智能系统学报
CAAI TRANSACTIONS ON INTELLIGENT SYSTEMS

面向异构分布式机器学习的动态自适应并行加速方法

马翔, 申国伟, 郭春, 崔允贺, 陈意

引用本文:
马翔,申国伟,郭春,崔允贺,陈意. 面向异构分布式机器学习的动态自适应并行加速方法[J]. 智能系统学报, 2023, 18(5): 1099–1107.
MA Xiang, SHEN Guowei, GUO Chun, et al. Dynamic adaptive parallel acceleration method for heterogeneous distributed machine learning[J]. *CAAI Transactions on Intelligent Systems*, 2023, 18(5): 1099–1107.

在线阅读 View online: https://dx.doi.org/10.11992/tis.202209024

DOI: 10.11992/tis.202209024
网络出版地址: https://kns.cnki.net/kcms2/detail/23.1538.TP.20230531.1529.004.html

面向异构分布式机器学习的动态自适应并行加速方法

马翔, 申国伟, 郭春, 崔允贺, 陈意
(贵州大学 计算机科学与技术学院, 贵州 贵阳 550025)

摘 要: 分布式机器学习因其优越的并行能力成为人工智能领域复杂模型训练的常用技术。然而, GPU 升级换代非常快, 异构集群环境下的分布式机器学习成为数据中心、研究机构面临的新常态。异构节点之间训练速度的差异使得现有并行方法难以平衡同步等待和陈旧梯度的影响, 从而显著降低模型整体训练效率。针对该问题, 提出了一种基于节点状态的动态自适应并行方法 (dynamic adaptive synchronous parallel, DASP), 利用参数服务器动态管理节点训练时的状态信息并对节点的并行状态进行划分, 通过节点状态信息自适应调整每个节点的并行状态, 以减少快速节点对全局模型参数的同步等待时间与陈旧梯度的产生, 从而加快收敛效率。在公开数据集上的实验结果表明, DASP 比主流方法收敛时间减少了 16.9%~82.1%, 并且训练过程更加稳定。

关键词: 异构集群; 机器学习; 数据并行; 分布式训练; 参数服务器; 落后者; 陈旧梯度; 大规模深度学习
中图分类号: TP18 **文献标志码**: A **文章编号**: 1673−4785(2023)05−1099−09

中文引用格式: 马翔, 申国伟, 郭春, 等. 面向异构分布式机器学习的动态自适应并行加速方法 [J]. 智能系统学报, 2023, 18(5): 1099–1107.
英文引用格式: MA Xiang, SHEN Guowei, GUO Chun, et al. Dynamic adaptive parallel acceleration method for heterogeneous distributed machine learning[J]. CAAI transactions on intelligent systems, 2023, 18(5): 1099–1107.

Dynamic adaptive parallel acceleration method for heterogeneous distributed machine learning

MA Xiang, SHEN Guowei, GUO Chun, CUI Yunhe, CHEN Yi
(College of Computer Science and Technology, Guizhou University, Guiyang 550025, China)

Abstract: Distributed machine learning has emerged as a common technique for training complex artificial intelligence models due to its excellent parallelism capability. However, GPU upgrades are exceedingly fast, and distributed machine learning in a heterogeneous cluster environment is increasingly being adopted by data centers and research institutions. The difference in training speed between heterogeneous nodes makes it difficult for existing parallel strategies to balance the effects of synchronized waits and stale gradients, considerably reducing the model's overall training efficiency. To address this problem, a node state-based dynamic adaptive parallel strategy, namely, dynamic adaptive synchronous parallel (DASP), is proposed using a parameter server to dynamically manage the state information of nodes during training and to divide the parallel states of nodes. The parallel state of each node is adaptively adjusted by the state information of the node to reduce the synchronization waiting time of fast nodes for global model parameters and the generation of stale gradients, speeding up the convergence efficiency. Experimental results on publicly available datasets show that DASP not only reduces the convergence time by 16.9%~82.1% compared to mainstream strategies but also makes the training process more stable.

Keywords: heterogeneous clusters; machine learning; data parallel; distributed training; parameter servers; stragglers; stale gradient; large-scale deep learning

近年来机器学习 (machine learning, ML) 广泛应用到了各种领域, 在图像分类[1]、目标检测[2-3]、语义分析[4] 等领域取得了显著的效果。然而, 随

收稿日期: 2022−09−14. **网络出版日期**: 2023−06−01.
基金项目: 国家自然科学基金项目 (62062022).
通信作者: 申国伟. E-mail: gwshen@gzu.edu.cn.

着训练数据量的增加和模型复杂度的提升，在单台机器上训练 ML 模型变得非常困难。因此，为了提高训练速度，将训练任务部署在多个计算节点进行并行处理是常用的技术手段[5]。学术界和工业界对许多分布式 ML 系统进行了深入研究，例如 PyTorch[6]、TensorFlow[7]。大多数现有系统利用数据并行模式将分布式机器学习任务部署在由高性能计算节点组建的集群之上，比如公共云环境。然而，随着硬件架构的迭代升级与云技术的高速发展，在实际应用场景中，分布式机器学习集群普遍存在异构性[8]。其异构性主要包括以下方面：1) 由于集群中计算节点配备的计算设备 (GPU) 类型不同，导致节点间的计算性能不一致 (即硬件异构性)；2) 集群中各计算节点与服务器相连的网络带宽与拓扑结构存在差异，导致服务器与各节点之间的通信时间不一致 (即网络异构性)[9]；3) 集群资源往往由多个用户所共享，用户之间向集群所提交的计算任务对资源的争用造成同一物理节点在连续迭代训练中计算速度动态变化 (即动态异构性)[10]。

