智能算力核心基础系统软件的现状与展望

Status and Prospect of Intelligent Computing Core Basic System Software

郑纬民/ZHENG Weimin，翟季冬/ZHAI Jidong，翟明书/ZHAI Mingshu
（清华大学，中国 北京 100084）
(Tsinghua University, Beijing 100084, China)

DOI：10.12142/ZTETJ.202402002
网络出版地址：http://kns.cnki.net/kcms/detail/34.1228.TN.20240408.1037.005.html
网络出版日期：2024-04-09
收稿日期：2024-02-18

摘要：智能算力对中国人工智能技术的进步具有重要意义。发展智能算力平台，做好核心基础系统软件尤其重要。梳理了智能算力平台中的10个核心基础系统软件，对这些软件的全球现状进行了详细介绍，并探讨了当前中国算力平台上系统软件栈建设的机遇和挑战。

关键词：人工智能；智能算力；大模型；系统软件

Abstract: Intelligent computing power is of great significance to the progress of artificial intelligence technology in China. It is crucial to develop intelligent computing power platforms and optimize foundational system software. Ten foundational systems software within intelligent computing power platforms are analyzed, and a detailed overview of their global status is provided, as well as the opportunities and challenges in the construction of system software stacks on Chinese computing power platforms.

Keywords: artificial intelligence; intelligent computing power; large model; system software

引用格式：郑纬民，翟季冬，翟明书. 智能算力核心基础系统软件的现状与展望 [J]. 中兴通讯技术，2024，30(2)：2-8. DOI：10.12142/ZTETJ.202402002

Citation: ZHENG W M, ZHAI J D, ZHAI M S. Status and prospect of intelligent computing core basic system software [J]. ZTE technology journal, 2024, 30(2): 2-8. DOI: 10.12142/ZTETJ.202402002

随着人工智能（AI）技术的飞速发展，深度学习大模型在文本、图像、视频等数据的理解和生成任务方面展现出强大的能力。这些智能模型的诞生，离不开算力的发展。得益于芯片技术的发展，大模型能力通过不断扩大的规模训练实现了大幅提升。

由于大模型规模急剧扩大，智能算力已成为大模型发展的稀缺资源和关键因素。智能算力规模的提升需要在软硬件技术上取得突破，其中增强核心系统软件尤为重要。

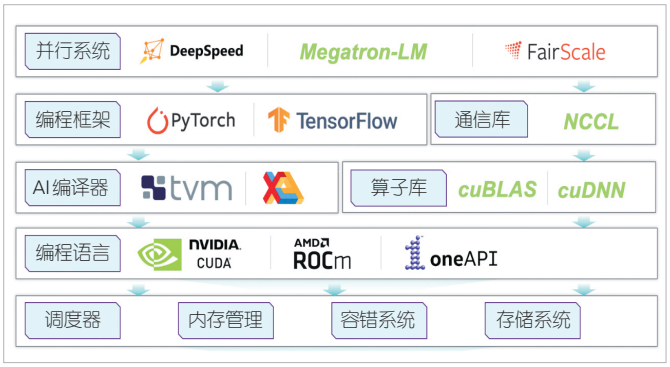
针对智能算力，我们总结出10个核心系统软件，如图1所示。它们在提升智能算力使用效率、降低大模型编程开发难度等方面起到了重要作用。根据功能，这些软件可分为四大类：

1）编程开发软件，包括编程语言和编程框架。它们是开发者和计算系统的交互接口，需要兼顾用户易用性和计算高效性。

2）并行加速软件，包括并行计算系统和通信库。它们使得大模型计算可以充分发挥多机多卡的分布式计算能力。

3）计算加速软件，包括算子库和AI编译器。它们使得大模型的计算负载能在加速卡等硬件上高效执行。

4）基础支撑软件，包括调度、容错、内存分配和存储系统。它们为大模型的易用性、高效性和鲁棒性做出了重要贡献。



▲图1 智能算力的10个核心基础软件

本文中，我们将对这些智能算力核心系统软件做出介绍，并通过回顾全球发展现状，探讨当前中国算力平台上系统软件所面临的机遇和挑战。

1 编程开发软件

大模型的编程开发软件主要有编程语言、编程框架。大模型的开发既需要适应模型的快速迭代，又需要利用各类硬件加速器来运行训练并推理所需的巨量计算。为满足这一需求，用于大模型的编程语言形成了模型开发语言与计算开发语言的分工，后者由前者通过深度学习编程框架来调用。目前主流的模型开发语言为Python语言，在其上实现的现代深度学习编程框架支持自动微分、Python绑定和直观的调试方法等功能，为算法开发者提供了良好的易用性。计算开发语言则由于各异的硬件体系结构呈现出百花齐放的状态。

