dimensions, including various code parameter selections, chunk placement schemes, and various repair methods. We quantify their performance and durability, and

ACM Reference Format:
Meng Wang, Jiajun Mao, Rajdeep Rana, John Bent, Serkay Olmez, Anjus George, Garrett Wilson Ransom, Jun Li, and Haryadi S. Gunawi. 2023. Design Considerations and Analysis of Multi-Level Erasure Coding in Large-Scale Data Centers. In The International Conference for High Performance Computing, Networking, Storage and Analysis (SC '23), November 12–17, 2023, Denver, CO, USA. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3581784.3607072

More on the Paper!

Prototyped 3 new optimizations

Small file first caching
Async data loading
File grouping

Conclusion

A GPU Emulator for Faster and **Cheaper** prototyping and evaluation of DL system research.

- ✚ Without GPUs
- ✚ 30+ models and 6 GPUs
- ✚ Easy to use
- ✚ Support both Single node & Distributed setups

DL **System** Stack

App1 | App2

GPEmu

CPU Preprocessing

Storage Systems

- **Time Emulation**
- **Memory Emulation**
- **Distributed Support**
- **Sharing Support**

Effectively reveal DL system **bottlenecks** and **quickly show** the **benefits** of new system optimizations

Thank you!
Questions?