Distributed Machine Learning: An Intro.

Chen Huang

Feature Engineering Group,
Data Mining Lab,
Big Data Research Center, UESTC

Contents

Background

Why Distributed ML?

- **Big Data Problem**
  - ✓ Efficient Algorithm

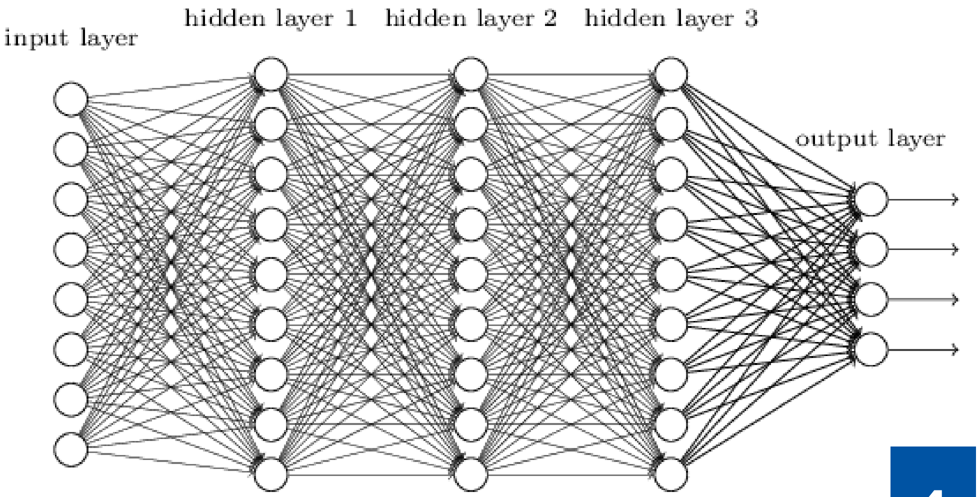
☹

  - ✓ Online Learning / Data Stream
    - Feasible.
    - What about high dimension?
  - ✓ Distributed Machine
    - The more, the merrier

Background

Why Distributed ML?

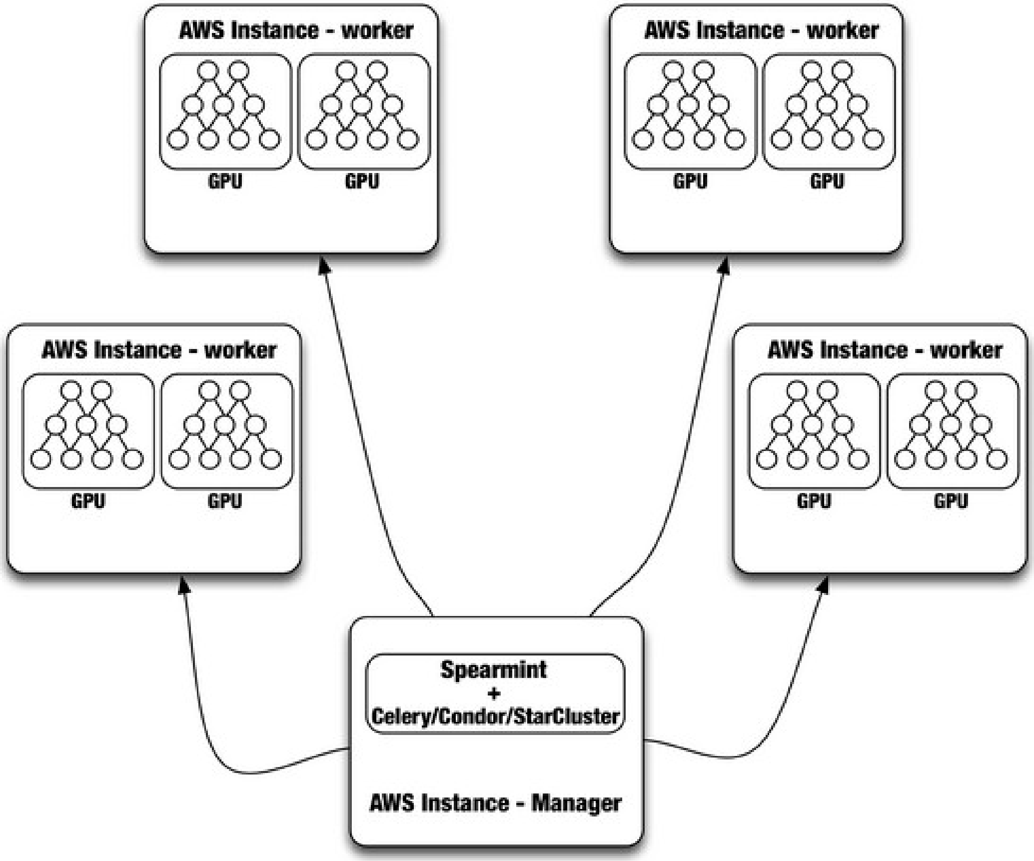
- **Big Data**
  - Efficient Algorithm
  - Online Learning / Data Stream
  - Distributed Machine
- **Big Model**
  - Model Split
  - Model Distributed



Background

Distributed Machine Learning

- **Big model over big data**



Background Overview

Distributed Machine Learning

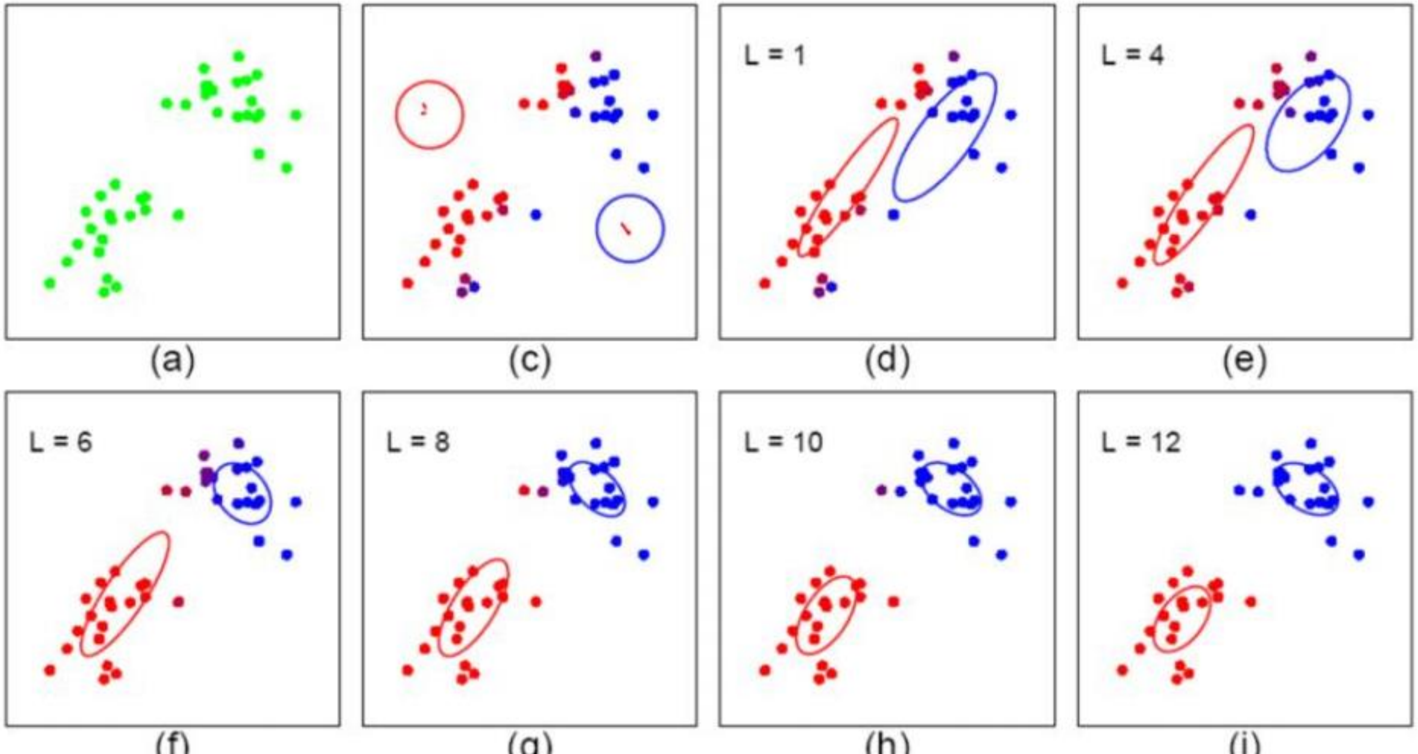
- **Motivation**
  - Big model over big data
- **DML**
  - Multiple workers cooperate each other with communication

- **Target**
  - Get the job done *(convergence, …)*
  - Min communication cost *(IO, …)*
  - Max effect *(Time, performance…)*