1.1 编程语言

为了适应各异的硬件体系结构，许多硬件厂商需要使用各自不同的计算开发语言来编写其硬件加速器上的程序。目前，国际上用于大模型的主流硬件是英伟达公司的图形处理器（GPU），主要使用CUDA[1]作为其编程语言。另外，第三方编程语言Triton[2]也被广泛用于编写英伟达GPU上的大模型相关计算。中国具有代表性的硬件公司也采用了不同的编程语言，例如寒武纪的BANGC[3]、华为的AscendC[4]、壁刃的BIRENSUPA[5]、摩尔线程的MUSA[6]。这些编程语言中使用了不同的抽象来描述并行、访存与计算，可面向不同硬件进行针对性优化，但同时这也使不同硬件平台上的软件环境互不兼容。这给开发者灵活利用不同的硬件资源带来了挑战。

为了应对这一挑战，OpenCL[7]、SYCL[8]等编程语言致力于以更广泛的抽象来表达多种不同硬件加速器上的程序。更进一步地，Mojo[9]不仅计划统一多种硬件加速器，还力求统一模型开发语言与计算开发语言，以一套语言完成大模型的全流程开发。但是，统一的语言仍需不同编译器针对具体硬件来逐一实现，例如英特尔公司的oneAPI DPC++[10]就是针对英特尔及英伟达GPU的编译器实现的。下文中我们也将介绍编译器的发展现状。

1.2 编程框架

大模型的复杂度使得直接使用编程语言来开发模型变得十分困难，因此编程框架成为深度学习算法开发者与计算系统交互的界面，对深度学习训练和推理任务的计算性能产生了很大的影响。

Caffe[11]是最早用于深度学习训练任务的框架之一，具有自动微分和GPU支持的功能。它提供了一个带有Python和MATLAB绑定的C++机器学习库，使用内置的应用程序编程接口（API）来定义和训练模型。为了更好地表达各种模型结构和操作，Google开发了TensorFlow[12]框架。TensorFlow将神经网络模型表示为数据流有向无环图（DAG）的框架。TensorFlow已应用于许多领域，包括自然语言处理、计算机视觉、基于物理的AI应用等。然而，在TensorFlow中，用户需要先使用一段代码来描述模型的结构，再使用一段代码来描述训练的过程。这一较为抽象的工作模式给开发和调试工作带来了额外的困难。PyTorch[13]是由Meta开发的神经网络训练和推理框架。相比TensorFlow，PyTorch基于动态计算图，在代码实现上更灵活，调试更容易。作为最用户友好的框架之一，PyTorch在工业界和学术界都被广泛使用。然而，其动态特性使得许多优化（例如内核融合和计算图转换）难以实现。

百度PaddlePaddle[14]和华为MindSpore[15]等新一代框架结合了TensorFlow和PyTorch的特性，通过同时支持静态和动态模式的方法，既保证了易用性，又可以基于编译技术对计算过程进行更多的优化，甚至可以通过自动张量切分等技术实现自动的多卡并行训练。

然而，目前国际上围绕PyTorch平台已形成了丰富的软件生态。例如Megatron-LM[16]、DeepSpeed[17]、HuggingFace[18]等围绕Transformer模型的上层应用软件均基于PyTorch来实现。这些基于PyTorch生态的软件为模型开发和使用者带来了极大的便利，从而进一步鼓励了开发者为这个生态系统开发更多的软件。而Paddle等国产编程框架的用户基数较小，在软件生态上与PyTorch还有着较大的差距。为迎头赶上，厂商需持续带动用户，积极建设中国平台上的开源社区。

2 并行加速软件

近年来，随着大模型规模的扩展，多机多卡协同分布式计算成为大模型的计算范式。并行计算系统及其通信库支撑了大模型的高性能分布式计算。针对大模型训练和推理的计算特征，并行计算系统中的多种并行策略被提出，以充分利用计算资源；通信库则提供了跨机跨卡的通信能力，并力求高效实现这些并行策略所需的复杂通信模式。

