Scaling Distributed Machine Learning

with the

Mu Li

muli@cs.cmu.edu
Overview

Join us in Broomfield, CO, October 6-8, 2014, for the 11th USENIX Symposium on Operating Systems Design and Implementation. **OSDI '14** will bring together professionals from academic and industrial backgrounds in what has become the premier forum for discussing the design, implementation, and implications of systems software. The [program](https://www.usenix.org/conference/osdi14) includes two poster sessions and over 40 paper presentations on data, security, cloud computing, storage, transactions, and much more.

Register Today!

The following workshops are co-located with OSDI '14, and will take place on Sunday, October 5, 2014:

- **Diversity '14**: 2014 Workshop on Supporting Diversity in Systems Research
- **HotDep '14**: 10th Workshop on Hot Topics in System Dependability
- **HotPower '14**: 6th Workshop on Power-Aware Computing and Systems
- **INFLOW '14**: 2nd Workshop on Interactions of NVM/Flash with Operating Systems and Workloads
- **TRIOS '14**: 2014 Conference on Timely Results in Operating Systems
user → query: “osdi” → search engine → charge → advertiser

Google search results for "osdi":
- About 588,000 results (0.30 seconds)
- Ad: Join Us at OSDI '14 - Register by September 11 and save
  - October 6-8, 2014. Broomfield, CO.
  - USENIX has 1,050 followers on Google+
  - You've visited usenix.org many times. Last visit: today
  - OSDI '14 Registration - OSDI '14 Program - OSDI '14 homepage - OSDI '14 Travel
Machine learning is concerned with systems that can learn from data.
Machine learning is concerned with systems that can learn from data.
Machine learning is concerned with systems that can learn from data.
Machine learning is concerned with systems that can learn from data.

Ad click prediction

We will report results using 1,000 machines!
Overview of machine learning

raw data -> training data -> machine learning system -> model (key,value) pairs
Overview of machine learning

raw data → training data → machine learning system → model (key,value) pairs

example by feature matrix

osdi  www.usenix.org  user_mu_li

1  1  1
Overview of machine learning

raw data → training data → machine learning system → model (key,value) pairs

Scale of Industry problems
- 100 billion examples
- 10 billion features
- 1T – 1P training data
- 100–1000 machines
Overview of machine learning

raw data → training data → machine learning system → model (key,value) pairs

Scale of Industry problems

✦ 100 billion examples
✦ 10 billion features
✦ 1T – 1P training data
✦ 100 – 1000 machines

✦ scale to industry problems
✦ efficient communication
✦ fault tolerance
✦ easy to use
Industry size machine learning problems
Data and model partition

Training data
Data and model partition

Training data

Worker machines
Data and model partition

Model

Training data

Worker machines
Data and model partition

Model

Server machines

Training data

Worker machines
Data and model partition

Model

Server machines

Worker machines

push

Training data
Data and model partition

Model

Server machines

Worker machines

push

pull

Training data
Example: distributed gradient descent

Server machines

Worker machines
Example: distributed gradient descent

Workers **pull** the working set of model
Example: distributed gradient descent

Workers **pull** the working set of model

Iterate until stop

workers compute **gradients**

Server machines

Worker machines
Example: distributed gradient descent

Workers pull the working set of model
Iterate until stop
workers compute gradients
workers push gradients
Example: distributed gradient descent

Server machines

Worker machines

Workers pull the working set of model

Iterate until stop

workers compute gradients

workers push gradients

update model
Example: distributed gradient descent

Server machines

Worker machines

Workers **pull** the working set of **model**

Iterate until stop

workers **compute** gradients

workers **push** gradients

update **model**

workers **pull** updated **model**
Industry size machine learning problems

Efficient communication
Challenges for data synchronization
Challenges for data synchronization

- Massive communication traffic
- Frequent access to the shared model
Challenges for data synchronization

- Massive communication traffic
- Frequent access to the shared model
- Expensive global barriers
- Between iterations
Task
Task

- a push / pull / user defined function (an iteration)
Task

- a push / pull / user defined function (an iteration)
- “execute-after-finished” dependency
Task

✦ a push / pull / user defined function (an iteration)
✦ “execute-after-finished” dependency

CPU intensive  Network intensive

iter 0 gradient push & pull
iter 1 gradient push & pull

Synchronous
Task

- a push / pull / user defined function (an iteration)
- “execute-after-finished” dependency
- executed asynchronously

- iter 0: gradient & push & pull
- iter 1: gradient & push & pull

CPU intensive: blue
Network intensive: red
Synchronous: purple
Task

- a push / pull / user defined function (an iteration)
- “execute-after-finished” dependency
- executed asynchronously