Example

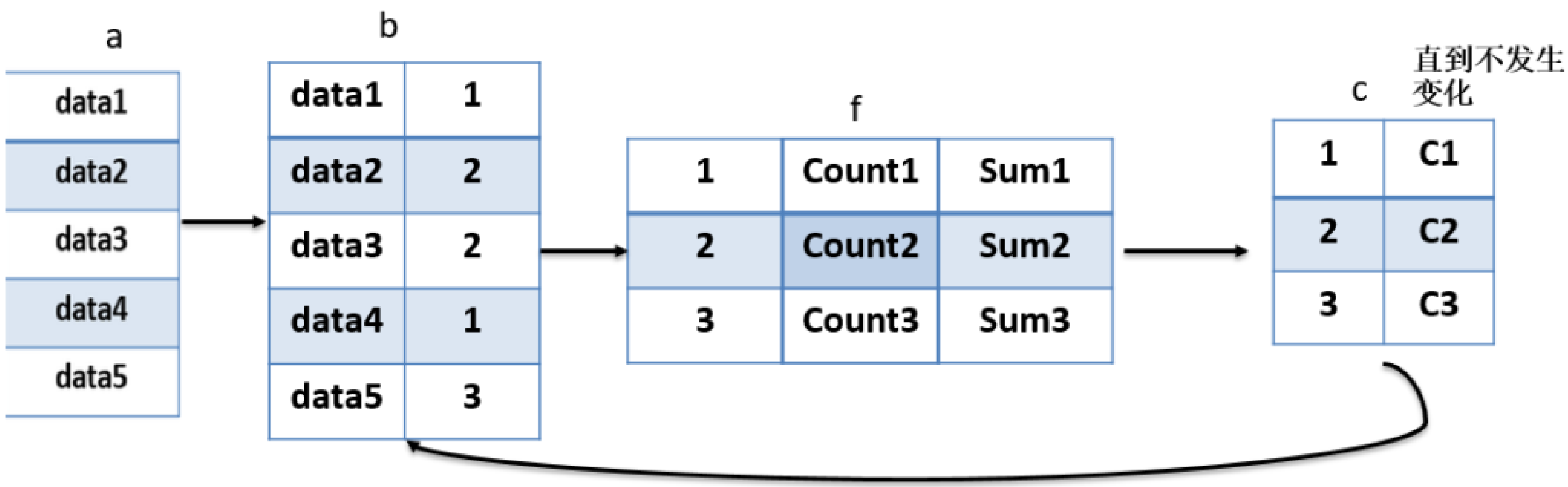
K-means

E-step	M-step
$Z_i^t = argmin_k\, (X_i - \mu_k^t)^T (\Sigma_k^t)^{-1} (X_i - \mu_k^t)$	$\mu_k^{t+1} = \frac{\sum_i \delta(Z_i^t, k) X_i}{\sum_i \delta(Z_i^t, k)}$



Example

Distributed K-means



Example

Spark K-means

```
val data:RDD[Array[Double]] = sc.textFile("hdfs://localhost:9000/kmeansData")
  .map(x => x.split(" ").map(_.toDouble))
val centers = data.takeSample(withReplacement = false, num = numClusters)
val dim = centers(0).length
```

Example

Spark K-means

```
while(currIter <= maxIter && !ConsistentFlag){
  println("====run time " + currIter)
  val centersBro:Broadcast[Array[Array[Double]]] = sc.broadcast(centers)
  val clusters = data.map{x =>
    var index = 0
    var min = Double.MaxValue
    var minIndex = -1
    for(index <- 0 until numClusters){
      val distance = calDis(x, centersBro.value(index))
      if(min > distance){
        minIndex = index
        min = distance
      }
    }
    (minIndex, x)
  }
```

Example

Spark K-means

```
//check
diff = 0.0
for(i <- 0 until numClusters){
  diff += calDis(newCenters.getOrElse(i, centersBro.value(i)), centersBro.value(i))
}
newCenters.keys.foreach{ index =>
  centers(index) = newCenters(index)
}
if(diff <= thred){
  ConsistentFlag = true
}
currIter += 1
}
println("diff = " + diff +" runTime = " + currIter)
centers.foreach(x => println(x.mkString(",")))
```

Example

Item filter

- Given two files, you need to output key-value pairs in file B, whose key exists in file A.
- File B is super large. (e.g. 100GB)

A

Key1
Key12
Key3
Key5
…

Key1, val1
Key2, val2
Key4, val4
Key3, val3
Key5, val5
…

B

- **What if A is also super large?**

Example

Item filter

A

Key1
Key12
Key3
Key5
…

→

Key1

Key12
Key3

Key5

…….

B

Key1, val1
Key2, val2
Key4, val4
Key3, val3
Key5, val5
…

→

Key1, val1

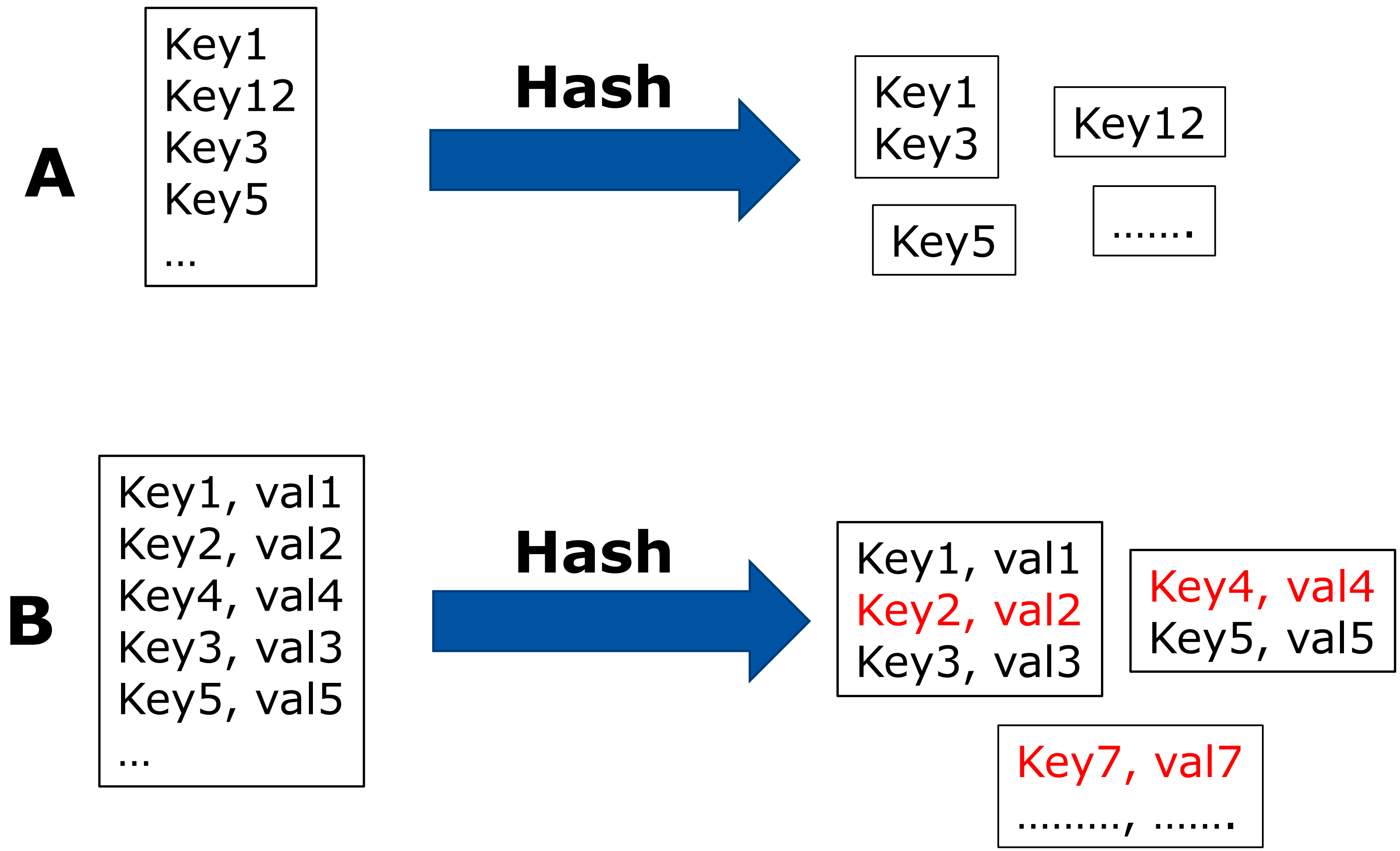
Key2, val2
Key4, val4

Key3, val3
Key5, val5
…

………, …….

