Communication-Efficient Distributed Deep Learning: A Comprehensive Survey

Abstract

- Distributed deep learning **reduce the overall training time**
- **Data communication** is a potential bottleneck
- **System-level** :design and implementation **Algorithm-level**: convergence(收敛) bounds and complexity
- taxonomy: **communication synchronization, architectures, compression, parallelism**

I. Introduction

Data communication should be well optimized（优化数据通信的开销） in distributed training of deep models.
Optimization problem(优化函数)

$$\min_{\mathbf{x}\in\mathbb{R}^N} f_s(\mathbf{x}) := \mathbb{E}_{\xi_i\sim\mathcal{D}} F(\mathbf{x};\xi_i), \quad (1)$$

A. Stochastic Gradient Descent(随机梯度下降)

Update procession:

$$G_t(\mathbf{x}_t) = \nabla F_t(\mathbf{x}_t;\xi_t) \quad (2)$$

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \gamma G_t(\mathbf{x}_t), \quad (3)$$

1. samples a mini-batch of data(取样)
2. feed-forward -->loss value(前向传播计算目标函数的loss值)
3. backward propagation ---->gradients(反向传播计算梯度)
4. updates parameters

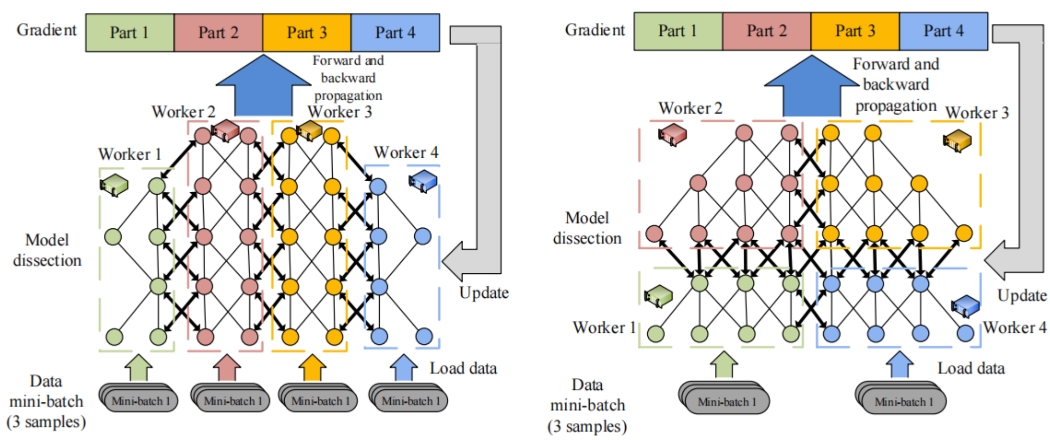
Parallelism schemes of distributed training: model parallelism(模型并行) and data parallelism(数据并行)

B. Model Parallelism

Splits model **parameters**(分割模型参数给不同的worker)

Each worker **exchange outputs**(神经元之间的高度依赖性) before next layer

> **Advantages**: training huge size model, each worker hold a subset of the model(内存需求小)
>
> **Issues:** unbalance parameter sizes(参数规模的不平衡)， computing dependency(计算依赖性) (Non-trivial and NP-complete)



(a) Type 1. (b) Type 2.

C. Data Parallelism

parameters are replicated(参数复制到每个worker中)

每个worker处理不同的mini-batches来计算local gradient updates(找到一个function使得每个worker的gradient求和取平均的值是最小的).

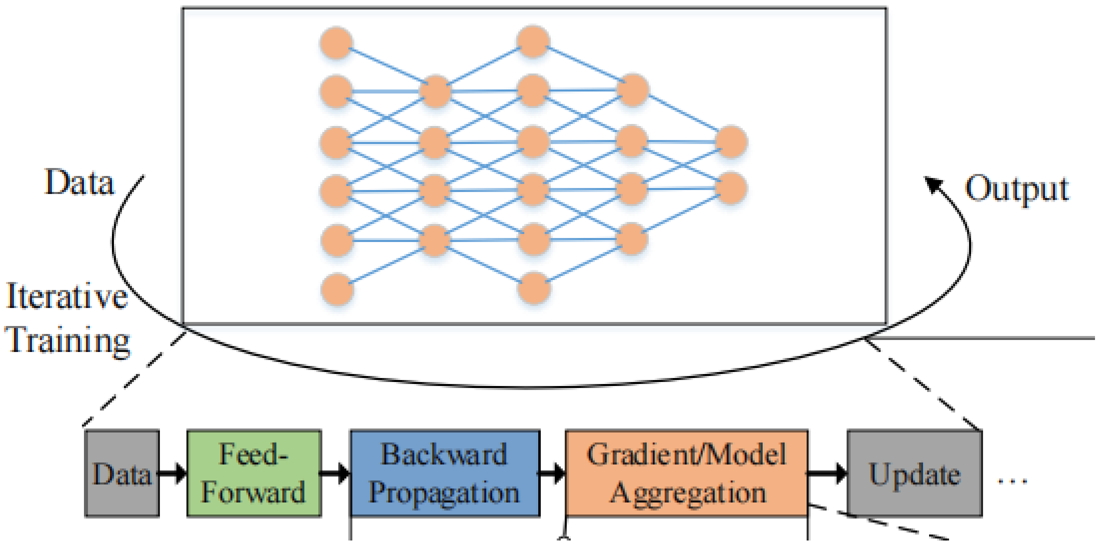
Optimization problem:

$$f^* := \min_{\mathbf{x}\in\mathbb{R}^d} \left[f(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} \underbrace{\mathbb{E}_{\xi_i\sim\mathcal{D}_i} F_i(\mathbf{x};\xi_i)}_{=:f_i(\mathbf{x})} \right] \tag{4}$$

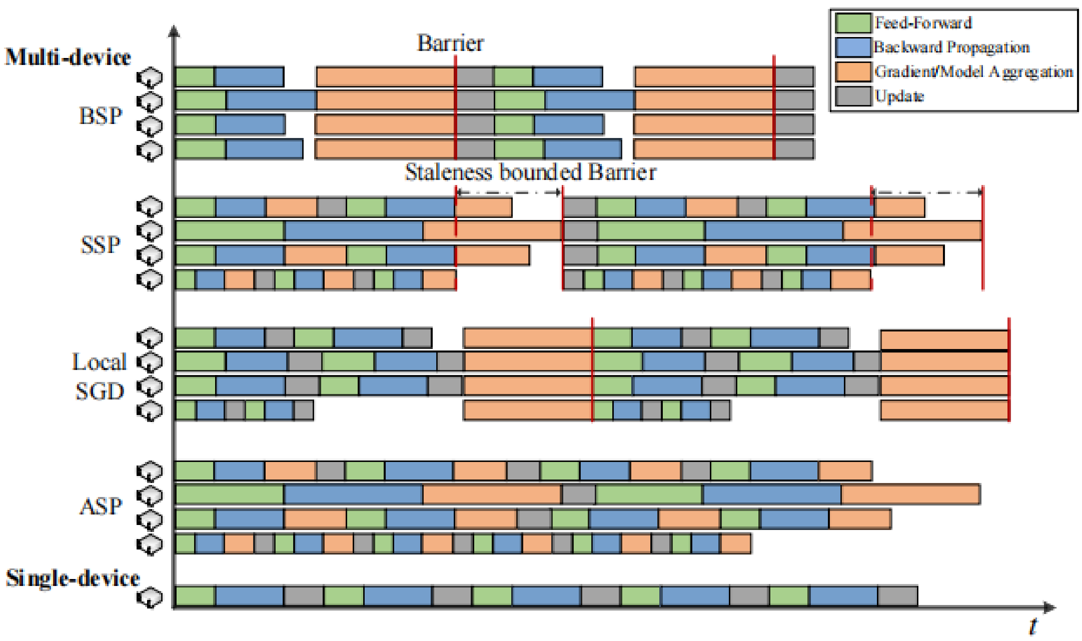
D. Distributed SGD

BSP-SGD(批量同步并行SGD)

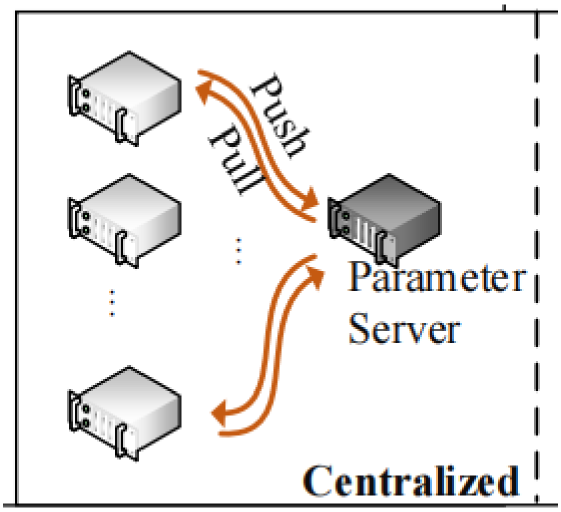
PS(参数服务器)



- worker 从PS中pulls down model，loads 不同的数据并独立计算gradients.
- update时，gradients 在PS中聚合，所有的workers在下次迭代前被barrier控制同步。



- PS对local gradients求平均来更新global model.