2.1 并行计算系统

并行计算系统满足了大模型训练和推理对存储和计算的高需求。大模型的参数量往往超过单个加速卡的内存容量；其训练所需的计算量也远超单卡的计算能力。国际上主流的

并行计算系统包括DeepSpeed[17]、Megatron[16]等。这些系统主要采用了3种并行策略（数据并行、张量并行和流水线并行），并通过它们的灵活组合提高了计算效率。由于英伟达GPU的高计算能力和其通过nvlink[19]高速互连技术实现的高通信能力，Megatron等并行计算系统已经能够在英伟达平台上对GPT、Llama等大模型实现较高的计算效率。

针对中国软硬件的并行计算系统仍有进步空间，目前主要面临两个问题。首先是大模型计算的自动并行难度较大，特别是在涉及3D并行的复杂组合时，需要并行计算系统根据硬件特性选择合适的并行策略组合。目前，主流的自动并行算法面向英伟达硬件特性设计；针对国产硬件的自动并行算法有待提高。另外一个问题是新并行策略的研究。新兴模型和场景，例如混合专家模型、多模态大模型、长序列处理、基于人类反馈的强化学习等，为并行策略的设计带来了新的挑战，需要经典3D并行之外的其他并行策略来支持[20–21]。

综合而言，并行计算系统在处理Transformer类大模型的计算任务上已经相对成熟，尤其在英伟达硬件上表现出色。然而，中国的并行计算系统在易用性和计算效率方面仍有提升空间，需要研究面向国产平台的自动并行算法和新场景下的并行策略，以满足不断发展的科研和工业需求。

2.2 通信库

通信库提供了多卡间的通信能力，是实现并行计算的基础组件。大模型计算对通信库有3点要求：易用、高效、鲁棒。通信库需要根据大模型的训练与推理提供完备的通信模式接口，降低上层使用者的使用难度。通信库提供的通信模式接口应该是高效的，在大规模集群进行大模型训练时，通信往往成为瓶颈，同时通信库需要对大模型的通信模式、底层硬件拓扑做针对性的通信优化。通信库也需要是鲁棒的，在大模型训练与推理中，通信是频繁的，因此系统的鲁棒性很大程度上取决于通信库的鲁棒性。

在国际上，现有的大模型系统常用的通信库是英伟达的NCCL[22]。它提供了常见的集合通信模式，也能在英伟达硬件平台上实现不错的性能，因此受到多数大模型系统的青睐；然而，NCCL通信库在高效性以及鲁棒性上仍有改进空间。国际上的知名公司也针对各硬件平台做出了针对性优化，推出了自己的通信库：如微软的MSCCL[23]、英特尔的OneCCL[24]、AMD的RCCL[25]、Meta的Gloo[26]。

中国算力平台的通信库设计也处在活跃发展期。阿里设计了ACCL[27]，华为设计了HCCL[28]；清华大学的研究团队开发了“八卦炉”系统，在新一代神威超级计算机上探索了针对百万亿参数量大模型训练的通信库设计。综合考虑上层应用的计算负载、底层硬件的性能特征，“八卦炉”系统实现了中国国产平台上的高效通信。

3 计算加速软件

算子库和AI编译器在硬件上高性能实现了大模型所需的基本操作，有效支撑了上述的编程框架、并行计算系统等软件。算子库和AI编译器是大模型训练和推理的基础设施，对于提高计算效率、降低计算能耗具有重要意义。算子库为模型的基本操作奠定了基础，而AI编译器通常为现有算子库无法支持的计算负载生成代码。

3.1 算子库

算子库使得深度学习算法中常用的函数及模型结构的实现更加高效。通过提高底层硬件的利用效率，优化模型计算过程，算子库可以提高大模型的训练和推理效率。不同平台的算子库暴露了相似接口，为上层跨平台深度学习框架的构建提供了便利。

在国际上，最常用的硬件平台为英伟达GPU。英伟达公司在研发高性能GPU的同时，也提供了cutlass[29]、cuBLAS[30]、cuDNN[31]等高性能GPU加速库。这些算子库封装了矩阵乘等常用算子，对上层深度学习框架隐藏了Tensor Core等硬件实现细节，使得各种深度学习框架均可在该平台上实现较优的运行速度。算子库中也提供了基于模板的底层API以简化自定义算子开发。超微公司亦提供了rocBLAS[32]、hipDNN[33]等接口相似的面向超微GPU架构的算子库。

