分布式技术在大模型训练和推理中的应用

郑纬民
清华大学计算机科学与技术系，北京 100084

摘要
近几年，人工智能被广泛应用于多个领域，大语言模型（以下简称大模型）的“预训练-微调”成为人工智能的最新范式。分布式技术存在于大模型生命周期的每一环，为大模型的发展助力。在数据获取环节，针对海量小文件的存储问题，研发了文件系统SuperFS，能够同时满足低延迟和可扩展的要求。在数据预处理环节，针对从分布式文件系统读取数据开销大的问题，研发了高效大数据处理引擎“诸葛弩”。在模型训练环节，针对检查点文件读写性能差的问题，提出了分布式检查点策略，加快了检查点文件的读写速度。在模型推理环节，针对KVCache对存储系统的挑战，研发了高吞吐推理方案FastDecode以及大模型推理架构Mooncake。分布式技术的应用，使大模型能够充分利用计算资源，加快训练速度，有利于人工智能领域的发展。

关键词
分布式技术；大模型；海量小文件；大数据处理引擎；检查点；KVCache

中图分类号：TP319　　文献标志码：A　　doi: 10.11959/j.issn.2096-0271.2024056

Application of distributed techniques in large language model training and inference

ZHENG Weimin
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China

Abstract
In recent years, artificial intelligence has been widely applied in multiple fields, and the "pre-training and fine-tuning" of large models (LLMs) has become the latest paradigm of artificial intelligence. Distributed technology exists at every stage of the lifecycle of LLMs, providing support for them. In the data acquisition process, the file system called "SuperFS", was developed to address the storage issue of massive small files, which can meet the requirements of low latency and scalability. In the data preprocessing stage, an efficient big data processing engine called "Chukonu" was developed to address the issue of high overhead in reading data from distributed file systems. In the model training stage, a distributed checkpoint strategy was proposed to address the problem of poor read and write performance of checkpoint files, greatly improving the read and write speed of checkpoint files. In the model inference stage, a high-throughput inference scheme called "FastDecode" and a LLM inference architecture called "Mooncake" were developed to address the challenge posed by KVCache to storage system. The applications of distributed technology enable LLMs to fully utilize computing resources, accelerate training speed, and benefit the development of the field of artificial intelligence.

Key words
distributed technology, large language model, massive small files, big data processing engine, checkpoint, KVCache

0 引言

近几年，人工智能被广泛应用于多个领域，例如具身智能、自动驾驶、公共安全、科学计算等。人工智能经历了符号学习、统计学习、深度学习的阶段，如今进入大模型时代。大模型的“预训练-微调”成为人工智能的最新范式，基础模型支持众多领域任务。例如超大规模预训练模型GPT-3可以通过学习少量样例完成十余种文本生成任务，入选2021年《麻省理工科技评论》发布的“十大突破性技术”。

分布式技术存在于大模型生命周期的每一环。大模型包含4个环节，分别是数据获取、数据预处理、模型训练、模型推理，每个环节都与分布式技术紧密相关。在数据获取环节，需要获取并存储不同类型的原始数据，海量小文件的存储对文件系统提出了新的要求。在数据预处理环节，需要对抓取的海量数据进行删冗等预处理，而海量数据处理对底层大数据处理框架来说是一个新的挑战。在模型训练环节，模型训练任务负载重，硬件出错概率高，需要检查点操作，如何解决大模型检查点文件的读写问题是存储系统的一个难点。在模型推理环节，需要加载庞大的模型参数以及保存中间结果，这是存储系统需要解决的问题。

1 数据获取

1.1 海量小文件的存储挑战

数据对于大模型训练来说至关重要。大模型训练需要收集海量多模态数据，这些数据呈现两个特点：一是多模态，数据的类型包括文本、图像、音频、视频等；二是小文件非常多，任一模态的数据集包含数亿甚至数百亿个小文件，比如Youtube-8M数据集有近1亿个小于2 MB的音频文件，DALL-E数据集有近120亿个小于20 KB的图像文件，Common Crawl数据集有500亿个小于8 KB的网页文件。

海量小文件的存储对文件系统提出了新的要求，其中一个挑战是元数据的管理。小文件的数量越多，元数据的管理就越困难。存储100亿的小文件需要管理7 TB的元数据。对于大文件来说，90%的时间在处理数据，10%的时间在处理元数据；对于小文件来说，65%的时间在处理元数据，数据部分的时间开销仅占35%。海量小文件的元数据处理时间长，这就要求文件系统一方面扩展性要好，另一方面读取速度要快。为满足数据分析、模型训练等应用的需求，一般要求文件系统的读取延迟达到百微秒级。因元数据管理的瓶颈，现有系统的延迟仅达到毫秒级，比如Ceph文件系统。由此可知，元数据开销成为海量小文件存储的瓶颈，元数据的管理对文件系统的扩展性和延迟提出了更高的要求。