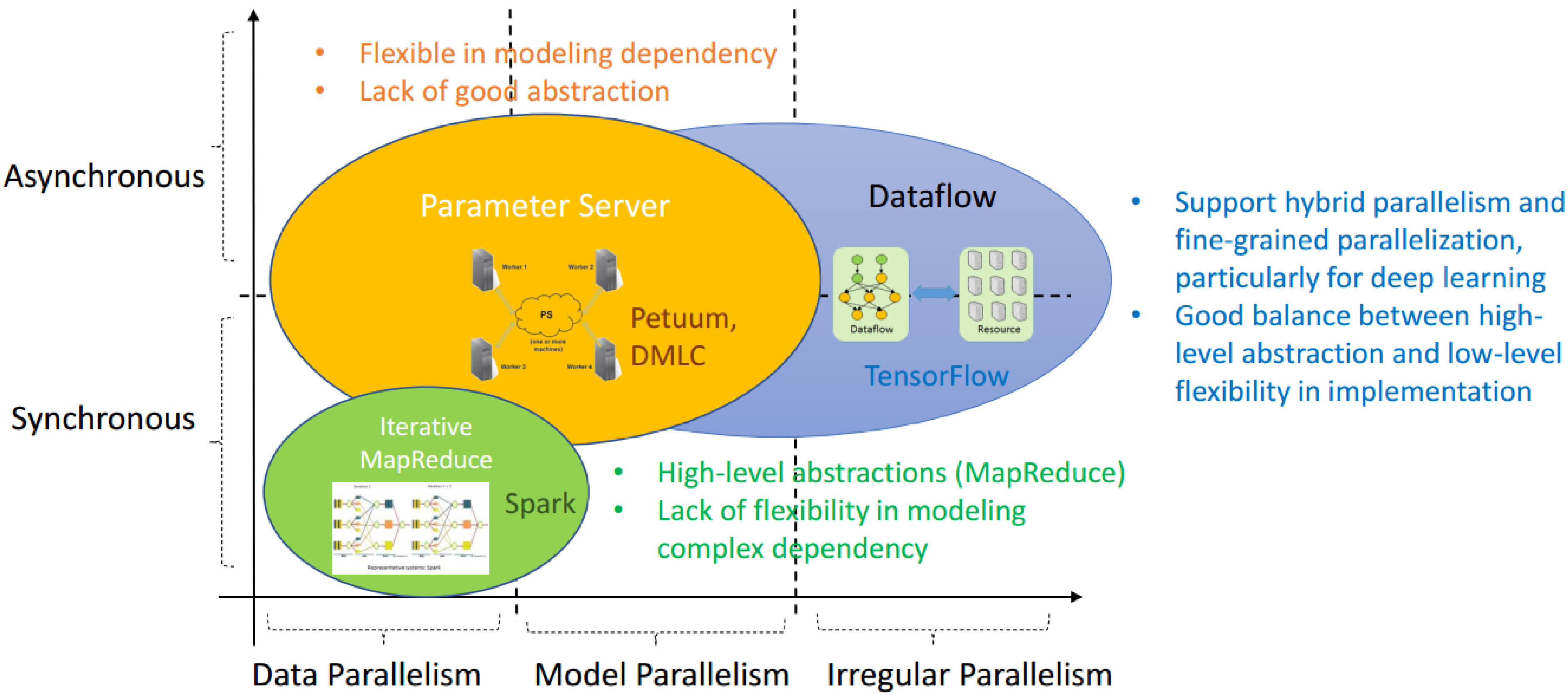
Example

Item filter



Distributed Machine Learning

Overview



* *AAAI 2017 Workshop on Distributed Machine Learning* for more information

Distributed Machine Learning
How To Distribute

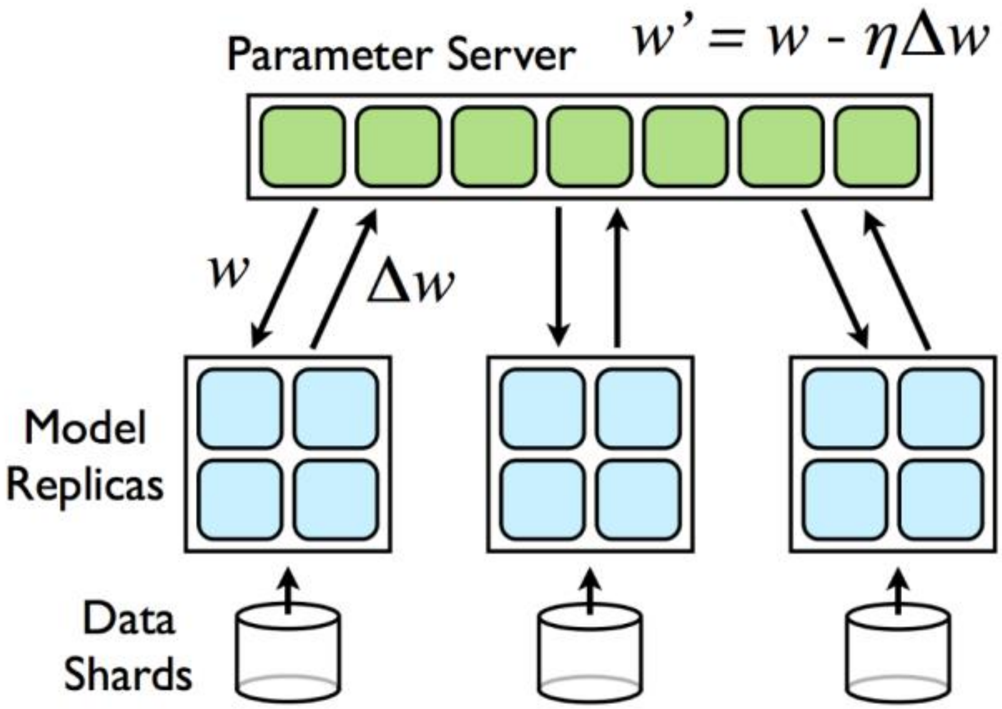
Key Problems

- **How to “split”**
  - Data parallelism / model parallelism
  - Data / Parameters dependency
- **How to aggregate messages**
  - Parallelization mechanisms
  - Consensus between local & global parameters
  - Does algorithm converge
- **Other concerns**
  - Communication cost, …

Distributed Machine Learning
How To Split

How To Distribute

– Data Parallelism



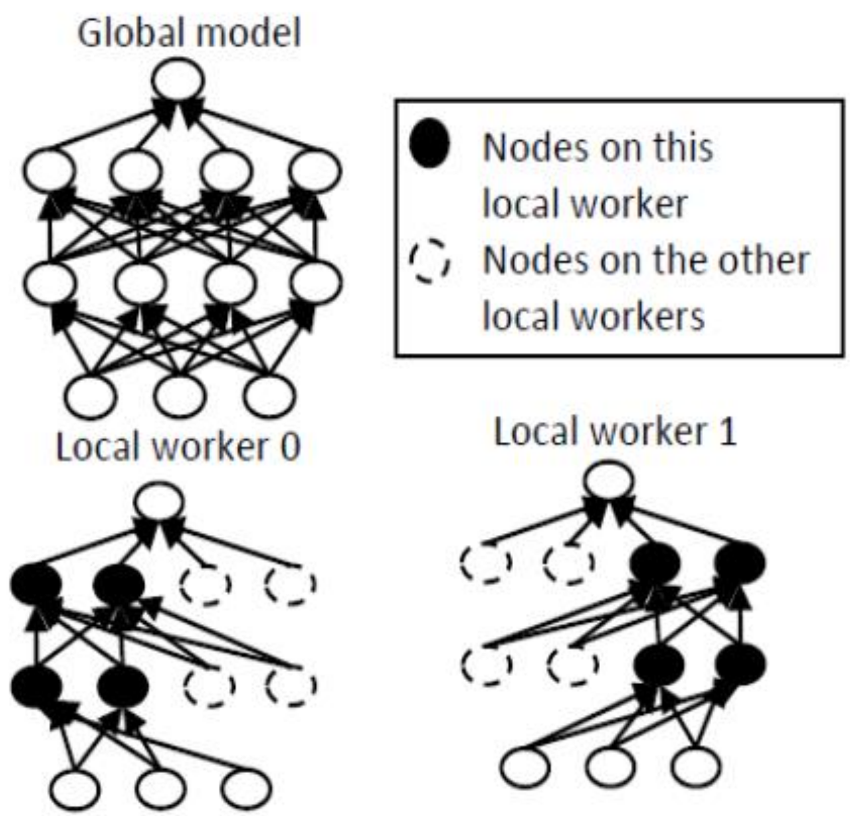
1. Data partition
2. Parallel training
3. Combine local updates
4. Refresh local model with new parameters