由于近两年美国的芯片限令，中国正在探索硬件的国产化，研发了一系列高性能人工智能芯片。特别是在高性能计算机方面，新一代的神威、天河等超级计算机采用了中国国产的处理器和加速器。这些国产加速器均提供了高性能的算子库，如申威架构的swBLAS、昇腾架构的AOL[34]库等。这些架构为国产硬件与TensorFlow、PyTorch等主流AI框架的兼容提供了基础保障，提升了国产硬件平台的易用性。但由于中国芯片的硬件架构与其他国家的GPU架构有较大差异，不能完全复刻其他国家算子库的开发方式，因此需要使用与国产硬件架构相匹配的算子开发方式以进一步提升算子库性能。

深度学习算子库在大模型的训练和推理中扮演着关键的角色，通过优化底层计算过程，提高了模型性能和效率。国际上已有成熟的深度学习算子库，中国也在通过持续的研发投入不断提升国产硬件的算子库性能。

3.2 AI编译器

AI编译器的主要用途和意义在于自动化地提高模型的执行效率和性能，同时保证优化前后程序的等价性。根据计算图这一通用的程序抽象，AI编译器的主要技术可分为高层次的计算图优化和低层次的算子优化。

目前存在多个被广泛使用的AI编译器系统，例如TVM[35]和XLA[36]。XLA通过静态图编译和优化技术，提供了高性能的执行引擎。它能够将TensorFlow等框架的计算图编译为高度优化的底层代码，利用冗余消除、等价计算图替换和算子融合等技术加速计算。此外，XLA具备内存优化和低精度计算等多种优化，进一步提高硬件加速器的利用效率。TVM提供了一套端到端的编译和优化工具链，旨在加速深度学习模型的推理和训练过程。TVM的核心思想在于针对用户给定的计算逻辑，通过自动调度计算的执行方案，自动化地优化和生成算子的执行代码。这使得TVM能够以相对较小的开发成本为不同硬件平台生成高效的代码，提供了一个跨平台的算子生成工具。

在中国，以PaddlePaddle[14]、Mindspore[15]为代表的通用深度学习框架均集成了大量的AI编译器技术，以提高程序的执行效率。针对中国硬件加速器种类多的现状，目前业界也出现了如InfiniTensor[37]、PowerFusion[38]等一系列新的框架，以提升国产硬件加速器的利用率。InfiniTensor框架探索了基于张量表达式推导的优化技术，尝试在更大的优化空间中发掘新的优化机会。PowerFusion通过细粒度算子拆分的方式，实现了更为高效的算子融合方案。

目前，AI编译器的发展仍然存在较大空间。一方面，针对国产加速器硬件利用率低的问题，加强对国产硬件平台的编译优化支持，提供更广泛的适配能力。另一方面，可以进一步优化编译器的自动优化算法，提高编译器生成的底层代码的效率和性能。此外，加强与中国深度学习框架的集成，提供更便捷的编译和优化工具，也具有重要意义。

4 基础支撑软件

大模型计算具有很高的复杂性，除了上述关键组件，还需要众多基础支撑软件，主要包括调度系统、容错系统、内存分配系统和存储系统。这些系统软件对大模型的易用性和高效性亦有重要意义。

4.1 调度系统

超大规模的模型训练不仅会带来巨大的计算成本，而且需要依赖大型的集群计算系统。在这样的背景下，充分挖掘和利用大规模集群计算能力变得非常关键，而一个高效且稳定的调度系统可以大幅度降低训练成本并显著提升训练效率。具体而言，调度器作为大规模集群系统的核心，主要有以下3个功能：

第一，弹性调度。调度器能够根据大模型训练任务的需要和集群的当前状态，动态分配资源，从而确保训练过程中资源的最优化利用。

第二，资源管理。调度器负责集群各种资源的规划和管理。大模型训练对计算、存储有着极大的需求，因此各种不同的资源，包括CPU、GPU和存储等要进行协同工作。调度器需要确保在有限的资源下最大化利用各个训练任务的性能，避免资源浪费。

第三，队列管理。调度器能够根据设定的优先级对训练任务进行排序，管理多任务的执行队列。调度器结合不同任务的资源需求、优先级高低来进行资源在时间和空间上的分配，从而确保每个训练任务都能获取到足够的资源并及时完成。

在国际上，Kubernetes（K8s）[39]是目前主流大模型训练在大规模集群系统上的调度器。它简化了容器化应用程序的部署、扩展和管理工作，成为了大规模集群管理和调度的主流软件。而在当前大模型兴起的背景下，Kubernetes同样凭借它在大规模系统上的高适用性、GPU等资源的细粒度管理和弹性调度，以及训练任务部署的灵活性等，成为目前大模型训练主要使用的软件。

