Orpheus: Efficient Distributed Machine Learning via System and Algorithm Co-design

Pengtao Xie (Petuum Inc)

Jin Kyu Kim (CMU), Qirong Ho (Petuum Inc), Yaoliang Yu (University of waterloo), Eric P. Xing (Petuum Inc)

Massive Data

1B+ USERS
30+ PETABYTES

32 million pages

100+ hours video uploaded every minute

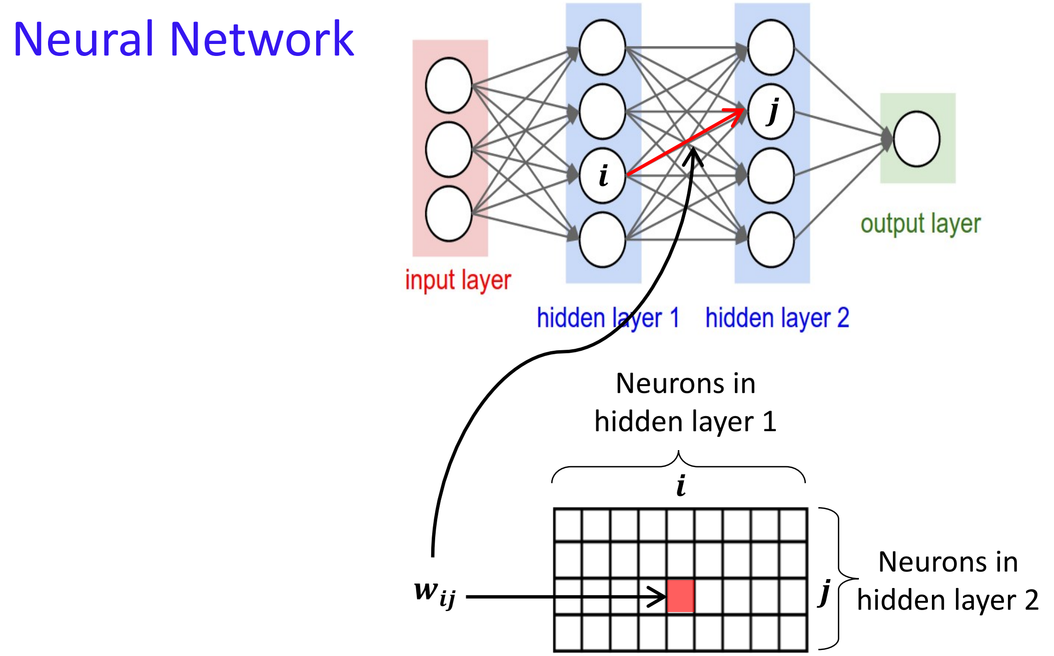
645 million users
500 million tweets / day

Distributed ML Systems

Parameter Server Systems	Yahoo LDA, DistBelief, Li & Smola PS, Project Adam, Bosen, GeePS
Graph Processing Systems	GraphLab, Pregel, GraphX
Dataflow Systems	Spark, hadoop
Hybrid Systems	TensorFlow™, mxnet

Matrix-Parameterized Models (MPMs)

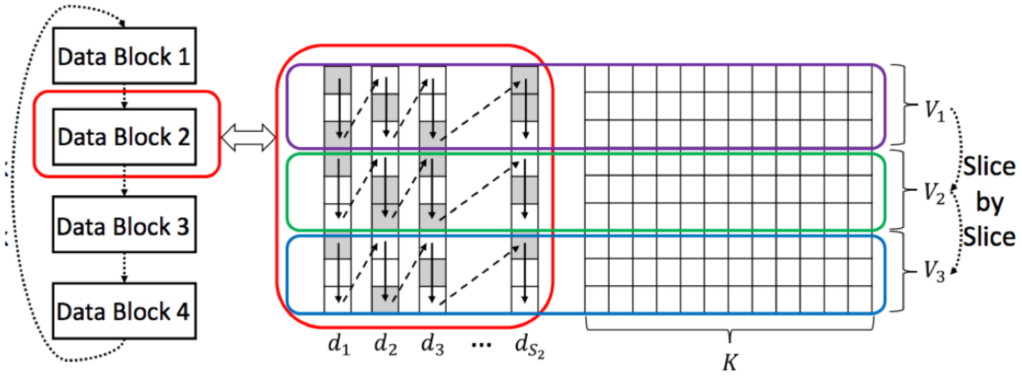
- Model parameters are represented as a matrix



- Other examples: Topic Model, Multiclass Logistic Regression, Distance Metric Learning, Sparse Coding, Group Lasso, etc.

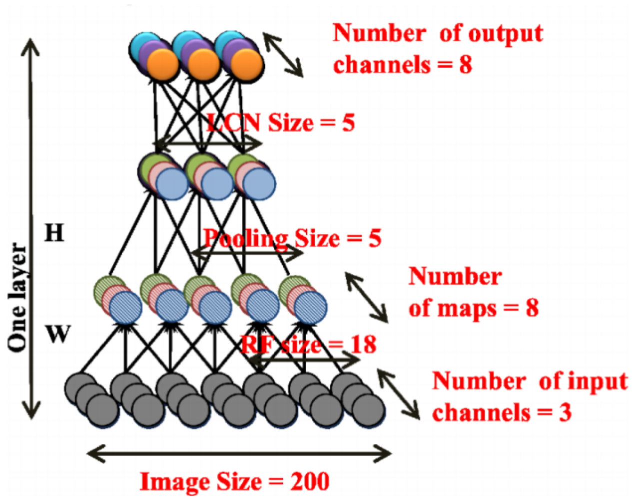
Parameter Matrices Could Be Very Large

LightLDA Topic Model
(Yuan et al. 2015)



The topic matrix has **50 billion** entries.

