Scaling Distributed Machine Learning with the

Mu Li
mulis.cs.cmu.edu
How does a search engine make money?

User

query: “osdi”

Search Engine
How does a search engine make money?

User

query: “osdi”

Search Engine

About 588,000 results (0.30 seconds)

Join Us at OSDI '14 - Register by September 11 and save
Ad www.usenix.org/osdi14/
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
How does a search engine make money?

Machine learning problem:
Predict whether or not a user will click on an ad
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.
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 x feature matrix
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
- 1TB – 1PB 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
- 1TB – 1PB training data
- 100 – 1000 machines

Scale to industry problems
- Efficient communication
- Fault tolerance
- Easy to use
Industry size machine learning problems

Fault tolerance

Easy to use

Evaluation

Efficient communication
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

push

Worker machines

Training data
Data and model partition

Model

Server machines

Worker machines

Training data

push

pull
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

Iterate until stop

Workers **pull** the working set of model

workers **compute** gradients

workers **push** gradients
Example: distributed gradient descent

Workers pull the working set of model

Iterate until stop

Workers compute gradients

Workers push gradients

Update model

Server machines

Worker machines
Example: distributed gradient descent

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

Fault tolerance

Easy to use

Evaluation
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
Challenges for data synchronization

- Massive communication traffic
  - Frequent access to the shared model
- Expensive global barriers between iterations
- Our solution is to relax the data consistency in two levels
  - Task dependency graph
  - User-defined filters
Range-based Push & Pull

- Programmer: fit most machine learning algorithms
- System: communication are batched
Range-based Push & Pull

- **Programmer:** fit most machine learning algorithms
- **System:** communication are batched

![Diagram showing three servers and two workers]
Range-based Push & Pull

- Programmer: fit most machine learning algorithms
- System: communication are batched

Server 0

Server 1

Server 2

Worker 0

Worker 1

Push { }
Range-based Push & Pull

- Programmer: fit most machine learning algorithms
- System: communication are batched
Task

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

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

iter 0
- CPU intensive: gradient
- Network intensive: push & pull

iter 1
- CPU intensive: gradient
- Network intensive: 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

Synchronous

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

Asynchronous
Data consistency

- Algorithm efficiency vs. system performance
- Flexible models via task dependency graph
Data consistency

- Algorithm efficiency vs. system performance
- Flexible models via task dependency graph
- Some examples [Bertsekas 89]
Data consistency

- Algorithm efficiency vs. system performance
- Flexible models via task dependency graph
- Some examples [Bertsekas 89]
Data consistency

- Algorithm efficiency vs. system performance
- Flexible models via task dependency graph
- Some examples [Bertsekas 89]
Data consistency

- Algorithm efficiency vs. system performance
- Flexible models via task dependency graph
- Some examples [Bertsekas 89]

Sequential / BSP

Bounded delay / SSP
[Langford 09, Cipar 13]

Eventual / Total asynchronous
[Smola 10]
Time to the Same Convergence Criteria

Ads click prediction

636TB data, 1TB model, and 1000 machines
Time to the Same Convergence Criteria

Ads click prediction

636TB data, 1TB model, and 1000 machines

Maximal delay

<table>
<thead>
<tr>
<th>Time (hour)</th>
<th>Computing</th>
<th>Waiting</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Time to the Same Convergence Criteria

Ads click prediction
636TB data, 1TB model, and 1000 machines

Sequential

<table>
<thead>
<tr>
<th>Maximal delay</th>
<th>time (hour)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>1.5</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>16</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximal delay</th>
<th>computing</th>
<th>waiting</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>16</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

Computing and waiting times in sequential order.
Time to the Same Convergence Criteria

Ads click prediction

636TB data, 1TB model, and 1000 machines

sequential

Maximal delay

<table>
<thead>
<tr>
<th>Maximal delay</th>
<th>Computing</th>
<th>Waiting</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>0</td>
</tr>
</tbody>
</table>

time (hour)

