Scalable HOGWILD! for Big Models

Chengjie Qin, Martin Torres, and Florin Rusu
University of California Merced

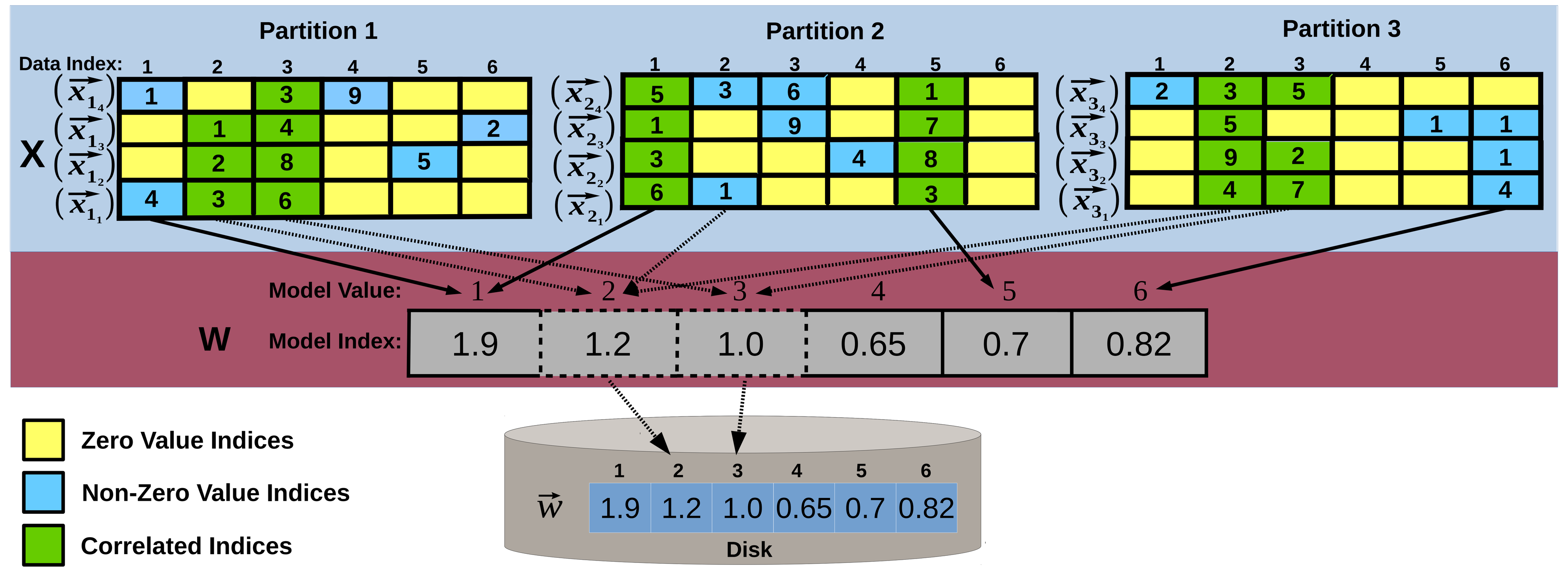
Background & Motivation

Example: Gradient in logistic regression where $\vec{x}_i$ is the training example; $\vec{w}$ is the model.

$$\sum_i \frac{e^{-y_i \vec{w}\cdot\vec{x}_i}}{1+e^{-y_i \vec{w}\cdot\vec{x}_i}}\left(-y_i \cdot \vec{x}_i\right)$$

Model size grows beyond the memory of a single machine.

