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
  - Memory hierarchy
  - Number of cores available
- 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)

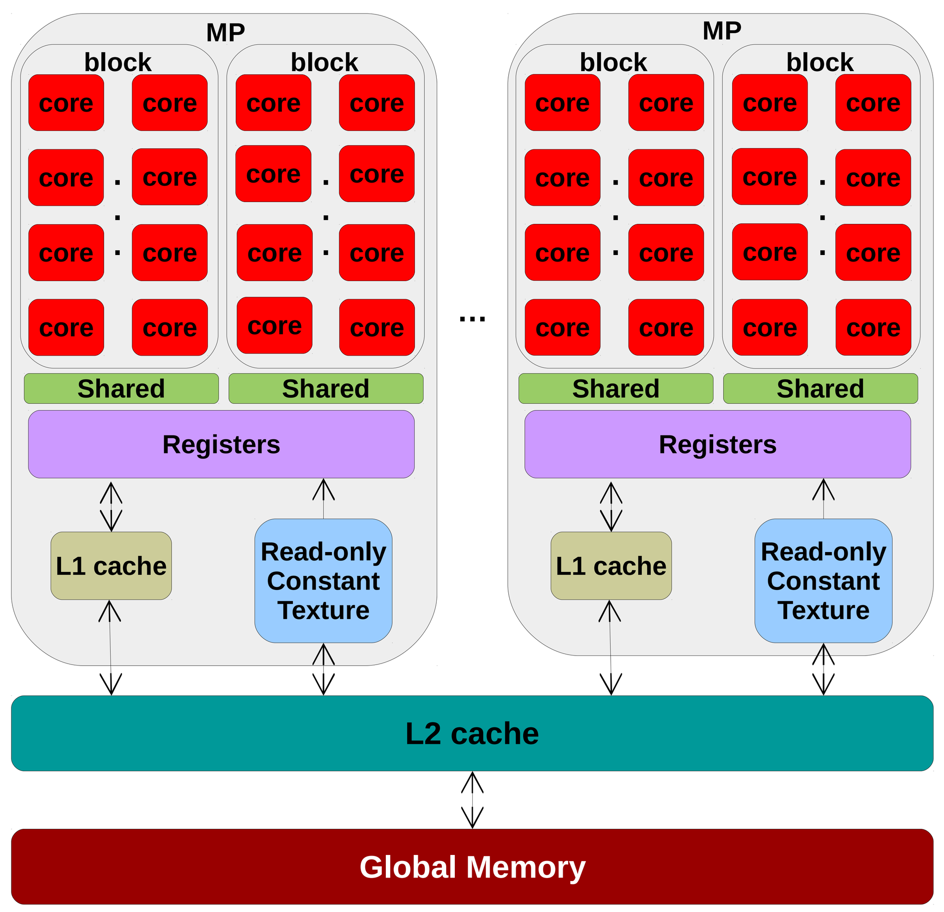
Gradient Descent Optimization

- Generalized linear model training with SGD

$$\Lambda(\vec{w}) = min_{w \in \mathbb{R}^d} \sum_{i=1}^{N} f(\vec{w}, \vec{x}_i; y_i)$$

$$\vec{w}^{(k+1)} = \vec{w}^{(k)} - \alpha^{(k)} \nabla f\left(\vec{w}, \vec{x}_{\eta(i)}; y_{\eta(i)}\right)$$