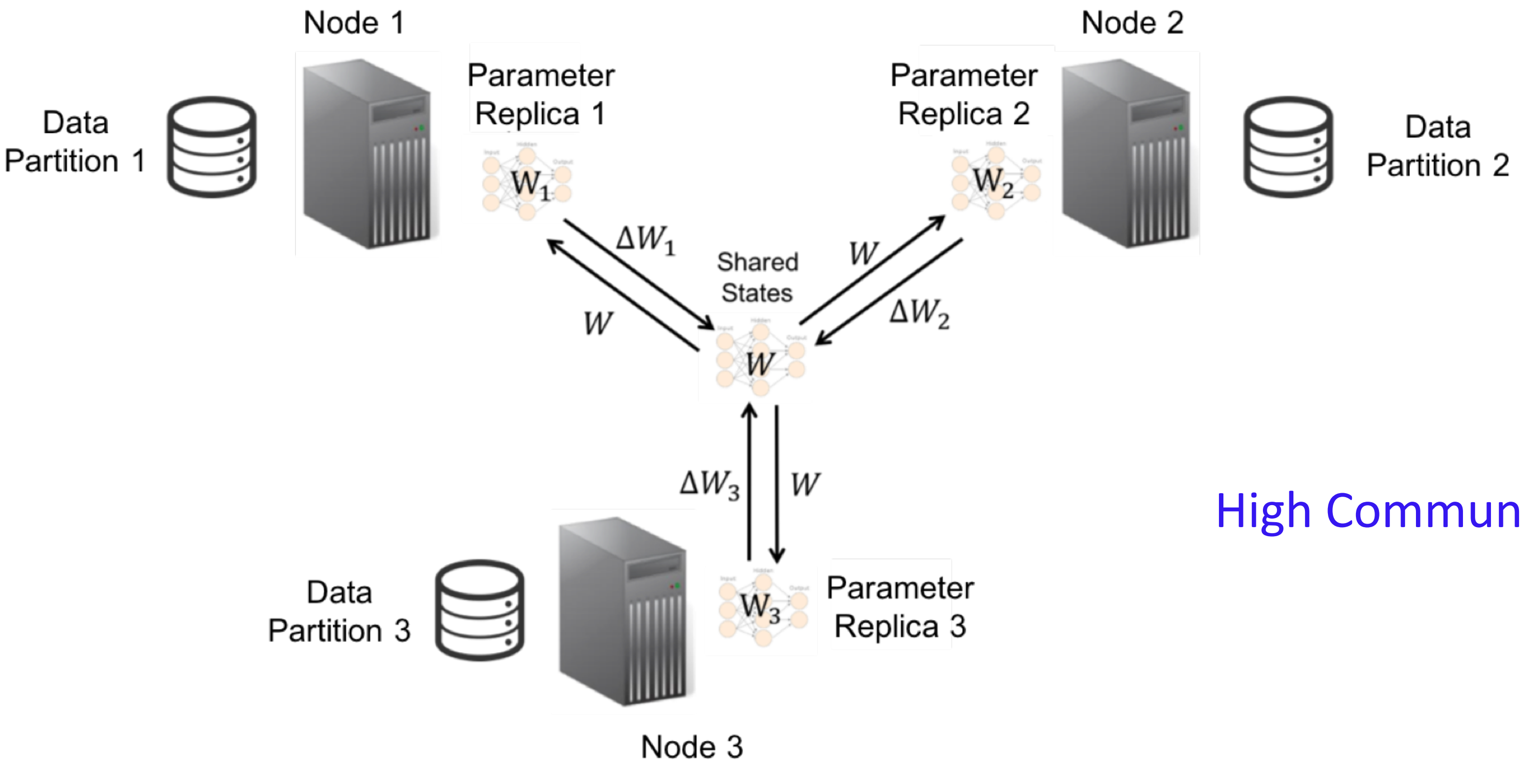
Google Brain Neural Network
(Le et al. 2012)



The weight matrices have **1.3 billion** entries.