1.2 现有文件系统的问题

针对元数据的管理，现有分布式文件系统有两种架构，分别是元数据集中式管理架构和元数据分布式管理架构，如**图1**所示。元数据集中式管理架构，将元数据存储在一台服务器上，文件系统的访问延迟低。但是采用元数据集中式管理架构的文件系统能存储的最大文件数受限，比如HDFS（Hadoop distributed file system）能存储的最大文件数为1亿，Lustre文件系统能存储的最大文件数为40亿，这两个文件系统都无法横向扩展，都无法支持AI场景的海量文件。元数据分布式管理架构，将元数据存储在多台服务器上，可横向扩

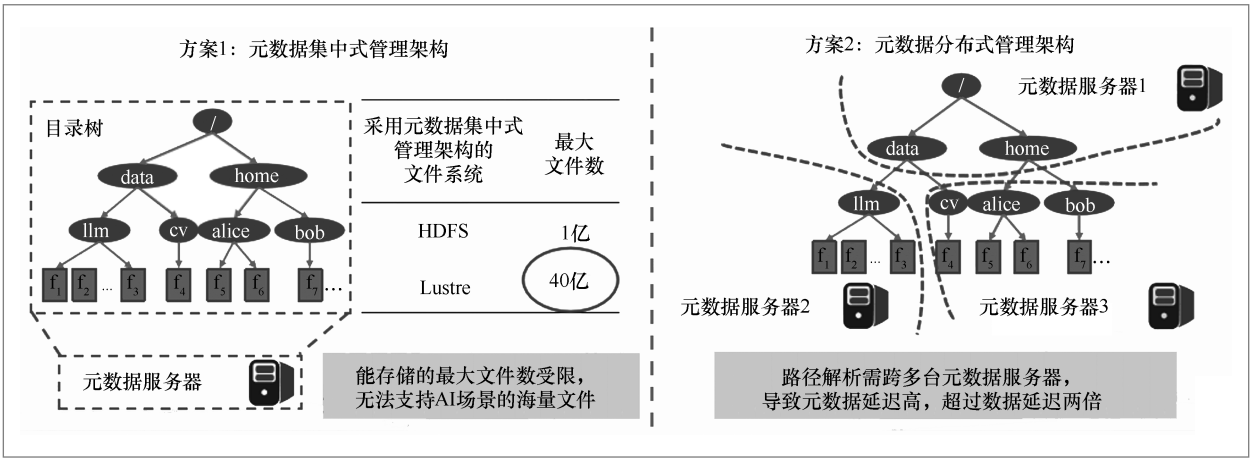

图 1　现有文件系统在处理海量文件时存在的问题

展。但是元数据分布式管理架构的路径解析需跨多台元数据服务器，导致元数据延迟高，甚至比数据延迟的两倍还高。现有分布式文件系统在处理海量小文件时无法同时满足可扩展和低延迟的要求。

1.3 文件系统SuperFS

为了解决以上问题，清华大学计算机科学与技术系高性能计算研究所自主研发了高性能文件系统SuperFS，结构如**图2**所示。SuperFS采取了解耦合的目录树存储策略，即解耦目录元数据和文件元数据，能够同时满足低延迟和可扩展要求。为满足低延迟要求，SuperFS将目录元数据集中在一台目录元数据服务器中，在目录元数据服务器本地完成路径解析，无跨网开销，实现了路径解析的低延迟。为满足可扩展要求，SuperFS将文件元数据分布到多台文件元数据服务器中，文件元数据服务器之间无共享，支持文件数目横向扩展。

实验结果表明，相比CephFS和Lustre文件系统，SuperFS的延迟能降低为现有文件系统的1/60～1/52。如**图3**所示，SuperFS在文件创建、统计、删除操作的延