GPU Architecture



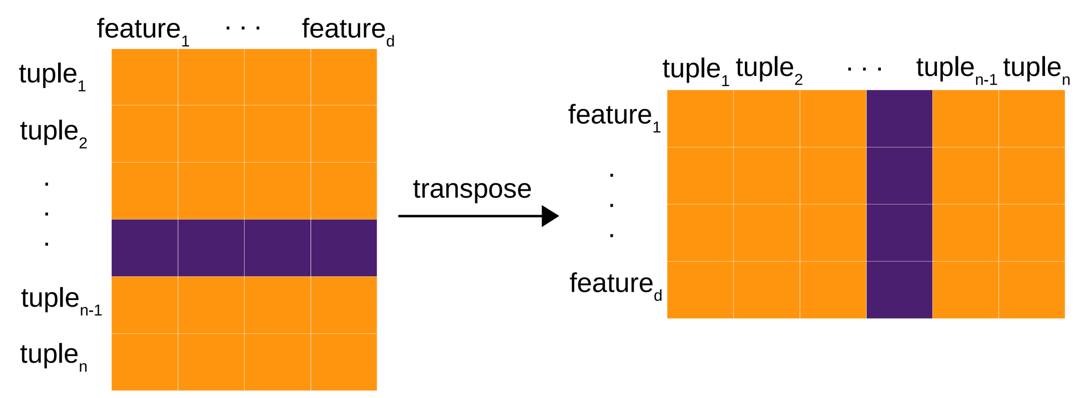
Number of cores	2496
Memory size	12 GB
Threads per warp	32
Max threads per block	1024
Max shared memory per block	48 KB

Parallel SGD: HOGWILD

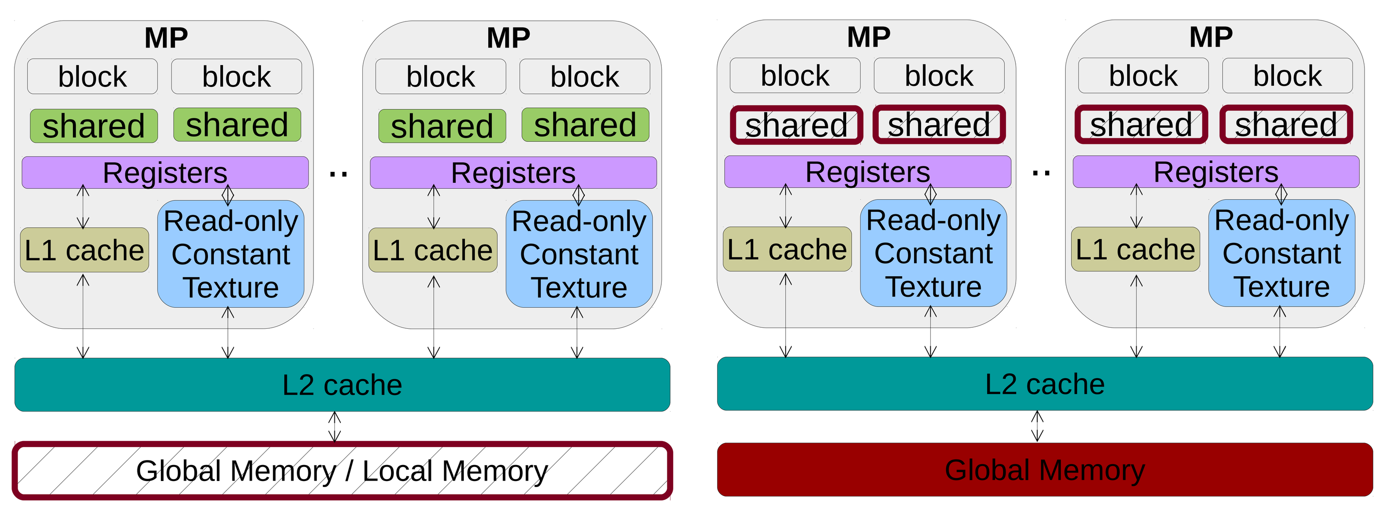
Algorithm 1 HOGWILD

1. **for** $i = 1$ **to** N **do** ***in parallel***
2. $\vec{w} \leftarrow \vec{w} - \alpha^{(k)} \nabla f\left(\vec{w}, \vec{x}_{\eta(i)}; y_{\eta(i)}\right)$

