



Carnegie
Mellon
University

Implementation

Tutorial of Parameter Server

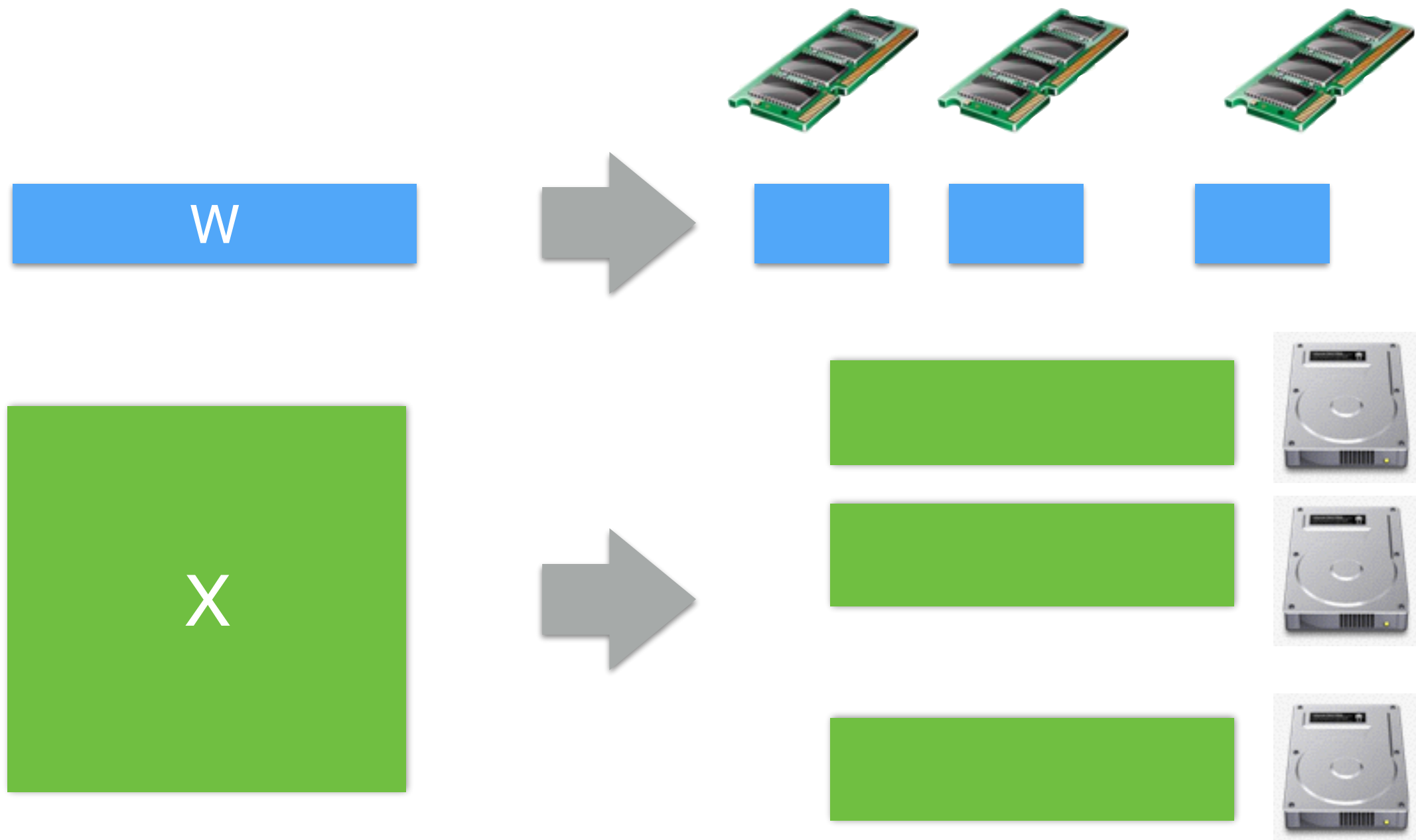
Mu Li

CSD@CMU & IDL@Baidu

mul@cs.cmu.edu

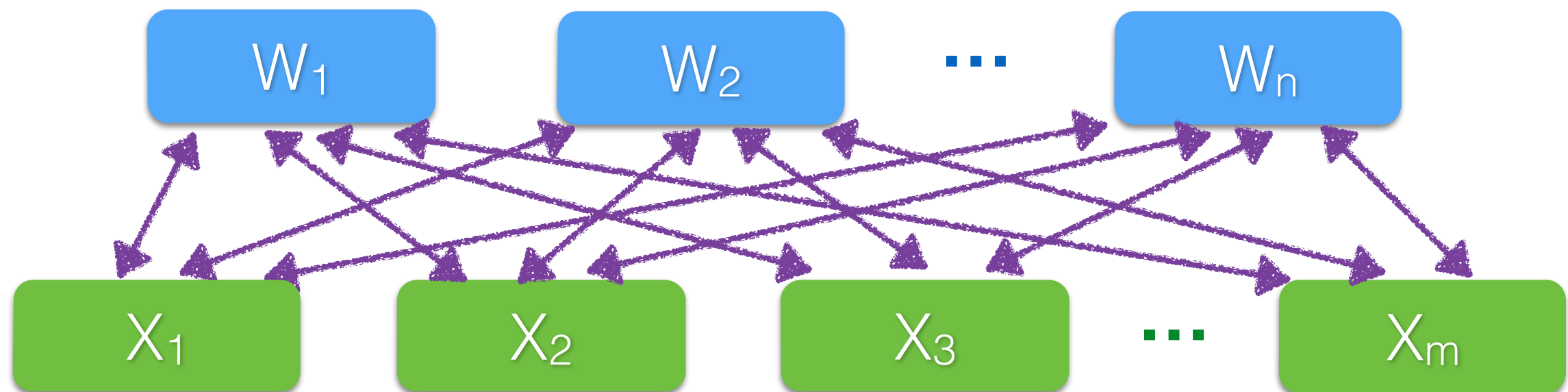
Model/Data partition

- ◆ 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

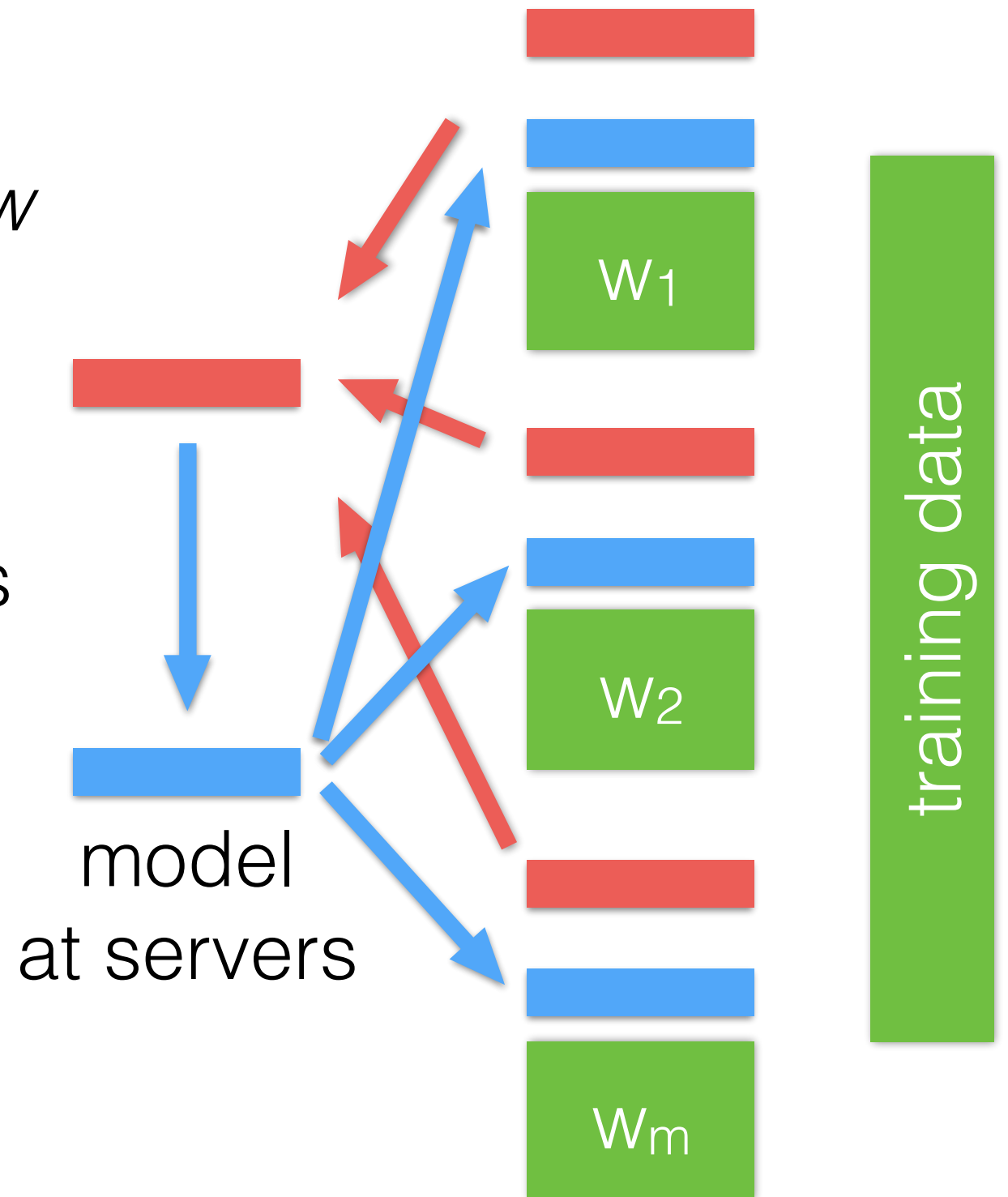