Existing Approaches

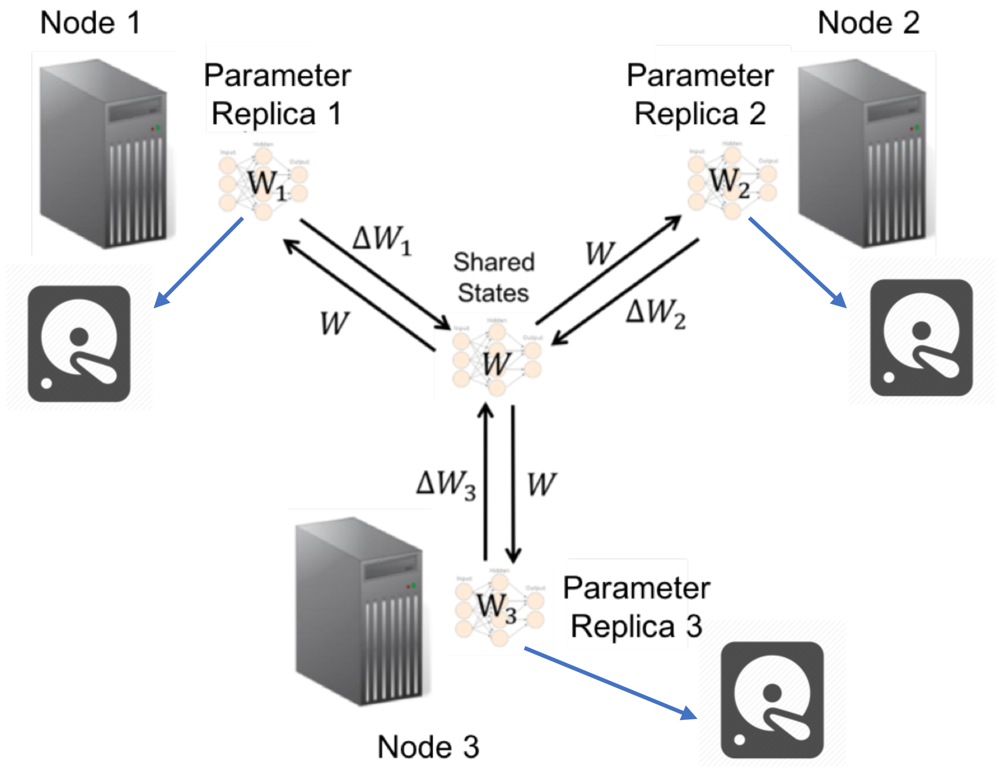
- Parameter server frameworks communicate matrices for parameter synchronization.



High Communication Cost

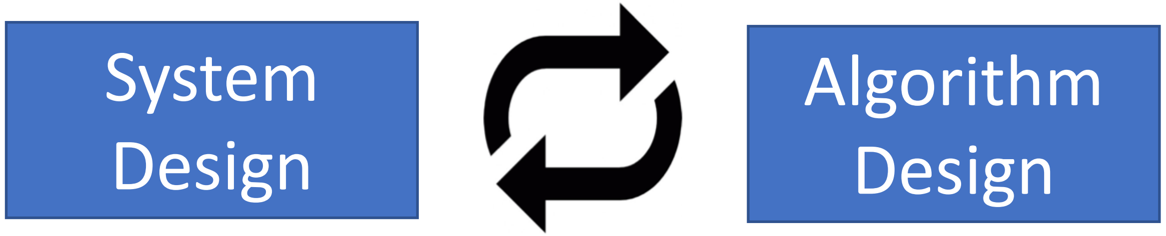
Existing Approaches (Cont'd)

- Parameter matrices are checkpointed to stable storage for fault tolerance.



High Disk IO

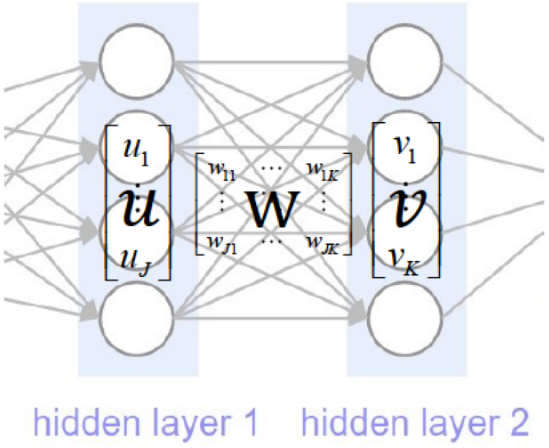
System and Algorithm Co-design



- System design should be tailored to the unique mathematical properties of ML algorithms
- Algorithms can be re-designed to better exploit the system architecture

Sufficient Vectors (SVs)

- Parameter-update matrix can be computed from a few vectors (referred to as sufficient vectors)



Sufficient Vectors

$$\overbrace{\Delta W}^{J\times K\text{ Entries}} = \overbrace{u \otimes v}^{J+K\text{ Entries}}$$

$$\begin{bmatrix} \Delta w_{11} & \cdots & \Delta w_{1K} \\ \vdots & \ddots & \vdots \\ \Delta w_{J1} & \cdots & \Delta w_{JK} \end{bmatrix} \begin{bmatrix} u_1 \\ \vdots \\ u_J \end{bmatrix} \begin{bmatrix} v_1 & \cdots & v_K \end{bmatrix}$$

(Xie et al. 2016)

System and Algorithm Co-design

System Design

- Random multicast
- Incremental SV checkpoint
- Periodic centralized synchronization
- Parameter-replicas rotation

