Carnegie Mellon University

Implementation

Tutorial of Parameter Server

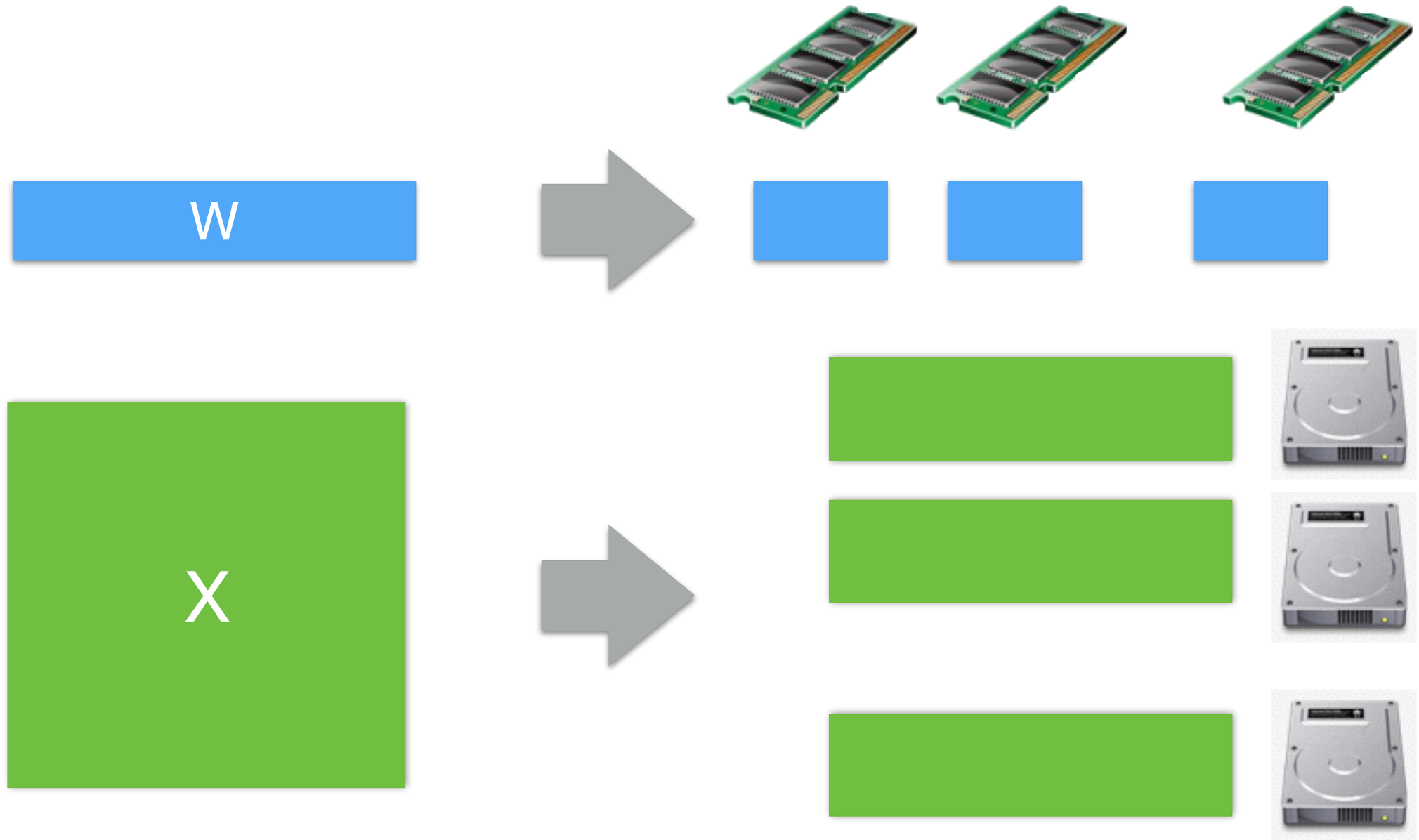
Mu Li

CSD@CMU & IDL@Baidu

muli@cs.cmu.edu

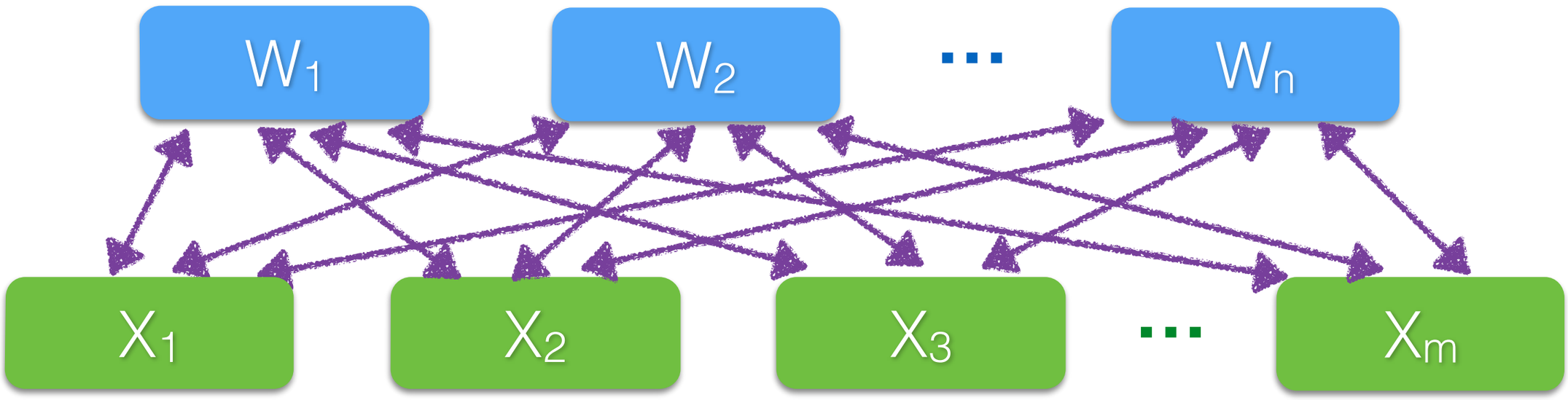
Model/Data partition

- Learn w from training data X



Worker/Server Architecture

- Servers maintain shared parameters
- A work owns part of training data and carries computation