Iterate $t = 1, 2, \dots$

workers compute gradients

servers aggregate gradients

update w

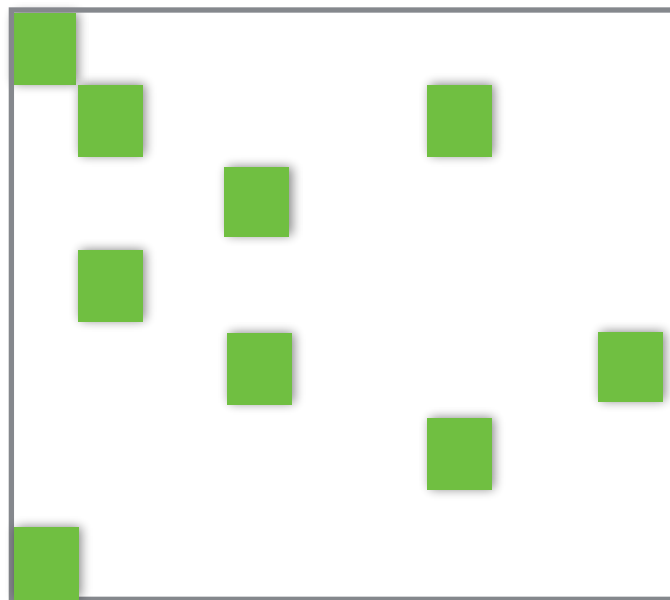
workers get updated w



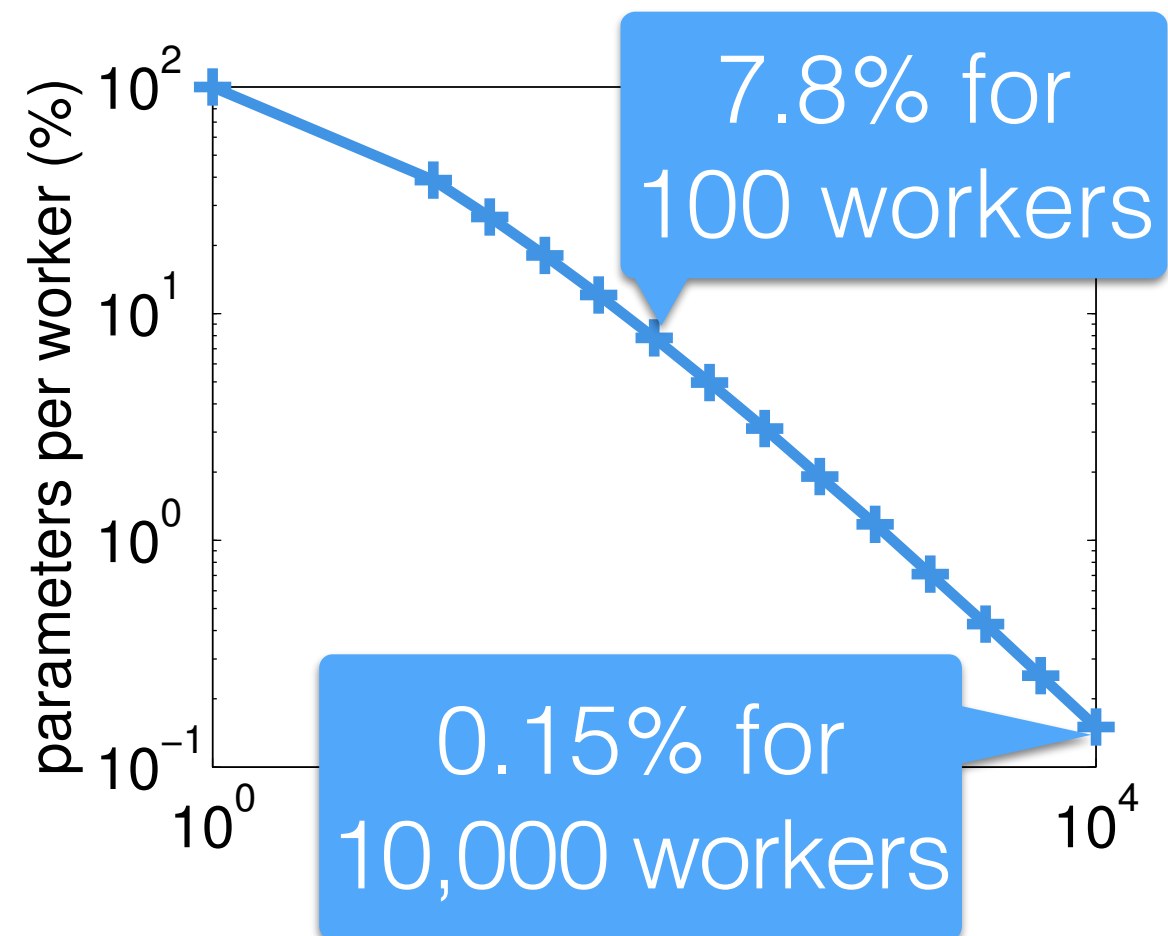
Billions of Parameters

- ✦ 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