Algorithm Design

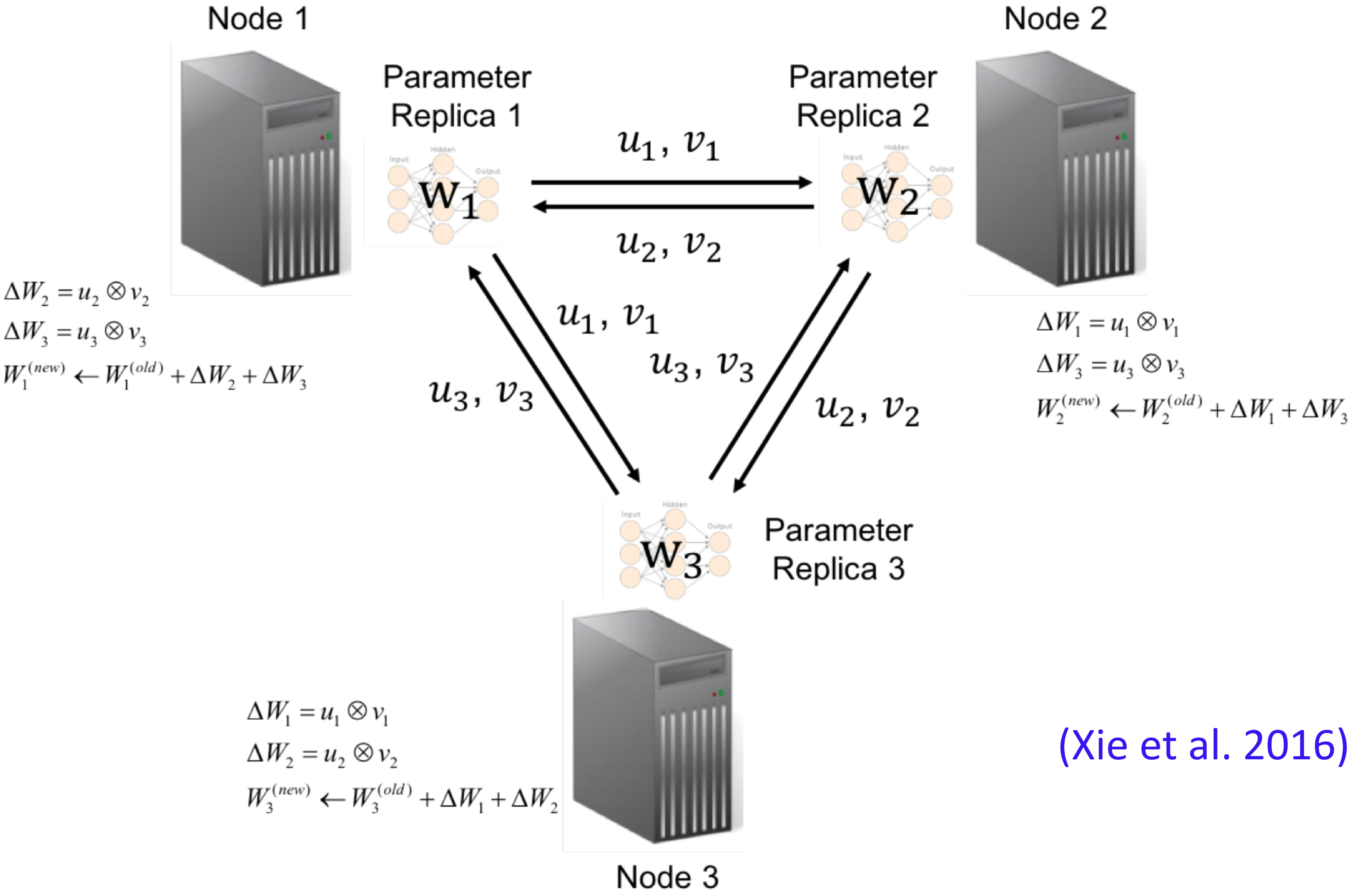
- SV selection
- Using SVs to represent parameter states
- Automatic identification of SVs

Communication, fault tolerance, consistency, programming interface

Outline

- Introduction
- Communication
- Fault tolerance
- Evaluation
- Conclusions

Peer-to-Peer Transfer of SVs



(Xie et al. 2016)

Cost Comparison

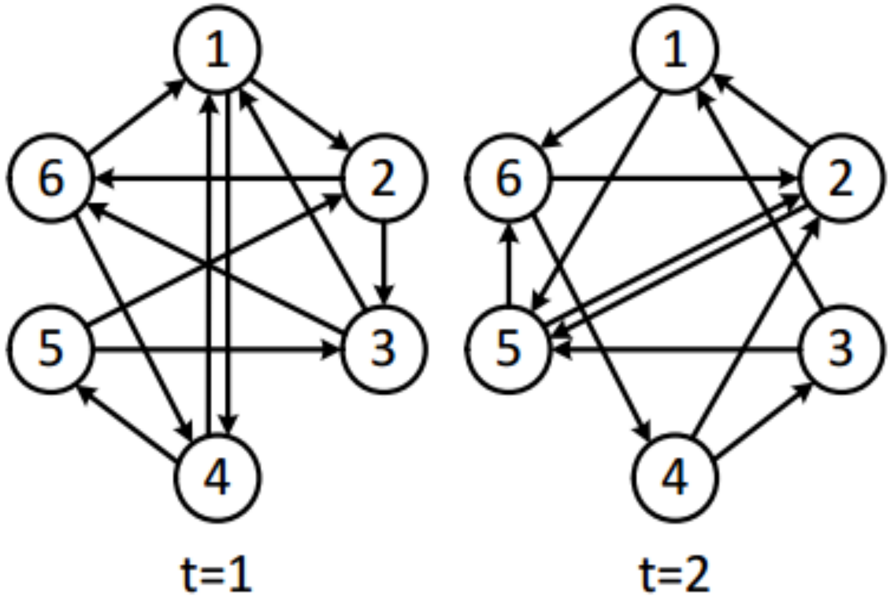
	Size of one message	Number of messages	Network Traffic
P2P SV-Transfer	$O(J+K)$	$O(P^2)$	$O((J+K)P^2)$
Parameter Server	$O(JK)$	$O(P)$	$O(JKP)$

J, K: dimensions of the parameter matrix
P: number of machines

How to reduce the number of messages in P2P?