Distributed Subgradient Descent

Partition data into workers

Workers get the working set of *w*

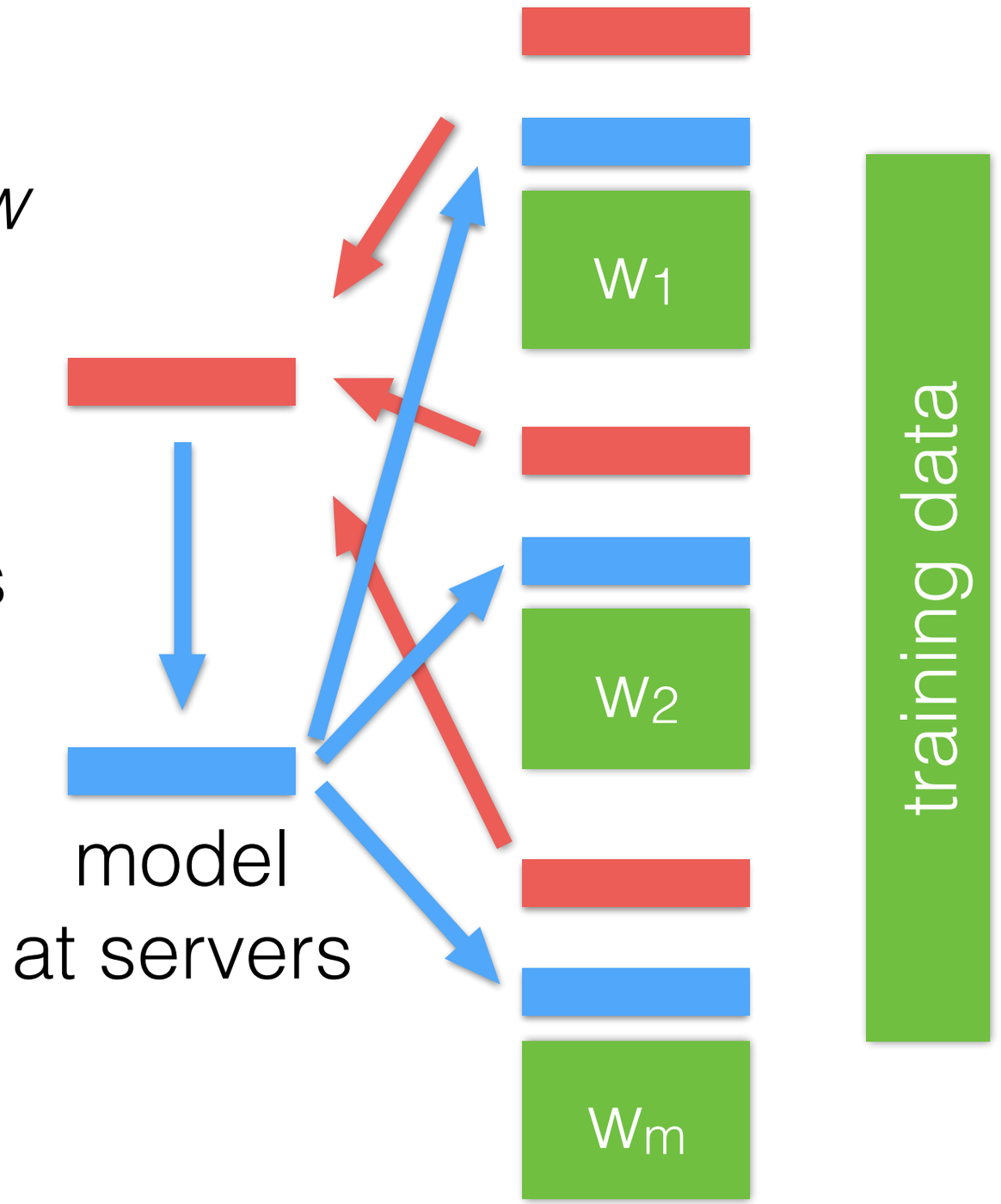
Iterate *t = 1, 2, …*

workers compute gradients

servers aggregate gradients

update *w*

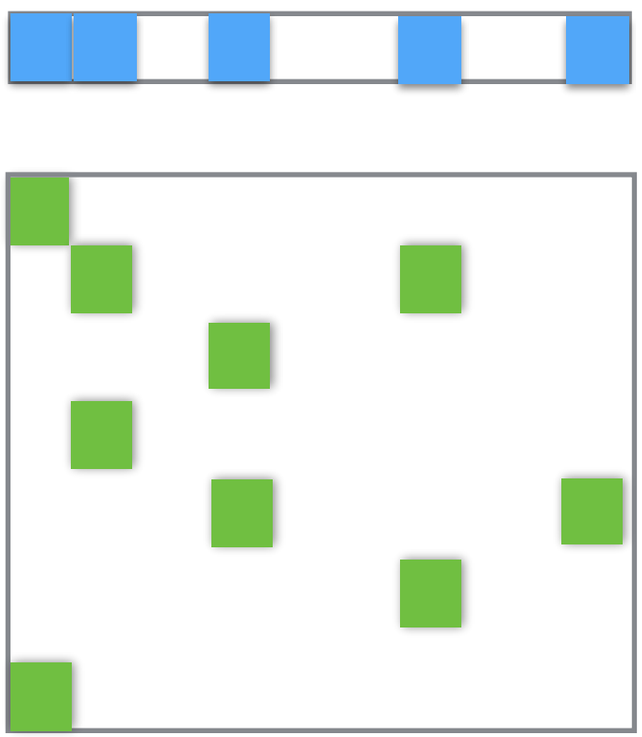
workers get updated *w*



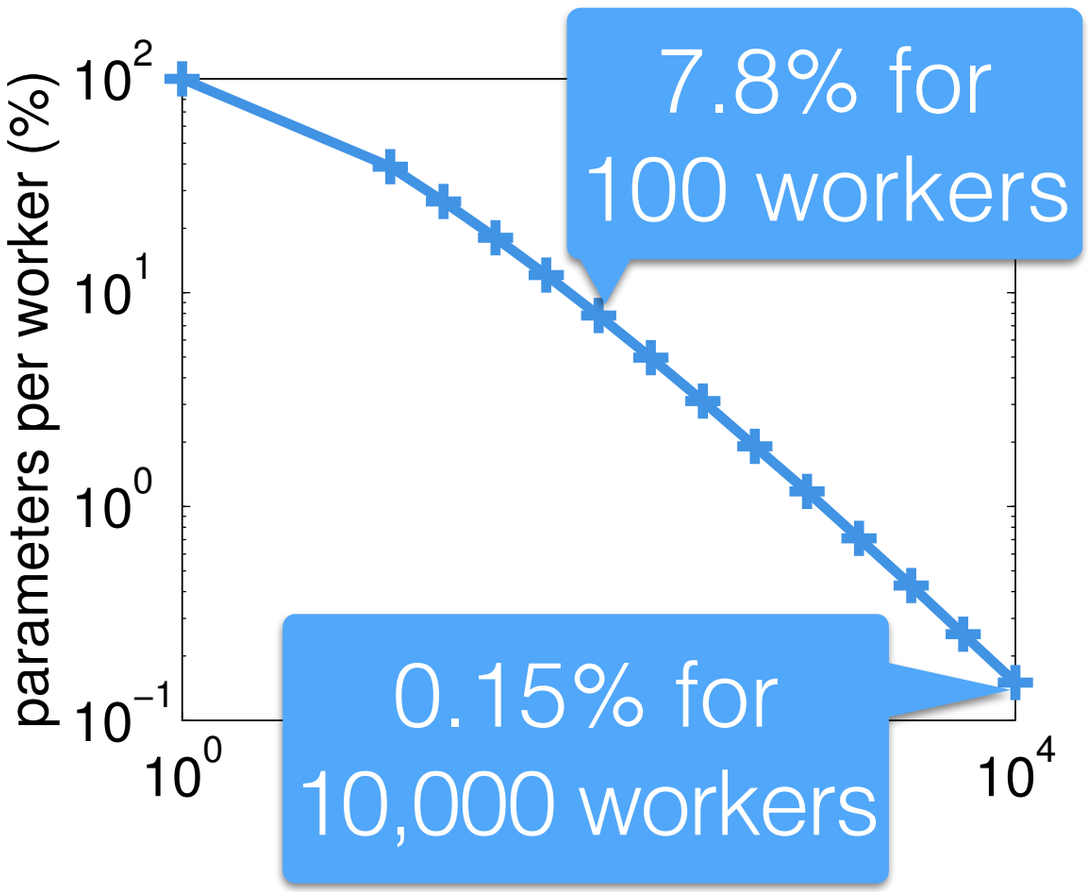
Billions of Parameters

- A worker cannot cache the whole model
- For sparse data and linear methods, a worker only needs to cache the working set

local cached model



local training data



Compact Representation

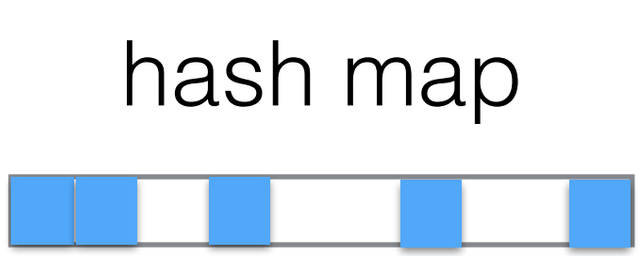
- A feature ID can be a string, or a random 64bit/ 128bit integer