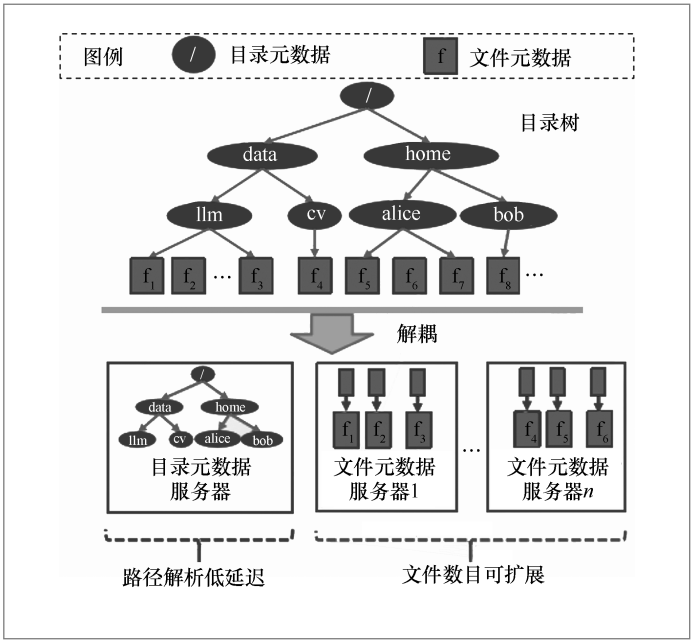

图 2　SuperFS 的结构

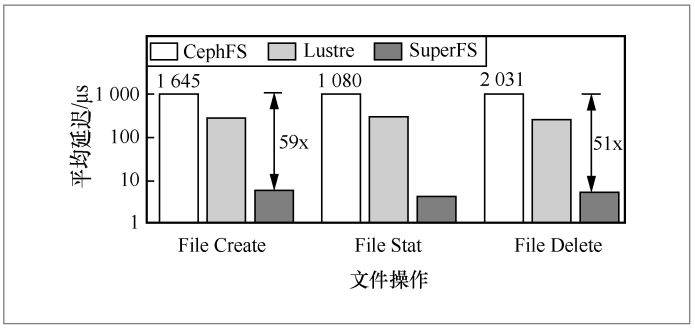

图 3　文件操作延迟

迟远低于CephFS和Lustre文件系统。此外，由清华大学计算机科学与技术系与鹏城实验室研制开发的“鹏城云脑II”存储系统蝉联超算存储500强（IO500）榜单全球第一。IO500是高性能计算领域针对存储性能评测的全球排行榜，是高性能计算领域最权威的榜单之一。SuperFS在元数据部分通过低开销索引、NUMA感知数据组织等技术，大幅提升了元数据性能，刷新了世界纪录。

2 数据预处理

2.1 海量数据预处理挑战

从各种平台、系统、网站收集的海量数据往往良莠不齐，直接用于模型训练很可能会影响模型的训练效果。为了获得高质量的数据样本，在模型训练前需要进行数据的预处理，如随机采样、数据解码、变换等操作。据谷歌数据中心统计，30%的训练时间用于数据预处理[1]。微软分析了9种常见模型，数据预处理最多占用65%的模型训练时间[2]。数据预处理开销逐渐成为大模型训练的瓶颈之一。如果数据预处理做得好，模型的训练时间将大幅减少。

数据预处理需要先从分布式文件系统读取数据，然后再处理数据，开销非常大。已有的方法通常以计算为中心，将需要处理的数据转移到进行计算任务的节点，但是待处理的数据可能分散在多个节点上，读取远端节点的数据会引入极大的网络开销。大模型的数据预处理对分布式技术提出新的挑战。

2.2 诸葛弩：以数据为中心的高效大数据处理引擎

针对上述问题，提出以数据为中心的理念，将计算任务动态地调度到所需数据所在的节点上，从分布式系统读取数据转换为从本地文件系统读取数据。“诸葛弩”（Chukonu）是清华大学高性能计算研究所自主研发的高性能计算引擎，应用了上述以数据为中心的理念，在大数据处理方面表现优异。诸葛弩是诸葛亮的发明，它一次能射出10根箭，可以连续发射、效率更高，因此将这个大数据处理引擎取名为“诸葛弩”。

目前，全世界应用最广泛的大数据处理系统是Apache Spark。这种系统生态好，具有高性能和广泛的适用性，可以处理更多类型的工作负载。但Spark也有两个缺点：一是Spark是用Java语言编写的，在大数据处理上性能不佳；二是Spark需要在内存中缓存数据，所需内存大小是待处理数据大小的20倍，比如处理1 TB的数据需要20 TB的内存，对内存大小的要求较高。诸葛弩引擎借鉴了Spark的优点，并针对以上两个问题进行改进：第一，不用Java，而是用C++编写，引擎性能得到提升；第二，减少内存需要缓存的数据量。

诸葛弩大数据处理引擎的架构如**图4**所示，其设计理念如下。

- 以数据为中心的执行模式：数据读入开销低，动态负载均衡。
- 兼容PySpark编程接口：对PySpark用户没有额外的学习成本。
- 采用大量编译优化技术：通过静态分析、算子融合、向量化、紧凑化数据排布等编译技术，降低数据处理开销。
- 提供良好的编程接口：提供基于C++ RDD编程接口，供性能工程师编写高性能计算模块，嵌入端到端PySpark数据预处理管线。

2.3 诸葛弩在大模型数据预处理中的应用案例

模糊删冗是大模型训练的数据预处理

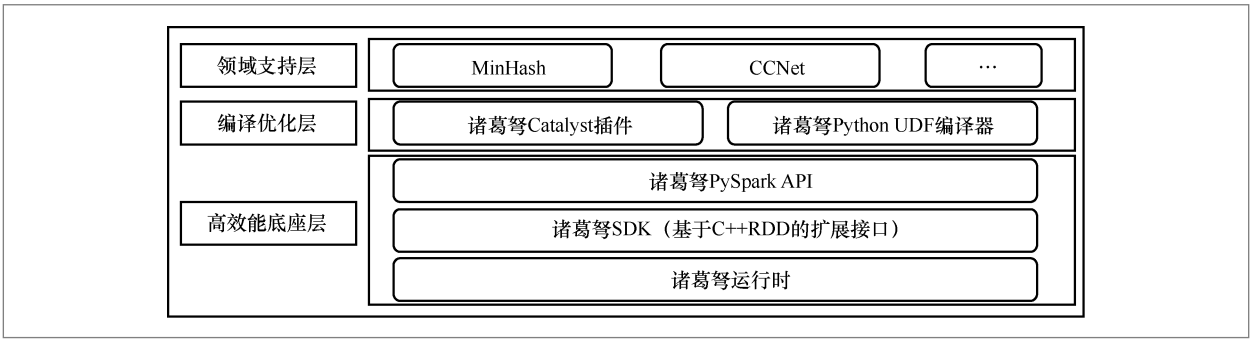

图 4 “诸葛弩”的架构

一个重要环节。流行MinHash算法可以进行模糊匹配，去除语料中的冗余数据，提高数据质量。团队基于诸葛弩系统开发了MinHash算法的高性能模块，并通过实验测试其性能。如图5所示，在144 GB数据规模下，诸葛弩所需的时间仅为PySpark的23.8%。目前，笔者团队正在推进诸葛弩大数据处理引擎在大模型公司的落地。

