Parallel & Distributed Optimization

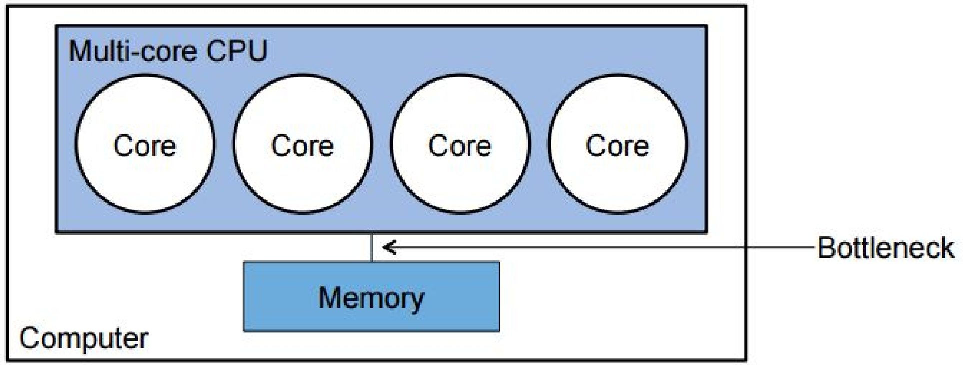
Based on Mark Schmidt’s slides

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

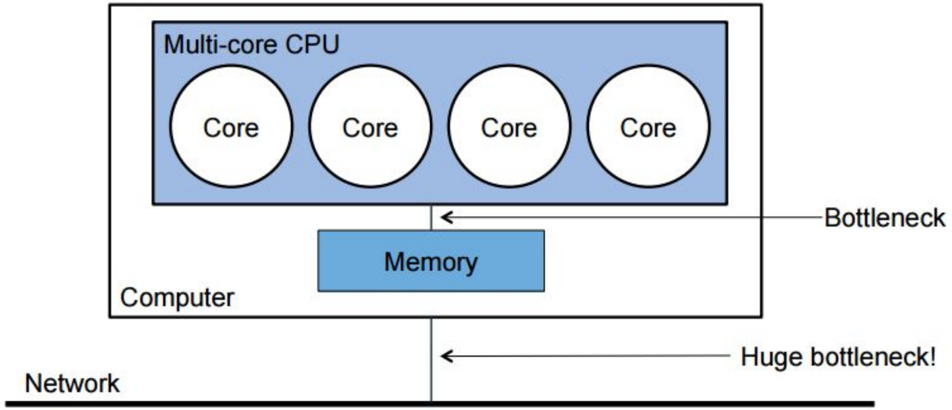
Introduction

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



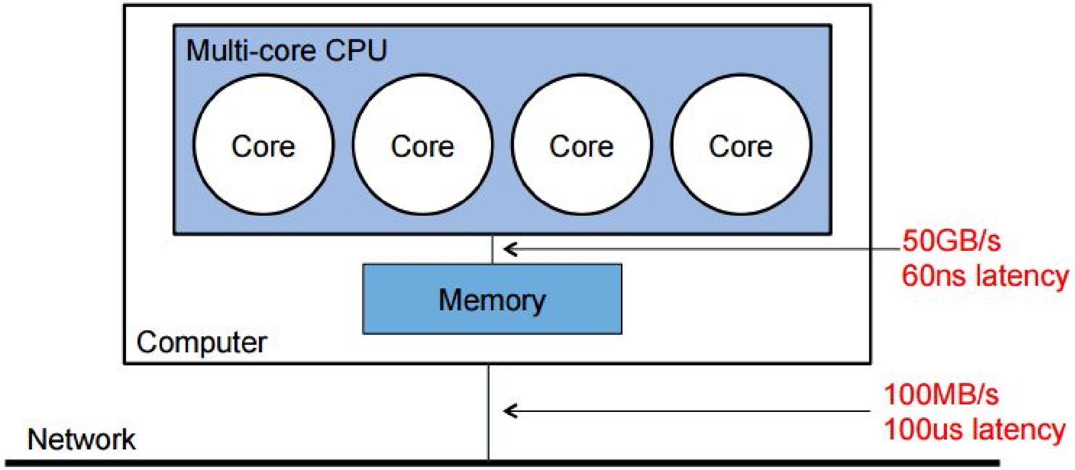
Introduction

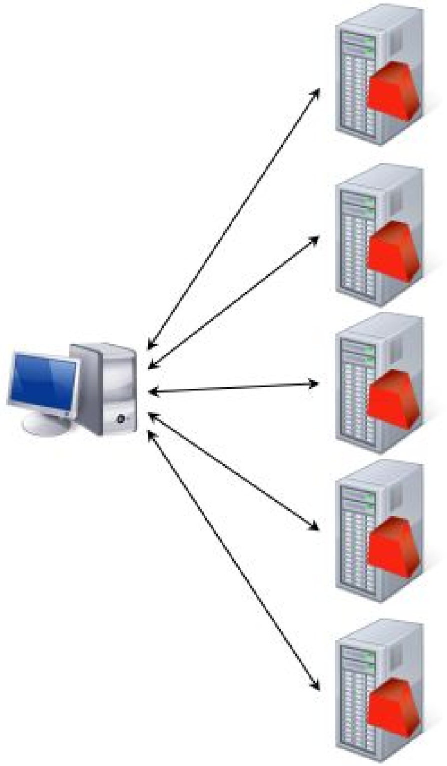
- Parallel Computing
  - One machine
    - Multiple processors (Quad-Core, GPU)
- Distributed Computing
  - Multiple computers, linked via network



Introduction

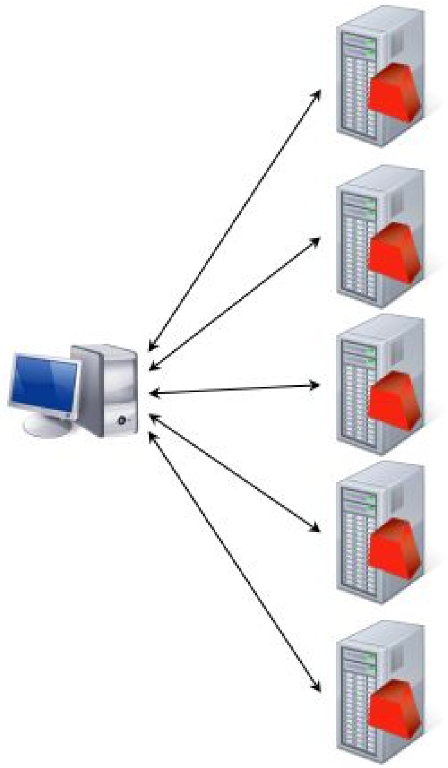
- Parallel Computing
  - One machine
    - Multiple processors (Quad-Core, GPU)
- Distributed Computing
  - Multiple computers, linked via network