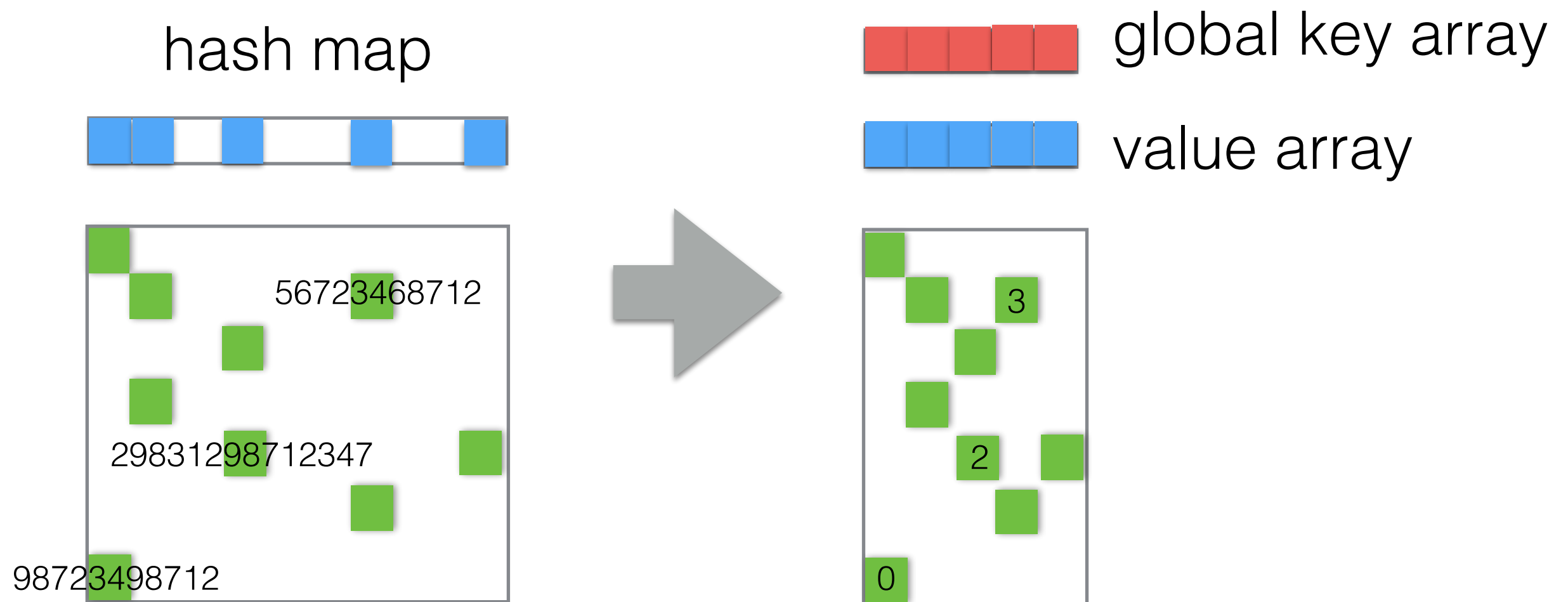
- ♦ A feature ID can be a string, or a random 64bit/128bit integer

```
1412483:58 5790070132776482915:58 15974512089734930947:58 7810722226892726644:127 70
37 8095857066022830044:87 9102701091304599223:87 1224857408590006:90 347665728436447
:95 11872390334613619361:96 5790111914819668529:96 13277206214814886052:114 97132286
9526847145255202931:2 9871936564037021082:3 4361561780455022438:125 4440476426564862
9622990838909:130 3431776019578786889:131 4857052053358408280:131 130855617801956688
```

- ♦ How to access $w[\text{feature_id}]$?
 - ★ hash map
 - general solution
 - could be slow

