高效训练百万亿参数预训练模型的系统挑战和对策

Challenges and Measures for Efficient Training of Trillion-Parameter Pre-Trained Models

马子轩/MA Zixuan，翟季冬/ZHAI Jidong，韩文弢/HAN Wentao，陈文光/CHEN Wenguang，郑纬民/ZHENG Weimin
（清华大学，中国 北京 100083）
(Tsinghua University, Beijing 100083, China)

DOI：10.12142/ZTETJ.202202008
网络出版地址：https://kns.cnki.net/kcms/detail/34.1228.tn.20220418.1725.002.html
网络出版日期：2022-04-19
收稿日期：2022-02-20

摘要：随着预训练模型规模的急剧增长，训练此类模型需要海量的计算和存储能力。为此，本工作在新一代国产高性能计算机上训练了一个174万亿参数的超大规模预训练模型，模型参数量可与人脑中的突触数量相媲美。重点讨论在训练这一超大规模预训练模型中遇到的几个关键系统挑战：如何选取高效并行策略，如何进行高效数据存储，如何选取合适的数据精度，以及如何实现动态负载均衡，并总结了针对上述挑战的一些解决方法。

关键词：人工智能；超级计算机；混合专家；异构系统

Abstract: As the size of pre-trained artificial intelligence models grows dramatically each year, training such models requires massive computing and memory capabilities. To this end, an unprecedentedly large-scale pre-trained model with 174 trillion parameters on an entire supercomputer is proposed, which rivals the number of synapses in a human brain. The key challenges encountered in such large-scale model training, including deciding efficient parallel strategy, performing efficient data storage, deciding appropriate data precision, and dynamic load balancing are proposed. Then the solutions to the above challenges are summarized.

Keywords: artificial intelligence; supercomputer; mixture of experts; heterogeneous architecture

1 大规模预训练模型的发展背景

近年来，深度学习在计算机视觉（CV）、自然语言处理（NLP）、推荐系统、决策模型等各个领域都发挥了重要作用。同时，大规模预训练模型技术已经成为深度学习领域最先进的技术，并在多个领域的下游应用中表现出优秀的性能。近年来，谷歌的推荐系统、搜索引擎，阿里巴巴的推荐系统、图像生成等任务均采用了预训练模型。而在预训练模型方面，学术界已达成共识：模型规模和模型准确度有明显的正相关关系[1-6]。表1展示了近年来预训练模型的发展趋势：从最早的1.1亿参数量的第1代预训练GPT发展到最新的万亿参数量的GPT-3、Switch Transformer等模型。探索更大参数量的模型具有重要的科学意义。

从计算的角度看，随着模型规模的增长，训练模型的数据会变多。此时，单机已经无法满足大规模模型训练的计算需求。在模型训练的过程中，计算分为3个步骤：前向、反向、更新。在前向过程中，模型使用输入数据和参数进行计算，得到预测结果并和标注数据进行对比，从而计算出该次预测的损失；在反向过程中，模型对损失进行传播，计算所有参数的梯度；在更新阶段，模型再使用计算出的梯度来更新参数。该过程在训练中不断迭代，最终达到模型的收敛状态。当模型规模较大时，单机串行执行已无法满足模型的计算需求。这时，我们需要使用一些并行策略来辅助训练，如数据并行和模型并行。

顾名思义，使用数据并行策略时，模型对输入的数据进

▼表1 近年发布的预训练模型

预训练模型	模型参数量	时间/年
GPT[7]	1.1亿	2018
BERT-Large[2]	3.4亿	2018
GPT-2[4]	15亿	2019
GShard[8]	6 000亿	2020
GPT-3[1]	1 750亿	2020
Switch Transformer[9]	1.6万亿	2021
BaGuaLu	174万亿	2021

行拆分。如图1所示，数据被划分到不同的节点后再进行计算，每个节点上都保存一份完整的模型。模型在正向和反向的过程中，均不会进行通信；而在正向结束计算梯度以及更新时，则会引入全局的All-Reduce通信。

使用模型并行策略时，模型对模型参数本身进行拆分，这可以理解成对参数矩阵进行切分。在执行过程中，模型执行被切分的计算时，会根据切分方式的不同引入不同的通信方式。如图1所示，在两种不同的模型并行执行模式中，不同的切分策略产生了两种不同的通信模式：All-Reduce和All-Gather。

无论是数据并行还是模型并行，在执行过程中，都需要通过对数据进行切分来减少每个节点的计算量和存储量，从而达到并行训练的目的。在模型训练中，参数量和训练速度是两个重要的衡量指标。经验证明，越大规模的模型所需的训练样本数就越多[10]。因此，大模型训练需要同时扩展参数量和训练吞吐量。在传统的并行模式中，通过对输入进行切分，数据并行可以有效扩展训练吞吐量。而由于数据并行需要在每个节点上都存储一份参数，这种并行模式不能扩展参数量。模型并行可以对参数进行切分，从而有效扩展参数量。切分后多个机器使用同一组输入数据，因此不能有效扩展训练吞吐量。传统的大模型训练通常使用混合数据并行和模型并行的方法进行训练[11]。

混合专家（MoE）模型是近年来一种新的预训练模型[8-9]。这种模型将一个大模型切分为多个小模型（专家）。在训练过程中，数据先通过一个小规模网络（一般称之为网关）进行路由，再选择适合的一部分专家进行计算，最后加权求和得到输出。

