人工智能

大模型统一算力基座：多GPU并行与动态调度创新

邬烨磊，卢雅楠，黄煜骁，张安达，江天松
（中国电信股份有限公司上海分公司，上海 200120）

摘　要：随着大语言模型（LLM）需求的持续增长，如何高效部署和推理这些模型成为一个重要的挑战。提出了一种大模型统一算力基座的多GPU并行与动态调度方法，专门解决多GPU并行、动态调度及高效模型推理和统一部署管理的复杂问题。该系统通过将多种模型和计算资源整合到一个框架中，提供了可扩展、灵活且高吞吐量的模型部署与推理方案。展示了该方法如何通过最新技术（如连续批处理、paged attention和动态负载均衡）提高资源利用率、降低响应时延，并提升系统的整体吞吐量。证明了大模型统一算力基座的变革潜力，显著提升了传统独立部署的性能。

关键词：动态调度；大模型；vLLM
中图分类号：TP393
文献标志码：A
doi: 10.11959/j.issn.1000−0801.2025055

0　引言

近年来，大语言模型（LLM）在自然语言理解、代码生成等众多任务中展现了前所未有的能力。然而，在大规模部署这些模型时，面临着大量的挑战，包括如何高效利用资源、保持低时延和提供高吞吐量，特别是在负载变化时。

传统的大模型部署方式通常是为每个应用单独部署实例，这导致了资源的低效使用、软件栈的重复搭建以及运维开销的增加。随着模型规模和复杂度的增加，计算需求显著增长，导致推理和扩展出现瓶颈。本文提出了一种统一的算力基座多GPU并行与动态调度方法，旨在应对这些问题，特别是多GPU并行、动态负载调度和高效的资源管理。

1　大模型部署面临的挑战

1.1　资源碎片化与重复性

独立的模型部署往往为特定应用量身定制，导致资源使用的碎片化。每个项目需要独立设置推理环境，导致了软件栈的重复部署以及GPU资源的低效利用。由于缺乏集中化的管理，资源利用率不高，且难以应对工作负载的变化。

1.2　推理效率低与时延问题

先前，上海公司内部的大模型项目分散部署，推理效率低下，环境部署耗时费力，且推理软件栈重复搭建，导致开发成本高昂。同时，面对多种模型需求，算力资源极为紧张。以往的部署方式，如Ollama，在应对高并发时存在明显的性能瓶颈，无法进行多卡并行推理。此外，Ollama仅支持单卡推理，导致其在应对复杂工作流和多模型场景时难以胜任。处理像Qwen-32B（通义千问-32B）和Qwen-7B（通义千问-7B）这样的超大模型需要优化GPU并行和批处理机制，以避免较高的时延。在高负载下，传统的推理方法往往无法提供实时响应，时延峰值可高达14 s。

收稿日期：2024−11−01；修回日期：2025−02−10

1.3 多样化模型的扩展难题

企业往往需要处理各种类型的模型，从大语言模型到特定的嵌入模型（embedding模型）和重排序模型（reranker模型）。独立部署这些模型增加了运维负担，限制了系统的扩展性，并减缓了模型迭代速度。

2 系统架构

该系统基于模块化架构设计，具有灵活性和可扩展性，系统架构如图1所示。

- 数据API层：提供了一个兼容OpenAI的HTTP API，使用Swagger 2.0支持基于流的推理，实现实时交互。
- 请求分发层：通过最少负载和最低时延的策略实现GPU与服务器之间的负载均衡，确保请求的最优处理。
- 模型量化与推理层：支持最新的量化方法生成预训练变换器量化方法（quantization-aware training for GPT，GPTQ）和自适应权重量化方法（activation weight quantization，AWQ），并采用可变大型语言模型（variable large language model，vLLM）和大规模语言模型（large language model meta AI，llama.cpp）等推理加速框架进行批量流式推理。
- 缓存数据库层：使用Redis和基于前缀的缓存机制，减少冗余计算并加快重复查询的处理速度。
- 运行环境：基于Docker和多GPU服务