Distributed Machine Learning
How To Split

How To Distribute

– Model Parallelism

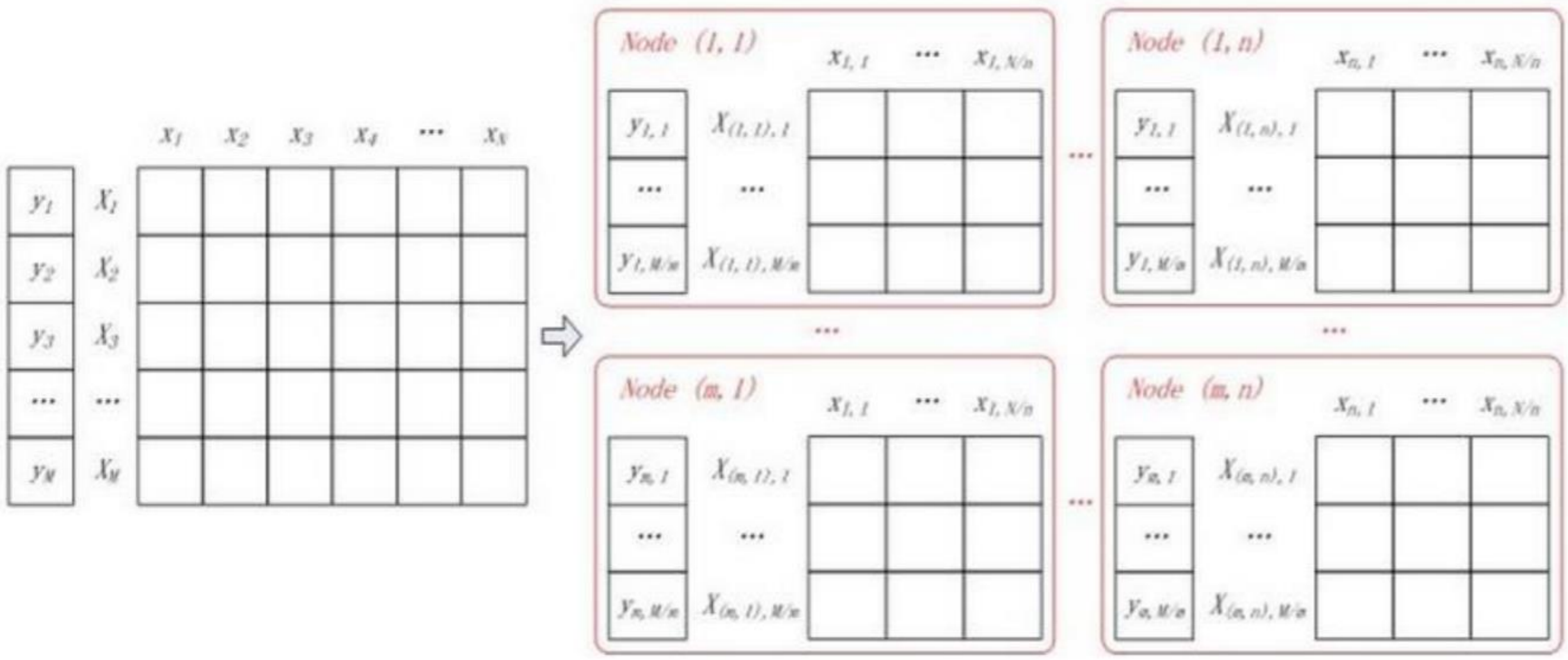
1. Partition model into multiple local workers
2. Workers collaborate with each other to perform optimization



Distributed Machine Learning
How To Split

How To Distribute

- **Model Parallelism & Data Parallelism**



Example: Distributed Logistic Regression

Distributed Machine Learning
How To Split

Categories

- **Data Parallelism**
  - Split data into many samples sets
  - Workers calculate the same parameter(s) on different sample set
- **Model Parallelism**
  - Split model/parameter
  - Workers calculate different parameter(s) on the same data set
- **Hybrid Parallelism**

Distributed Machine Learning
How To Split

Data / Parameter Split

- **Data Allocation**
  - Random selection. (Shuffling)
  - Partition. (e.g. Item filter, word count)
  - Sampling
  - Parallel graph calculation (for non-i.i.d. data)

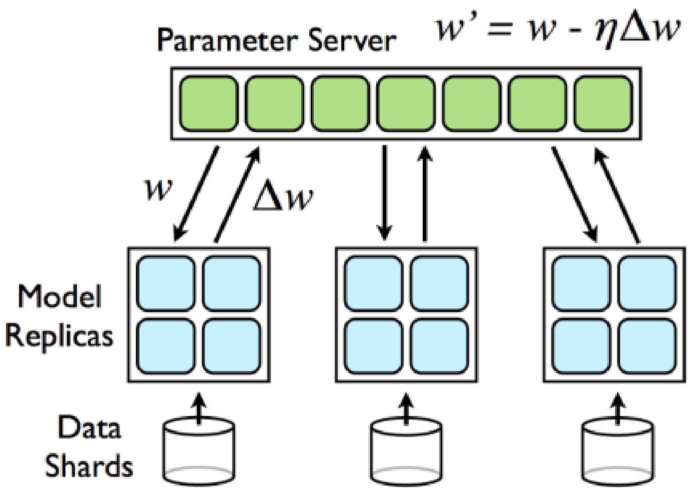
- **Parameter Split**
  - Most algorithms assume parameter independent and randomly split parameters
  - **Petuum** *(KDD'15, Eric Xing)*

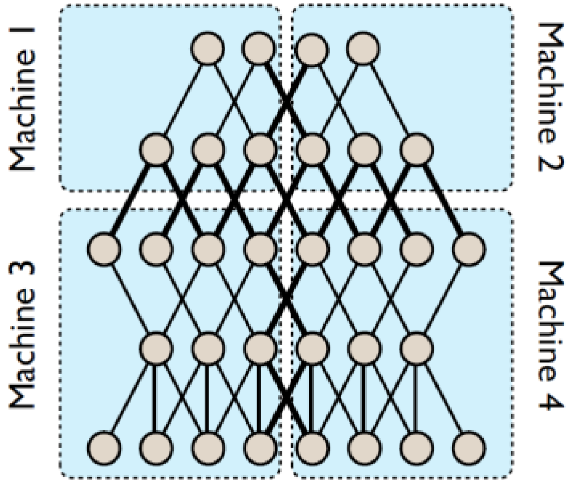
Distributed Machine Learning
How To Aggregate Messages

Parallelization Mechanisms

- Given the feedback $g_i(w)$ of worker i, how can we update the model parameter W?

$$W = f\left(g_1(w), g_2(w), \ldots, g_m(w)\right)$$





Distributed Machine Learning Parallelization Mechanism

Bulk Synchronous Parallel (BSP)

- **Synchronous update**
  - Update parameter until all workers are done with their job

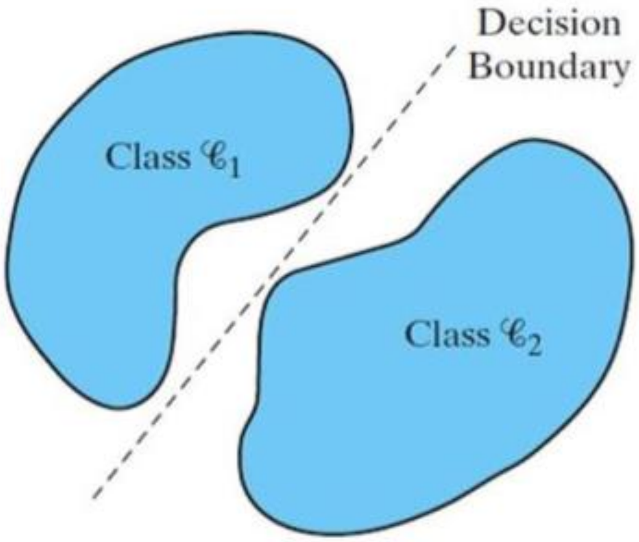
```
W = initValue;
Workers.foreach{
        worker => worker.doJob(W);
}
Update(W, workers.values());
```

  - Example: *Sync SGD (Mini-batch SGD)* , *Hadoop*

Distributed Machine Learning

Sync SGD

– Perceptron



$$f(x) = sign(wx + b)$$

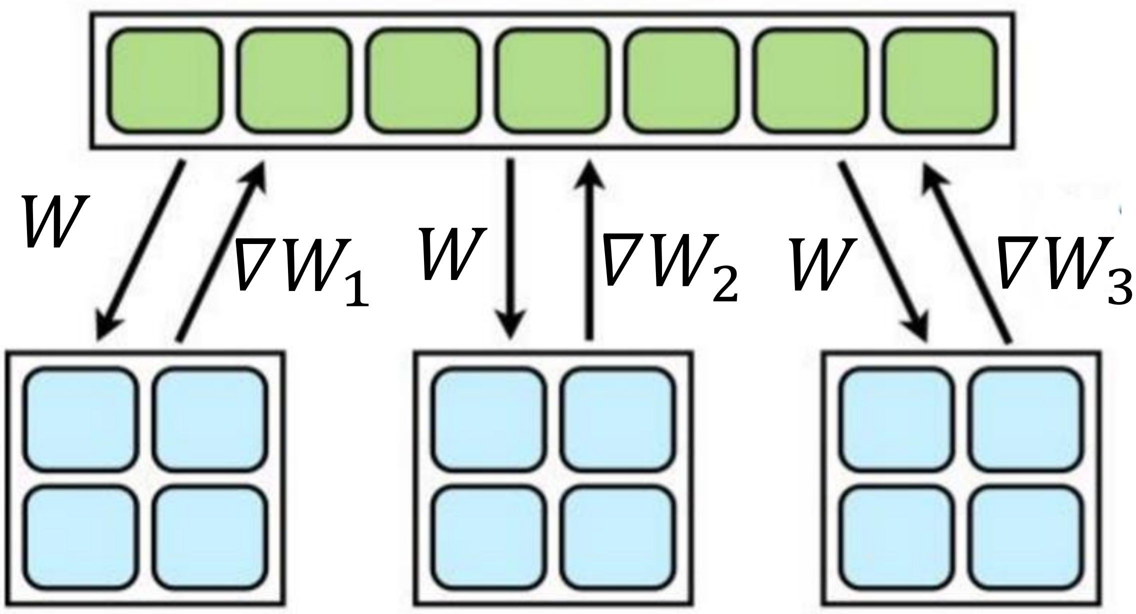
$$sign(\mathrm{x}) = \begin{cases} -1, x < 0 \\ +1, x \geq 0 \end{cases}$$