为保证分布式训练实现与顺序训练相同的收敛性，集群中的节点需要应用相应参数并行方法同步各自的进度。当前主流并行方法有基于同步思想的批量同步并行方法 (bulk synchronous parallel, BSP) 与基于异步思想的异步并行方法 (asynchronous parallel, ASP)。然而，由于集群异构的特点，主流并行方法并不能表现出良好的迭代质量与收敛效率。其中，BSP 每次迭代同步所有节点的模型参数以保证迭代质量,但是快速节点等待慢速节点 (stragglers) 同步所消耗的时间受集群环境影响大，并且延迟快速节点开始新一轮的迭代降低了计算资源的利用率和收敛效率[11-12]。ASP 各节点独立迭代以最大化迭代速度，但是节点参数的不及时同步造成节点间参数新旧各异，旧参数所产生的陈旧梯度增加了训练的不稳定性与全局模型收敛的不确定性[13-14]。

因此，在异构集群环境中部署分布式学习任务时，如何能充分利用计算资源来加快迭代速度又能降低陈旧梯度的产生成为一个关键性问题。针对此问题，本文提出了基于节点状态的动态自适应同步并行方法 (dynamic adaptive synchronous parallel, DASP)，通过对节点训练状态的管理和节点并行状态的动态调整提高迭代速度与收敛效率。

1　相关工作

异构环境下的分布式计算一直是分布式计算社区感兴趣的话题[15]。针对异构集群下，主流分布式训练并行方法带来的同步等待与陈旧值问题。研究人员主要从减少同步频率、降低同步代价与消除 stragglers 等方面提出了多种解决方案。

在减少同步频率方面，Ho 等[16] 提出了延迟同步并行方法 (stale synchronous parallel, SSP)，该方法在训练开始之前设置延迟同步阈值，当节点之间的迭代次数不超过阈值时，计算节点使用 ASP 进行训练，否则快速节点等待 stragglers 进行同步以消除滞后参数。Li 等[17] 也引入了有界延迟，不同的是它考虑了所有工作节点的迭代，而不是让每个节点计算自己的迭代。由于节点训练状态的变化，固定的有界阈值可能并不适合整个训练过程，因此 Zhao 等[18] 提出了动态延迟同步并行 (dynamic stale synchronous parallel, DSSP) 来解决静态有界延迟问题。它设定延迟阈值上下限，允许节点延迟在训练过程中在该范围内动态变化。上述方法都有一定局限性，并且在异构集群中，以节点迭代次数作为参考指标可能会退化为 BSP[19-20]。

在降低同步代价方面，Chen 等[21] 尝试通过添加备份节点来优化 BSP。即在 BSP 中有 N 个节点，然后添加 m 个备份节点，因此在训练期间有 $N+m$ 个节点。在每次迭代完成时，参数服务器只接收前 N 个节点的梯度，并删除 m 个慢速节点的梯度，以减少同步等待时间。但是每次迭代都会随机浪费 m 个节点的计算数据。根据此思想，Teng 等[22] 提出了贝叶斯分布式随机梯度下降 (Bayesian distributed stochastic gradient descent, BDSGD)，该算法通过选择一个截止值来减轻同步代价，而超出该截止值所提交的梯度消息会被忽略。Sun 等[23] 从节点静态性能出发提出分组延迟同步并行 (grouping stale synchronous parallel, GSSP) 方法，该方法将性能相近的工作节点分为同一组，组内采用 ASP，组间采用 SSP。该方法限制了 stragglers 的影响范围，但是需要在训练前确定集群中各节点性能，并且没有关注训练过程中集群资源共享导致节点性能动态变化的问题。

从消除 stragglers 角度出发，Harlap 等[24] 提出了 FlexRR 灵活并行方案，该方案在计算节点之间动态点对点工作重新分配，将 stragglers 的训练任务按比例卸载到快速节点中。FlexRR 在训练过程中能根据节点迭代状态实现最优工作分配，但是在进行任务动态分配时需要花费更长的网络传输时间。Xu 等[25] 提出了动态批处理大小机制，该机制通过增大 stragglers 的 Batch size 来加快其处理速度，但是在异构特点突出的集群中，调整 Batch size 带来速度提升不足以消除 stragglers 的

影响, 并且可能出现 Batch size 过大的风险。

上述工作虽然在一定程度上缓解了异构集群下 stragglers 与陈旧梯度对模型收敛效率的影响, 但是依然存在不足, 以节点迭代次数表示陈旧参数不够精准和灵活, 未有效地利用节点训练时的状态信息, 因此难以适应资源共享情况下节点迭代速度的动态变化。

2　动态自适应同步并行方法

针对当前并行方法的不足与缺点, 本文提出 DASP 并行方法, 框架如图 1 所示。该方法通过参数服务器管理所有工作节点的状态信息, 从中快速识别 stragglers 与参数滞后节点。其次, 参数服务器利用状态阈值对并行状态进行合理划分, 并且根据工作节点运行时状态信息判断所处状态, 有效地控制同步等待时间与滞后参数的影响。最后, 参数服务器根据节点的并行状态, 对各节点采用合适的参数更新机制。下面将分别讨论 DASP 方法的节点状态管理、并行状态管理以及参数更新管理。

