# PASSING:分布式机器学习的混合参数同步策略

余晓杉,顾华玺*,周肇星,王佳昆
(西安电子科技大学通信工程学院,陕西西安 710000)

**摘 要:** 随着机器学习模型的参数量与训练数据集爆炸式增长,单一计算节点已无法满足人工智能(Artificial Intelligence,AI)大模型的算力需求,分布式机器学习系统成为支持模型训练的主要平台,该系统通过数万设备的并行训练缩短机器学习的训练时间. 其中数据并行是一种常用的分布式训练并行框架,该框架将训练数据划分至不同的计算节点,通过节点间周期性参数同步实现训练任务的协同,由于计算节点在每轮迭代前需要传输大量数据以完成参数同步,通信成为影响计算效率的关键因素. 经典参数同步策略存在通信次数较多或接收端链路拥塞的问题,基于网内聚合的参数同步策略则存在交换机计算、存储能力有限、服务器输出端口拥塞的问题,对此本文提出一种混合参数同步策略 PASSING(hybrid Parameter Synchronization Strategy with In-host and In-network Aggregation),该策略首先在服务器内或机架内预先进行模型参数的本地同步,随后利用可编程交换机完成全局的参数同步,这种方式既保证了机内小规模计算节点间的高效通信,也减轻了交换机侧的计算和通信负载. 本文使用多 GPU(Graphics Processing Unit)服务器和可编程交换机搭建了实验平台,并部署了所提出的混合同步策略,实验结果表明 PASSING 相较于传统的参数服务器算法最多提升了 65.25% 的训练性能,有效加速了分布式训练的速度.

**关键词:** 分布式训练;数据并行;参数同步;网内聚合;混合同步策略
**基金项目:** 国家重点研发计划(No.2018YFE0202800)
**中图分类号:** TP393 **文献标识码:** A **文章编号:** 0372-2112(2025)08-2636-13
**电子学报 URL:**http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20250207

# PASSING: A Hybrid Parameter Synchronization Strategy in Distributed Machine Learning

YU Xiao-shan, GU Hua-xi*, ZHOU Zhao-xing, WANG Jia-kun
(*School of Telecommunications Engineering*, *Xidian University*, *Xi'an*, *Shannxi* 710000, *China*)

**Abstract:** With the explosive growth in the number of parameters of machine learning models and the scale of training datasets, a single computing node can no longer meet the computational demands of large artificial intelligence (AI) models. Distributed machine learning systems have become the primary platform for supporting AI model training. The training time can be reduced by implementing parallel training across tens of thousands of computing nodes. In particular, data parallelism is a widely used parallel training framework in distributed training. It splits the training dataset across many computing nodes and then trains the model collaboratively through periodic parameter synchronization among those nodes. Since computing nodes need to transmit a large amount of data to complete the parameter synchronization before each round of iteration, communication becomes the key factor that affects computational efficiency. Traditional parameter synchronization strategies suffer from the problem of excessive communication rounds or congestion at the receiver's link. In contrast, parameter synchronization strategies based on in-network aggregation face issues such as limited computing and storage capabilities of the switches, and congestion at server output ports. To this end, a hybrid parameter synchronization strategy termed PASSING (hybrid Parameter Synchronization Strategy with In-host and In-network Aggregation) is proposed. It implements a local pre-aggregation of the model parameters within the host prior to transferring the data to programmable switches. Subsequently, the local aggregation parameters are sent to the programmable switches to implement the global parameter synchronization. This approach not only ensures efficient communication between the small-scale computing nodes with the host but also reduces the computational and communication load on the switch side. We built a test-

收稿日期:2025-03-20;录用日期:2025-08-18;责任编辑:覃怀银
*通讯作者:顾华玺

bed using the multi-GPU (Graphics Processing Unit) servers and programmable switches and deployed PASSING in this testbed. The experimental results demonstrate that PASSING, when compared to traditional parameter synchronization strategies, enhances training performance by up to 65.25%, thus effectively accelerating the speed of distributed training.

Key words: distributed training; data parallelism; parameter synchronization; in-network aggregation; hybrid parameter synchronization strategy

Foundation Item(s): National Key Research and Development Program of China (No.2018YFE0202800)

## 1 引言

作为人工智能领域中得到广泛应用的重要分支,机器学习代表了一种以数据样本驱动的智能分析方法,旨在对复杂数据样本中的复杂关联深入分析与建模.机器学习的学习过程需要经过多轮迭代积累,以此不断积累经验,进而挖掘出数据样本中的规律性信息与模式,并最终能够对未知数据进行预测和决策.凭借着自主化的迭代能力与智能化的决策能力,机器学习深刻促进了计算机视觉、自然语言处理、推荐系统等诸多学术领域的发展,并已在诸如自动驾驶、语音助手、医疗诊断等与人类生活切实相关的领域得到广泛应用[1~4].

深度学习是一种基于深度神经网络架构(Deep Neural Network,DNN)的机器学习方法,其通过模拟人脑构建多层非线性变换,从而获得特征学习、数据解释能力的显著提升.深度学习能够在大规模数据中获得卓越的性能,是机器学习迅速发展与普及的重要推动技术,其中卷积神经网络(Convolution Neural Network,CNN)[5~7]、循环神经网络(Recurrent Neural Network,RNN)[8,9]等神经网络模型已经在特定领域取得了出色的表现.例如基于循环神经网络可使心脏房颤的检测准确率达到97.72%、灵敏度达到93.09%[10],又如谷歌提出的EfficientNet-B7利用卷积神经网络在ImageNet数据集[11]上实现了高达97.1%的Top-5图像分类准确率[12].

随着训练数据集的数据量与复杂性的持续增长,人工智能的计算复杂度呈现显著的指数级增长.以OpenAI公司为例,2019年发布的自然语言模型——GPT2具有15亿的参数量,需要约40 GB的预训练数据量,而2020年升级发布的GPT3模型的参数量与预训练数据量分别激增至1 750亿与45 TB[13].然而业界普遍认为摩尔定律逐渐放缓甚至即将失效,单个计算节点已无法满足如此大规模机器学习任务对计算和存储所提出的苛刻需求,例如OpenAI公司为训练GPT4模型需使用数万张A100显卡[14],因此分布式训练是使能大规模数据集和复杂模型训练的关键技术,并成为业界广泛采用的解决方案.学术界与工业界为此针对模型并行、数据并行两类常用并行方式展开了广泛的研究与实践.

在模型并行的分布式训练加速方面,该策略首先于2012年在文献[15]中被提出,模型并行通过横向按层划分、纵向跨层划分方法等方式将模型切分为多个子模型,并将子模型分配到不同的计算节点上进行计算,模型大小因此能突破单个计算节点内存空间大小的限制.然而在每次前向传播和反向传播中均需要额外的计算节点间通信,无法获得理想的计算效率提升.研究人员因此聚焦于采用张量并行[16,17]、流水线并行[18,19]、专家并行[20,21]或结合数据并行的混合并行[22~24]策略.例如谷歌于2019年发表的GPipe[22]将神经网络模型划分至独立的计算节点中,并将小批量训练样本进一步划分为更小的微批量尺寸,同时利用流水线的方案使得计算节点的计算空闲时间在一定条件下可被忽略,模型并行时的设备利用率得到了有效的提升.

在数据并行的分布式训练加速方面,各计算节点上均放置有完整的训练模型,但每个计算节点仅处理数据样本中的一个子集,并在每轮训练完成后进行模型参数的同步[25].同步更新是经典的参数同步策略,每一轮迭代的参数更新需要在所有计算节点完成训练任务后才能进行.然而在实际的工业部署中,由于节点计算能力的异构性与偶然的设备异常,训练速度慢的节点将成为制约每一轮迭代训练的瓶颈,导致集群中的大量计算节点处于停滞等待状态,训练的任务完成时间随之被极大延长.与同步更新策略不同,异步更新策略中的各计算节点在完成每轮迭代训练后立即更新参数服务器中的模型参数,有效避免了停滞等待的发生,然而计算速度较快的节点可能会从参数服务器中获取陈旧参数,导致无效的重复训练,模型的收敛性受到极大的影响[26].

分布式训练将大规模数据集或复杂模型拆分至集群的各计算节点进行并行计算,并最终将结果汇聚.因此分布式训练涉及多个计算节点之间频繁的数据交换与通信,通信时延随着集群规模的增长而爆炸性增长,计算节点往往处于等待数据到达的计算空转状态,通信开销随之取代计算成为分布式训练的关键瓶颈性问题.特别是在采用数据并行的分布式训练中,加速比难以与集群规模呈线性关系,集群中大量的计算资源无法得到充分利用.例如在基于100 Gbps InfiniBand网络

互连的8台服务器、32块RTX 2080Ti集群中，训练Bert模型的速率仅比单块RTX 2080Ti训练速度提高约8倍[27,28].

