Good morning!

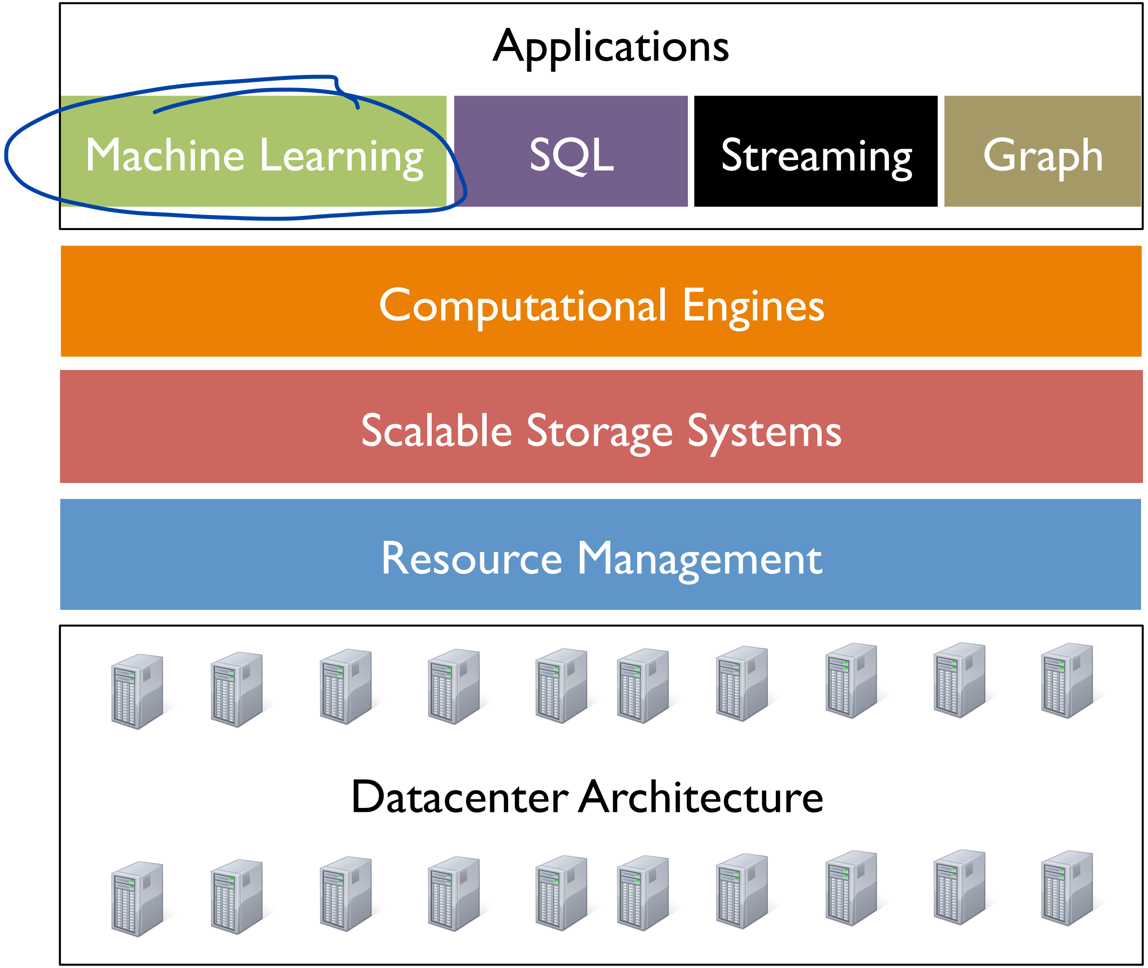
CS 744: PARAMETER SERVERS

Shivaram Venkataraman

Fall 2019

ADMINISTRIVIA

- Assignment 2 is out!
- Course project groups due Oct 7
- Introductions due Oct 17


Applications
Machine Learning
SQL
Streaming
Graph
Computational Engines
Scalable Storage Systems
Resource Management
Datacenter Architecture