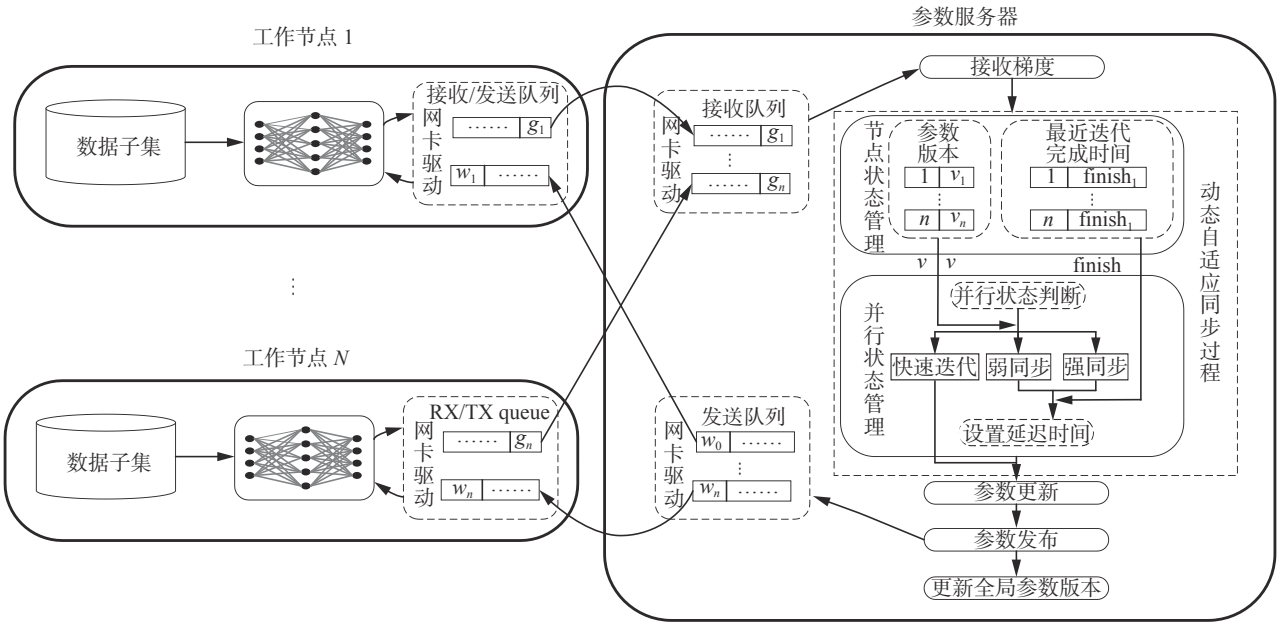

图 1　DASP 框架

Fig. 1　DASP Framework

2.1　节点状态管理

为了准确实时地标识使用陈旧参数的节点, 参数服务器管理所有节点运行时的状态信息, 其主要是工作节点当前使用的参数版本 v 与工作节点最近一轮迭代完成时间 f。假设 t 时刻工作节点参数版本 $v=<v_1^t, v_2^t, \cdots, v_n^t>$ 与最近一轮迭代完成时间 $f=<f_1^t, f_2^t, \cdots, f_n^t>$, 其中 v_1^t 表示 1 号工作节点在 t 时刻使用的参数版本, f_1^t 表示 1 号工作节点 t 时刻最近迭代完成时间, 并且利用 $v_{\max}^t$ 表示 t 时刻最新的参数版本。假设在 $t+1$ 时刻收到 1 号工作节点梯度, 参数服务器对全局模型权重更新之后进一步更新 1 号工作节点的参数版本 $v_1^{t+1}=v_{\max}^t+1$ 和最近迭代完成时间 $f_1^t=f_1^{t+1}$。通过管理节点运行时状态信息, 参数服务器能够及时准确地判断哪些工作节点使用了陈旧模型参数, 且对于一个需要进行同步操作的工作节点, 根据运行时从每个工作节点收集的最近迭代完成时间, 动态生成该工作节点与其他工作节点的最小等待时间。

在 DASP 中, 针对 stragglers 的处理采用了参数版本标识的方法, 以代替传统以迭代次数标识 stragglers。因为一些迭代次数较少的计算节点在完成局部梯度传输并更新全局模型参数后, 它们所使用的模型参数也是新于其他计算节点。

DASP 通过对节点状态信息的管理, 参数服务器利用工作节点参数版本能准确反映节点陈旧参数的使用程度, 同时, 结合参数版本信息与工作节点最近迭代完成时间确定工作节点的同步点并动态设定最小等待时间以减少同步频率与同步代价。

2.2　并行状态管理

在 DASP 中, 为了更好平衡同步等待与陈旧梯度对全局模型收敛效率的影响, 如图 2 所示, 并行状态管理利用状态阈值 $S_{\min}$ 与 $S_{\max}$ 将工作节点的同步状态合理划分为 3 个状态, 即快速迭代状态 (quick iteration state, QIS) 、弱同步状态 (weak synchronization state, WSS) 与强同步状态 (force

synchronization state, FSS)。当参数服务器收到工作节点的局部梯度值时，并行状态管理模块根据节点状态管理中的工作节点参数版本信息与状态阈值决定节点所处的并行状态 P_{state}，后续参数更新管理根据节点所处的并行状态设定对全局模型参数的更新机制。