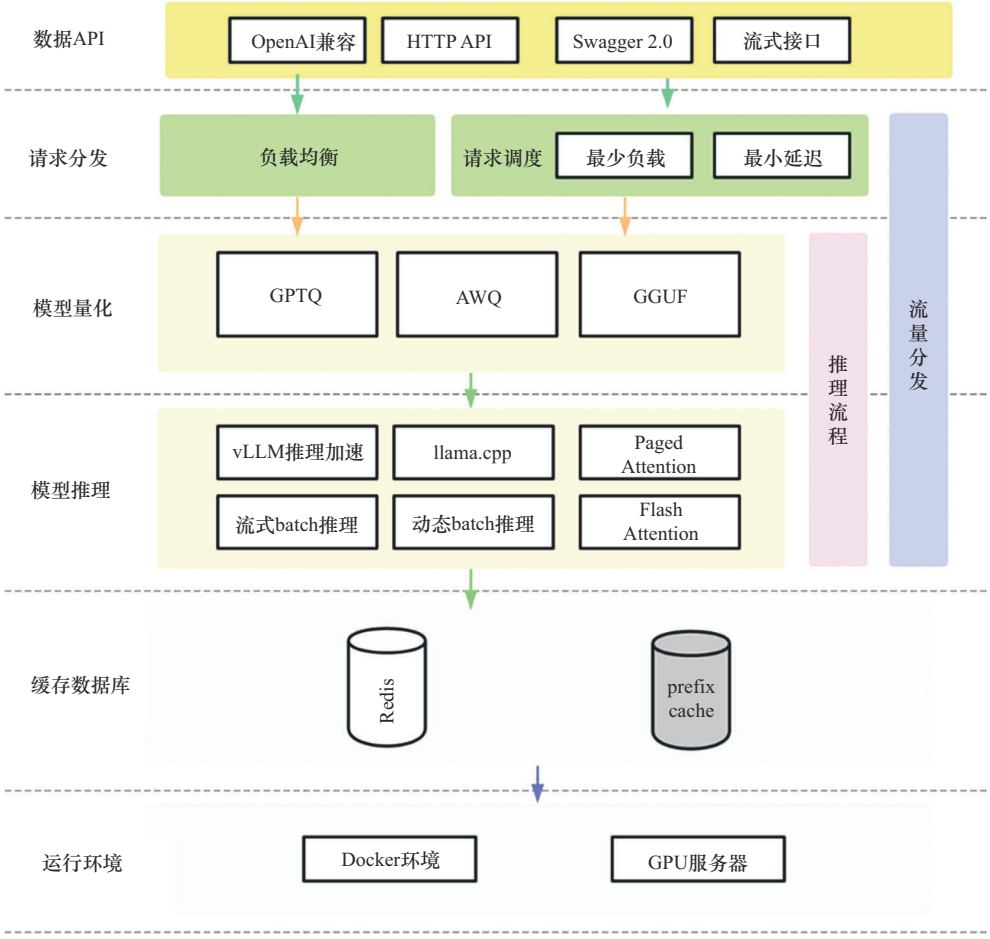


图1 系统架构

器集群，提供了便捷的部署与扩展能力。

3　统一算力基座的设计与实现

本文提出的统一算力基座方法通过将模型推理整合到一个共享平台中，提供了高效的动态调度和多GPU并行能力，解决了上述问题。

3.1　高性能接口服务

为了提升模型服务的通用性与易用性，本文在统一算力基座中实现了一个兼容OpenAI API的接口层。这个接口层不仅与OpenAI API保持高度兼容，还能够与其他开发框架或平台无缝集成，尤其是与可视化工作流编排平台的深度融合。这种设计使工作流编排平台的开发者得以利用他们熟悉的OpenAI API标准，轻松地调用基座中的各种大语言模型，无须为每个模型编写不同的集成代码。

3.1.1　实现细节

遵循OpenAI API的标准，包括completions、chat、embeddings等端点，提供统一的API访问方式。通过这套标准化接口，用户可以像使用OpenAI服务一样，访问部署在统一算力基座中的模型。具体而言，接口层负责接收用户请求，解析模型调用参数，并将其转发给对应的推理服务实例。这些推理实例可以是不同模型（如Qwen-32B或Qwen-7B），也可以是同一模型的不同版本或量化版本。