Distributed optimization

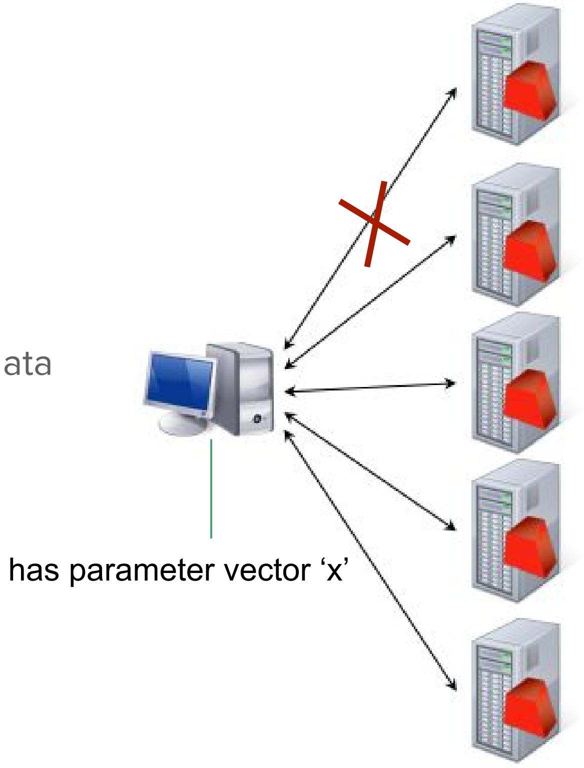
- Strategy
  - Each machine handles a subset of the dataset
- Issues
  - link failures between machines
    - devise algorithms that limits communication
    - decentralize optimization
- Synchronization
  - wait for slowest machine when the all machines depend on the va coordinates



Distributed optimization

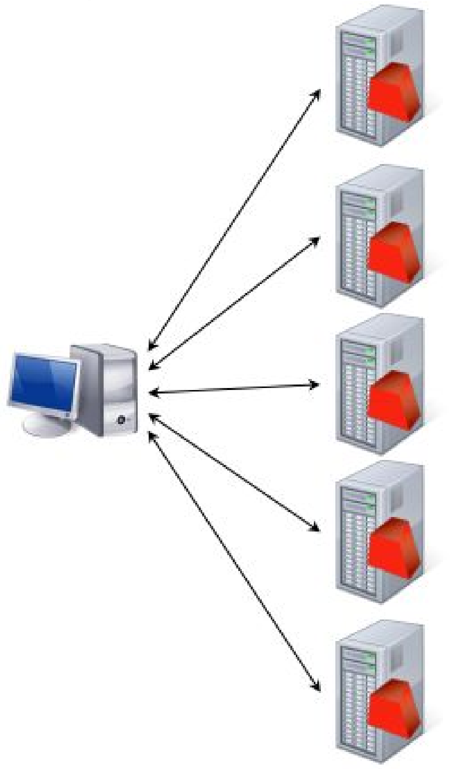
- Strategy
  - Each machine handles a subset of the dataset
- Issues
  - **link failures between machines**
  - **synchronization**
    - wait for the slowest machine to complete processing its data

- Solutions
  - devise algorithms that limits communication
  - decentralize optimization



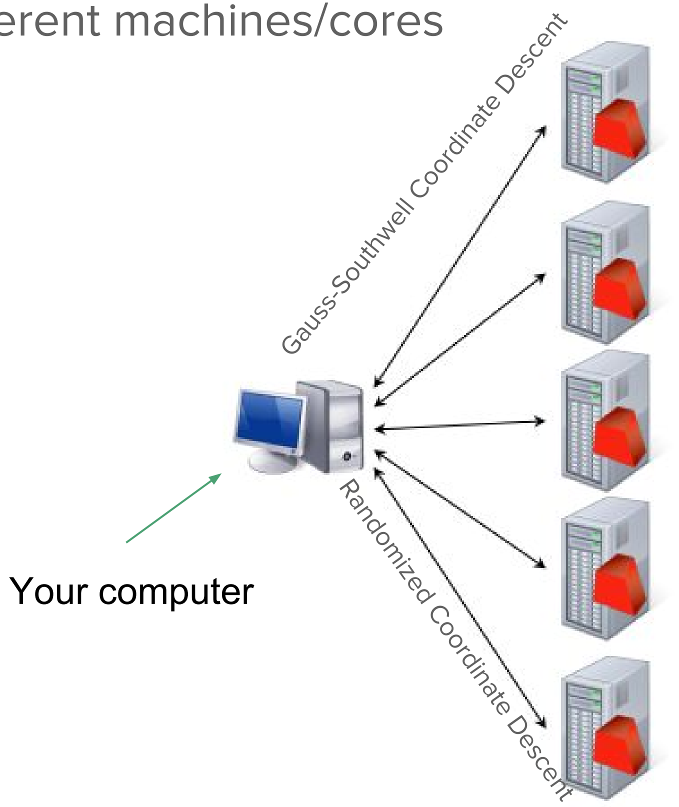
Straightforward distributed optimization