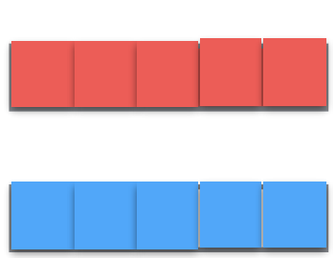
```
1412483:58 57900701327764829l5:58 159745120897349309l7:58 781072222689272664l:127 70
87 809585706602283004l:87 910270109130459922l:87 1224857408590006:90 347665728436447
:95 118723903346136193l6:96 579011191481966852l:96 132772062148148860l2:114 97132286
952684714525520293l:2 987193656403702108l:3 436156178045502243l8:125 4440476426564862
9622990838909:130 343177601957878688l:131 485705205335840828l0:131 130855617801956688
```

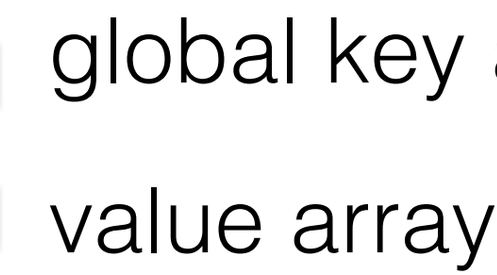
- How to access w[feature_id]?
  - ★ hash map
    - general solution
    - could be slow