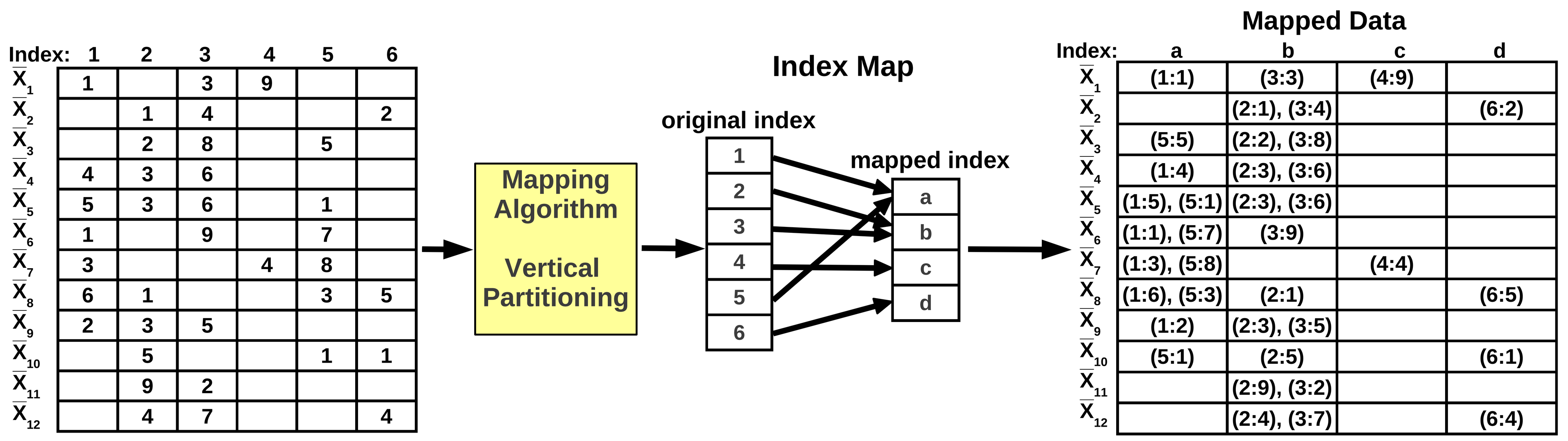
Scalable HOGWILD! Framework



Immediate Extension of Big Model HOGWILD!

for $i = 1$ **to** N **do** ***in parallel***
  for each non-zero feature $j \in \{1,\ldots,d\}$ in $\vec{x}_{\eta^{(i)}}$ **do**
    `get` $\vec{w}^{(k)}[j]$
    compute $\nabla\Lambda\left(\vec{w}^{(k)}[j], \vec{x}_{\eta^{(i)}}; y_{\eta^{(i)}}\right)$
  end for
  $\vec{w}^{(k+1)} \leftarrow \vec{w}^{(k)} - \alpha^{(k)}\nabla\Lambda\left(\vec{w}^{(k)}, \vec{x}_{\eta^{(i)}}; y_{\eta^{(i)}}\right)$
  for each non-zero feature $j \in \{1,\ldots,d\}$ in $\vec{x}_{\eta^{(i)}}$ **do**
    `put` $\vec{w}^{(k+1)}[j]$
  end for
end for

Offline Model Vertical Partitioning



Model Vertical Partitioning Algorithm

Require:
Model index set $I = \{1, 2, \ldots, d\}$
Sparse vector set $X = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\}$
Cost function $cost(s)$ for partition with size s
Ensure: Model partitions $P = \{P_1, P_2, \ldots\}$
for each index $i \in I$ **do** $P_i \leftarrow \{i\}$