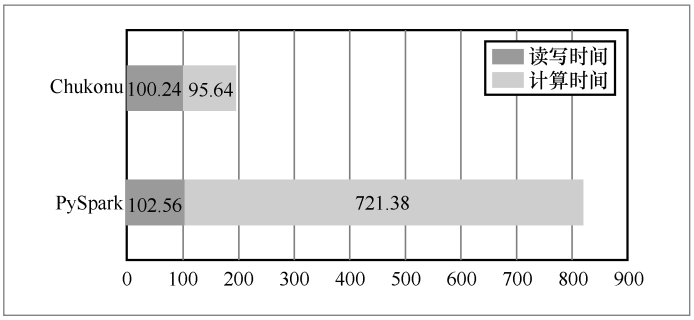

图 5 MinHash 时间开销 /s

3 模型训练

3.1 检查点文件的读写挑战

模型训练任务负载重，硬件出错概率高。为了提高模型训练效率，在训练过程中需要保存模型参数、中间结果，将其记录在检查点文件，出错后直接读取检查点文件，继续模型的训练。以神威平台10万卡规模训练万亿参数量模型为例，在训练过程中，平均每小时会发生一次硬件、软件错误。但是写检查点需要耗费大量时间，这个万亿模型的参数检查点包含近12 TB的模型参数，若未经优化，写一次检查点需要花费3 h。大模型检查点文件的读写对存储系统来说是一个新的挑战。

影响检查点读写性能的核心因素是存储系统架构，而网络利用效率会直接影响存储系统性能。神威平台存储系统与计算网络系统共享同一套链路，新一代神威平台的存储与网络架构如图6所示。以神威平台10万卡规模万亿参数量检查点的读写为例，默认策略是在每个专家的0号进程上写数据。第一种方案是单超节点写，容易出

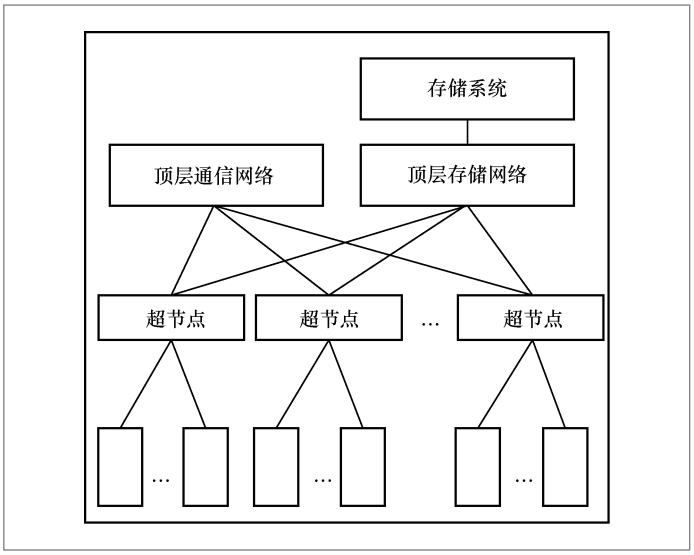

图 6 新一代神威平台的存储与网络架构

现负载不均的问题，所需时间超过10 h；第二种方案是跨超节点写，其进程数少，所需时间为3 h左右。由以上两个方案可知，默认读写策略性能差有两个原因，一是进程数不足，无法充分利用网络链路带宽；二是负载不均，即进程分布不均匀，无法利用所有交换机资源。

3.2 分布式检查点策略

针对以上问题，提出分布式检查点策略，即数据均匀分布在所有参与并行计算的节点上，如图7所示。为适应神威平台的存储架构特点，分布式检查点策略进行了以下调整：一是每个节点只需要写自己分配的一部分数据；二是在每个超节点交换机上有足够的I/O进程数；三是不同超节点的I/O请求被均匀划分。采用分布式检查点策略，10万亿参数量模型的检查点读写只需花费10 min左右。

