In-Depth Analysis of HOGWILD SGD on GPU
Yujing Ma, Martin Torres, Florin Rusu
University of California Merced

Motivation & Goal
- As datasets and models grow larger and more complex, we want to leverage GPU-accelerated computing in machine learning training
- GPU architecture introduces many unexplored design considerations for implementing the HOGWILD asynchronous stochastic gradient descent (SGD) algorithm
- Number hierarchy
- Tradeoffs examined from the CPU problem space of hardware and statistical efficiency do not necessarily translate to the GPU
- HOGWILD may suffer from greater cache-coherency issues with the larger number of cores the GPU has to offer compared to the CPU
- We study variations of HOGWILD implemented on the GPU and examine the effects on convergence with respect to time (hardware efficiency) and number of iterations (statistical efficiency)

Gradient Descent Optimization
- Generalized linear model training with SGD
  \[ \hat{\alpha} = \arg \min_{\alpha} \sum_{i=1}^{N} f(\alpha; x_i, y_i) \]

GPU Architecture
- Number of cores: 2496
- Memory size: 12 GB
- Threads per warp: 32
- Max threads per block: 1024
- Max shared memory per block: 48 KB

Parallel SGD: HOGWILD

Algorithm 1 HOGWILD
1. for \( i = 1 \) to \( N \) do in parallel
2. \( \hat{w} \leftarrow \hat{w} - \alpha (\nabla f(\hat{w}; x_i, y_i)) \)

Access Methods
- Row-storage
- Column-storage

Model Replication
- PerMachine / PerCore
- PerNode

Data Replication
- Sharding + Round-robin
- Sharding + Chunking

Datasets

<table>
<thead>
<tr>
<th>Dataset Name</th>
<th>Examples</th>
<th>Features</th>
<th>NNZ/Row</th>
</tr>
</thead>
<tbody>
<tr>
<td>forest</td>
<td>581,012</td>
<td>54</td>
<td>1 to 16,423</td>
</tr>
<tr>
<td>news</td>
<td>19,996</td>
<td>1,355,191</td>
<td>54 to 54</td>
</tr>
</tbody>
</table>

Experiments
- forest

Data Replication
- Full replication + Round-robin
- Full replication + Chunking