对MoE模型而言，一个整体的大模型由多个小模型构成。这样一来，这类大模型的参数量可以和传统的稠密大模型一样大或更大。同时，由于每次运算只是从离散的模型中稀疏地选取一部分来激活，MoE模型的整体计算量与稠密小模型相当。因此，MoE模型可以在扩展模型参数、增大模型规模的同时，在相同的训练时间内，达到优于稠密大模型的准确度。谷歌最新的Switch Transformer[9]就采用了MoE的架构。

在MoE大模型中，并行训练会引入两种划分：数据划分和专家划分。在训练过程中，每一组数据需要选择合适的专家，在不同节点间交换数据。这个过程会引入全局的All-to-All通信。MoE的并行训练很好地满足了大模型训练对吞吐量和模型规模的需求。在MoE并行训练中，不同机器存储不同的专家，有效扩展了模型的参数规模。同时，由于不同机器选取不同的输入，该方法也有效扩展了模型训练的吞吐量。因此，MoE模型非常适合对大模型进行扩展。

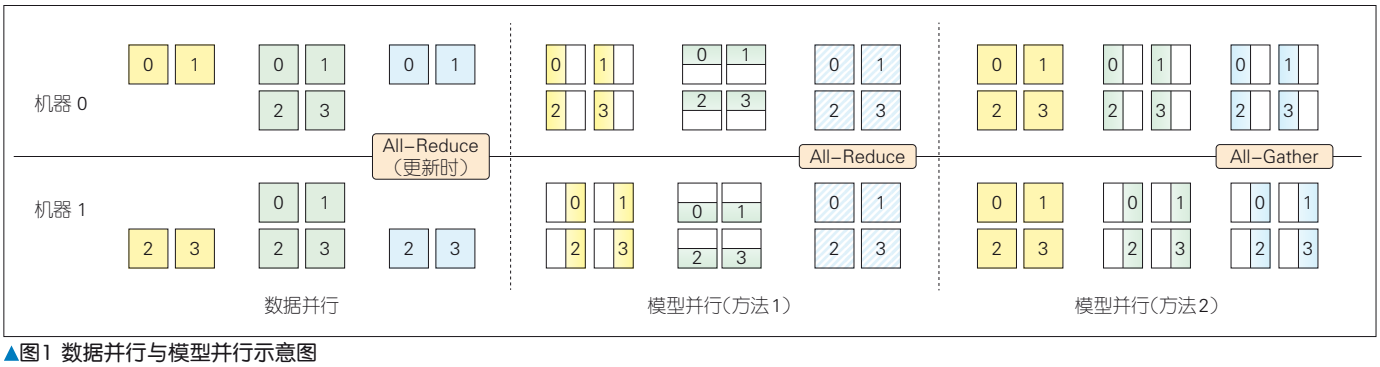
2 大规模预训练模型面临的挑战

当模型扩展到超大规模时，会遇到多方面的挑战。例如，如何选取高效的并行策略，如何进行高效数据存储，如何选取合适数据精度以及如何实现动态负载均衡。

2.1 如何选取高效的并行策略

在大模型训练中，有多种并行策略可供选择，如数据并行、模型并行、流水线并行以及MoE并行等各种并行模式。这些并行策略将大模型训练按照不同的方式进行划分，从而使任务可以很好地在多机上执行。而不同的划分方法，对底层有不同的计算与通信需求。例如，一类并行策略的通信需求虽然小，但可能会引入更多的计算；另一类并行策略虽然不引入额外计算，但可能会引入更多的通信量。在训练过程中，如何选择合适的并行策略是一个重要的挑战。

以16个节点为例，这些节点在训练过程中需要被切分的部分包括输入数据与模型参数。如果使用简单的数据并行，那么模型参数并不被划分，每台机器上都有一份完整复制的模型。那么，数据并行的核心是把输入数据简单地切分成16份。与此类似，模型并行把模型切成16份。如果是MoE并行，则有16个专家，每个节点1个专家。在此基础



▲图1 数据并行与模型并行示意图

上，不同并行策略之间还可以进行混合，例如数据并行加模型并行的混合方式。

这些策略的组合给并行设计带来了巨大的选择空间。在此基础上，在很多系统中，网络拓扑会带来不同节点间通信性能的不同。例如，在胖树结构中，通常会存在网络裁剪的问题，这导致一部分节点之间的通信延迟更高、带宽更低。这种差异导致相同并行策略在不同情况下的性能有所不同，从而使并行策略选择问题更加复杂。因此，如何选择一个高效的并行策略具有很大的挑战性。

2.2 如何高效存储和划分数据

以万亿参数量的模型为例，如果模型精度是32位，模型参数本身就会占用4 TB的空间。与此同时，模型的梯度需要占用4 TB的空间，优化器的更新参数需要占用8 TB的空间，计算中间值需要约8 TB的空间。为了保证模型的训练性能，这些数据需要存储在内存中。如果使用NVIDA Tesla V100训练该模型，仅在显存中存储这些数据就需要768块显卡。在这种情况下，不同的划分方式对底层的计算和通信也会产生不同的影响。因此，用高效划分数据来支持高效训练同样非常具有挑战性。