在中国，华为的ModelArts[20]等平台也提供了高效的人工智能开发环境和强大的集群管理功能。它支持弹性调度，可以灵活地管理包括英伟达GPU和国产加速器在内的多种硬件资源。但是，目前各家调度器仍存在一些尚未解决的问题，例如：大模型训练调度中硬件资源选择和不同并行训练策略的协同、多种训练任务的稳定性，以及效率管理等问题。

总体来说，尽管目前大规模集群调度器已经相对成熟，但为了满足不断发展的大模型训练需求，在大规模系统中更高效地实施调度策略仍然是值得进一步研究的内容。

4.2 容错系统

容错在当今AI领域扮演着至关重要的角色，其主要作用是确保训练和推理过程的稳定性及可靠性。在大规模系统运行过程中，难免出现软硬件故障或异常，有效的容错技术可以确保系统在面对各种异常情况时能够继续正常运行，避免训练终止，从而节约时间和资源，并提高大模型训练的效率。大模型训练的容错技术包括检查点恢复、动态调整学习率、硬件故障处理等，其中最常用的容错技术是基于检查点

的恢复机制。该机制在训练过程中定期保存大模型的中间数据，以便在发生错误或异常时能够快速恢复至上一个检查点。然而，随着当前大模型参数量增大（万亿级），训练所需的集群规模随之扩增（千卡至万卡），训练时间也更长（数星期至数月），训练期间软硬件故障或异常出现的频次增加，高成本的检查点读写开销将不足以支持大规模系统的有效容错。因此，一些研究工作尝试结合并行策略所引入的冗余性，以降低检查点的读写成本，实现低成本容错，为用户提供更加稳定和高效的训练环境。

在国际上，大模型训练的容错技术得到不断地发展和完善，诸如 TensorFlow-Extended[12]、PyTorch（Torch - Elastic）[13]、Horovod[21]、Ray[40]等典型人工智能训练框架均包含了针对分布式训练的容错机制。这些框架大多采用“监控器+任务”的机制对模型训练任务进行监控调度，以快速识别故障或异常的任务，并使用检查点恢复的机制重启异常任务。OpenAI、Google 已经成功在配备成千上万个 GPU 的集群上训练出 ChatGPT、Gemini、Sora 等大模型，足以证明其在大模型训练的大规模分布式容错方面具备丰富经验的。

中国目前在大模型训练容错技术上也具备较为丰富的经验。值得一提的是，“八卦炉”[41]训练框架支持在新一代国产神威超算系统上 10 万节点（57 万个核组、3 700 万核）的大模型训练，该框架同样采用了基于检查点的恢复机制。在如此庞大规模系统上训练时，平均每 1 h 就会出现一次故障。“八卦炉”采用分布式检查点技术将读写检查点的开销控制在 3 min 以内，实现了低成本的有效容错。此外，字节跳动、快手、商汤等公司各自研发的大模型训练框架中均实现了有效的容错机制。中国在大模型训练容错技术方面具备一定的实力和竞争力。

综上所述，大模型训练容错技术是大模型的关键系统软件，可以提高大模型训练和推理过程的效率和可靠性，为人工智能技术的发展和应用奠定更加稳定和可靠的基础。目前全球业界均具备在大规模系统上开展大模型训练的经验，大多采取基于检查点的恢复机制进行容错。随着大模型参数的扩增，训练所需的系统规模随之增加，低成本的大模型容错技术已逐渐成为一项重要的研究主题。结合并行策略、检查点校验算法与硬件容错等方法或将成为降低容错成本的潜在技术方案。

4.3 内存分配系统

大模型的计算需要庞大而复杂的数据支持，这也同时直接对应着硬件内存资源的需求。内存分配因此面临着前所未有的挑战。内存分配指的是在程序执行期间，动态地分配和管理内存资源的过程。这一过程旨在高效利用内存，避免资源浪费，同时保证程序的高效运行。内存管理策略通常包括静态分配和动态分配。静态分配在程序编译时完成，而动态分配则允许程序在运行时根据需要分配内存。例如，以 TensorFlow 为代表的框架，采用的就是静态分配策略，而以 PyTorch 为代表的一系列框架采用的是动态分配策略。相比较来说，静态分配效率高，但框架受限严重，不易用；动态分配效率较低，但框架灵活。当前最为主流的计算框架均基于动态分配。