- 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


Gauss-Southwell Coordinate Descent
Your computer
Randomized Coordinate Descent

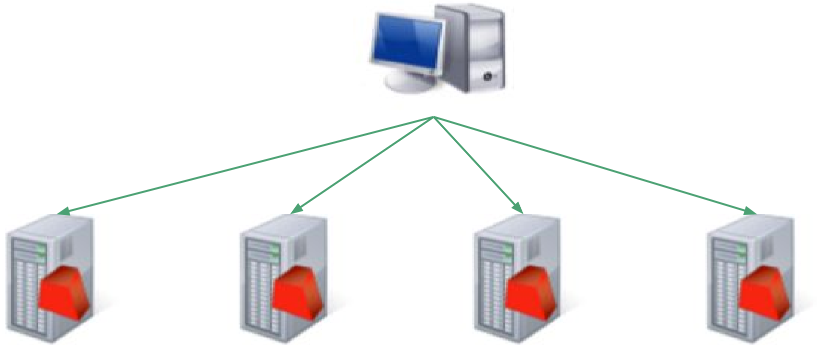
Parallel first-order methods

- Synchronized deterministic Gradient Descent

$$\min_{x \in R^p} \{F(x) = \tfrac{1}{n} \textstyle\sum_{i=1}^{n} F_i(x)\}$$

- Can be broken into separable components

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



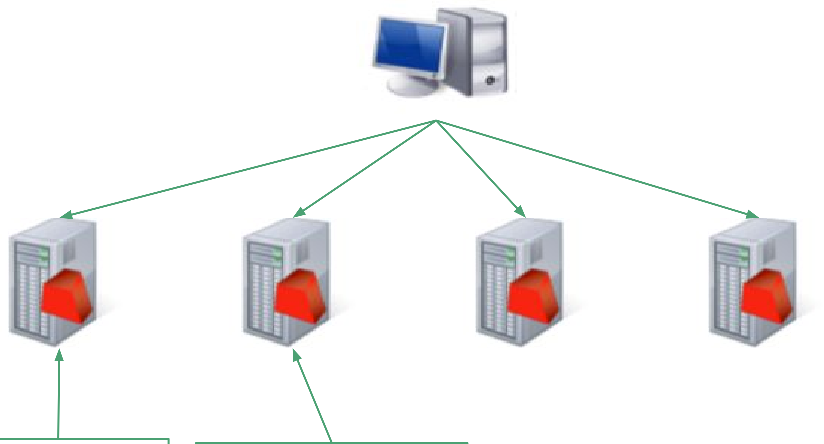
n samples

m machines

Parallel first-order methods

- Synchronized deterministic Gradient Descent

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



$$\frac{1}{n}\sum_{i}^{n}\nabla F_i(x)=\frac{1}{N}\left(\boxed{\sum_{i=1}^{n/m}\nabla F_i(x)}+\boxed{\sum_{i=n/m+1}^{2n/m}\nabla F_i(x)}+\dots\right)$$

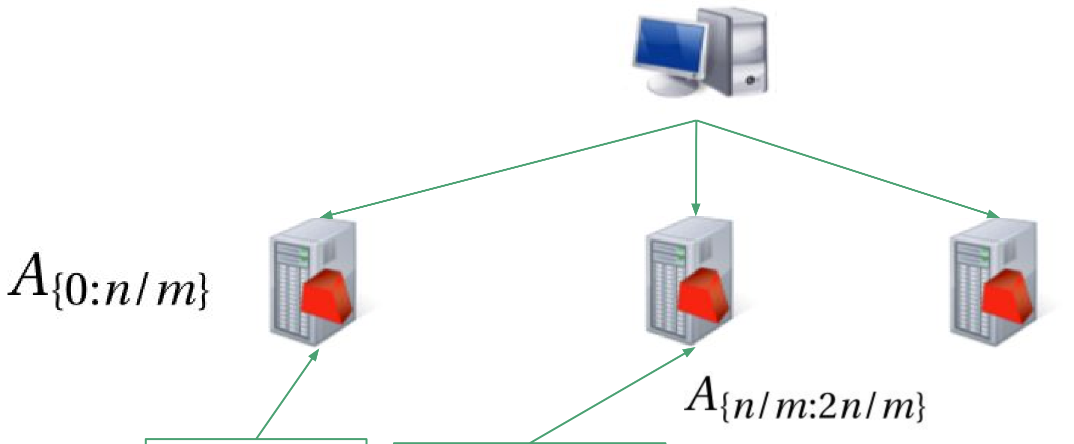
n samples

m machines

Parallel first-order methods

- Synchronized deterministic Gradient Descent

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



$$\frac{1}{n}\sum_{i}^{n} \nabla F_i(x) = \frac{1}{N}\left(\boxed{\sum_{i=1}^{n/m} \nabla F_i(x)} + \boxed{\sum_{i=n/m+1}^{2n/m} \nabla F_i(x)} + \dots \right)$$

n samples

m machines

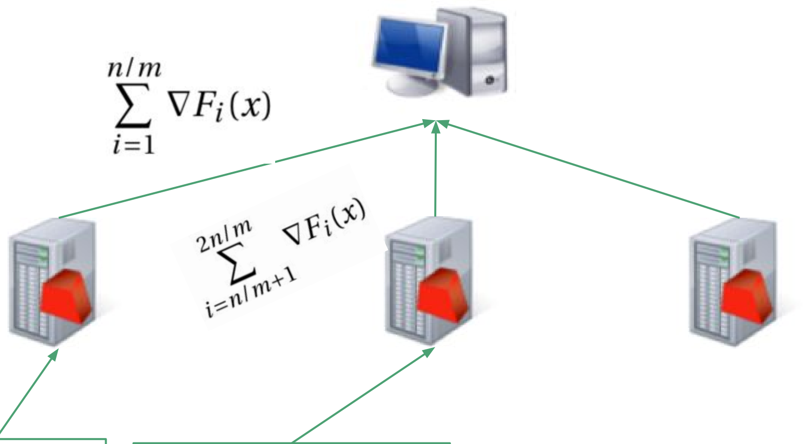
Parallel first-order methods

m machines

- Synchronized deterministic Gradient Descent

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

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


$\sum_{i=1}^{n/m}\nabla F_i(x)$
$\sum_{i=n/m+1}^{2n/m}\nabla F_i(x)$

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

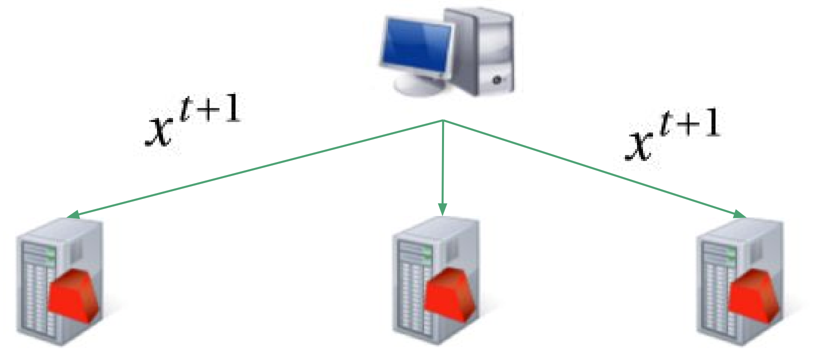
n samples

Parallel first-order methods

m machines

- Synchronized deterministic Gradient Descent

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



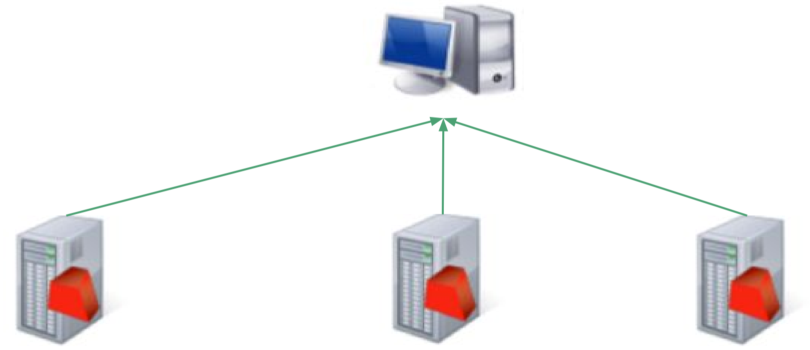
- These allow optimal linear speedups
  - You should always consider this first!