2.3 如何选取合适的存储精度

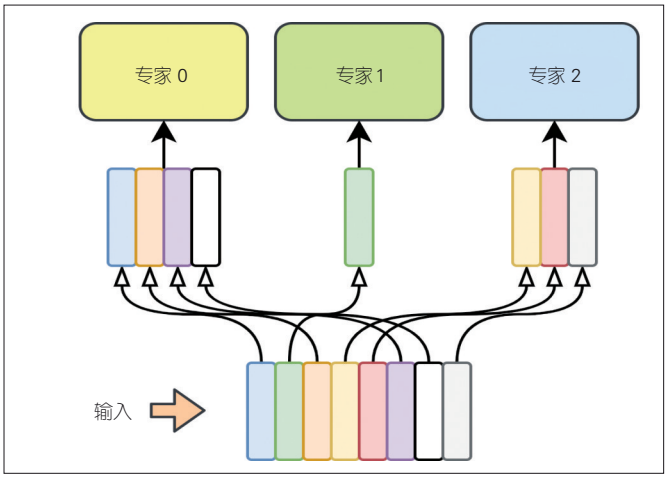
在计算过程中，使用更低的精度（如16位）训练时，模型需要的内存量更小且性能更好，但准确度有所下降，训练轮数有所增加；使用更高精度（如32位或64位）训练时，模型需要的内存量更大，计算的要求更高，但模型准确度高，训练轮数少。因此，如何在模型中选择合适的精度进行计算，以平衡模型精度与模型训练时间，给模型扩展带来了新的挑战。

2.4 如何实现动态负载均衡

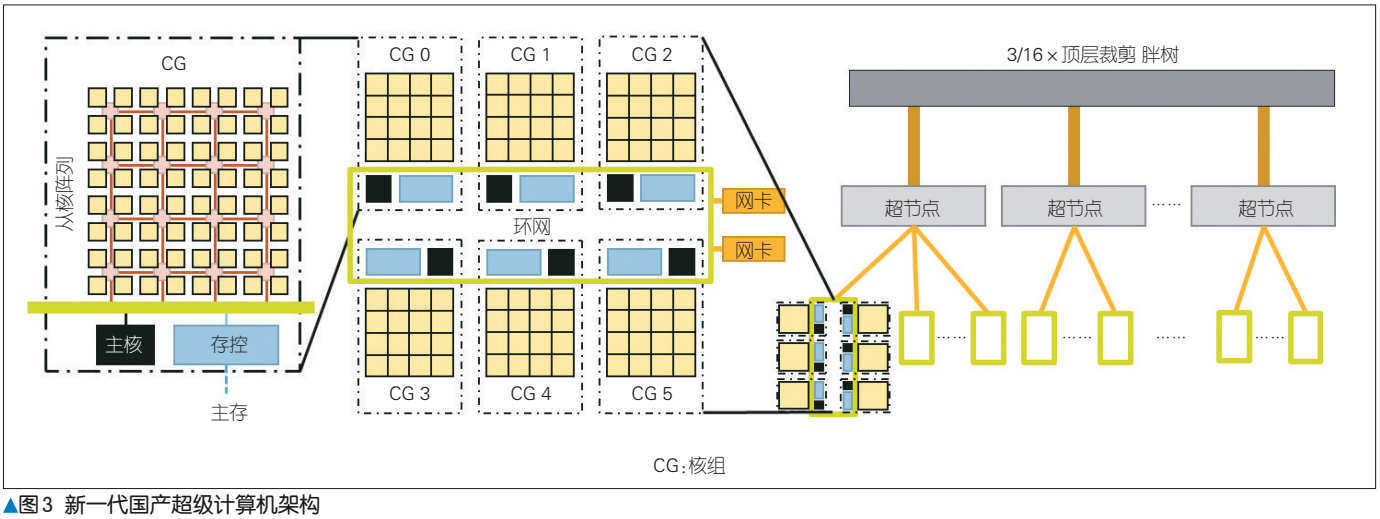
MoE模型会通过网关对不同的输入样本选择不同的专家进行计算。在实际运行过程中，样本的选择存在严重的负载不均衡问题。例如，在语言处理中，常用语言元素出现的频率高，对应的专家负载更高，如图2所示的专家0。此时，如果训练规模非常大，部分节点的高负载就会降低整体系统的执行效率。因此，如何动态改善负载均衡，以提高模型的训练效率成为大模型训练的重要挑战。

3 新一代国产超级计算机

新一代国产神威超级计算机[13]采用新一代神威处理器芯片——申威26010-Pro，其架构如图3所示。其中，每个节点包括6个核组（CG），每个核组包含1个主核（计算控制



▲图2 混合专家模型中的负载均衡问题



▲图3 新一代国产超级计算机架构

核心）和64个从核（计算核心CPE）。加速计算主要使用从核阵列。如果使用中央处理器–图形处理器（CPU–GPU）系统来做类比，我们可以认为主核相当于其中的CPU，从核阵列相当于GPU。神威处理器将6个核组通过1个环形网络连接起来，并封装到1个芯片中。同时，每个节点上包含2个自主研发的高速网络芯片，所有的节点通过网络再进行连接。神威网络的架构是一个双层胖树[12]，最下面的一层把256个节点组织成一个超节点。超节点的内部节点间完全互连；超节点之间通过顶层网络互联，跨超节点的通信需要经过顶层网络。

出于成本考虑，两层的网络拓扑会带来网络裁剪。当通信跨越超节点时，带宽会降低到超节点内通信带宽的3/16。正如前文所述，因为训练对数据和模型的需求不同，将并行模式扩展到超大规模的异构系统时，需要考虑如何实现高效的映射。