BSP-SGD optimization function:

$$G_{i,t}(\mathbf{x}_t) = \nabla F_{i,t}(\mathbf{x}_t; \xi_{i,t}) \tag{5}$$

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \gamma \frac{1}{n} \sum_{i=1}^{n} G_{i,t}(\mathbf{x}_t), \tag{6}$$

communication cost(通信开销):

- 通信开销和频率

解决方法：

1. 改变同步策略

asynchronous SGD (ASGD) 异步SGD

synchronous parallel SGD(SSP-SGD) 陈旧？并行SGD

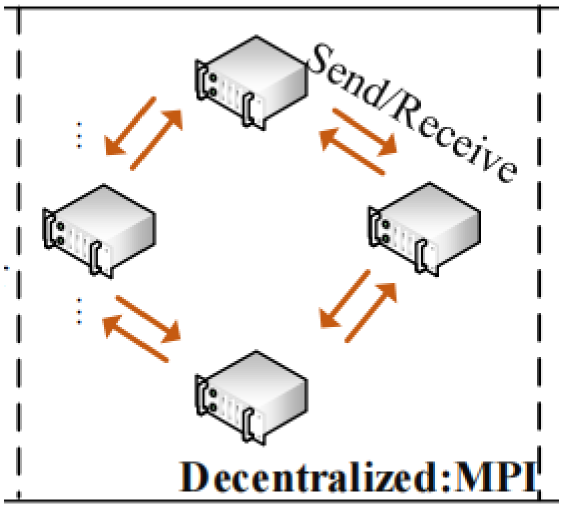
使得每个worker在每次iteration中不用等待另外的n-1个workers

2. 一次迭代中，进行多次的计算，降低通信频率

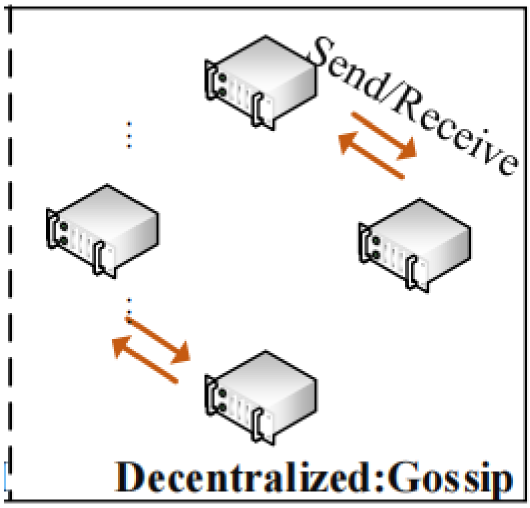
- 分布式存储器中对n个vectors的聚合算法

解决方法：

1. 在PS架构中解决congestion problem， MPI中的collective 算法是另一种分布式梯度的聚合算法设计



2. gossip architecture，workers从peers中接收models，最终收敛于同一个solution, 减缓了peers的communication



- 与模型维数相关的通信流量

Communication Compression(通信压缩): gradients/model 可以通过 quantization or sparsification (量化或稀疏)

Parallelism of Computations and Communications: 通信和计算任务的调度算法可以减少等待时间

II. Taxonomy of distributed SGD

BSP-SGD由于communication synchronization 和 aggregation会有通信的瓶颈

Communication-efficient:

1. communication synchronization
2. system architectures
3. compression techniques-->压缩通信数据
4. parallelism of communication and computing

TABLE I
TAXONOMY OF DISTRIBUTED SGD

Dimension	Method
Communication Synchronization	Synchronous
	Stale-Synchronous
	Asynchronous
	Local SGD
System Architectures	Parameter Server
	All-Reduce
	Gossip
Compression Techniques	Quantization
	Coding
	Sparsification
Parallelism of Communication and Computing	Pipelining
	Scheduling

Dimensions:

A. Communication synchronization

data parallel被分为4种frameworks: synchronous, stale-synchronous, asynchronous, local SGD.