Parallel first-order methods

m machines

- Synchronized deterministic Gradient Descent

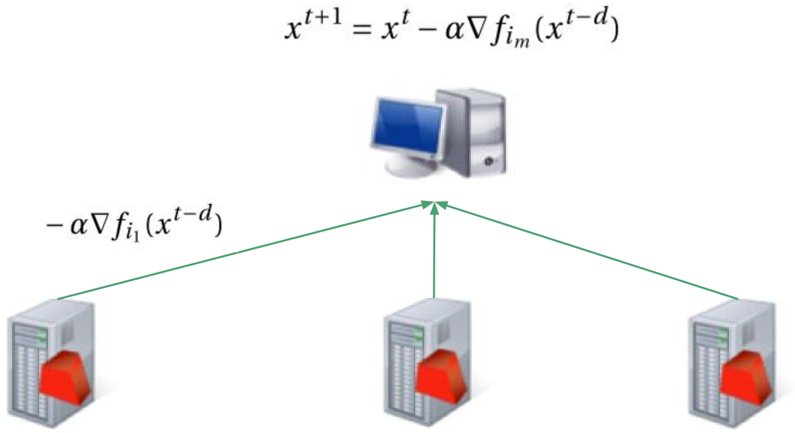
$$\min_{x\in R^p} \left\{F(x) = \frac{1}{n}\sum_{i=1}^{n} F_i(x)\right\}$$



- Issue
    - What if one of the computers is very slow ?
    - What if one of the links failed ?

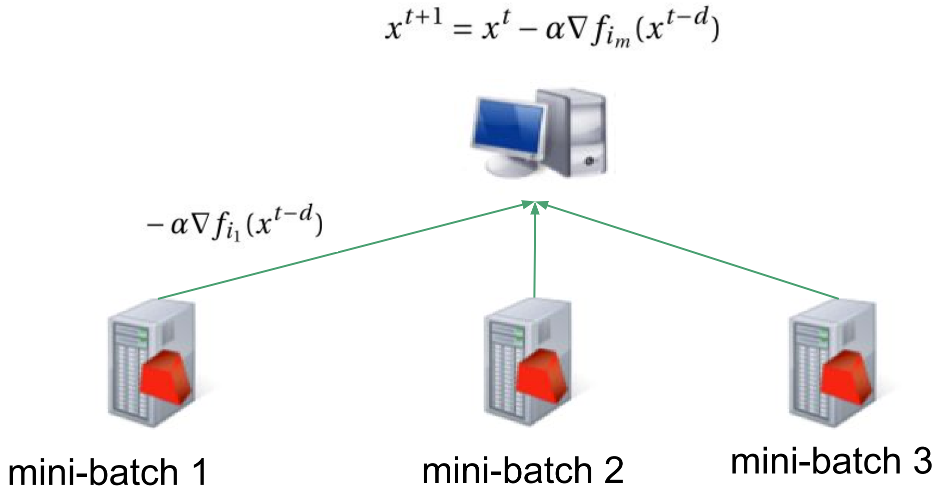
Centralized Gradient descent (HogWild)

- Update *'x'* asynchronously - saves a lot of time
- Stochastic gradient method on shared memory


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

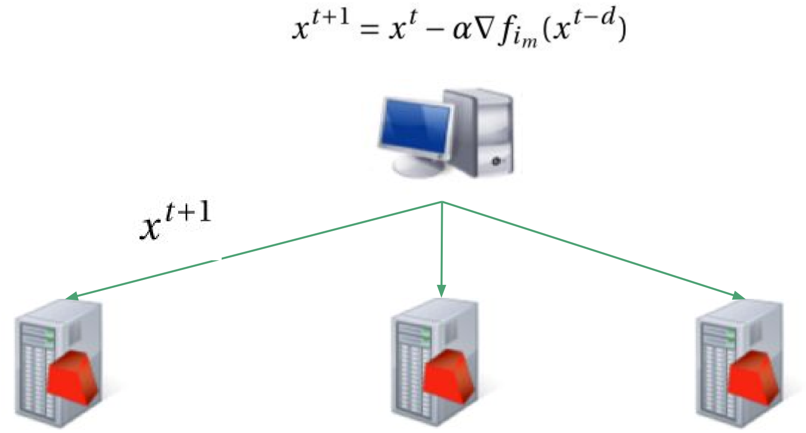
Centralized Gradient descent (HogWild)

- Update '*x*' asynchronously - saves a lot of time
- Stochastic gradient method on shared memory


$x^{t+1} = x^t - \alpha \nabla f_{i_m}(x^{t-d})$
$-\alpha \nabla f_{i_1}(x^{t-d})$
mini-batch 1
mini-batch 2
mini-batch 3