Bismarck

Supervised learning
Unified Interface

Shared memory
Model fits in memory

IGD

Sampling / ordering

Parallelism

Machine Learning

MOTIVATION

- Large training data 1TB to 1PB
- Models with 10^9 to 10^{12} parameters

8GB 8TB if 8 bytes per parameter

- Goals
  - Efficient communication
  - Flexible synchronization
  - Elastic Scalability
  - Fault Tolerance and Durability

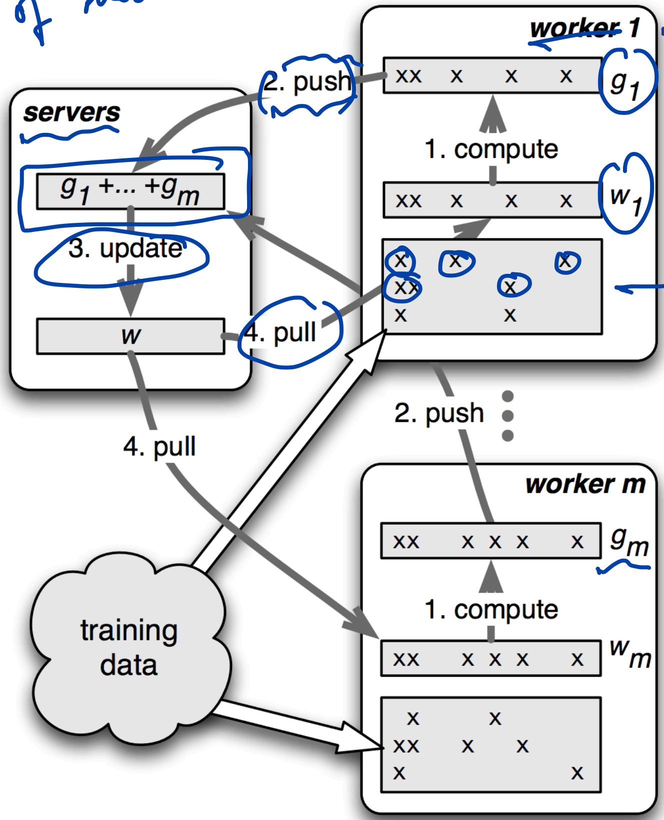
EXAMPLE WORKLOAD

$w_{t+1} = w_t - \gamma \sum_{i=1}^{k} g_i$

network traffic ≈ size of model

Ad Click Prediction

- Trillions of clicks per day
