arXiv:2301.02959v1 [cs.LG] 8 Jan 2023

FLEXSHARD：面向工业级序列推荐模型的灵活分片

Geet Sethi [1 2] **Pallab Bhattacharya** [2] **Dhruv Choudhary** [2] **Carole-Jean Wu** [2] **Christos Kozyrakis** [1]

摘要

基于序列的深度学习推荐模型（DLRM）是一类新兴的深度学习推荐模型，与以往基于求和池化的模型相比，在捕捉用户长期兴趣方面取得了显著提升。然而，这些提升带来了巨大的系统成本：在单次迭代中，每个加速器都需要动态物化并传输大量数据。为解决这一日益严重的瓶颈，我们提出了*FlexShard*，一种新型分层序列嵌入表分片算法，它以逐行粒度 运行，并利用了并非每一行都相同这一洞见。FlexShard 根据嵌入行底层的概率分布进行精确复制，并引入一种适用于真实集群网络拓扑异构且性能倾斜特点的新分片策略，从而在不增加额外内存 的情况下，大幅降低通信需求，且性能优于此前最先进的方法。在生产规模序列 DLRM 上进行评估时，FlexShard 将全局全互换通信流量总体降低了超过 *85%*，使端到端训练通信延迟较此前最先进的方法提升近 *6* 倍 。

1 引言

过去十年间，深度学习（DL）的应用已无处不在。深度学习应用涵盖众多领域（Deng 等，2009年；Devlin 等，2019年；Hazelwood 等，2018年；He 等，2016年；Jumper 等，2021年；Naumov 等，2020年；Vaswani 等，2017年；Weyn 等，2020年），并成为许多互联网规模服务的支柱，例如网页搜索、社交媒体和视频流媒体（Cheng 等，2016年；Covington 等，2016年；Gomez-Uribe 和 Hunt，2016年；Gupta 等，2020年b；Jouppi 等，2017年；Raimond，2018年；Sullivan，2016年；Zhao 等，2019年；Zhou 等，2019年a；Wu 等，2021年）。支撑这些服务的一类重要深度学习应用是推荐系统。现代推荐系统基于深度学习推荐模型，为用户个性化定制内容，从而提升体验质量。深度学习推荐模型（DLRM）在工业数据中心使用的人工智能计算周期中占据了显著且不断增长的比例。例如，在2019年和2020年，Facebook 超过50%的人工智能训练用量和超过80%的人工智能推理用量都用于 DLRM（Gupta 等，2020年b；Acun 等，2021年）。与用于图像识别或语言翻译等任务的深度学习架构不同，后者主要由通用矩阵乘法（GEMM）层组成，每个参数的计算强度较高；而构成 DLRM 的大多数参数都包含在嵌入层（Naumov 等人，2020；Anil 等人，2022）。嵌入层实际上是大规模并行查找表，具有很高的内存强度（Gupta 等人，2020b；Ke 等人，2020）。近期研究表明，2017年至2021年间，DLRM 嵌入表的内存存储需求增长了16倍以上；最新模型包含数万亿个参数，代表数千亿个嵌入，需要数太字节（TB）的存储空间来存放模型权重。此外，同一时期 DLRM 的内存带宽需求增长了近30倍，单个训练批次就需要读取数太字节的数据（Mudigere 等人，2021；Sethi 等人，2022）。

在这项工作中，我们提出了一种新方法来解决分片新兴序列推荐深度学习推荐模型的问题。为了实现超大规模深度学习推荐模型所需的训练吞吐量，这些模型的训练数据量达拍字节级（Mudigere 等人，2021年；Zhao 等人，2022年），人们采用了配备高带宽内存（HBM）的 GPU 来加速深度学习推荐模型的训练。然而，最新的 GPU 所配备的 HBM 容量仅为数十 GB，比训练这些模型所需的总内存占用低几个数量级，因此系统必须扩展至分布式多 GPU 训练

[1]美国加利福尼亚州斯坦福的斯坦福大学 [2]美国加利福尼亚州门洛帕克的 Meta。通信请联系：Geet Sethi<geet@cs.stanford.edu>。

版权所有 © 2022年，作者所有。

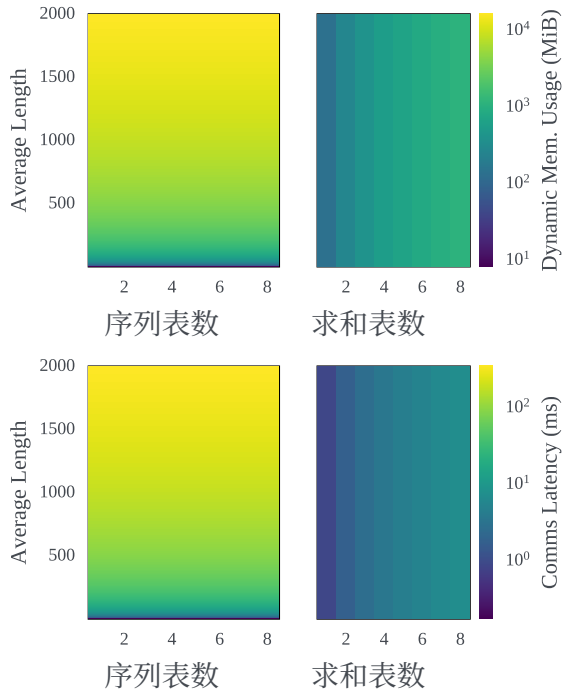


图*1*。对于一个包含最多 8 个用户历史嵌入表的 DLRM，随着总历史长度从 1 变化至 2000，预期的每 GPU 内存和通信开销；其中，特征以序列形式（左列）或求和池化形式（右列）使用。最先进的 TorchRec（Ivchenko 等人，2022年）框架在包含 32 个 GPU 的 4个节点系统上按行分片，批量大小为 4096，嵌入维度为 256，类型为 FP32。求和池化嵌入表的内存和通信开销随表数增加而扩展，不受特征长度影响；而序列嵌入表的开销随特征长度线性扩展，而不随表数增加。

以实现所需的吞吐量，并由此引出了分片问题。分片问题旨在对嵌入参数进行分区，并将其部署到系统拓扑中的不同位置，以最大限度地提高资源利用率和效率。相比于采用深度学习框架中经过人工调优或自动生成的数据并行、模型并行和/或流水线并行组合（Abadi 等人，2015年；Li 等人，2020年）来解决问题，分片问题更难之处在于对嵌入参数的访问具有不规则性且依赖于输入数据。

此外，尽管近年来由于分片问题的重要性，相关研究日益增多（Adnan 等人，2021年；Lui 等人，2021年；Sethi 等人，2022年；Zha 等人，2022年），据我们所知，这些研究都假定待分片的嵌入表要么是独热——即每个训练样本至多访问每张表中的一行嵌入——要么是求和池化——即训练样本访问的表中所有嵌入行都会先通过求和进行聚合，再输入模型。

这一基本假设会显著影响模型训练所需的动态（即运行时）内存用量和互连通信带宽。此外，这一假设并不适用于 Transformer 等最先进的新兴学习方法；在这些方法中，嵌入以序列的形式输入，从而大幅提高推荐准确率（Chen 等人，2019年；Kang 和 McAuley，2018年；Li 等人，2019年；Pi 等人，2019年；Sheng 等人，2021年；Sun 等人，2019年）。图1 展示了基于序列的嵌入所带来的内存和通信开销迅速增长，其数量级远高于对同一组嵌入进行求和池化。

为解决日益严重的训练系统瓶颈，并支持大规模下一代基于序列的 DLRM，我们提出*FlexShard*——一种新型分层分片方法，可按逐嵌入行粒度对序列 DLRM 嵌入表进行分区和放置。FlexShard 基于这样一个洞见：并非每一行都相同，不同嵌入行的系统需求各异；与现有最先进的业界序列分片策略相比，谨慎、精确地复制某些行，能够同时节省时间和内存。此外，FlexShard 利用真实硬件网络拓扑往往具有异构性这一特点，即加速器之间的带宽并不完全相同，从而进一步提升通信性能。总体而言，本工作的主要贡献如下：

- 我们构建了一个分析框架，用于对现有序列嵌入分片策略的内存和通信扩展进行建模，并展示了通过更细粒度的分片节省内存和通信开销的潜力。

- 我们提出了一种新的分片策略——*Flex*——可适应真实网络互连（即节点内与节点间带宽）和通信集合操作（即全归约与全互换）性能的不均衡。

- 我们提出了 FlexShard——一种新的多层序列分片算法，可按精细的逐行粒度复制和分片嵌入表。

- 在生产规模的序列深度学习推荐模型上进行评估，结果表明，与基线最先进方法（*Ivchenko* 等人，2022年）相比，FlexShard 可将全局全互换通信流量减少超过 85%，使端到端训练通信延迟几乎降低至六分之一。此外，FlexShard 还能降低峰值内存利用率。