Random Multicast

- Send SVs to a random subset of Q (Q<<P) machines
- Reduce number of messages from $O(P^2)$ to $O(PQ)$



Random Multicast (Cont'd)

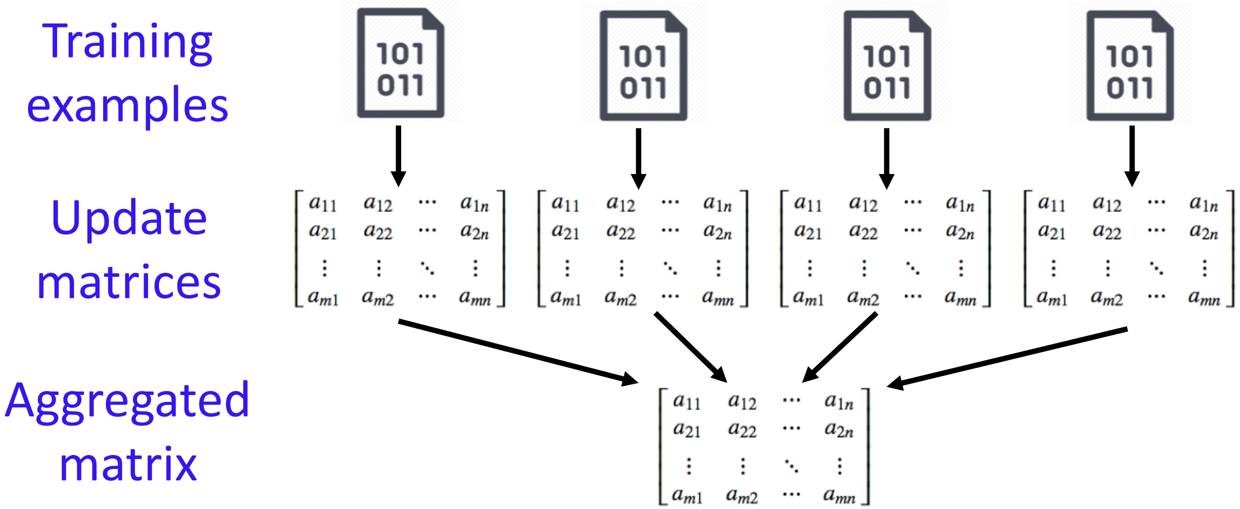
- Correctness is guaranteed due to the error-tolerant nature of ML.

Corollary 1. *Denote* $\{\mathbf{W}_p^c\}$, $p = 1, \ldots, P$, *and* $\{\mathbf{W}^c\}$ *as the local sequences and the auxiliary sequence generated by multicast SFB, respectively, and assume that each machine aggregates the stochastic gradients from* $Q < P$ *other machines at every iteration. Let Assumption 1 hold. Then, with learning rate* $\eta_c \equiv O(\frac{1}{L(P-Q)C})$, *we have*

$$\min_{c=1,\ldots,C} \mathbb{E}\| \sum_{p=1}^{P} \nabla f_p(\mathbf{W}_p^c)\|^2 \leq O\left(L(P-Q) + \frac{(Ls + L_f)\sigma^2}{KPL(P-Q)C} \right).$$

Mini-Batch

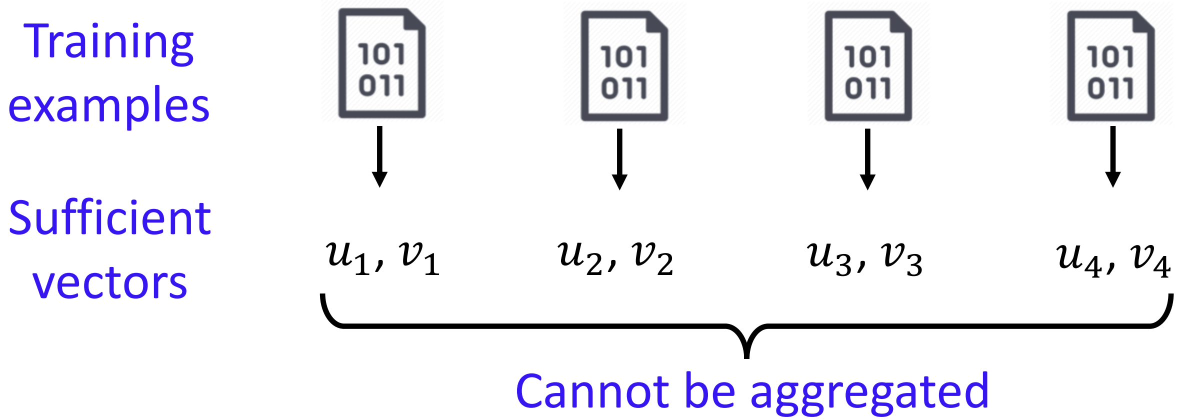
- It is common to use a mini-batch of training examples (instead of one) to compute updates
- If represented as matrices, the updates computed w.r.t different samples can be aggregated into a single update matrix to communicate