Localize Keys

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

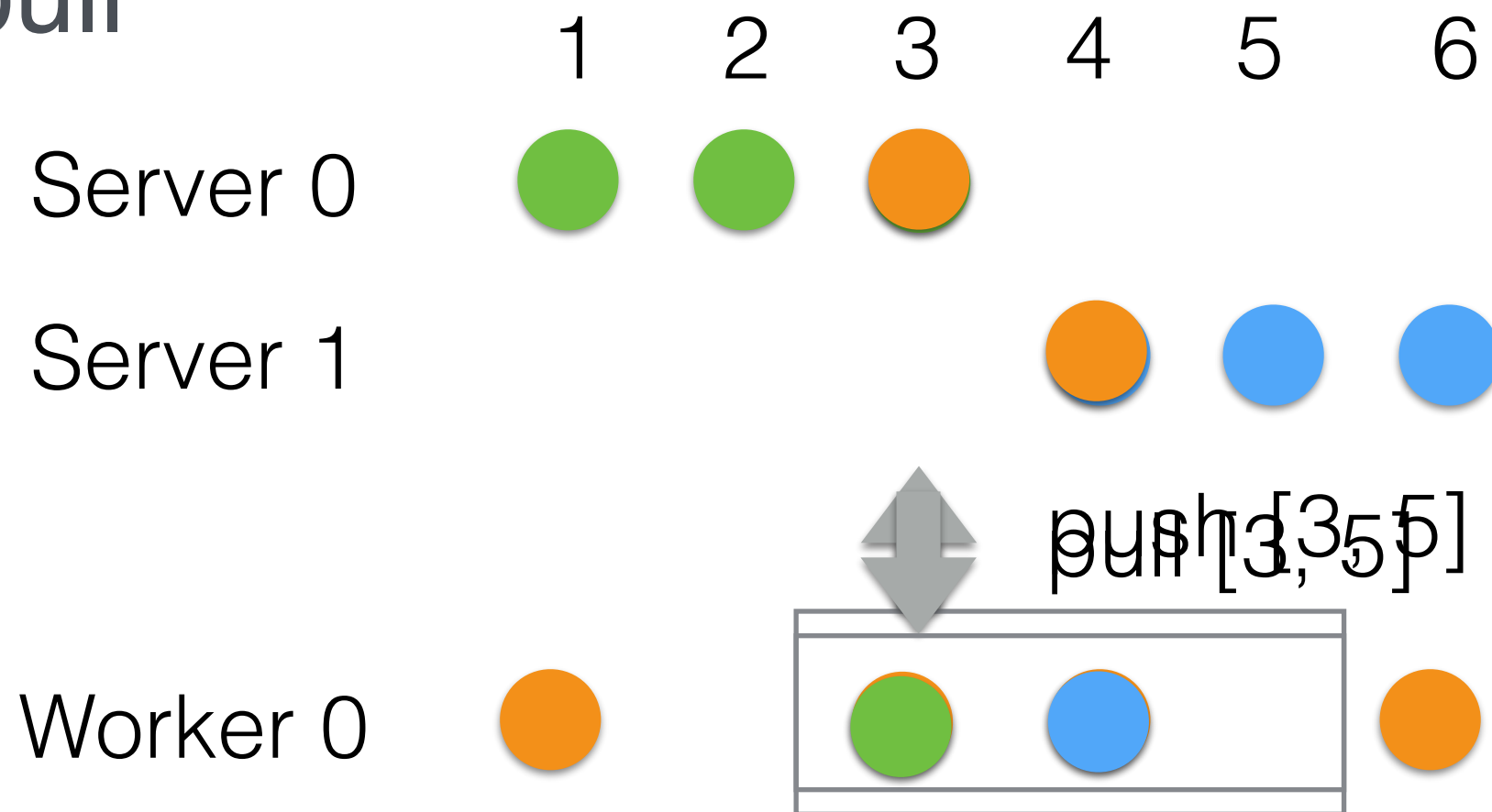


Localized Keys

- ♦ 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

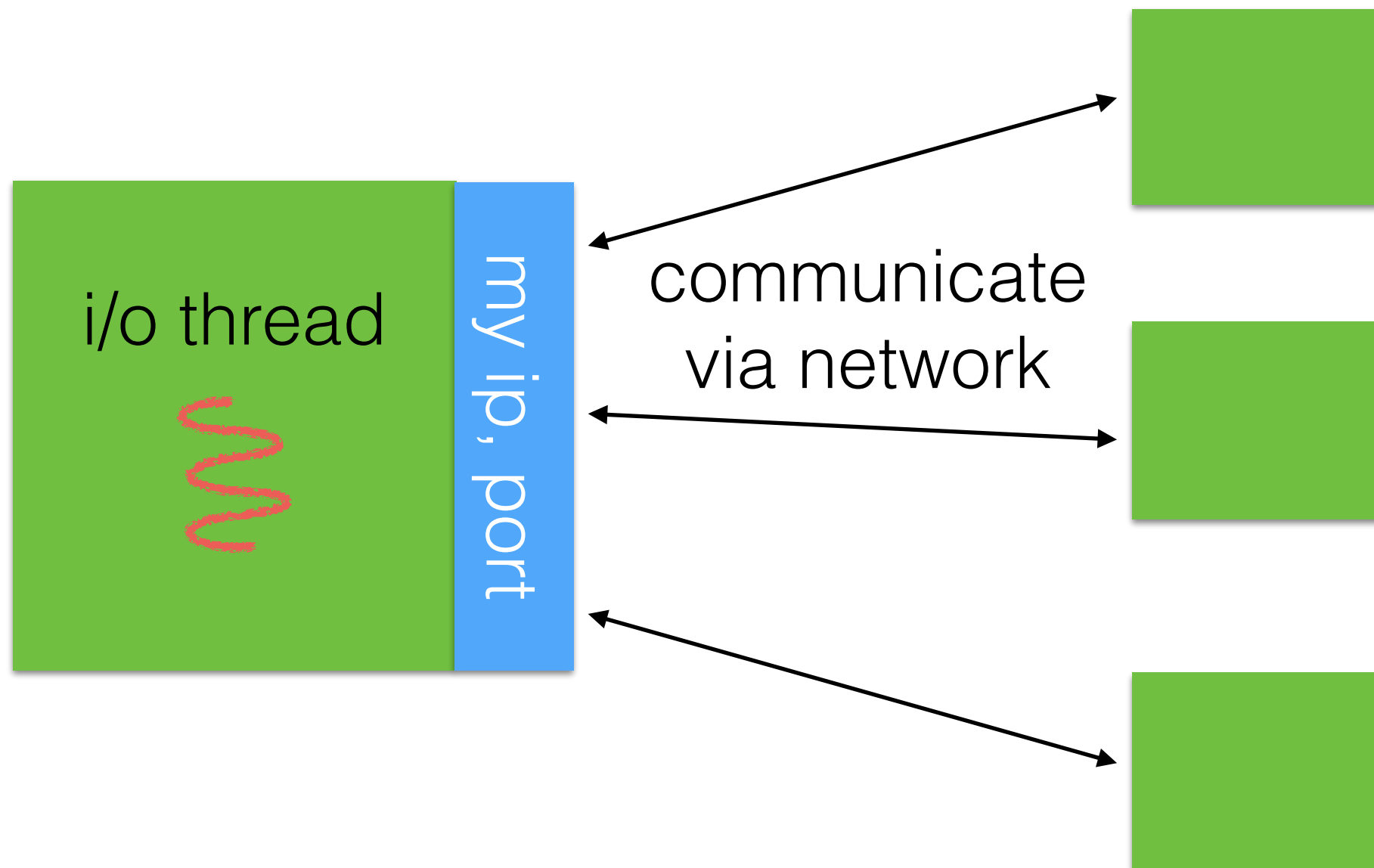
System components

- ◆ 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

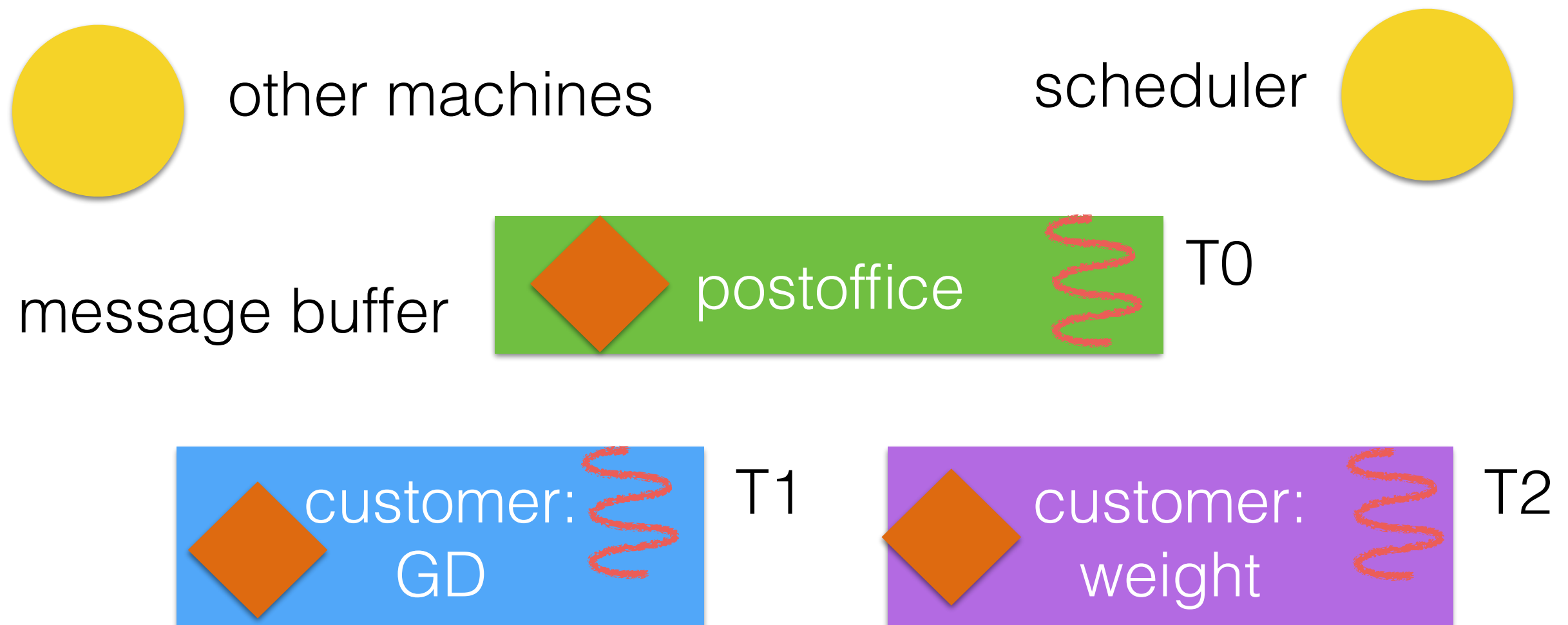
Van



- ♦ 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



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

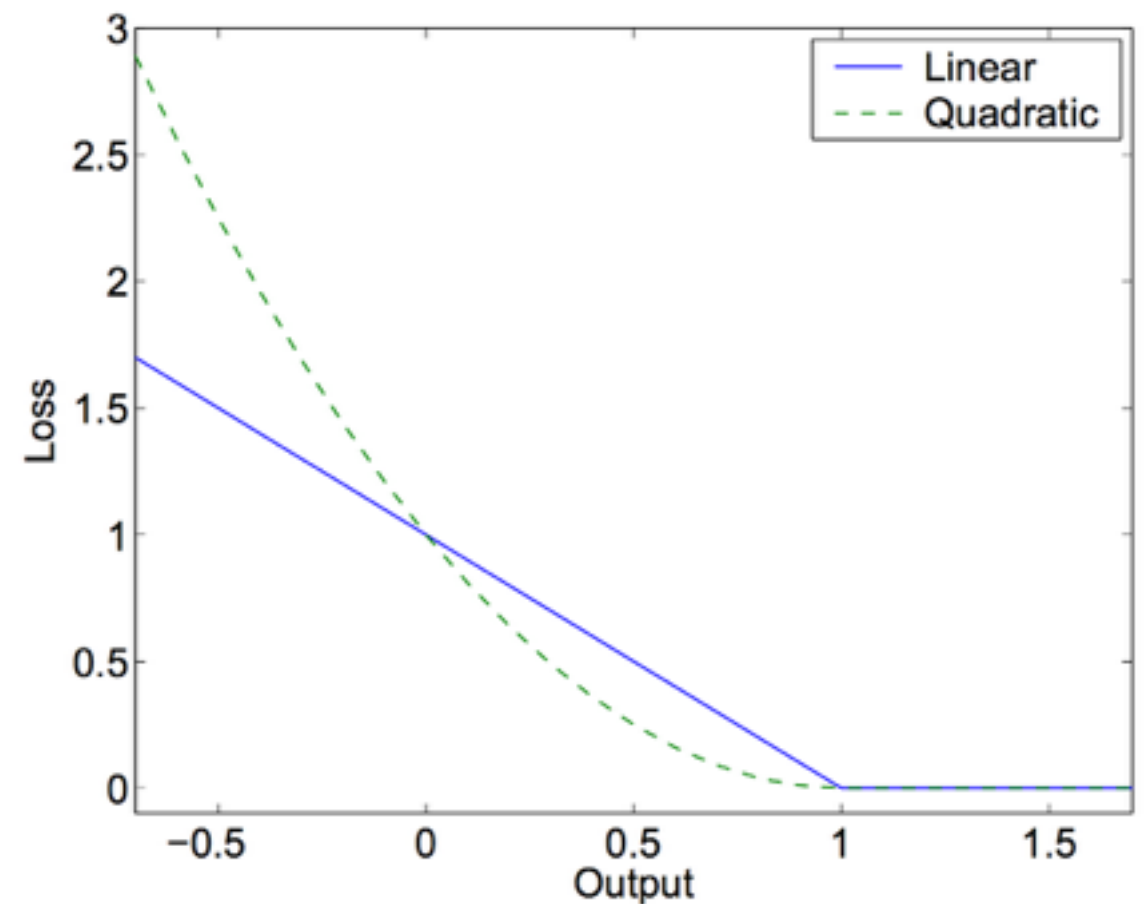
```
79:193 14942169923068616809:194 1876149161638679552:195 14747626071133469856:196 4336171036202202871:197 227542
023120650430:200 9318518278239696964:201 1316082436507117695:202 575942884264631676:204 575894505760120134:205
75869217003858363:206 575923093098738664:207 15659113762335723260:208 14358460775360110427:209 1520774004890315
756:209 4112554004588519437:231 8640740498329500703:232 2157884130424611056:233 17509169618099859841:234 189107
250861037178:235 2759606485408004053:236 15631838959359127081:236 3520579166615622293:236 6277520903722817690:2
7 3608808272766715750:237 17121157462558744300:237 7370011619909727872:238 1270396324967536174:239 169775446400
2812179:240 5305789169439131604:241 5373711697909914489:242 5416094781371033115:243 1960433140644839:244 111710