数据并行与模型并行在分布式训练中的作用存在较大的差异. 数据并行以训练的加速为目标，将数据样本划分至不同的计算节点，而模型并行侧重于解决单计算节点无法承载大模型的问题，在训练加速方面的作用较为局限. 因此学术界在数据并行方面展开了诸多探索，也成为工业界中常用的并行模式，本文也将针对数据并行策略下的通信瓶颈问题展开研究，并提供一种结合传统同步策略和网内计算的混合参数同步策略，具体而言，该策略先在各服务器内部执行部分参数的聚合，使得每块计算卡获得聚合数据的一部分，随后各服务器把机内聚合数据推送给参数服务器，参数服务器完成聚合后广播给各计算服务器. 通过参数服务器同步策略与环规约同步策略的结合，计算服务器与参数服务器间的通信量能够得到显著减少，缓解了参数服务器的通信压力. 在此基础上，所提出策略进一步支持使用可编程交换设备在网内执行参数聚合与分发，通过将聚合功能卸载至网络，能够避免参数服务器处的通信瓶颈问题，缩短训练任务的完成时间，提升分布式训练系统的加速比. 经过实验验证，PASSING(hybrid Parameter Synchronization Strategy with In-host and In-network Aggregation)策略较参数服务器算法提升了高达65.25%的训练性能.

## 2 相关工作

对于数据并行的分布式训练框架，其通信目的在于完成模型参数的同步，即各计算节点根据本地数据样本完成一轮训练后，需要将模型参数进行聚合，随后各计算节点获取聚合后的参数用以更新本地模型. 传统同步策略由服务器完成参数聚合，随着可编程交换机的引入，基于网内聚合的同步策略在近年来得到广泛关注.

### 2.1 传统参数同步策略

传统参数同步策略主要包括环形同步策略和基于参数服务器的同步策略. 在环形同步策略中，每个计算节点均参与模型参数的汇总任务，而在基于参数服务器的同步策略中，模型参数的汇总由特定的一个或多个计算节点完成.

环形同步策略使计算节点在逻辑上以环形拓扑的形式依次相连，数据沿着约定方向在闭环中传递，因此确保了每个节点都能公平地与邻居节点进行高速数据传输，同时具有极高的分布并行性，适合用于并行任务中的数据同步操作.

百度率先提出在分布式训练领域引入环形同步算法Ring Allreduce[29]. 该算法要求全部$N$个计算节点形成一个环形拓扑，并在初始时将每个计算节点的数据切分为$N$份数据块. 算法首先进行$N-1$次Reduce-Scatter迭代操作，每次迭代中的计算节点将依顺时针的顺序向下一个邻居节点发送一份数据块，从上一个邻居节点接收一份数据块并聚合该数据块，因此每个计算节点在经过$N-1$次迭代将获得最终聚合数据的$1/N$块. 算法最后通过$N-1$次All-Gather操作交换这些数据块，使得所有计算节点获得完整的最终聚合数据. 假设每份数据块的大小为$K/N$，则每个计算节点中在经过$2(N-1)$轮通信后共需发送$2(N-1)K/N$大小的数据量，近似固定大小的通信开销与并行化的通信方式有效避免了中心化网络中的通信瓶颈现象，并已被证明是带宽开销最优的同步算法.

Ring Allreduce由于单环拓扑而不适用于大规模训练场景，同步时延将随着拓扑直径的增加而显著延长. 为了降低同步时延并提高同步并行性，腾讯进一步提出了分层环形同步算法Hierarchical Allreduce[30]. 该算法将$N$个计算节点划分为$D$个组，每组中的$N/D$个计算节点以环形拓扑进行互连，同时每组随机挑选一个节点作为根计算节点，根计算节点彼此互连构建一个高层次的环形拓扑. Ring Allreduce算法将依次在组内、组间进行，最后由每组的根计算节点将聚合数据传输到组内的其他各计算节点. 该同步算法能够通过降低时延与拓扑直径有效提升分布式训练的效率，分组分层的同步方式也适合进行实际的集群部署与互连布线.

受Hierarchical Allreduce算法分层思想的启发，索尼、谷歌、IBM等公司的研究人员分别在各类环形拓扑中实施了相应的分层同步策略. 索尼基于节点度更高的2D-Torus拓扑提出了同步算法——2D-Torus Allreduce[31]，该算法依次在拓扑的水平方向、垂直方向进行Reduce-Scatter操作与All-Reduce操作，最后在拓扑的垂直方向进行All-Gather操作，因此相较于Hierarchical Allreduce减少了通信数据量，提升了同步过程的维度并行性. 谷歌基于2D-Mesh拓扑及自研的TPU双路网卡提出了2D-Mesh AllReduce同步算法[32]，该算法将各计算节点中的数据等分为两部分，借助于双端口网卡的并行数据通信能力，计算节点能够同时在2D-Mesh拓扑的水平方向与垂直方向上分别对两部分数据进行Ring Allreduce操作，因此通过两次Ring Allreduce操作即可完成两部分数据在集群中的同步，并有效解决2D-Torus Allreduce算法中链路利用率不高的问题. IBM基于实际部署中各级交换机带宽差异的事实提出Blue Connect同步算法[33]，该算法根据节点间的带宽情况将Ring Allreduce算法分解至三个维度，大幅提升了同步算法的并行度.

参数服务器同步策略基于参数服务器架构(Parameter Server,PS)与MapReduce计算框架设计,该方法将集群中的节点区分为参数服务节点与计算工作节点两类.参数服务节点采用Hadoop分布式文件系统,以分布式存储的方式存储、维护全局共享模型参数.计算工作节点则基于Map操作分配的部分数据样本进行训练,训练所得参数将推送至参数服务节点进行Reduce操作.得益于分布式文件系统驱动的参数服务器架构具有优异的一致性、容错性和灵活的可扩展性.该架构能够轻易部署多样化的参数更新机制和同步策略,因此研究人员能专注于分布式训练算法的设计.

DistBelief[15]是早期采用参数服务器架构的大规模分布式训练方案,该方案采用了数据并行结合模型并行的方式.DistBelief将计算节点划分为多个节点组,节点组之间以数据并行的方式通过参数服务器进行异步训练,而每个节点组内对模型副本再进行拆分,以模型并行的方式完成训练.DistBelief较好地综合了数据并行和模型并行的优点,使得方案能适应于各类模型大小与集群的规模,但也存在因节点性能差异而造成的同步等待问题.为了协调计算节点性能的异构性,香港中文大学团队在参数服务器架构的基础上设计了主从体系结构FlexPS[34],主节点中的调度器跟踪各节点中的资源利用与负载情况,并根据数据局部性、任务并行度、资源占用等因素将训练任务不同阶段调度分配至合适的节点,实现了集群计算资源的合理高效利用,显著提升了分布式训练的效率.

参数服务器同步策略将模型参数维护与训练计算进行功能解耦,有效利用了异构的CPU(Central Processing Unit)/GPU(Graphics Processing Unit)资源,但在CPU或参数服务器数量不足时易形成多到一的流量热点,进而导致网络拥塞,计算节点的带宽资源无法得到充分利用.字节跳动为解决该问题推出了分布式训练框架BytePS[35],该框架通过利用CPU的计算能力减少瓶颈链路中的数据传输量,并充分利用GPU之间的带宽资源.BytePS利用PCIe或NVLink链路首先在GPU之间进行Reduce-Scatter或Reduce操作,聚合后的参数再由CPU通过网卡发送至参数服务器.BytePS通过显著减少数据传输量,有效地避免了PCIe链路及参数服务器处的链路竞争与流量热点问题.

### 2.2　基于网内聚合的参数同步策略

网内计算是近数据处理领域中的热门研究方向,它通过传输路径上的数据处理,尽可能减少了数据的传输量,是后摩尔时代的关键技术之一[36,37].为进一步缓解参数服务器的流量热点情况,减少对参数服务器数量的需求,微软率先提出了融合网内计算思想的SwitchML方案[38],该方案将参数聚合卸载至具有超高吞吐量的可编程交换机.计算节点负责数据传输的可靠性管理,并将浮点数形式的参数转换为交换机能处理的整数形式.可编程交换机则借助对数据包的线速处理能力进行高速的聚合操作,并将结果推送至参数服务器及各计算节点.为解决可编程交换机内存空间的限制,SwitchML采用基于存储资源池的流式聚合技术,每次仅对有限数量的参数元素进行聚合处理.因此SwitchML通过聚合操作的卸载减少了分布式训练过程中的数据传输通信量,也随之缓解了参数服务器处的通信负荷.NetReduce[39]是一个兼容远程直接内存访问(Remote Direct Memory Access,RDMA)协议的网内聚合解决方案,与SwitchML相比,NetReduce使用NetFPGA设计硬件系统,因此提供了更高的灵活性和更大的内存,另外NetReduce通过RoCEv2协议实现了更好的拥塞控制和可靠传输.ATP(in-network Aggregation for multi-Tenant learning Processing)[40]同样使用可编程交换机实现网内聚合,但不同于SwichML将聚合任务全部卸载到交换机,ATP保留了传统的参数服务器同步策略,用于处理可编程交换机内存不足时的聚合任务.除上述网内聚合方案外,研究人员还提出了专门针对稀疏卷积神经网络模型的网内聚合方案Libra[41],考虑负载均衡的网内聚合方案GRID(Gradient Routing with In-network aggregation for Distributed training)[42],以及基于智能网卡和新型加速器的网内聚合方案FLARE(FLexible in-network AllREduce)[43]、SHARP(Scalable Hierarchical Aggregation and Reduction Protocol)等.

