

# 硕士学位论文

## 面向大模型的推理任务批式调度 和 KV 缓存优化

**BATCH SCHEDULING AND KV CACHE  
OPTIMIZATION FOR LARGE LANGUAGE  
MODEL INFERENCE**

研 究 生：何忆源  
指 导 教 师：徐敏贤副研究员

南方科技大学  
二〇二六年一月



国内图书分类号: XXxxxx.x

国际图书分类号: xx-x

学校代码: 14325

密级: 公开

## 工学硕士专业学位论文

# 面向大模型的推理任务批式调度 和 KV 缓存优化

学位申请人: 何忆源

指导教师: 徐敏贤副研究员

专业类别: 计算机技术

答辩日期: 2026 年 3 月

培养单位: 深圳理工大学

学位授予单位: 南方科技大学



# **BATCH SCHEDULING AND KV CACHE OPTIMIZATION FOR LARGE LANGUAGE MODEL INFERENCE**

A dissertation submitted to  
Southern University of Science and Technology  
in partial fulfillment of the requirement  
for the professional degree of  
Master of Engineering

by

He Yiyuan

Supervisor: Associate Researcher Minxian Xu

March, 2026



# 学位论文公开评阅人和答辩委员会名单

## 公开评阅人名单

刘 XX	教授	南方科技大学
陈 XX	副教授	XXXX 大学
杨 XX	研究员	中国 XXXX 科学院 XXXXXXX 研究所

## 答辩委员会名单

主席	赵 XX	教授	南方科技大学
委员	刘 XX	教授	南方科技大学
	杨 XX	研究员	中国 XXXX 科学院 XXXXXXX 研究所
	黄 XX	教授	XXXX 大学
	周 XX	副教授	XXXX 大学
秘书	吴 XX	助理研究员	南方科技大学



# 南方科技大学学位论文原创性声明和使用授权说明

## 南方科技大学学位论文原创性声明

本人郑重声明：所提交的学位论文是本人在导师指导下独立进行研究工作所取得的成果。除了特别加以标注和致谢的内容外，论文中不包含他人已发表或撰写过的研究成果。对本人的研究做出重要贡献的个人和集体，均已在文中作了明确的说明。本声明的法律结果由本人承担。

作者签名： 日期：

## 南方科技大学学位论文使用授权书

本人完全了解南方科技大学有关收集、保留、使用学位论文的规定，即：

1. 按学校规定提交学位论文的电子版本。
2. 学校有权保留并向国家有关部门或机构送交学位论文的电子版，允许论文被查阅。
3. 在以教学与科研服务为目的前提下，学校可以将学位论文的全部或部分内容存储在有关数据库提供检索，并可采用数字化、云存储或其他存储手段保存本学位论文。
  - (1) 在本论文提交当年，同意在校园网内提供查询及前十六页浏览服务。
  - (2) 在本论文提交  当年/ 年以后，同意向全社会公开论文全文的在线浏览和下载。
4. 保密的学位论文在解密后适用本授权书。

作者签名： 日期：

指导教师签名： 日期：



## 摘要

随着大语言模型（LLMs）在云计算环境中的广泛部署，其推理服务面临资源消耗巨大、服务等级目标（SLO）要求严格、预填充（Prefill）与解码（Decode）阶段资源需求异构以及负载动态多变等多重挑战。据统计，云平台中约 90% 的人工智能计算资源用于模型推理而非训练，而现有系统受限于粗粒度的静态资源配置，GPU 利用率通常仅为 20%–40%，且在突发流量下因扩容滞后导致严重的 SLO 违约。针对这些问题，本文围绕“面向大模型的推理任务批式调度和 KV 缓存优化”这一主题，构建了从静态优化到动态调控的技术体系，实现了对 LLM 推理服务资源管理的系统性优化。