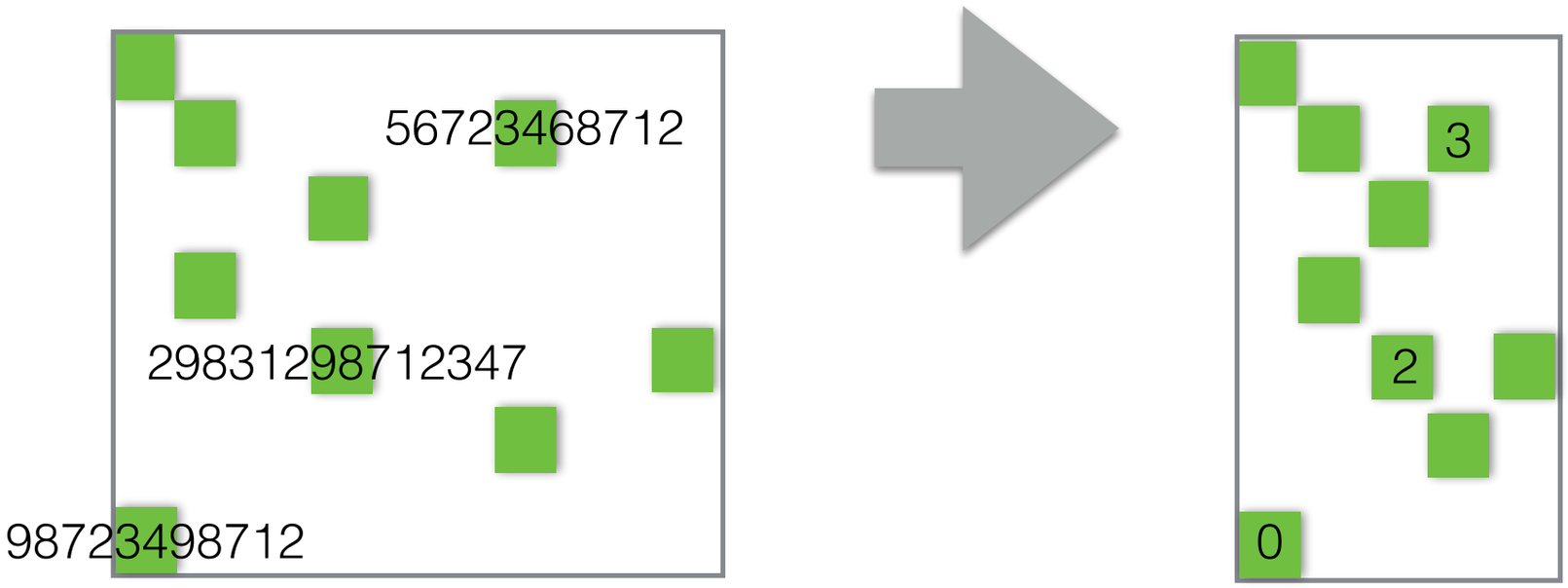
Localize Keys

- Each worker machine maps global features id into local features id 0, 1, 2, …







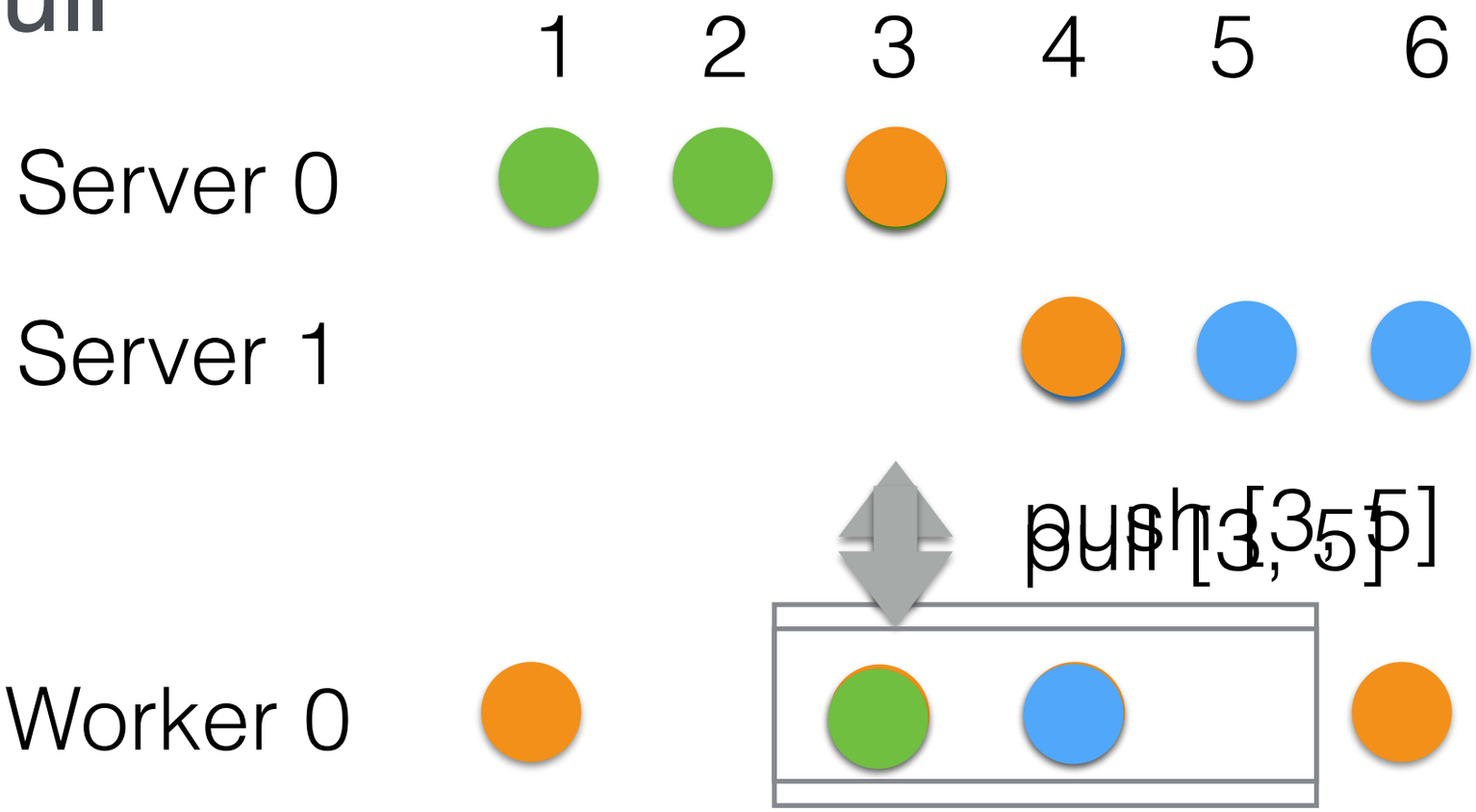


Localized Keys

- Save space
  - string or 64/128bit integer to 32bit integer
- Faster access
  - array versus hash map
- Reuse existing libraries
  - eigen3
- Need preprocessing
- May be slow if there are a lot of insert
  - online learning / LDA

Communication API

- Communication over global keys
- Batched communication via range-based push and pull



Message

- Data transmission format
  - protobuf header (command, timestamp, etc..)
  - a list of keys
  - a list of values
- Will split into several message if send to multiple machines

Reduce message size

- Key cache
  - worker 0 sends to server 0
    - T 1: (2, 2.3), (4, 6.1), (8, 9.9)
    - …
    - T 6: (~~2~~, 5,4), (~~4~~, 2.5), (~~8~~, 2.9)

Both sender and receiver cache the key list. If hit cache, then only send a checksum