大模型对内存计算和通信都有非常大的需求，新一代的超级计算机有PB级的内存空间、强大的计算能力、自主可控的高速网络以及灵活的通信接口。因此，新一代的国产超级计算机为大规模模型训练奠定了基础。

4 在国产超算平台上训练大模型

根据国产超算平台的特点，我们将大模型训练扩展到了千万核心规模。我们的工作核心在于验证当模型做到一定规模时，并行加速还面临哪些挑战，以及如何应对这些挑战。当前，我们的训练参数规模达到了174万亿，已达到人脑神经元突触数量的规模。针对前文所述的4个挑战，我们提出了一些解决方案。尽管这些策略并不一定是最优的，但仍可以为接下来进一步的优化扩展提供参考。

4.1 根据通信带宽采用合适的并行策略

新一代神威超级计算机的网络拓扑分为两层，其中上层是3/16的裁剪网络。此时，采用单一的并行策略，如数据并行或模型并行，会带来全局通信。受限于顶层带宽和参与通信的高进程数，全局通信带宽性能相比于小规模（如一个超节点以内）有显著的下降。那么，如何选择合适的并行策略来规避这类问题呢？

本质上，该问题属于高性能计算的基本问题。我们将应用程序的通信模式与底层的网络拓扑相结合，构建了一个性能模型，并通过性能模型判断并行策略在大规模下的执行效率。最终，如图4所示，我们在全机规模下的解决方案是在超节点内（256个节点）做数据并行，在跨超节点做MoE并行。相比于对称的方案（即超节点内做MoE并行，超节点间做数据并行），该方案可以将性能提高1.6倍。

4.2 实现高效不重复的参数存储和划分

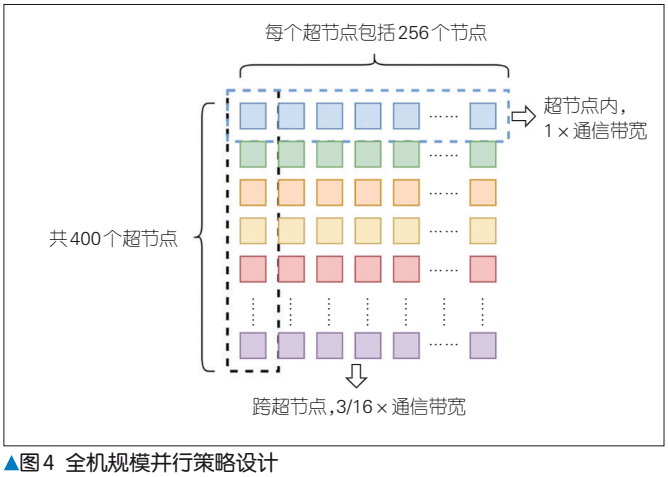
前文提到的数据存储问题，在本质上为如何将参数进行高效划分。在训练过程中，参数主要分为3部分：模型参数、梯度参数和更新参数。其中，模型参数指训练需要的计算参数；梯度参数指反向传播时每个模型参数的梯度；更新参数指优化器所需的参数，例如Adam优化器[14]中需要的参数。这些参数的规模都和模型参数的大小相当。此外，在混合精度训练中，我们还需要考虑混合精度策略需要存储的额外参数，如主参数与主梯度。因此，如何对这些参数做切分是一个复杂的问题。

我们以APEX O2[15]的混合精度训练为例。在训练过程中，模型的更新参数会占用75%的空间。如果简单地采用数据并行，模型参数在参与数据并行的节点中将被重复存储，浪费了宝贵的存储资源。微软于2019年提出的工作ZerO[16–17]就对数据并行的参数存储进行了优化，其核心思想是把参数分布在不同节点来进行更新，并在不影响通信量的前提下，将参数分布式存储到不同机器中。如图5所示，分布式参数更新有效降低了模型参数的内存。

我们对ZerO的方法进行了深度扩展。在裁剪网络中，采用分层训练策略，并针对全局和部分通信，扩展了通信算法，从而在不影响通信量的前提下，在分层并行策略中实现了分布式参数更新。

4.3 设计合适的混合精度策略

在新一代神威超级计算机中，神威CPU同时支持双精度和半精度计算。单精度计算使用双精度计算模拟，因此半精度性能为单精度的4倍。在机器学习训练中，广泛使用的



▲图4 全机规模并行策略设计

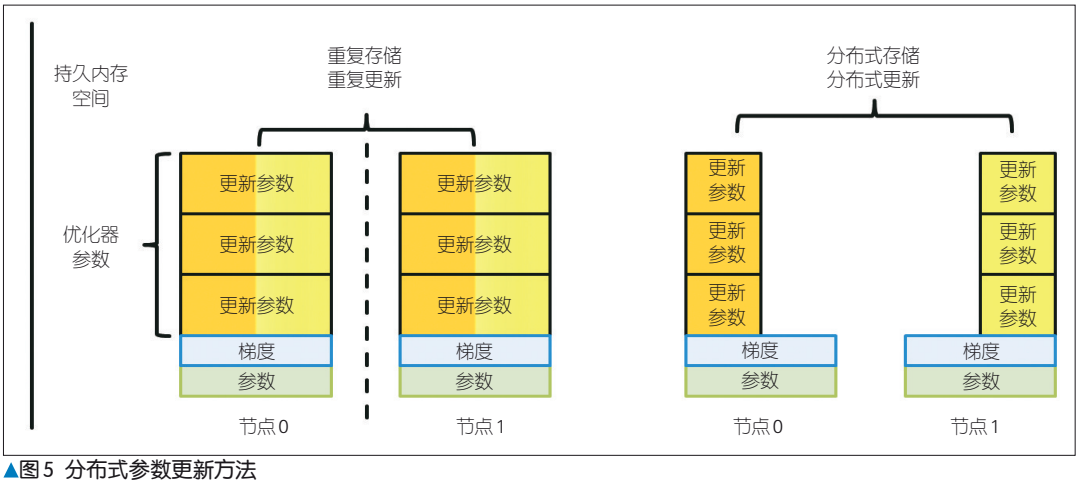
是单精度与半精度类型。此时，如何针对国产超级计算机设计合适的混合精度策略，成为影响性能优化的重要因素。