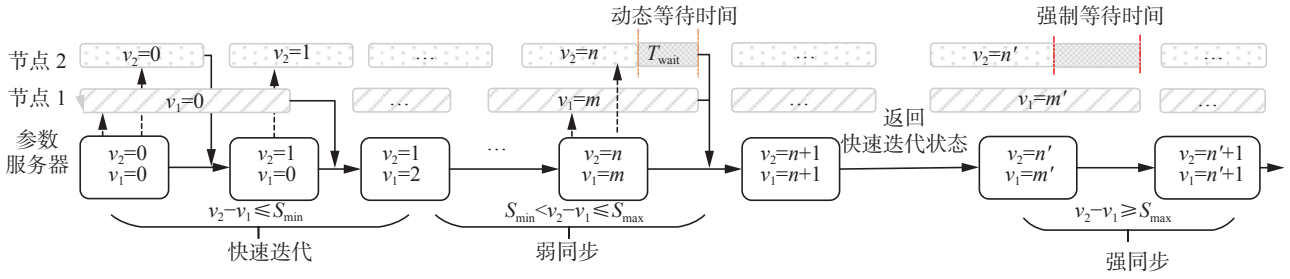

图 2　DASP 并行状态管理

Fig. 2　DASP parallel state management

1) 快速迭代。

从减少同步频率角度出发，DASP 引入快速迭代状态。所有工作节点开始迭代训练时，节点状态管理中的参数版本信息都为 0。此时不存在陈旧参数，并行状态管理将所有工作节点初始并行状态设置为 QIS。QIS 不对工作节点设立同步屏障，工作节点能完全异步地对参数服务器中全局模型进行更新，使节点可以快速开始新一轮的迭代，同时为了避免节点之间参数差距过大而造成收敛方向偏差，DASP 设立状态阈值下限 $S_{\min}$，保证 QIS 下的工作节点之间参数版本之差不超过 $S_{\min}$。因此，在 DASP 方法下允许工作节点进行异步迭代提高资源利用率，同时控制节点之间的参数陈旧程度在一定范围之内。

2) 弱同步。

当集群中某几个工作节点版本差距超过 $S_{\min}$ 时，DASP 并不会立即强制所有节点进行同步操作，因为此时只有少数节点参数版本具有较大差距，所以只需尝试控制版本相差较大节点，从而限制同步操作影响的范围。因此，DASP 在并行状态管理中引入弱同步状态。在工作节点之间的参数版本超过 $S_{\min}$ 而不大于 $S_{\max}$ 时，并行状态管理会根据节点状态所维护的节点最近迭代完成时间设置合理延迟时间：

$$T_{\text{wait}}^{i} = \alpha |f_i - f_j| \tag{1}$$

式中：T_{wait}^{i} 表示最新参数版本的节点 i 在开始新一轮迭代之前延迟等待时间，α 为权重值，f_i 为节点 i 的最近迭代完成时间，f_j 为参数版本最小 $v_{\min}$ 节点的最近迭代完成时间。如图 2 中，在动态延迟等待时间内，最新参数版本的节点有可能与最旧参数版本节点进行同步。因此，DASP 方法可以在弱同步阶段利用很小的同步代价消除陈旧参数的影响，并且使节点状态并行状态再次转换为 QIS。

3) 强同步。

WSS 在一定程度上缓解了工作节点同步等待问题与陈旧参数的影响，不过在工作节点之间具有突出异构性时，节点并不能在动态延迟时间 T_{wait} 内成功完成同步。因此为了使 DASP 适应各种集群环境并且保证在节点之间参数版本太大时能及时进行参数同步消除陈旧参数，DASP 在并行状态管理引入了强同步状态，一旦工作节点之间参数版本差值超过 $S_{\max}$，那么节点就处于强同步状态。此时 DASP 会设立显式同步栅栏，保证最新参数版本节点与最旧参数版本的节点在同步约束下消除陈旧参数，以防止过度滞后的梯度增加训练的不稳定性。所以 DASP 最后通过 FSS 同步约束下保证集群所有节点参数陈旧程度在 $S_{\max}$ 范围内，进而防止陈旧参数影响的扩大。

并行状态管理具体算法如算法 1 所示。

算法 1　PS 并行状态管理算法

输入　节点参数版本 v，节点最近迭代完成时间 f

输出　节点 P_{state}

1) while receive local gradient g_n^i from node n do

2) $f_n^i \leftarrow t_n^i - t_n^{i-1}$; /*$t_n^i$ 表示接收到 n 节点第 i 轮迭代梯度的时间*/

3) $v_n \leftarrow \text{IdToV}(v,n)$;

4) $v_{\min} \leftarrow \min(v)$; /*获取当前最旧参数版本*/

5) if $v_n - v_{\min} \leqslant S_{\min}$ then

6) $P_{\text{state}}^n \leftarrow$QIS; /*节点并行状态为 QIS*/

7) else if $S_{\min} < v_n - v_{\min} \leqslant S_{\max}$ then

8) $P_{\text{state}}^n \leftarrow$WSS;

9) $\text{id}_{\min} \leftarrow \text{VTOId}(v_{\min})$; /*获取最旧参数版本的节点 id*/

10) $f_{\text{id}_{\min}}^{i-1} \leftarrow \text{FindTime}(\text{id}_{\min}, f)$ /*找到最旧参数版本节点的最近迭代完成时间*/

11) $T^n_{\text{wait}} = \alpha|f^i_n - f^{i-1}_{\text{id}_{\min}}|$ /*设置动态延迟时间*/

12) Wait time T^n_{wait}

13) else

14) $P^n_{\text{state}} \leftarrow$ FSS;

15) Wait for the iteration of the oldest parameter version node $\text{id}_{\min}$ to complete

16) end if

17) end while

2.3　参数更新管理

参数更新管理模块基于分布式小批量 SGD 算法并根据并行状态管理模块所确定的节点并行状态对全局模型参数进行更新。在参数更新管理中, 并行状态与参数更新规则为

$$\begin{cases} w_{t+1} = w_t - \gamma \dfrac{1}{N}\dfrac{1}{B}\left[\displaystyle\sum_{j=1}^{B}\nabla f(w_t; d^i_{t,j})\right] \\ w_{t+1} = w_t - \gamma \dfrac{1}{N}\dfrac{1}{B}\left[\displaystyle\sum_{i=1}^{K}\sum_{j=1}^{B}\nabla f(w_t; d^i_{t,j})\right] \end{cases} \tag{2}$$

式中: N 是工作节点个数, B 表示批大小, $f(w_t; d^i_{t,j})$ 为第 j 批训练样本的损失值, K 表示节点在节点延迟时间内参数服务器所接收到局部梯度节点的数量。处于 QIS 下节点的对服务器全局参数更新规则, 处于 WSS 与 FSS 节点的服务器全局参数更新更新规则。