3.1.2　优势

（1）兼容性与简化集成。使用兼容OpenAI的标准API，开发者不需要重新学习新的接口标准。这意味着现有使用OpenAI API的应用程序或脚本可以直接接入统一算力基座，无须额外修改代码。对于上海电信内部的多个模型需求，无论是提供对话接口、文本生成，还是嵌入式表示的生成，均可通过一个统一的API层来访问不同的模型资源。这种方式大大简化了跨模型管理和调用的复杂性。

（2）开发与部署成本降低。通过标准化接口层，上海电信企业内部可以避免重复开发不同模型的API，从而节约开发资源。特别是在公司需要同时使用多个大语言模型时，开发者只需要与一个API对接，无须为每个模型分别设计、部署和维护不同的接口服务。此外，由于接口层与模型实例的解耦，模型的升级、切换或扩展可以在后台透明完成，而不影响前端应用的正常使用。

（3）性能提升。通过批处理、缓存以及负载均衡的多种优化手段，确保API层在处理高并发请求时仍能保持较高的性能水平。即使在高峰期，时延也控制在4 s以内，而在非高峰期，响应时间更是保持在1 s内。这样，企业的实时性要求可以得到充分保障，同时高效利用底层的计算资源。

3.2　基于vLLM的高性能推理实现

以往的部署方式，如Ollama，在应对高并发时存在明显的性能瓶颈，无法进行多卡并行推理。对Ollama的深入研究发现，其在并发处理和推理效率上存在以下几个缺陷。

（1）单卡限制。Ollama只能利用单卡进行推理，导致其在处理大模型时性能不足，无法充分利用多卡并行的优势。

（2）并发问题。Ollama在高并发场景下容易发生卡死现象，影响了支撑上层工作流编排平台的稳定性。

针对这些问题，本文引入了vLLM多卡并行推理框架，能够实现跨卡推理，并显著提升吞吐量。实验中，双卡部署Qwen1.5-32B模型的吞吐量达到600字/秒，四卡部署Qwen2-72B模型的吞吐量达到540字/秒。同时，vLLM具备更强的容错机制，能够在负载均衡的基础上快速响应单一服务实例的故障，确保服务高可用性。

3.3 动态批处理与模型推理优化

为了解决时延问题，本文采用了连续和动态的批处理机制。连续批处理会动态地将到达的请求分组成更大的批次并行处理，从而减少资源开销并提高模型吞吐量。此外，系统还集成了flash attention和paged attention等技术，以优化内存访问并加速推理。

3.3.1 连续与动态批处理

（1）连续批处理的概念

连续批处理（continuous batching）是一种根据请求的到达时间，动态调整批次大小的处理机制。与传统的静态批处理不同，连续批处理可以在请求到达后进行灵活的批次合并。这意味着在实际运行过程中，不必固定批次大小，而是通过监测请求的到达频率、资源利用率等指标，自动将多个请求打包为一个更大的批次进行推理，连续与动态批处理示意图如图2所示。这样可以最大化地利用计算资源，避免因小批次导致的资源浪费。

（2）动态批处理的优势

- 降低计算开销。将多个请求合并到一个批次中，可以减少每次推理时的启动、同步等开销。这使每个请求的处理效率得以提升，同时降低了单次推理的资源占用。
- 提升吞吐量。当请求较多时，批次中的请求量增大，模型推理能够在一次计算中处理更多的请求，这提高了系统的整体吞吐量。系统根据当前的负载自动调整批次大小，确保在高并发的情况下保持较高的推理效率。
- 优化响应时间。对于低并发场景，动态批处理能够避免长时间等待批次填满的问题。通过灵活的批处理机制，即使请求频率较低，系统也可以根据需求快速启动小批次推理，保证响应时间的可控性。

3.3.2 闪存注意力优化与分页注意力优化技术

除了批处理机制，算力基座还集成了多项针对内存优化的技术，以进一步加速模型推理，尤其是对于大规模Transformer模型。这些技术旨在通过优化内存访问的方式，减少模型推理中的内存瓶颈，从而提升整体推理性能。