首先，针对静态场景下批处理策略粗放、部署配置忽视网络拓扑异构性、缺乏服务质量（SLO）感知等问题，本文提出了 UELLM（Unified and Efficient LLM Inference Serving）统一高效批式调度与部署框架。该框架通过微调 ChatGLM3-6B 模型对请求输出长度进行预测，分桶准确率达到 99.51%，为后续批处理优化提供关键先验知识；设计了 SLO 与输出长度驱动的动态批处理算法（SLO-ODBS），通过权重化目标函数平衡总延迟与总输出长度，采用贪心策略将长度相近的请求组合成批，有效减少 KV Cache 冗余填充和计算浪费；提出了基于动态规划的高效低延迟资源分配算法（HELR），综合考虑集群网络拓扑异构性（NVLink/PCIe 带宽差异）和 GPU 计算能力差异，自动优化层到 GPU 的映射策略，支持高利用率（HE）与低延迟（LR）两种模式灵活切换。在 4-GPU 异构集群上的验证表明，UELLM 相比 Morphling 和 S<sup>3</sup> 等先进方案，降低推理延迟 72.3% 至 90.3%，提升 GPU 利用率 1.2 倍至 4.1 倍，吞吐量提高 1.92 倍至 4.98 倍，并实现零 SLO 违约。

其次，面向 PD（Prefill-Decode）分离架构中固有的计算–内存负载失衡、静态资源配置无法适应动态负载变化、前缀缓存感知路由导致的热点倾斜等问题，本文提出了 BanaServe 细粒度弹性资源管理与 KV 缓存优化框架。该框架创新性地引入了模块级细粒度迁移机制，包括层级（Layer-level）权重迁移与注意力级（Attention-level）KV Cache 迁移：前者支持连续 Transformer 层的动态重配置，实现粗粒度负载均衡；后者通过按注意力头维度切分 KV Cache，选择性将部分注意力头状态迁移至辅助 GPU，实现细粒度计算卸载，且无需传输模型权重，迁移开销极低；设计了全局 KV Cache 存储与层间流水线重叠传输机制，通过解耦缓存状态与计算位置，消除前缀缓存对调度决策的约束，使路由器可基于纯负载指标进行调度，同

## 摘要

---

时利用 Transformer 逐层计算特性，将第  $i$  层计算与第  $i + 1$  层 KV Cache 预取重叠，隐藏通信延迟；提出了基于实时负载感知的请求调度算法，周期性地测量各 Prefill 实例的综合负载（计算 + 内存利用率），将新请求分发至负载最轻的实例，配合动态迁移机制实现快速负载均衡。在 13B 参数模型（LLaMA-13B/OPT-13B）和公开基准（Alpaca/LongBench）上的评估表明，BanaServe 相比 vLLM 吞吐量提升 1.2 倍至 3.9 倍、总处理时间降低 3.9% 至 78.4%；相比 DistServe 吞吐量提升 1.1 倍至 2.8 倍、延迟降低 1.4% 至 70.1%，并在突发流量场景下展现出优异的鲁棒性。

本文所提出的方法在真实 GPU 集群和公开基准数据集上得到了充分验证，构建的“静态批处理优化–动态层级资源调控”技术体系，为大模型推理服务的高效部署与资源管理提供了新的理论依据和技术路径，对推动大模型技术在智能客服、内容创作、知识检索等延迟敏感场景的规模化应用具有重要意义。

**关键词：**大语言模型推理；批式调度；PD 分离架构；动态批处理；异构部署优化；弹性伸缩；层级资源迁移；KV Cache 优化；SLO 保障

## Abstract

With the widespread deployment of Large Language Models (LLMs) in cloud computing environments, inference services face critical challenges including massive resource consumption, strict Service Level Objective (SLO) requirements, heterogeneous resource demands between the prefill and decode phases, and highly dynamic workload variations. Statistics indicate that approximately 90% of AI computing resources in cloud platforms are dedicated to model inference rather than training. However, existing systems are constrained by coarse-grained static resource configurations, resulting in GPU utilization typically ranging merely between 20%–40%, and severe SLO violations due to delayed scaling during bursty traffic. To address these issues, this thesis focuses on “Batch Scheduling and KV Cache Optimization for Large Language Model Inference,” constructing a technical architecture spanning from static optimization to dynamic scaling, thereby achieving systematic optimization of resource management for LLM inference services.