Main loop

while true **do**

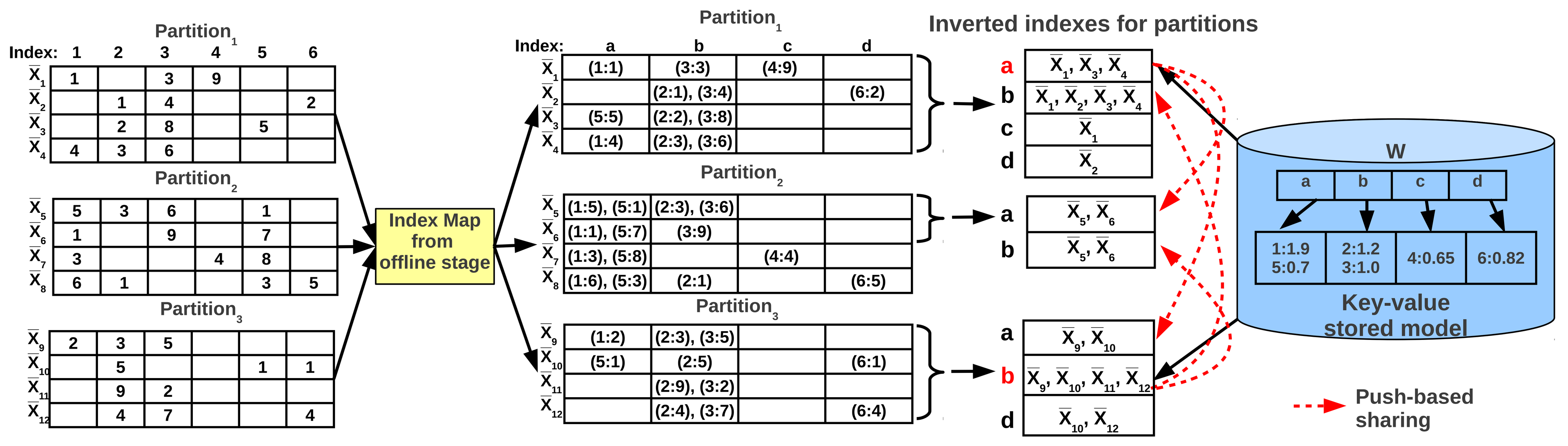
Compute affinity matrix AF for partitions in P

for each vector $\vec{x}_i \in X$ **do**
Collect partitions accessed by $\vec{x}_i$ in $P_{\vec{x}_i}$
Compute affinity $AF[i][j]$ for all pairs (P_i, P_j) in $P_{\vec{x}_i}$
end for
for each partition $P_i \in P$ **do**
Compute cost $C_i \leftarrow AF[i][i] \cdot cost(|P_i|)$

Select the best pair of partitions to merge

for each pair (P_i, P_j) in P **do**
Compute cost C_{ij} for partition $P_{ij} = P_i \cup P_j$:
$C_{ij} = (AF[i][i] + AF[j][j] - AF[i][j]) \cdot cost(|P_{ij}|)$
Compute reduction in cost if P_i and P_j are merged:
$\Delta_{ij} = C_i + C_j - C_{ij}$
end for
Pick the pair $(P_{i'}, P_{j'})$ with largest reduction in cost $\Delta_{i'j'}$
if $\Delta_{i'j'} = 0$ **return** partition P
else Merge $P_{i'}$ and $P_{j'}$
end while

Online Model and Data-Parallel Asynchronous Training



Experimental Evaluation

Dataset	# Dims	# Examples	Size	Model
`uniform`	1B	80K	4.2 GB	12 GB
`skewed`	1B	1M	4.5 GB	12 GB
`splice`	13M	500K	30 GB	156 MB
`MovieLens`	6K x 4K	1M	24 MB	120 MB

Table : Datasets used in the experiments.

Table : Vertical partitioning time (in seconds) as a function of the pruning ratio K.

K	`skewed`	`uniform`	`splice`	`MovieLens`
100	722	79	79	55
500	2028	83	1404	181
1000	3054	79	4075	600

Table : Execution time per iteration (in seconds) as a function of the pruning ratio K.

K	`skewed`	`uniform`	`splice`	`MovieLens`
100	244	1055	923	356
500	236	1055	894	350
1000	234	1055	897	373