参数更新具体如算法 2 所示。

算法 2　参数更新算法

输入　节点个数 N, 学习率 γ

输出　最新参数版本参数 w_{new}

1) Receive local gradient g^i_{n} from node n

2) Paralle state management sets the parallel state of node n

3) if P^n_{state} == QIS then

4) $\Delta w_i \leftarrow \dfrac{1}{N}\dfrac{1}{B}\left[\displaystyle\sum_{j=1}^{B}\nabla f\left(w_t; d^n_{i,j}\right)\right]$;

5) else

6) $\Delta w_i \leftarrow \dfrac{1}{N}\dfrac{1}{B}\left[\displaystyle\sum_{i=1}^{K}\sum_{j=1}^{B}\nabla f\left(w_t; d^n_{i,j}\right)\right]$;

7) $w_{i+1} \leftarrow w_i - \gamma\Delta w_i$;

8) Send(w_{i+1},n); /*将新版本参数发送给节点 n*/

9) Atomic operation updating node n parameter version

3　实验评估

3.1　实验设置

为了验证 DASP 策略的有效性, 本文针对同构与异构两种集群环境进行了实验。该集群中拥有 6 个计算节点与一个参数服务器节点。参数服务器节点的硬件信息与软件信息如表 1 所示, 工作节点硬件信息与软件信息如表 1、2 所示。

表 1　同构环境下工作节点信息

Table 1　Information on the work nodes in the homogeneous environment

型号/版本	详细信息
CPU型号	Intel(R) Xeon(R) Gold 5220 CPU @ 2.20 GHz
GPU型号	NVIDIA GeForce RTX 2080Ti
内存容量	256 GB
操作系统	Ubuntu 18.04 x86_64
网卡信息	Intel Corporation Ethernet Connection X722
CUDA版本	CUDA版本
PyTorch版本	1.4.0

表 2　异构环境下异构工作节点信息

Table 2　Information on heterogeneous work nodes in a heterogeneous environment

型号/版本	详细信息
CPU型号	AMD EPYC 7742 64-Core Processor
GPU型号	NVIDIA DGX A100-SXM4-40 GB
内存容量	1 TB
操作系统	Ubuntu 18.04 x86_64
网卡信息	Intel Corporation I210
CUDA版本	11.1
PyTorch版本	1.4.0

本实验采用了 MNIST、CIFAR-10 两种常用的图像识别数据集来进行分布式训练。在实验过程中, 本文采用 LeNet-5、ResNet-18[26] 和 VGG-19[27] 常用的深度学习模型进行训练, 并应用 BSP、ASP、SSP、DSSP 与 DASP 进行对比分析。

为了量化不同并行策略的性能, 使用了多个常用的评估标准, 包括平均迭代时间 (average iteration time)、收敛时间 (convergence time), 并且利用模型在测试集上准确率的波动程度表示陈旧梯度对收敛方向的影响。其中训练具体参数如表 3 所示。

表 3　具体训练参数

Table 3　Specific training parameters

参数名	参数值
算法	mini-batch SGD
学习率	0.001
动量值	0.9
batch大小	64
SSP延迟阈值	3
$S_{\min}$	3
$S_{\max}$	6

3.2　实验结果分析

为了量化陈旧梯度对收敛趋势的影响程度，首先测试了在 BSP、ASP、SSP、DSSP 与 DASP 并行方法下模型在测试集上准确率变化情况。如图 3 与图 4 所示，对于复杂度较低的 LeNet-5，在两种集群下不同并行方法都能快速稳定地达到收敛。对于 ResNet-18 与 VGG19_bn，BSP 在同构与异构环境下都能稳定达到收敛，这是因为其同步特性，所以节点之间每次迭代都使用相同的全局模型，不存在参数偏差的影响。而 ASP 出现了明显的波动，这是由于 ASP 异步特性无法及时解决节点所使用的陈旧参数而最终影响整体模型。相较于 ASP，SSP、DSSP 与 DASP 在准确率上的表现比较稳定，并且随着集群异构性越明显，DASP 的表现要比 SSP 和 DSSP 越好。因为在 DASP 下处于弱同步状态下的高参数版本节点可以通过动态延迟等待一定时间，在这段时间内低参数版本可能与高参数版本节点进行同步从而减少陈旧参数的产生，而当节点间参数版本相差较大时，在强同步状态的控制下高参数版本节点会强制与低参数版本节点进行同步，从而限制陈旧参数影响的扩散。DASP 通过这两种状态降低了节点陈旧参数对全局模型的影响。

