Parallel & Distributed Optimization

Based on Mark Schmidt’s slides
Motivation behind using parallel & Distributed optimization

● **Performance**
  ○ Computational throughput have increased exponentially in linear time (Moore’s law)
  ○ But only so many transistors can fit in limited space (atomic size)
  ○ Serial computation throughput plateaued (Moore’s law coming to an end)

● **Space**
  ○ Large datasets cannot fit on a single machine
Introduction

- Parallel Computing
  - One machine
    - Multiple processors (Quad-Core, GPU)
Introduction

- **Parallel Computing**
  - One machine
    - Multiple processors (Quad-Core, GPU)

- **Distributed Computing**
  - Multiple computers, linked via network
Introduction

- **Parallel Computing**
  - One machine
    - Multiple processors (Quad-Core, GPU)
- **Distributed Computing**
  - Multiple computers, linked via network
Distributed optimization

- **Strategy**
  - Each machine handles a subset of the dataset

- **Issues**
  - Link failures between machines
    - Devise algorithms that limits communication
    - Decentralize optimization

- **Synchronization**
  - Wait for slowest machine when all machines depend on the variable coordinates
Distributed optimization

**Strategy**
- Each machine handles a subset of the dataset

**Issues**
- link failures between machines
- synchronization
  - wait for the slowest machine to complete processing its data

**Solutions**
- devise algorithms that limits communication
- decentralize optimization
Straightforward distributed optimization

- Run different algorithms/strategies on different machines/cores
  - First one that finishes wins
    - Gauss-Southwell Coordinate Descent
    - Randomized Coordinate Descent
    - Gradient Descent
    - Stochastic Gradient Descent
Straightforward distributed optimization

- Run different algorithms/strategies on different machines/cores
  - First one that finishes wins
    - Gauss-Southwell Coordinate Descent
    - Randomized Coordinate Descent
    - Gradient Descent
    - Stochastic Gradient Descent

Your computer
Parallel first-order methods

- Synchronized deterministic Gradient Descent

\[
\min_{x \in \mathbb{R}^p} \{ F(x) = \frac{1}{n} \sum_{i=1}^{n} F_i(x) \}
\]

- Can be broken into separable components

\[
\frac{1}{n} \sum_{i}^{n} \nabla F_i(x) = \frac{1}{N} \left( \sum_{i=1}^{n/m} \nabla F_i(x) + \sum_{i=n/m+1}^{2n/m} \nabla F_i(x) + \ldots \right)
\]
Parallel first-order methods

- Synchronized deterministic Gradient Descent

\[
\min_{x \in \mathbb{R}^p} \left\{ F(x) = \frac{1}{n} \sum_{i=1}^{n} F_i(x) \right\}
\]

\[
\frac{1}{n} \sum_{i}^{n} \nabla F_i(x) = \frac{1}{N} \left( \sum_{i=1}^{n/m} \nabla F_i(x) + \sum_{i=n/m+1}^{2n/m} \nabla F_i(x) + \ldots \right)
\]

n samples
m machines
Parallel first-order methods

- Synchronized deterministic Gradient Descent

\[ \min_{x \in \mathbb{R}^p} \left\{ F(x) = \frac{1}{n} \sum_{i=1}^{n} F_i(x) \right\} \]

\[ \frac{1}{n} \sum_{i=1}^{n} \nabla F_i(x) = \frac{1}{N} \left( \sum_{i=1}^{n/m} \nabla F_i(x) + \sum_{i=n/m+1}^{2n/m} \nabla F_i(x) + \ldots \right) \]

n samples
m machines
Parallel first-order methods

- Synchronized deterministic Gradient Descent

\[
\min_{x \in \mathbb{R}^p} \left\{ F(x) = \frac{1}{n} \sum_{i=1}^{n} F_i(x) \right\}
\]

\[
x^{t+1} = x^t - \alpha \frac{1}{n} \sum_{i} \nabla F_i(x^t)
\]

\[
\frac{1}{n} \sum_{i}^{n} \nabla F_i(x) = \frac{1}{N} \left( \sum_{i=1}^{n/m} \nabla F_i(x) + \sum_{i=n/m+1}^{2n/m} \nabla F_i(x) + \ldots \right)
\]

\(m\) machines

\(n\) samples
Parallel first-order methods

- Synchronized deterministic Gradient Descent

\[ \min_{x \in \mathbb{R}^p} \left\{ F(x) = \frac{1}{n} \sum_{i=1}^{n} F_i(x) \right\} \]

- These allow optimal linear speedups
  - You should always consider this first!
Parallel first-order methods