- Very sparse feature vectors



Training data

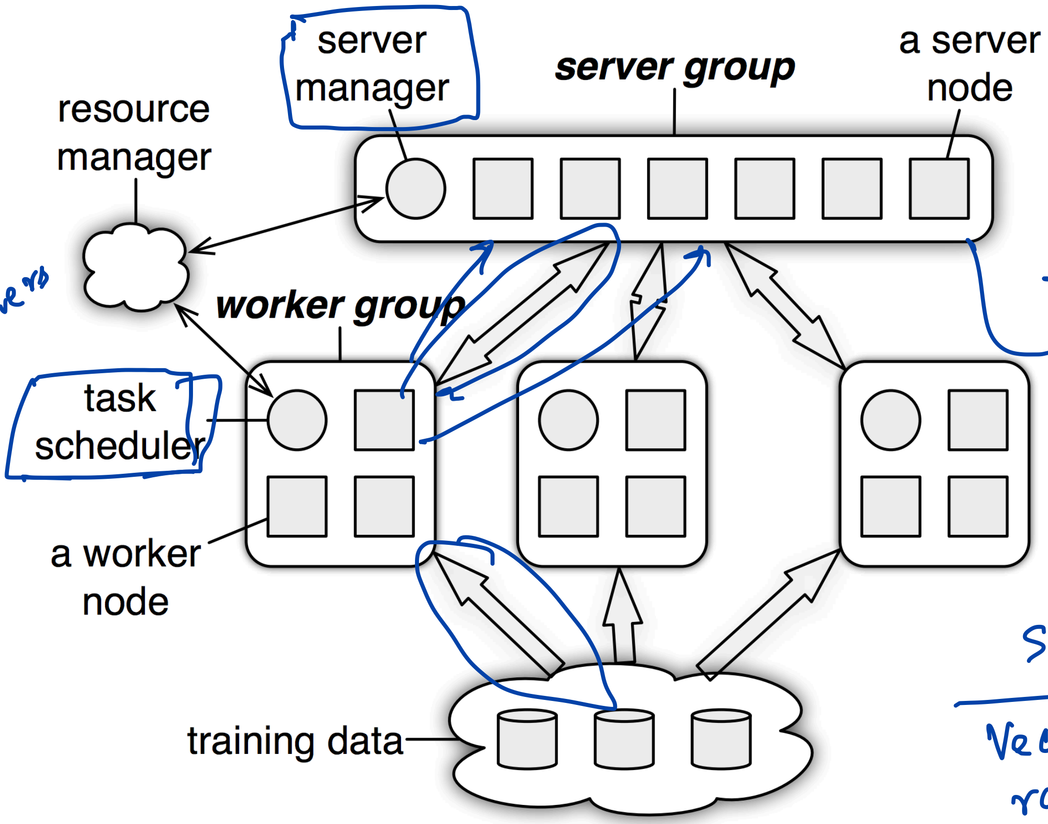
$m \approx$ 100s or 1000s

Computation flow

1. Sparse training data per worker
2. Compute $w^T x_i = g_i$
3. Update $w_2 = w_1 - \gamma \sum g_i$
4. Pull w_2 ⇒ How long should we wait?

ARCHITECTURE

Number of workers != Num Servers



Parameters are split across servers

parameter has a version

n servers (0, ..., n-1)

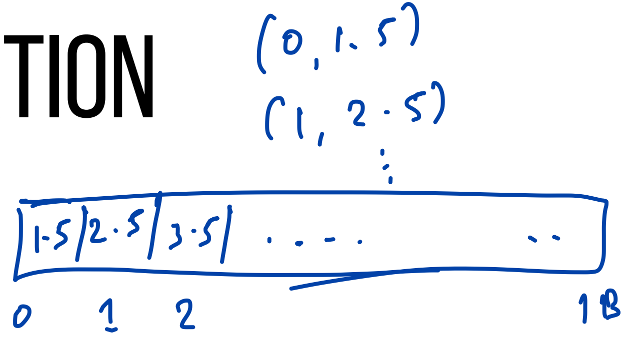
d parameters

Space : nd

Vector clock

range of value = ||R||

REPRESENTATION



- Key value pairs e.g., (featureID, weight)

- Assume keys are ordered.