2 背景

深度学习推荐模型。 图2 概述了广义深度学习推荐模型架构，该架构由四个主要组件构成：

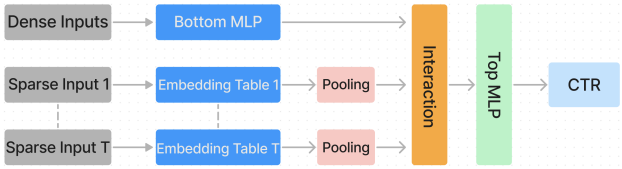


图 2. 典型的 DLRM 架构。

嵌入层、池化函数/层、多层感知机以及交互函数/层（Naumov 等人，2019年）。

深度学习推荐模型通过预测用户与特定内容互动的概率（通常称为点击率（CTR）），解决推荐问题并提供个性化推荐。为生成这些预测，深度学习推荐模型以两类代表用户和内容的特征作为输入：稠密和稀疏。稠密特征表示连续的输入信息，由底部多层感知机层处理；稀疏特征表示离散的类别数据，并以输入 ID 的形式传递给嵌入层。嵌入层的作用是将这些离散 ID 转换为经过学习的稠密连续值嵌入向量，这些向量编码了每个 ID 的潜在表示，供模型的其余部分使用。

对于工业规模的推荐系统，稀疏特征的基数达到数亿乃至数十亿并不少见，其中每个不同的 ID 都代表一个实体，例如 YouTube 上的单个视频。由于规模庞大，用于编码这些特征的嵌入表可能各自需要数十至数百 GB 的存储空间（Kang 等人，2021年；Mudigere 等人，2021年；Sethi 等人，2022年；Zhao 等人，2020年）。此外，单个类别特征通常会同时编码每位用户的多种信息，例如用户最近观看、点赞、分享等内容的 ID 历史记录；其中每一种信息都可能构成独立的特征。由于多层感知机等稠密神经网络（NN）层需要固定大小的张量作为输入，因此必须以某种方式聚合并缩减不同用户长度不一的历史记录（每条记录都汇集数量可变的嵌入向量作为输入），将其转换为固定大小的输出嵌入，这通常称为池化嵌入。池化后，各种嵌入输出与底部多层感知机层的输出一起，通过点积等函数进行特征交互，随后经过顶部多层感知机层，最终预测点击率。

深度学习推荐模型分片。 由于深度学习推荐模型通常包含多个嵌入表，所需存储容量超过了任何单个 GPU 的可用内存，因此训练深度学习推荐模型需要采用模型并行，并解决分片问题。 分片问题的目标是生成均衡且高效的在系统拓扑中对嵌入表进行划分和放置，使得每个 GPU 在每个与嵌入相关阶段的总执行时间尽可能接近。这种均衡对于超大规模 DLRM 训练至关重要，因为此类 DLRM 通常以同步方式进行训练：在一个训练批次的前向传播过程中，整个系统中每个模型权重的所有副本都完全一致；每次迭代结束时，梯度更新都会在全局范围内同步，然后才开始下一次迭代（Mudigere 等人， 2021年； Sethi 等人，2022年）。

与简单地对所有嵌入的权重进行静态分区相比，分片问题更具挑战性的原因在于计算与通信的平衡。每个嵌入表的计算需求各不相同，取决于其输入 ID（这些 ID 具有不规则性且依赖于数据）以及其池化函数。此外，模型并行需要在 GPU 之间通信和分发嵌入，这可能需要针对每个批次进行，以满足每个 GPU 的局部数据依赖。

嵌入池化。 一种简单且常见的池化方法是对收集到的所有嵌入逐元素求和，以生成池化后的输出嵌入；这种方法称为求和池化（Gupta 等人，2020a；Ke 等人，2020年；Kwon 与 Rhu，2022年；Mudigere 等人，2021年；Sethi 等人，2022年；Zha 等人，2022年；Zhou 等人，2019b；Wilkening 等人，2021年）。使用求和池化进行嵌入聚合，在分片问题上具有一些优势。首先，在现代硬件上，求和运算的每字节计算强度很低，因此计算负载的平衡可简化为内存带宽需求的平衡（即同时访问的 ID 数量）。其次，求和是一种可交换、可结合的运算，因此可以任意对嵌入进行分片，并在局部进行部分聚合，而不影响正确性。第三，每个嵌入表分片的池化输出始终与单个嵌入行大小相同，并且可以原地生成，无需将所有相关行中间物化。第四，求和支持使用高性能集合通信操作（如归约散布），在分发嵌入的同时对部分聚合的输出进行池化。

然而，求和忽略了用户历史记录和行为的序列性，并在某个时间范围内对用户的所有交互赋予相同权重。例如，对于 Netflix 和 YouTube 等视频流媒体服务，直觉上，用户按时间排序的历史记录对其接下来可能想要参与的内容具有不可忽视的重要性。因此，类似于自然语言处理（NLP）方法的发展历程——从词袋方法演进到序列式、基于句子的方法——近期研究表明，将用户的所有嵌入输入学习得到的 聚合方法，

例如成对自注意力和 Transformer，可以显著提升推荐效果（Chen 等人，2019年；Kang 和 McAuley，2018年；Li 等人，2019年；Pi 等人，2019年；Sheng 等人，2021年；Sun 等人，2019年）。

据我们所知，然而，此前针对 DLRM 分片问题提出创新解决方案的研究，均假设所收集的嵌入在被后续模型层使用之前都会进行求和池化（Adnan 等人，2021年；Lui 等人，2021年；Sethi 等人，2022年；Zha 等人，2022年）。遗憾的是，这些优势并不能直接适用于基于序列的池化方法，因为在这类方法中，序列中的每个嵌入都会与其他所有嵌入进行交互，因此必须将完整序列置于请求方 GPU 上才能执行该操作。因此，每个嵌入都必须在发送方 GPU 上实例化，以便进行通信并发送至请求方 GPU，这要求内存使用量随发送方 GPU 上的序列长度动态变化动态内存使用量与发送方 GPU 上的序列长度成正比。此外，序列嵌入无法在传输过程中进行归约，因此需要使用与请求方 GPU 上的序列长度相等的 动态 内存，并且还必须使用速度更慢的全互换通信集合操作。动态内存，其大小等于请求方 GPU 上的序列长度，同时还需要使用速度较慢的全互换通信集合操作。

此外，缺乏求和池化固有的优势，带来了此前研究未曾讨论的问题，且超出了它们为嵌入表探索过的分片维度：静态内存使用、查找时间和通信时间。这个问题涉及动态内存和通信使用；对于大型互联网公司当前常见的、轻易生成的序列长度（例如与过去 30 天内服务过的最近 500 个项目的交互），动态内存使用可能远远超过每个图形处理器上对应静态嵌入表分片的大小。因此，除了此前研究采用的性能维度外，还必须考虑动态内存利用率，才能生成高效且性能良好的序列嵌入分片方案。

嵌入访问模式。除了嵌入表之间大小和平均计算需求（即平均池化大小不同）各异之外，嵌入层还有另一项重要且独特的属性，使其不仅区别于其他神经网络层，也彼此不同：它们的稀疏、不规则访问模式（Adnan 等人，2021年；Kwon 与 Rhu，2022年；Sethi 等人，2022年；Wilkening 等人，2021年）。单个数据样本中访问的嵌入行数量比表中的总行数少多个数量级；而被访问的行通常呈幂律分布，少数行占据了绝大多数访问量。尽管长尾部分很少被访问，但由于其对推荐质量和用户体验至关重要，因此不可忽视（Zhao 等人，2020年）。这种访问分布偏斜带来了有趣的系统级优化机会。

3 黑盒分片

目前，主流分片算法将序列嵌入表视为整体的黑盒单元（Ivchenko 等人，2022年；Mudigere 等人，2021年），这些算法利用整个表和／或特征级的信息，例如特征的平均长度（即池化大小）。这种粒度的信息无法提供有关构成每个表的行之特征的细致信息，因此这些算法所采用的策略不得不将每个表内的内存访问模式视为随机。