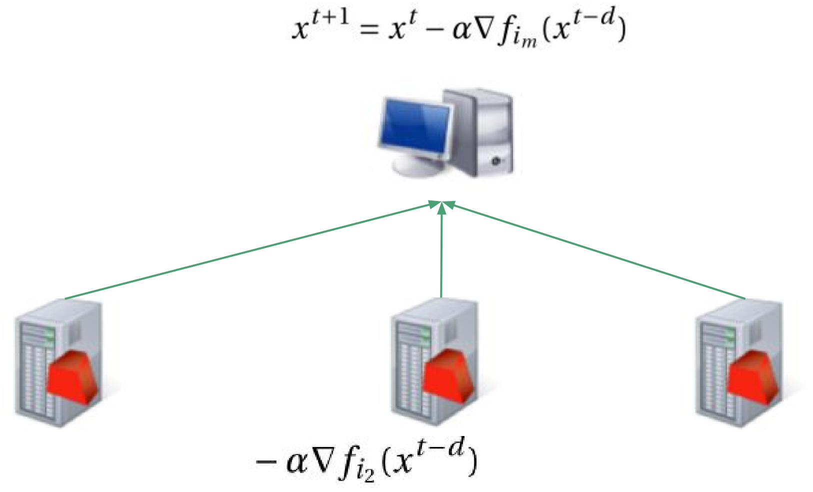
Centralized Gradient descent

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



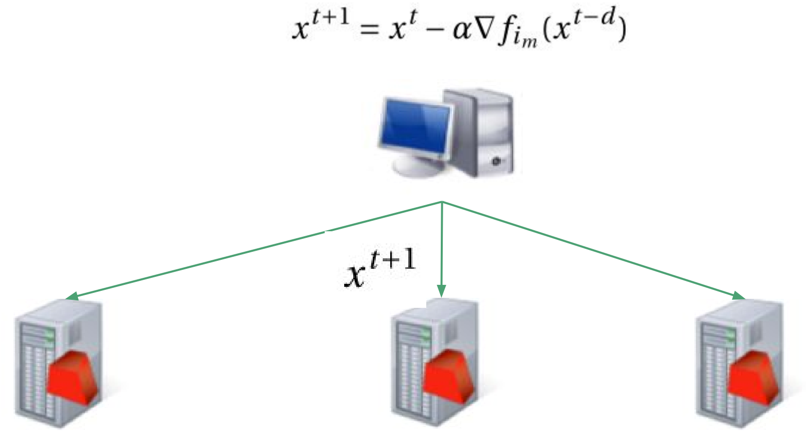
Centralized Gradient descent

- Update '*x*' asynchronously - saves a lot of time
- Stochastic gradient method on shared memory


$x^{t+1} = x^t - \alpha \nabla f_{i_m}(x^{t-d})$
$-\alpha \nabla f_{i_2}(x^{t-d})$

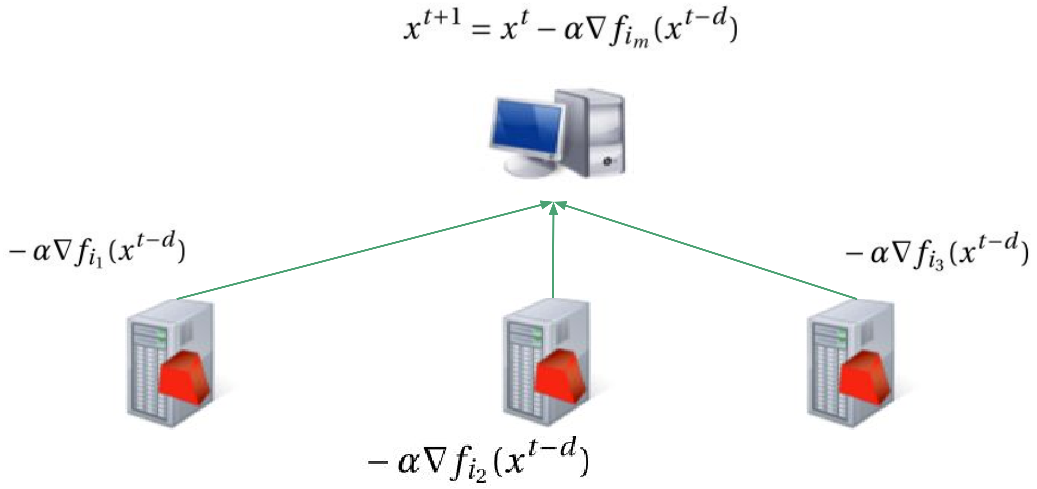
Centralized Gradient descent

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



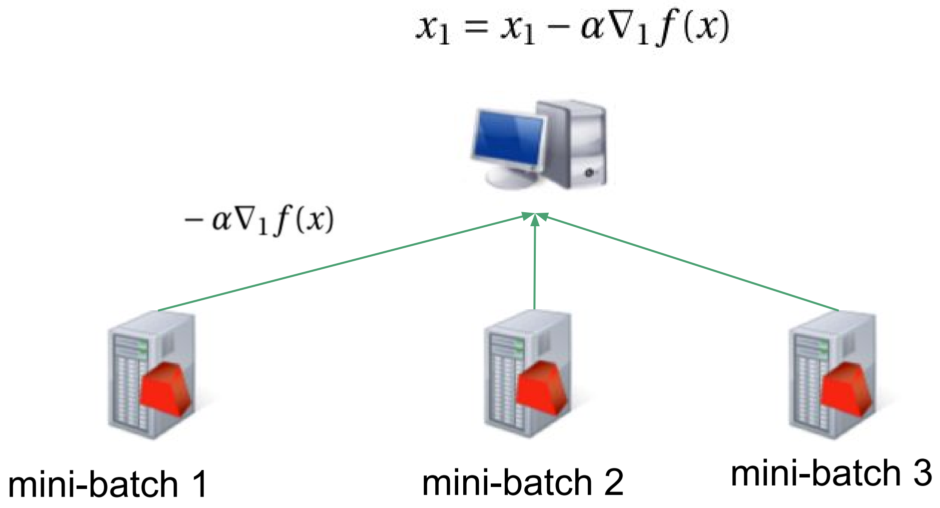
Centralized Gradient descent

- Update *'x'* asynchronously - saves a lot of time
- Stochastic gradient method on shared memory


$x^{t+1} = x^t - \alpha \nabla f_{i_m}(x^{t-d})$
$-\alpha \nabla f_{i_1}(x^{t-d})$
$-\alpha \nabla f_{i_3}(x^{t-d})$
$-\alpha \nabla f_{i_2}(x^{t-d})$