714681665:245 10690962950458853198:246 2839777854769539078:246 10518710414933393693:246 12390552841551525060:24
4539367741567243644:247 12218300306026065555:247 13049089524482127631:248 14748679415574799493:249 12424827760
50025518:250 279279972872262:254
1266355920254809400 1 0 13153889713695934697:1 18100530651055025325:7 1142392708993513:6 8742217430409874:15 17
40576521945757256:49 913695042229848:54 5616374076813480375:58 11991110041445596640:58 18057724478060466481:58
310722226892726644:127 14924322747647350103:85 2852314638569914752:85 15772457356035563224:85 89189290751847845
8:85 4553327236008865880:87 3150550196793221649:87 8918931274239689099:87 15708051881978934016:87 2350757346477
42:90 7135832082497995:90 11920906818518748:90 1563168796566011214:95 847150333188154181:95 5829905034113817010
95 3886986624173017168:96 210385768206166281:96 18057766260103652095:96 4080487034368285402:114 541479570208929
717:117 16715873079735457469:118 5022638387691639250:119 57556138661983649:120 6576594664184952662:2 1099237519
447707899:3 1995284382201631415:125 2154170522196769373:126 6282817907622730477:129 5527021448212153755:130 434
725969074943923:130 9405117706390927487:130 10339226503522335144:130 2350728885485622348:131 117343340634841251