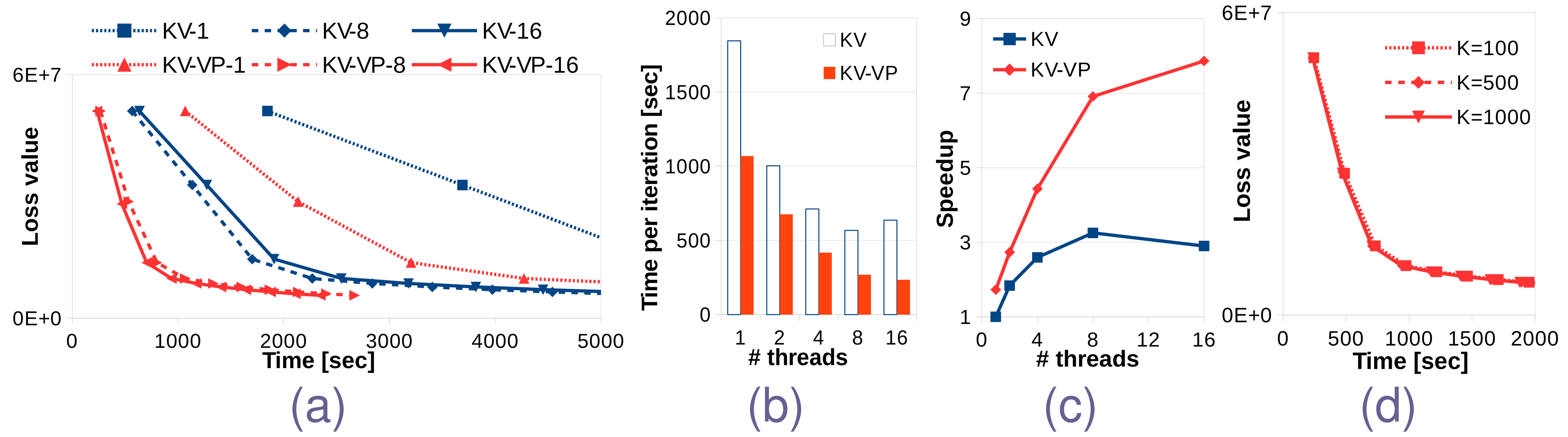


Figure : SVM on `skewed`. (a) Convergence over time. (b) Speedup over serial KV. (c) Time per iteration. (d) Convergence over K.

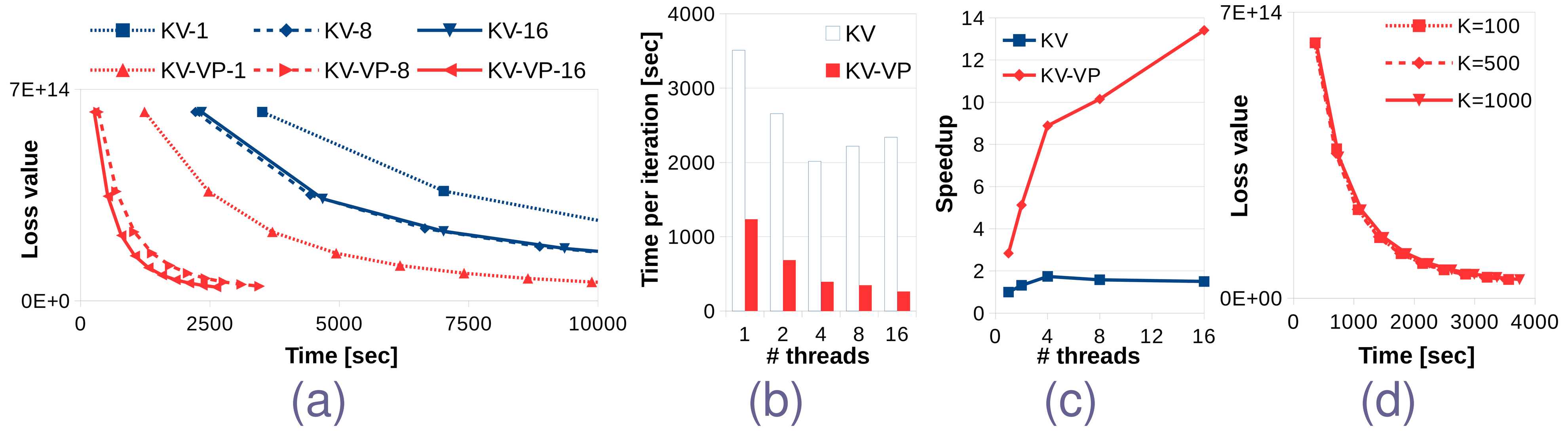


Figure : LMF on `MovieLens`. (a) Convergence over time. (b) Speedup over serial KV. (c) Time per iteration. (d) Convergence over K.

Email: {cqin3, mtorres58, frusu}@ucmerced.edu