HI!

CS 744: BISMARCK

Shivaram Venkataraman

Fall 2019

ADMINISTRIVIA

- Assignment 2 out!
- Project groups extension → till Oct 7th
- OH / Setup meeting by email

Oct 17th Introduction

COURSE PROJECT PROPOSAL

Propose topic, group (2 sentences) – Oct 7

Project Proposal (2 pages) – Oct 17

- Introduction
- Related Work
- Timeline (with eval plan)

WRITING AN INTRODUCTION

1-2 paras: what is the problem you are solving
why is it important (need citations)

→ Crucial

1-2 paras: How other people solve and why they fall short

↳ Target audience : Familiar with literature

1-2 paras: How do you plan on solving it and why your approach is better

1 para: Anticipated results or what experiments you will use

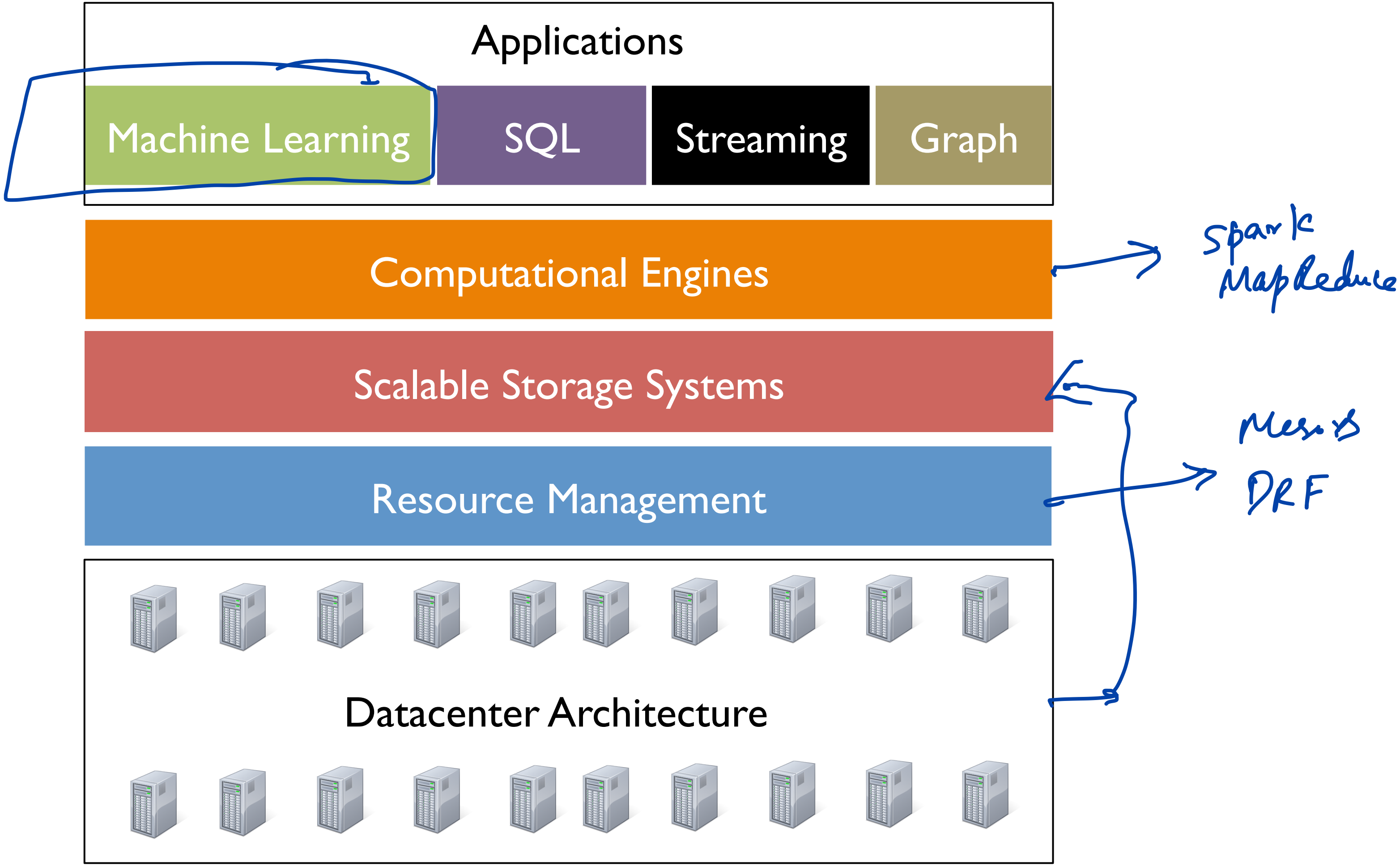
WRITING RELATED WORK

Group related work into two/three buckets
(1-2 para per bucket)

Scheduling Frameworks

Explain what the papers / projects do
Why are they different / insufficient

ML Frameworks:


Applications
Machine Learning
SQL
Streaming
Graph
Computational Engines
Scalable Storage Systems
Resource Management