（1）闪存注意力优化

采用闪存注意力（flash attention）优化技术专门针对Transformer模型中自注意力（self-attention）机制进行优化。传统的自注意力计算中，计算复杂度为$O(n^2)$，内存访问成本较高，尤其是在处理长文本序列时，容易造成显存瓶颈。采用flash attention优化通过重组计算过程，减少了显存中的中间变量存储需求，从而降低了内存开销并加速推理，flash attention优化示意图如图3所示。

- 减少内存访问。传统自注意力需要存储大量中间结果（如键值对矩阵），而flash attention优化的实现方式则减少了这些中间变量的存储需求，直接在寄存器中完成自注意力计算，显著降低了对

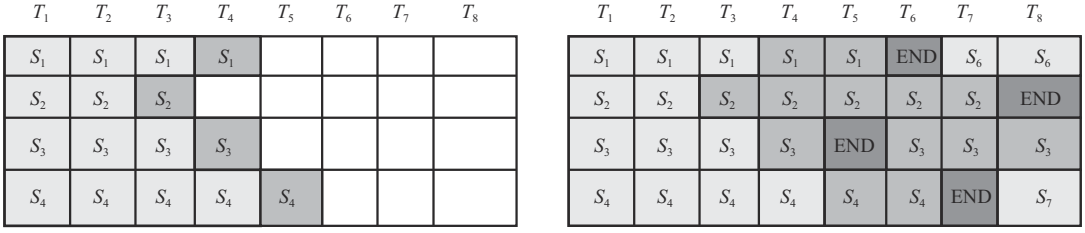


图2 连续与动态批处理示意图

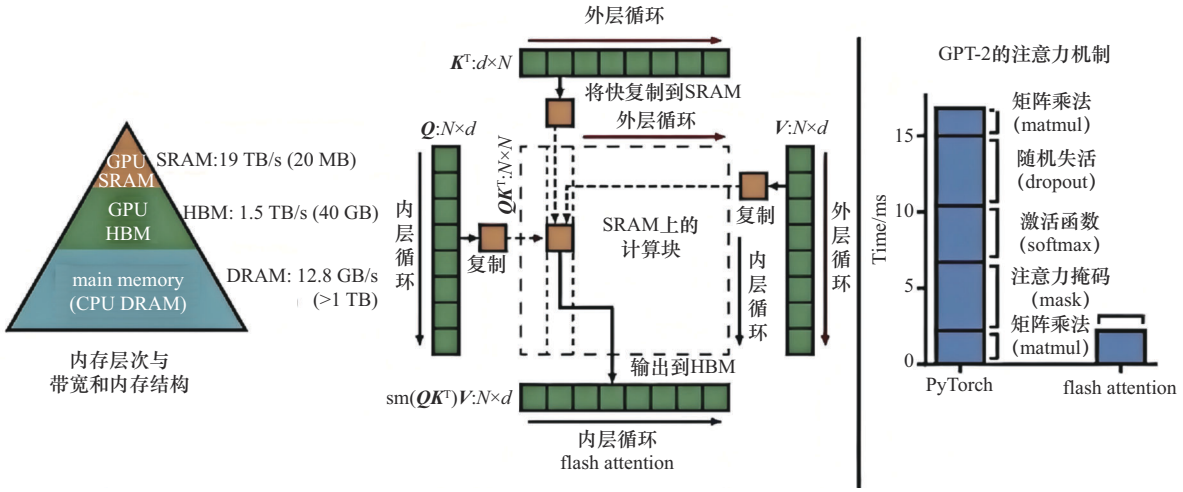


图3　flash attention优化示意图

全局显存的依赖。

- 加速长序列处理。flash attention优化能够更高效地处理长文本输入，极大降低了模型在处理长序列时的时延。这使得批处理机制可以在一次推理中处理更长的上下文，进一步提升了系统的吞吐量。

（2）分页注意力优化

采用另一项用于优化内存访问的技术，分页注意力（paged attention）优化，如图4所示，其特别适用于大模型在多层结构中自注意力机制的高效实现。该技术通过分块（paging）的方式，将大规模计算任务分解为更小的计算单元，分阶段地进行推理，避免了大批次计算时的显存溢出问题。