3.1 均匀策略

一种利用这一层级信息对表进行分片的策略，是在所有 rank 上均匀分片，例如采用按列（CW）和按行（RW）策略，通过模型并行（MP）将表划分为互不重叠的分区；或者采用数据并行（DP），复制整张表。CW 通过按嵌入维度（即列）拆分嵌入表来进行分片，使访问负载均衡，因为每次查找都会均匀分布到所有分片上。一个不那么直观、可能令人惊讶的结果是，按行均匀分片表——也就是将表拆分为大小相等的连续行块——也能大致均匀地分配查找负载，使访问频繁的行分散到所有分片上。这一良好结果源于原始内容 ID（通常是任意分配的 64 位整数）会通过整数哈希函数映射到相应的嵌入 ID，而这些函数会将键均匀分布到哈希空间（即嵌入表大小）中。采用 DP 时，嵌入表会复制到所有 GPU 上，从而均衡负载，因为每个 GPU 都会对其局部批次中的所有内容执行查找。

当均匀分片到所有 *GPU*时，对于所有均匀策略而言，每个 *GPU* 在全局批次中访问的行数所对应的内存访问负载，其期望值相同。不过，正如我们将在下一节探讨的那样，对于内存占用、通信负载和输入数据分布等其他重要属性，不同策略之间可能存在很大差异。

3.2 黑盒性能建模

为了正确理解现有黑盒均匀策略的扩展特性，并为制定性能更优的策略奠定基础，我们首先针对单个嵌入表在单次训练迭代中各黑盒策略下每个 *GPU* 的预期工作量，建立解析模型，并采用以下 6 种不同指标进行分析：

1. 静态内存成本（字节）：所需的内存

参数	描述
E	行数
D	嵌入维度
$_{\text{数据类型}}$	嵌入标量的数据类型
L	平均特征长度
W	一个节点中的 GPU 数量
N	节点数量
U	GPU 总数（$= N * W$）
B	每个 GPU 的本地批量大小
$a2a_{\text{全局}}$	全局全对全带宽
$_{\text{全互换}_{intra}}$	节点内全对全带宽
$ar_{\text{全局}}$	全局全归约带宽
ar_{cross}	跨节点全归约带宽

表*1.* 分片扩展方程中使用的参数。

分片类型	静态内存	访问的行	动态内存
RW	$\frac{E}{U} * D * dtype$	$U * B * \frac{L}{U} * D$	$2 * B * L * D * dtype$
CW	$E * \frac{D}{U} * dtype$	$U * B * L * \frac{D}{U}$	$2 * B * L * D * dtype$
DP	$6 * E * D * dtype$	$B * L * D$	$B * L * D * dtype$
Flex	$6 * \frac{E}{W} * D * dtype$	$W * B * \frac{L}{W} * D$	$2 * B * L * D * dtype$

表*2.* 不同分片策略下，基于序列的嵌入表每GPU的预期内存开销。注意，RW、CW、DP 和 Flex 分别代表按行、按列、数据并行和我们提出的设计。

存储嵌入表行的分片

2. 输入数据分布成本：查找开始前预期需要通信的 ID 数量 3. 访问的行：预期从本地嵌入表分片中查找的行数 4. 动态内存成本（字节）：用于存储物化查找值和通信缓冲区的预期运行时内存使用量；在前向传播和反向传播中都会产生 5. 动态通信成本（延迟）：根据预期与图形处理器之间通信的最大字节数和互连性能计算的预期动态通信延迟；在前向传播和反向传播中都会产生 6. 静态通信成本（延迟）：无论批次和特征属性如何都会产生的通信延迟；仅在反向传播中产生

分片类型	输入数据分布	访问的行	动态通信	静态通信
RW	$B * L$	$U * B * \frac{L}{U} * D$	$\frac{B*L*D*dtype}{a2a_global}$	N/A
CW	$U * B * L$	$U * B * L * \frac{D}{U}$	$\frac{B*L*D*dtype}{a2a_global}$	N/A
DP	N/A	$B * L * D$	N/A	$\frac{E*D*dtype}{ar_global}$
Flex	$B * L$	$W * B * \frac{L}{W} * D$	$\frac{B*L*D*dtype}{a2a_intra}$	$\frac{\frac{E}{W}*D*dtype}{ar_cross}$

表*3*。不同分片策略下，序列型嵌入表的每GPU预期通信开销。

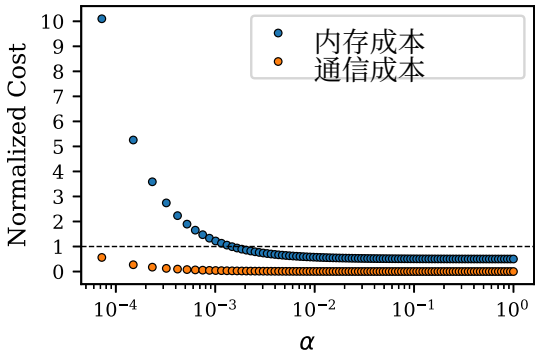


图*3*。在嵌入表大小固定为 E、$L = \alpha E$ 变化的情况下，数据并行相对于行分片的归一化内存和通信开销。目标训练配置为32个 GPU、批量大小为4096、嵌入维度为256，且全局带宽为 $all\ reduce_{bw} = \frac{1}{3} \cdot all\ to\ all_{bw}$。

定义这些指标后，我们可以推导出每种均匀策略在各个轴上的扩展方程，如表2 和 3所示；表2展示内存扩展，表3展示通信扩展，而表 1 总结了方程中使用的参数。

尽管这些策略所采用的黑盒扩展方程是根据粗粒度信息推导得出的——尤其是用于静态开销的表大小 E，以及用于动态开销的特征平均长度 L——但它们仍然为理解不同策略之间的权衡提供了重要见解。

对一对策略之间的权衡进行建模的一种方法，是计算它们扩展方程的比值，在两个策略共享的变量发生变化时，求出一种策略相对于另一种策略的归一化相对开销。均匀策略可根据其处理参数的方式明确分为两组：DP——仅在反向传播中发生基于*all reduce* 的通信；或 MP（即 CW 和 RW）——在前向传播和反向传播中均发生基于*all to all* 的通信。此外，我们可以将 DP 与 MP 策略之间的权衡建模简化为仅对 DP 与 RW 策略之间的权衡进行建模，因为从期望上看，除输入数据分布外，RW 和 CW 在所有指标上都相同；而在输入数据分布方面，CW 所需的通信严格更多。

在图 3 中，我们展示了 DP 相对于 RW 的总归一化内存和通信开销，并固定 E、改变 L，具体如下：

$$L = \alpha \cdot E \tag{1}$$

此外，我们绘制了水平线 $y = 1$，它将每条曲线划分为两个区域：DP 比 RW 更高效的区域（即 $y < 1$），以及效率较低的区域（即 $y > 1$）。由于通信开销取决于目标硬件的拓扑结构（具体而言是其带宽），图 3 采用与我们的实验相同的拓扑结构进行建模，详情见第 5节。

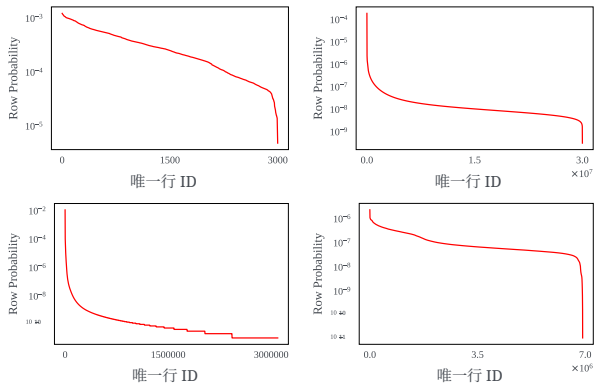


图 *4*。生产环境中若干深度学习推荐模型所用稀疏特征的有序归一化值频率分布。尽管分布形态差异很大，图中所有分布的平均特征长度都相同。这些分布由数十亿个随机抽取的训练数据样本生成。

鉴于现有黑盒算法在表级别进行粗粒度决策，我们强调，在实践中，这些方法几乎从不对序列嵌入表采用数据并行。这是因为序列特征的基数很高，导致嵌入表大小达到数千万行；即使序列特征的平均长度约为数千，也会导致 $\alpha \sim O(10^{-4})$；如图3所示，此时数据并行的内存需求远高于行分片。因此，现有方法无法实现数据并行带来的任何通信优势。

然而，图3 也表明，当平均特征长度与表大小之比足够大（即 α）时，与行分片相比，数据并行既能节省时间，也能节省内存。如前所述，这种情况不太可能在整个表粒度上出现，但可以通过行子集构造出这样的比值。这一发现启发了 FlexShard：通过以逐行粒度而非表粒度进行分片，它能够尝试兼取数据并行与行分片的优势。