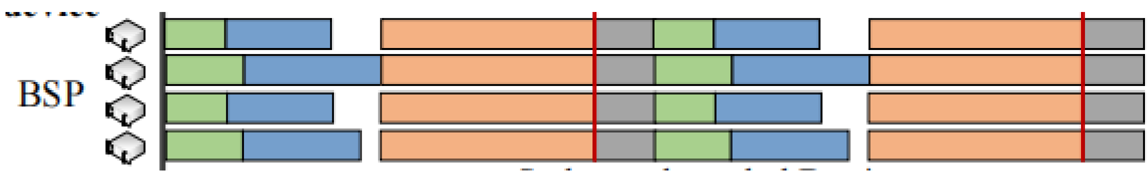
1. **synchronous**

在下一次train之前，每个worker必须等待所有workers在当前iteration种完成参数传输。

BSP就是经典的算法

最慢的worker限制了系统的throughput

data aggregation导致了高通信开销，从而限制了系统规模

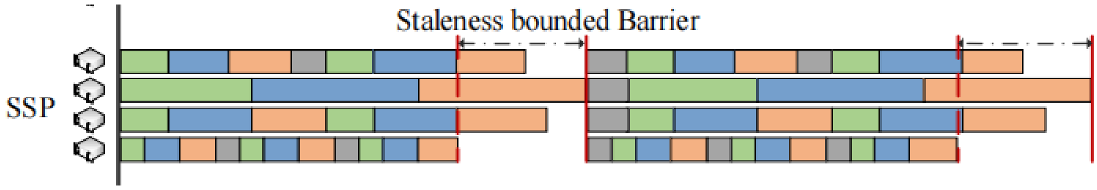


2. **stale-synchronous**

 stale-synchronous parallel(SSP)，在不失去同步的情况下缓解吞吐量问题。

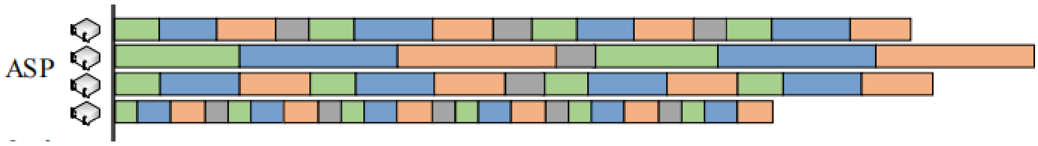
允许更快的worker做更多的updates来减少等待时间

但是最快和最慢的worker之间有界限barrier。



3. **Asynchronous**

   asynchronous parallel (ASP) 彻底消除了同步，PS收到worker计算的gradient就更新



$$\mathbf{x}_{t+1} = \mathbf{x}_t - \gamma \sum_{i=1}^{n} G_{i,t-\tau_{i,k}}(\mathbf{x}_{i,t-\tau_{k,i}}), \tag{8}$$

4. **Local SGD**

所有workers本身运行多个iteration，对local model使用Model Average的方法更新为全局模型，在本地运行大量迭代，并同步

$$\mathbf{x}_{i,t+1} = \begin{cases} \mathbf{x}_{i,t} - \gamma G_{i,t}(\mathbf{x}_{i,t}), & \text{if } t+1 \notin \mathcal{I}_T \\ \mathbf{x}_{i,t} - \gamma \frac{1}{n} \sum_{i=1}^{n} G_{i,t}(\mathbf{x}_{i,t}), & \text{if } t+1 \in \mathcal{I}_T \end{cases} \tag{9}$$