$$L(w, b) = -\sum_{x_i \in M} y_i(wx_i + b)$$

Distributed Machine Learning

Sync SGD

$$W \leftarrow W - \sum \nabla W_i$$



$$\nabla W_3 = -\sum_{x_i \in M} x_i y_i$$

Distributed Machine Learning Parallelization Mechanism

Asynchronous Parallel

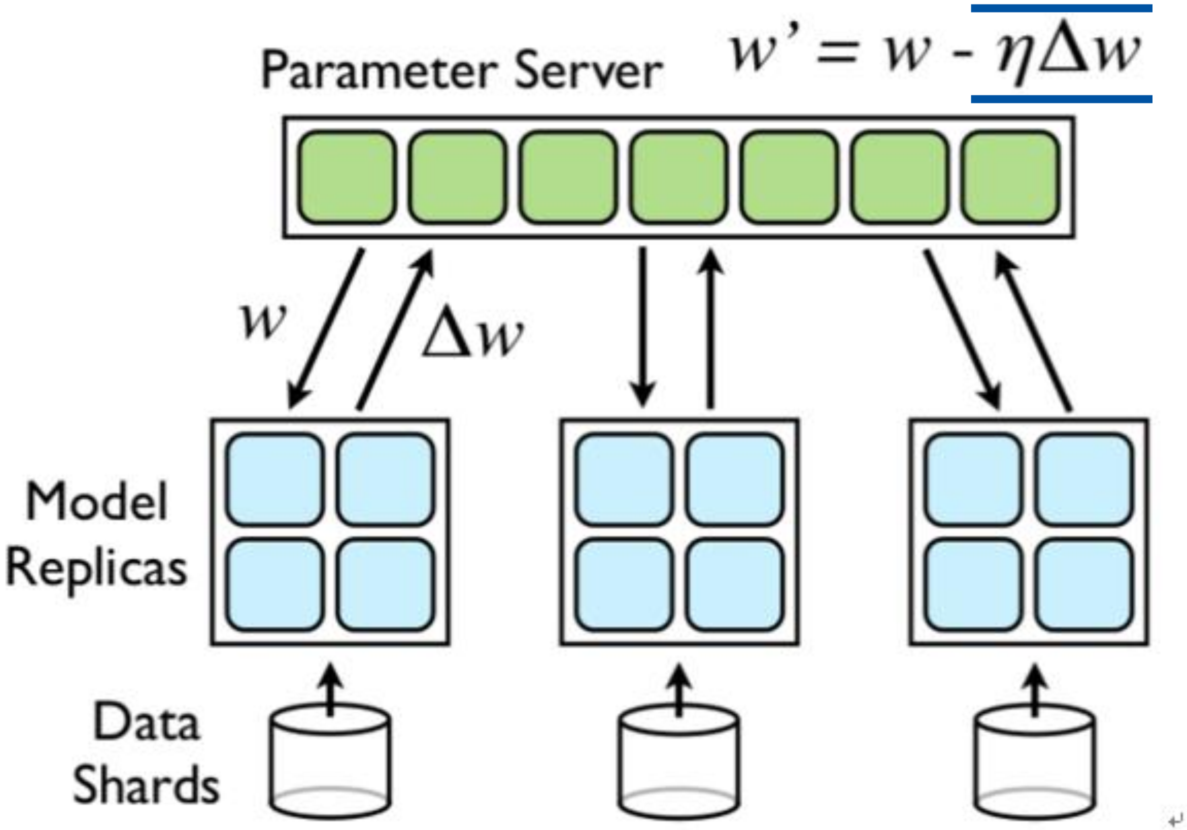
- **Asynchronous update**
  - Update parameter whenever received the feedback of workers

```
W = initValue;
Workers.foreach{
        Update(worker.doJob(), W);
}
```

  - Example: *Downpour SGD (NIPS'12)*

Distributed Machine Learning

Downpour SGD



Distributed Machine Learning

Async. V.S. Sync.

- **Sync.**
  - Single point of failure: it has to wait until all workers finished his job. The overall efficiency of algorithm is determinated by the slowest worker.
  - Nice convergence

- **Async.**
  - Very fast!
  - Affect the convergence of algorithm. (e.g. expired gradient)
  - Use it, if model is not sensitive to async. update

Distributed Machine Learning Parallelization Mechanism

ADMM for DML

- **Alternating Direction Method of Multipliers**
  - Augmented Lagrangian + Dual Decomposition

$$\min_x f_1(x_1) + f_2(x_2) \text{ s.t. } A_1 x_1 + A_2 x_2 = b$$

$$\Downarrow$$

For DML case: replace x_2^{k-1} with $mean(x_2^{k-1})$ and x_1^k with mean$\left(x_1^k\right)$ when updating

$$x_1^k = argmin_{x_1} f_1(x_1) + \frac{\rho}{2} \left\| A_1 x_1 + A_2 x_2^{k-1} - b + w^{k-1} \right\|_2^2$$

$$x_2^k = argmin_{x_2} f_2(x_2) + \frac{\rho}{2} \left\| A_1 x_1^k + A_2 x_2 - b + w^{k-1} \right\|_2^2$$

$$w^k = w^{k-1} + A_1 x_1^k + A_2 x_2^k - b$$

- Famous optimization algorithm for both industrial and academic. (e.g. computing advertising)

Distributed Machine Learning Parallelization Mechanisms

Overview

- Sync.
- Async
- ADMM
- Model Average
- Elastic Averaging SGD *(NIPS'15)*
- Lock Free: Hogwild! *(NIPS'11)*
- ……

Distributed ML Framework

Distributed Machine Learning Frameworks

This is a joke, please laugh…

机器学习相关岗位面试中，有哪些加（zhuang）分（bi）项？

我：目前深度学习当中用mapreduce的比较少，因为我们经常要SGD，

M：哦我猜一下，所以你们用MPI，然后你要优化Allreduce。

我：。。。对的，然后很多时候网络会有瓶颈，

M：恩，因为你们不想上infiniband。

我：。。。对的，

M：然后你们 网络的 吞吐速度是够的，但是延迟不理想。

我：。。。对的，

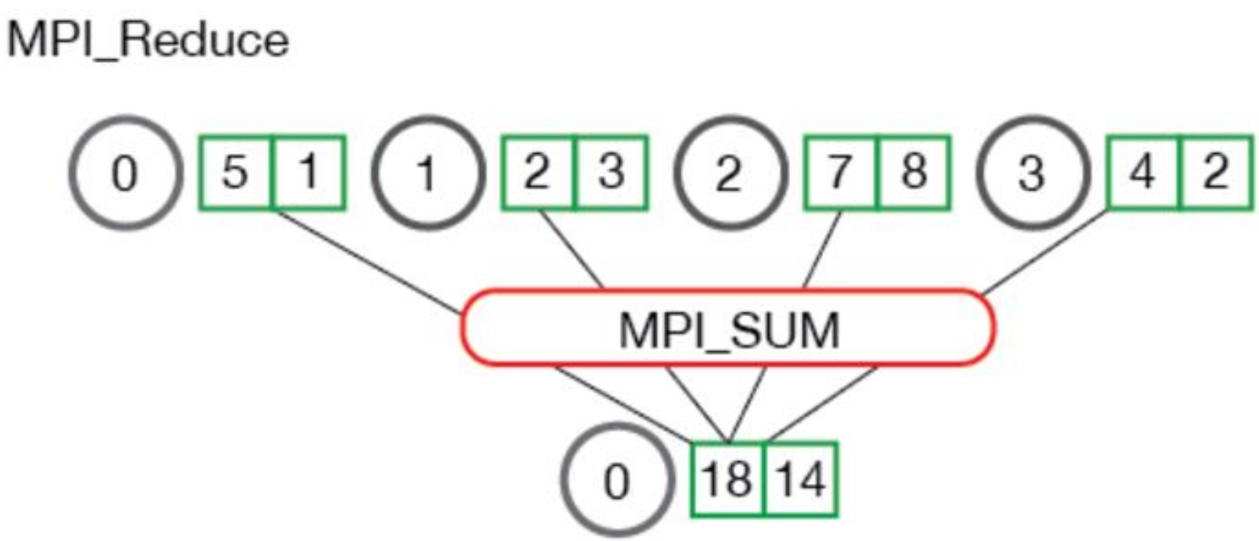
M：所以你们想要有异步通信，但是同时又要控制 模型不发散。

我：。。。对的。

Distributed Machine Learning Frameworks

Message Passing Interface (MPI)

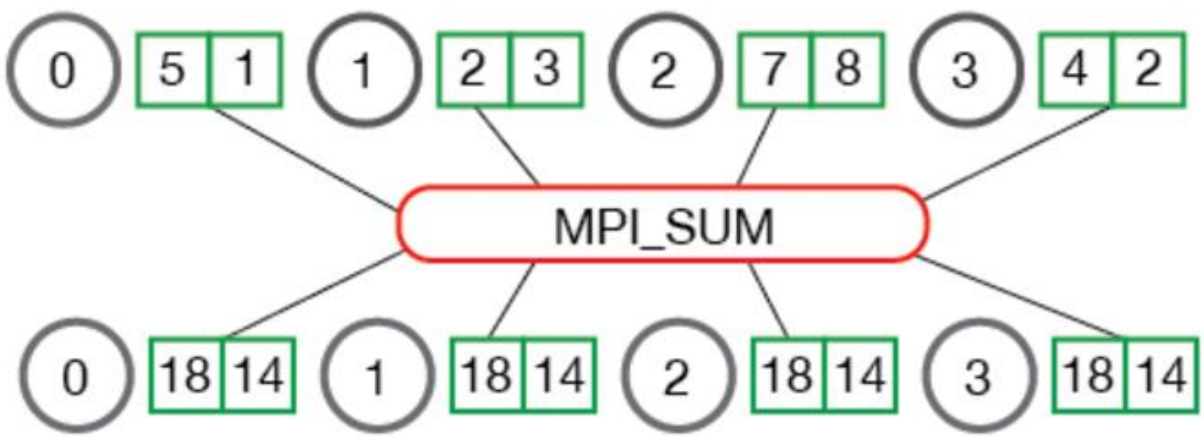
- Parallel computing architecture
- Many operations:
  - send, receive, broadcast, scatter, gather…



