Scalable deep learning on distributed GPUs with a GPU-specialized parameter server

Henggang Cui

Gregory R. Ganger and Phillip B. Gibbons
PARALLEL DATA LABORATORY
Carnegie Mellon University
Image classification w/ deep learning

Unidentified flying raptor

Eagle? Vulture? Osprey? Accipiter?
Image classification w/ deep learning

Deep neural network: interconnected neurons

Machine learning program

Model parameters: connection weights (solution)

Input training data: images w/ labels
Distributed deep learning

Partitioned input training data → Distributed ML workers → Shared model parameters

Parameter server for GPUs

Eagle, Vulture, Osprey, Accipiter

Read, update
Outline

• Background
  • Deep learning with GPUs
  • Parallel ML using parameter servers
• GeePS: GPU-specialized parameter server
  • Maintaining the parameter cache in GPU memory
  • Batch operations for higher throughput
  • Managing limited GPU device memory
• Experiment results
A machine with GPU device

DRAM (CPU memory)

Local storage

NIC

Network

CPU cores

GPU device

GPU cores

GPU memory (a few GB)
A machine with GPU device

- Large number of small cores
- Single instruction, multiple data
A machine with GPU device

- Small GPU memory
- Expensive to copy between GPU/CPU mem

Local storage

Network

CPU cores

GPU device

GPU memory (a few GB)
Single GPU machine learning

Input data file

- Staging memory for input data batch
- Train in mini-batches
- CPU memory
- GPU memory

Input data
Intermediate states
Parameter data
GPU ML on CPU parameter servers

IterStore: a CPU Param Server
GPU ML on CPU parameter servers

- Staging memory for PS access
- Staging memory for input data batch
- Input data file
- Sub-optimal PS throughput
- Expensive CPU/GPU data movement
- Only works when data fits in GPU memory

- Parameter cache
- Parameter server shard 0
- CPU memory
- Network
- GPU memory

IterStore: a CPU Param Server
Outline

• Background
  • Deep learning with GPUs
  • Parallel ML using parameter servers

• GeePS: GPU-specialized parameter server
  • Maintaining the parameter cache in GPU memory
  • Batch operations for higher throughput
  • Managing limited GPU device memory

• Experiment results
GPU ML on GeePS

GeePS: a GPU Param Server

Input data file

Access PS through GPU memory

Input data
Intermediate states
Parameter working copy
Parameter cache

CPU/GPU copy in the background

CPU memory
GPU memory

Network

Parameter server shard 0

Staging memory for input data batch
Staging memory for parameter cache

Henggang Cui © February 16
http://www.pdl.cmu.edu/
Parallelism w/ batched access

- Applications access data in batches
  - Read_batch
  - Update_batch
- Millions of values in one batch
  - Allow all GPU cores to execute in parallel
Outline

• Background
• GeePS: GPU-specialized parameter server
  • Maintaining the parameter cache in GPU memory
  • Batch operations for higher throughput
  • Managing limited GPU device memory
• Experiment results
Managing limited GPU memory

- Memory usage fluctuation of a neural network

Only 20% of total memory at peak usage
Managing limited GPU memory

Input data file

Staging memory for input data

App has its own buffer

Parameter server shard 0

Parameter working copy

Intermediate states

Parameter cache

Input data

Network

CPU memory

Staging memory for parameter cache

GPU memory
GeePS-managed buffers

- Input data file
- Staging memory for input data batch
- Parameter server shard 0
- Staging memory for parameter cache
- CPU memory
- Network
- GPU memory
- Input data
- Intermediate states
- Access buffer pool
- Parameter cache

GeePS manages all buffers
Interface to GeePS-managed buffer

- **Read**
  - Buffer “allocated” by GeePS
  - Data copied to buffer
- **Post-read**
  - Buffer reclaimed
- **Pre-update**
  - Buffer “allocated” by GeePS
- **Update**
  - Updates applied to data
  - Buffer reclaimed
GeePS manages local data also

- Memory usage fluctuation of a neural network

Intermediate states consume most memory
GeePS manages local data also

- Input data file
- Staging memory for input data batch
- Parameter server shard 0
- Staging memory for parameter cache
- CPU memory
- Network
- GPU memory
- Input data
- Intermediate states
  - Access buffer pool
  - Parameter cache

http://www.pdl.cmu.edu/
GeePS manages local data also

Input data file

Staging memory for input data batch

Parameter server shard 0

Staging memory for parameter cache

CPU memory

Network

GPU memory

Parameter cache

Local data

Access buffer pool

GeePS manages local data also
Swapping data to CPU memory

- Input data file
- Staging memory for input data batch
- Local data (CPU part)
- Parameter cache (CPU part)
- Parameter server shard 0
- Staging memory for parameter cache
- Pinned local data
- Pinned param cache
- Access buffer pool
- Network

CPU memory

GPU memory

Henggang Cui © February 16
http://www.pdl.cmu.edu/
Data placement policy

- The set of data entries used by the app is known
- Can gather such info from iterative ML apps [iterstore]
- Access buffers need to be in GPU memory
- Must be big enough for peak memory usage
- Local data kept in GPU memory needs no access buffers
Data placement policy

- Pin as much local data as can in GPU memory
- Select local data that causes peak usage
Data placement policy

Pin as much local data as can in GPU memory
Select local data that causes peak usage
Data placement policy

- Keep moving local data to GPU memory
Data placement policy

- Keep moving local data to GPU memory
- If still have space, put parameter data
Data movement in background

- Read/update with two background threads
  - Allocator for read and pre-update
  - Reclaimer for post-read and update
- Overlap data movement with computation
Outline

• Background

• GeePS: GPU-specialized parameter server
  • Maintaining the parameter cache in GPU memory
  • Batch operations for higher throughput
  • Managing limited GPU device memory

• Experiment results
Experimental setups

• Cluster information
  • Each machine has
    – one Tesla K20C GPU with 5 GB GPU memory
  • Machines connected with 40 Gbps Ethernet

• Neural network implementation
  • Caffe: single-machine GPU deep learning system
  • Use GeePS to store param and local data

• Image classification dataset
  • ImageNet: 14 million images with 22,000 classes

• Neural network size
  • 25 layers, 2.4 billion connections
System baselines

- Single GPU worker
  - The original unmodified Caffe
- GPU workers with CPU-based parameter server
  - Caffe linked with IterStore
- CPU workers with CPU-based parameter server
  - ProjectAdam (reported numbers from OSDI’14)
Training throughput

Initial overhead for keeping data in GeePS
Training throughput

- GeePS scales close to linear with more machines
- With 16 machines, GeePS runs 9.5x faster than Caffe
70% more throughput than IterStore
Training throughput

- GeePS with 4 machines close to ProjectAdam with 108
Image classification accuracy

To reach 5% classification accuracy:
- Caffe needs 12.2 hours
- GeePS needs only 3.3 hours (3.7x speedup)
Image classification accuracy

- ProjectAdam with 58 machines reach 13.6% in one day
- GeePS with 8 machines spends only 12.6 hours
Managing GPU memory

- Memory usage fluctuation of the neural network
  - Same figure as shown before
Throughput vs. memory budget

- 88% of the original throughput with 38% memory usage
- Can do 2.5x bigger problems with little overhead
Conclusion

• GPU-specialized parameter server for GPU ML
  • 10x throughput speedup using 16 machines
  • About 2x faster compared to CPU-based PS
• Managing limited GPU memory
  • By managing all GPU memory inside GeePS
  • Swap data to CPU memory if necessary
  • Can handle 2.5x larger problems with low overhead

→ Enable use of data-parallel PS model
References


Additional related works