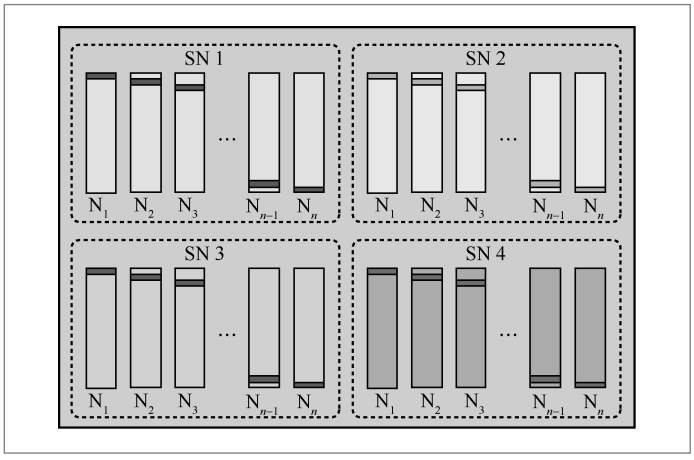

图 7 分布式检查点策略

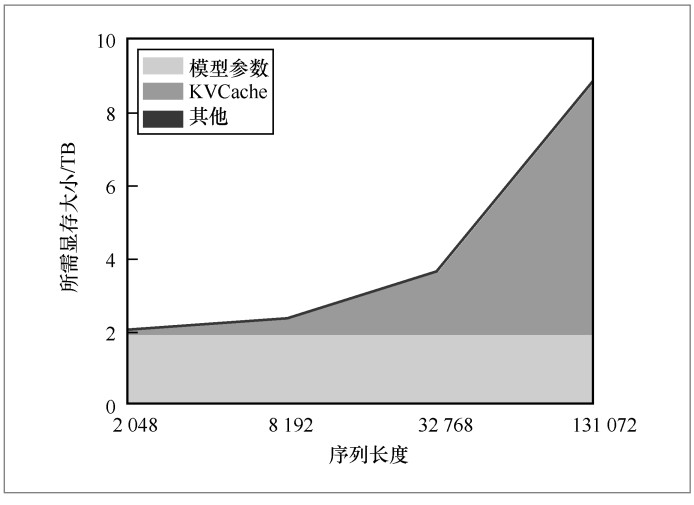

图 8 模型参数和 KVCache 所需显存大小

4 模型推理

4.1 KVCache的存储挑战

在模型推理过程中，需要在推理卡上存储模型参数以及模型推理的中间结果KVCache[3]。KVCache的大小能够达到数百TB乃至PB级别，相比模型参数，其数据量更大。复用KVCache将极大地节省算力，是大模型推理优化的关键。模型参数和KVCache所需显存大小如图8所示，KVCache的大小随着模型大小、序列长度以及批次大小的增大而显著增大。在长序列、多批次的场景下，KVCache的大小远超模型本身的大小。以万亿模型为例，模型大小为2 TB，需26个GPU存储参数，但KVCache大小为7 TB，需要86个GPU存储。

4.2 FastDecode：高吞吐大模型推理方案

如何为KVCache设计高容量、高带宽的存储系统？如何多次复用KVCache以减少对算力资源的需求？解决的思路是以存强算、以存换算。针对上述问题，提出了一种高吞吐推理方案FastDecode。如图9所示，传统解码方案在计算加速卡中进行顺序计算，批处理规模（batch size）小，计算效率低，显存占用量大。FastDecode[4]是一种WA分离推理方案，在计算加速卡

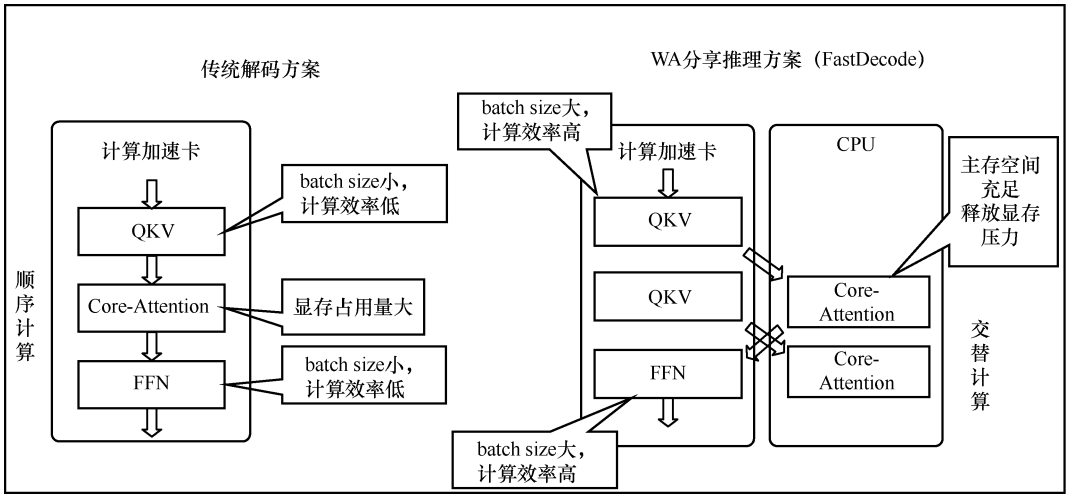

图 9　传统推理方案和 FastDecode 的结构

和CPU上交替计算。该方案的存储空间充足，减轻了显存压力，批处理规模大，计算效率明显提高。

FastDecode对于CPU的利用进行了更加大胆的尝试，其核心思想是使用闲置CPU和主存来处理KVCache。根据所需的计算/访存比例CPU更适合用于处理KVCache，而且CPU主存容量更大，可容纳更多KVCache，可同时处理更多序列。仅需4个CPU服务器，即可容纳8 TB的KVCache。FastDecode有两个显著优势，一是batch size不再受到KVCache显存占用限制，GPU利用率显著提升；二是聚合存储带宽高，KVCache处理吞吐量提升，成本降低。实验结果表明，相比广泛使用的大模型推理系统vLLM等，在使用单GPU进行推理时，FastDecode可将KVCache的可用存储容量提升100倍以上，大模型推理GPU吞吐量提升1.8~14倍，如图10所示。