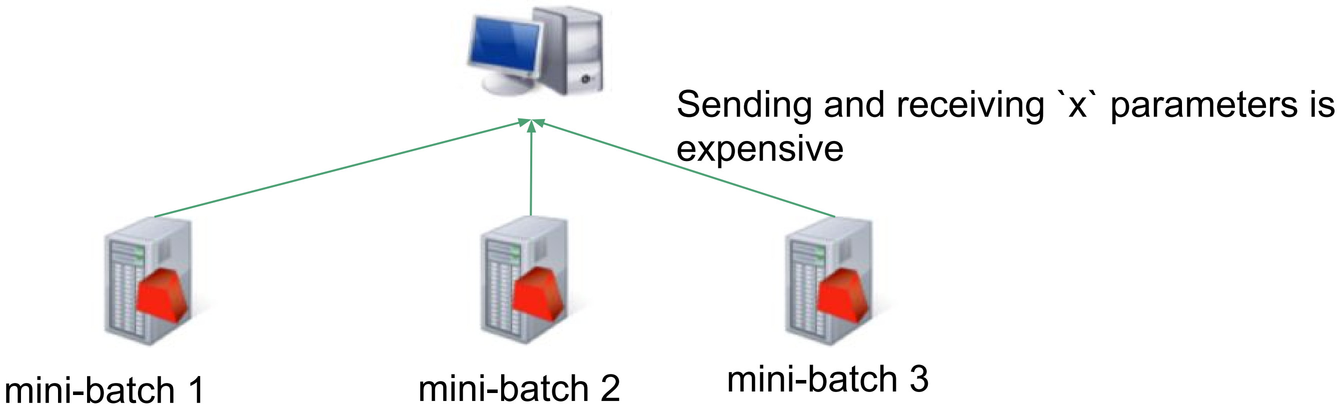
Centralized Coordinate descent

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



Centralized Coordinate descent

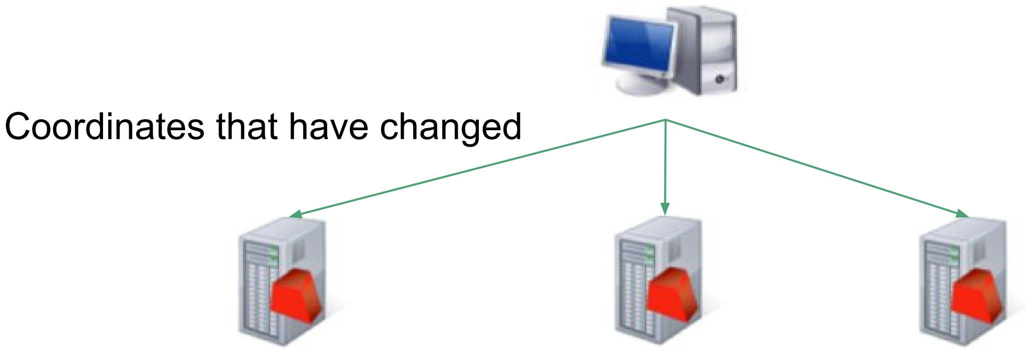
- Communicating parameters ‘*x*’ can be expensive



Centralized Coordinate descent

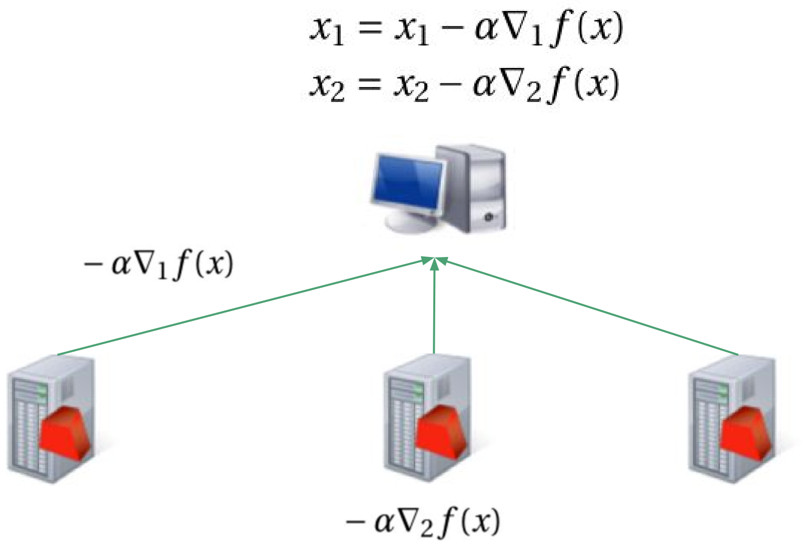
- Communicating parameters ‘*x*’ can be expensive
- Use coordinate descent to transmit one coordinate update at a time

$$x_1 = x_1 - \alpha \nabla_1 f(x)$$



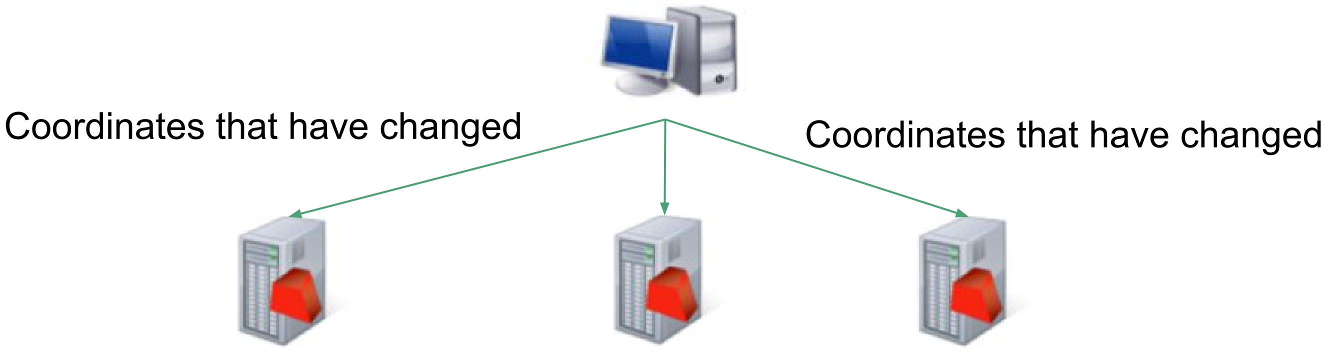
Centralized Coordinate descent

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


$x_1 = x_1 - \alpha\nabla_1 f(x)$
$x_2 = x_2 - \alpha\nabla_2 f(x)$
$-\alpha\nabla_1 f(x)$
$-\alpha\nabla_2 f(x)$

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

$$f(x) = \frac{1}{2}||Ax - b||^2$$

Sparse **A**

0	1	0	0
1	0	0	0
1	0	0	1
0	0	1	0

- 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**

0	1	0	0
1	0	0	0
1	0	0	1
0	0	1	0

Coordinates ? & ?

Coordinate ?

Coordinate ?

Decentralized Coordinate descent for sparse datasets

- Least square problem

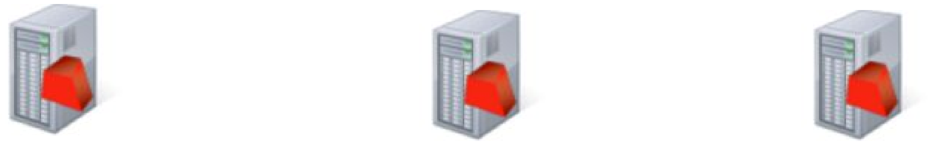
$$f(x) = \frac{1}{2}||Ax - b||^2$$

- Update rule

$$x_i^{t+1} = x_i^t - \alpha A^i (Ax - b)$$

Sparse **A**

0	1	0	0
1	0	0	0
1	0	0	1
0	0	1	0



Decentralized Coordinate descent for sparse datasets

- Least square problem

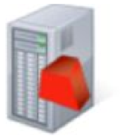
$$f(x) = \frac{1}{2}||Ax - b||^2$$

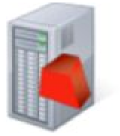
- Update rule

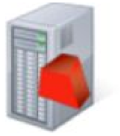
$$x_i^{t+1} = x_i^t - \alpha A^i (Ax - b)$$

Sparse **A**

0	1	0	0
1	0	0	0
1	0	0	1
0	0	1	0


Coordinates ? & ?