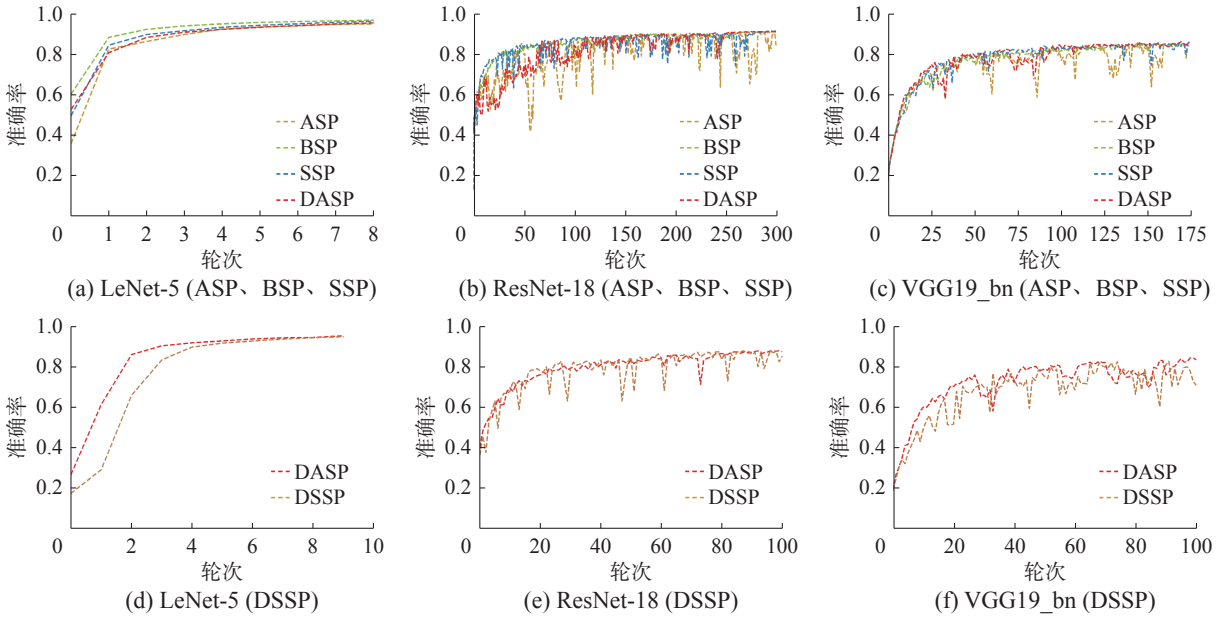

图 3　同构集群下不同并行方法 LeNet-5、ResNet-18、VGG19_bn 测试集准确率

Fig. 3　Accuracy of test sets of LeNet-5, ResNet-18, VGG19_bn with different parallel methods under homogeneous clusters

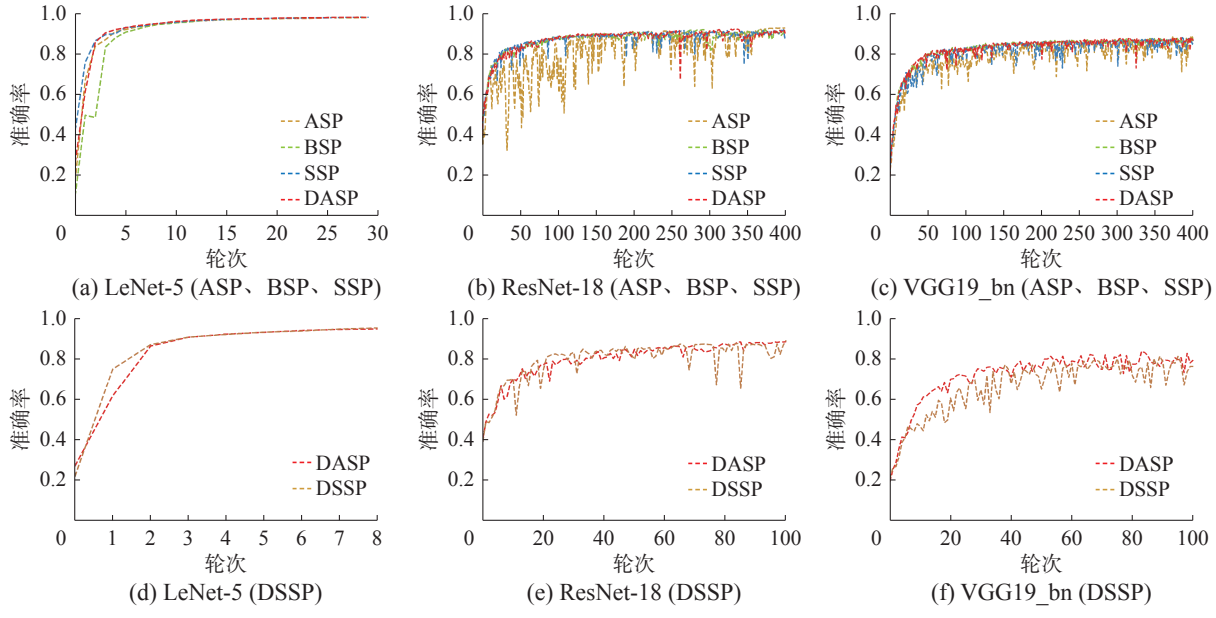

图 4　异构集群下不同并行方法 LeNet-5、ResNet-18、VGG19_bn 测试集准确率

Fig. 4　Accuracy of test sets of LeNet-5, ResNet-18, VGG19_bn with different parallel methods under heterogeneous clusters

为了反映在 DASP 下 stragglers 对快速节点影响情况, 统计了不同集群中模型平均迭代时间。平均迭代时间越低表示在参数服务器进行参数聚合时, 集群中 stragglers 对快速节点的影响更小。如表 4 与表 5 所示, BSP 每次迭代都需要进行节点之间参数同步, 所以平均迭代时间在两种集群下都远高于其他并行方法, 而 ASP 由于节点之间异步迭代训练, 所以平均迭代时间最低。总体来看, 对于 LeNet-5、ResNet-18 与 VGG 19_BN 3 种模型, 在同构环境下, DASP 对比 BSP 分别减少了 16.9%、69.2% 与 39.5% 的迭代时间, 对比 SSP 分别减少了 1.5%、32.1% 与 7.9% 的迭代时间, 对比 DSSP 分别减少了 4.7%、7.0% 与 7.6% 的迭代时间。在异构集群下, DASP 对比 BSP 分别减少了 19.3%、82.1% 与 66.5% 的迭代时间, 对比 SSP 分别减少了 5.3%、41.4% 与 7.2% 的迭代时间, 对比 DSSP 分别减少 3.2%、1.5% 与 3.2% 的迭代时间。同时, DASP 受集群中节点性能差异的影响最小。这是因为 DASP 是基于节点参数版本来管理节点的同步状态, 所以在模型训练过程中减少快速节点不必要的等待, 从而减少了迭代时间, 进而提高集群资源的利用率。