Datacenter Architecture
Spark
MapReduce
Mesos
DRF

MACHINE LEARNING

challenge

Classification

Gmail → spam filter

IMAGENET

Recommendation

NETFLIX

amazon

Supervised learing

Training data → labels

model $(\text{data}_{\text{new}})$ → ??

OPTIMIZATION

$$\min_{w \in \mathbb{R}^d} \sum_{i=1}^{N} \boxed{f}(w, z_i) + P(w)$$

Regularization

Loss

Function

Model

Data (Examples)

CONVEX OPTIMIZATION

$$\min_{w \in \mathbb{R}^d} \quad \sum_{i=1}^{N} f(w, z_i) + P(w)$$

What is convex ?

Linear Regression, Linear SVM

Kernel SVMs, Logistic Regression,

What is not convex ?

Graph mining, Deep Learning

GRADIENT DESCENT

$$w^{(k+1)} = w^{(k)} - \alpha_k \nabla f(w^{(k)})$$

model at the k^{th} iteration

step size

Initialize w → random initialization

For many iterations:

Compute Gradient → How many data points?

Update model

End

GD : All of your training data (n)

IGD : One training data point (1)

SGD : Stochastic Gradient (B = 128 or 512)

INCREMENTAL GRADIENT DESCENT

$$w^{(k+1)} = w^{(k)} - \alpha_k \nabla f_{\eta(k)}(w^{(k)})$$

Initialize w

For many iterations:

  Pick one point

  Compute Gradient

  Update model

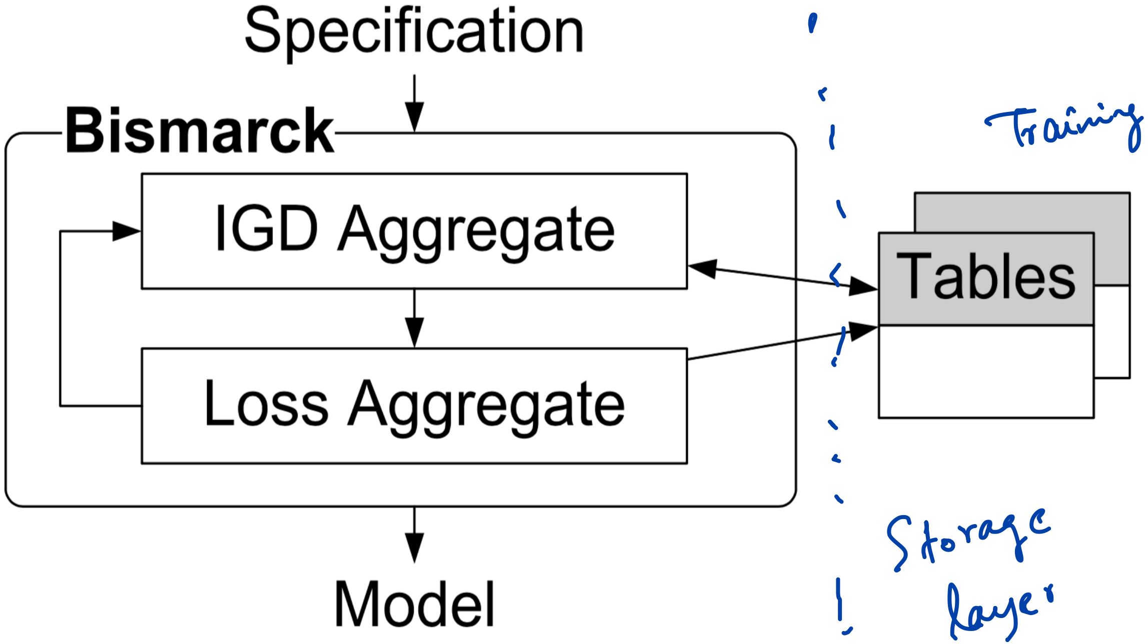
End

Analytics Task	Objective
Logistic Regression (LR)	$\sum_i \log(1 + exp(-y_i w^T x_i)) + \mu\|\vec{w}\|_1$
Classification (SVM)	$\sum_i (1 - y_i w^T x_i)_+ + \mu\|\vec{w}\|_1$
Recommendation (LMF)	$\sum_{(i,j)\in\Omega} (L_i^T R_j - M_{ij})^2 + \mu\|L, R\|_F^2$
Labeling (CRF) [48]	$\sum_k \left[\sum_j w_j F_j(y_k, x_k) - \log Z(x_k)\right]$
Kalman Filters	$\sum_{t=1}^T \|\|Cw_t - f(y_t)\|\|_2^2 + \|\|w_t - Aw_{t-1}\|\|_2^2$
Portfolio Optimization	$p^T w + w^T \Sigma w \quad \text{s.t.} \quad w \in \Delta$

BISMARCK ARCHITECTURE

Compute layer

- Selecting
- Gradient
- Updating



Training data

Storage layer

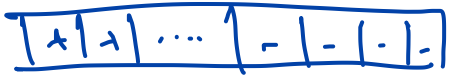
BISMARCK: USER DEFINED AGGREGATE

Three steps:

1. initialize(state) = initialize the model
   ↳ zero weights / random weights
2. transition(state, data)
   ↳ $\langle$data, label$\rangle$
   ↳ model
   update model based on data
3. terminate(state)
   ↳ Loss value $< \|\epsilon\|$
   ↳ Fixed num iterations / epochs

BISMARCK: LOGISTIC REGRESSION

```
LR_Transition(ModelCoef *w, Example e) { ...
  wx = Dot_Product(w, e.x);
  sig = Sigmoid(-wx * e.y);
  c = stepsize * e.y * sig;
  Scale_And_Add(w, e.x, c); ... }
```



DATA ORDERING

$$n(t_{shuff} + t_{epoch}) \quad \begin{matrix} \leq \\ \geq \\ = \end{matrix} \quad n_1 t_{epoch} + t_{shuff}$$

while iter:

$\boxed{\text{Pick data pt}}$

Grad

Update

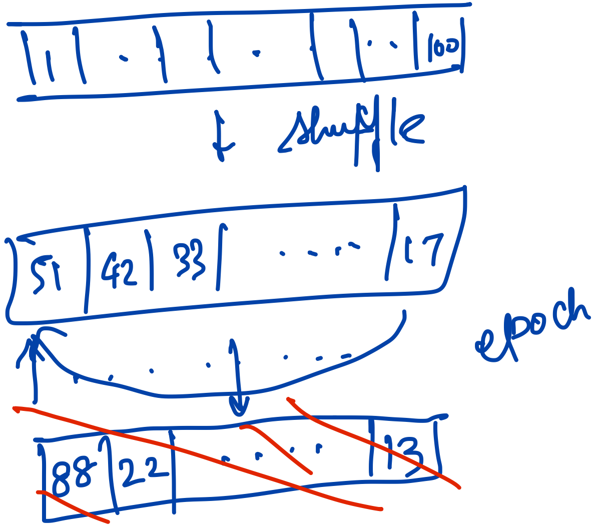
end

Random sampling

- Sample without replacement
- Shuffle the data after each epoch

Shuffle once

- Avoids pathological ordering
- Much cheaper



B+1 B+1 B

$\boxed{\text{Disk}}$ Seq access

first B

RESERVOIR SAMPLING

Select first m items

On the k^{th} additional item

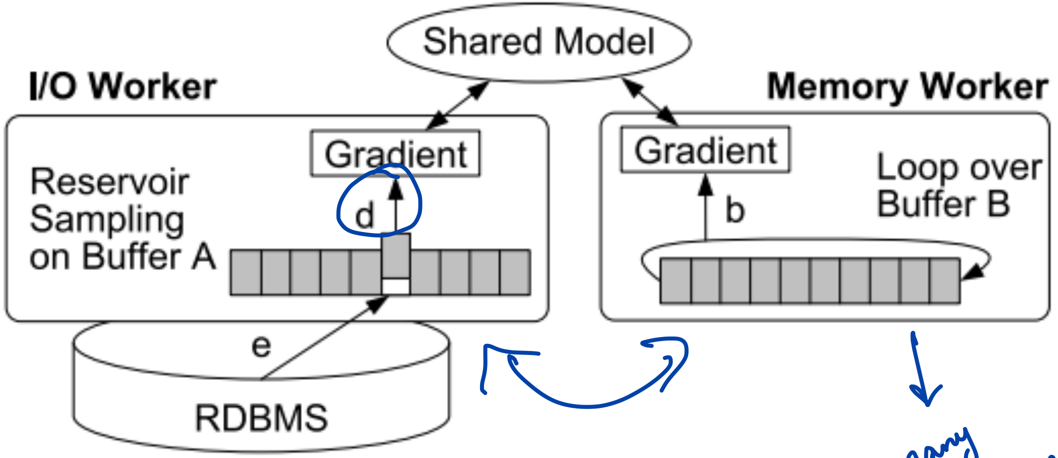
s = random in [0, $m + k$)

if $s < m$

Put in slot s

else

Drop the item



Resvoir sampling eliminated

→ Keys

IGD

many iterations over fixed buffer

PARALLEL GRADIENTS

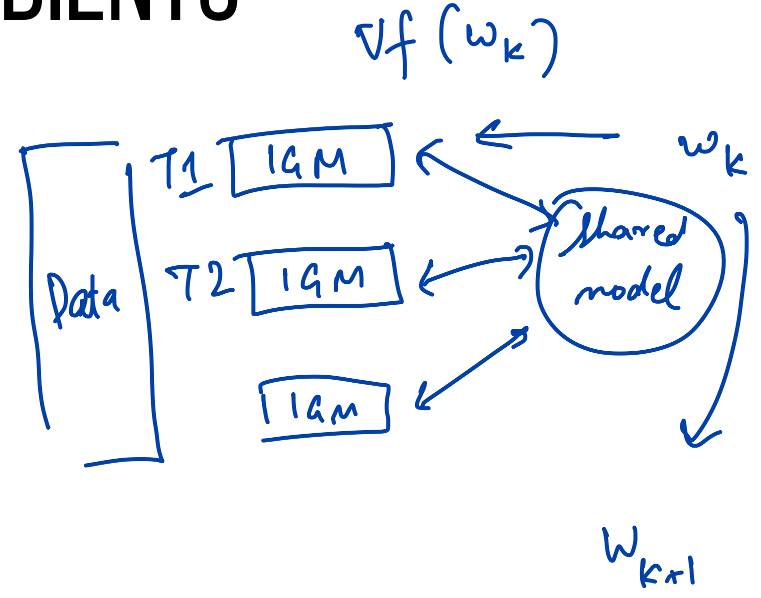
Async update

T1: w_{k+1} . . . . w_{k+2}

Async

Shared Memory:

- Compute gradients in parallel
- Average their updates
- Or update in parallel
    - Locks?



- AIG : Atomic Increment Instructions
  ↳ fast, implemented in hardware

- Lock-free : Iteration times very fast, conflicts?