大模型系统软件的内存分配面临多重挑战。首先是数据规模，大量的数据和模型，需要巨大的内存资源来处理。通常来说，硬件的内存资源决定了模型规模的上限，因此内存资源的利用效率极为重要，也给内存分配带来了挑战。其次是性能，大模型训练需要非常频繁的申请与加速器上内存资源的释放。与此同时，加速器的直接内存的分配时间也各不相同。相比于 CPU 来说，加速器的直接内存需要更多的额外时间。因此，如何设计内存分配器，提高内存分配过程的速度，尽量少地直接分配硬件内存资源，都给大模型应用带来了巨大的挑战。

为了应对这些挑战，研究人员针对大型模型应用开发了一系列内存分配策略和优化技术。通过利用内存池技术预先分配内存块，有效减少了内存碎片和直接分配的时间开销。例如，清华大学的研究团队开发的 swAlloc[42]内存分配器，专门针对神威芯片的特性和大模型应用需求，设计了特殊的内存分配策略。这一策略解决了在神威系统上内存分配效率和速度的核心问题，为新一代神威超级计算机支撑的大规模训练系统“八卦炉”奠定了坚实的基础。

此外，在涉及多台机器的大规模应用场景中，内存资源的高效分配和利用显得尤为关键，同时也蕴含着巨大的优化潜力。特别是面对混合专家（MoE）模型等复杂的新型模型结构时，如 DeepSpeed[17]所采用的 ZERO 优化器以及“八卦炉”[41]使用的 PARO 优化器，均在减少模型内存需求和提升训练效率方面展示了能力。在这种复杂环境下，进一步提高内存资源的使用效率，成为了大模型系统研究中最为关键的研究内容之一。

大模型系统软件的内存分配是确保性能和效率的关键因素。面对大模型系统软件在规模和性能等方面存在的挑战，设计研究合适的内存分配策略和优化技术至关重要。

4.4 存储系统

在大模型训练中，存储系统不仅可以保存数据，而且在

保障训练效率和稳定性方面也发挥着至关重要的作用。随着模型规模的不断扩大，如Meta的Llama-2 70B模型涉及的数据量高达约8 TB，这对存储系统也有着更高的要求。这些数据在训练过程中将被频繁且随机地读取。同时，为了模型的持续迭代和优化，新的数据不断被添加到训练集中。此外，在大模型训练过程中可能会遇到的硬件故障或算法错误要求存储系统必须具备有效的容错机制，例如通过定期创建模型参数的检查点来实现错误恢复，保证训练的连续性和数据的完整性。因此，为了支持高效的大模型训练，存储系统需要综合优化并发读效率、异步写效率和检错纠错等能力。

近年来，为了应对这些挑战，业界已经有了不少研究和进展。例如，Google在训练Gemini模型时，将传统硬盘检查点更换为内存检查点，并设置冗余存储，这样既减少了写入时间，又确保了在部分节点出错时能够恢复完整的参数和优化器状态。中国的MegaScale[43]万卡训练系统引入了两阶段检查点机制，即首先将GPU显存状态传输至CPU内存，随后异步将数据传输至分布式文件系统。这种设计大幅提高了训练效率，同时保持了GPU的计算任务和存储系统之间的非阻塞传输[44-45]。

尽管在存储系统方面已取得一定的进展，但如何在提升效率、保障训练稳定性与优化训练结果之间找到最佳平衡点，依然是一个值得深入研究和探讨的重要课题。

5 结束语

高效的系统软件是发挥底层硬件性能的必要条件。中国智能算力平台的硬件能力已经接近国际领先水平，但是其上的核心基础系统软件仍有较大的进步空间。为了提升中国算力平台竞争力，更好地服务大模型预训练等重要应用场景，清华大学的研究团队在核心基础系统软件层做了深入研究。2021年，清华团队开发了“八卦炉”系统，对上述智能算力软件栈的10个核心基础软件深入优化，成功在新一代神威超级计算机上高效支持了百万亿参数量大模型预训练任务。目前，“八卦炉”系统仍在持续迭代，在更多的国产芯片平台（天数、沐曦、壁仞、寒武纪等）上深度优化核心软件栈，为充分发挥国产算力硬件能力做好软件支持。

参考文献

[1] NVIDIA Corporation. CUDA toolkit [EB/OL]. [2024-02-20]. https://developer.nvidia.com/cuda-toolk it