CPU intensive

Network intensive

Synchronous

Asynchronous
Flexible consistency

- Trade-off between algorithm efficiency and system performance
Flexible consistency

- Trade-off between algorithm efficiency and system performance

Sequential
Flexible consistency

- Trade-off between algorithm efficiency and system performance

Sequential

![Sequential diagram](image1)

Eventual

![Eventual diagram](image2)
Flexible consistency

- Trade-off between algorithm efficiency and system performance

- Sequential

- 1-bounded delay

- Eventual
Results for bounded delay
Results for bounded delay

Ad click prediction

<table>
<thead>
<tr>
<th>Time (hour)</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bounded delay</td>
<td>0.45</td>
<td>0.9</td>
<td>1.35</td>
<td>1.8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Green** represents computing time
- **Red** represents waiting time
Results for bounded delay

Ad click prediction

Sequential

Computing
Waiting

Time (hour)

Bounded delay
Results for bounded delay

Ad click prediction

sequential

time (hour)

computing

waiting

Bounded delay

0.45

1.35

1.8
Results for bounded delay

Ad click prediction

Sequential trade-off

Time (hour)

Computing

Waiting

Best trade-off
User-defined filters
User-defined filters

- Selectively communicate (key, value) pairs
Selectively communicate (key, value) pairs

E.g., the KKT filter

Send pairs if they are likely to affect the model

> 95% keys are filtered in the ad click prediction task
Industry size machine learning problems

Efficient communication

Fault tolerance
Machine learning job logs in a three-month period:
Machine learning job logs in a three-month period:

<table>
<thead>
<tr>
<th>#machine x time (hour)</th>
<th>failure rate %</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>26</td>
</tr>
<tr>
<td>1000</td>
<td>19.5</td>
</tr>
<tr>
<td>10000</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>6.5</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
Machine learning job logs in a three-month period:

<table>
<thead>
<tr>
<th>#machine x time (hour)</th>
<th>failure rate %</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>6.5</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
</tr>
<tr>
<td>10</td>
<td>19.5</td>
</tr>
<tr>
<td>20</td>
<td>26</td>
</tr>
</tbody>
</table>
Machine learning job logs in a three-month period:

![Bar chart showing failure rate percentage against #machine x time (hour). The chart has two bars: one at 6.5% for 100 machines and one at 13% for 1000 machines.]
Machine learning job logs in a three-month period:

![Bar chart showing failure rate % vs. #machine x time (hour)]
Fault tolerance
Fault tolerance

- Model is partitioned by consistent hashing
Fault tolerance

- Model is partitioned by consistent hashing
- Default replication: Chain replication (consistent, safe)
Fault tolerance

- Model is partitioned by consistent hashing
- Default replication: Chain replication (consistent, safe)
Fault tolerance

- Model is partitioned by consistent hashing
- Default replication: Chain replication (consistent, safe)
- Option: Aggregation reduces backup traffic (algo specific)
Fault tolerance

- Model is partitioned by consistent hashing
- Default replication: Chain replication (consistent, safe)
- Option: Aggregation reduces backup traffic (algo specific)
Fault tolerance

- Model is partitioned by consistent hashing
- Default replication: Chain replication (consistent, safe)

- Option: Aggregation reduces backup traffic (algo specific)

implemented by efficient vector clock
Industry size machine learning problems

Efficient communication

Fault tolerance

Easy to use
(Key, value) vectors for the shared parameters

math sparse vector

\[ i_1 \quad i_2 \quad i_3 \]

(key, value) store

\[ (i_1, \quad) \quad (i_2, \quad) \quad (i_3, \quad) \]
(Key, value) vectors for the shared parameters

- **Good for programmers**: Matches mental model
- **Good for system**: Expose optimizations based upon structure of data
(Key, value) vectors for the shared parameters

- Good for programmers: Matches mental model
- Good for system: Expose optimizations based upon structure of data

Example: computing gradient

\[
\text{gradient} = \text{data}^\top \times (-\text{label} \times 1 / (1 + \exp(\text{label} \times \text{data} \times \text{model}))
\]
Industry size machine learning problems