以 Multibox SSD[18]模型为例，对该模型参数分布的分析[19]如图 6 所示，横坐标表示参数数值的指数分布。红色区域表示 FP16 可以表示的范围。如果该模型的参数直接采用半精度存储，那么大部分参数会失真或无法表示。因此，我们采用了混合精度训练中常用的动态损失缩放技术[20]：在正向计算结束后对损失进行缩放，即乘以一个系数，然后在反向计算结束后再进行反缩放，并动态调整缩放系数，保证在不发生上溢出的情况下，数值尽量大。理想情况下，缩放后的表示范围如图 6 中绿色区域所示。此时，大部分参数都可以被半精度表示。

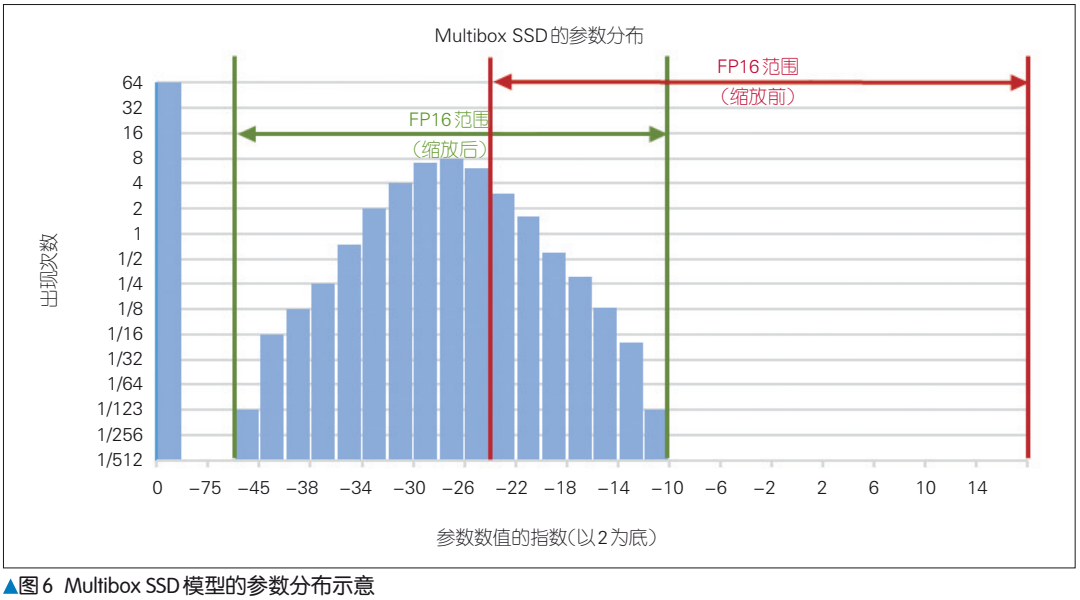
NVIDIA 的 APEX[15]系统是针对 NVIDIA GPU 的混合精度框架。APEX 提供了 4 种训练策略，分别是 O0（单精度训练）、O1（只优化计算）、O2（优化除更新外的部分）、O3（全部优化）。在 NVIDIA 平台中，使用较多的是 O2 策略。我们将 APEX 的策略移植到了神威平台。遗憾的是，这 4 种策略并不能直接在神威平台的大模型训练中使用。测试结果表明：O0 和 O1 策略可以正常收敛，但性能差；O2、O3 性能好，但不能收敛。

因此，我们提出了一种分层的混合精度策略，如图 7 所示。通过对不同层进行小规模验证，我们发现不同层对精度的要求不同。例如，训练中前馈网络（FFN）层和注意力层对精度不敏感，可以使用半精度计算、单精度更新；而其他层对精度敏感，因此可以使用单精度训练。这种策略在保证收敛的前提下，可以获得较高的训练性能。

此外，混合精度训练还包含更多的探索方向，例如，训练时间对训练精度的敏感性等。另外，在大的科学计算程序中，不同迭代使用不同精度。在迭代的开始、中间和最后，迭代都可以选择不同的混合精度策略。以动态的方式使计算和内存达到最高效率也是一个比较前沿的研究方向。



▲图 5 分布式参数更新方法



▲图 6 Multibox SSD 模型的参数分布示意

4.4 实现高效均衡负载

为了解决 MoE 模型负载不均衡的问题，我们提出了一个算法 SWIPE。算法的核心思想是当一个专家非常受欢迎，且样本数多于均值时，就将多余样本分配给其他专家，以保证全局的绝对均衡。这个策略虽然简单直接，但验证结果显示模型训练能够有效收敛。