[2] TILLET P, KUNG H T, COX D. Triton: an intermediate language and compiler for tiled neural network computations [C]// Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages. ACM, 2019: 10 - 19. DOI: 10.1145/3315508.3329973

[3] 中科寒武纪科技股份有限公司. 寒武纪基础软件平台 [EB/OL]. [2024-02-22]. https://www. cambricon. com/index. php? m=content&c=index&a=lists&catid=71

[4] 华为技术有限公司. Ascend C：面向算子开发场景的编程语言 [EB/OL]. [2024-02-22]. https://www.hiascend.com/zh/ascend-c

[5] 上海壁仞科技股份有限公司. BIRENSUPA™软件开发平台 [EB/OL]. [2024-02-22]. https://www.birentech.com/product_details/1.html

[6] 摩尔线程智能科技（北京）有限责任公司. MUSA SDK [EB/OL]. [2024-02-21]. https://developer.mthreads.com/musa/musa-sdk

[7] STONE J E, GOHARA D, SHI G C. OpenCL: a parallel programming standard for heterogeneous computing systems [J]. Computing in science and engineering, 2010, 12(3): 66-73

[8] The Khronos® SYCL™ Working Group. SYCL™ 2020 Specification (revision 8) [EB/OL]. [2024-02-24]. https://registry. khronos. org/SYCL/specs/sycl-2020/html/sycl-2020.html

[9] Modular Inc. Mojo - the programming language for all AI developers [EB/OL]. [2024-02-22]. https://www. modular. com/max/mojo

[10] Intel Corporation. Intel® oneAPI DPC++ Library. [EB/OL]. [2024-02-22]. https://www. intel. com/content/www/us/en/developer/tools/oneapi/dpc-library.html

[11] JIA Y Q. Caffe [EB/OL]. [2024-02-22]. https://caffe.berkeleyvision.org/

[12] Google Inc. Tensorflow [EB/OL]. [2024-02-22]. https://www.tensorflow.org/

[13] Meta Platform Inc. Pytorch [EB/OL]. [2024-02-23]. https://pytorch.org/

[14] 百度在线网络技术（北京）有限公司. 源于产业实践的开源深度学习平台 [EB/OL]. [2024-02-22]. https://www.paddlepaddle.org.cn/

[15] 华为技术有限公司. 昇思 MindSpore [EB/OL]. [2024-02-22]. https://www.mindspore.cn/

[16] SHOEYBI M, PATWARY M, PURI R, et al. Megatron-lm: training multi-billion parameter language models using model parallelism [EB/OL]. [2024-02-25]. https://arxiv. org/abs/1909.08053

[17] Microsoft Corporation. Deepspeed [EB/OL]. [2024-02-25]. www.deepspeed.ai

[18] Huggingface Inc. Huggingface [EB/OL]. [2024-02-25]. https://huggingface.co/

[19] NVIDIA Corporation. NVLink and NVSwitch [EB/OL]. [2024-02-25]. https://www.nvidia.com/en-us/data-center/nvlink

[20] HE J A, ZHAI J D, ANTUNES T, et al. FasterMoE: modeling and optimizing training of large-scale dynamic pre-trained models [C]//Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. ACM, 2022: 120-134. DOI: 10.1145/3503221.3508418

[21] ZHAI M S, HE J A, MA Z X, et al. SmartMoE: efficiently training sparsely-activated models through combining offline and online parallelization [EB/OL]. [2024-02-21]. https://www. usenix. org/conference/atc23/presentation/zhai

[22] NVIDIA Corporation. NVIDIA collective communications library [EB/OL]. [2024-02-25]. https://developer.nvidia.com/nccl

[23] Microsoft Corporation. MSCCL Leaderboard [EB/OL]. [2024-02-25]. https://microsoft.github.io/msccl-leaderboard/

[24] Intel Corporation. Intel® oneAPI collective communications library [EB/OL]. [2024-02-20]. https://www. intel. com/content/www/us/en/developer/tools/oneapi/oneccl.html

[25] AMD Corporation. RCCL Documantation [EB/OL]. [2024-02-20]. https://rocm.docs.amd.com/projects/rccl/en/latest/api.html

[26] Meta Platform Inc. Gloo Documentation [EB/OL]. [2024-02-20]. https://github. com/facebookincubator/gloo/blob/main/docs/readme.md