  Easier to apply linear algebra operations

(0 - 100) [. . . . . . .]
100 floats

- Interface supports range push and pull

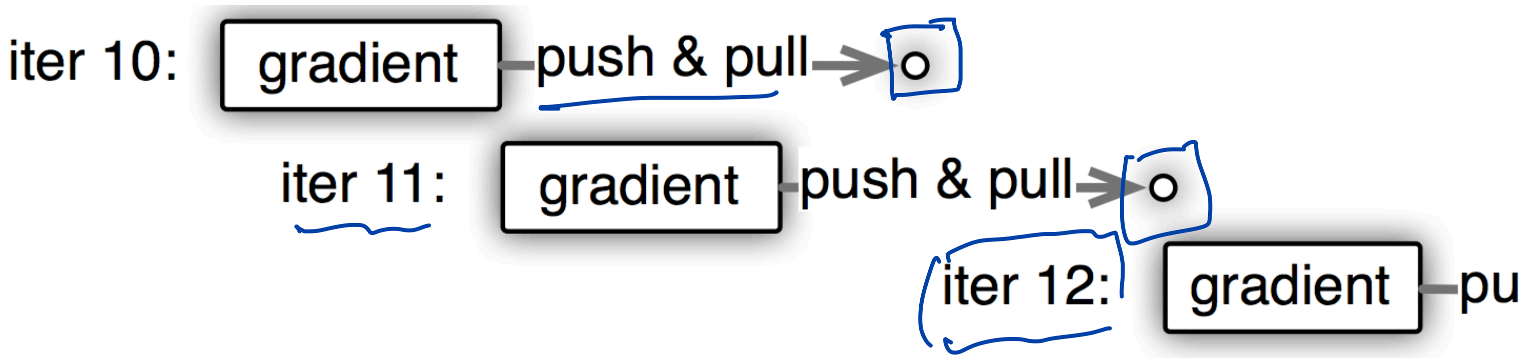
  w.push(R, dest) → destination server

  R → range

w[R] = w[1:100] → destination

- Support for user-defined functions on server-side

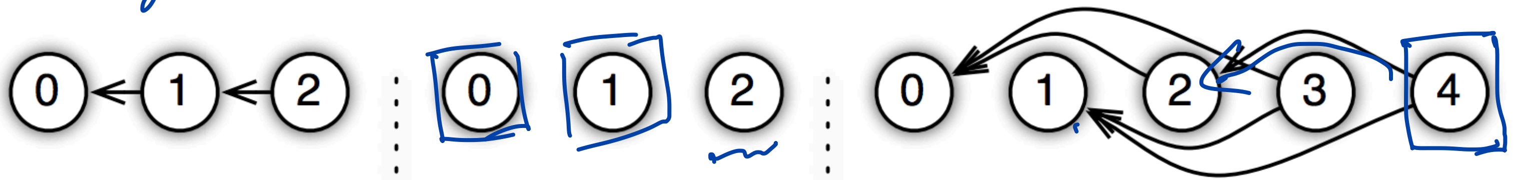
TASK DEPENDENCY



workers have the option of starting iteration with/without prior iteration

CONSISTENCY MODELS

Flexible Consistency



(a) Sequential (b) Eventual (c) 1 Bounded delay

BSP

T- Bounded

↳ Upper bound on the staleness

User defined filters

Significantly modified filter

KKT filter

Not all gradients need to be transmitted

$\nabla f : [\ldots 1e\text{-}7 \ldots 1e\text{-}10 \ldots]$

million

IMPLEMENTATION: VECTOR CLOCKS

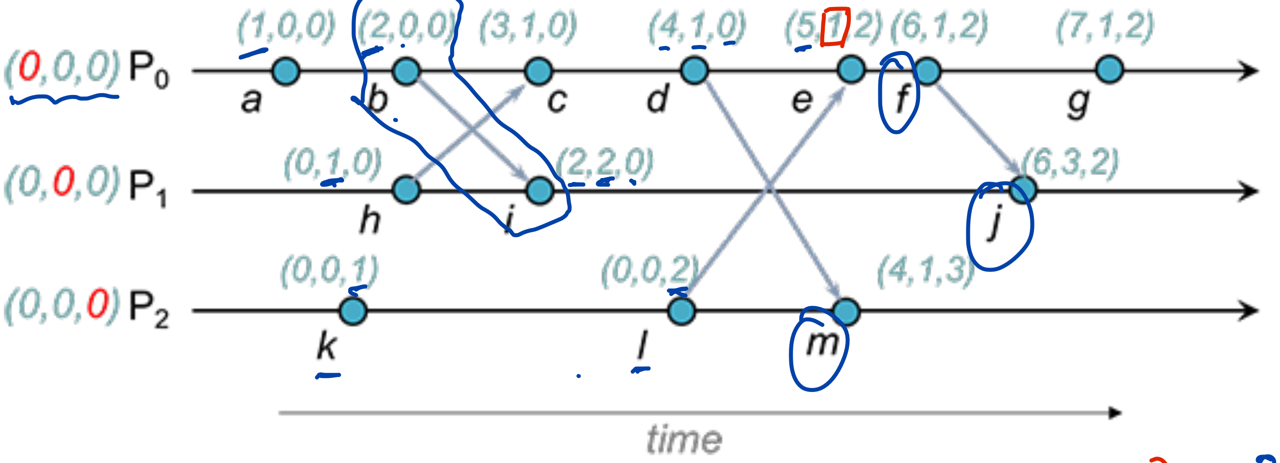
(4, 1, 0) ← (0, 0, 2)

number of processes

Compute
→ inc local

Send
→ inc local

recv
→ inc local
copy counter other process
↳



if j before or after f ?

