SySCD: A System-Aware Parallel Coordinate Descent Algorithm
Nikolas Ioannou*, Celestine Mendler-Dünnen* †, Thomas Parnell†

Contributions
✓ Identify bottlenecks in state-of-the-art parallel coordinate descent solvers
✓ Algorithmic improvements to reduce runtime
✓ Over 10x faster training compared to optimized system-agnostic parallel implementations
✓ SySCD inherits convergence guarantees from distributed methods and improves their sample efficiency through a dynamic repartitioning scheme

Baseline
min_α f(α) + \sum_{i \in [n]} g_i(α_i)

Algorithm 1 Parallel SDCA for training GLM
1: Input: Training data matrix A = [x_1, ..., x_n] ∈ R^n×d
2: Initialize model α and shared vector \psi = \{\alpha_1, \psi_1, ..., \alpha_n, \psi_n\}
3: for t = 1, 2, ..., Nepochs do
4: \textbf{parfor} j \in \text{random permutation}(n) do
5: \textbf{if} j \neq \text{random permutation}(n) \textbf{then}
6: Read current state of model \alpha_j = \text{READ}(\alpha_j)
7: Read current state of shared vector \psi = \text{READ}(\psi)
8: \textbf{end if}
9: \textbf{end parfor}
10: Update \alpha_i \leftarrow \text{UPDATE}(\alpha_i)
11: end for

in parallel
across threads.

Dynamic Partitioning
- Increase exchange between parallel workers
- Better convergence behavior
- Shuffling of coordinates not local to each thread

Convergence Analysis
SySCD can be analyzed as a hierarchical distributed method which locally implements a randomized block coordinate descent solver

Data Parallelism
\- Reduce shared vector \psi access
\- Parallel shuffling of local coordinates
\- Local-only view hurts convergence

Bottlenecks
1) model access pattern (α)
2) random shuffling of coordinates
3) shared vector updates \psi

Baseline
Train time per epoch (s)

Performance Results
SySCD is implemented in Snap ML [3] and results in average speeds of 5x vs. baseline 18x vs. sklearn