Computing and waiting times for sequential processing.
Time to the Same Convergence Criteria

Ads click prediction
636TB data, 1TB model, and 1000 machines

Sequential

Computing: waiting

Best trade-off: 1.6x

Maximal delay

Time (hour)

0 1 2 4 8 16
Filters: control data consistency in a task
Filters: control data consistency in a task

(key, value) pairs of a task → filter 1 → new (key, value) pairs
Filters: control data consistency in a task

(key, value) pairs of a task → filter 1 → filter 2 → filter 3 → new (key, value) pairs
Filters: control data consistency in a task

(key, value) pairs of a task

filter 1  filter 2  filter 3

new (key, value) pairs

A filter could

► Selectively filter (key, value) pairs
► Transform data
A filter could
- Selectively filter (key, value) pairs
- Transform data

Examples
- Significantly modified filter: send entry e only if $|\Delta e| > \epsilon$
- Randomly skip filter: randomly skip entries
Key caching

- The keys sent in a task could be resent by another task
  - Push the gradients, then pull the updated weights
  - Keys are fixed between iterations
- Do cache. If hit the cache, then only send the signature
Key caching

- The keys sent in a task could be resent by another task
  - Push the gradients, then pull the updated weights
  - Keys are fixed between iterations
- Do cache. If hit the cache, then only send the signature

Node A

Task 1: Keys

Node B

Values

Task 10:
Key caching

- The keys sent in a task could be resent by another task
  - Push the gradients, then pull the updated weights
  - Keys are fixed between iterations
- Do cache. If hit the cache, then only send the signature
Key caching

- The keys sent in a task could be resent by another task
  - Push the gradients, then pull the updated weights
  - Keys are fixed between iterations
- Do cache. If hit the cache, then only send the signature

Node A

The key signature

Node B

Task 1:
- Keys

Task 10:
- Keys
  - Values
Key caching

- The keys sent in a task could be resent by another task
  - Push the gradients, then pull the updated weights
  - Keys are fixed between iterations

- Do cache. If hit the cache, then only send the signature

Node A

Task 1:

- Keys

The key signature

Node B

Task 10:

- Keys

  - Values
Key caching

- The keys sent in a task could be resent by another task
  - Push the gradients, then pull the updated weights
  - Keys are fixed between iterations
- Do cache. If hit the cache, then only send the signature

```
Node A          Node B
   The key signature

Task 1:         Task 10:
Keys            Keys
```
Data Compressing

- Values are often compressible numbers
  - 0s, small integers, numbers with more than enough precision
- Lossness compression algorithms: LZ, LZR, ...
- Variable-length integers
- Lossy compression such as fixed-point encoding
KKT Filter
KKT Filter

- Only send the subset of gradient that is likely to affect the model
KKT Filter

- Only send the subset of gradient that is likely to affect the model
- Assume we use sparse penalty $\lambda |w|_1$, then according to the KKT condition:

\[ \text{iteration } i: \ w[\text{key}] = 0 \text{ and } |\text{gradient}[\text{key}]| < \lambda \]

\[ \downarrow \]