- Value compression
  - may contains a lot 0s: sparse model, user-defined filter

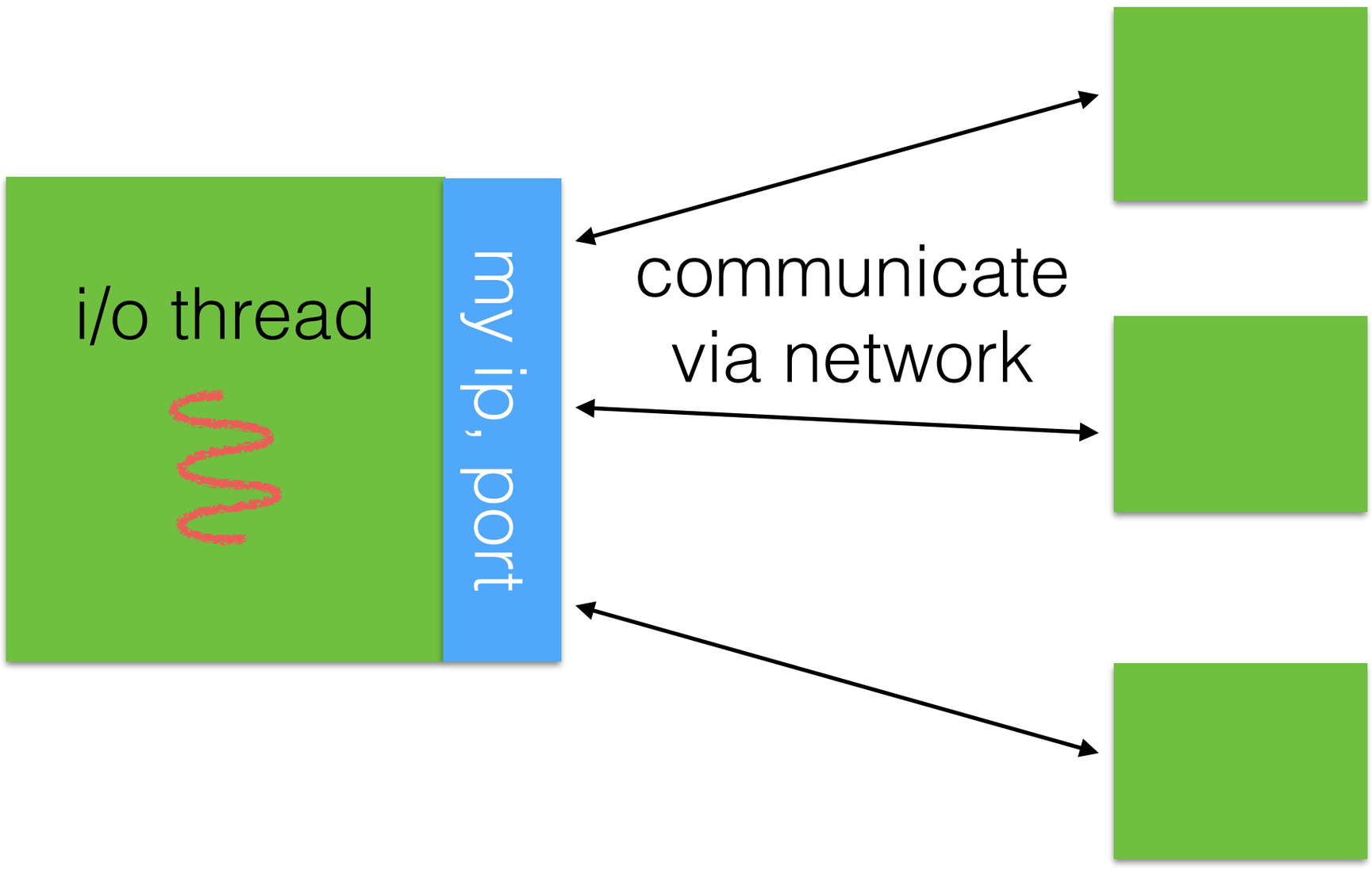
System components

- System: core system
- App: optimization application
  - gradient descent, coordinate descent
- Parameters: globally shared parameters
  - sort key-value vector, maps, …
- Loss:
- Penalty:
- etc…

Terminology

- Message: data for communication
- Van: send/receive message
- Customer: an application or a shared parameters
- Postoffice: allow customers to deliver messages and notify customers incoming messages
- Yellowpages: all customers and live machines

Van



- Zeromq: a convenient socket-like library
- RDMA: increasing interests

Thread Model

- T0: pass received messages to customers
- T1: receive update_model task, calculate gradients, call parameter's push
- T2: receive pulled values, updating local data