4 FLEXSHARD

基于迄今为止的这些发现，我们现在对 *FlexShard* 进行形式化并加以开发。FlexShard 是一种分片算法，它根据底层特征分布，以逐行粒度对序列嵌入表进行最优分片。

4.1 并非所有行都相同

图 4 展示了稀疏特征按值频率归一化排序后的计数——也就是每个唯一特征值的累计出现次数——这些特征来自数十亿条生产训练数据样本。尽管并不完全相同，但它们都遵循相同的总体形状，对应于底层的幂律分布，其中少数 ID 占据了大多数访问量。

此外，尽管每种分布都有独特的形状，它们的 L 相同。因此，每种分布 都将被视为相同，并由黑盒策略以相同方式进行分片。

为了解决这一明显局限，并为利用上一节的见解提供途径，我们提出了一个定义，用于计算特征的 L。据我们所知，这一定义不同于以往及相关研究，它是直接根据特征底层的值分布计算得出的。图4 中分布的每个点都表示某一行（嵌入索引）出现在该特征的随机数据样本中的概率。这等价于该行的二项概率 p，其在随机样本中出现的期望值为 $E[X_i] = p$。基于这一表述，我们将值分布定义为所有 E——即行数——二项随机变量的集合。

根据此定义，我们可以计算该特征的随机数据样本中预期包含的行数

$$E[X] = \sum_{i=1}^{E} p_i = L \tag{2}$$

这是基于期望的线性性。请注意，我们不对行之间是否独立作任何假设。换言之，我们不对特定行同时出现时是否存在相关性作任何假设。这是因为期望的线性性~~不要求相关~~随机变量之间相互独立。

以这种方式计算 L，我们可以获得两个重要见解：首先，我们可以根据各行的单行概率，计算任意行子集的 L；其次，并非每一行的权重都必然相同，也就是说，并非所有行都相等。

4.2 分离成本

有了这一见解，现在让我们重新审视第3.2节中的扩展方程和归一化成本模型。将L 替换为公式 2 中基于单行的求和形式后，我们可以将表的总成本分解为各行单行成本之和，其中每行的 L 就是该行的单行概率。这样一来，我们还可以将 $E = 1$代入公式 1，得到图 3，根据行的概率对单行的数据并行复制成本与行分片成本进行建模。

此外，求解每条曲线上与直线 $y = 1$ 的交点，可得到两个 alpha 值，或断点，它们确定了在这些行概率下，数据并行与行分片在内存或通信方面达到均衡 。因此，它们也确定了概率空间中的区域：在这些区域内，就内存成本、通信成本或两者而言，对一行进行数据并行处理比行分片更高效。

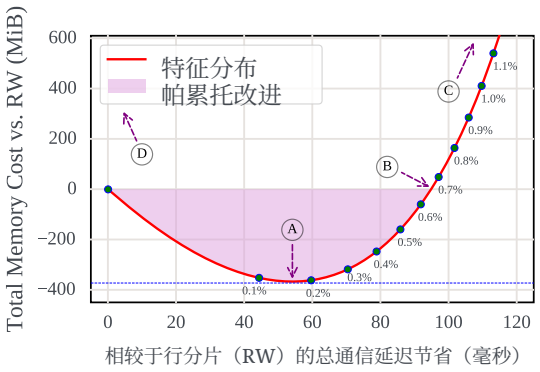


图 *5.* 将生产环境中的序列嵌入表中越来越多按频率降序排列的行采用数据并行（DP）复制，而非行分片（RW）后，预期的内存和通信开销。

4.3 逐行分片

如何利用这一点进行分片？由于断点将一张表的值分布划分为不同区域，如果我们按频率顺序遍历表的值分布，那么对于所有断点，接下来遇到的行始终是该断点下剩余行中边际效率最高的行（正值最大，负值最小）。

我们在图5中展示了这一点。图中，随着我们按频率降序遍历序列嵌入表值分布中不断增加的行数（展示了前 1.1% 的行，并以 0.1% 为增量标记），绘出了复制某一行并采用数据并行，与对其进行行分片相比所产生的累积边际内存成本（纵轴）和通信成本（横轴）。该图直观呈现了在对其余行进行行分片的同时，对特定子集的行采用数据并行复制所产生的边际内存成本与通信成本。沿着曲线，我们根据断点及其底层分布确定了四个重要的关注点。

第一个点A 位于曲线的最低点，代表能够带来最大内存节省的分片方案，以及相应的通信节省。当降序排列的值分布中下一行的概率低于内存中性断点概率时，就会出现这一点。从此以后，所有剩余行相对于行分片都会产生边际内存成本。

第二个点 B 位于 x 轴截距处，表示累积边际内存成本降至零的时刻。这个点确定了内存中性分片方案，该方案能够在不增加内存成本的情况下实现最大的通信节省。识别这种分片设计十分重要，因为它代表了对于可能已采用 RW 等基线策略运行的现有模型和训练拓扑而言，可直接采用的最大吞吐量提升。此外，由于目标训练拓扑在分片过程中保持固定，且我们不希望高带宽内存利用不足，因此本文其余部分及评估均以此为目标分片设计，并将其称为*FlexShard* 两级。

第三个点 C 出现在达到通信中性断点时。这标志着能够产生最大通信节省的分片；此后，与 RW 相比，所有行都会产生边际通信成本。最后一个点 D 则表示相对于按 RW 分片，按 DP 复制整个表所产生的边际内存和通信总成本。

虽然这些点根据断点和底层分布清楚地标示出值得关注的潜在分片设计，但我们要强调，曲线上的每个点都代表一种潜在分片，而曲线本身是逐行采用 DP 与 RW 分片时内存与通信权衡的帕累托前沿。因此，对于任何指定的边际内存总成本，只要存在相应方案，FlexShard 都能输出通信延迟最低的最优混合 DP/RW 分片。

4.4 适配现实世界硬件拓扑

迄今提出的两层 FlexShard 设计建立在黑盒算法采用的相同均匀策略之上：DP 和 RW。这些策略的一个固有特性是全局均匀，也就是说，所有 *GPU* 执行的工作以及彼此之间的工作量实际上完全相同。虽然全局均匀性更易于分析，却忽略了现实世界硬件拓扑往往具有异构的分层结构这一事实。此类工业级和云规模训练系统包括 Meta 的 ZionEX 节点（Facebook，2021年；Mudigere 等人，2021年）、Azure 的 NDv4 虚拟机（Microsoft，2022年）、AWS 的 EC2 P4d 实例（Amazon，2022年）以及 NVIDIA 的 DGX 节点（NVIDIA，2022b）。这些系统均通过 NVLink/NVSwitch 互连实现了更高的节点内带宽；相比之下，其基于 RoCE 和 InfiniBand 等技术的节点间互连则带宽较低，采用的是 RDMA 节点间互连。

回顾迄今为止的经验，我们开发了一种新的分片策略——*Flex*——利用 FlexShard 根据底层分布以逐行粒度进行分片的能力，同时适应异构网络拓扑中不均衡的通信带宽。

为平衡每行概率相对于内存中性断点进一步降低时，按行进行数据并行所导致的内存成本呈指数级增长，同时利用更高的节点内带宽而非节点间带宽，Flex 策略会在节点内对一组行进行行分片，然后在各节点间对每个分片进行数据并行复制。这样一来，Flex 策略既能利用跨节点数据并行的通信效率（此时带宽利用率的成本要高得多），又能同时利用行分片

利用可用的更高节点内带宽，降低受容量限制的每 GPU 内存占用（见表2 和3）。

4.5 跨三层分片

为了利用 Flex 策略，我们将此前介绍的两层算法更新为 *FlexShard* 三层，具体如下。首先，按频率降序排列，从头开始将所有行复制到所有 GPU 上进行数据并行，直至内存中性断点（A，见图 5），从而节省内存和通信开销。接下来，将各行按 Flex 方式分片，直到耗尽所有由此产生的内存节省（（B，见图5），或达到 Flex 与 RW 的通信中性断点。最后，将所有剩余行按 RW 方式分片。由此得到内存中性的*FlexShard* 三层。

4.6 协同分片，一次完成

由于 FlexShard 采用均衡的逐行设计，因此将 FlexShard 扩展到多表序列分片十分简单。为此，只需将所有嵌入表的概率分布合并，并排序成一个大型分布。随后，FlexShard 只需对该分布进行分片，即可在精确的逐行粒度上同时对底层分布进行最优权衡，无需任何额外输入。