- Communication cost does not grow with mini-batch size

Mini-Batch (Cont'd)

- If represented as SVs, the updates computed w.r.t different samples cannot be aggregated into a single SV



- The SVs must be transmitted individually
- Communication cost grows linearly with mini-batch size

SV Selection

- Select a subset of “representative” SVs to communicate
- Reduce communication cost
- Does not hurt the correctness of updates
  - The aggregated update computed from the selected SVs are close to that from the entire mini-batch
  - The selected SVs can well represent the others

SV Selection (Cont'd)

- Algorithm: joint matrix column subset selection

$$\min_I \sum_{r=1}^{R} \left\| X^{(r)} - S_I^{(r)} \left(S_I^{(r)} \right)^{\dagger} X^{(r)} \right\|_2$$

Algorithm 2 Joint Matrix Column Subset Selection

Input: $\{X^{(r)}\}_{r=1}^{R}$
Initialize: $\forall\, r,\, X_0^{(r)} = X^{(r)},\, S_0^{(r)} = []$
for $t \in \{1, \ldots, C\}$ **do**
  Compute the squared L2 norm of column vectors in $\{X_{t-1}^{(r)}\}_{r=1}^{R}$
  Sample a column index i_t
  $\forall\, r,\, X_t^{(r)} \leftarrow X_{t-1}^{(r)} / \{x_{i_t}^{(r)}\},\, S_t^{(r)} \leftarrow S_{t-1}^{(r)} \cup \{x_{i_t}^{(r)}\}$
  $\forall\, r,\, X_t^{(r)} \leftarrow X_t^{(r)} - S_t^{(r)} (S_t^{(r)})^{\dagger} X_t^{(r)}$
end for
Output: $\{S^{(r)}\}_{r=1}^{R}$

Outline

- Introduction
- Communication
- Fault tolerance
- Evaluation
- Conclusions

SV-based Representation

- SV-based representation of parameters
  - At iteration T, the state W_T of the parameter matrix is

$$W_T = \overbrace{W_0}^{\text{Initialization}} + \overbrace{\Delta W_1 + \cdots\cdots \Delta W_T}^{\text{Update matrices}}$$

SV Representation (SVR) $\quad W_T = u_0 v_0^{\mathrm{T}} + u_1 v_1^{\mathrm{T}} \quad \cdots\cdots \quad u_T v_T^{\mathrm{T}}$

Fault Tolerance

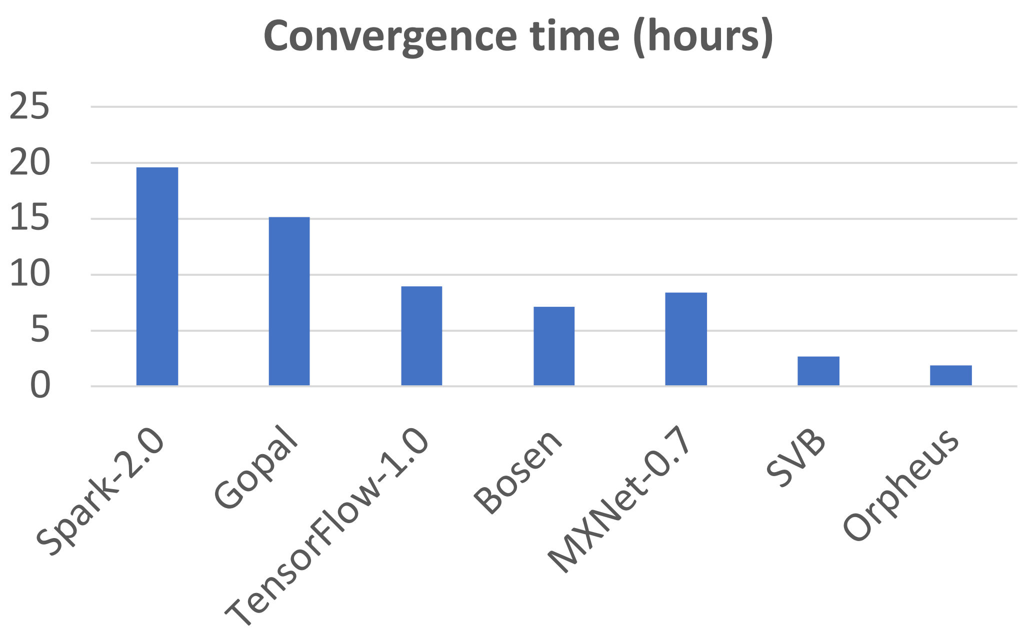
- SV-based checkpoint: save SVs computed in each clock on disk
  - Consume little disk bandwidth
  - Do not halt computation
- Recovery: transform saved SVs into parameter matrix
  - Can rollback to the state of every clock