Distributed Machine Learning Frameworks

Message Passing Interface (MPI)

- Parallel computing architecture
- Many operations:
  - **AllReduce** = reduce + broadcast

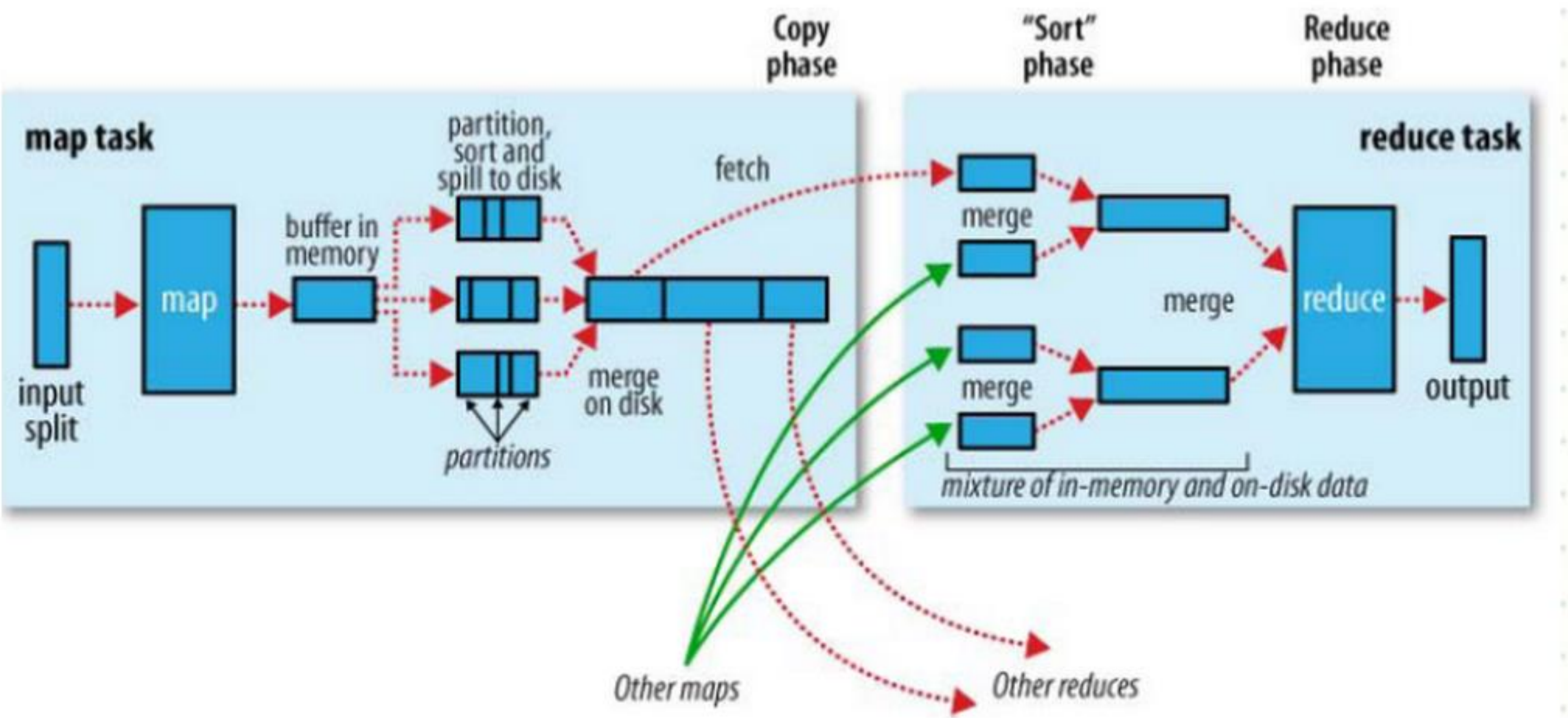


- **Hard to write code!**

Distributed Machine Learning Frameworks

MapReduce

- Well-encapsulated code, user-friendly!
- Designed scheduler,
- Integration with HDFS / fault-tolerant /….



Distributed Machine Learning Frameworks

MapReduce

- **Synchronous parallel, single point of failure.**
- **数据溢写** (I don't know how to translate…)

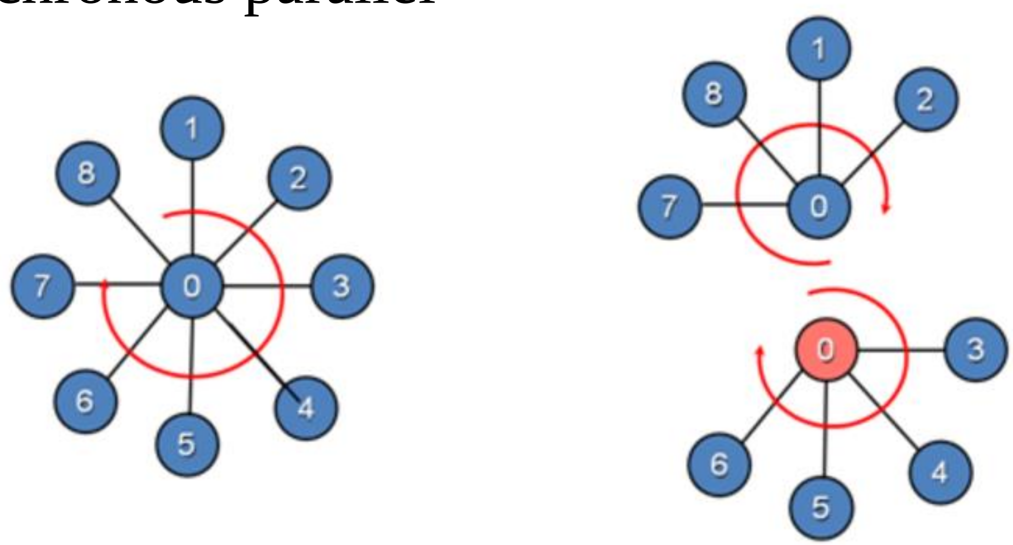
- **Not so suitable for machine learning task.**
  - Many ML models are solved in iterative manner, and Hadoop/MapReduce does not naturally support iteration calculation
  - Spark does

- Iterative MapReduce Style Machine Learning Toolkits
  - Hadoop Mahout
  - Spark MLlib

Distributed Machine Learning Frameworks

GraphLab *(UAI'10, VLDB'12)*

- **Distributed computing framework for graph**
- Split graph into sub-graphs by **<u>node cut</u>**
- Asynchronous parallel



V0=sum(V1,V2,…,V8)

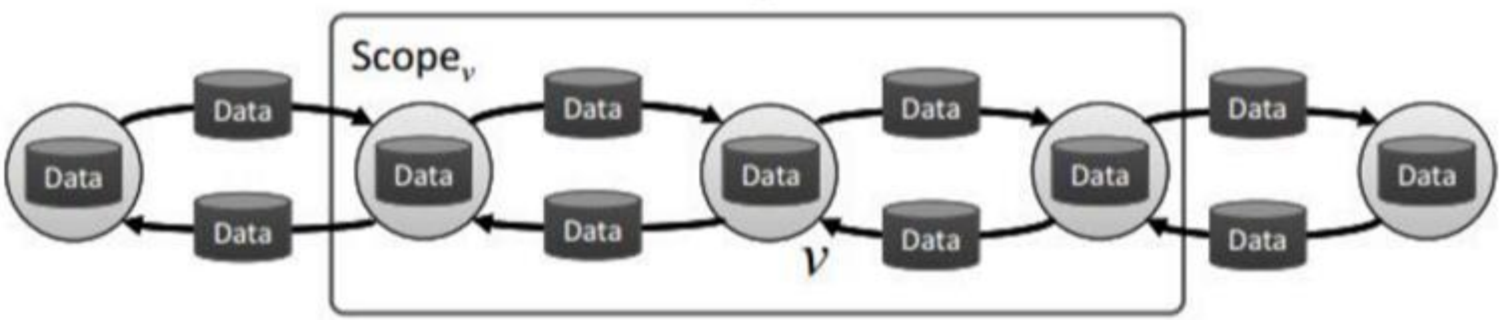
V00=sum(V1,V2,V7,V8)
V01=sum(V3,V4,V5,V6)
V0=sum(V00,V01)

Distributed Machine Learning Frameworks

GraphLab *(UAI'10, VLDB'12)*

- Data Graph + Update Function + Sync Operation
- **Data Graph**
- **Update function**: user-defined function, working on scopes
- **Sync**：global parameter update