5 实验方法

基线。 据我们所知，我们是首个探索序列嵌入分片算法性能的工作，因此我们采用 TorchRec（Ivchenko 等人，2022年）作为基线。TorchRec 是 Meta 最先进的开源生产级分片框架，与 PyTorch 集成（Li 等人，2020年），支持序列嵌入并实现了逐行分片策略。

FlexShard。 为恰当地评估 FlexShard 和所提出的 Flex 分片策略，我们实现并比较了两个版本的 FlexShard，并以与基线相同的内存占用为目标；第一个是两层 FlexShard，仅采用两种可能的分片策略：数据并行和行分片；第二个是三层 FlexShard，采用本文讨论的全部三种策略：数据并行、行分片和 Flex。

数据集。 据我们所知，目前没有专为序列深度学习推荐模型设计的公开产业规模数据集。此外，现有的、为求和池化深度学习推荐模型发布的学术数据集主要包含独热数据（即每个样本中，每个嵌入表最多访问一行）~~，且其设计所针对~~的模型能够完全装入最新的 GPU，因此无需考虑分片问题。因此，我们通过在生产环境的机器学习训练数据集上训练四种不同的产业规模序列深度学习推荐模型来评估本项工作；每种模型都具有独特的底层特征分布。

训练系统规格。所有实验均在以下配置的 4 个训练节点组成的集群上进行，每个节点包含：配备 1.5 TB 动态随机存取存储器的四路 CPU，以及 8 张 NVIDIA A100 40GB GPU，通过混合立方网格拓扑中的高带宽 NVLink 3.0 在节点内互连。对于节点间通信，所有 GPU 均配有专用的 200 Gbps RoCE 网卡。通信库使用 NCCL。

模型	序列 平均长度	序列 表大小	嵌入 维度	静态 Size	局部 激活大小
RM-1	~ 1000	30M	256	30 GB	4.19 GB
RM-2	~ 1000	30M	256	30 GB	4.19 GB
RM-3	~ 500	10M	256	10 GB	2.10 GB
RM-4	~ 500	10M	256	10 GB	2.10 GB

表 *4.* DLRM 序列嵌入规格。

训练配置与实现。 所有使用的模型均采用 PyTorch 编写，其规格详见表 4。随后，我们使用 TorchRec 库对模型进行封装、分片和训练；两个 FlexShard 版本均采用 CUDA/Python 实现，并与 PyTorch/TorchRec 集成。训练使用的局部批量大小为 4,096，因此全局批量大小为 131,072。

性能分析。 我们的目标是评估 FlexShard 解析模型的准确性，以及其多层级分片设计带来的通信延迟改进，而不仅仅是端到端训练吞吐量的提升。因此，我们通过跟踪每个图形处理器的执行情况来进行评估。为确保准确性，我们在 50 次预热迭代后，跨所有 32 个 GPU 同步执行跟踪，并使用与 PyTorch Profiler 集成的生产级 libkineto 分析库。采用这种方式进行分析，不仅可以分析逐内核计时信息，还能记录和查看其他信息，例如每个通信集合操作的有效载荷大小。为准确测量内存利用率，我们还使用禁用 CUDA 缓存分配器的 PyTorch 版本进行了额外实验，并通过使用 NVIDIA 数据中心 GPU 管理器（ NVIDIA，2022a ）收集的指标，持续监测训练期间每个 GPU 的高带宽内存利用率。

6 评估

现在，我们介绍 FlexShard 的评估结果及其各项贡献。首先，我们给出并讨论 FlexShard 各种设计采用的输出分片，以及在内存中性优化目标下分析预期的通信改进。接下来，我们展示并分析三种分片设计中各类通信集合操作实际观测到的通信数据量和延迟，比较它们与分析预期值的差异，并介绍由此带来的端到端吞吐量提升。最后，我们展示并讨论各分片设计在所有 GPU 上实际观测到的高带宽内存利用率，以及不同设计之间的比较。

6.1 生成的分片方案及预期改进

表5 展示了每个 FlexShard 版本在各策略中放置的行数，以及相应覆盖的嵌入通信量占比的分析值。其中有几点值得注意。首先，尽管每种深度学习推荐模型的分布中都有一部分区域可用于实现内存中性的通信节省，但与表总大小相比，这一区域非常小，不到表的 1%。部分原因在于，现有深度学习框架中数据并行参数的内存成本很高（估计的 $6x$ 倍数取自 TorchRec），远高于仅复制参数所需的“基本” $1x$ 成本（每个 GPU 上）。此外，三层 FlexShard 下能够进行 Flex 分片的行数也进一步印证了这一点；该方案每个节点仅保留一个副本，而非每个 GPU 都保留一个副本。因此，即使模型和训练硬件保持不变，随着框架软件的改进，内存中性目标带来的通信节省和吞吐量提升仍将持续增加。

其次，尽管移至数据并行或 Flex 的行总数相对于原始表大小而言较少，但由于底层嵌入输入数据呈幂律分布，与基线 RW 相比，我们观察到 FlexShard 对总嵌入通信量的覆盖率呈现出不对称性。因此，RM-2 的三层 FlexShard 仅需将 0.4% 的行通过数据并行复制、将 8% 的行进行 Flex 分片，就能覆盖 85.6% 的全互交换嵌入通信量。

第4.1节的见解——并非每一行都同等重要——也体现在比较 RM-1 与 RM-2、RM-3 与 RM-4 的生成分片方案中。尽管 RM-{1,2}和 RM-{3,4} 的平均长度（近似）相同，因此基线最先进方法会对它们一视同仁，但 FlexShard 会根据底层分布调整每种分片方案，从而生成内存开销相当的设计；这些设计在各策略所分配的行数以及所覆盖的访问量方面均有所不同。

6.2 实际实现的通信与吞吐量提升

表6总结了模型训练期间的结果，证实了预测的改进：从仅采用基线 RW 的方案开始，随着全局全互交换规模大幅缩减，改进效果也随之提升
分片方案改为两层和三层 FlexShard 设计。以 RM-1 为例，FlexShard 两级和三级设计的全局全对全通信量分别减少了 53.7% 和 69.4%，与分析预测的 52.5% 和 67.4% 降幅十分接近。这些降幅是以增加规模更小、速度更快且可能可重叠执行的集合通信为代价的。总体而言，与 RM-1 的基线按行分片相比，端到端吞吐量分别提升了 61% 和 81%

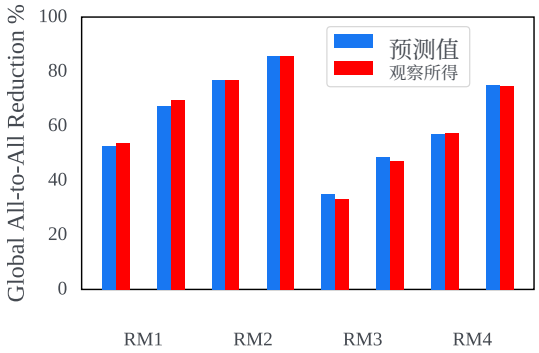


图 *6*。FlexShard 全局全互归约占比。

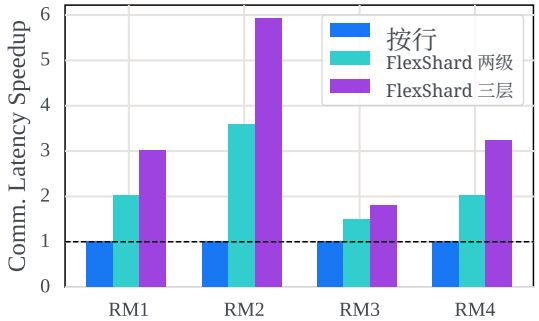


图 *7*。FlexShard 两级和三层相较于最先进的逐行基线，在 RM1、RM2、RM3 和 RM4 上端到端训练通信延迟的改善。

，分别对应 FlexShard 两级和三级设计。

在图 6 中，我们绘制了所有 RM 上两种 FlexShard 设计所预测与实际观测到的全局全互换流量减少量。可以很快看出，FlexShard 使用的解析模型能够根据目标拓扑和底层特征分布，相当准确地预测通信开销。

6.3 内存利用率

为了准确评估 FlexShard 相较于 RW 基线的实际内存用量，我们在禁用 CUDA 缓存分配器的情况下，对 RM-1 进行了实验。这会迫使 PyTorch 持续分配和释放图形处理器内存，从而使我们能够更准确地测量模型采用每种分片策略时的实际内存用量。我们观察到，与预期的内存占用不变不同，FlexShard 的两种策略相较于基线按行分片，实际上都显著降低了峰值内存用量 。具体而言，采用两级设计时，FlexShard 每个图形处理器约少用 2.5 GiB；采用三级设计时，每个图形处理器约少用 1.5 GiB。出现这一偏差（尽管是有利的）的原因，在于分析模型中使用的数据并行内存开销乘数。所用的 $6x$ 乘数取自