other machines

scheduler

message buffer

T0

T1

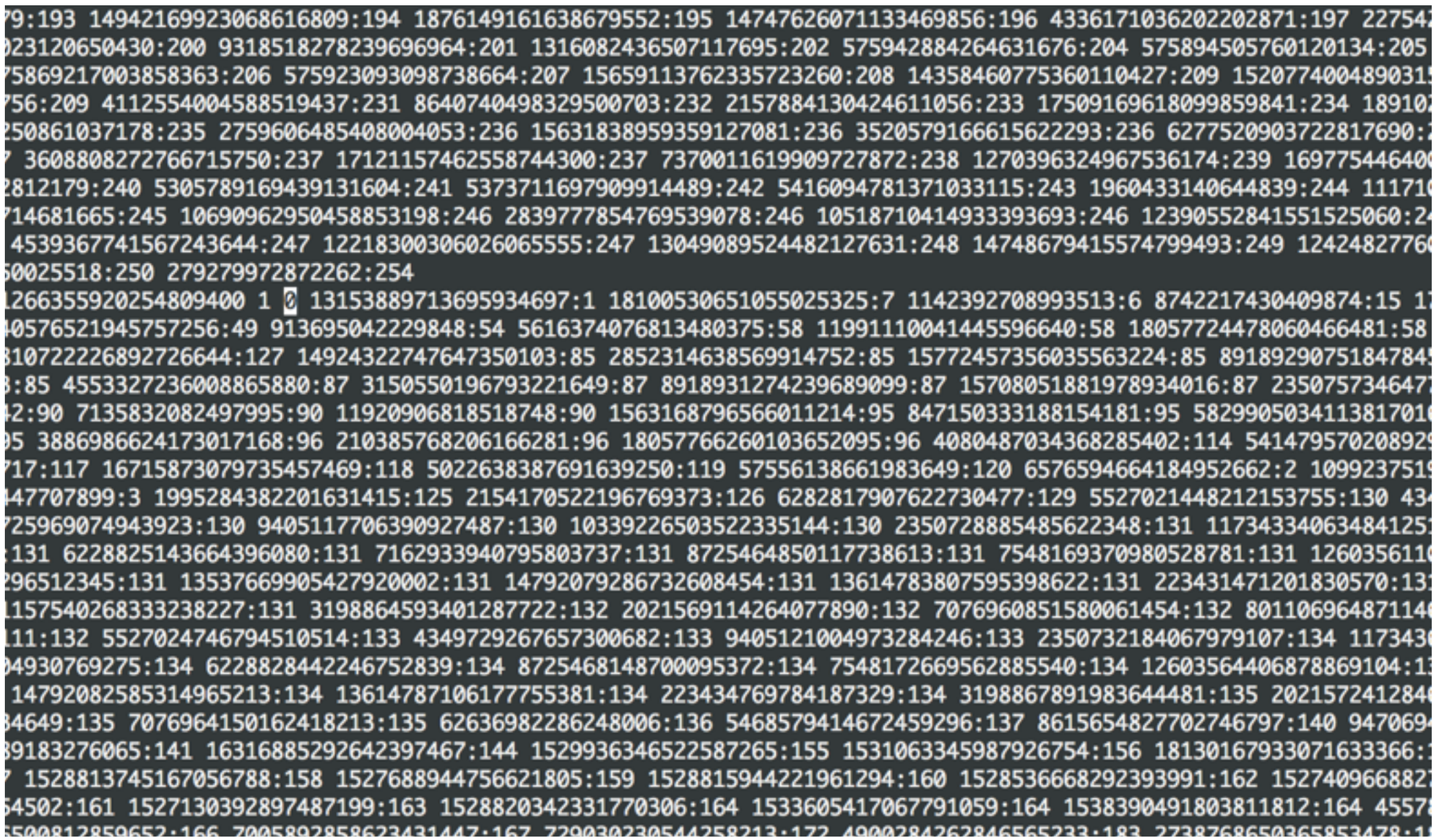
T2

Conclusion

- Support extremely large model
- Asynchronous
- Flexible consistency model
- Use like writing single-thread matlab/numpy/R-like codes
- A little bit complex to under the system
  - necessary to reach the best performance

