Example: Click prediction for online ads

- Predict **click/no-click** given ad and webpage
- Training data from past click logs
- Subsampled to **17B examples** for computational feasibility
- Logistic regression with **16M parameters**
Distributed machine learning

- Distributed network of communicating nodes
Distributed machine learning

- Distributed network of communicating nodes
- Each node possesses a subset of data

Train a model on entire data

\[ \min_w \sum_{i=1}^{n} \text{loss}(w^T x_i, y_i) + \lambda R(w) \]
Distributed machine learning

- Distributed network of communicating nodes
- Each node possesses a subset of data
- Train a model on entire data

$$
\min_w \sum_{i=1}^{n} \text{loss}(w^T x_i, y_i) + \lambda R(w)
$$

Entire data
Outline

- Communication framework: AllReduce
- Computational algorithms: Hybrid online and batch approach
- Experimental evaluation
Approach 1: Map-Reduce

- Typical optimization methods are iterative (e.g. gradient descent, Quasi-Newton, ...)
- Each iteration as a Map-Reduce job
- Nodes compute gradient (approx. Hessian etc.) on local data
- Model update in reduce step
Approach 1: Map-Reduce

- Typical optimization methods are iterative (e.g. gradient descent, Quasi-Newton, ...)
- Each iteration as a Map-Reduce job
- Nodes compute gradient (approx. Hessian etc.) on local data
- Model update in reduce step
- Widely available, robust, comes with DFS and scheduler
Approach 1: Map-Reduce

- Typical optimization methods are iterative (e.g. gradient descent, Quasi-Newton, ...)
- Each iteration as a Map-Reduce job
- Nodes compute gradient (approx. Hessian etc.) on local data
- Model update in reduce step
- Widely available, robust, comes with DFS and scheduler
- Data transfer, scheduling overheads for each iteration
- Disk-based communication in reduce step is slow
- Can be prohibitively slow for distributed machine learning
Desired computation/communication pattern

- Each node receives a subset of data
- Performs some local computation
- Communication over network to synchronize
- Resume computation locally
Desired computation/communication pattern

- Each node receives a subset of data
- Performs some local computation
- Communication over network to synchronize
- Resume computation locally
- Persistent nodes across communication rounds
Approach 2: AllReduce

- Every node begins with a number (vector)

```
1
/   \
2   3
/ \
4 5 6
```

- Extends to other functions: max, average, gather, ...
Approach 2: AllReduce

- Every node begins with a number (vector)

![Diagram]

- Extends to other functions: max, average, gather, ...
Approach 2: AllReduce

- Every node begins with a number (vector)

```
28
/   \
/     \
/       \
4       5       6       7
```

- Extends to other functions: max, average, gather, ...
Approach 2: AllReduce

- Every node begins with a number (vector)
- Every node ends up with the sum
- Ideal to sum local gradients, weights, ...

Extends to other functions: max, average, gather, ...
Distributed machine learning with AllReduce

- Each node receives a subset of data
- Create a spanning tree over the nodes
- At each iteration
  - Nodes compute gradient over local data at current model
  - AllReduce computes gradient over entire data
Distributed machine learning with AllReduce

- Each node receives a subset of data
- Create a spanning tree over the nodes
- At each iteration
  - Nodes compute gradient over local data at current model
  - AllReduce computes gradient over \textit{entire data}
- Similar implementations for parallel L-BFGS, stochastic gradient descent, etc
AllReduce in Hadoop

- AllReduce typically provided in MPI
  - Not robust to failures
  - No scheduler, DFS, ...
AllReduce in Hadoop

- AllReduce typically provided in MPI
  - Not robust to failures
  - No scheduler, DFS, ...
- We developed a Hadoop-compatible AllReduce
  - Hadoop provides robustness to node failures, slowdowns
  - Hadoop provides scheduler, DFS, ...
  - AllReduce provides fast communication
AllReduce typically provided in MPI
- Not robust to failures
- No scheduler, DFS, ...