模型	分片	DP 行数	预测 DP 覆盖访问率	Flex 行数	预测 Flex 覆盖访问率	RW 行数	预测 RW 覆盖访问率	预测附加项内存成本
RM-1	两层	346,144	52.5%	N/A	N/A	29,653,856	47.5%	∼ 0
RM-1	三层	100,064	36.8%	1,184,544	30.6%	28,715,392	32.6%	∼ 0
RM-2	两层	507,072	77.0%	N/A	N/A	29,492,928	23.0%	∼ 0
RM-2	三层	128,736	63.9%	2,397,216	21.7%	27,474,048	14.4%	∼ 0
RM-3	两层	115,296	34.9%	N/A	N/A	9,884,704	65.1%	∼ 0
RM-3	三层	36,704	22.7%	318,560	25.9%	9,644,736	51.4%	∼ 0
RM-4	两层	184,864	57.0%	N/A	N/A	9,815,136	43.0%	∼ 0
RM-4	三层	55,040	39.0%	592,992	36.0%	9,351,968	25.0%	∼ 0

表5。行数 FlexShard 两级和三级在每个 RM 中各分片层级所放置的行数及其预测访问覆盖率 .

模型	分片	全局全对全大小（每个方向）	全局全对全归约百分比	全局全对全总时间	组内全对全总时间（可重叠）	全局全归约总时间	跨组全归约总时间（可重叠）
RM-1	RW	972,424,960	N/A	307 毫秒	N/A	N/A	N/A
RM-1	两层	450,510,848	53.7%	149 毫秒	N/A	3 ms	N/A
RM-1	三层	297,697,024	69.4%	99 毫秒	30 毫秒	3 ms	14 毫秒
RM-2	RW	958,656,768	N/A	308 毫秒	N/A	N/A	N/A
RM-2	两层	220,126,208	77.0%	80 毫秒	N/A	6 ms	N/A
RM-2	三层	138,198,784	85.6%	49 毫秒	21 毫秒	3 ms	27 毫秒
RM-3	RW	466,826,496	N/A	156 毫秒	N/A	N/A	N/A
RM-3	两层	311,369,728	33.3%	101 毫秒	N/A	4 ms	N/A
RM-3	三层	246,077,440	47.3%	85 毫秒	13 毫秒	2 ms	5 ms
RM-4	RW	473,709,312	N/A	155毫秒	N/A	N/A	N/A
RM-4	两层	201,887,232	57.4%	72毫秒	N/A	5 ms	N/A
RM-4	三层	118,944,768	74.8%	45毫秒	18毫秒	3 ms	9 ms

表 6。FlexShard 两级（数据并行/按行）和 FlexShard 三层（数据并行/按行/灵活）在 RM1、RM2、RM3 和 RM4 上的实测通信性能结果，与最先进的按行基线的比较。

TorchRec 实际上高估了对一组参数进行数据并行所需的内存。因此，FlexShard 所实现的通信延迟和吞吐量提升，同时也降低了内存利用率。

7 相关工作

由于 DLRM 训练对现实世界的影响，以及相较于更传统的深度学习架构所面临的独特挑战，其系统层面的影响正日益受到关注。

针对其规模问题，已有研究提出了多种优化方法，例如：根据访问频率逐行缩放嵌入维度（Ginart 等人，2019年）；对嵌入表进行张量分解和近似（Yin 等人，2021年）；甚至用专门的深度神经网络替换整个嵌入层（Kang 等人，2021年）。这些优化引入了权衡，因为它们试图在降低内存容量需求的同时，平衡计算需求的增加以及可能造成的准确率影响。

另一类优化旨在降低嵌入表的访问延迟。这些研究主要聚焦于两种方法之一：动态缓存，或嵌入表的静态放置（即分片）。ScratchPipe（Kwon 与 Rhu，2022年）和 RecShard（Sethi 等人，2022年）针对混合 CPU-GPU 训练系统中的嵌入访问延迟问题。ScratchPipe 通过使用 GPU 侧的预读缓存来尝试确保所有嵌入访问都命中局部 GPU 高带宽内存，而 RecShard 则使用混合整数线性规划和每个嵌入表的分布来静态地将访问最频繁的行放置在 GPU 高带宽内存中。AutoShard（Zha 等人，2022年）聚焦于仅使用多 GPU 的训练系统中的嵌入表分片，并使用深度强化学习和基于神经网络的成本模型来做出放置决策。

虽然动机相似，但这些工作是为求和池化嵌入表设计的，无法直接应用于本文所研究的基于序列的 DLRM。

8 结论

本文聚焦于面向工业规模深度学习推荐模型的序列嵌入分片。我们深入分析了用于在硬件拓扑中划分和放置这些序列嵌入表的各种分片策略，并对其系统扩展性进行建模；此外，我们还提出了一种新的分片策略，可适应真实训练集群中异构的网络拓扑。进一步地，我们展示了硬件所承受的系统需求如何随底层序列特征分布而扩展。在此基础上，我们提出 FlexShard，这是一种分片技术，可针对给定的硬件拓扑和内存约束，以逐行粒度在多种潜在分片策略间对序列嵌入表进行最优划分和放置。我们基于最先进的开源 DLRM 训练框架 TorchRec，使用生产数据实现并评估了 FlexShard。在具有代表性的工业规模序列深度学习推荐模型上，FlexShard 能够将全局全互换通信流量减少 85% 以上，使端到端通信延迟几乎加速 6 倍，

这类模型具有代表性的工业规模序列深度学习推荐模型。

参考文献

Abadi, M., Agarwal, A., Barham, P., Brevdo,E., Chen, Z.,Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin,M.,Ghemawat, S., Goodfellow, I., Harp, A., Irving, G.,Isard,M., Jia,Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Lev-enberg, J., Man´e,D.,Monga, R., Moore,S., Murray, D.,Olah, C., Schuster, M., Shlens, J., Steiner, B.,Sutskever,I., Talwar,K., Tucker, P., Vanhoucke, V., Vasudevan,V.,Vi´egas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow：在异构系统上进行大规模机器学习，2015年。网址https://www.tensorflow.org/。软件可从tensorflow.org 获取。

Acun, B., Murphy, M., Wang, X., Nie, J., Wu, C.-J. 和 Hazelwood, K. 了解大规模深度学习推荐模型的训练效率。载于 *2021 IEEEInternational Symposium onHigh PerformanceComputerArchitecture(HPCA)*，2021年。

Adnan, M., Maboud, Y. E., Mahajan, D. 和 Nair, P. J. 通过利用推荐系统中的热门嵌入实现高性能训练。 *CoRR*，2021年。网址：https://arxiv.org/abs/2103.00686。Amazon。Amazon EC2 P4d 实例。2022年。网址：https://aws.amazon.com/ec2/instance-types/p4/。

Anil, R., Gadanho, S., Huang, D., Jacob, N., Li, Z., Lin, D., Phillips, T., Pop, C., Regan, K., Shamir, G. I., Shivanna, R. 和 Yan, Q. 走进工厂车间：面向工业规模广告推荐模型的机器学习工程。2022年。doi：10.48550/ARXIV.2209.05310。网址：https://arxiv.org/abs/2209.05310。Chen, Q., Zhao, H., Li, W., Huang, P. 和 Ou, W. 阿里巴巴的电子商务推荐行为序列 Transformer。载于 *Proceedings of the 1st InternationalWorkshopon DeepLearning Practice for High-Dimensional SparseData*，DLP-KDD '19，纽约州纽约，美国，2019年。计算机协会。ISBN 9781450367837。doi：10.1145/3326937.3341261。网址：https://doi.org/10.1145/3326937.3341261。

Cheng, H.-T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., Anil, R., Haque, Z., Hong, L., Jain, V., Liu, X., 和 Shah, H. 面向推荐系统的宽深度学习。发表于推荐系统深度学习研讨会，2016年。

Covington, P., Adams, J., 和 Sargin, E. 用于 YouTube 推荐的深度神经网络。发表于*ACM* 推荐系统会议，2016年。

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., 和 Fei-Fei, L. ImageNet：一个大型分层图像数据库。载于 *2009*年 *IEEE* 计算机视觉与模式识别会议，第 248–255 页，2009年。doi：10.1109/CVPR.2009.5206848。

Devlin, J., Chang, M.-W., Lee, K., 和 Toutanova, K. BERT：用于语言理解的深度双向变换器预训练。载于*2019*年计算语言学协会北美分会会议论文集：人类语言技术，第 *1* 卷（长篇与短篇论文），第 4171–4186 页，明尼阿波利斯，明尼苏达州，2019年6月。计算语言学协会。doi：10.18653/v1/N19-1423。网址：https://aclanthology.org/N19-1423。

Facebook. Facebook Zion系统规范 1.0. 2021年. URL https://www.opencompute.org/documents/facebook-zion-system-spec-1-0-pdf.