\[ \text{iteration } i+1: \ w[\text{key}] = 0 \]
Traffic Reduction by Filters

Ad click prediction

636TB data, 1TB model, and 1000 machines

Server                      Worker
Traffic Reduction by Filters

Ad click prediction

636TB data, 1TB model, and 1000 machines

Server

Baseline  Key Caching  Compressing  KKT Filter

Worker

Baseline  Key Caching  Compressing  KKT Filter
Traffic Reduction by Filters

Ad click prediction

636TB data, 1TB model, and 1000 machines

Server

Worker

Baseline  Key Caching  Compressing  KKT Filter

Baseline  Key Caching  Compressing  KKT Filter

2x
Traffic Reduction by Filters

Ad click prediction

636TB data, 1TB model, and 1000 machines

Server

Worker

- Baseline
- Key Caching
- Compressing
- KKT Filter

Baseline: 100%

Key Caching: 2x

Compressing: 40x

KKT Filter: 40x
Traffic Reduction by Filters

Ad click prediction

636TB data, 1TB model, and 1000 machines

Server

Baseline  Key Caching  Compressing  KKT Filter

Worker

Baseline  Key Caching  Compressing  KKT Filter

2x  40x  40x
Traffic Reduction by Filters

Ad click prediction

636TB data, 1TB model, and 1000 machines

Server

<table>
<thead>
<tr>
<th>Method</th>
<th>Baseline</th>
<th>Key Caching</th>
<th>Compressing</th>
<th>KKT Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>100</td>
<td>4x</td>
<td>2x</td>
<td>2x</td>
</tr>
<tr>
<td>Key Caching</td>
<td>100</td>
<td>40x</td>
<td>40x</td>
<td>40x</td>
</tr>
</tbody>
</table>

Worker

<table>
<thead>
<tr>
<th>Method</th>
<th>Baseline</th>
<th>Key Caching</th>
<th>Compressing</th>
<th>KKT Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>100</td>
<td>2x</td>
<td>2.5x</td>
<td>2x</td>
</tr>
<tr>
<td>Key Caching</td>
<td>100</td>
<td>2x</td>
<td>2.5x</td>
<td>2x</td>
</tr>
<tr>
<td>Compressing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KKT Filter</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Traffic Reduction by Filters

Ad click prediction

636TB data, 1TB model, and 1000 machines

Server

Worker
The relax consistency guarantees convergence?
The relax consistency guarantees convergence?

Short answer: Yes
The relax consistency guarantees convergence?

- Short answer: Yes
- An algorithm often has a tunable step size, a small value guarantees convergence
The relax consistency guarantees convergence?

- Short answer: Yes
- An algorithm often has a tunable step size, a small value guarantees convergence
- Theorem. Assume
  - maximal $\tau$ delays
  - properly stack previous filters
The relax consistency guarantees convergence?

- Short answer: Yes
- An algorithm often has a tunable step size, a small value guarantees convergence
- Theorem. Assume
  - maximal $\tau$ delays
  - properly stack previous filters
- Then a large group of algorithms convergence if
  \[ \text{step size} \leq O \left( \frac{1}{\Delta_1 + \tau \Delta_2} \right) \]
  data correlation within a task $\Delta_1$ / between tasks $\Delta_2$
Industry size machine learning problems

Efficient communication

fault tolerance

Easy to use

Evaluation
Machine learning job logs in a three-month period:
Machine learning job logs in a three-month period:

failure rate %

#machine x time (hour)
Machine learning job logs in a three-month period:

failure rate %

#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

Evaluation
(Key, value) vectors for the shared parameters

math sparse vector

\[ \begin{align*}
  &i_1 &i_2 &i_3 \\
\end{align*} \]

(key, value) store

\[ \begin{align*}
  &(i_1, \text{blue}) & (i_2, \text{green}) & (i_3, \text{orange}) \\
\end{align*} \]
(Key, value) vectors for the shared parameters

Math sparse vector

\[
\begin{array}{ccc}
  i_1 & i_2 & i_3 \\
\end{array}
\]

(key, value) store

\[
( i_1, \text{blue} ) \ ( i_2, \text{green} ) \ ( i_3, \text{orange} )
\]

- 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

\[
\text{gradient} = \text{data}^T \times ( - \text{label} \times 1 / (1 + \exp(\text{label} \times \text{data} \times \text{model}))
\]
Industry size machine learning problems
Efficient communication
Fault tolerance
Easy to use
Evaluation
Sparse Logistic Regression

- Predict whether ads will be clicked or not
- Baseline: two production systems of a leading Internet company
Sparse Logistic Regression

- Predict whether ads will be clicked or not
- Baseline: two production systems of a leading Internet company

<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 whether ads will be clicked or not
- Baseline: two production systems of a leading Internet company