表 4　同构集群下不同并行方法平均迭代完成时间

Table 4　Average iteration completion time for different parallel methods under homogeneous clusters

并行策略	LeNet-5/ms	ResNet-18/s	VGG19_BN/s
BSP	77.11	3.70	4.99
ASP	62.93	0.99	2.87
SSP	65.21	1.68	3.28
DSSP	67.28	1.22	3.25
DASP	64.34	1.14	3.02

表 5　异构集群下不同并行方法平均迭代完成时间

Table 5　Average iteration completion time of different parallel methods under heterogeneous clusters

并行策略	LeNet-5/ms	ResNet-18/s	VGG19_BN/s
BSP	87.78	7.44	9.27
ASP	62.93	1.02	2.84
SSP	74.60	2.23	3.35
DSSP	73.26	1.35	3.21
DASP	70.96	1.33	3.11

最后统计了在 BSP、ASP、SSP、DSSP 与 DASP 并行方法下模型到达相同准确率所需时间, 该时间为迭代平均完成时间与迭代次数的乘积。ASP 虽然平均迭代时间短, 但是由于陈旧参数的影响而导致收敛需要更多轮次。如表 6 与表 7 所示, DASP 在同构与异构集群下都具有良好的收敛时间。总体来看, 对于 LeNet-5 模型, DASP 与其他并行方法都能快速收敛, 但在同构集群下, DASP 并没有表现出良好的收敛效果, 其主要原因是该模型迭代时间非常低以至于节点之间的等待时间不足以覆盖状态调整时所需的额外开销。对于 ResNet-18 与 VGG19_BN 复杂度较高的模型, 在同构集群下, DASP 对比 BSP 收敛时间减少了 10%、24.4%, 对比 ASP 减少了 23.5%、10%, 相比 SSP 减少了 5.9%、1.8%,相比 DSSP 减少了 3.5%、4.9%。在异构集群下, DASP 对比 BSP 收敛时间减少了 61.2%、22.3%, 对比 ASP 减少了 42.7%、29.8%, 相比 SSP 减少了 16.3%、16.9%, 相比 DSSP 减少了 14.8%、10.1%。

表 6　同构集群下不同并行方法收敛时间

Table 6　Convergence time of different parallel strategies under homogeneous clusters

并行策略	LeNet-5/s	ResNet-18/h	VGG19_BN/h
BSP	60.45	4.98	13.80
ASP	62.93	5.86	11.59
SSP	61.23	4.76	10.62
DSSP	71.92	4.64	10.94
DASP	70.96	4.48	10.43

表 7　异构集群下不同并行方法收敛时间

Table 7　Convergence time of different parallel strategies under heterogeneous clusters

并行策略	LeNet-5/s	ResNet-18/h	VGG19_BN/h
BSP	96.46	10.56	29.10
ASP	59.35	7.13	32.20
SSP	70.47	4.90	27.20
DSSP	64.35	4.71	24.89
DASP	58.56	4.10	22.60

从实验结果得出, DASP 相比主流并行方法更有效地缓解 stragglers 与陈旧梯度问题。最终, 提高了资源利用率并降低了收敛时间。

4　结束语

针对异构集群下分布式机器学习参数同步并行策略的问题, 提出了一种基于节点状态的动态自适应同步并行方法 DASP。DASP 并行方法通过工作节点训练状态信息管理使参数服务器可以更精准标识 stragglers 与产生陈旧梯度的节点, 解决了异步并行方法的问题。针对同步等待问题, 通过在运行时分析工作节点的状态信息为每个节点实现不同的并行范例, 提高 stragglers 的更新效

率并消除旧参数来避免陈旧梯度的产生，以此来缓解它们的影响。最后，通过使用典型的深度学习模型和数据集分别在同构集群与异构集群下开展了实验。实验表明，在将模型收敛到相同测试准确率的情况下，DASP 相比与基准方法收敛时间减少了 16.9%~82.1%。

后续工作将继续探索如何更有效地利用节点状态信息对并行状态阈值进行动态设定，从而进一步降低同步代价。

参考文献：

[1] SZEGEDY C, LIU Wei, JIA Yangqing, et al. Going deeper with convolutions[C]//2015 IEEE Conference on Computer Vision and Pattern Recognition. Boston: IEEE, 2015: 1−9.

[2] SZEGEDY C, TOSHEV A, ERHAN D. Deep neural networks for object detection[C]//Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2. New York: ACM, 2013: 2553−2561.

[3] 叶正喆, 苍岩. 基于卷积神经网络的行人检测方法 [J]. 应用科技, 2022, 49(2): 55–62.
YE Zhengzhe, CANG Yan. A pedestrian detection method based on convolutional neural network[J]. Applied science and technology, 2022, 49(2): 55–62.

[4] KENTON J D M W C, TOUTANOVA L K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding[C]//Proceedings of NAACL-HLT. Minnesota: Association for Computational Linguistics, 2019: 4171−4186.

[5] 窦勇敢, 袁晓彤. 基于隐式随机梯度下降优化的联邦学习 [J]. 智能系统学报, 2022, 17(3): 488–495.
DOU Yonggan, YUAN Xiaotong. Federated learning with implicit stochastic gradient descent optimization[J]. CAAI transactions on intelligent systems, 2022, 17(3): 488–495.

[6] PASZKE A, GROSS S, MASSA F, et al. PyTorch: an imperative style, high-performance deep learning library[C]// Proceedings of the 33rd International Conference on Neural Information Processing Systems. New York: Curran Associates Inc, 2019: 8026−8037.

[7] ABADI M, BARHAM P, CHEN Jianmin, et al. TensorFlow: a system for large-scale machine learning[C]//Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation. New York: ACM, 2016: 265−283.