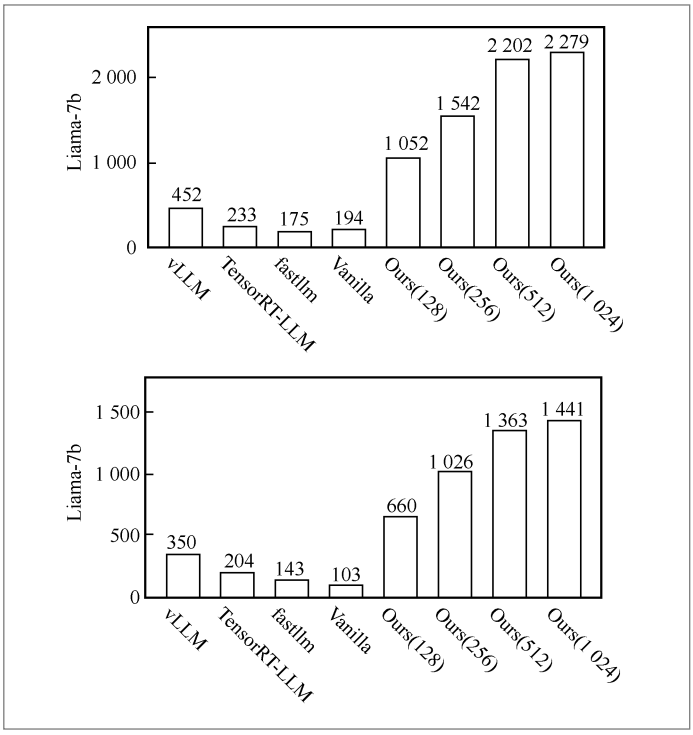

图 10　FastDecode 的性能

4.3 Mooncake: 大模型推理架构

Mooncake[5]是以KVCache为中心的大模型推理架构，其架构如图11所示。国产大模型Kimi使用的就是Mooncake推理架构，Kimi的底层推理架构承载其80%以上的流量。Mooncake通过以存换算，将Kimi的吞吐量提升了75%以上，并采用以超大规模分离式内存池为中心的KVCache