[27] DONG J B, WANG S C, FENG F, et al. ACCL: architecting highly

scalable distributed training systems with highly efficient collective communication library [J]. IEEE micro, 2021, 41(5): 85–92. DOI: 10.1109/MM.2021.3091475
[28] 华为技术有限公司. 昇思通信 [EB/OL]. [2024–02–20]. https://www. mindspore. cn/docs/zh–CN/r1.10/api_python/mindspore.communication.html
[29] NVIDIA Corporation. cutlass: Fast linear algebra in CUDA C++ [EB/OL]. [2024–02–26]. https://developer. nvidia. com/blog/cutlass–linear–algebra–cuda
[30] NVIDIA Corporation. cuBLAS: Basic linear algebra on NVIDIA GPU [EB/OL]. [2024–02–26]. https://developer.nvidia.com/cublas
[31] NVIDIA Corporation. cuDNN: the NVIDIA CUDA deep neural network library [EB/OL]. [2024–02–20]. https://developer. nvidia.com/cudnn
[32] Advanced Micro Devices. rocBLAS library [EB/OL]. [2024–02–22]. https://github.com/ROCm/rocBLAS
[33] Advanced Micro Devices. hipDNN library [EB/OL]. [2024–02–23]. https://github.com/ROCm/hipDNN
[34] 华为技术有限公司. AI异构计算架构, 昇腾计算服务层, 昇腾算子库AOL [EB/OL]. [2024–02–20]. https://www. hiascend. com/software/cann
[35] The Apache Software Foundation. Apache TVM [EB/OL]. [2024–02–23]. https://tvm.apache.org
[36] OpenXLA. XLA (Accelerated Linear Algebra) [EB/OL]. [2024–02–23]. https://openxla.org/xla
[37] ZHENG L Y, WANG H J, ZHAI J D, et al. EINNET: optimizing tensor programs with derivation–based transformations [EB/OL]. [2024–02–20]. https://www. usenix. org/conference/osdi23/presentation/zheng
[38] MA Z X, WANG H J, XING J Z, et al. PowerFusion: a tensor compiler with explicit data movement description and instruction–level graph IR [EB/OL]. (2023–07–11)[2024–02–20]. https://arxiv.org/abs/2307.04995
[39] The Kubernetes Authors. Production–grade container orchestration [EB/OL]. [2024–02–20]. https://kubernetes.io
[40] 华为技术有限公司. AI开发平台_ModelArts_AI智能开放平台_人工智能平台_机器学习–华为云 [EB/OL]. [2024–02–20]. https://www.huaweicloud.com/product/modelarts.html
[41] The Linux Foundation. Horovod [EB/OL]. [2024–02–22]. https://horovod.ai
[42] Ray–project. Ray [EB/OL]. [2024–02–24]. https://www.ray.io
[43] MA Z X, HE J A, QIU J Z, 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. DOI: 10.1145/3503221.3508417
[44] 王豪杰, 马子轩, 郑立言, 等. 面向新一代神威超级计算机的高效内存分配器 [J]. 清华大学学报(自然科学版), 2022, 62(5): 943–951. DOI: 10.16511/j.cnki.qhdxxb.2022.22.007
[45] JIANG Z H, LIN H B, ZHONG Y M, et al. MegaScale: scaling large language model training to more than 10, 000 GPUs [EB/OL]. (2024–02–23) [2024–02–25]. https://arxiv. org/abs/2402.15627

作 者 简 介

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、并行算法和系统的研究，提出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戈登·贝尔奖；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，以及首届中国存储终身成就奖；发表学术论文400余篇，编写和出版相关教材和专著10部。

翟季冬，清华大学计算机系长聘教授、博士生导师，国家杰出青年科学基金获得者，国家重点研发计划项目负责人，清华大学计算机系高性能所副所长，中国计算机学会（CCF）高性能计算专委副主任、CCF杰出会员，ACM中国高性能计算专家委员会秘书长；主要研究领域包括并行计算、编程模型与编译优化；研究成果获IEEE TPDS 2021最佳论文奖、IEEE CLUSTER 2021最佳论文奖、ACM ICS 2021最佳学生论文奖等；获教育部科技进步奖一等奖、中国计算机学会自然科学奖一等奖、CCF–IEEE CS青年科学家奖；发表论文100余篇，出版专著1部。

翟明书，清华大学计算机系高性能所在读博士研究生；主要研究领域包括高性能计算、机器学习系统；曾担任清华大学学生超算团队队长，获得两次世界冠军；发表多篇论文。