g : (7, 1, 2)
j : (6, 3, 2)
Concurrent

j : (6, 3, 2)
f : (6, 1, 2)

f happens before j

IMPLEMENTATION

Key Caching

- Worker might send the same key lists again
- Receiving node caches the key lists
- Sender only needs to send a hash of the list

0 - 100
:
255 - 302
:

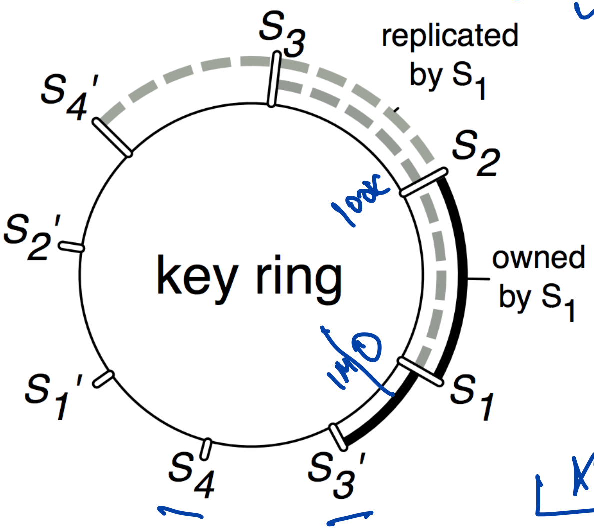
Upto 2x

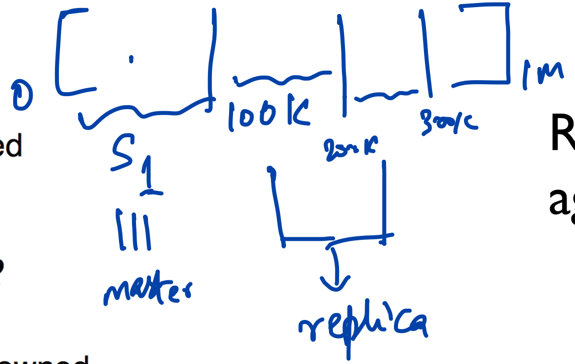
Value Compression

- Lots of repeated values, zeros
- Use Snappy to compress messages

1. Filter
2. Values compress
3. Key Caching

IMPLEMENTATION: REPLICATION

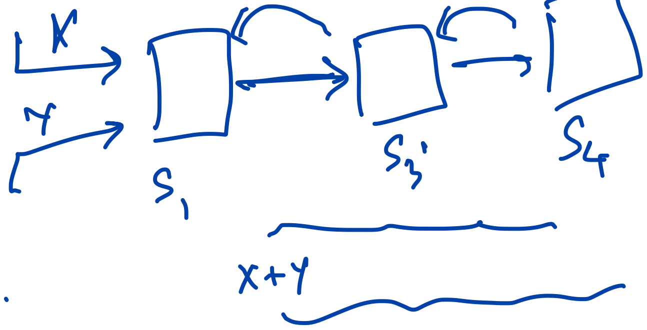




Replication after aggregation

Specialization based on workload

Chain replication



FAULT TOLERANCE

1. Server manager assigns the new node a key range to serve as master.

2. The node fetches the range of data to maintains as master and k additional ranges to keep as slave.

3. The server manager broadcasts the node changes. The recipients of the message may shrink their own data

Worker node fails

- Restart another worker operates on same training data

- Ignore it

- Move training data to existing workers

Algorithm 3 Delayed Block Proximal Gradient [31]

Scheduler:

1: Partition features into b ranges $\mathcal{R}_1, \ldots, \mathcal{R}_b$
2: **for** $t = 0$ **to** T **do**
3: Pick random range $\mathcal{R}_{i_t}$ and issue task to workers
4: **end for**