B. System Architectures

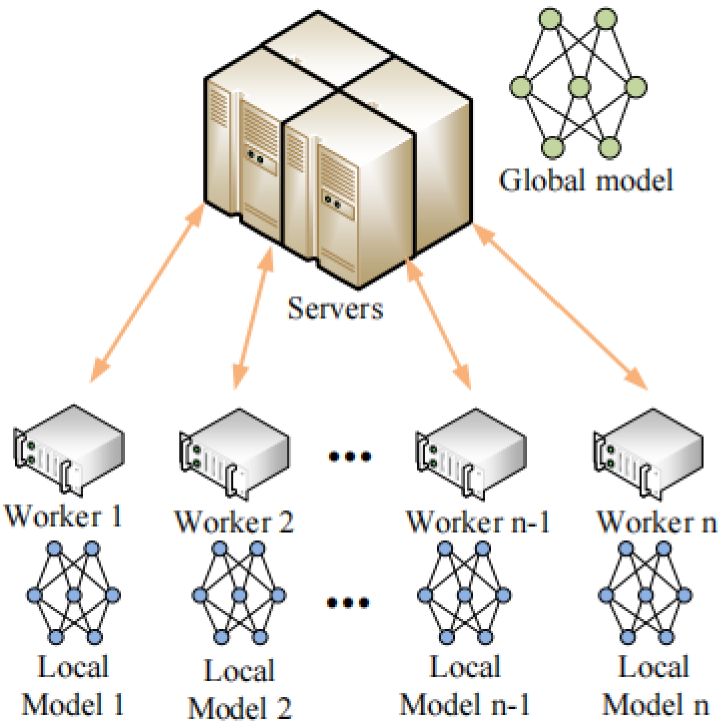
为了average model parameters/gradients

3种方法：Parameter Server (PS), All-Reduce, Gossip

1. **Parameter Server**

servers 是拓扑通信的中心，中心化的架构

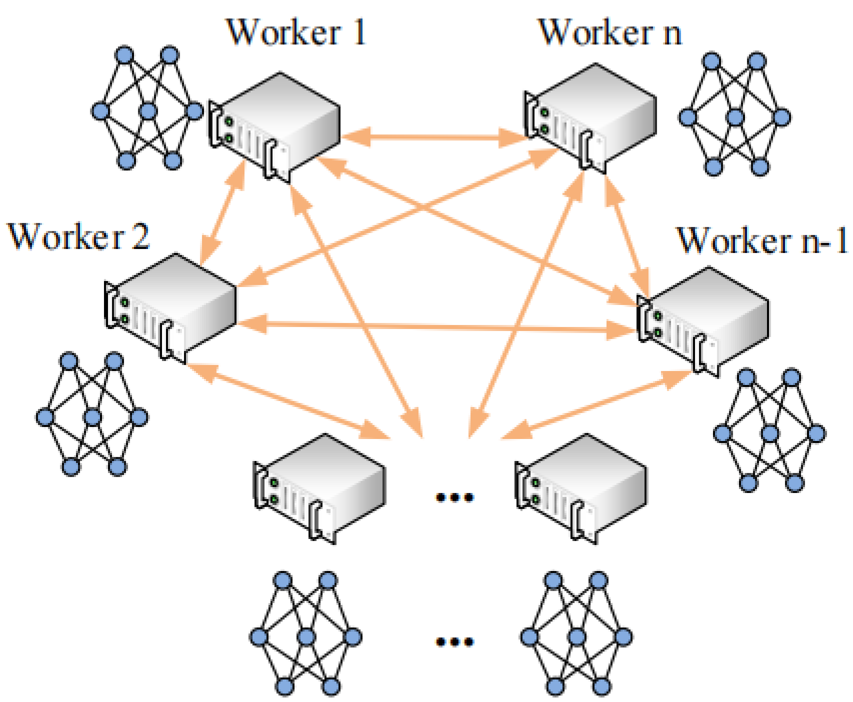
最大问题是Server的通信拥塞--->所有的worker都要与server通信



(a) PS architecture.

2. **All-Reduce**

没有中心server的情况下实现gradient的aggregation,所有的workers都要通信，同步模式适合，Asynchronous不适合



(b) All-Reduce architecture.

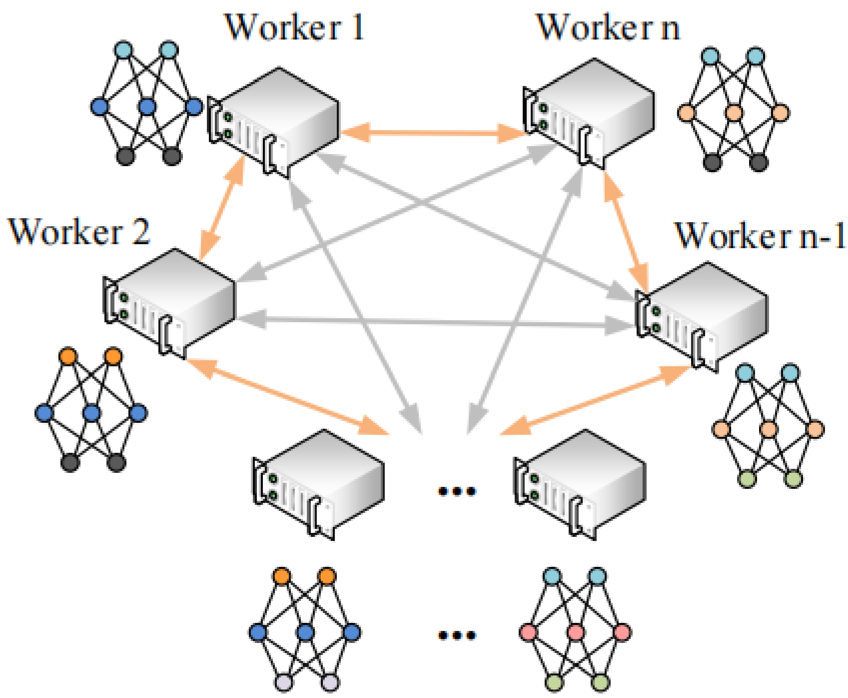
3. **Gossip**

为了解决分布式的average problem。

不仅没有PS,也没有global model(被local model代替)，只与neighbors/peers通信updates，不需要跟其他的worker通信。

一次通信后，workers的参数以及local model可能不一致，但是当算法结束时可以保证一致性。

注意：在异步模型和有ps的ssp模型中，local model也是不同的，但是PS中保留了一个global model



(c) Gossip architecture.