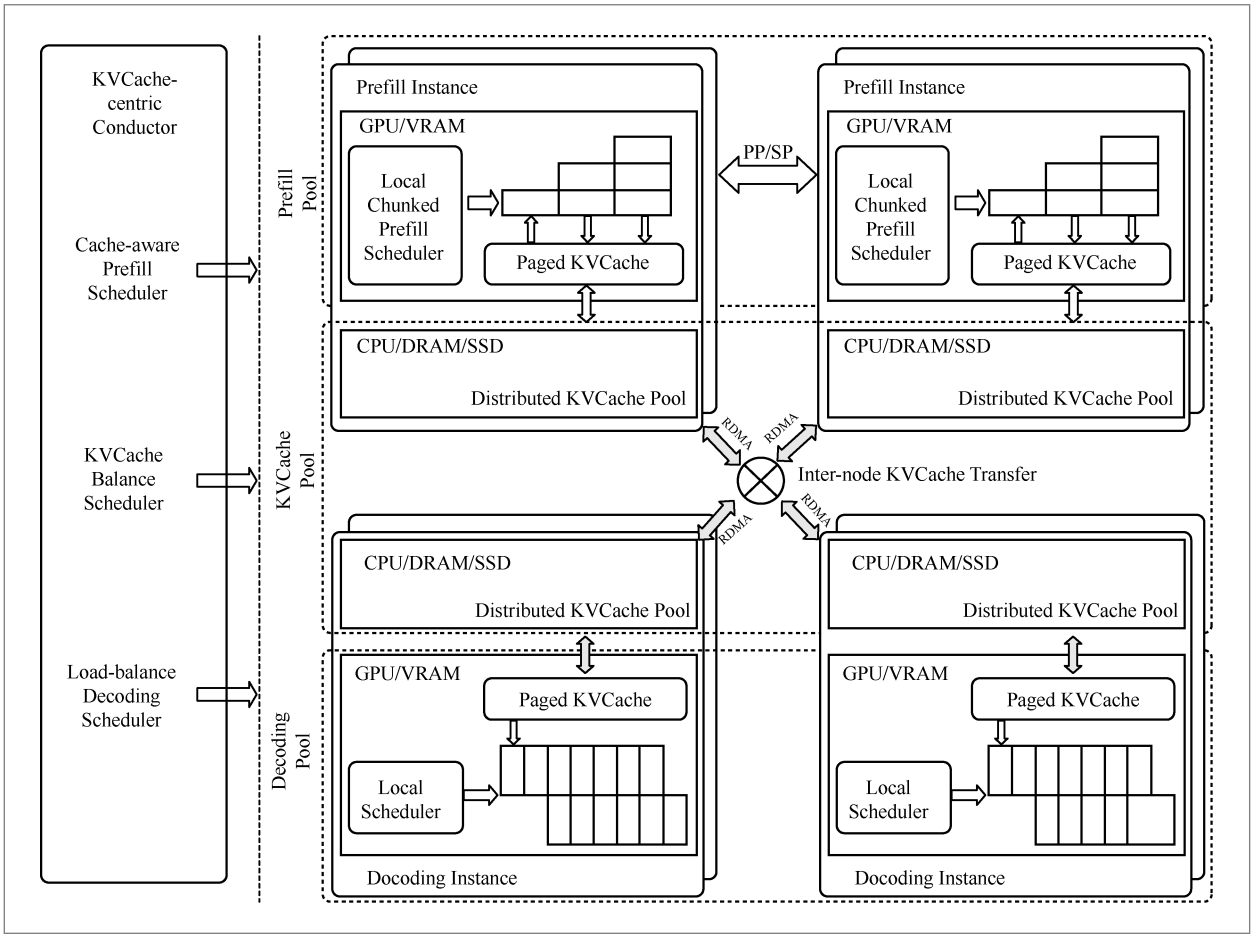

图 11 Mooncake 结构

缓存和调度，强调用户体验优先，面向过载场景进行调度。以大模型辅助阅读论文场景为例，不同的用户会对同一篇论文进行不同角度的提问。只要能将共享的、可复用的部分保存下来进行多次复用，就可以大幅度降低算力开销，所有查询均可复用，其中热点论文复用率高。

其中，KVCache缓存池是充分利用高速互连环境的多级透明缓存，存得多，传得快，性价比高，其结构如图12所示。KVCache充分利用当前GPU集群中闲置的内存容量和互联带宽，节省成本的同时降低响应延迟。透明多级缓存，未来可以进一步下沉到底层廉价存储。

在基于采样Kimi真实负载的模拟实验中，Mooncake使用10个预填充实例和10个解码模拟实例（Mooncake-[10P+10D]），vLLM使用20个标准模拟实例（vLLM-[20M]）。如图13所示，在满足服务水平目标（SLO）的前提下，Mooncake-[10P+10D]相较于vLLM-[20M]可多处理75%的请求。