131 6228825143664396080:131 7162933940795803737:131 8725464850117738613:131 7548169370980528781:131 1260356110
296512345:131 13537669905427920002:131 14792079286732608454:131 13614783807595398622:131 223431471201830570:131
115754026833238227:131 3198864593401287722:132 2021569114264077890:132 7076960851580061454:132 801106964871140
11:132 5527024746794510514:133 4349729267657300682:133 9405121004973284246:133 2350732184067979107:134 1173430
04930769275:134 6228828442246752839:134 8725468148700095372:134 7548172669562885540:134 12603564406878869104:13
14792082585314965213:134 13614787106177755381:134 223434769784187329:134 3198867891983644481:135 2021572412840
84649:135 7076964150162418213:135 62636982286248006:136 5468579414672459296:137 8615654827702746797:140 9470694
89183276065:141 16316885292642397467:144 1529936346522587265:155 1531063345987926754:156 18130167933071633366:1
7 1528813745167056788:158 1527688944756621805:159 1528815944221961294:160 1528536668292393991:162 1527409668827
54502:161 1527130392897487199:163 1528820342331770306:164 1533605417067791059:164 1538390491803811812:164 45578
500812859652:166 7005892858623431447:167 729030230544258213:172 4900284262846565233:183 2738768650365853478:18
```