Coordinate ?


Coordinate ?

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**

0	1	0	0
1	0	0	0
1	0	0	1
0	0	1	0

Coordinates 1 & 4
Samples 2 & 3

Coordinate 2
Sample 1

Coordinate 3
Sample 4

Decentralized Coordinate descent for sparse datasets

- Update rule

$$x_i^{t+1} = x_i^t - \alpha A^i (Ax - b)$$

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

Coordinates 1 & 4
Samples 2 & 3

Coordinate 2
Sample 1

Coordinate 3
Sample 4

Sparse **A**

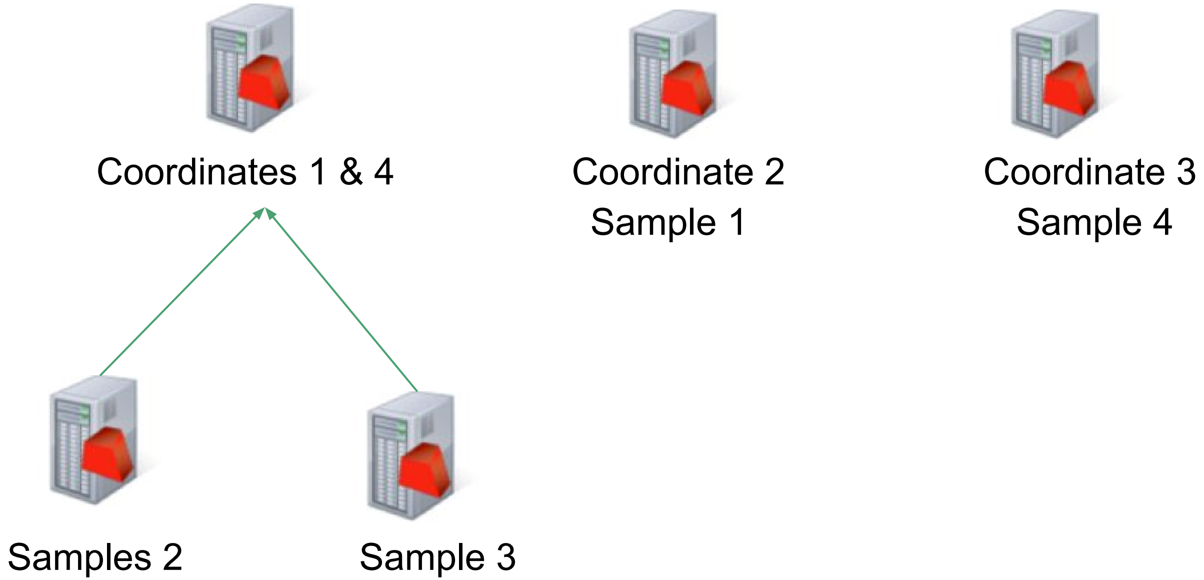
0	1	0	0
1	0	0	0
1	0	0	1
0	0	1	0

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**

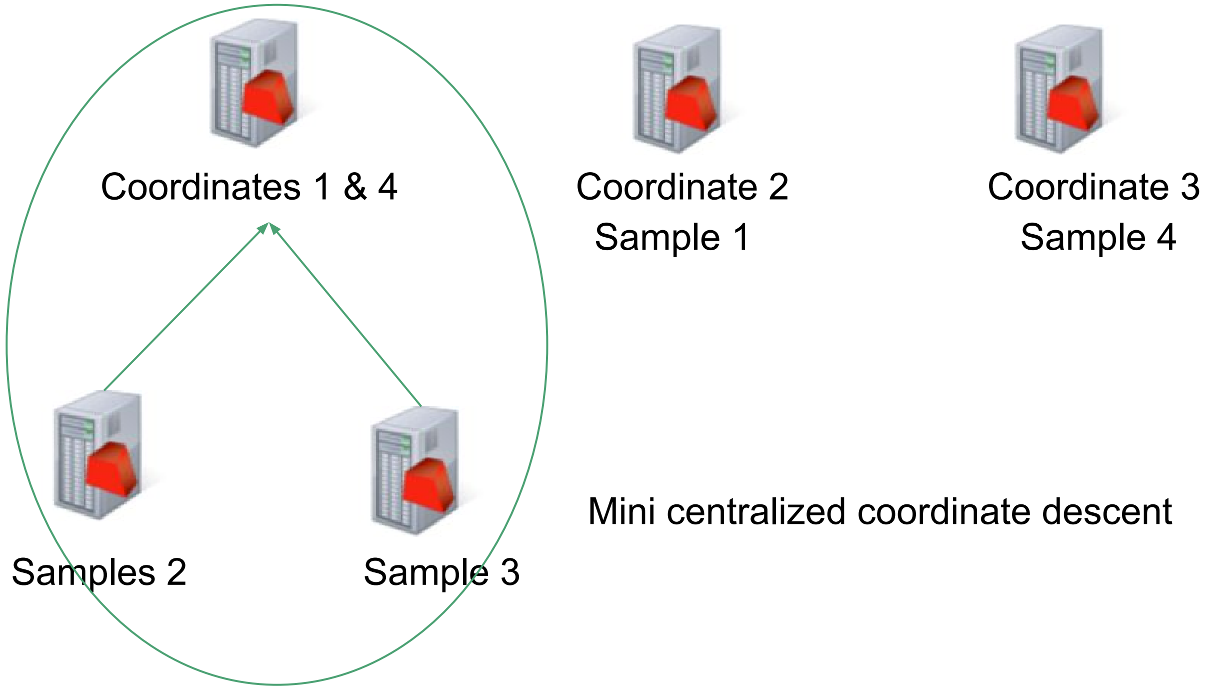
0	1	0	0
1	0	0	0
1	0	0	1
0	0	1	0

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**

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**

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**

Sparse *A*

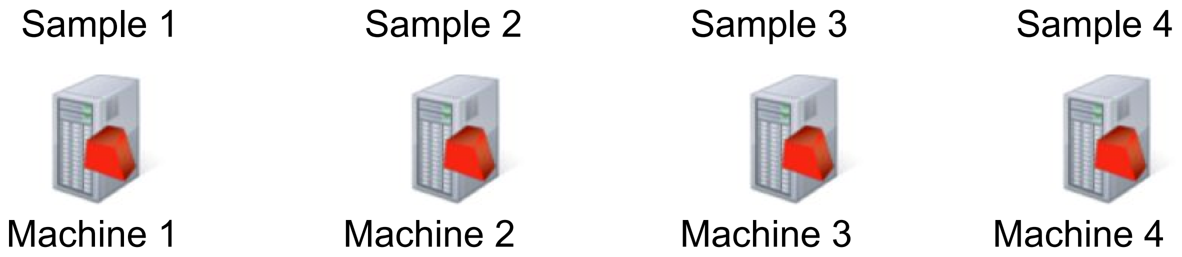
0	1	0	0
1	0	0	0
1	0	0	1
0	0	1	0

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

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

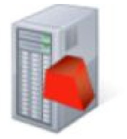
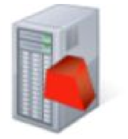
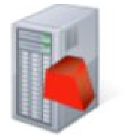
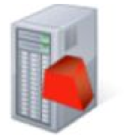
- Distribute the data across machines.
- We may not want to update a ‘centralized’ vector x
- **Decentralized Gradient Descent**
  - **Each machine has its own data samples**
  - **Each machine has its own parameter vector** x_m

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

$x_1 =$? $x_2 =$? $x_3 =$? $x_4 =$?

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

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

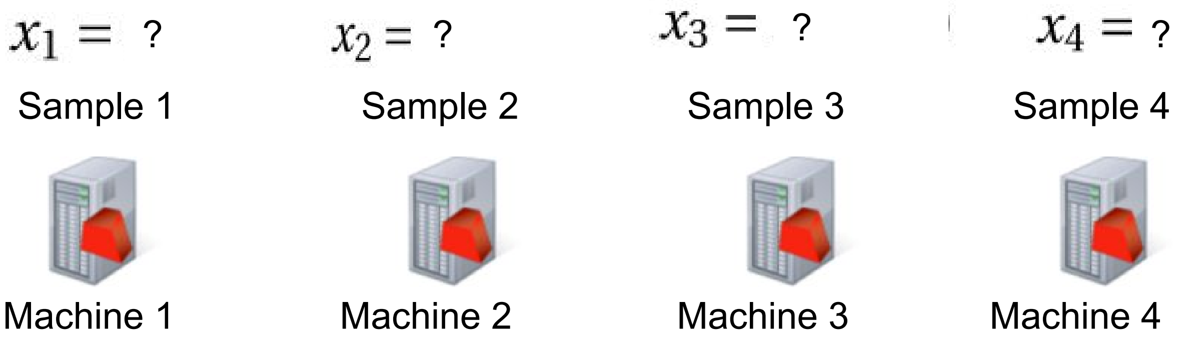