## 3　参数同步策略的通信瓶颈

### 3.1　通信问题分析

随着模型的增大,计算节点之间需要同步的参数量随之增加,每轮分布式训练完成后,集群内所有计算节点同时将本地训练的参数注入网络,此时网络需要提供高带宽、低时延的传输路径以确保通信任务快速完成,否则计算节点将陷入空闲等待状态.受到物理拓扑限制,现有参数同步算法在部署至实际系统时,可能存在链路拥塞、负载分布不均、通信时延较长等问题.

参数服务器同步策略在通信过程中易产生Incast问题,如图1所示,当所有计算节点同时通过网络向参数服务器发送模型参数时,短时激增的流量将导致交换机的瓶颈端口处出现拥塞,致使缓存队列处发生数据包的溢出,大量的数据包丢失触发频繁的数据包超时重传,进而导致拥塞点更加劣化、网络吞吐量显著下降、通信时延显著增加,严重制约了分布式训练集群的性能,并且随着集群规模的增加,更多的Incast流量会导致更严重的网络拥塞与性能劣化.尽管使用图2所示的多参数服务器架构能够减少发往每台参数服务器

的流量，从而缓解交换机输出端口位置的拥塞，但该方案将在一定程度上增加设备开销和通信复杂度[44,45].

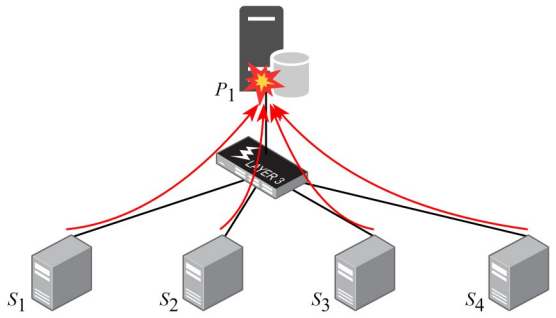


图1 参数服务器同步策略中的Incast现象

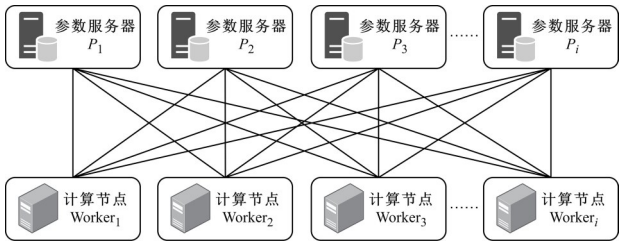


图2 多参数服务器架构示意图

基于网内聚合的参数同步策略可有效降低网络传输的数据总量，避免交换机输出端口的负载拥塞. 但与参数服务器相比，可编程交换机的计算资源和存储容量极为有限，容易成为参数同步过程的瓶颈节点. 另外在如图3所示的传统多GPU卡的服务器架构中，服务器内通常会配置多张GPU卡和一张高速网卡. 在该结构中，当多张GPU卡同时将参数发往可编程交换机时，会首先在服务器的网卡位置发生异常严重的拥塞.

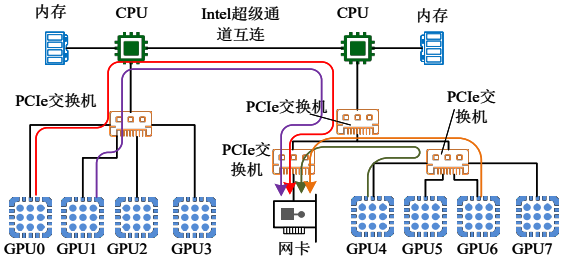


图3 传统多GPU卡服务器架构

环形同步策略将参数服务器同步策略中的“多对一”或“多对多”通信模式修改为“一对一”通信，因此不容易造成网络拥塞，但环形同步策略需要的通信次数较多. 对于包含$N$个计算节点的集群，若采用环形同步策略，则完成一次参数同步需要$2(N-1)$次通信. 随着集群规模的扩大，环形同步策略的通信时延将成为通信的瓶颈.

### 3.2 实际系统下的通信瓶颈观察

上一小节对现有参数同步策略的通信问题进行了分析，本小节将通过系统实测，进一步论证通信对分布式机器学习系统计算效率的影响. 我们使用加速比来衡量分布式机器学习系统的计算效率. 加速比是指同一个任务在单机系统和分布式并行系统中完成任务所需时间的比值. 我们在多台服务器搭建的分布式集群中测试了各规模下的加速比. 在该集群中，各服务器配备有多块NVIDIA Tesla K80 12 GB显卡、运行TensorFlow 1.15.0、Horovod 0.20.3框架与NCCL 2.6.4通信库. 在网络互连方面，各服务器通过搭载的Mellanox ConnectX-4网卡以10 Gbps速率以星形拓扑的方式接入交换机. 实验在环形同步策略与基于参数服务器的同步策略下分别进行，选用通信密集型模型VGG(Visual Geometry Group)19、计算密集型模型ResNet50进行，以Cifar-10作为训练数据集训练100轮次.

VGG19模型与ResNet50模型计算与通信的时间开销如图4与图5所示. 无论是通信密集型或是计算密集型模型，分布式训练虽然能够起到并行加速的目的，但每轮迭代过程中需消耗大量的时间进行通信. 在基于参数服务器的同步策略下，随着集群规模的增加，网络将存在更多的通信流量，拥塞问题将导致通信时间进一步延长，同时计算节点之间的算力差异也会更加显著，同步等待时间也随之增加，通信开销占比的增加制约了分布式训练性能的提升. 在环形同步策略下，尽管避免了中心节点的通信瓶颈，但随着节点数量增加，环路传递路径变长，单轮通信延迟相应上升，且算力差异和通信差异引发的同步等待依然存在，通信开销仍然制约了整体性能的提升.

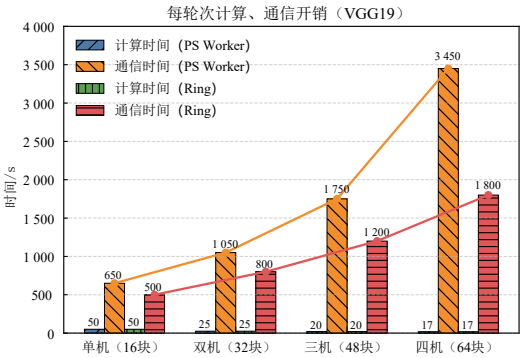


图4 VGG19模型训练计算与通信的时间开销

不同模型与场景下的分布式训练加速比如图6所示. 在16块GPU的场景中，通信已然成为瓶颈，导致各场景下的最大加速比只能达到10，远低于其理想加速比16. 随着集群规模的增加，加速比首先呈现缓慢上升的趋势，此时增加节点数量虽能够起到正向的增益效

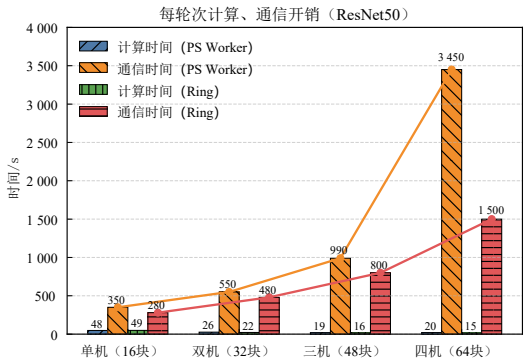


图5　ResNet50模型训练计算与通信的时间开销

果,但通信成为分布式训练中的主导瓶颈,更大规模的训练集群难以获得显著的加速效果与性能提升.