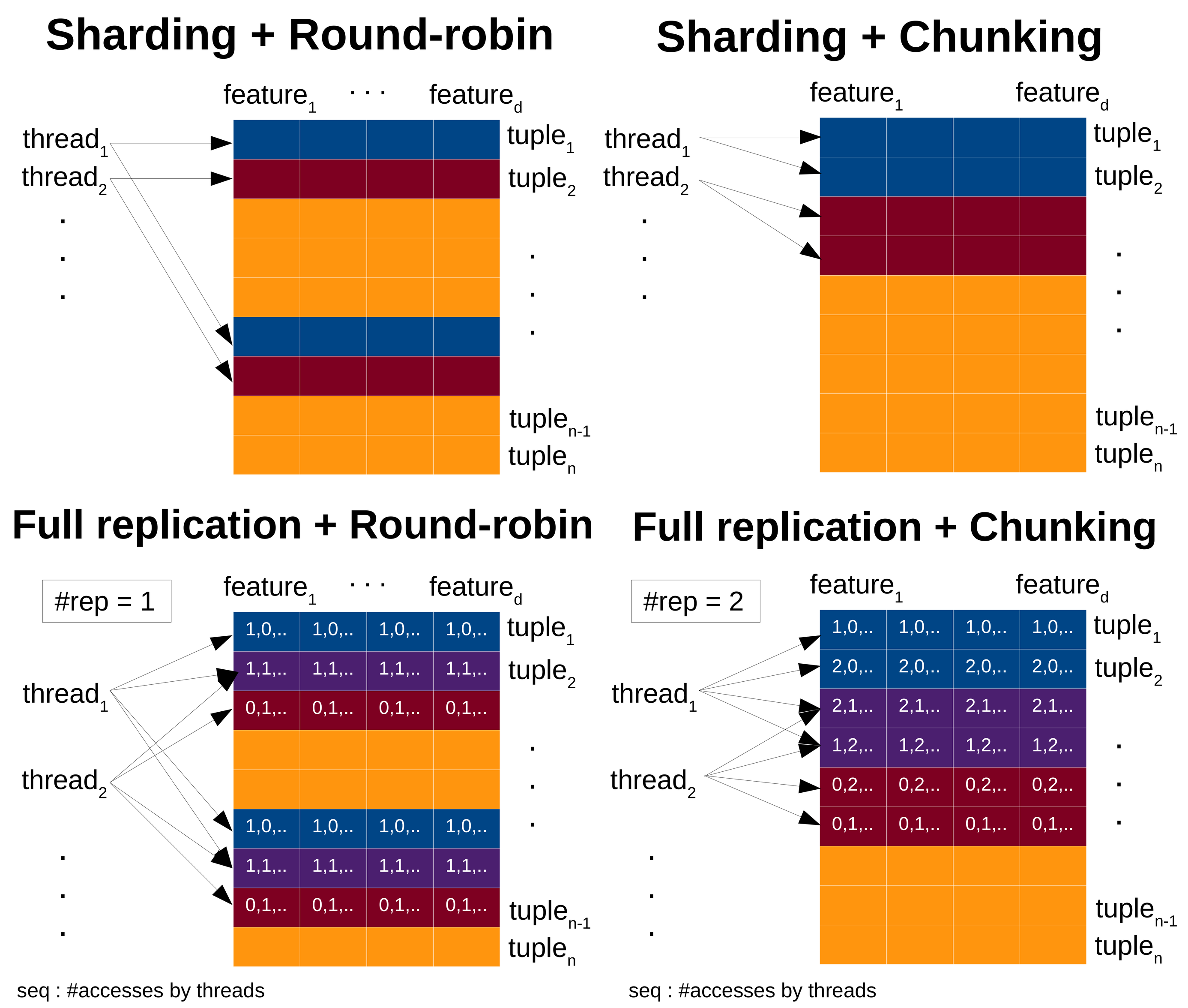
Access Methods



Model Replication



Data Replication

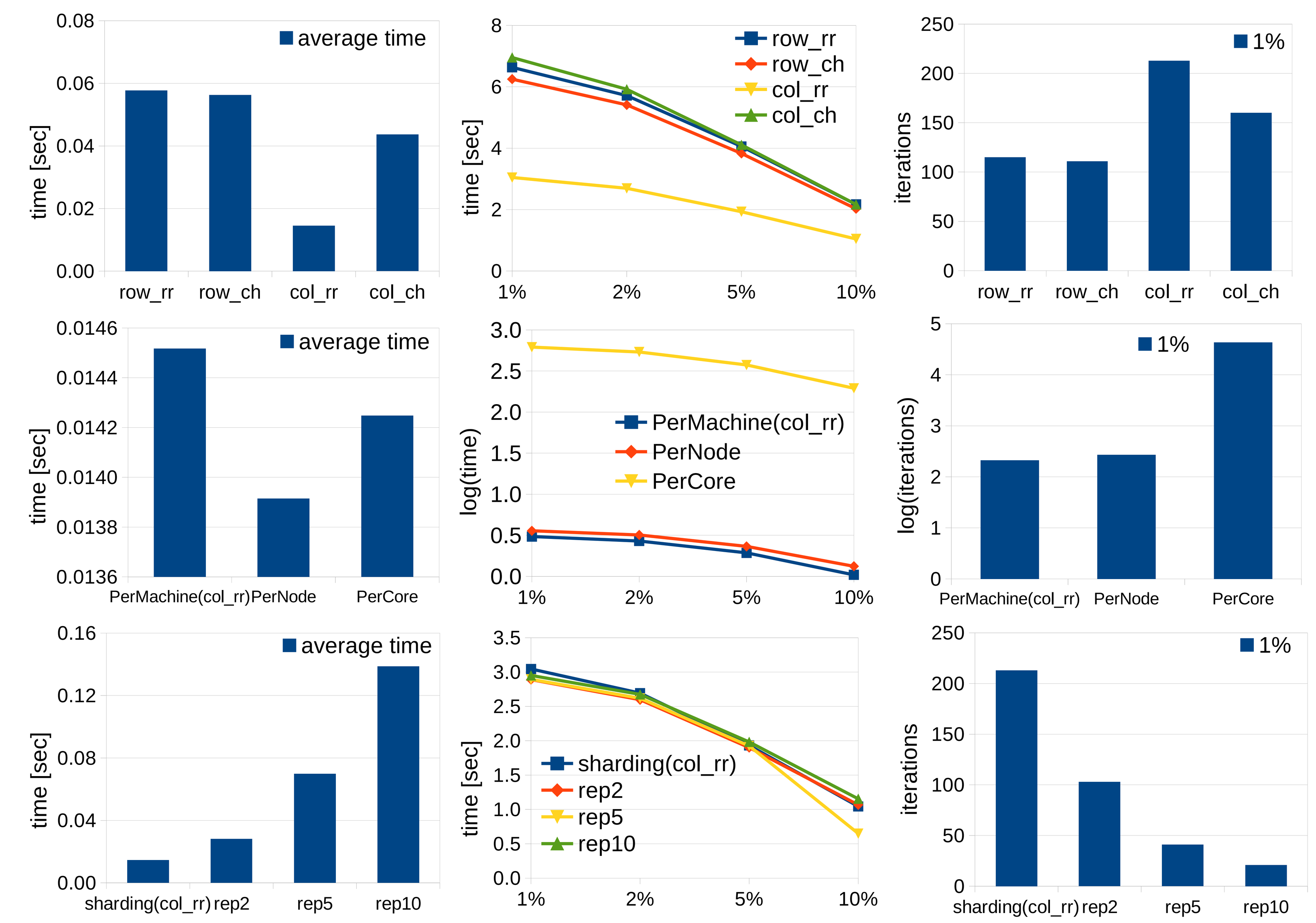


