Solving combinatorial problems at scale

Srikrishna Sridhar

www.cs.wisc.edu/~srikris

Joint work with
Victor Bittorf, Ji Liu, Stephen J. Wright, Chris Ré
One Slide Summary

• Applications
• Optimization toolbox Example
  – Linear Programming
  – Support Vector Machines (SVM)
  – Linear Systems
• Asynchronous Algorithms
• Performance Evaluation
• Future work
One Slide Motivation

Optimization provides a powerful toolbox for data analysis and machine learning problems.

Systems interactions with optimization research
Multicore and clusters

Take away message
Asynchronous algorithms are the key to speedups in modern architectures.
Classification: SVM
Matrix Completion

<p>| | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

User → Movie
Matrix Completion

\[
\begin{array}{cccc}
4 & 1 & 4 & 1 \\
4 & 1 & 4 & 1 \\
4 & 5 & 2 & 2 \\
\end{array}
\]
= 

\[
\begin{array}{cccc}
4 & 1 & 4 & 1 \\
4 & 1 & 4 & 1 \\
4 & 5 & 2 & 2 \\
\end{array}
\]

= 

\[
\begin{array}{cccc}
4 & 1 & 4 & 1 \\
4 & 1 & 4 & 1 \\
4 & 5 & 2 & 2 \\
\end{array}
\]
Linear Systems

Seismic imaging process

1. Data collection from seismic survey
2. Data pre-processing
3. Choosing modeling methodology
4. **Inverse model to fit data**

$$Ax = b$$

Slide Source: Rashmi Raghu
Pictures of natural objects are not random!
They usually have areas of non-constant intensity with sharp edges.
Optimization

Several problems in data analysis, machine learning and predictive analytics can be posed as an optimization problem.
Data: Observations (typically known) which we used to aid decision making.

Decision Variables: The set of decisions that we are seeking to optimize.

Objective: A mathematical quantification of the quality of outcomes made by decisions.

Model: The relationship between the decisions we are trying to make, the outcome and the
Optimization in Analytics

- **BIG DATA**

- **Iterative:** Need to make several passes over the data.

- **Accuracy:** Sometimes, approximate is good enough.

- **Structure:** Seek solutions with a desired structure (like simplicity, robustness etc.)
How big is BIG DATA?

<table>
<thead>
<tr>
<th>Sl.</th>
<th>Variables</th>
<th>Size</th>
<th>Solve Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cplex(B)</td>
</tr>
<tr>
<td>1</td>
<td>123</td>
<td>0.2 MB</td>
<td>0.031</td>
</tr>
<tr>
<td>2</td>
<td>12596</td>
<td>1.2 GB</td>
<td>5882.5</td>
</tr>
<tr>
<td>3</td>
<td>129136</td>
<td>125 GB</td>
<td>--</td>
</tr>
</tbody>
</table>

S: Simplex  
B: Barrier (Interior point method)  
Note: -- indicates timed out at 2 hours
How big is BIG DATA?

<table>
<thead>
<tr>
<th>Sl.</th>
<th>Variables</th>
<th>Size</th>
<th>Solve Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cplex(B)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cplex(S)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Gurobi(B)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Gurobi(S)</td>
</tr>
<tr>
<td>1</td>
<td>123</td>
<td>0.2 MB</td>
<td>0.031</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.065</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.081</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.03</td>
</tr>
<tr>
<td>2</td>
<td>12596</td>
<td>1.2 GB</td>
<td>5882.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1690.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3001.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2707.8</td>
</tr>
<tr>
<td>3</td>
<td>129136</td>
<td>125 GB</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>--</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>--</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>--</td>
</tr>
</tbody>
</table>

S: Simplex  
B: Barrier (Interior point method)  
Note: -- indicates timed out at 2 hours
## How big is BIG DATA?

<table>
<thead>
<tr>
<th>Sl.</th>
<th>Variables</th>
<th>Size</th>
<th>Solve Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cplex(B)</td>
</tr>
<tr>
<td>1</td>
<td>123</td>
<td>0.2 MB</td>
<td>0.031</td>
</tr>
<tr>
<td>2</td>
<td>12596</td>
<td>1.2 GB</td>
<td>5882.5</td>
</tr>
<tr>
<td>3</td>
<td>129136</td>
<td>125 GB</td>
<td>--</td>
</tr>
</tbody>
</table>

Gradient Descent Methods

\[
\min f(x)
\]

\[
x_{k+1} = x_k - \alpha_k \nabla f(x_k)
\]
Gradient Methods

- Computing the gradient requires an entire pass on the data!
- We need to make several passes over the data.
- Hard to parallelize

\[ x_{k+1} = x_k - \alpha_k \nabla f(x_k) \]
Co-ordinate Descent Methods (Nestrov 2010)

min \( f(x) \)

Gradient along a co-ordinate

\[ x_{k+1}^i = x_k^i - \alpha_k \left[ \nabla f(x_k) \right]_i \]