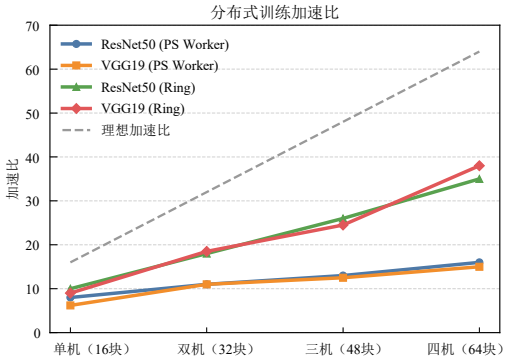


图6　分布式训练加速比与集群节点规模的关系

## 4　混合同步策略

针对现有参数同步策略的通信瓶颈问题,本文以减少总通信数据量、缓解交换机输出端口拥塞、适配传统多GPU卡服务器结构为目标,提出了一种混合同步策略PASSING. 该策略的实现包括两步:第一步,将环形同步策略与参数服务器同步策略混合,在服务器内使用环形同步策略,在服务器间使用参数服务器同步策略,我们将该步实现的同步策略称为PSRing;第二步,将PSRing策略中的参数服务器同步策略更换为基于网内计算的同步策略,最终实现PASSING. 下面详细描述每一步的设计.

### 4.1　第一步:PSRing同步策略

PSRing混合同步策略所适用的拓扑架构如图7所示,分布式集群包含$s$台计算服务器、$p$台参数服务器,每台服务器配备$n$块GPU,第$i$台计算服务器中的第$j$块GPU记为$G_{i,j}$. 计算服务器$S$与参数服务器$P$通过商用以太网网卡接入多台交换机组成的网络,采用该分布式集群训练的模型参数量大小为$K$.

混合同步算法的核心思想是先行在计算服务器内部进行计算节点之间的部分参数聚合,再由参数服务器主导全局参数的同步,并将聚合后的模型参数广播至各GPU进行更新.

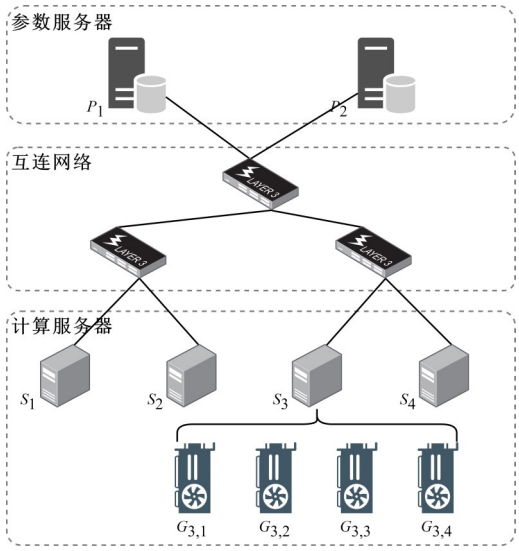


图7　混合同步策略所适配的拓扑结构

在每轮训练迭代结束后,每台服务器内部的$n$块GPU通过Intel超级通道互连线路(Intel Ultra Path Interconnect,UPI)和PCIe总线的互连构成环形逻辑拓扑,位于该逻辑环上的全部GPU通过Reduce-Scatter操作进行参数数据的交换与聚合. 具体如图8所示,服务器中的每块GPU将大小为$K$的模型参数均等切分为$n$块,其中的第$k$块记为$\text{chunk}_k$. 在首轮迭代中,编号为$G_{i,j}$的GPU将第$j$块数据$\text{chunk}_j$依逻辑环的顺时针发送至相邻的$G_{i,j'}$,其中$j'=(j+1)\%n$,该邻居GPU将收到的数据与本地参数进行聚合,并在下一轮将聚合结果发送至邻居节点. 如图8所示,在经过$n-1$轮迭代后,每块GPU持有模型聚合数据的$1/n$块,这些参数块拼接起来可组成完整的服务器内模型聚合数据. 混合同步算法在计算服务器内执行Reduce-Scatter操作的任务时间如式(1)所示,GPU在每轮迭代传输的数据量为$K/n$,瓶颈带宽取决于UPI带宽$B_{\text{UPI}}$与PCIe总线带宽$B_{\text{PCIe}}$的瓶颈值.

$$T_{\text{server}}=\frac{(n-1)K}{n\cdot\min\{B_{\text{PCIe}},B_{\text{UPI}}\}} \tag{1}$$

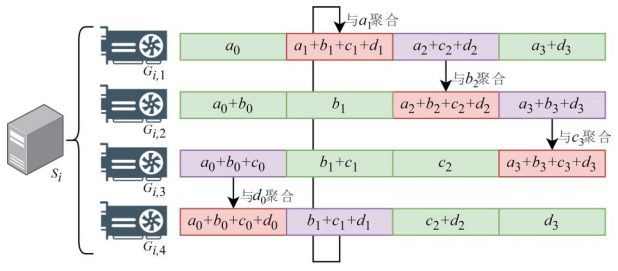


图8　机内Reduce-Scatter操作的第$n-1$次迭代

在全部计算服务器完成机内的数据聚合后，机内GPU可组合出完整的模型聚合参数，随后即可在参数服务器处进行机间的数据聚合．如图9所示，各服务器从各GPU取出所持有的1/$n$块机内聚合数据，并通过网络推送至参数服务器集群．参数服务器在完成参数聚合后，将最终的聚合结果广播至各计算服务器，此时机内的每块GPU将持有聚合结果的1/$n$块，$n$块聚合数据可共同覆盖完整的模型聚合数据．随后通过机内高带宽互连执行一次All-Gather操作，即可使每块GPU获得完整的聚合梯度数据．

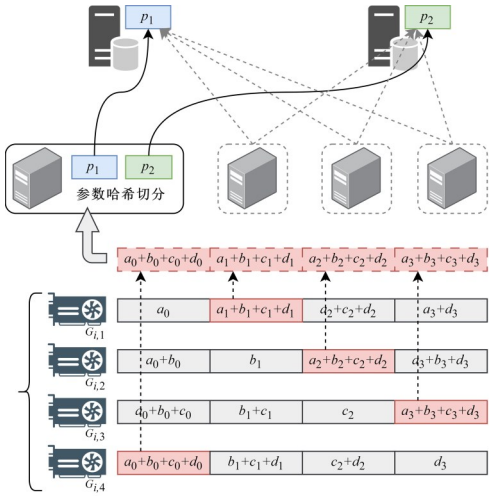


图9 计算服务器推送机内聚合数据

相较于传统的参数服务器同步策略，混合同步策略只需每台计算服务器向参数服务器推送一份模型参数，即需要传输的数据量减少1/$n$至$K$，因此来自$s$台计算服务器发送至$p$台参数服务器的通信量共为$s\cdot K$大小，各参数服务器的通信负荷即为$s\cdot K/p$，参数服务器在广播聚合数据时同样只需要向各计算服务器推送一份模型参数，进而根据网卡带宽$B_{\mathrm{NIC}}$与链路带宽$B_{\mathrm{Link}}$的瓶颈值可推导出服务器间All-Gather操作所需的时间．最终服务器将在机内的GPU内完成模型参数的广播与更新，由于通信瓶颈仍来自参数服务器，因此该过程所需的时间可重叠于服务器间参数广播过程．综上所述，All-Gather操作所需时间可如式(2)表示：

$$T_{\mathrm{param}}=2\frac{s\cdot K}{p\cdot\min\{B_{\mathrm{NIC}},B_{\mathrm{Link}}\}} \tag{2}$$

PSRing的通信总开销即如式(3)所示，为服务器内执行Reduce-Scatter以及服务器间进行All-Gather操作时间的总和．

$$T_{\mathrm{param}}=\frac{(n-1)K}{n\cdot\min\{B_{\mathrm{PCIe}},B_{\mathrm{UPI}}\}}+2\frac{s\cdot K}{p\cdot\min\{B_{\mathrm{NIC}},B_{\mathrm{Link}}\}} \tag{3}$$

在传统的参数服务器同步方法中，集群内共$n$台计算服务器、共$s\cdot n$块GPU均需向$p$台参数服务器推送大小为$K$的模型参数，同步过程所需的总时间如式(4)所示：

$$T_{\mathrm{param}}=2\frac{s\cdot n\cdot K}{p\cdot\min\{B_{\mathrm{NIC}},B_{\mathrm{Link}}\}} \tag{4}$$

因此所提出混合同步算法可以显著减少计算服务器与参数服务器之间的总数据通信量，从而缓解了参数服务器节点处的通信瓶颈现象，提高了基于参数服务器同步方法的分布式训练效率．

### 4.2 第二步：PASSING同步策略

PSRing同步策略在参数汇聚时仍会在参数服务器处产生通信热点，该通信瓶颈只有在参数服务器集群的总带宽与计算服务器集群的总带宽相匹配时才能消除．为避免参数服务器的通信负荷，可进一步通过可编程交换机的网内计算能力实现分布式训练的加速．