C. Compression Techniques

Compression : gradients 和 parameters

大多数是lossy compression（有损压缩），gradients/parameters难以完全recovered

两种压缩方法：quantization量化 、sparsification稀疏化

1、quantization

少量bits编吗，导致low precision

在Low-precision gradients 的情况下完成deep learning

2、Sparsification

减少每次iteration中传输的element的数量

只有significant的gradients是被需要的，不需要的部分则在梯度向量中被归零

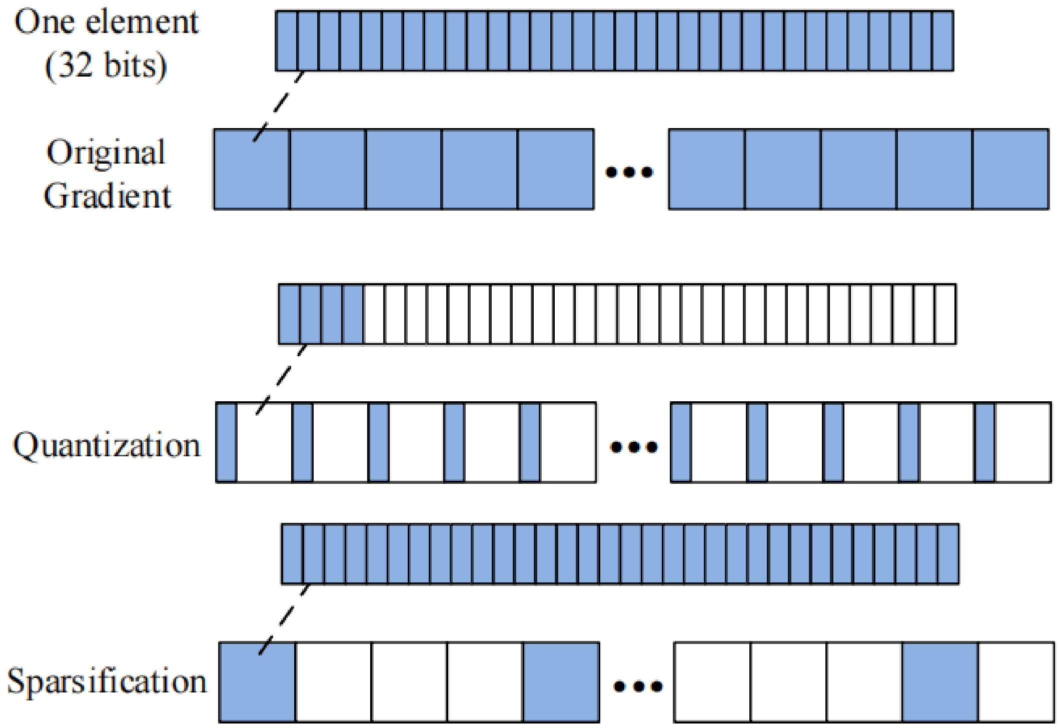


Fig. 6. Comparison of Quantization and Sparsification.

D. Parallelism of Communication and Computing

优化计算和通信的顺序，使得通信成本最小

pipelining techniques：

wait-free backward propagation(WFBP)

merged-gradient WFBP (MG-WFBP)

不同system architecture 和 synchronous scheme组合的影响：

TABLE II
INFLUENCES OF DIFFERENT COMBINATIONS OF ARCHITECTURES AND SYNCHRONIZATION SCHEMES

Architecture	Synchronization	Model consistency	Communication Frequency	Communication Congestion	Convergence
PS	BSP	high	high	high	easy
	SSP	normal	high	normal	normal
	ASP	low	high	low	difficult
	Local SGD	normal	low	high	difficult
All-Reduce	BSP	high	high	low	easy
	SSP	-	-	-	-
	ASP	-	-	-	-
	Local SGD	normal	low	low	difficult
Gossip	BSP	low	high	low	normal
	SSP	-	-	-	-
	ASP	low	high	low	difficult
	Local SGD	low	low	low	difficult

III. Synchronous/Asynchronous framework

具体介绍4种通信方案：

A. Synchronous

不论哪种system architecture，data-parallel总是在下次train之前，每个worker等待所有worker完成parameter的transmission

PS architecture: 同步barrier控制

All-Reduce：所有worker等待update到相同的global model

Decentralized architecture: 所有worker等待通信的结束，但不需要keep the same model.