- Synchronized deterministic Gradient Descent

\[
\min_{x \in \mathbb{R}^p} \left\{ F(x) = \frac{1}{n} \sum_{i=1}^{n} F_i(x) \right\}
\]

- Issue
  - What if one of the computers is very slow?
  - What if one of the links failed?
Centralized Gradient descent (HogWild)

- Update ‘x’ asynchronously - saves a lot of time
- Stochastic gradient method on shared memory

\[ x^{t+1} = x^t - \alpha \nabla f_{i_n}(x^{t-d}) \]
Centralized Gradient descent (HogWild)

- Update ‘x’ asynchronously - saves a lot of time
- Stochastic gradient method on shared memory

\[ x^{t+1} = x^t - \alpha \nabla f_{i_n}(x^{t-d}) \]
Centralized Gradient descent

- Update ‘x’ asynchronously - saves a lot of time
- Stochastic gradient method on shared memory

\[ x^{t+1} = x^t - \alpha \nabla f_{i_n}(x^{t-d}) \]
Centralized Gradient descent

- Update ‘x’ asynchronously - saves a lot of time
- Stochastic gradient method on shared memory

\[ x^{t+1} = x^t - \alpha \nabla f_{i_1}(x^{t-d}) \]

\[ -\alpha \nabla f_{i_2}(x^{t-d}) \]
Centralized Gradient descent

- Update ‘x’ asynchronously - saves a lot of time
- Stochastic gradient method on shared memory

$x^{t+1} = x^t - \alpha \nabla f_{i_n}(x^{t-d})$
Centralized Gradient descent

- Update ‘x’ asynchronously - saves a lot of time
- Stochastic gradient method on shared memory

\[ x^{t+1} = x^t - \alpha \nabla f_{i_0}(x^{t-d}) \]

- \( -\alpha \nabla f_{i_1}(x^{t-d}) \)
- \( -\alpha \nabla f_{i_2}(x^{t-d}) \)
- \( -\alpha \nabla f_{i_3}(x^{t-d}) \)
Centralized Coordinate descent

- Communicating parameters ‘x’ can be expensive
- Use coordinate descent to transmit one coordinate update at a time

\[ x_1 = x_1 - \alpha \nabla_1 f(x) \]
Centralized Coordinate descent

- Communicating parameters ‘x’ can be expensive

Sending and receiving `x` parameters is expensive
Centralized Coordinate descent

- Communicating parameters ‘x’ can be expensive
- Use coordinate descent to transmit one coordinate update at a time

\[ x_1 = x_1 - \alpha \nabla f(x) \]
Centralized Coordinate descent

- Communicating parameters ‘x’ can be expensive
- Use coordinate descent to transmit one coordinate update at a time

\[ x_1 = x_1 - \alpha\nabla_1 f(x) \]
\[ x_2 = x_2 - \alpha\nabla_2 f(x) \]
Centralized Coordinate descent

- Communicating parameters ‘x’ can be expensive
- Use coordinate descent to transmit one coordinate update at a time

- Need to decrease step-size for convergence (it’s stochastic coordinate descent).
Decentralized Coordinate descent for sparse datasets

- Least square problem
  \[ f(x) = \frac{1}{2} ||Ax - b||^2 \]

- Update rule
  \[ x_{i}^{t+1} = x_{i}^{t} - \alpha A^i (Ax - b) \]

- Doesn’t seem separable at first sight.
  - But, \( A^i \) can have many non-zero entries - most entries in \((Ax - b)\) will be unnecessary
Decentralized Coordinate descent for sparse datasets

- Least square problem
  \[ f(x) = \frac{1}{2} \|Ax - b\|^2 \]

- Update rule
  \[ x_i^{t+1} = x_i^t - \alpha A_i (Ax - b) \]

Sparse \( A \):

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Decentralized Coordinate descent for sparse datasets

- Least square problem
  \[ f(x) = \frac{1}{2} ||Ax - b||^2 \]

- Update rule
  \[ x_i^{t+1} = x_i^t - \alpha A_i^i (Ax - b) \]

Sparse \( A \)

\[
\begin{array}{cccc}
0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 \\
\end{array}
\]
Decentralized Coordinate descent for sparse datasets

- Least square problem
  \[ f(x) = \frac{1}{2} \|Ax - b\|^2 \]

- Update rule
  \[ x_{i}^{t+1} = x_{i}^{t} - \alpha A_i (Ax - b) \]

Sparse \( A \):

\[
\begin{array}{cccc}
0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 \\
\end{array}
\]
Decentralized Coordinate descent for sparse datasets

- Least square problem
  \[ f(x) = \frac{1}{2}||Ax - b||^2 \]

- Update rule
  \[ x^t_{i} + 1 = x^t_{i} - \alpha A^i (Ax - b) \]