- 分段内存管理：paged attention优化通过分段管理的方式，将模型的自注意力计算拆分为若干较小的内存单元进行处理，从而减少每次推理时对显存的峰值占用。这种技术使得大模型能够在有限的硬件资源下高效运行，尤其是当模型推理的批次较大时，paged attention优化能够帮助模型推理在不损失精度的情况下，完成更高效的推理任务。
- 提升大模型推理效率：paged attention优化允许系统在高并发请求中处理更多的输入请求，尤其适合需要大批量并发处理的场景。通过优化内存访问，该技术

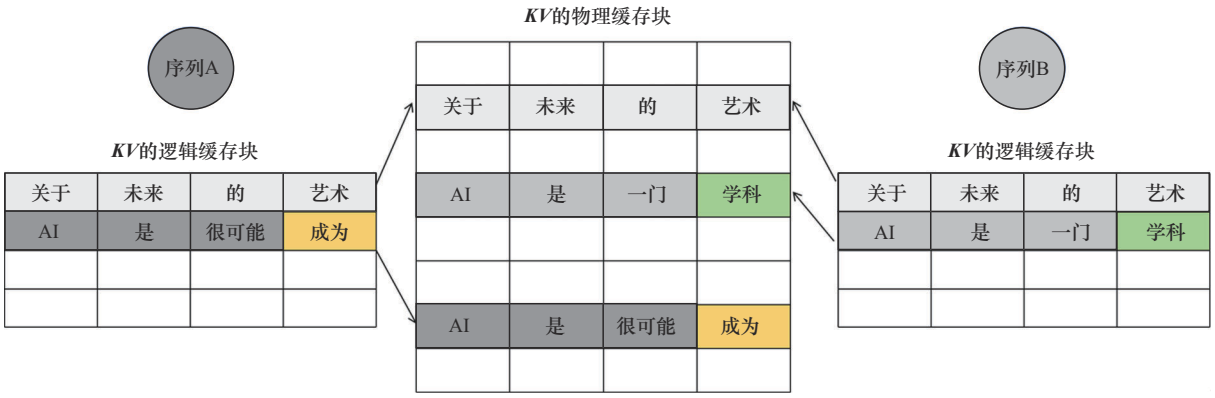


图4　paged attention优化示意图

能够有效提升模型的推理效率，缩短批处理时间。

3.4 多GPU并行与负载均衡

为了应对大规模模型推理的高计算需求，系统设计了多GPU并行机制，结合动态调度和负载均衡技术，确保了计算资源的充分利用，并大幅提升了推理性能。基于模型的特性，系统灵活采用了多种并行策略，包括张量并行、模型并行以及负载均衡，确保了不同模型在不同规模和负载下都能获得最佳的计算效率。

3.4.1 张量并行

张量并行是一种将模型内部的计算任务分解到多个GPU上并行执行的技术。它特别适合处理超大规模的深度学习模型，如Qwen-32B这种具有数百亿参数的Transformer模型。在张量并行的过程中，如图5所示，模型的张量操作（如矩阵乘法、点积运算）会被拆分为多个子任务，这些子任务在不同的GPU上同时执行，从而加快了模型的推理速度。

- 并行化计算。将大张量操作分割并分发到多个GPU上执行，如图5（a）、图5（b）、图5（c）、图5（d）所示，张量并行大幅度减少了单个GPU的计算负担，使得多个GPU协同工作来完成大规模推理任务。例如，Qwen-32B模型在双GPU环境下通过张量并行，每秒能够处理高达600个token，大大提升了处理能力。
- 减少单点瓶颈。由于大型模型往往对显存和计算资源的需求非常高，单个GPU可能无法高效处理全部任务。张量并行通过分割计算负载，减少了单个GPU的资源瓶颈，使模型能够在多GPU环境中流畅运行。

3.4.2 负载均衡

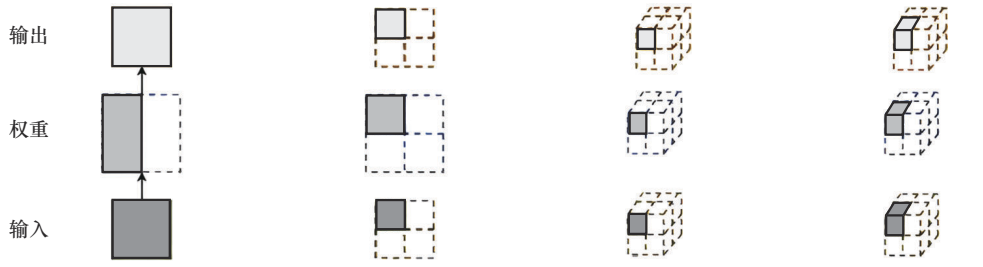
除了张量并行，系统还采用了负载均衡策略来提升整体推理效率。负载均衡是一种将推理请求智能分发到多个模型实例或GPU上的方法，确保各个GPU的计算负载尽可能均匀地分布，避免某个GPU成为瓶颈。