Datasets

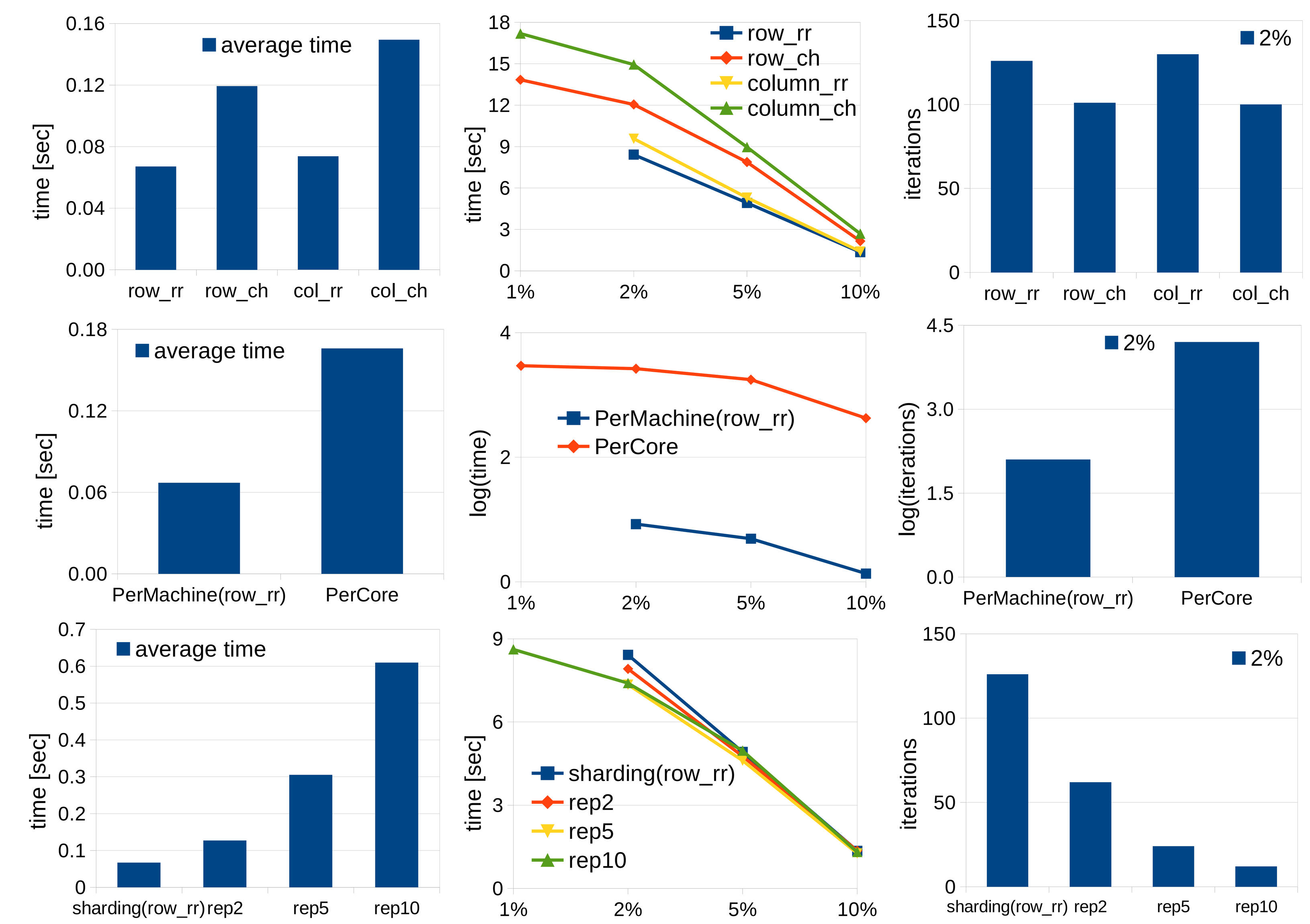
Dataset Name	Examples	Features	NNZ/Row
forest	581,012	54	54 to 54
news	19,996	1,355,191	1 to 16,423

Experiments

- forest



- news



{yma33, mtorres58, frusu}@ucmerced.edu