B. Stale-synchronous

1、Chen et model:

a mini-batch的gradient是被workers的一个子集计算得到，这个子集是Backup Workers

PS停止等待并更新parameters，只要有n个gradients到达即可，剩下额外的e个worker迟到的gradients会被dropped

2、Ho et model:

faster workers可以提前计算和更新更多的iterations

fastest worker和lowest workers之间有一个threshold 来保证收敛性

C. Asynchronous

PS可以在几个workers的gradients的情况下完成对global model的update

提升了大规模系统的鲁棒性

异步算法会使用一些非中心化的框架

Distributed Alternating Direction Method of Multipliers (D-ADMM)

D. Local SGD

less communication --> more information loss

less frequency -->degraded performance

- the distributed momentum SGD and the PR-SGD 提升local SGD的性能，可以在train的时候提供线性的增速
- quantization method together with local SGD, 更低的通信复杂度
- 增大batch 的size ， Catalyst-like algorithm 每次train之后动态增大batch sizes
- post-local SGD,解决了large-batch training的泛化问题，将训练过程划分为两个阶段 1、mini-batch SGD, 2、local SGD

IV. Centralized/Decentralized framework

A. Parameter Servers

B. All-Reduce

C. Gossip

consensus共识，所有的workers有相同的model parameters

symmetric communication,需要自身避免deadlock

V. Quantization methods

- Distributed mean estimation,Mean Squared Error(MSE)测量quantization methods的准确性

增加量化级数的数量来减少MSE error，通信成本会增加

减少通信成本的两种方法：

Stochastic Rotated Quantization，clients和server生成 a global random rotation matrix，找到an orthogonal matrix R(正交矩阵)，实现low MSE

Variable Length Coding,使用Huffman Coding，对应于每个量化值出现的次数

- 1-bit SGD减少传输的数据量

每个element的gradient减少到1bit

$$G_{i,t}^{quant}(\mathbf{x}_t) = Quant(G_{i,t}(\mathbf{x}_t) + \delta_{i,t}(\mathbf{x}_t)) \tag{10}$$

$$\delta_{i,t}(\mathbf{x}_t) = G_{i,t}(\mathbf{x}_t) - Quant^{-1}(G_{i,t}^{quant}(\mathbf{x}_t)) \tag{11}$$

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \gamma \frac{1}{n} \sum_{i=1}^{n} G_{i,t}^{quant}(\mathbf{x}_t) \tag{12}$$

- a fixed threshold,encoded the gradient element
- adaptive quantization methods
- Quantized SGD(QSGD), a probabilistic approach to quantify a vector(量化向量的概率方法)

quantized gradient 是原向量的无偏估计

- TernGrad使用ternary gradients(三元梯度)加速distributed deep learning.

$$\tilde{Q}_t(G(\mathbf{x}_t)) = ternarize(G(\mathbf{x}_t)) = s_t \cdot sign(G(\mathbf{x}_t)) \circ \mathbf{b}_t \tag{15}$$

where $s_t := max(abs(G(\mathbf{x}_t)))$, and $\circ$ represents the Hadamard product. The $sign(\cdot)$ is the same as $sgn(\cdot)$ in QSGD. Each element of $\mathbf{b}_t$ follows the distribution

$$\begin{aligned} P(b_{t,j} = 1 | G_t(\mathbf{x}_t)) &= |G_{t,j}(\mathbf{x}_t)/s_t| \\ P(b_{t,j} = 0 | G_t(\mathbf{x}_t)) &= 1 - |G_{t,j}(\mathbf{x}_t)/s_t| \end{aligned} \tag{16}$$

- Parameter Localization,每个worker在本地存储parameters的副本

，只需要从server pulling quantized gradients。Scalar Sharing,选择最大Scalar的一个，并在所有worker中共享。

layer-wise ternarizing （分层标量）and Gradient Clipping（渐变剪切）

- Sign-SGD,把gradient量化为一个二进制数，每个worker将自己的梯度向量的sign交换到server后，总体更新由多数vote决定
- Atomic Sparsification (ATOMO).（原子稀疏化）目标是最小化在原子基础上稀疏的稀疏梯度的方差
- DIANA，分割整个gradient vector为几个sub-vectors，然后分别量化几个sub-vectors
- LAQ,先量化当前gradient和之前的gradient之间的差距，如果innovation不够大，则跳过gradient communication
- Natural Compression
- AsyLPG，low-precision,异步架构，

VI. Sparsification methods

quantization methods仅仅可以在single-precision 浮点型实现最大32倍的compression

但是如果减少传输elements的数量，compression rate将会进一步提升，这就是稀疏化方法,sparsification methods，仅有一部分的elements会被选中并transmit

memory space约束，test-time computation的约束

Sparsification communication methods:

（1）coordinate descent（坐标下降）

（2）random sparsification

(3) deterministic sparsification

(4) proximal methods

A. Random Sparsification

随机选择一些entries来通信并更新，

random-*k*--->Random Mask--->Subsampling (unbiased!!!)