- 最少负载调度。系统会持续监测每个GPU的当前负载情况，并将新的推理请求分配给负载最少的GPU。这样可以确保各个GPU的计算资源被充分利用，避免因个别GPU过载导致的响应时延。
- 最低时延分发。在负载均衡的过程中，系统不仅会考虑GPU的负载，还会参考每个GPU的响应时延。对于高优先级请求，系统会优先将任务分配给响应最快的GPU，确保请求的时延最小化。

3.4.3 并行策略设计

基座基于GPU负载、响应时间、任务类型等多维度数据，结合张量并行和负载均衡的技术，实现了灵活的并行部署策略。

- 灵活选择并行方式。根据不同模型的特



（a）1维张量并行　（b）2维张量并行　（c）2.5维张量并行　（d）3维张量并行

图5　张量并行

性和推理任务的需求，基座会智能地选择最适合的并行方式。例如，对于超大模型（如Qwen-32B），系统会优先采用张量并行，以分担模型内部的计算负荷；而对于较小模型（如Qwen-7B），由于单GPU即可满足计算需求，系统则更倾向于采用负载均衡的方式，最大化GPU利用率。在负载较轻时，基座可以选择较少的GPU进行推理，节省资源；而在高负载时，基座则会动态增加GPU参与推理。

- 动态批处理与并行结合。系统的动态批处理机制与多GPU并行策略无缝结合，即便在高并发请求场景下，系统依然能够保持高吞吐量。具体来说，批处理的请求会被分解后，通过张量并行或模型并行在多个GPU上同时处理，最终合并结果返回给请求方。这种结合既保证了推理的速度，又保证了对GPU资源的高效利用。

为进一步提升多GPU推理的效率，基座系统实现了智能的请求路由机制。请求路由不仅根据当前的负载情况分发请求，还能动态监控各个GPU的健康状态和响应能力。在实际运行中，基座系统通过监测每个GPU的处理能力、内存占用和温度等因素，自动调整任务分配策略，确保每个GPU始终处于最佳负载状态。当某个GPU的负载过高或出现性能问题时，系统能够迅速将新的请求路由到其他可用的GPU上，避免因单个GPU性能下降影响整体推理效果。

3.4.4　基于liteLLM的负载均衡动态调度和请求路由技术

基座采用liteLLM作为负载均衡工具，并对其进行了灵活的配置和调度与路由优化。

（1）动态调度

基座系统通过动态调度策略，对推理任务的优先级、请求的紧急程度以及当前GPU的资源状况进行综合判断。在推理任务达到一定规模或负载时，调度器会实时调整任务的分发策略，优先选择可用资源较多的GPU进行推理。这样不仅能减少高负载GPU的响应时间，还能避免个别GPU资源被过度占用，进而提升整个系统的整体性能。

（2）动态请求路由

为了确保高并发请求场景下的稳定性，基座系统采用了动态请求路由技术。该技术通过实时监控所有GPU节点的健康状况，确保当某个节点出现故障或负载过重时，新的请求能够自动路由到其他健康节点。此外，动态路由还支持根据历史数据进行预测性调度。例如，可以根据某个GPU的历史负载情况，提前预判其可能的负载峰值，从而将新的任务分配到其他节点上，进一步提升了系统的稳定性和可靠性。

（3）基于API key的用户请求限流机制

在提供高效推理服务的同时，基座系统还通过基于API Key的用户请求限流机制，确保资源的公平分配，防止滥用和恶意攻击行为，从而提升整体服务的可靠性与稳定性。每个用户在发送请求时，必须使用专属的API Key进行认证。系统根据用户的API Key分配不同的限流策略，包括每秒允许的请求数、每分钟或每小时的请求配额等。通过限制单个用户的最大请求速率，基座能够有效避免单个用户因过度使用资源导致其他用户的服务体验下降。请求限流机制有效防止了滥用与攻击，提升了服务可靠性与稳定性。