Ginart, A., Naumov, M., Mudigere, D., Yang, J., and Zou, J. 混合维度嵌入及其在内存高效推荐系统中的应用。 *arXiv*预印本 *arXiv:1909.11810*，2019年。Gomez-Uribe, C. A. and Hunt, N. Netflix 推荐系统：算法、商业价值与创新。 *ACM*管理信息系统汇刊，6(4)，2016年12月。ISSN 2158-656X。doi: 10.1145/2843948。URLhttps://doi.org/10.1145/2843948。Gupta, U., Hsia, S., Saraph, V., Wang, X., Reagen, B., Wei, G.-Y., Lee, H.-H. S., Brooks, D., and Wu, C.-J. DeepRecSys：一种用于优化大规模端到端神经推荐推理的系统。发表于 *ACM/IEEE*计算机体系结构年度国际研讨会论文集，2020a。

Gupta, U., Wu, C.-J., Wang, X., Naumov, M., Reagen, B., Brooks, D., Cottel, B., Hazelwood, K., Hempstead, M., Jia, B., Lee, H.-H. S., Malevich, A., Mudigere, D., Smelyanskiy, M., Xiong, L., and Zhang, X. Facebook基于DNN的个性化推荐的体系结构影响。发表于 *2020*年*IEEE*高性能计算机体系结构国际研讨会（*HPCA*），2020b。Hazelwood, K., Bird, S., Brooks, D., Chintala, S., Diril, U., Dzhulgakov, D., Fawzy, M., Jia, B., Jia, Y., Kalro, A., Law, J., Lee, K., Lu, J., Noordhuis, P., Smelyanskiy, M., Xiong, L., and Wang, X. Facebook的应用机器学习：数据中心基础设施视角。

发表于 *2018*年*IEEE*高性能计算机体系结构国际研讨会（*HPCA*），2018年。

He, K.、Zhang, X.、Ren, S. 和 Sun, J. 深度残差学习用于图像识别。载于*2016*年 *IEEE* 计算机视觉与模式识别会议（*CVPR*），第 770–778 页，2016年。doi：10.1109/CVPR.2016.90。

Ivchenko, D.、Van Der Staay, D.、Taylor, C.、Liu, X.、Feng, W.、Kindi, R.、Sudarshan, A. 和 Sefati, S. TorchRec：用于推荐系统的 PyTorch 领域库。载于第 *16* 届 *ACM* 推荐系统会议论文集，RecSys '22，第 482–483 页，美国纽约州纽约，2022年。计算机协会。ISBN 9781450392785。doi：10.1145/ 3523227.3547387。网址：https://doi.org/10.1145/3523227.3547387。

Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., Boyle, R., Cantin, P.-l., Chao, C., Clark, C., Coriell, J., Daley, M., Dau, M., Dean, J., Gelb, B., Ghaemmaghami, T. V., Gottipati, R., Gulland, W., Hagmann, R., Ho, C. R., Hogberg, D., Hu, J., Hundt, R., Hurt, D., Ibarz, J., Jaffey, A., Jaworski, A., Kaplan, A., Khaitan, H., Killebrew, D., Koch, A., Kumar, N., Lacy, S., Laudon, J., Law, J., Le, D., Leary, C., Liu, Z., Lucke, K., Lundin, A., MacKean, G., Maggiore, A., Mahony, M., Miller, K., Nagarajan, R., Narayanaswami, R., Ni, R., Nix, K., Norrie, T., Omernick, M., Penukonda, N., Phelps, A., Ross, J., Ross, M., Salek, A., Samadiani, E., Severn, C., Sizikov, G., Snelham, M., Souter, J., Steinberg, D., Swing, A., Tan, M., Thorson, G., Tian, B., Toma, H., Tuttle, E., Vasudevan, V., Walter, R., Wang, W., Wilcox, E., 和 Yoon, D. H. 数据中心内张量处理单元的性能分析。载于*ACM/IEEE* 第*44*届国际计算机体系结构年度研讨会论文集，2017年。

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., ˇZ´ıdek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., Back, T., Petersen, S., Reiman, D., Clancy, E., Zielinski, M., Steinegger, M., Pacholska, M., Berghammer, T., Bodenstein, S., Silver, D., Vinyals, O., Senior, A. W., Kavukcuoglu, K., Kohli, P., and Hassabis, D. 使用 AlphaFold 实现高精度蛋白质结构预测。《自然》，2021年。

Kang, W.-C. 和 McAuley, J. 自注意力序列推荐。载于*2018*年 *IEEE*国际数据挖掘会议（*ICDM*），第197–206页，2018年。doi：10. 1109/ICDM.2018.00035。

Kang, W.-C., Cheng, D. Z., Yao, T., Yi, X., Chen, T., Hong, L. 和 Chi, E. H. 学习在不使用嵌入表的情况下嵌入类别特征以用于推荐。*CoRR*，2021年。网址：https://arxiv.org/abs/2010.10784。

Ke, L.、Gupta, U.、Cho, B. Y.、Brooks, D.、Chandra, V.、Diril, U.、Firoozshahian, A.、Hazelwood, K. M.、Jia, B.、Lee, H. S.、Li, M.、Maher, B.、Mudigere, D.、Naumov, M.、Schatz, M.、Smelyanskiy, M.、Wang, X.、Reagen, B.、Wu, C.、Hempstead, M. 和 Zhang, X. RecNMP：利用近内存处理加速个性化推荐。载于 第*47*届 *ACM/IEEE*计算机体系结构年度国际研讨会，*ISCA 2020*，西班牙瓦伦西亚，*2020*年*5*月*30*日至*6*月*3*日，第790–803页。IEEE，2020年。doi：10.1109/ISCA45697.2020.00070。URLhttps://doi.org/10.1109/ISCA45697.2020.00070。

Kwon, Y. 和 Rhu, M. 从头开始训练个性化推荐系统（图形处理器）：向前看，而非回头看。载于 第*49*届年度国际计算机体系结构研讨会论文集，ISCA '22，第860–873页，纽约州纽约，美国，2022年。计算机协会。ISBN 9781450386104。doi: 10.1145/3470496.3527386。网址：https://doi.org/10.1145/3470496.3527386。

Li, C., Liu, Z., Wu, M., Xu, Y., Zhao, H., Huang, P., Kang, G., Chen, Q., Li, W. 和 Lee, D. L. 用于天猫推荐的动态路由多兴趣网络。载于 第*28*届 *ACM* 国际信息与知识管理会议论文集，CIKM '19，第2615–2623页，纽约州纽约，美国，2019年。计算机协会。ISBN 9781450369763。doi: 10.1145/3357384.3357814。网址：https://doi.org/10.1145/3357384.3357814。

Li, S., Zhao, Y., Varma, R., Salpekar, O., Noordhuis, P., Li, T., Paszke, A., Smith, J., Vaughan, B., Damania, P., and Chintala, S. PyTorch分布式：加速数据并行训练的经验。*VLDB*学报论文集，13(12):3005–3018，2020年8月。ISSN 2150-8097。doi: 10.14778/3415478.3415530。URLhttps://doi.org/10.14778/3415478.3415530。

Lui, M., Yetim, Y., Ozkan, O., Zhao, Z., Tsai, S.-Y., Wu, C.-J., 和 Hempstead, M. 理解容量驱动的横向扩展神经推荐推理。载于*2021*年 *IEEE*系统与软件性能分析国际研讨会*(ISPASS)*，2021年。Microsoft。Microsoft Azure NDv4实例。2022年。网址 https://azure.microsoft.com/en-us/blog/azure-announces-general-availability-of-scaleup-scaleout-nvidia-a100-gpu-instances-claims-title-of-fastest-public-cloud-super/。