We developed a Hadoop-compatible AllReduce
- Hadoop provides robustness to node failures, slowdowns
- Hadoop provides scheduler, DFS, ...
- AllReduce provides fast communication

Easy to parallelize existing ML software
AllReduce v/s MapReduce

<table>
<thead>
<tr>
<th></th>
<th>Full size</th>
<th>10% sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>MapReduce</td>
<td>1690</td>
<td>1322</td>
</tr>
<tr>
<td>AllReduce</td>
<td>670</td>
<td>59</td>
</tr>
</tbody>
</table>

- Overheads of MapReduce lead to slower run-time
- More pronounced for smaller datasets
Outline

- Communication framework: AllReduce
- **Computational algorithms:** Hybrid online and batch approach
- Experimental evaluation
Recall the problem of interest

\[
\min_w \sum_{i=1}^n \text{loss}(w^T x_i, y_i) + \lambda R(w)
\]

Two main classes of optimization algorithms:
- **Online/stochastic**, e.g.: stochastic gradient descent, dual averaging
- **Batch**, e.g.: gradient descent, Newton method, L-BFGS
Optimization algorithms

- Recall the problem of interest
  \[
  \min_w \sum_{i=1}^{n} \text{loss}(w^T x_i, y_i) + \lambda R(w)
  \]

- Two main classes of optimization algorithms:
  - **Online/stochastic**, e.g.: stochastic gradient descent, dual averaging
  - **Batch**, e.g.: gradient descent, Newton method, L-BFGS

- **Stochastic** methods converge *fast initially*, slow towards the end
- **Batch algorithms** have a fast convergence from a *good initialization*
Optimization algorithms

- Recall the problem of interest

\[
\min_w \sum_{i=1}^n \text{loss}(w^T x_i, y_i) + \lambda R(w)
\]

- Two main classes of optimization algorithms:
  - **Online/stochastic**, e.g.: stochastic gradient descent, dual averaging
  - **Batch**, e.g.: gradient descent, Newton method, L-BFGS

- **Stochastic** methods converge *fast initially*, slow towards the end
- **Batch algorithms** have a fast convergence from a *good initialization*
Hybrid approach

- Each node starts with stochastic optimization on local data
- Average solutions after one pass over local data across network
Hybrid approach

- Each node starts with stochastic optimization on local data
- Average solutions after one pass over local data across network
- Warmstart L-BFGS from averaged solution
- Repeat until convergence
  - Each node computes gradient over local data
  - AllReduce to sum gradients over all examples
  - Each node computes updated weights locally
Warmstart gains

- Plot of optimization error v/s passes over data
- Can do more online passes before L-BFGS
Outline

- Communication framework: AllReduce
- Computational algorithms: Hybrid online and batch approach
- Experimental evaluation
Datasets

- Display advertising
  - Predict click given \((context, ad)\) pair
  - 2.3B samples
  - 16M dimensions, 125 non-zero per example

- Splice site recognition
  - Predict human acceptor splice site
  - 50M examples
  - 11M dimensions, 3300 non-zero per example
Parallel learning speedup

- Speedup to a *fixed test error* on display advertising data
- Speedup relative to 10 nodes
- Similar speedups for splice site data
Scalability

- 8 times larger version of advertising data (16B examples)
- Cluster of 1000 nodes
- 10 passes over data for training, net time **70 minutes**
Scalability

- 8 times larger version of advertising data (16B examples)
- Cluster of 1000 nodes
- 10 passes over data for training, net time 70 minutes
- Throughput (non-zero features / s): 470M features / s
Comparison with some previous work

- Zinkevich et al.: Overcomplete partition, averaged SGD
- Dekel et al.: Parallel mini-batch SGD

Splice site recognition

Display advertising
Conclusion

- Linear learning system scalable to large datasets and clusters
- *AllReduce* as an effective communication paradigm
- Hadoop implementation improves reliability and scalability
- *Hybrid optimization* approach beneficial in distributed settings
- More detailed empirical evaluation in the paper
Future directions

- Synchronization at each step, driven by slowest node
- Parameter averaging does not work for non-linear learners

Open source software at [http://hunch.net/~vw](http://hunch.net/~vw)

Future directions

- Synchronization at each step, driven by slowest node
- Parameter averaging does not work for non-linear learners
- Open source software at http://hunch.net/~vw