Scope allows overlapping

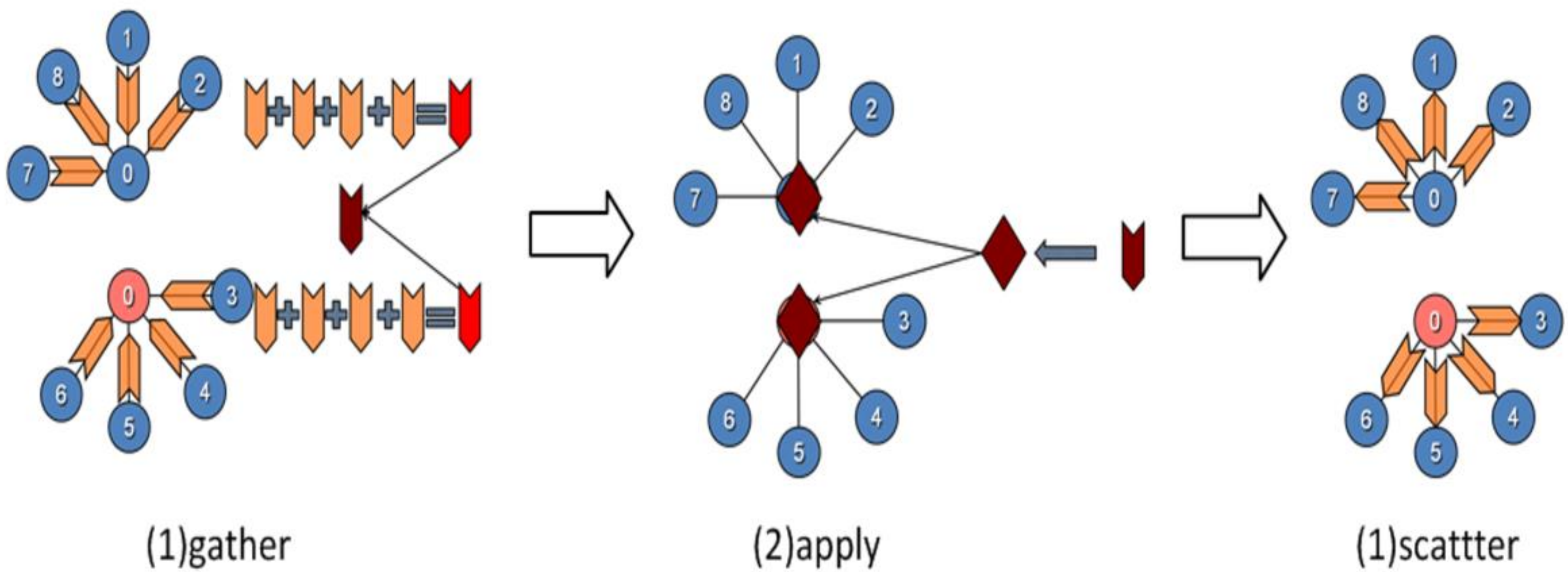


(a) Scope

Distributed Machine Learning Frameworks

GraphLab *(UAI'10, VLDB'12)*

- Data Graph + Update Function + Sync Operation
- Three Steps = Gather + Apply + Scatter



Read Only **Write Node Only** **Write Edge Only**

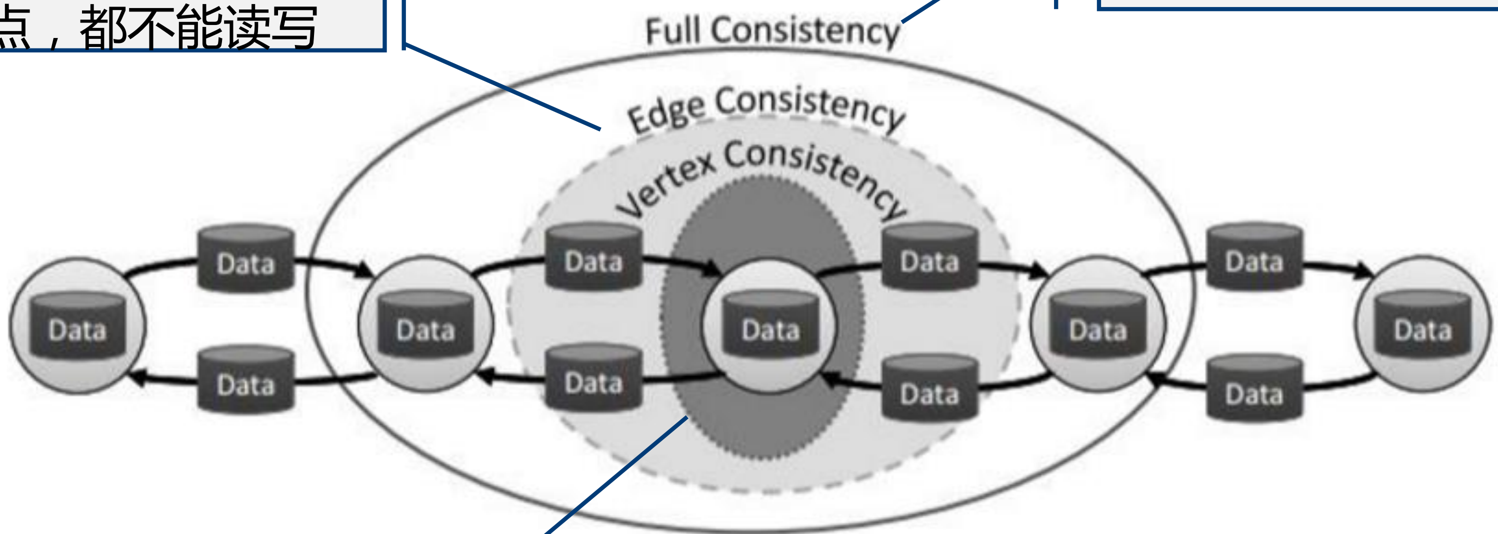
Distributed Machine Learning Frameworks

GraphLab: Consistency Control

– **Trade-off between conflict and parallelization**

Scope内，除了邻居节点，都不能读写

Scope内不能读写



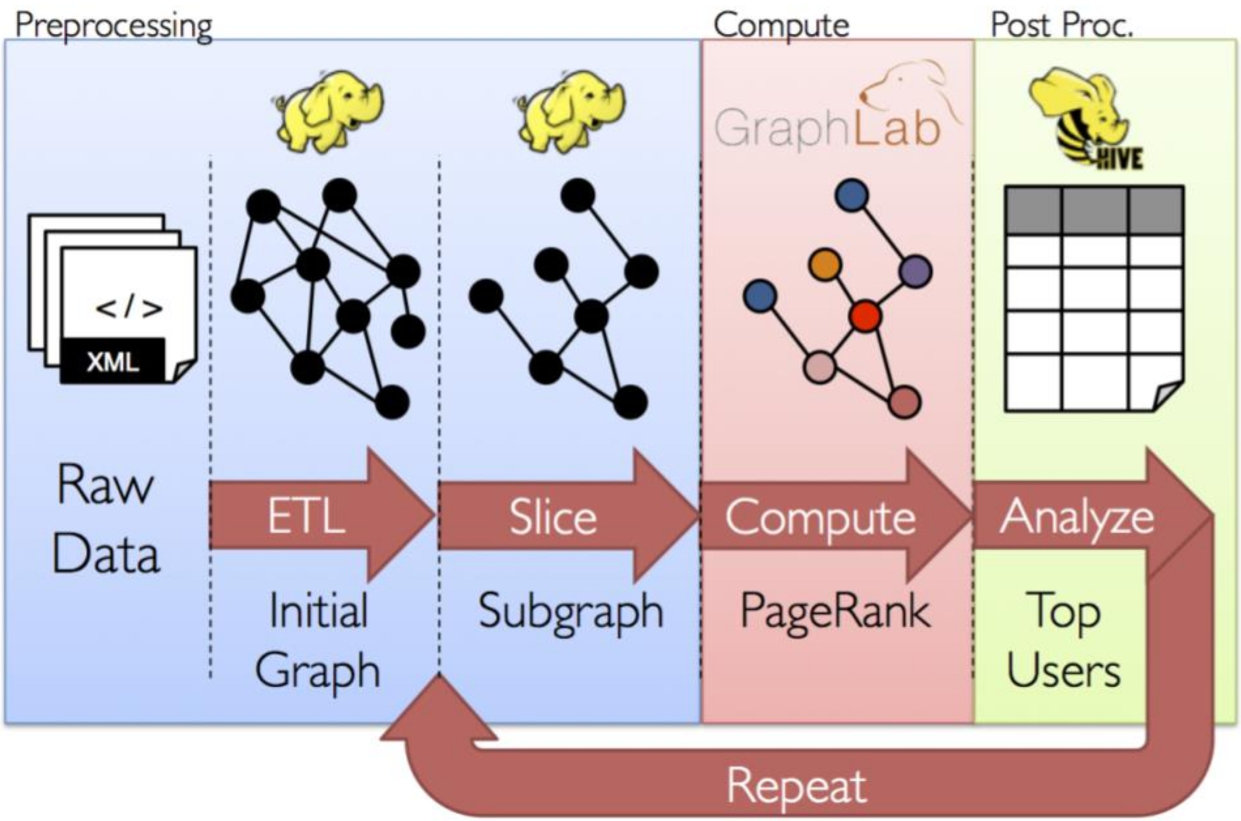
(b) Consistency Models

一次更新中，其他操作不能读写该节点

Distributed Machine Learning Frameworks

Spark GraphX

- Avoid the cost of moving sub-graphs among workers by combining Table view & Graph view