5 结束语

基于“预训练-微调”的大模型技术成为当前人工智能领域的典型范式。分布式技术被广泛应用于大模型的整个生命周期（即数据获取、数据预处理、模型训

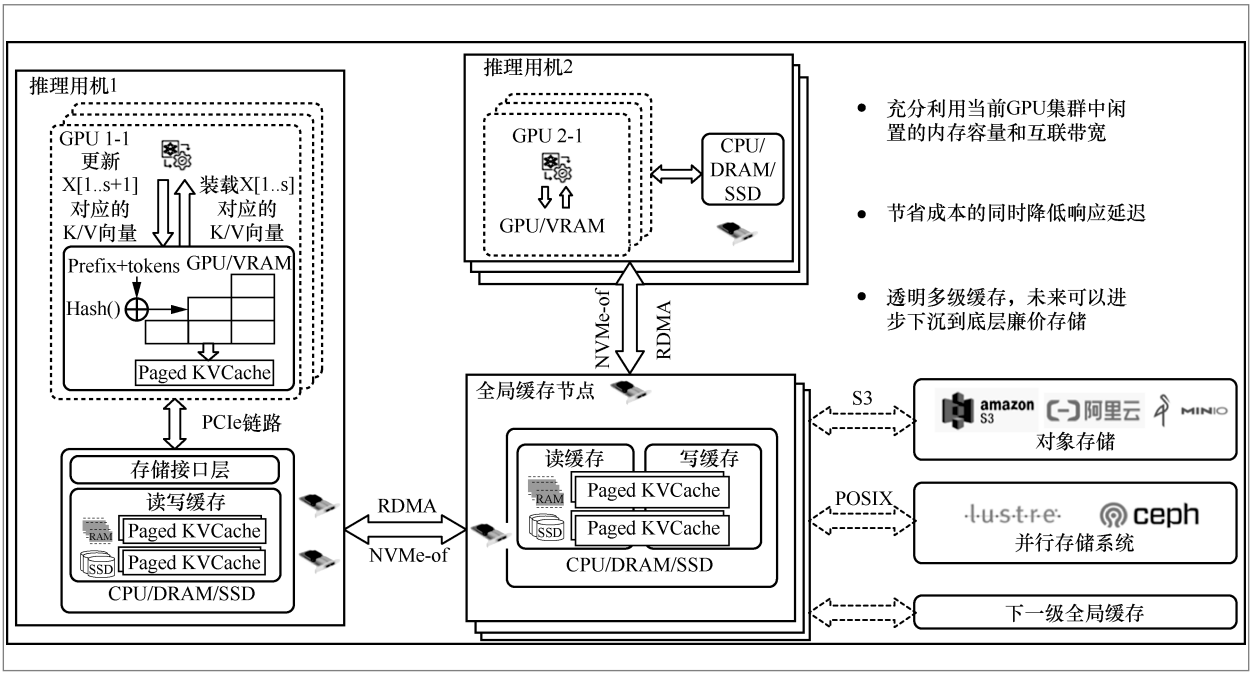

图 12　KVCache 缓存池的结构

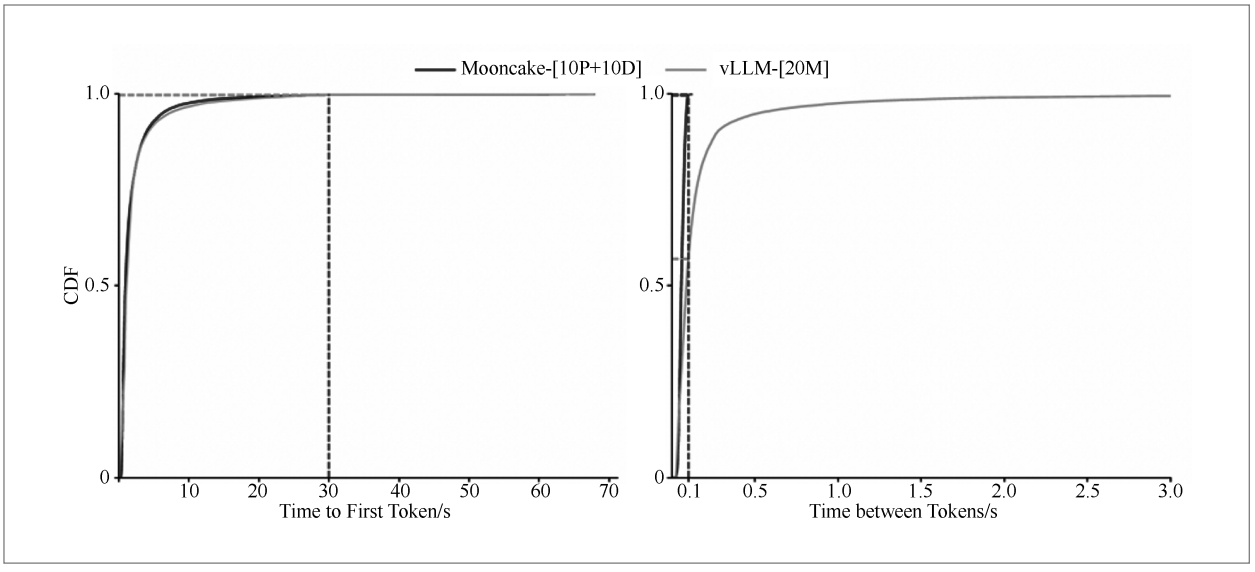

图 13　在真实负载下的请求 TTFT 和 TBT 分布

练、模型推理），是大模型发展的关键技术之一。本文总结了大模型生命周期每个环节面临的主要系统挑战，并针对这些挑战，提出一些初步的解决思路和方案，希望给大模型的发展和应用提供一定的参考。

参考文献:

[1] MURRAY D G, ŠIMŠA J, KLIMOVIC A, et al. tf.data: a machine learning data processing framework[J]. Proceedings

of the VLDB Endowment, 2021, 14(7): 2945-2958.

[2] MOHAN J, PHANISHAYEE A, RANIWALA A, et al. Analyzing and mitigating data stalls in DNN training[J]. Proceedings of the VLDB Endowment, 2021, 14(5): 771-784.

[3] KWON W, LI Z H, ZHUANG S Y, et al. Efficient memory management for large language model serving with PagedAttention[C]//Proceedings of the 29th Symposium on Operating Systems Principles. New York: ACM, 2023: 611-626.

[4] HE J A, ZHAI J D. FastDecode: high-throughput GPU-efficient LLM serving using heterogeneous pipelines[EB]. arXiv preprint, 2024, arXiv: 2403.11421.

[5] QIN R Y, LI Z M, HE W R, et al. Mooncake: a KVCache-centric disaggregated architecture for LLM serving[EB]. arXiv preprint, 2024, arXiv: 2407.00079.

作者简介

郑纬民(1946-),男,中国工程院院士,清华大学计算机科学与技术系教授,中国计算机学会第十届理事长,数博会专家咨询委员会委员,何梁何利科学与技术进步奖获得者,中国存储终身成就奖获得者,享受国务院政府特殊津贴,《大数据》主编。获北京市优秀教师奖和北京市教学名师称号,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,2016年获ACM戈登·贝尔奖。2019年当选中国工程院院士。主要研究方向为网络存储系统,长期从事网络存储系统科学研究、工程建设和人才培养,在存储系统扩展性、可靠性和集约性等科学问题和工程技术方面,取得了国内外同行认可的创新性成果;研制的网络存储系统、容灾系统和自维护存储系统在多个重大工程中发挥了重要作用。教学方面长期讲授计算机系统结构课程,2008年被评为国家级精品课程;已编写和出版计算机系统结构教材和专著10本,与合作者一起发表论文530余篇。

收稿日期: 2024-07-28
基金项目: 国家自然科学基金项目(No.U23A6007)
Foundation Item: The National Natural Science Foundation of China(No.U23A6007)