Outline

- Introduction
- Communication
- Fault tolerance
- Evaluation
- Conclusions

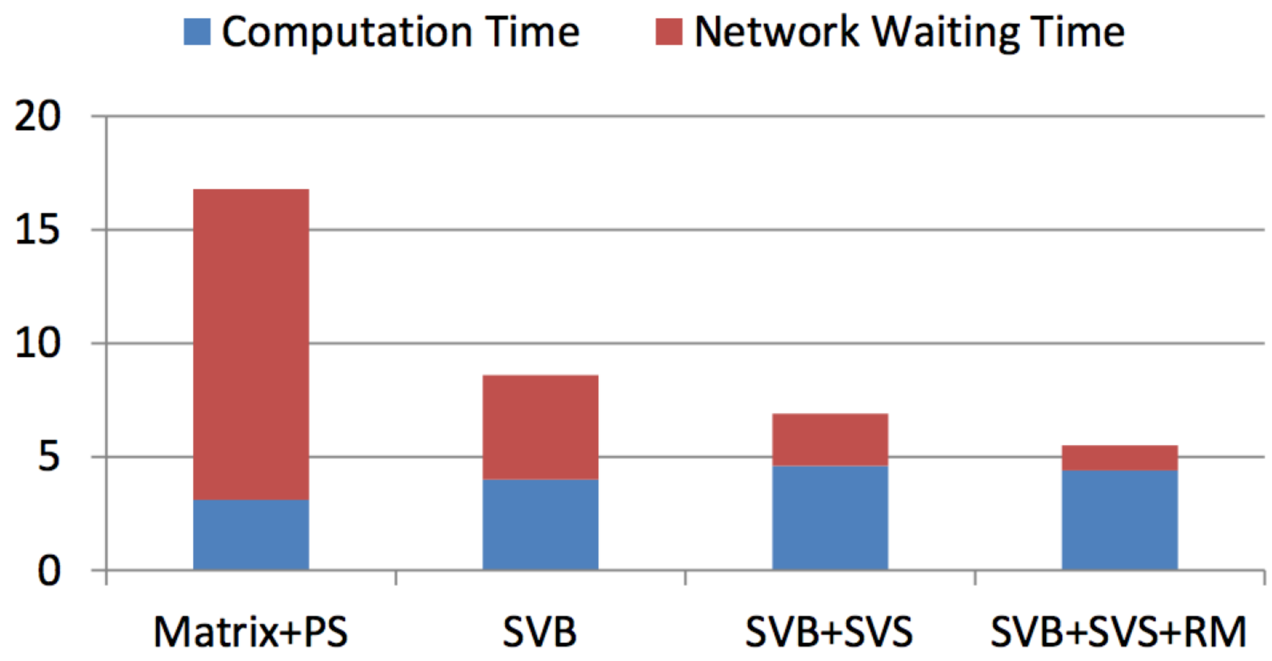
Convergence Speed

Multi-class Logistic Regression (MLR)
Weight matrix: 325K-by-20K

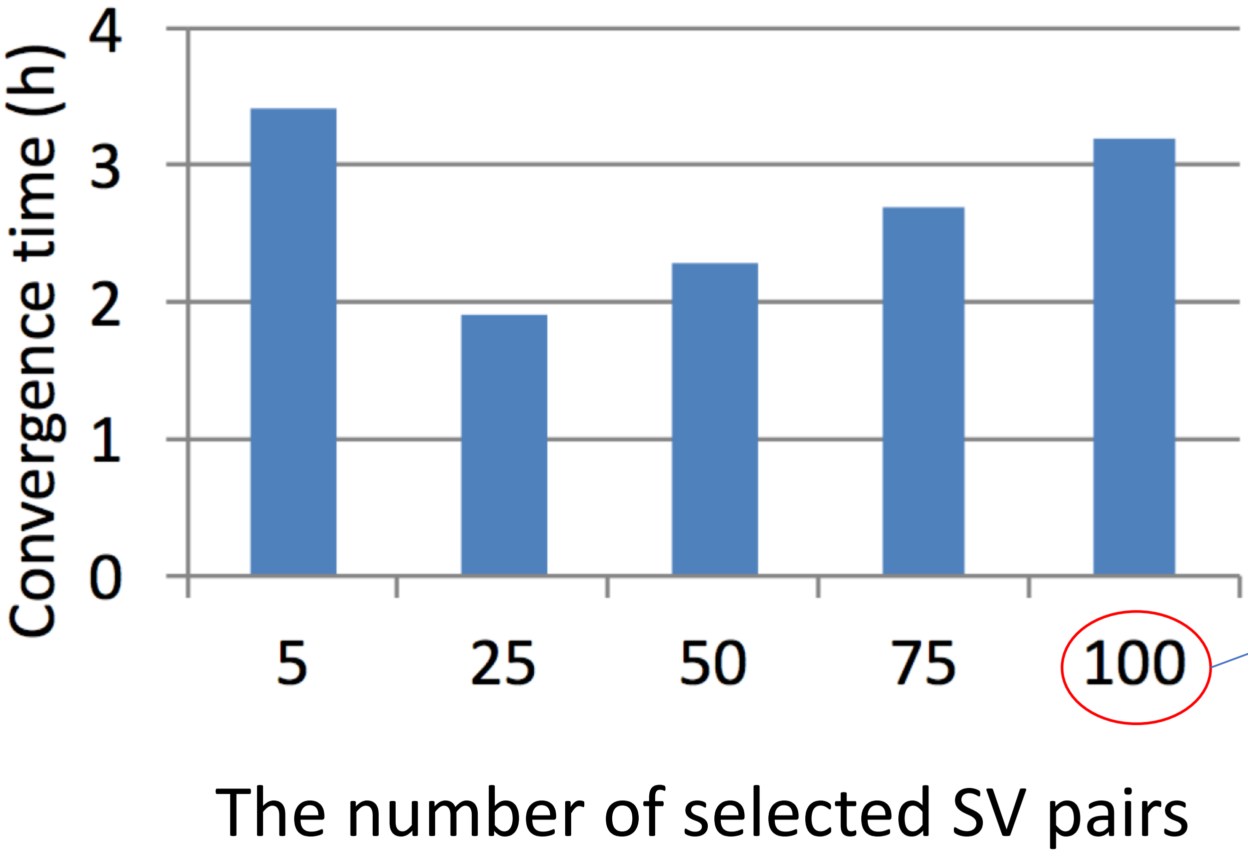


	Convergence time (h)
Matrix+PS	7.1
SVB	2.7
SVB+SVS	2.3
SVB+SVS+RM	2.1

Breakdown of Network Waiting Time and Computation Time

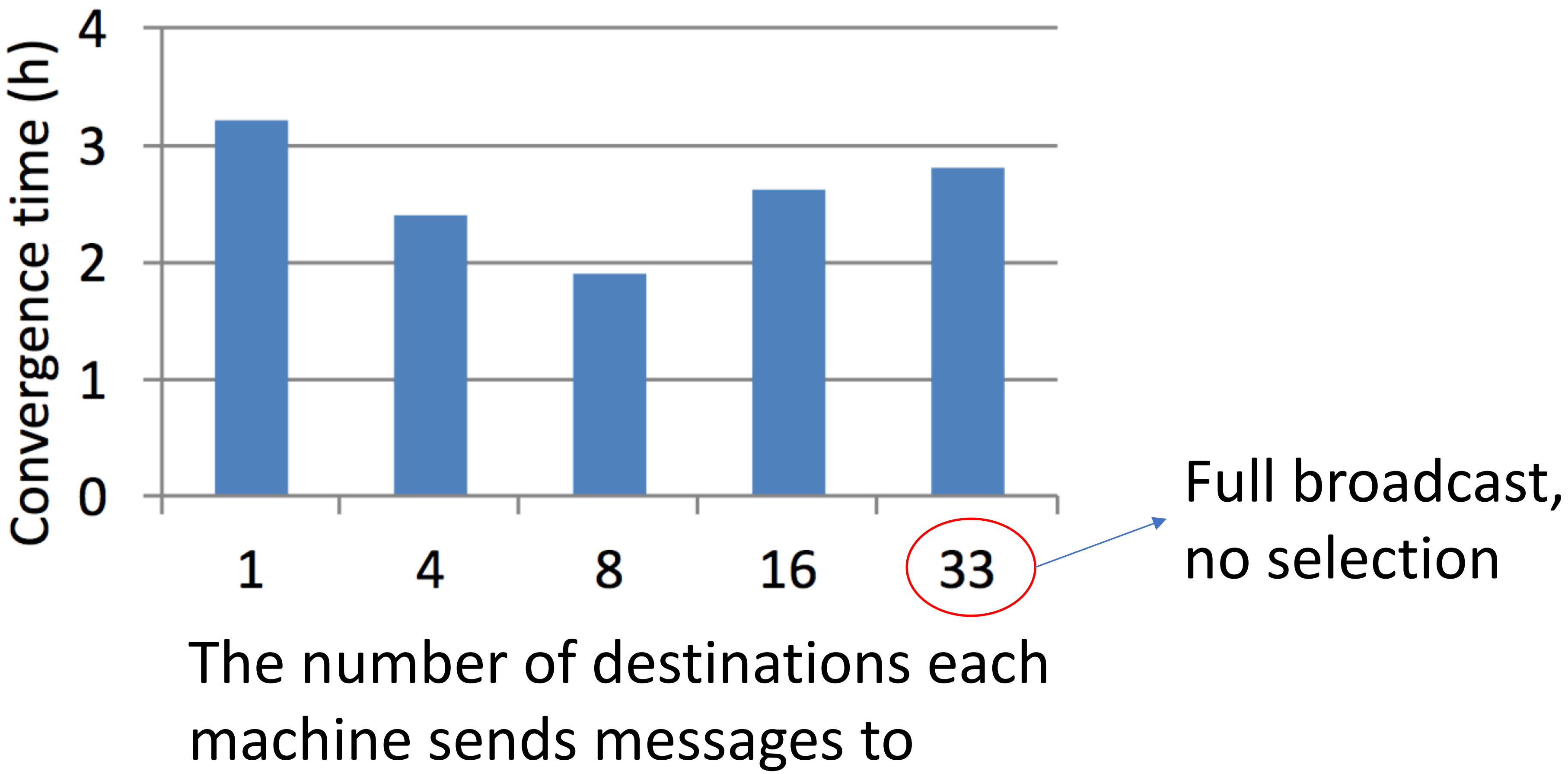


SV Selection



Full batch,
no selection

Random Multicast



Fault Tolerance

	MLR
No checkpoint	1.9
Matrix-based checkpoint	2.4
SV-based checkpoint	1.9
Matrix-based recovery	2.9
SV-based recovery	2.3

Conclusions

System Design

- Random multicast
- Incremental SV checkpoint
- Periodic centralized synchronization
- Parameter-replicas rotation

Algorithm Design

- SV selection
- Using SVs to represent parameter states
- Automatic identification of SVs

Communication, fault tolerance, consistency, programming interface