4.5 国产系统底层优化

此外，为了适配新一代神威超级计算机，我们针对神威平台的软件系统、算子库、计算框架和基础设施等进行了大量的优化工作。例如，针对神威系统动态模式下内存分配的性能问题，

根据机器学习应用的特点，我们设计了高效内存分配器SWAlloc；针对神威平台，实现了一套高性能算子库SW-Tensor；同时深度重构了SWPyTorch，以支持各类自定义算子表示。该部分工作包含约60 000行C/C++代码，实现了100余个定制算子，有效地支持了模型训练。

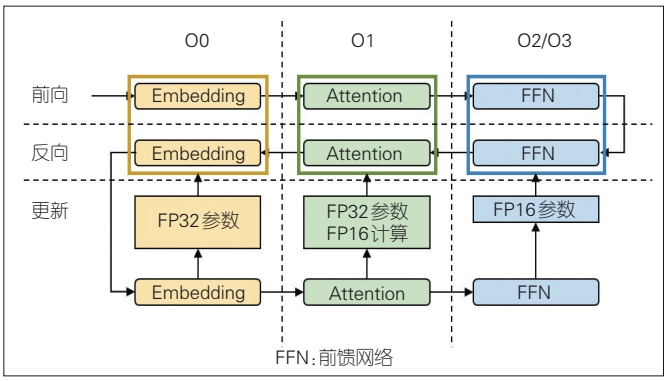
5 实验结果

最终，我们在新一代神威平台上整合了上述优化方法，促成了百万亿参数模型的训练。

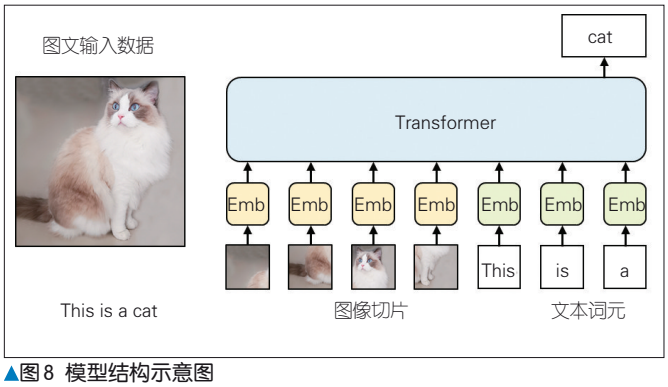
我们的模型基于阿里巴巴的中文预训练处理模型M6[21]，模型输入包含文本和图像数据。如图8所示，模型将输入图像拆分为多个块，并使用ResNet[22]等预训练模型来提取特征，然后将图像特征与词向量连接起来形成一个序列，再由Transformer模型处理并生成高级表示。

该模型在中文最大的多模态数据集M6-Corpus[21]上进行训练。数据来源包括百科全书、电子商务平台和其他类型的网页。最终处理的数据集的详细统计情况如表2所示，其中，其中，图像部分提取特征后大小为16 TB。

我们共测试了3个不同的参数量模型的性能，模型超参数如表3所示。其中，d_{model}表示模型的隐藏层规模，d_{ff}表示FFN内部层规模。在训练的3个模型中，MoDa-174T规模达到了百万亿参数量（我们获得的最大规模模型）。



▲图7 分层的混合精度策略



▲图8 模型结构示意图

▼表2 预训练数据集规模

数据集	文本规模/GB	图片规模/TB
百科	15.0	0.1
网页	70.0	1.5
电子商务	12.2	0.3
总计	97.2	1.9

▼表3 实验用到的测试模型超参数

模型	参数量/万亿	层数	头数	d_{model}	d_{ff}	专家数
MoDa-1.93T	1.93	12	8	4 096	4 096×12	400
MoDa-14.5T	14.50	10	8	4 096	4 096×18	2 400
MoDa-174T	173.90	3	8	4 096	4 096×18	96 000

训练过程与性能实验均在新一代神威超级计算机上进行。其中，通过对PyTorch的前向、反向和更新时间的检测，可以得到时间数据。浮点运算次数（FLOPs）通过神威系统的性能计数器得到，主核和从核分别计数，累加结果作为总浮点运算次数。

性能测试结果如表4所示。在1.93万亿参数规模时，模型训练的计算性能达到了1.18 EFLOPS（百亿亿次浮点计算每秒）；当模型增大至14.5万亿时，混合精度性能达到1.00 EFLOPS；当模型扩展到174万亿时，由于模型规模太大，模型训练无法进行混合并行，并行策略只有一维，因此通信性能会有明显下降，只达到0.230 EFLOPS。

图9和图10分别展示了MoDa-1.93T模型的弱扩展性与强扩展性测试结果。在实验中，我们将问题规模定义为专家个数和样本个数。在弱扩展性测试中，两者随测试规模的变化发生变化，即在测试规模为全系统的1/n的测试中，模型规模和样本个数均为全系统测试的1/n。因此，每个计算节点处理的样本数和模型规模均保持不变。测试结果表明，我们的系统可以做到接近线性的可扩展性。