<table>
<thead>
<tr>
<th>System</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 whether ads will be clicked or not
- Baseline: two production systems of a leading Internet company

<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>

Only for sparse logistic regression

Implement the method used by System-B with another 10K system codes
Sparse Logistic Regression

- Predict whether ads will be clicked or not
- Baseline: two production systems of a learning Internet company
- Collected 636TB real ads data
  - 170 billion examples, 65 billion features
- Train 1TB model
  - 1,000 machines with 16,000 CPU cores
Progress

time (hour)

large

small

error
Progress

large
small
error

System A

time (hour)
Progress

System A
System B

large
small

time (hour)

large error

small error

System A
System B
Time to the Same Convergence Criteria

time (hour)
Time to the Same Convergence Criteria

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

computing | waiting
Time to the Same Convergence Criteria

- System-A
- System-B
- Parameter Server

computing: 2x waiting
Time to the Same Convergence Criteria

- **System-A**: 3.75 hours
  - Computing: 2.5 hours
  - Waiting: 1.25 hours

- **System-B**: 2x
  - Computing: 1.25 hours
  - Waiting: 1.25 hours

- **Server**: 4x
  - Computing: 2 hours
  - Waiting: 2 hours
Topic Modeling ("LDA")

- Gradient descent with eventual consistency
- Click logs for 5B users, then group users into 1,000 categories based on clicked URLs
Topic Modeling ("LDA")

- Gradient descent with eventual consistency
- Click logs for 5B users, then group users into 1,000 categories based on clicked URLs
Topic Modeling ("LDA")

- Gradient descent with eventual consistency
- Click logs for 5B users, then group users into 1,000 categories based on clicked URLs
Topic Modeling ("LDA")

- Gradient descent with eventual consistency
- Click logs for 5B users, then group users into 1,000 categories based on clicked URLs

![Graph showing error reduction over time for 1,000 and 6,000 machines, with a 4x speed up for 1,000 machines compared to 6,000 machines.]

- 1,000 machines
- 6,000 machines

Time (hour):
- 0
- 10
- 20
- 30
- 40
- 50
- 60
- 70

Error:
- Large
- Small
Largest experiments of related systems

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

Data were collected on April’14

- REEF (LR)
- Naiad (LR)
- VW (LR)
- MLbase (LR)
- Petuum (Lasso)

Parameter Server
Sparse LR

Model size vs. #cores graph:
- Log scale for model size and #cores
- Data points for each system

Note: Data were collected on April’14.
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
- Adam (DNN)
- Petuum (Lasso)
- Naiad (LR)
- VW (LR)
- MLbase (LR)
- REEF (LR)
- Distbelief (DNN)
- YahooLDA (LDA)
- Graphlab (LDA)

model size

#cores

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

Efficient communication

Fault tolerance

Easy to use

Evaluation
Data and model partition

Efficient communication

Fault tolerance

Easy to use

Evaluation
Industry size machine learning problems

Data and model partition

Relax data consistency:
1. Task dependency graph
2. User defined filters

Fault tolerance
Easy to use
Evaluation
Industry size machine learning problems

Efficient communication

Fault tolerance

Easy to use

Evaluation

Data and model partition

Relax data consistency:
1. Task dependency graph
2. User defined filters

1. Consistency hashing
2. (Aggregated) chain replication

Easy to use
Data and model partition

Relax data consistency:
1. Task dependency graph
2. User defined filters

1. Consistency hashing
2. (Aggregated) chain replication

Use as math objects

Evaluation
Industry size machine learning problems

Data and model partition

Relax data consistency:
1. Task dependency graph
2. User defined filters

1. Consistency hashing
2. (Aggregated) chain replication

Use as math objects

Two large scale applications
Data and model partition

Relax data consistency:
1. Task dependency graph
2. User defined filters

Codes are available at parameterserver.org

1. Consistency hashing
2. (Aggregated) chain replication

Use as math objects

Two large scale applications