3.5　模型量化以提高效率

为了应对大模型推理中的内存和计算资源的高需求，本系统在统一算力基座中利用了先进的模型量化技术，如GPTQ、AWQ和广义图通用格式（generalized graph-universal format，GGUF），来显著减少模型的内存占用和计算负担，同时保

证推理性能和精度的平衡。这种优化策略不仅提升了推理模型的效率，还为支持多个高需求的模型提供了有力的保障。

- GPTQ。GPTQ是一种常用于Transformer模型的量化技术，能够在训练阶段就考虑量化误差，将模型权重从32位浮点数量化为4位或8位。它通过针对性地减少数值精度来降低内存开销，同时维持较高的模型性能和推理准确度。在本系统中，GPTQ技术用于对诸如Qwen-32B和Qwen-7B等大规模模型的推理进行优化，能够在不显著影响推理精度的前提下，大幅降低GPU的显存占用，使得单个GPU可以同时运行更多模型实例。
- AWQ。AWQ技术专注于对模型中的激活值和权重进行动态量化，通过将激活值和权重在推理过程中按需进行精度压缩，从而降低推理时的内存和计算复杂度。AWQ能够显著减少计算需求，并且适用于不同模型的量化。
- GGUF。GGUF是一种高效的模型压缩格式，特别适用于分布式推理环境。它通过对模型的计算图进行优化，减少了传输开销和模型的部署复杂度。GGUF技术有助于在多服务器环境中部署多种模型，而不需要显著增加内存负担或硬件需求。在推理模型中，GGUF用于embedding模型的优化，使得大规模语义搜索任务能够在多GPU之间高效分布式执行，最大化了资源利用率。
- 多模型支持与灵活部署。通过上述量化技术的应用，本系统的基座能够同时支持多个高需求的模型，包括内部自研的Telechat大模型和用于语义搜索的embedding模型。这些量化技术显著减少了每个模型的内存和计算资源占用，使得同一台GPU或同一组GPU能够同时运行多个不同的模型实例，大大提升了资源利用效率。例如，在处理Telechat大模型与embedding模型的混合负载时，系统能够动态调整内存分配，确保每个模型都能够以最优配置运行，而不会因为资源紧张导致推理性能下降。
- 资源效率的提升。模型量化不仅降低了硬件的资源开销，还可以在现有的硬件条件下支持更多用户的并发请求。通过减少模型对显存和计算资源的占用，量化技术帮助系统降低了成本，并且减少了推理时的功耗，使得整个推理模型更具经济效益。

3.6 容错性与可扩展性

为了确保在高并发和复杂场景下的稳定性与可扩展性，本文的推理模型设计了多重容错功能。首先，系统通过跨服务负载均衡来智能分配请求，将负载动态分散到不同的服务实例和GPU资源上，避免了单个节点的过载。同时，自动请求排队机制会在系统接近最大负载时生效，保证过多的请求不会直接拒绝，而是有序排队等待处理。

这些容错机制能够有效应对突发流量，保障在高峰期或异常情况下，系统仍能平稳运行，服务不中断。例如，在支持40个用户同时在线的情况下，系统通过动态调度与负载均衡，确保每个用户的请求都能被及时处理。同时，结合健康检查和故障恢复机制，系统能够自动检测服务异常并快速恢复，进一步提高了容错能力和服务的可扩展性。

这种设计不仅提升了系统的稳定性，还为未来的扩展提供了足够的灵活性，能够随着用户需求的增加轻松扩展资源。

3.7　可视化工作流编排的创新集成

为了提升系统的易用性和灵活性，本文将统一算力基座与可视化工作流编排平台进行了无缝集成。通过该工作流编排平台，用户能够以图形化的方式进行工作流设计，无须编写复杂代码即可实现大模型的调用与管理。这种低代码集成方式不仅提升了开发效率，还允许用户根据需求动态调整模型与算力的分配，满足了复杂业务场景下的灵活性需求。工作流编排平台还具备自动生成 API 的能力，使用户能够通过页面或 API 直接访问基座的服务，进一步降低了开发门槛。同时，统一算力基座中的 API 与 OpenAI 标准兼容，开发者无须关注底层模型部署细节即可完成大模型的快速集成。