PSRing同步策略通过机内参数聚合降低了网络侧的通信量，而PASSING同步算法则可通过可编程交换机进一步优化数据传输．如图10所示，为实现参数服务器数据聚合能力的卸载，机内聚合数据将被推送至可编程交换机中，可编程交换机将依次对数据进行聚合，并将最终的聚合结果广播至各台计算服务器．然而可编程交换机为支持数据包线速处理能力，相较于参数服务器在处理能力、存储资源方面存在诸多限制．首先可编程交换机为保证数据包的处理速率，解析器单次解析、处理数据包的字节量存在最大限制，记交换机单次最多处理$x$个整数．再者可编程交换机内存资源有限，通常无法存储完整的模型参数，因此交换机内部的内存以资源池的形式组织，内存资源划分为$y$块存储块，每个存储块支持存储$x$个整数，算法以资源池的形式进行规划与数据同步．

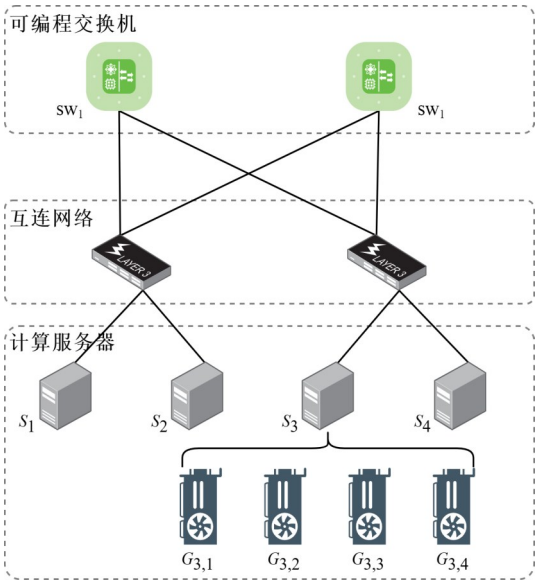


图10 网内计算同步算法拓扑示例

与PSRing同步策略一致,各服务器在内部执行完Reduce-Scatter操作后,每块GPU将持有机内聚合数据的1/$n$块,即第$j$块GPU的chunk$_{j'}$为机内聚合数据的一部分,其中$j'=(j+1)\%n$. 随后计算服务器如图11、算法1所示负责机内聚合数据的组合与推送,各服务器首先将待推送的模型参数通过乘以合适的缩放因子转换为整数形式,以便仅能支持整数运算的可编程交换机能够处理. 缩放因子的选择应根据模型参数的数值分布进行合理的设置,其基本原则为:缩放因子的选择需要在保证聚合时不会发生数值溢出的前提下,尽可能取最大值,以减少对实际训练中模型收敛性和最终精度的影响[38,40]. 例如在模型VGG19、ResNet50的训练中,推荐缩放因子的值设置为$10^8$. 服务器随后通过哈希函数切分为$p$个向量$\boldsymbol{k}_1,\boldsymbol{k}_2,\cdots,\boldsymbol{k}_p$,各服务器中的参数向量$\boldsymbol{k}_i$将在可编程交换机sw$_i$上汇聚. 由于每个交换机采用存储块的组织形式,服务器按序将各参数向量切分为$y$个包含$x$个整数的片段,并以$\boldsymbol{k}_i^j$标识参数向量$\boldsymbol{k}_i$的第$j$个片段. 当交换机资源池预制的存储块数量$y$小于模型向量$\boldsymbol{k}_i$的片段数时,服务器需分多轮完成同步,首轮取各模型向量$\boldsymbol{k}_i$的前$y$个片段,分别封装为$y$个独立的数据包,并发送至可编程交换机sw$_i$. 为便于可编程数据包进行数据聚合,数据包额外封装有服务器节点编号、模型向量编号$i$与片段编号$j$,交换机根据数据包的信息指示将封装的$x$个整数加和在第$j\%y$个存储块中. 如算法2所示,可编程交换机为各存储块维护一张位图,当存储块接收到服务器节点的数据包后,对应的位图通过更新位的方式记录该服务器节点. 当存储块接收到全部计算服务器发送的共$s$条片段后,存储块所存储的聚合数据将广播至全部计算服务器,同时存储块需完成位图的置零与初始化. 可编程交换机支持通过多播引擎在硬件层面实现数据包的复制并转发至多个目标端口,因此其聚合数据的广播过程处理延迟和转发处理开销极低.

在计算服务器接收到包含聚合结果的数据包后,如算法3所示,节点通过逆运算获取整数数据对应的原始数值,并根据数据包内封装的模型向量编号$i$与片段编号$j$更新相应位置的数据. 若由于交换机的存储资源限制而需要多轮同步聚合,各计算服务器需为模型向量$K_i$的第$j+y$个片段构造数据包,并同样遵循上述过程,直至全部模型向量的全部片段均完成同步聚合. 此时在各计算服务器中,第$j$块GPU的chunk$_{j'}$为最终聚合结果的一部分,其中$j'=(j+1)\%n$. 为完成最后的结果同步,需在机内的GPU间进行图12所示的All-Gather操作. 编号为$G_{i,j}$的GPU将第$(j+1)\%n$块数据依逻辑环的顺时针发送至相邻的GPU进行覆盖操作,并在下一轮

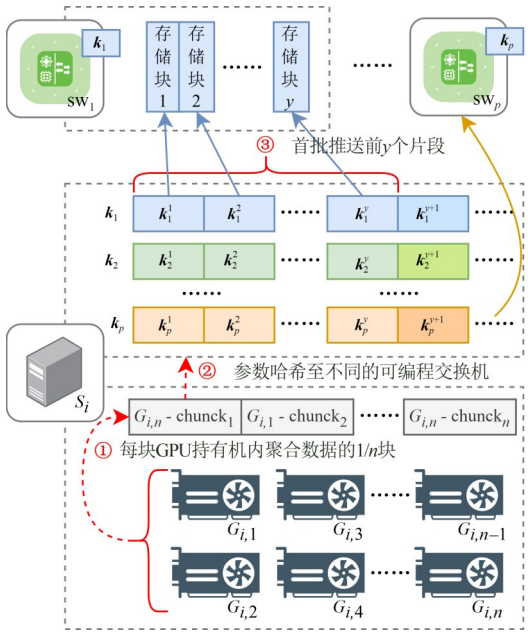


图11　网内计算同步中机内聚合数据的推送

**算法1　机内聚合数据的首批推送**

**输入**:可编程交换机数量$p$, 存储块数量为$y$, 存储块大小为$x$, 机内GPU数量$n$, 计算服务器节点编号$S_i$, 可编程交换机集合**sw** ={sw$_1$, sw$_2$,…, sw$_p$}, 机内聚合数据**chunk**=**chunk**$_1$∪**chunk**$_2$∪…∪**chunk**$_p$

**输出**:待发送数据包集合**Pkt**

1: for element in **chunk**$_i$ do:
2: $t$←convert(element)
3: $\boldsymbol{k}_i$←$\boldsymbol{k}_i$ // $t$
4: for $k=1,2,\cdots,p$ do:
5: $\left[\boldsymbol{k}_k^1,\boldsymbol{k}_k^2,\cdots,\boldsymbol{k}_k^m\right]$←$\boldsymbol{k}_k$,$m=\left\lceil\left|\boldsymbol{k}_k\right|/x\right\rceil$
6: $l$←min$(m,y)$,$m=\left\lceil\left|\boldsymbol{k}_k\right|/x\right\rceil$
7: for $j=1,2,\cdots,l$ do:
8: pkt.**payload**←$\boldsymbol{k}_k^j$
9: pkt.src←$S_i$
10: pkt.dst←sw$_k$
11: pkt.part←$j$
12: **Pkt**←**Pkt**∪pkt

中发送至邻居节点、覆盖相应位置的数据块. 在经过$n-1$轮迭代后,各计算服务器内的各GPU内的模型参数均完成了聚合与更新,集群内的全部计算节点即可利用相同的模型参数进行后续的训练.

综上所述,PASSING同步策略在服务器内的各GPU节点之间先行执行Reduce-Scatter操作,通过机内的预汇聚使得网内数据的传输量仅为传统算法的1/$n$,缓解了参数服务器处的通信热点问题. 再借助可编程交换机的线速处理能力将聚合功能进行了卸载,极大地降低了分布式训练的通信瓶颈,提高了训练的效率.