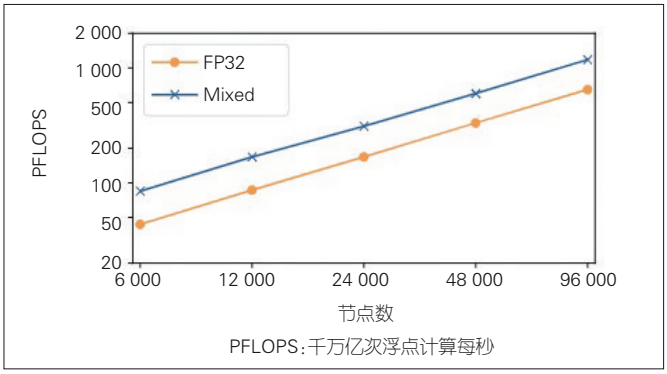
而在强扩展性测试中，我们固定专家个数和样本个数。此时，在测试规模为全系统的1/n时，每个节点处理的样本为全系统测试中每个节点的n倍。测试结果表明，从全系统1/16扩展到整机时，性能提升约为12倍。我们的系统表现出很好的扩展性。

为了验证系统的正确性，我们将MoDa-1.93T模型训练了500步，收敛曲线如图11所示。损失在500步后下降到3.46，根据参考文献[21]，我们认为训练接近收敛，并且可以验证系统的正确性。

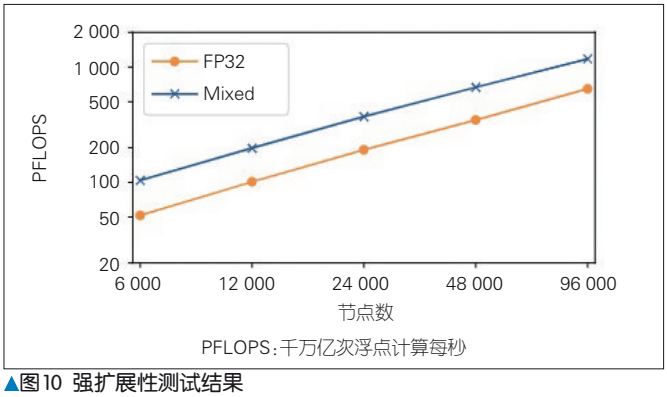
▼表4 性能测试结果

模型规模(参数量)/万亿	单精度性能/PFLOPS	单精度时间（每轮迭代）/s	混合精度性能/EFLOPS	混合精度时间（每轮迭代）/s
1.93	647	14.50	1.18	7.78
14.50	525	18.70	1.00	10.20
174	198	13.10	0.23	10.80

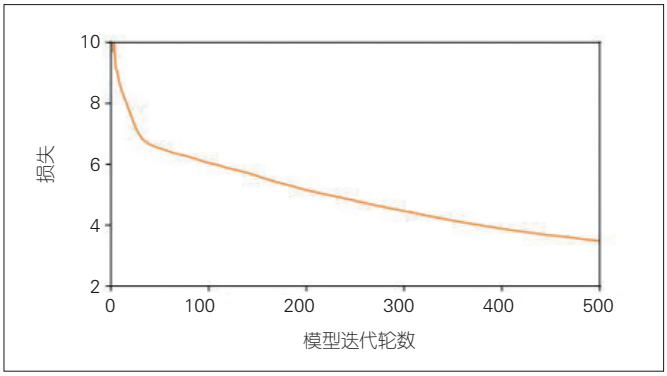
PFLOPS：千万亿次浮点计算每秒　EFLOPS：百亿亿次浮点计算每秒



▲图9 弱扩展性测试结果



▲图10 强扩展性测试结果



▲图11 MoDa-1.93T收敛曲线

6 结束语

我们把目前的工作命名为“八卦炉”[23]。“八卦炉”是一个高性能计算（HPC）和人工智能（AI）结合得较好的例子。在系统方面，结合以上提到的优化策略，我们进行了预训练模型加速。在此过程中，我们发现HPC存在许多挑战，例如网络裁剪等。本研究为将来探索新的大模型训练系统提供了宝贵的经验。

