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

- Facebook: 1B+ users, 30+ petabytes
- Wikipedia: 32 million pages
- YouTube: 100+ hours video uploaded every minute
- Twitter: 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         | Pregel     | GraphX      |
|                                  | GraphLab   |             |

| 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)

Google Brain Neural Network
(Le et al. 2012)

The topic matrix has **50 billion** entries.
The weight matrices have **1.3 billion** entries.
Existing Approaches

- Parameter server frameworks communicate matrices for parameter synchronization.
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)

\[
\Delta W = u \otimes v
\]

\[
\begin{bmatrix}
\Delta w_{1} & \cdots & \Delta w_{i} \\
\vdots & \ddots & \vdots \\
\Delta w_{j} & \cdots & \Delta w_{K}
\end{bmatrix}
\begin{bmatrix}
u_{1} \\
\vdots \\
u_{J}
\end{bmatrix}
\begin{bmatrix}
v_{1} \\
\vdots \\
v_{K}
\end{bmatrix}
\]

(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

\[ \Delta W_1 = u_1 \otimes v_1 \]
\[ \Delta W_2 = u_2 \otimes v_1 \]
\[ W_1^{(new)} \leftarrow W_1^{(old)} + \Delta W_1 + \Delta W_2 \]

\[ u_1, v_1 \quad u_3, v_3 \quad u_2, v_2 \]

\[ \Delta W_3 = u_3 \otimes v_3 \]
\[ W_3^{(new)} \leftarrow W_3^{(old)} + \Delta W_1 + \Delta W_2 \]

(Xie et al. 2016)
Cost Comparison

<table>
<thead>
<tr>
<th></th>
<th>Size of one message</th>
<th>Number of messages</th>
<th>Network Traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2P SV-Transfer</td>
<td>$O(J + K)$</td>
<td>$O(P^2)$</td>
<td>$O((J + K)P^2)$</td>
</tr>
<tr>
<td>Parameter Server</td>
<td>$O(JK)$</td>
<td>$O(P)$</td>
<td>$O(JKP)$</td>
</tr>
</tbody>
</table>

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 \( \{W_p^c\}, p = 1, \ldots, P, \) and \( \{W^c\} \) 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 \( \eta_c \equiv O\left(\frac{1}{L(P-Q)C}\right) \), we have

\[
\min_{c=1,\ldots,C} \mathbb{E}\left\| \sum_{p=1}^{P} \nabla f_p(W_p^c) \right\|^2 \leq O\left( L(P-Q) + \frac{(L_s + L_f)\sigma^2}{KPL(P-Q)C} \right).
\]
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

![](attachment:image)

- 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

\[
\min_{l} \sum_{r=1}^{R} \left\| X^{(r)} - S_{l}^{(r)} \left( S_{l}^{(r)} \right)^{\dagger} X^{(r)} \right\|_{2}
\]

```
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 ∈ \{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

\[
W_T = W_0 + \Delta W_1 + \cdots + \Delta W_T
\]

SV Representation (SVR)

\[
W_T = u_0 v_0^T + u_1 v_1^T + \cdots + u_T v_T^T
\]
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

<table>
<thead>
<tr>
<th>Method</th>
<th>Convergence time (h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix+PS</td>
<td>7.1</td>
</tr>
<tr>
<td>SVB</td>
<td>2.7</td>
</tr>
<tr>
<td>SVB+SVS</td>
<td>2.3</td>
</tr>
<tr>
<td>SVB+SVS+RM</td>
<td>2.1</td>
</tr>
</tbody>
</table>
Breakdown of Network Waiting Time and Computation Time
SV Selection

The number of selected SV pairs

Convergence time (h)

5 25 50 75 100

Full batch, no selection
Random Multicast

The number of destinations each machine sends messages to

- Full broadcast, no selection
Fault Tolerance

<table>
<thead>
<tr>
<th>Method</th>
<th>MLR</th>
</tr>
</thead>
<tbody>
<tr>
<td>No checkpoint</td>
<td>1.9</td>
</tr>
<tr>
<td>Matrix-based checkpoint</td>
<td>2.4</td>
</tr>
<tr>
<td>SV-based checkpoint</td>
<td>1.9</td>
</tr>
<tr>
<td>Matrix-based recovery</td>
<td>2.9</td>
</tr>
<tr>
<td>SV-based recovery</td>
<td>2.3</td>
</tr>
</tbody>
</table>
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