**算法2 可编程交换机的数据聚合**

**输入**：交换机编号 $sw_i$，存储块数量为 $y$，计算服务器数量 $s$，计算服务器集合 $\boldsymbol{S}=\{S_1,S_2,\cdots,S_s\}$，存储块资源 $\textbf{buckets}=\{\textbf{bucke}\,\boldsymbol{t}_1,\textbf{bucke}\,\boldsymbol{t}_2,\cdots,\textbf{bucke}\,\boldsymbol{t}_y\}$，存储块位图 $\textbf{bitmaps}=\{\textbf{bitma}\,\boldsymbol{p}_1,\textbf{bitma}\,\boldsymbol{p}_2,\cdots,\textbf{bitma}\,\boldsymbol{p}_y\}$，接收到的数据包 pkt

**输出**：待发送数据包集合 **Pkt**

$m,n,q\leftarrow$pkt.src,pkt.part%$y$,pkt.part

**bucke** $\boldsymbol{t}_n\leftarrow$**bucke** $\boldsymbol{t}_n+$pkt.**payload**

**bitma** $\boldsymbol{p}_n[m]\leftarrow 1$

if $\sum_j$ **bitma** $\boldsymbol{p}_n[j]=s$:

  for $j=1,2,\cdots,s$ do:

    pkt.**payload**$\leftarrow$**bucke** $\boldsymbol{t}_n$

    pkt.src$\leftarrow sw_i$

    pkt.dst$\leftarrow S_j$

    pkt.part$\leftarrow q$

    **Pkt**$\leftarrow$**Pkt**$\cup$pkt

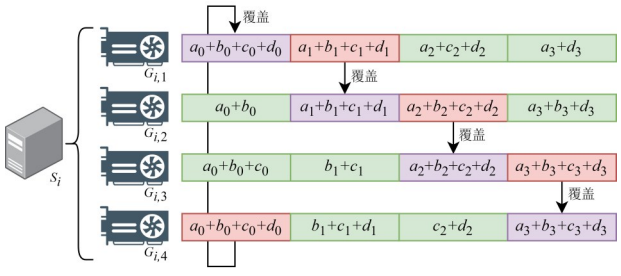


图12 网内计算同步算法中机内 All-Gather 操作的示例

**算法3 机内聚合数据的后续推送**

**输入**：服务器节点编号 $S_i$，算法1所得 $\boldsymbol{k}_i$，计算节点数量 $s$，存储块数量为 $y$，存储块大小为 $x$，接收到的数据包 pkt

**输出**：待发送数据包集合 **Pkt**

$sw_m,n\leftarrow$pkt.src,pkt.part

$\boldsymbol{k}_m^n\leftarrow \text{convert}^{-1}$(pkt.**payload**/$s$)

if $n+y\leqslant\lceil|\boldsymbol{k}_k|/x\rceil$:

  $n\leftarrow n+y$

  pkt.**payload**$\leftarrow\boldsymbol{k}_m^n$

  pkt.src$\leftarrow S_i$

  pkt.dst$\leftarrow sw_m$

  pkt.part$\leftarrow n$

  **Pkt**$\leftarrow$**Pkt**$\cup$pkt

## 5 实验结果与分析

我们搭建了实验平台，实现并部署了所提出的混合同步策略 PASSING. 本节通过实验结果的对比分析，说明混合同步策略 PASSING 所带来的性能优势.

### 5.1 实验环境

实验在5台配置有56核心的 Intel Xeon E5-2680 v4 @ 2.40 GHz CPU、128 GB DDR4 内存并运行 Ubuntu 20.04.6 LTS 操作系统的服务器上进行，同时每台服务器通过 PCIe 3.0 搭载4块 NVIDIA Tesla K80 12 GB 显卡. 在网络互连方面，可编程交换机采用1台搭载 Barefoot SDE 9.90 环境、Ubuntu 18.04.6 LTS 操作系统的 P4 交换机 EdgeCore Wedge100BF-32X，该交换机具有 32 个 100GbE QSFP28 端口. 每台服务器均搭载一块 Mellanox ConnectX-6-DX 网卡，并以图13所示的星型拓扑结构接入 P4 交换机. 值得注意的是：SwitchML 面向每台服务器配备单块 GPU 的场景而设计. 尽管理论上通过为每台服务器配置多张网卡、并将每张网卡与特定 GPU 绑定的方式，可以将 SwitchML 的部署扩展至多 GPU 环境，但该方式增加了服务器与可编程交换机之间的带宽和系统设备开销. 为确保所有策略的实验对比在统一的硬件环境下进行，我们统一为每台服务器配置单独一张网卡，并将该网卡连接至可编程交换机的一个端口. 针对 SwichML 的部署，我们通过网卡的多队列机制为每块 GPU 分配一个独立的虚拟队列，并通过网卡的多队列轮询机制实现对出口带宽的公平共享.


图13 搭建的分布式训练集群

在上述分布式训练集群中，其中四台服务器作为计算服务器. 为与分布式训练框架兼容，一台服务器承担参数服务器的功能，可编程交换机将聚合后的数据广播至各计算服务器与参数服务器. 各服务器采用 PyTorch 1.10.1 框架与 NCCL 2.10.3、GLOO 通信库部署深度学习环境. 为支持网内聚合功能，P4 交换机的内存划分出具有 32 个存储块的资源池，每个存储块提供 64 个 32 位整数的存储，即各数据包能携带 64 个 32 位整数.

### 5.2 实验性能分析

实验分别基于通信密集型模型 VGG19、计算密集型模型 ResNet50 进行，以 Cifar-10 作为训练数据集训练 100 轮次，以平均每轮次的训练时间为指标评估的指标. 实验使用基于参数服务器的同步策略、SwitchML 和 ATP 同步策略作为对比方案.

由于集群规模的限制，为充分体现大规模集群中

的真实场景,实验同比例减少各服务器的通信带宽为10 Gbps. 同时实验分别以下4种场景下进行:2台计算服务器、各服务器配备2块GPU(记为2节点/2GPU);2台计算服务器、各服务器配备4块GPU(记为2节点/4GPU);4台计算服务器、各服务器配备2块GPU(记为4节点/2GPU);4台计算服务器、各服务器配备4块GPU(记为4节点/4GPU),各场景下均设置1台服务器作为参数服务器. VGG19与ResNet50模型的实验结果分别如图14、图15所示. PASSING同步策略相较于参数服务器同步策略表现出理想的加速效果.

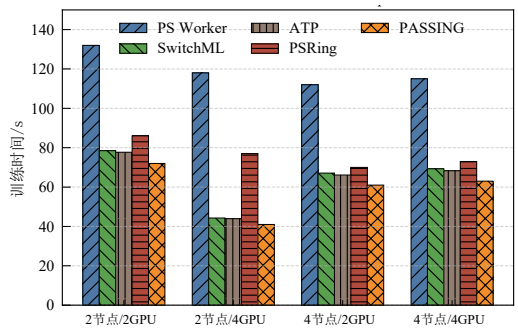


图14　10 Gbps带宽下VGG19每轮次的训练时间

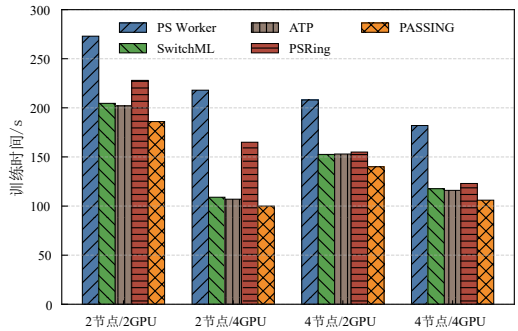


图15　10 Gbps带宽下ResNet50每轮次的训练时间

参数服务器同步策略在进行模型参数同步时,每台计算服务器中的全部$n$块GPU都将模型参数完整地推送至参数服务器,在计算服务器的网络出口处及参数服务器的网络入口处均将产生严重的拥塞问题. SwitchML和ATP通过在可编程交换芯片上实现数据平面的参数聚合,虽然在一定程度上降低了通信量并缩短了训练时间,但无法有效缓解计算服务器出口的通信压力. PSRing策略通过机内PCIe的高速链路在机内以极低的时延完成数据聚合,并通过多台参数服务器实现通信负荷均衡,有效减少计算服务器在推送数据、参数服务器在广播聚合结果时的通信量. 基于此,PASSING策略则进一步通过可编程交换机有效释放参数服务器的通信负荷.

在2节点/2GPU的实验中,PSRing策略凭借机内聚合与多参数服务器负载均衡机制,相较于参数服务器同步策略在ResNet50上节省了16.48%的训练时间. 尽管PASSING在实验中仅采用一台可编程交换机,但其凭借机内聚合与网内聚合能力,相较于SwitchML与ATP分别在ResNet50上分别节省了9.09%、7.92%的训练时间.作为通信密集型模型,VGG19在分布式训练中产生的通信数据量显著高于计算密集型的ResNet50,PSRing与PASSING策略因而能够更加显著地缓解通信瓶颈问题. 具体而言,PSRing在VGG19上的训练时间较参数服务器同步策略节省了34.85%的训练时间,PASSING相较于SwitchML与ATP则分别节省了8.26%、7.34%的训练时间.