Update only a single co-ordinate
Co-ordinate Descent Methods

- Computing the ‘partial’ gradient requires only a single row (or column) of the data!
- Converges slowly. We need several passes over the data.
- Possible to parallelize

\[ \min f(x) \]

Gradient along a co-ordinate

\[ x_{k+1}^i = x_k^i - \alpha_k [\nabla f(x_k)]_i \]

Update only a single co-ordinate
Co-ordinate Descent Methods

- Computing the ‘partial’ gradient requires only a single row (or column) of the data!
- Converges slowly. We need several passes over the data.
- Possible to parallelize

\[
\min f(x)
\]

Gradient along a co-ordinate:

\[
x_{k+1}^i = x_k^i - \alpha_k \left[ \nabla f(x_k) \right]_i
\]

Update only a single co-ordinate
Stochastic Gradient Methods

- Computing the partial gradients are cheap!
- Converges very slowly. We need several passes over the data.
- Possible to parallelize

\[
\min \sum_{i=1}^{n} f_i(x)
\]

Gradient of a single function

\[
x_{k+1} = x_k - \alpha_k \nabla f_i(x_k)
\]

Update only a single co-ordinate
Parallel Co-ordinate Descent Methods

• Each core grabs a centrally located x, evaluates the gradient and then writes the update back to x.

• Updates may be old by the time they are applied.

• Processors don’t overwrite each other’s work.
Parallel Co-ordinate Descent Methods

Each processor
1. Pick a co-ordinate $i$
2. Read the current state of $x_k$
3. Compute the gradient along the
   
   $[\nabla f(x_k)]_i$
Parallel Co-ordinate Descent Methods

**Global locking:** Lock the shared memory \( x \) for reading and writing operations. Cores acquire the lock in a round robin fashion. (Langford 2009)

**Global locking:** Lock the shared memory \( x \) for reading and writing operations. Cores acquire a lock in any order.

**Asynchronous:** No locking! Cores may overwrite
Comparison of Parallel SCD schemes
Algorithmic & Implementation Speedups
What does Async buy us?

<table>
<thead>
<tr>
<th>Sl.</th>
<th>Variables</th>
<th>Size</th>
<th>Solve Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cplex(B)</td>
</tr>
<tr>
<td>1</td>
<td>123</td>
<td>0.2 MB</td>
<td>0.031</td>
</tr>
<tr>
<td>2</td>
<td>12596</td>
<td>1.2 GB</td>
<td>5882.5</td>
</tr>
<tr>
<td>3</td>
<td>129136</td>
<td>125 GB</td>
<td>--</td>
</tr>
</tbody>
</table>

S: Simplex  
B: Barrier (Interior point method)  
Note: -- indicates timed out at 2 hours
Extreme Linear Programming

Can we leverage async algorithms for large scale combinatorial problems?
LP Rounding

1. Formulate Integer Program (IP)
2. Relax IP to form a Linear Program (LP)
3. Solve the LP
4. Round LP solution

**difficult step!**
LP Rounding: Examples

vertex cover

multiway cut

independent set
Results: Quality & Speed

Solution quality = Rounded solution / Optimal solution
Results reported on Vertex cover
## Results: Quality & Speed

<table>
<thead>
<tr>
<th>instance</th>
<th>type</th>
<th>Cplex IP</th>
<th>Cplex LP</th>
<th>Us</th>
</tr>
</thead>
<tbody>
<tr>
<td>frb59-26-1</td>
<td>VC</td>
<td>-</td>
<td>5.1</td>
<td>0.65</td>
</tr>
<tr>
<td>Amazon</td>
<td>VC</td>
<td>44</td>
<td>22</td>
<td>4.7</td>
</tr>
<tr>
<td>DBLP</td>
<td>VC</td>
<td>23</td>
<td>21</td>
<td>3.2</td>
</tr>
<tr>
<td>Google+</td>
<td>VC</td>
<td>-</td>
<td>62</td>
<td>6.2</td>
</tr>
<tr>
<td>LiveJournal</td>
<td>VC</td>
<td>-</td>
<td>-</td>
<td>934</td>
</tr>
<tr>
<td>frb59-26-1</td>
<td>MC</td>
<td>54</td>
<td>360</td>
<td>29</td>
</tr>
<tr>
<td>Amazon</td>
<td>MC</td>
<td>-</td>
<td>-</td>
<td>131</td>
</tr>
<tr>
<td>DBLP</td>
<td>MC</td>
<td>-</td>
<td>-</td>
<td>158</td>
</tr>
<tr>
<td>Google+</td>
<td>MC</td>
<td>-</td>
<td>-</td>
<td>570</td>
</tr>
</tbody>
</table>

Note: - indicates timed out at 1 hours
Conclusion

Optimization provides a powerful toolbox for data analysis and machine learning problems.

Take away message

Asynchronous algorithms may be the key to speedups in modern architectures.