Demo 2

- ♦ Implement the squared hinge loss

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

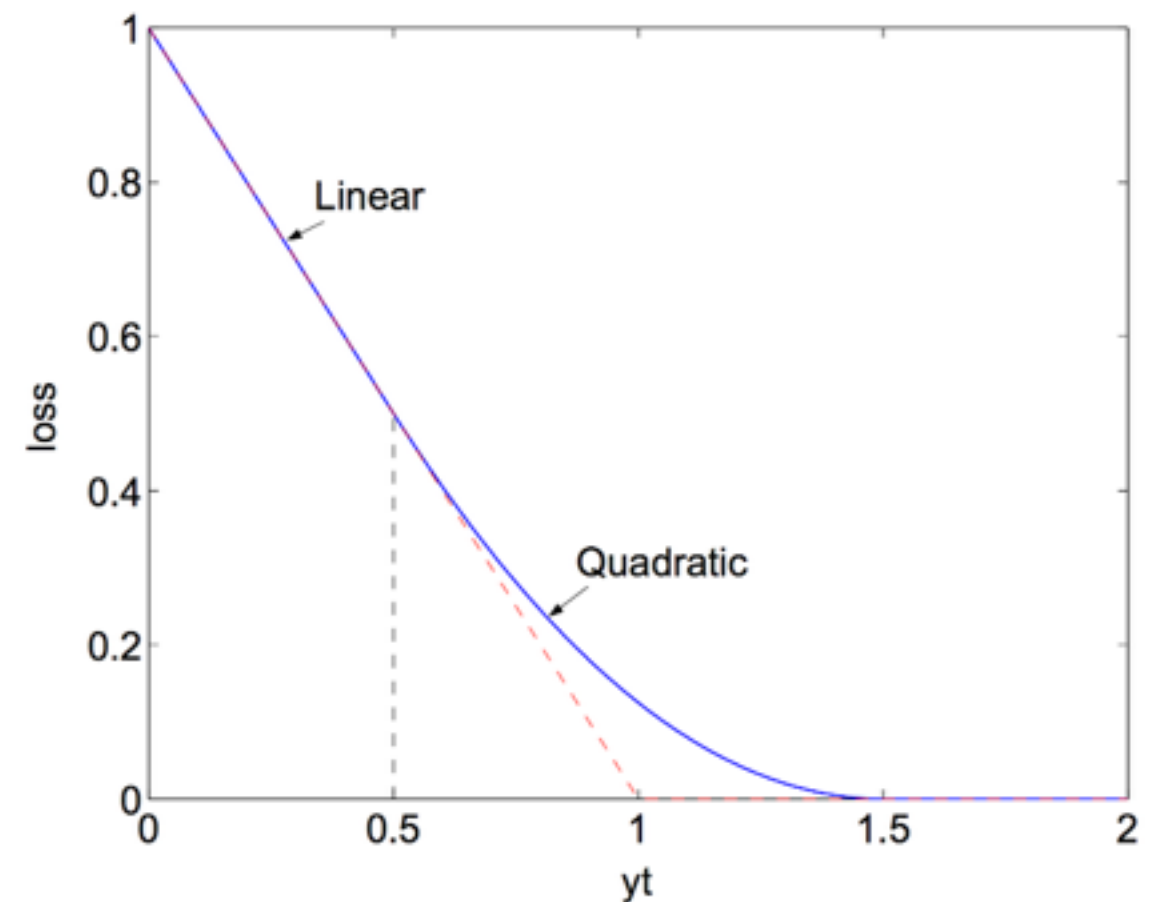
where $t = \langle w, x \rangle$



Exercise

- ♦ Implement the huber hinge loss

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



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