随着分布式训练集群规模的增加,由于通信瓶颈对通信时延的影响呈现着近似指数级的关系,因此线性地减少通信量能够更加显著地优化加速效果. 在4节点/4GPU的实验中,PSRing策略相较于参数服务器同步策略在ResNet50模型和VGG19模型下分别提升了32.42%与36.52%的性能. 类似的,PASSING策略相较SwitchML与ATP在ResNet50下分别提升了9.91%与8.62%的性能,在VGG19模型下分别提升了9.09%与7.75%的性能. 因此随着训练集群规模的增加,PASSING策略训练加速的效果将更加显著.

为进一步评估在低带宽网络环境下,各方案在分布式训练时的性能差异. 各服务器网卡限速至1 Gbps,实验结果如图16、图17所示. 在低带宽的网络环境下,由于计算服务器内部的各GPU均需发送完整的模型参数,各计算服务器的网络出口处即出现严重的拥塞问题,PSRing的优势则能得到充分体现. 在4节点/4GPU的实验中,PSRing相较于参数服务器同步策略在ResNet50模型和VGG19模型下的性能提升分别高达75.82%与78.57%.PASSING策略与Switch、ATP均具有网内计算能力,凭借通信量的显著降低,PASSING在VGG19模型下相较于Switch与ATP性能分别提升了9.09%与7.47%.

实验同时如图18、图19所示统计了各方案在10 Gbps带宽下的扩展效率. 扩展效率指加速比与计算节点数

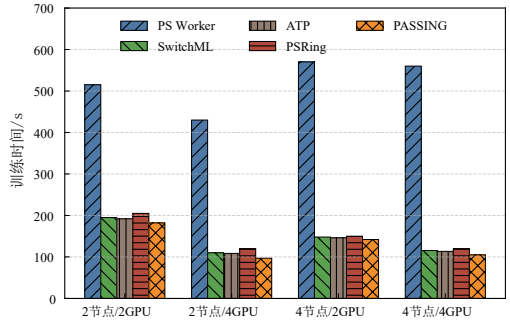


图16　1 Gbps带宽下VGG19每轮次的训练时间

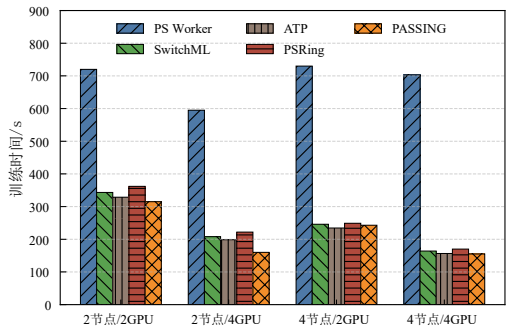


图17 1 Gbps带宽下ResNet50每轮次的训练时间

量的比值,能够直观地反映各计算节点的计算资源利用率. 由于通信开销的存在,各算法的拓展效率随着计算节点数量的增加而呈下降趋势,但由于PSRing与PASSING减少了通信开销,因此呈现出优于其他同步策略的扩展效率. 在VGG19模型中,PASSING的扩展效率在2节点/4GPU、4节点/4GPU场景下分别优于参数服务器同步策略187.80%、82.54%. 在ResNet50模型中,PASSING的扩展效率在2节点/4GPU、4节点/4GPU场景下分别优于参数服务器同步策略118.00%、71.70%.

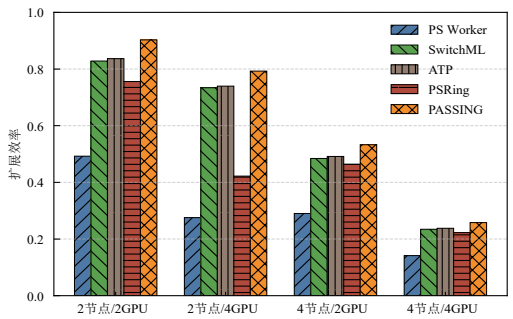


图18 10 Gbps带宽、VGG19模型下各算法的拓展效率

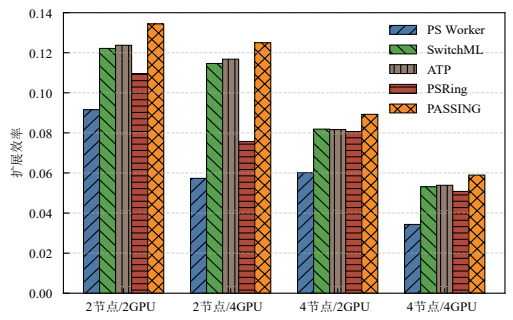


图19 10 Gbps带宽、ResNet50模型下各算法的拓展效率

综上所述,PASSING同步策略在低带宽、高带宽等各类实验场景中均表现出相较于参数服务器同步策略显著的优势,PASSING能够有效缓解分布式训练同步过程中的通信瓶颈问题,使得分布式训练在大规模集群中仍能具有优异的加速性能.

## 6 结束语

本文针对分布式训练参数服务器同步模式中的网络拥塞问题,提出了一种基于混合参数同步策略PASSING,该策略首先通过环形同步策略与参数服务器同步策略的混合,在服务器内实现了机内参数的优先聚合,避免将更多的通信负载注入网络,随后利用网内计算将参数服务器的聚合功能进行卸载,由可编程交换机负责模型参数的聚合与广播,进而实现了网络通信负荷的有效减少,提升了分布式训练的性能. 在未来的工作中,将考虑进一步探索网内计算与异步同步机制结合的可能性,以提升所提策略在异构系统中的适用性与鲁棒性.

### 参考文献

[1] ZHOU H, HU C M, YUAN Y, et al. Large language model (LLM) for telecommunications: A comprehensive survey on principles, key techniques, and opportunities[J]. IEEE Communications Surveys & Tutorials, 2025, 27(3): 1955-2005.

[2] ZHOU X C, LIU M Y, YURTSEVER E, et al. Vision language models in autonomous driving: A survey and outlook[J]. IEEE Transactions on Intelligent Vehicles, 2024, PP(99): 1-20.

[3] ZHA D C, BHAT Z P, LAI K H, et al. Data-centric artificial intelligence: A survey[J]. ACM Computing Surveys, 2025, 57(5): 1-42.

[4] QIU J N, LI L, SUN J K, et al. Large AI models in health informatics: Applications, challenges, and the future[J]. IEEE Journal of Biomedical and Health Informatics, 2023, 27(12): 6074-6087.

[5] THAKUR A, BISWAS S K, MAJUMDAR S, et al. A comprehensive review on different CNN architectures[C]// 2025 3rd International Conference on Intelligent Data Communication Technologies and Internet of Things. Piscataway: IEEE, 2025: 1997-2004.

[6] ZHAO X, WANG L M, ZHANG Y F, et al. A review of convolutional neural networks in computer vision[J]. Artificial Intelligence Review, 2024, 57(4): 99.

[7] ALZUBAIDI L, ZHANG J L, HUMAIDI A J, et al. Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions[J]. Journal of Big Data, 2021, 8(1): 53.

[8] YADAV S P, ZAIDI S, MISHRA A, et al. Survey on machine learning in speech emotion recognition and vision systems using a recurrent neural network (RNN)[J]. Archives of Computational Methods in Engineering, 2022, 29(3): 1753-1770.

[9] 梁宏涛, 刘硕, 杜军威, 等. 深度学习应用于时序预测研究综述[J]. 计算机科学与探索, 2023, 17(6):1285-1300.
LIANG H T, LIU S, DU J W, et al. Review of deep learning applied to time series prediction[J]. Journal of Frontiers of Computer Science and Technology, 2023, 17(6): 1285-1300. (in Chinese)

[10] LAGHARI A ALI, SUN Y Q, ALHUSSEIN M, et al. Deep residual-dense network based on bidirectional recurrent neural network for atrial fibrillation detection[J]. Scientific Reports, 2023, 13: 15109.

[11] RUSSAKOVSKY O, DENG J, SU H, et al. ImageNet large scale visual recognition challenge[J]. International Journal of Computer Vision, 2015, 115(3): 211-252.

[12] TAN M X, LE Q V. EfficientNet: Rethinking model scaling for convolutional neural networks[EB/OL]. (2020-09-11) [2024-12-26]. https://arXiv.org/abs/1905.11946.

[13] BROWN T B, MANN B, RYDER N, et al. Language models are few-shot learners[C]//Proceedings of the 34th International Conference on Neural Information Processing Systems. New York: ACM, 2020: 1877-1901.