Demo I

- Run on data with 64-bit features

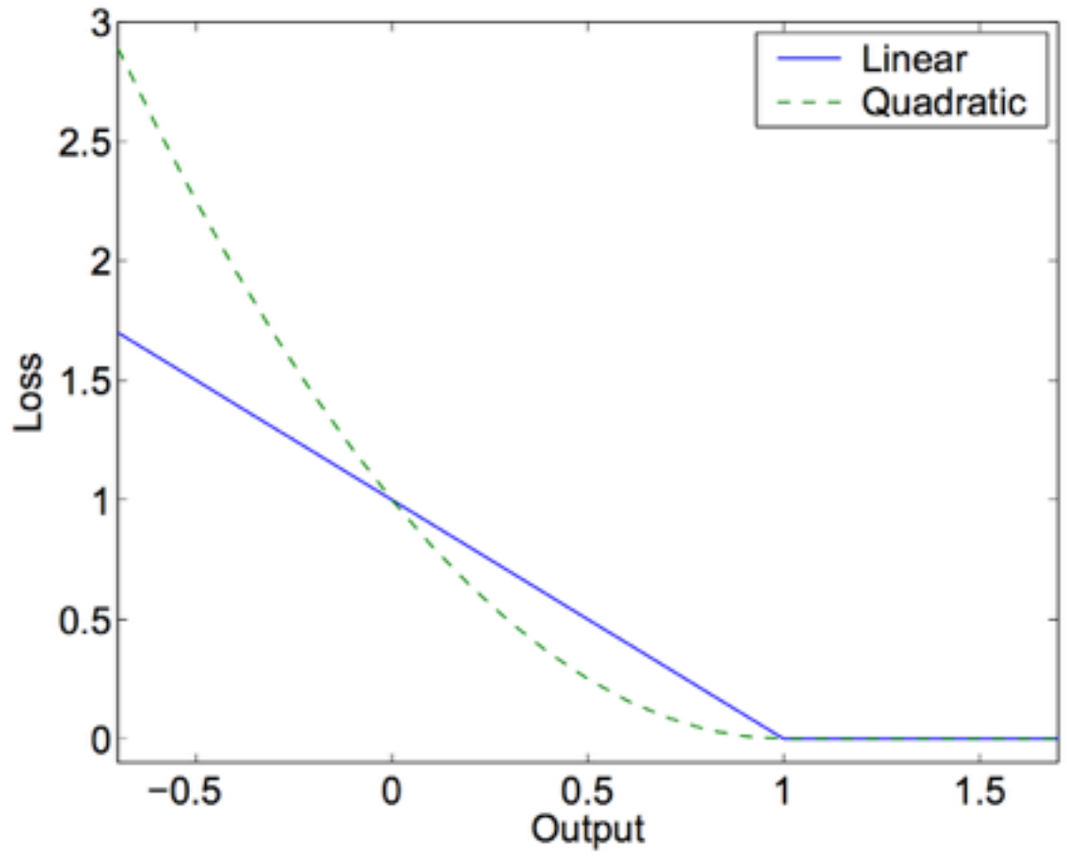


Demo 2

- Implement the squared hinge loss

$$L(y,t) = \big(\max(0, 1 - yt)\big)^2$$

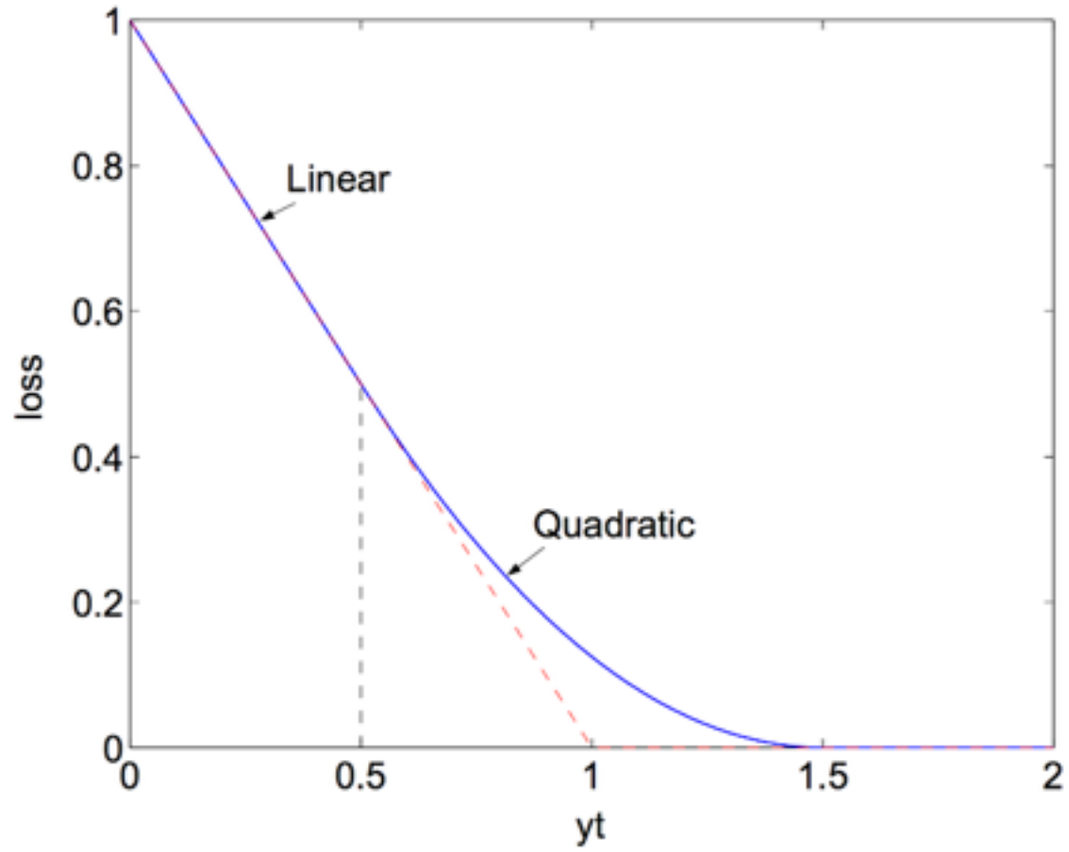
where $t = \langle w, x \rangle$



Exercise

- Implement the huber hinge loss

$$L(y,t) = \begin{cases} 0 & \text{if} \quad yt > 1+h \\ \frac{(1+h-yt)^2}{4h} & \text{if} \quad |1-yt| \leq h \\ 1-yt & \text{if} \quad yt < 1-h \end{cases}$$



Conclusion

- Industrial-scale problems:
  - applications
  - 100B examples, 10B features, 1T-1P size
  - limited machine resources
- Distributed implementation
  - gradient descent
  - coordinate descent
  - stochastic gradient descent
- Implement and use parameter server