4　结果与性能评估

4.1　吞吐量提升与时延降低

统一推理模型在测试中展现了显著的性能提升。在测试环境下，双 GPU 的 Qwen-32B 模型实现了每秒 600 token 的吞吐量，而单 GPU 的 Qwen-2-7B 模型实现了每秒 1 400 token 的吞吐量。响应时延始终保持在较低水平，非高峰期首个 token 的响应时间低于 1 s，高峰期时延不超过 4 s。

4.2　资源利用率

通过整合模型部署，推理模型提高了 50% 的 GPU 利用率，使多个模型可以共享计算资源而不会产生资源争用。连续批处理的使用进一步提升了效率，使系统能够在高负载下平稳运行，而不会出现显著的性能下降。

4.3　开发与集成时间减少

统一的 API 使模型集成与部署时间减少了 50%，开发者可以专注于应用逻辑，而无须管理底层推理模型的复杂性。这种简化的集成流程大大加快了新模型的原型设计与生产部署。

5　结束语

本文提出了一种大模型统一算力基座的多 GPU 并行与动态调度方法，解决了资源碎片化、推理效率低和多样化模型扩展的核心问题。通过引入动态批处理、多 GPU 并行和先进的模型量化技术，系统显著提升了模型吞吐量与响应速度，同时减少了开发与运维的复杂性。该推理模型引入了多项关键创新。

- 统一的模型部署与动态调度：提供了一个统一的平台来部署多种大语言模型，通过动态调度优化资源分配，提升了 50% 及以上的吞吐量。
- 连续与动态批处理：通过动态将请求分组进行批量处理，降低推理时延并提升系统吞吐量。
- 多 GPU 并行：将推理任务分布到多个 GPU 上，使大模型能够高效扩展，并处理更高的用户并发。
- 容错与可扩展架构：通过自动请求排队和跨服务负载均衡，确保系统在高负载下的稳定性。

这一技术为企业提供了一个可扩展的高性能解决方案，帮助高效部署和管理大模型。

参考文献：

[1] FRANTAR E, ASHKBOOS S, HOEFLER T, et al. GPTQ: accurate post-training quantization for generative pre-trained transformers[J]. arXiv preprint, arXiv:2210.17323, 2022.

[2] LIN J, TANG J M, TANG H T, et al. AWQ: activation-aware weight quantization for on-device LLM compression and acceleration[J]. GetMobile Mob Comput Commun, 2023, 28: 12-17.

[3] DEANGELO G, BATABYAL A A, KUMAR S. An analysis of economic cost minimization and biological invasion damage control using the AWQ criterion[J]. The Annals of Regional Science, 2007, 41(3): 639-655.

[4] RAJPUT S, SHARMA T. Benchmarking emerging deep learning quantization methods for energy efficiency[C]//2024 IEEE

21st International Conference on Software Architecture Companion (ICSA-C). Piscataway: IEEE press, 2024: 238-242.

[5] DAFAVI-NAINI S A A, ALI S, SHAHAB O, et al. Vision-language and large language model performance in gastroenterology: GPT, Claude, Llama, Phi, Mistral, Gemma, and Quantized Models[J]. arXiv preprint, arXiv: 2409.00084, 2024.

[作者简介]

邬烨磊（1996– ），男，中国电信股份有限公司上海分公司工程师，主要研究方向为大模型研发、IT 系统设计。

卢雅楠（1998– ），女，现就职于中国电信股份有限公司上海分公司，主要研究方向为大模型研发、IT 系统设计。

黄煜骁（1993– ），男，中国电信股份有限公司上海分公司工程师，主要从事大数据和 AI 平台架构和能力建设、项目规划、大模型研发工作。

张安达（1995– ），男，中国电信股份有限公司上海分公司助理工程师，主要研究方向为大模型研发。

江天松（1997– ），男，现就职于中国电信股份有限公司上海分公司，主要研究方向为可验证计算、机器学习。