Sparse \( A \):

\[
\begin{array}{cccc}
0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 \\
\end{array}
\]

Coordinates 1 & 4
Samples 2 & 3

Coordinate 2
Sample 1

Coordinate 3
Sample 4
Decentralized Coordinate descent for sparse datasets

- Update rule
  \[ x_i^{t+1} = x_i^t - \alpha A_i^i (Ax - b) \]

- Say, you can only fit one sample in the machine

<table>
<thead>
<tr>
<th>Coordinates 1 &amp; 4</th>
<th>Coordinate 2</th>
<th>Coordinate 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Samples 2 &amp; 3</td>
<td>Sample 1</td>
<td>Sample 4</td>
</tr>
</tbody>
</table>

Sparse A

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Decentralized Coordinate descent for sparse datasets

- Update rule

\[ x_i^{t+1} = x_i^t - \alpha A_i(Ax - b) \]

- Say, you can only fit one sample in a machine

Sparse \( A \)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Decentralized Coordinate descent for sparse datasets

- Update rule
  \[ x_{i}^{t+1} = x_{i}^{t} - \alpha A_{i} (Ax - b) \]

- Say, you can only fit \textbf{one} sample in a machine

Sparse \( A \):
\[
\begin{pmatrix}
0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0
\end{pmatrix}
\]
Decentralized Gradient Descent

- Distribute the data across machines.
- We may not want to update a ‘centralized’ vector $x$
- **Decentralized Gradient Descent**
  - Each machine has its own data samples

Sparse $A$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Decentralized Gradient Descent

- Distribute the data across machines.
- We may not want to update a ‘centralized’ vector $x$

**Decentralized Gradient Descent**
- Each machine has its own data samples

**Sparse $A$**

```
0  1  0  0
1  0  0  0
1  0  0  1
0  0  1  0
```
Decentralized Gradient Descent

- Distribute the data across machines.
- We may not want to update a ‘centralized’ vector $x$
- **Decentralized Gradient Descent**
  - Each machine has its own data samples
  - Each machine has its own parameter vector $x_m$

```
Sparse A

<table>
<thead>
<tr>
<th></th>
<th>Sample 1</th>
<th>Sample 2</th>
<th>Sample 3</th>
<th>Sample 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample 1</td>
<td>0  1  0  0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sample 2</td>
<td>1  0  0  0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sample 3</td>
<td>1  0  0  1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sample 4</td>
<td>0  0  1  0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Sample 1  Sample 2  Sample 3  Sample 4

Machine 1  Machine 2  Machine 3  Machine 4
Decentralized Gradient Descent

- Distribute the data across machines.
- We may not want to update a ‘centralized’ vector $x$

**Decentralized Gradient Descent**

- Each machine has its own data samples
- Each machine has its own parameter vector $x_m$

<table>
<thead>
<tr>
<th>Sample 1</th>
<th>Sample 2</th>
<th>Sample 3</th>
<th>Sample 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 0 0</td>
<td>1 0 0 0</td>
<td>1 0 0 1</td>
<td>0 0 1 0</td>
</tr>
</tbody>
</table>

Sparse $A$

$x_1 = \ ? \quad x_2 = \ ? \quad x_3 = \ ? \quad x_4 = \ ?$

Machine 1  Machine 2  Machine 3  Machine 4
Decentralized Gradient Descent

- Distribute the data across machines.
- We may not want to update a ‘centralized’ vector $x$
- **Decentralized Gradient Descent**
  - Each machine has its own data samples
  - Each machine has its own parameter vector $x_m$

<table>
<thead>
<tr>
<th>Sparse $A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample 1</td>
</tr>
<tr>
<td>Sample 2</td>
</tr>
<tr>
<td>Sample 3</td>
</tr>
<tr>
<td>Sample 4</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
x_1 &= ? \\
x_2 &= ? \\
x_3 &= ? \\
x_4 &= ?
\end{align*}
\]

Sample 1

Sample 2

Sample 3

Sample 4

Machine 1

Machine 2

Machine 3

Machine 4
Decentralized Gradient Descent

- Distribute the data across machines.
- We may not want to update a ‘centralized’ vector $x$.

**Decentralized Gradient Descent**
- Each machine has its own data samples
- Each machine has its own parameter vector $x_m$