Mudigere, D., Hao, Y., Huang, J., Tulloch, A., Sridharan, S., Liu, X., Ozdal, M., Nie, J., Park, J., Luo, L., Yang, J. A., Gao, L., Ivchenko, D., Basant, A., Hu, Y., Yang, J., Ardestani, E. K., Wang, X., Komuravelli, R., Chu, C.-H., Yilmaz, S., Li, H., Qian, J., Feng, Z., Ma, Y., Yang, J., Wen, E., Li, H., Yang, L., Sun, C., Zhao, W., Melts, D., Dhulipala, K., Kishore, K., Graf, T., Eisenman, A., Matam, K. K., Gangidi, A., Chen, G. J., Krishnan, M., Nayak, A., Nair, K., Muthiah, B., khorashadi, M., Bhattacharya, P., Lapukhov, P., Nau mov, M., Qiao, L., Smelyanskiy, M., Jia, B., 和 Rao, V. 大规模深度学习推荐模型的高性能分布式训练。*CoRR*，2021年。

Naumov, M., Mudigere, D., Shi, H.-J. M., Huang, J., Sun- daraman, N., Park, J., Wang, X., Gupta, U., Wu, C.-J., Azzolini, A. G., Dzhulgakov, D., Mallevich, A., Cher- niavskii, I., Lu, Y., Krishnamoorthi, R., Yu, A., Kon- dratenko, V., Pereira, S., Chen, X., Chen, W., Rao, V., Jia, B., Xiong, L., and Smelyanskiy, M. 面向个性化和推荐系统的深度学习推荐模型。2019年。

Naumov, M., Kim, J., Mudigere, D., Sridharan, S., Wang, X., Zhao, W., Yilmaz, S., Kim, C., Yuen, H., Ozdal, M., Nair, K., Gao, I., Su, B.-Y., Yang, J., andSmelyanskiy, M. Facebook数据中心中的深度学习训练：横向扩展与纵向扩展系统的设计。*CoRR*，2020年。

NVIDIA。NVIDIA 数据中心 GPU 管理器。2022a。链接：https://developer.nvidia.com/dcgm。

NVIDIA。NVIDIA 数据中心 GPU 管理器。2022b。链接：https://www.nvidia.com/en-us/data-center/dgx-systems/。

Pi, Q.、Bian, W.、Zhou, G.、Zhu, X. 和 Gai, K. 关于长序列用户行为建模以预测点击率的实践。载于 第 *25* 届 *ACM SIGKDD* 国际知识发现与数据挖掘会议论文集，KDD '19，第 2671–2679 页，纽约州纽约，美国，2019年。计算机协会。ISBN 9781450362016。doi：10.1145/ 3292500.3330666。链接：https://doi.org/10.1145/3292500.3330666。

Raimond, Y. 推荐系统的深度学习，2018年。https://www.slideshare.net/moustaki/deep-learning-for-recommender-systems-86752234。

Sethi, G.、Acun, B.、Agarwal, N.、Kozyrakis, C.、Trippel, C. 和 Wu, C.-J. Recshard：面向行业级神经推荐的基于统计特征的内存优化。载于第*27*届*ACM*编程语言与操作系统架构支持国际会议（*ASPLOS*），2022年。

Sheng, X.-R.、Zhao, L.、Zhou, G.、Ding, X.、Dai, B.、Luo, Q.、Yang, S.、Lv, J.、Zhang, C.、Deng, H. 和 Zhu, X. 一个模型服务所有场景：用于多领域点击率预测的星型拓扑自适应推荐器。载于第*30*届*ACM*信息与知识管理国际会议论文集，CIKM '21，第4104–4113页，纽约州纽约，美国，2021年。计算机协会。ISBN 9781450384469。doi：10.1145/3459637.3481941。URL：https://doi.org/10.1145/3459637.3481941。

Sullivan, D. 谷歌在每次搜索中都使用 RankBrain，影响了其中"很多"搜索结果的排名，2016年。https://searchengineland.com/google-loves-rankbrain-uses-for-every-search-252526。

Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., 和 Jiang, P. Bert4rec：采用 Transformer 双向编码器表示的序列推荐。载于第*28*届 *ACM*信息与知识管理国际会议论文集，CIKM '19，第 1441–1450 页，纽约州纽约，美国，2019年。计算机协会。ISBN 9781450369763。doi: 10.1145/3357384.3357895。网址 https://doi.org/10.1145/3357384.3357895。

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, u., 和 Polosukhin, I. 你只需要注意力。载于第*31*届神经信息处理系统国际会议论文集，NIPS'17，第 6000–6010 页，雷德胡克，纽约州，美国，2017年。Curran Associates Inc. ISBN 9781510860964。

Weyn, J. A., Durran, D. R., 和 Caruana, R. 使用立方球上的深度卷积神经网络改进数据驱动的全球天气预报。《地球系统建模进展》，12(9)，2020年9月。ISSN 1942- 2466。doi: 10.1029/2020ms002109。URLhttp://dx.doi.org/10.1029/2020MS002109。

Wilkening, M., Gupta, U., Hsia, S., Trippel, C., Wu, C.-J., Brooks, D., 和 Wei, G.-Y. Recssd：面向基于固态硬盘的推荐推理的近数据处理。载于第*26*届 *ACM* 编程语言与操作系统架构支持国际会议论文集，2021年。

Wu, C.-J., Raghavendra, R., Gupta, U., Acun, B., Ardalani, N., Maeng, K., Chang, G., Behram, F. A., Huang, J., Bai, C., Gschwind, M., Gupta, A., Ott, M., Melnikov, A., Candido, S., Brooks, D., Chauhan, G., Lee, B., Lee, H.-H. S., Akyildiz, B., Balandat, M., Spisak, J., Jain, R., Rabbat, M. 和 Hazelwood, K. 可持续人工智能：环境影响、挑战与机遇。*CoRR*，abs/2111.00364，2021年。

Yin, C., Acun, B., Liu, X. 和 Wu, C. Tt-rec：面向深度学习推荐模型的张量链压缩。*CoRR*，abs/2101.11714，2021年。网址https://arxiv.org/abs/2101.11714。

Zha, D.、Feng, L.、Bhushanam, B.、Choudhary, D.、Nie, J.、Tian, Y.、Chae, J.、Ma, Y.、Kejariwal, A. 和 Hu, X. AutoShard：推荐系统的自动嵌入表分片。载于第*28*届 *ACM SIGKDD* 知识发现与数据挖掘会议论文集，KDD '22，第4461–4471页，纽约州纽约，美国，2022年。计算机协会出版。ISBN 9781450393850。doi：10.1145/ 3534678.3539034。网址：https://doi.org/10.1145/3534678.3539034。

Zhao, M.、Agarwal, N.、Basant, A.、Gedik, B.、Pan, S.、Ozdal, M.、Komuravelli, R.、Pan, J.、Bao, T.、Lu, H.、Narayanan, S.、Langman, J.、Wilfong, K.、Rastogi, H.、Wu, C.-J.、Kozyrakis, C. 和 Pol, P. 理解大规模深度推荐模型训练的数据存储与摄取：工业产品。载于第*49*届国际计算机体系结构年度研讨会论文集，ISCA '22，第1042–1057页，纽约州纽约，美国，2022年。计算机协会出版。ISBN 9781450386104。doi：10.1145/ 3470496.3533044。网址：https://doi.org/10.1145/3470496.3533044。

Zhao, W., Xie, D., Jia, R., Qian, Y., Ding, R., Sun, M. 和 Li, P. 面向大规模深度学习广告系统的分布式分层GPU参数服务器。载于《机器学习与系统论文集》，2020年。

Zhao, Z., Hong, L., Wei, L., Chen, J., Nath, A., Andrews, S., Kumthekar, A., Sathiamoorthy, M., Yi, X. 和 Chi, E. 推荐接下来观看什么视频：一个多任务排序系统。载于《第*13*届*ACM*推荐系统会议论文集》，RecSys '19，第43–51页，纽约州纽约，美国，2019年。计算机协会。ISBN 9781450362436。doi：10.1145/3298689.3346997。网址https://doi.org/10.1145/3298689.3346997。

Zhou, G., Mou, N., Fan, Y., Pi, Q., Bian, W., Zhou, C., Zhu, X., and Gai, K. 用于点击率预测的深度兴趣演化网络。载于*AAAI*人工智能会议论文集，第 33 卷，第 5941–5948 页，2019a。

Zhou, G., Mou, N., Fan, Y., Pi, Q., Bian, W., Zhou, C., Zhu, X., and Gai, K. 用于点击率预测的深度兴趣演化网络。*AAAI*人工智能会议论文集，33 (01):5941–5948，2019 年 7 月。doi: 10.1609/aaai.v33i01.33015941。网址：
h
t
tps://ojs.aaai.org/index.php/AAAI/article/view/4545。