Efficient communication

Fault tolerance

Easy to use

Evaluation
Sparse Logistic Regression

- Predict ads will be clicked or not
Sparse Logistic Regression

- Predict ads will be clicked or not
- Baseline: two systems in production

<table>
<thead>
<tr>
<th></th>
<th>Method</th>
<th>Consistency</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>System-A</td>
<td>L-BFGS</td>
<td>Sequential</td>
<td>10K</td>
</tr>
<tr>
<td>System-B</td>
<td>Block PG</td>
<td>Sequential</td>
<td>30K</td>
</tr>
<tr>
<td>Parameter Server</td>
<td>Block PG</td>
<td>Bounded Delay + KKT</td>
<td>300</td>
</tr>
</tbody>
</table>
Sparse Logistic Regression

- Predict ads will be clicked or not
- Baseline: two systems in production

<table>
<thead>
<tr>
<th>Method</th>
<th>Consistency</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>System-A</td>
<td>L-BFGS</td>
<td>10K</td>
</tr>
<tr>
<td>System-B</td>
<td>Block PG</td>
<td>30K</td>
</tr>
<tr>
<td>Parameter Server</td>
<td>Block PG</td>
<td>Bounded Delay + KKT</td>
</tr>
</tbody>
</table>

- 636T real ads data
- 170 billions of examples, 65 billions of features
- 1,000 machines with 16,000 cores
Progress

System A

large error

small error

time (hour)

0.1 1 10
Progress

System A

System B

time (hour)

large

error

small
Progress

System A
System B
Parameter Server

large error

small error

time (hour)

0.1 1 10
Time decomposition

time (hour)
Time decomposition

- **System-A**
- **System-B**
- Parameter Server

- **computing**
- **waiting**

<table>
<thead>
<tr>
<th>Time (hour)</th>
<th>System-A</th>
<th>System-B</th>
<th>Parameter Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.75</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Time decomposition

- **System-A**: 2.5 hours computing, 1.25 hours waiting
- **System-B**: 3.75 hours computing, 0.75 hours waiting
- **Parameter**: 5.0 hours computing, 0.0 hours waiting
Time decomposition

- System-A: 3.75 hours (computing 2.5 hours, waiting 1.25 hours)
- System-B: 1 hour (computing 0.5 hour, waiting 0.5 hour)
- Parameter Server: 0 hours
Time decomposition

- **System-A**:
  - Computing: 3.75 hours
  - Waiting: 1.25 hours

- **System-B**:
  - Computing: 1.25 hours
  - Waiting: 0.75 hours

- **Parameter Server**:
  - Computing: 1.0 hour
  - Waiting: 0.0 hours

Time (hour)
Topic Modeling ("LDA")

- Gradient descent with eventual consistency
- 5B users’ click logs, Group users into 1,000 groups based on URLs they clicked
Topic Modeling ("LDA")

- Gradient descent with \textit{eventual} consistency
- 5B users’ click logs, Group users into 1,000 groups based on URLs they clicked
Topic Modeling ("LDA")

- Gradient descent with eventual consistency
- 5B users’ click logs, Group users into 1,000 groups based on URLs they clicked
Topic Modeling ("LDA")

- Gradient descent with eventual consistency
- 5B users’ click logs, Group users into 1,000 groups based on URLs they clicked

![Graph showing 4x speed up with 1,000 machines compared to 6,000 machines. The y-axis represents error, and the x-axis represents time (hour). The graph shows a steep decline in error over time for both large and small groups, with the 1,000 machines curve reaching a lower error level faster than the 6,000 machines curve.]
Largest experiments of related systems

Data were collected on April’14
Largest experiments of related systems

Data were collected on April’14

Parameter Server

Sparse LR

model size

model size

#cores

Petuum (Lasso)

Naiad (LR)

VW (LR)

MLbase (LR)

REEF (LR)
Largest experiments of related systems

Data were collected on April’14

- Parameter Server
- Sparse LR
- LDA

model size

- Petuum (Lasso)
- Naiad (LR)
- VW (LR)
- MLbase (LR)
- REEF (LR)
- YahooLDA (LDA)
- Graphlab (LDA)

#cores

Data were collected on April’14.
Largest experiments of related systems

Data were collected on April’14

Model size vs. #cores graph showing the model size vs. number of cores for various systems:
- Parameter Server
- Sparse LR
- Adam (DNN)
- Petuum (Lasso)
- Naiad (LR)
- VW (LR)
- MLbase (LR)
- REEF (LR)
- Graphlab (LDA)
- YahooLDA (LDA)
- Distbelief (DNN)
- Graphlab (LDA)

Data were collected on April’14.
Industry size machine learning problems

Efficient communication

Fault tolerance
Easy to use
Evaluation
Industry size machine learning problems

Efficient communication

Code available at parameterserver.org

Fault tolerance

Easy to use

Evaluation
Q&A