总的来说，该项研究主要针对国产系统的大型模型加速训练，并结合研究中遇到的问题提出了一系列解决方案。目前，这些方案仍处于研究阶段，希望有更多学者能一起参与讨论。

参考文献

[1] BROWN T B, MANN B, RYDER N, et al. Language models are few-shot learners [EB/OL]. [2022-01-10]. https://paperswithcode. com/paper/language-models-are-few-shot-learners
[2] DEVLIN J, CHANG M W, LEE K, et al. Bert: pre-training of deep bidirectional transformers for language understanding [EB/OL]. [2022-01-10]. https://paperswithcode.com/paper/bert-pre-training-of-deep-bidirectional
[3] LIU Y H, OTT M, GOYAL N, et al. RoBERTa: a robustly optimized BERT pretraining approach [EB/OL]. [2022-01-10]. https://paperswithcode. com/paper/roberta-a-robustly-optimized-bert-pretraining
[4] RADFORD A, WU J, CHILD R, et al. Language models are unsupervised multitask learners [EB/OL]. [2022-01-10]. https://paperswithcode. com/paper/language-models-are-unsupervised-multitask
[5] RAFFEL C, SHAZEER N, ROBERTS A, et al. Exploring the limits of transfer learning with a unified text-to-text transformer [EB/OL]. [2022-01-10]. https://paperswithcode.com/paper/exploring-the-limits-of-transfer-learning
[6] YANG Z L, DAI Z H, YANG Y M, et al. XLNet: generalized autoregressive pretraining forlanguage understanding [EB/OL]. [2022-01-10]. https://paperswithcode.com/paper/xlnet-generalized-autoregressive-pretraining
[7] RADFORD A, NARASIMHAN K. Improving language understanding by generative pre-training [EB/OL]. [2022-01-10]. https://paperswithcode. com/paper/improving-language-understanding-by
[8] LEPIKHIN D, LEE H, XU Y Z, et al. GShard: scaling giant models with conditional computation and automatic sharding [EB/OL]. [2022-01-10]. https://paperswithcode. com/paper/gshard-scaling-giant-models-with-conditional
[9] FEDUS W, ZOPH B, SHAZEER N M. Switch transformers: scaling to trillion parameter models with simple and efficient sparsity [EB/OL]. [2022-01-10]. https://paperswithcode. com/paper/switch-transformers-scaling-to-trillion
[10] KAPLAN J, MCCANDLISH S, HENIGHAN T, et al. Scaling laws for neural language models [EB/OL]. [2022-01-10]. https://paperswithcode. com/paper/mutual-information-scaling-and-expressive
[11] SHOEYBI M, PATWARY M, PURI R, et al. Megatron-LM: training multi-billion parameter language models using GPU model parallelism [EB/OL]. (2019-09-17)[2022-01-10]. https://arxiv.org/abs/1909.08053v2
[12] LEISERSON C E. Fat-trees: universal networks for hardware-efficient supercomputing [J]. IEEE transactions on computers, 1985, 100(10): 892-901. DOI:10.1109/TC.1985.6312192
[13] FU H, LIAO J, YANG J, et al. The Sunway TaihuLight supercomputer: system and applications [J]. Science China information sciences, 2016, 59 (7): 1-16
[14] KINGMA D P, BA J. Adam: a method for stochastic optimization [EB/OL].

(2019-09-17)[2022-01-10]. https://paperswithcode.com/paper/adam-a-method-for-stochastic-optimization
[15] NVIDIA. Apex (A PyTorch Extension) [EB/OL]. [2022-01-10]. https://nvidia.github.io/apex/
[16] RAJBHANDARI S, RASLEY J, RUWASE O, et al. ZeRO: memory optimizations toward training trillion parameter models [C]//Proceedings of SC20: International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, 2020: 1-16. DOI: 10.1109/SC41405.2020.00024
[17] RASLEY J, RAJBHANDARI S, RUWASE O, et al. DeepSpeed: system optimizations enable training deep learning models with over 100 billion parameters [C]//Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. ACM, 2020: 3505-3506. DOI: 10.1145/3394486.3406703
[18] LIU W, ANGUELOV D, ERHAN D, et al. SSD: single shot MultiBox detector [EB/OL]. [2022-01-10]. https://paperswithcode.com/paper/ssd-single-shot-multibox-detector
[19] NVIDIA. Training with mixed precision [EB/OL].[2021-01-10]. https://docs.nvidia.com/deeplearning/performance/mixed-precision-training/index.html
[20] ZHAO R Z, VOGEL B, AHMED T. Adaptive loss scaling for mixed precision training [EB/OL]. [2022-01-10]. https://paperswithcode.com/paper/adaptive-loss-scaling-for-mixed-precision
[21] LIN J Y, MEN R, YANG A, et al. M6: a Chinese multimodal pretrainer [EB/OL]. [2022-01-10]. https://paperswithcode.com/paper/m6-a-chinese-multimodal-pretrainer
[22] HE K M, ZHANG X Y, REN S Q, et al. Identity mappings in deep residual networks [EB/OL]. [2022-01-10]. https://paperswithcode.com/paper/identity-mappings-in-deep-residual-networks
[23] MA Z, HE J, QIU J, et al. BaGuaLu: targeting brain scale pretrained models with over 37 million cores [C]//Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. ACM, 2022: 192-204

作 者 简 介

马子轩，清华大学计算机科学与技术系在读硕士研究生；主要研究方向为高性能计算、编译优化。

翟季冬，清华大学计算机科学与技术系长聘副教授、博士生导师，担任多个国际学术期刊编委；主要研究方向为高性能计算、性能评测和编译优化等；获中国计算机学会科学技术奖自然科学奖一等奖、《IEEE TPDS》杰出编委奖、CCF-IEEE CS青年科学家奖等。

韩文弢，清华大学计算机科学与技术系讲师，担任中国计算机学会 NOI 科学委员会副主席、中国科协“英才计划”计算机学科工作委员会委员等职务；主要研究方向为并行与分布式计算，特别是大数据处理系统和机器学习系统；发表论文10余篇。

陈文光，清华大学计算机科学与技术系教授，现为中国计算机学会会士、杰出讲者、副秘书长、青年科技论坛荣誉委员，并担任ACM中国理事会常务理事、北京计算机学会副理事长等职务；主要研究方向为操作系统、程序设计语言与并行计算；获国家科技进步奖二等奖1次，部级科技一等奖2次。

郑纬民，清华大学计算机系教授、中国工程院院士；长期从事高性能计算机体系结构、并行算法和系统研究；提出了可扩展的存储系统结构及轻量并行的扩展机制，发展了存储系统扩展性理论与方法，在中国率先研制并成功应用集群架构高性能计算机，在国产神威太湖之光上研制的极大规模天气预报应用获得ACM Gordon Bell奖；曾获国家科技进步奖一等奖1项、二等奖2项，国家技术发明奖二等奖1项，何梁何利基金科学与技术进步奖，首届中国存储终身成就奖；发表学术论文500余篇，编写和出版相关教材和专著10部。