```
<table>
<thead>
<tr>
<th>Sample 1</th>
<th>Sample 2</th>
<th>Sample 3</th>
<th>Sample 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
```

$\begin{align*}
x_1 &= \text{?} \\
x_2 &= \text{?} \\
x_3 &= \text{?} \\
x_4 &= \text{?}
\end{align*}$
Decentralized Gradient Descent

- Distribute the data across machines.
- We may not want to update a ‘centralized’ vector $x$

**Decentralized Gradient Descent**
- Each machine has its own data samples
- Each machine has its own parameter vector $x_m$

<table>
<thead>
<tr>
<th>Sparse $A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample 1</td>
</tr>
<tr>
<td>Sample 2</td>
</tr>
<tr>
<td>Sample 3</td>
</tr>
<tr>
<td>Sample 4</td>
</tr>
</tbody>
</table>
Decentralized Gradient Descent

- Distribute the data across machines.
- We may not want to update a ‘centralized’ vector $x$
- **Decentralized Gradient Descent**
  - Each machine has its own data samples
  - Each machine has its own parameter vector $x_m$

```
<table>
<thead>
<tr>
<th>Sample 1</th>
<th>Sample 2</th>
<th>Sample 3</th>
<th>Sample 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 0 0</td>
<td>1 0 0 0</td>
<td>1 0 0 1</td>
<td>0 0 1 0</td>
</tr>
</tbody>
</table>
```

$x_1 =$ ? $\quad$ $x_2 =$ ? $\quad$ $x_3 =$ ? $\quad$ $x_4 =$ ?

Sample 1 $\quad$ Sample 2 $\quad$ Sample 3 $\quad$ Sample 4

Machine 1 $\quad$ Machine 2 $\quad$ Machine 3 $\quad$ Machine 4
Decentralized Gradient Descent

- Distribute the data across machines.
- We may not want to update a ‘centralized’ vector $x$

**Decentralized Gradient Descent**
- Each machine has its own data samples
- Each machine has its own parameter vector $x_m$

$\begin{align*}
x_1 &= x(2) \\
x_2 &= x([1,4]) \\
x_3 &= x([1,4]) \\
x_4 &= x(3)
\end{align*}$

Sample 1: 0 1 0 0
Sample 2: 1 0 0 0
Sample 3: 1 0 0 1
Sample 4: 0 0 1 0

Sparse $A$
Decentralized Gradient Descent

- Distribute the data across machines.
- We may not want to update a ‘centralized’ vector $x$

**Decentralized Gradient Descent**
- Each machine has its own data samples
- Each machine has its own parameter vector $x_m$
- Update rule

\[
x_m = \frac{1}{|\text{nei}(m)|} \sum_{k \in \text{nei}(m)} x_m - \alpha \sum_{i \in k} \nabla f_i(x_m)
\]

$\begin{pmatrix}
x_1 = x(2) \\
no\ communication
\end{pmatrix}
\begin{pmatrix}
x_2 = x([1, 4]) \\
communicates\ with\ machine\ 3
\end{pmatrix}
\begin{pmatrix}
x_3 = x([1, 4]) \\
communicates\ with\ machine\ 2
\end{pmatrix}
\begin{pmatrix}
x_4 = x(3) \\
No\ communication
\end{pmatrix}

Sparse $A$

| Sample 1 | 0 | 1 | 0 | 0 |
| Sample 2 | 1 | 0 | 0 | 0 |
| Sample 3 | 1 | 0 | 0 | 1 |
| Sample 4 | 0 | 0 | 1 | 0 |
Decentralized Gradient Descent

- Distribute the data across machines.
- We may not want to update a ‘centralized’ vector $x$
- **Decentralized Gradient Descent**
  - Each machine has its own data samples
  - Each machine has its own parameter vector $x_m$
  - Update rule
    $$x_m = \frac{1}{|\text{nei}(m)|} \sum_{k \in \text{nei}(m)} x_m - \alpha \sum_{i \in k} \nabla f_i(x_m)$$
  - Similar convergence to the gradient descent with central communication
    - The rate depends on the sparsity of the dataset

<table>
<thead>
<tr>
<th></th>
<th>Sample 1</th>
<th>Sample 2</th>
<th>Sample 3</th>
<th>Sample 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sparse $A$</td>
<td>0 1 0 0</td>
<td>1 0 0 0</td>
<td>1 0 0 1</td>
<td>0 0 1 0</td>
</tr>
</tbody>
</table>
Summary

- Using parallel and distributed systems is important for speeding up optimization for big data

- **Synchronized Deterministic Gradient Descent**
  - Optimization halts with link failure or when a machine is slow at processing its data

- **Centralized Asynchronous Gradient Descent**
  - Communicating vector ‘x’ is costly

- **Centralized Asynchronous Coordinate descent**
  - Centralization causes additional overhead - communication

- **Decentralized Asynchronous Coordinate descent**
  - Helpful for sparse datasets
  - No communication between machines

- **Decentralized Gradient descent**
  - Helpful for sparse datasets
  - Machines have to communicate with few neighbors only