[8] 曹嵘晖, 唐卓, 左知微, 等. 面向机器学习的分布式并行计算关键技术及应用 [J]. 智能系统学报, 2021, 16(5): 919–930.
CAO Ronghui, TANG Zhuo, ZUO Zhiwei, et al. Key technologies and applications of distributed parallel computing for machine learning[J]. CAAI transactions on intelligent systems, 2021, 16(5): 919–930.

[9] 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 Shuai, LI Dan. Research progress on network performance optimization of distributed machine learning system[J]. Chinese journal of computers, 2022, 45(7): 1384–1411.

[10] MIAO Xupeng, NIE Xiaonan, SHAO Yingxia, et al. Heterogeneity-aware distributed machine learning training via partial reduce[C]//Proceedings of the 2021 International Conference on Management of Data. New York: ACM, 2021: 2262−2270.

[11] 舒娜, 刘波, 林伟伟, 等. 分布式机器学习平台与算法综述 [J]. 计算机科学, 2019, 46(3): 9–18.
SHU Na, LIU Bo, LIN Weiwei, et al. Survey of distributed machine learning platforms and algorithms[J]. Computer science, 2019, 46(3): 9–18.

[12] FAN Wenfei, HE Kun, LI Qian, et al. Graph algorithms: parallelization and scalability[J]. Science China information sciences, 2020, 63(10): 203101.

[13] JIANG Jiawei, CUI Bin, ZHANG Ce, et al. Heterogeneity-aware distributed parameter servers[C]//Proceedings of the 2017 ACM International Conference on Management of Data. New York: ACM, 2017: 463−478.

[14] 朱泓睿, 元国军, 姚成吉, 等. 分布式深度学习训练网络综述 [J]. 计算机研究与发展, 2021, 58(1): 98–115.
ZHU Hongrui, YUAN Guojun, YAO Chengji, et al. Survey on network of distributed deep learning training[J]. Journal of computer research and development, 2021, 58(1): 98–115.

[15] XU Ning, CUI Bin, CHEN Lei, et al. Heterogeneous environment aware streaming graph partitioning[J]. IEEE transactions on knowledge and data engineering, 2015, 27(6): 1560–1572.

[16] HO Q, CIPAR J, CUI Henggang, et al. More effective distributed ML via a stale synchronous parallel parameter server[J]. Advances in neural information processing systems, 2013, 2013: 1223–1231.

[17] LI M, ANDERSEN D G, SMOLA A, et al. Communication efficient distributed machine learning with the parameter server[C]//Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 1. Massachusetts: MIT press, 2014: 19−27.

[18] ZHAO Xing, AN Aijun, LIU Junfeng, et al. Dynamic stale synchronous parallel distributed training for deep learning[C]//2019 IEEE 39th International Conference on

Distributed Computing Systems. Dallas: IEEE, 2019: 1507−1517.

[19] FAN Wenfei, LU Ping, YU Wenyuan, et al. Adaptive asynchronous parallelization of graph algorithms[J]. ACM transactions on database systems, 2020, 45(2): 1–45.

[20] 王恩东, 闫瑞栋, 郭振华, 等. 分布式训练系统及其优化算法综述 [J/OL]. 计算机学报, 2023: 1−29. (2023−04−06) [2023−05−01]. https://kns.cnki.net/kcms/detail/11.1826.tp.20230404.1510.002.html.
WANG Endong, YAN Ruidong, GUO Zhenhua, et al. A survey of distributed training system and its optimization algorithms[J/OL]. Chinese journal of computers, 2023: 1−29. (2023−04−06)[2023−05−01]. https://kns.cnki.net/kcms/detail/11.1826.tp.20230404.1510.002.html.

[21] CHEN Jianmin, PAN Xinghao, MONGA R, et al. Revisiting distributed synchronous SGD[EB/OL]. (2017−03−21)[2022−07−11]. https://arxiv.org/abs/1604.00981.

[22] TENG M, WOOD F. Bayesian distributed stochastic gradient descent[C]//Proceedings of the 32nd International Conference on Neural Information Processing Systems. New York: ACM, 2018: 6380−6390.

[23] SUN Haifeng, GUI Zhiyi, GUO Song, et al. GSSP: eliminating stragglers through grouping synchronous for distributed deep learning in heterogeneous cluster[J]. IEEE transactions on cloud computing, 2022, 10(4): 2637–2648.

[24] HARLAP A, CUI Henggang, DAI Wei, et al. Addressing the straggler problem for iterative convergent parallel ML[C]//Proceedings of the Seventh ACM Symposium on Cloud Computing. New York: ACM, 2016: 98−111.

[25] XU Hongfei, VAN GENABITH J, XIONG Deyi, et al. Dynamically adjusting transformer batch size by monitoring gradient direction change[C]//Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Stroudsburg: Association for Computational Linguistics, 2020: 3519−3524.

[26] HE Kaiming, ZHANG Xiangyu, REN Shaoqing, et al. Deep residual learning for image recognition[C]//2016 IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas: IEEE, 2016: 770−778.

[27] SIMONYAN K, ZISSERMAN A. Very deep convolutional networks for large-scale image recognition[EB/OL]. (2015−04−10)[2022−07−11]. https://arxiv.org/abs/1409.1556.

作者简介:

马翔, 硕士, 主要研究方向为分布式机器学习、深度学习。

申国伟, 教授, 博士, 主要研究方向为大数据及网络安全、分布式机器学习。主持国家自然科学基金项目 2 项, 主持产学研各类项目 10 余项, 申请专利 10 余项, 主持产学研各类项目发表学术论文 30 余篇。

郭春, 副教授, 博士, 主要研究方向为网络安全、入侵检测。