Worker r at iteration t

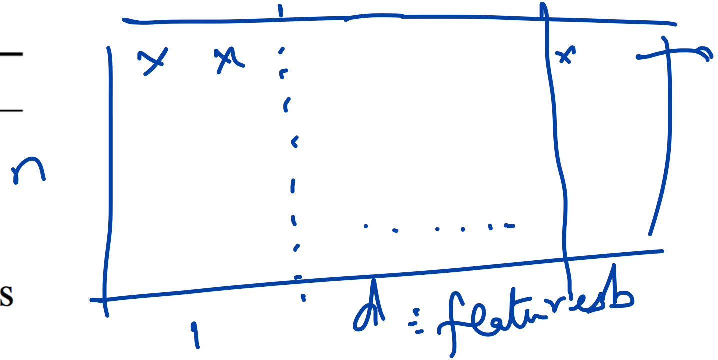
1: Wait until all iterations before $t - \tau$ are finished
2: Compute first-order gradient $g_r^{(t)}$ and diagonal second-order gradient $u_r^{(t)}$ on range $\mathcal{R}_{i_t}$
3: Push $g_r^{(t)}$ and $u_r^{(t)}$ to servers with the KKT filter
4: Pull $w_r^{(t+1)}$ from servers ← block

Servers at iteration t

1: Aggregate gradients to obtain $g^{(t)}$ and $u^{(t)}$
2: Solve the proximal operator

$$w^{(t+1)} \leftarrow \underset{u}{\operatorname{argmin}}\, \Omega(u) + \frac{1}{2\eta}\|w^{(t)} - \eta g^{(t)} + u\|_H^2,$$

where $H = \operatorname{diag}(h^{(t)})$ and $\|x\|_H^2 = x^T H x$



Bounded Staleness

SPARSE LR

DISCUSSION

https://forms.gle/35vrxyG6WLmSvCs38

What are some of the downsides of using PS compared implementing Gradient Descent in Bismarck / Spark?

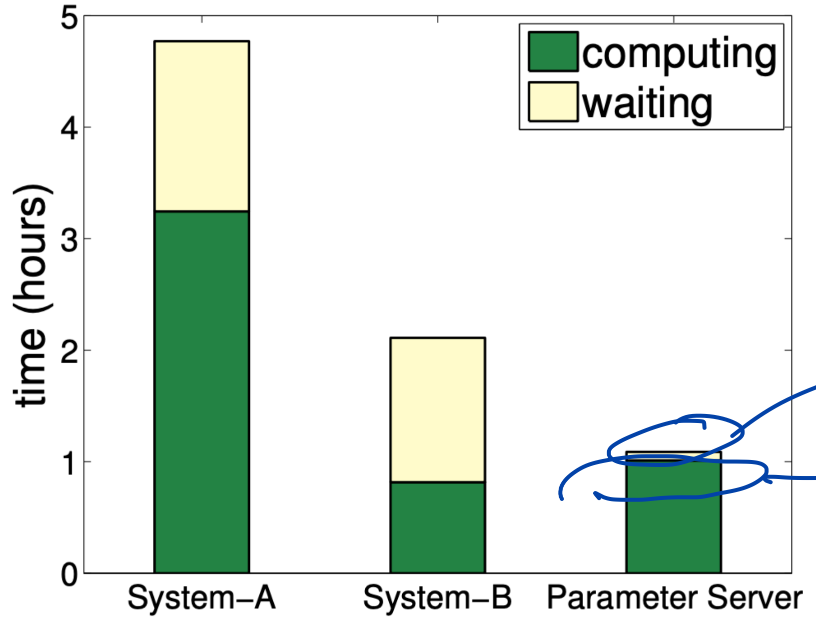
Training data fits in memory $\Rightarrow$ PS network overhead

Do we need full gradient? $\Rightarrow$ PS gradient is split.

Consistency (or lack of) lead to slow down

Sparse data PS is good $\Rightarrow$ What about dense data?

Extra resources to run servers



less waiting

more iterations
&
more compute

same algorithm

How would you integrate PS with a resource manager like Mesos? What would be some of the challenges?

↳ Allocate resources to run servers & workers

⟹ At least some fixed for servers

Not that flexible

↳ Long running workers & servers → min resource

↓

pre-empt → time to converge

↳ Key cache change ⟹ training data changes

NEXT STEPS

Next class: Tensorflow

Assignment 2 is out!

Course project deadlines

- Oct 7 (topics)
- Oct 17 (proposals)