$x_1 =$?	$x_2 =$?	$x_3 =$?	$x_4 =$?
Sample 1	Sample 2	Sample 3	Sample 4
Machine 1	Machine 2	Machine 3	Machine 4

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

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

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

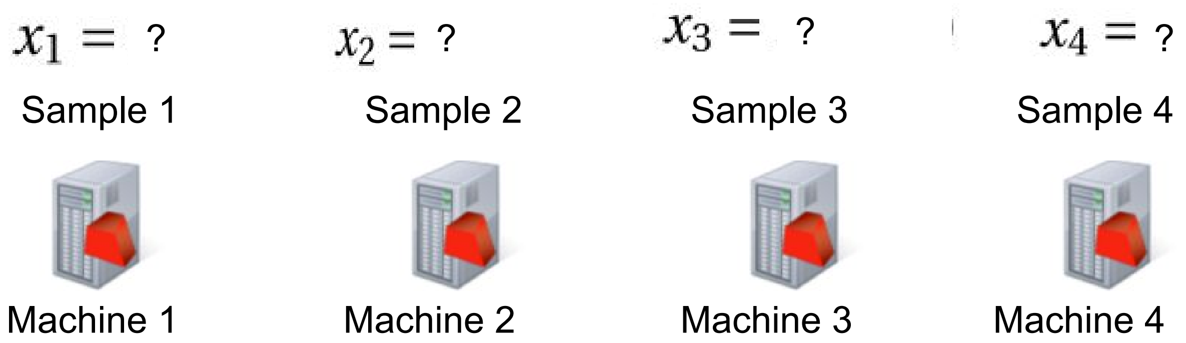
$x_1 =$?	$x_2 =$?	$x_3 =$?	$x_4 =$?
Sample 1	Sample 2	Sample 3	Sample 4
Machine 1	Machine 2	Machine 3	Machine 4

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

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

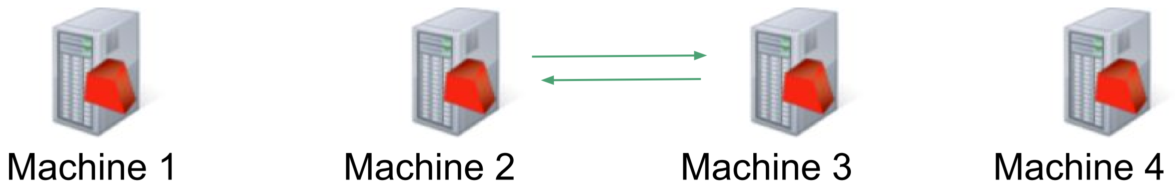
- Distribute the data across machines.
- We may not want to update a ‘centralized’ vector x
- **Decentralized Gradient Descent**
  - **Each machine has its own data samples**
  - **Each machine has its own parameter vector** x_m

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

$x_1 = x(2)$ $x_2 = x([1,4])$ $x_3 = x([1,4])$ $x_4 = x(3)$

Sample 1 Sample 2 Sample 3 Sample 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
  - **Update rule**

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

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

$x_1 = x(2)$

no communication

$x_2 = x([1,4])$

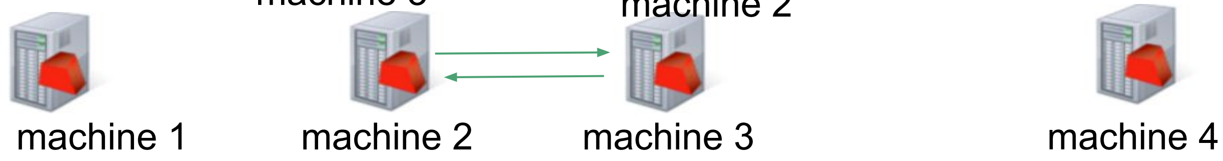
communicates with machine 3

$x_3 = x([1,4])$

communicates with machine 2

$x_4 = x(3)$

No communication



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

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

  - Similar convergence to the gradient descent with central communication
    - The rate depends on the sparsity of the dataset

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

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