[14] OPENAI, ACHIAM J, ADLER S, et al. GPT-4 technical report[EB/OL]. (2024-03-04) [2024-12-16]. https://arXiv.org/abs/2303.08774.

[15] DEAN J, CORRADO G S, MONGA R, et al. Large scale distributed deep networks[C]//Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1. New York: ACM, 2012: 1223-1231.

[16] SHOEYBI M, PATWARY M, PURI R, et al. Megatron-LM: Training multi-billion parameter language models using model parallelism[EB/OL]. (2023-03-13)[2024-12-16]. https://arXiv.org/abs/1909.08053.

[17] LI S G, LIU H X, BIAN Z D, et al. Colossal-AI: A unified deep learning system for large-scale parallel training[C]// Proceedings of the 52nd International Conference on Parallel Processing. New York: ACM, 2023: 766-775.

[18] SUN Z B, CAO H Q, WANG Y W, et al. AdaPipe: Optimizing pipeline parallelism with adaptive recomputation and partitioning[C]//Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3. New York: ACM, 2024: 86-100.

[19] ZHANG Z, XIA Y Q, WANG H L, et al. MPMoE: Memory efficient MoE for pre-trained models with adaptive pipeline parallelism[J]. IEEE Transactions on Parallel and Distributed Systems, 2024, 35(6): 998-1011.

[20] CHEN Z, DENG Y, WU Y, et al. Towards understanding the mixture-of-experts layer in deep learning[J]. Advances in Neural Information Processing Systems, 2022, 35: 23049-23062.

[21] HWANG C, CUI W, XIONG Y F, et al. Tutel: Adaptive mixture-of-experts at scale[EB/OL]. (2023-06-05) [2024-12-26]. https://arXiv.org/abs/2206.03382.

[22] HUANG Y P, CHENG Y L, CHEN D H, et al. GPipe: Efficient training of giant neural networks using pipeline parallelism[C]//Neural Information Processing Systems. New York: Curran Associates Inc., 2018: 103-112.

[23] NARAYANAN D, HARLAP A, PHANISHAYEE A, et al. PipeDream: Generalized pipeline parallelism for DNN training[C]//Proceedings of the 27th ACM Symposium on Operating Systems Principles. New York: ACM, 2019: 1-15.

[24] NARAYANAN D, PHANISHAYEE A, SHI K Y, et al. Memory-efficient pipeline-parallel DNN training[C]//International Conference on Machine Learning. Vancouver: Proceedings of Machine Learning Research, 2020: 7937-7947.

[25] WANG H, TIAN H, CHEN J R, et al. Towards domain-specific network transport for distributed DNN training[C]// Proceedings of the 21st USENIX Symposium on Networked Systems Design and Implementation. New York: ACM, 2024: 1421-1443.

[26] AHMAD AWAN A, HAMIDOUCHE K, HASHMI J M, et al. S-caffe: Co-designing MPI runtimes and caffe for scalable deep learning on modern GPU clusters[J]. ACM SIGPLAN Notices, 2017, 52(8): 193-205.

[27] DEVLIN J, CHANG M W, LEE K, et al. BERT: Pre-training of deep bidirectional transformers for language understanding[EB/OL]. (2019-05-24) [2024-12-25]. https://arXiv.org/abs/1810.04805.

[28] TANG Z H, SHI S H, WANG W, et al. Communication-efficient distributed deep learning: A comprehensive survey [EB/OL]. (2023-09-01)[2024-12-26]. https://arXiv.org/abs/2003.06307.

[29] GIBIANSKY A. Bringing HPC techniques to deep learning[EB/OL]. (2017-02-21) [2024-12-26]. http://research.baidu.com/bringing-hpc-techniques-deep-learning/.

[30] JIA X Y, SONG S T, HE W, et al. Highly scalable deep learning training system with mixed-precision: Training ImageNet in four minutes[EB/OL]. (2018-07-30) [2024-12-26]. https://arXiv.org/abs/1807.11205.

[31] TANAKA Y, KAGEYAMA Y. ImageNet ResNet-50 training in 224 seconds[EB/OL]. (2019-03-05) [2024-12-26]. https://arxiv.org/abs/1811.05233v1.

[32] YING C, KUMAR S, CHEN D H, et al. Image classification at supercomputer scale[EB/OL]. (2018-12-02) [2024-12-26]. https://arXiv.org/abs/1811.06992.

[33] CHO M, FINKLER U, SERRANO M, et al. BlueConnect: Decomposing all-reduce for deep learning on hetero-

geneous network hierarchy[J]. IBM Journal of Research and Development, 2019, 63(6): 1: 1-1: 11.

[34] PEI J, AMER-YAHIA S, HUANG Y Z, et al. FlexPS: Flexible parallelism control in parameter server architecture[J]. Proceedings of the VLDB Endowment, 2018, 11(5): 566-579.

[35] JIANG Y M, ZHU Y B, LAN C, et al. A unified architecture for accelerating distributed DNN training in heterogeneous GPU/CPU clusters[C]//Proceedings of the 14th USENIX Conference on Operating Systems Design and Implementation. New York: ACM, 2020: 463-479.

[36] 刘忠沛, 杨翔瑞, 杨凌, 等. CAInNet: 面向AI加速的通算一体网内计算模型[J]. 计算机学报, 2025, 48(1): 19-34.
LIU Z P, YANG X R, YANG L, et al. CAInNet: In-network computing model for AI acceleration[J]. Chinese Journal of Computers, 2025, 48(1): 19-34. (in Chinese)

[37] 刘宏岩, 张栋, 吴春明. 基于可编程交换机的网内灰色故障检测技术研究进展[J]. 电子学报, 2024, 52(10): 3613-3622.
LIU H Y, ZHANG D, WU C M. Empowering in-network gray failure detection with programmable switches[J]. Acta Electronica Sinica, 2024, 52(10): 3613-3622. (in Chinese)

[38] SAPIO A, CANINI M, HO C Y, et al. Scaling distributed machine learning with in-network aggregation[EB/OL]. (2020-09-30)[2024-12-26]. https://arXiv.org/abs/1903.06701.

[39] LIU S, WANG Q L, ZHANG J Y, et al. In-network aggregation with transport transparency for distributed training[C]//Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3. New York: ACM, 2023: 376-391.

[40] LAO C I, LE Y F, MAHAJAN K S, et al. ATP: In-network aggregation for multi-tenant learning[C]//Symposium on Networked Systems Design and Implementation. Boston: USENIX Association, 2021: 741-761

[41] PAN H, CUI P L, LI Z Y, et al. Enabling fast and flexible distributed deep learning with programmable switches[EB/OL]. (2022-08-10)[2024-12-19]. https://arXiv.org/abs/2205.05243.

[42] FANG J, ZHAO G M, XU H L, et al. GRID: Gradient routing with in-network aggregation for distributed training[J]. IEEE/ACM Transactions on Networking, 2023, 31(5): 2267-2280.

[43] DE SENSI D, DI GIROLAMO S, ASHKBOOS S, et al. Flare: Flexible in-network allreduce[C]//International Conference for High Performance Computing, Networking, Storage and Analysis. Piscataway: IEEE, 2022: 1-15.

[44] LIU L, ZHOU P, SUN G, et al. Topologies in distributed machine learning: Comprehensive survey, recommendations and future directions[J]. Neurocomputing, 2024, 567: 127009.

[45] 王帅, 李丹. 分布式机器学习系统网络性能优化研究进展[J]. 计算机学报, 2022, 45(7): 1384-1411.
WANG S, LI D. Research progress on network performance optimization of distributed machine learning system[J]. Chinese Journal of Computers, 2022, 45(7): 1384-1411. (in Chinese)

## 作者简介

**余晓杉** 男, 1986年出生于陕西省汉中市. 西安电子科技大学通信工程学院讲师、硕士生导师. 主持国家自然科学基金青年项目、博士后面上项目、陕西省重点研发子课题项目等, 主要研究方向为高性能互连网络、光交换技术、分布式人工智能加速技术等.
E-mail: xsyu@xidian.edu.cn

**顾华玺** 男, 1977年出生于河北省石家庄市. 西安电子科技大学通信工程学院教授、博士生导师. 主要研究方向包括网络交换技术、光互连、片上网络、数据中心网络以及面向分布式AI应用的网络关键技术等. 中国电子学会会员编号: E190015869S.
E-mail: hxgu@xidian.edu.cn

**周肇星** 男, 1997年出生于安徽省铜陵市. 现为西安电子科技大学通信工程学院博士研究生. 主要研究方向为数据中心网络与光交换技术.
E-mail: zxzhou_xidian@163.com

**王佳昆** 男, 1999年出生于江西省丰城市. 硕士毕业于西安电子科技大学. 主要研究方向为分布式训练加速.
E-mail: jiakunwang_xidian@163.com