Randomly drop out

$$\begin{aligned}
\text{original vector } \mathbf{g} &= [g_1, g_2, \cdots, g_N], \\
\text{probability vector } \mathbf{p} &= [p_1, p_2, \cdots, p_N], \\
\text{selection vector } \mathbf{Z} &= [Z_1, Z_2, \cdots, Z_N], \\
\text{sparsified vector } \mathbf{Q}_{spar}(\mathbf{g}) &= \left[Z_1 \frac{g_1}{p_1}, Z_2 \frac{g_2}{p_2}, \cdots, Z_N \frac{g_N}{p_N}\right]
\end{aligned} \tag{18}$$

B. Deterministic Sparsification

在全连接deep model中的weights可以接近于zero

稀疏权值导致零梯度，零梯度就不需要communication

sparsify gradients:

(1)Fixed Threshold

gradient绝对值小于一个之前设置的threshold，则这个gradient element将会被discard

缺点：难以找到一个合适的threshold，在Error Feedback下，会导致大量的gradient被传送

(2)Adaptive Threshold

Top-k sparsification algorithms

- 设置一个固定的proportion(比例)，可以保证compression ratio。此方法的前提是对所有的gradient vector排序.
- 只选择一个绝对值的threshold
- AdaComp
- SBC，Sparse Binary Compression，结合了稀疏化和量化方法，cutoff低绝对值的元素，分别求正的均值和负数均值，正数绝对值大，则丢弃所有负数元素，并设置所有的正数为正数的均值，反之亦然
- Sparse Ternary Compression (STC)，结合了Top-k稀疏化和Ternary量化的方法，比SBC更适合联邦学习。Top-k稀疏化，模型更新的大小取决于workers的数量
- gTop-k，在聚合完所有梯度后，会再次稀疏化global gradient vector

C. Coordinate Descent

将所有变量分割为多个blocks，然后更新其中一个block，同时修复其他的blocks。

一次iteration中，所有blocks都会被依次update

Gradient-free methods用于解决gradient vanishing

D. Proximal Methods

发送variance足够小的元素，以获得通信的稀疏性

Count Sketch，将梯度vector G，压缩为S(G),近似于L2范式，压缩后发送到server，server恢复梯度和最大的d个坐标，然后更新

VII. Scheduling of communication and computing

layer-wise structure(分层结构)，可以使得通信和计算并行进行

通信和计算任务表示为有向无环图DAG

wait-free backward propagation (WFBP)(无等待的反向传播)

分层的梯度稀疏化中，可以pipeline三种任务（梯度计算、梯度稀疏化、梯度通信）

在不同任务的等待时间之间，可以通过调度顺序来减少等待时间

VIII. Convergence Analysis

A. PS/All-Reduce Architecture

1) BSP

PS和All-Reduce架构有相同的迭代方程

2) SSP/Asynchronous

3)Local SGD

B. Gossip Architecture

all of the workers own an individual model in the gossip architecture(保证一致性是前提)

1) BSP

2) Asynchronous

IX. Auxiliary Technologies

A. Error Accumulation

1bit-SGD,误差累积导致所有梯度累加误差到模型中，保证准确性

EF-SIGNSGD,error被本地存储并添加到下一步，同样属于error accumulation

$$\begin{aligned} &\text{gradient compression} && C_{i,t} = Sparse(v_{i,t-1} + \nabla_{i,t}), \\ &\text{error accumulation} && v_{i,t} = \nabla_{i,t} - C_{i,t} \\ &\text{update weight} && x_{t+1} = x_t - \gamma \tfrac{1}{n} \textstyle\sum_{i=1}^{n} C_{i,t} \end{aligned}$$

ECQ-SGD (Error Compensated Quantized SGD，不仅累加当前iteration的compression error，还要考虑previous error

B. Momentum Correction

Momentum SGD的error accumulation

$$\begin{aligned} &\text{momentum accumulation} && u_{i,t} = m u_{i,t-1} + \nabla_{i,t}, \\ &\text{error accumulation} && v_{i,t} = v_{i,t-1} + u_{i,t}, \\ &\text{update weight} && x_{t+1} = x_t - \gamma \textstyle\sum_{i=1}^{n} sparse(v_{i,t}) \end{aligned}$$

图中，m是动量系数

C. Local Gradient Clipping

Gradient clipping可以避免梯度爆炸现象

通过adjust clipping 的threshold，local gradient clipping可以恢复聚合模型的原始方差

D. Warm-up Training

将training过程划分为两个阶段：

warm-up period-->不激进的学习率和低的稀疏性

normal-training period--->高的稀疏度和低的学习率

XI. Conclusion And Future Direction

communication-efficient 分布式深度学习分类 4个维度：

1）synchronous scheme

2）system architecture

3）compression techniques

4）parallelism of communication and computing

一些挑战和未来方向：

1）更多communication方式的组合

2）更高的压缩比、压缩层级

能否在更高的压缩比情况下，不损失训练性能

在保持压缩精度的同时，提高压缩比

3）自适应压缩

平衡压缩比和收敛速度

4）提高算法在异构设备中的容错性