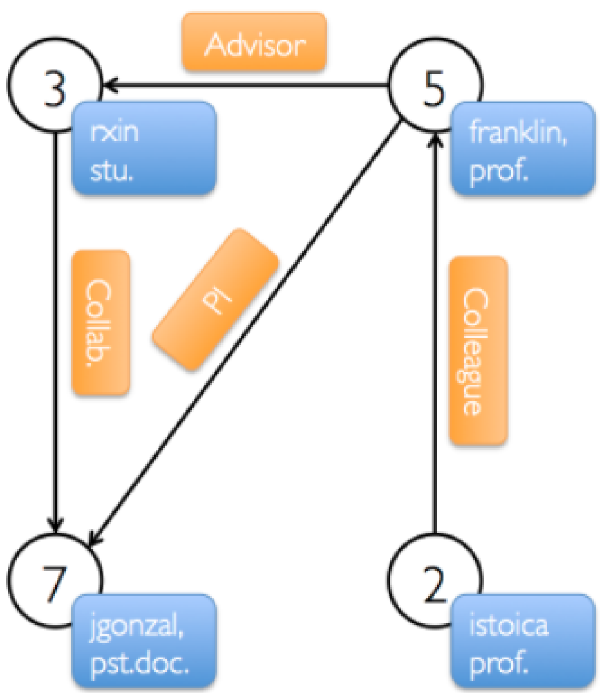


Distributed Machine Learning Frameworks

Spark GraphX

– Avoid the cost of moving sub-graphs among workers by combining Table view & Graph view

Property Graph



Vertex Table

Id	Property (V)
3	(rxin, student)
7	(jgonzal, postdoc)
5	(franklin, professor)
2	(istoica, professor)

Edge Table

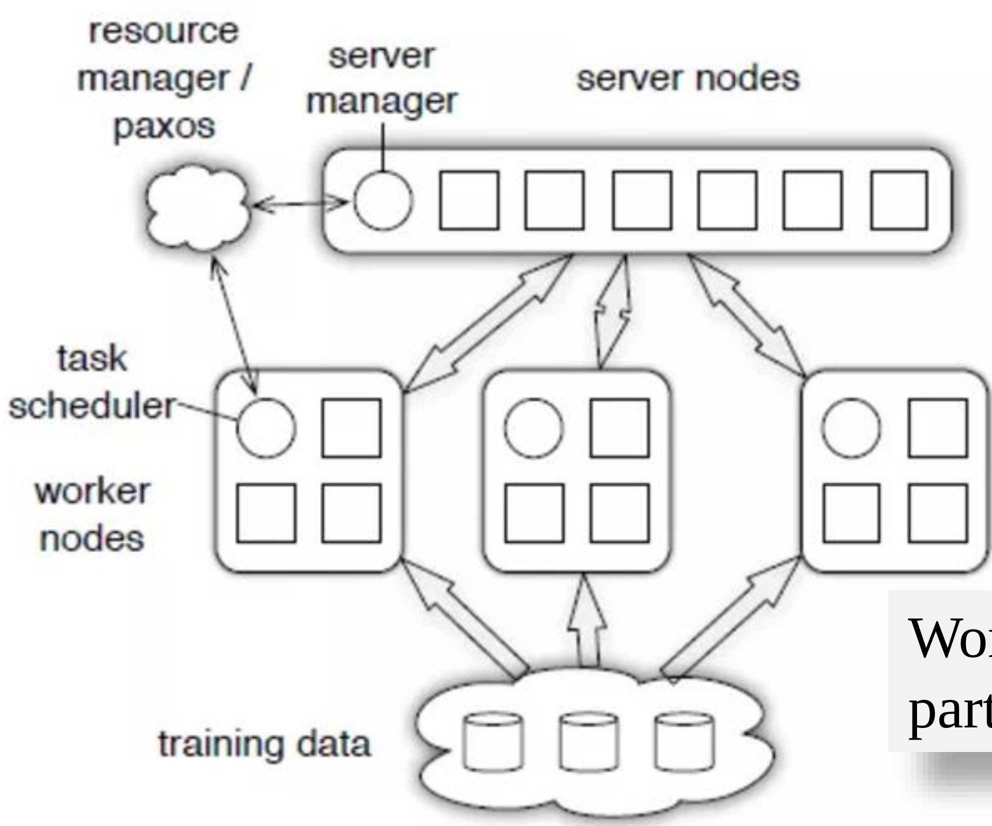
SrcId	DstId	Property (E)
3	7	Collaborator
5	3	Advisor
2	5	Colleague
5	7	PI

Distributed Machine Learning Frameworks

Parameter Server

1. Workers query for current parameters
2. Parameters are stored in distributed way, among server nodes

- Asynchronous parallel

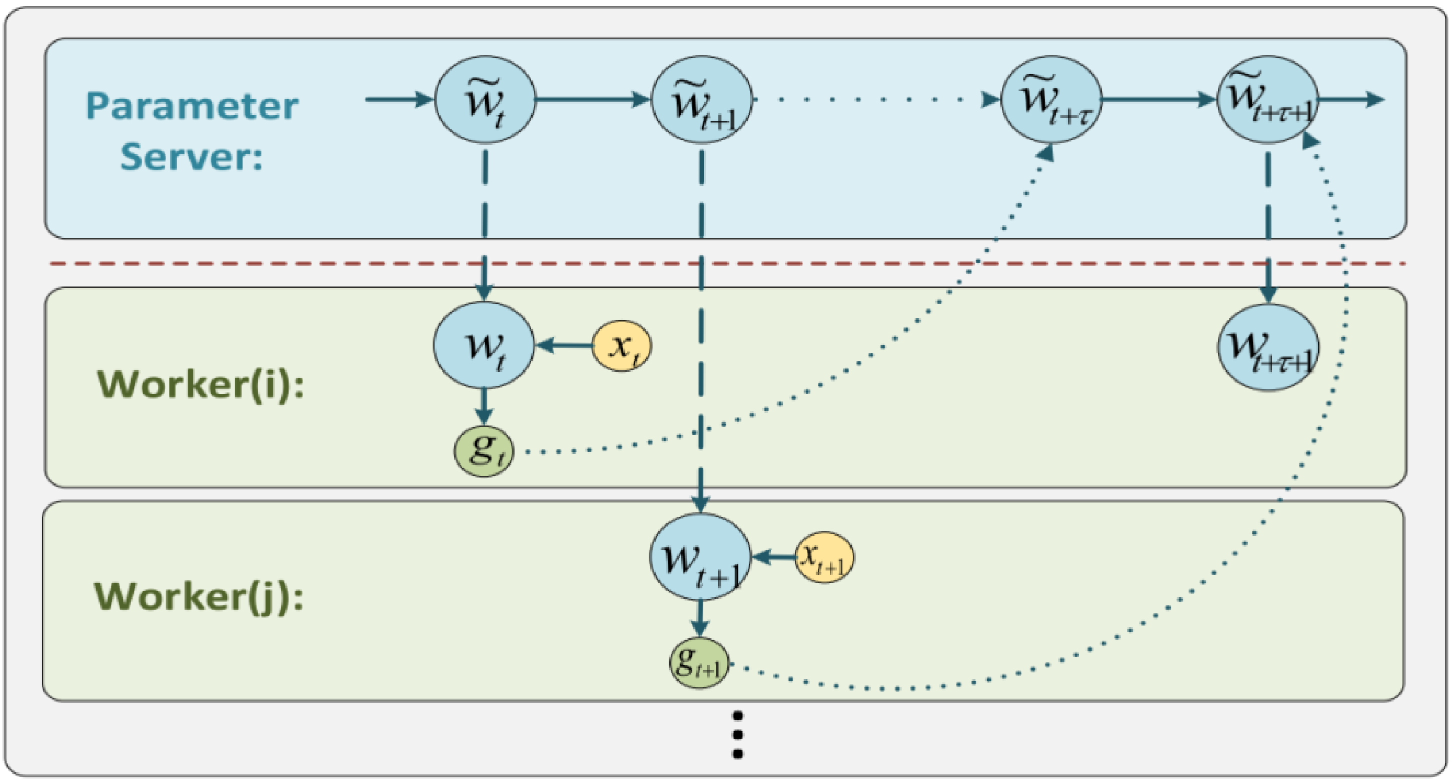


Workers calculate partial parameters

Distributed Machine Learning Frameworks

Parameter Server

- Asynchronous parallel



Distributed Machine Learning

DML Trends Overview

- For more information, please go to:
  AAAI-17 Tutorial on Distributed Machine Learning

Components	Basic Research	Advanced Research
Sequential algorithms	Convex	Non-convex, faster algorithm
Data Allocation	Gap between theory and practice	Theoretical analysis of practical data allocation
Synchronization	BSP, ASP, etc.	Handling communication delay
Aggregation	Model average	Other alternatives
Theory	Convergence	Generalization

Distributed Machine Learning
Take Home Message

- **How to "split"**
  - Data parallelism / model parallelism
  - Data / Parameters dependency
- **How to aggregate messages**
  - Parallelization mechanisms
  - Consensus between local & global parameters
  - Does algorithm converge
- **Frameworks**

Thanks