DISCUSSION

https://forms.gle/nFNEi2NZMNhZio1f7

How would an implementation of GD look in Spark? Try to sketch an implementation. What would be similar / different to Bismarck?

data: sc.textFile (" ")

data. shuffle ( )

grad : data. map (x => gradComp(x))

. reduce (_ + _)

model : model - step * grad

loop

check for terimination

model

shuffle everytime?

broadcast

Shared memory

no broadcast

AIG & lock free

What are some ML scenarios where Bismarck architecture might prove to be limited?

Non convex : Converge ? Shuffle once

lock free

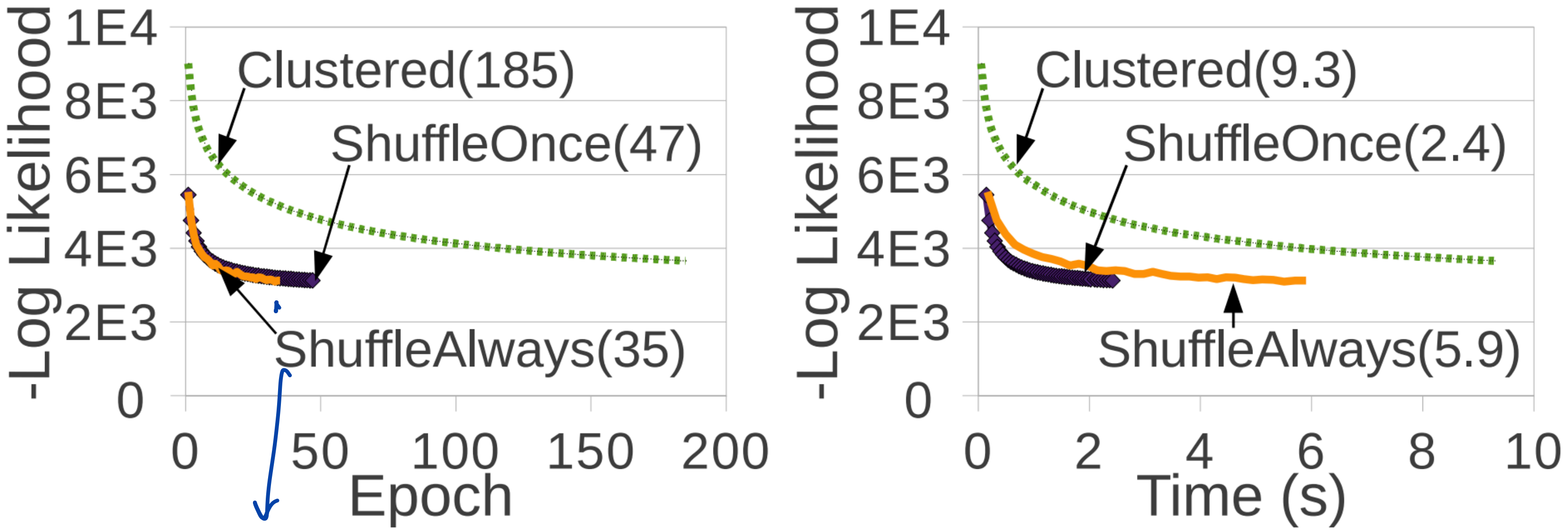
Skewed dataset : Conflicts → lock-free ?

Complex model

Overfitting ?

Model fits in memory :

$||^{l}$ism limited ?



-Log Likelihood
1E4
8E3
6E3
4E3
2E3
0
Clustered(185)
ShuffleOnce(47)
ShuffleAlways(35)
0
50
100
150
200
Epoch
-Log Likelihood
1E4
8E3
6E3
4E3
2E3
0
Clustered(9.3)
ShuffleOnce(2.4)
ShuffleAlways(5.9)
0
2
4
6
8
10
Time (s)

least
epoch

NEXT STEPS

Next class: Parameter Server

Assignment 2 out!

Project Proposal

- Groups by Oct 7
- 2 pager by Oct 17