Firstly, to address coarse-grained batching strategies, deployment configurations that neglect network topology heterogeneity, and the lack of Service Level Objective (SLO) awareness in static scenarios, this thesis proposes UELLM (Unified and Efficient LLM Inference Serving), a unified batch scheduling and deployment optimization framework. The framework employs a fine-tuned ChatGLM3-6B model to predict request output lengths, achieving a bucketing accuracy of 99.51% to provide critical prior knowledge for subsequent batching optimization. It introduces the SLO and Output-Driven Dynamic Batch Scheduler (SLO-ODBS), which balances total latency and total output length through a weighted objective function, and adopts a greedy strategy to group requests with similar lengths into batches, effectively reducing KV Cache redundancy and computational waste. Additionally, the High-Efficiency Low-Latency Resource Allocation (HELR) algorithm based on dynamic programming is proposed, which automatically optimizes layer-to-GPU mapping strategies by comprehensively considering cluster network topology heterogeneity (NVLink/PCIe bandwidth variations) and GPU computational capability differences, supporting flexible switching between High Efficiency (HE) and Low Latency (LR) modes. Experimental validation on a 4-GPU heterogeneous cluster demonstrates that UELLM reduces inference latency by 72.3% to 90.3%, improves GPU utilization by 1.2 $\times$  to 4.1 $\times$ , and increases throughput by 1.92 $\times$  to 4.98 $\times$  compared to

---

## Abstract

---

state-of-the-art methods such as Morphling and S<sup>3</sup>, while achieving zero SLO violations.

Secondly, targeting the inherent compute-memory load imbalance, inability of static resource configurations to adapt to dynamic workload changes, and hotspot skews caused by prefix cache-aware routing in PD (Prefill-Decode) disaggregated architectures, this thesis presents BanaServe, a fine-grained elastic resource management and KV cache optimization framework. The framework introduces novel module-level migration mechanisms, including layer-level weight migration and attention-level KV Cache migration: the former supports dynamic reconfiguration of consecutive Transformer layers for coarse-grained load rebalancing; the latter achieves fine-grained computation offloading by partitioning KV Cache along the attention head dimension and selectively migrating partial attention head states to auxiliary GPUs without transferring model weights, incurring minimal migration overhead. It designs a Global KV Cache Store with layer-wise pipelined transmission to eliminate constraints imposed by prefix caching on scheduling decisions by decoupling cache state from compute placement, enabling purely load-based routing decisions, while simultaneously hiding communication latency by overlapping the computation of layer  $i$  with the prefetching of KV Cache for layer  $i + 1$  leveraging the layer-wise execution characteristic of Transformers. Furthermore, a real-time load-aware request scheduling algorithm is proposed, which periodically measures the combined load (compute and memory utilization) of each prefill instance and dispatches new requests to the least-loaded instance, working in coordination with dynamic migration for rapid load balancing. Evaluation on 13B-parameter models (LLaMA-13B and OPT-13B) and public benchmarks (Alpaca and LongBench) demonstrates that BanaServe achieves 1.2×–3.9× higher throughput with 3.9%–78.4% lower total processing time compared to vLLM, and 1.1×–2.8× throughput improvement with 1.4%–70.1% latency reduction compared to DistServe, exhibiting superior robustness under bursty traffic scenarios.

The proposed methods have been extensively validated on real GPU clusters and public benchmarks. The technical architecture of “Static Batching Optimization – Dynamic Hierarchical Resource Scaling” provides novel theoretical foundations and technical pathways for efficient deployment and resource management of LLM inference services, and holds significant importance for promoting the large-scale application of LLM technologies in latency-sensitive scenarios such as intelligent customer service, content creation, and knowledge retrieval.

---

Abstract

---

**Keywords:** Large Language Model Inference; Batch Scheduling; PD Disaggregated Architecture; Dynamic Batching; Heterogeneous Deployment Optimization; Elastic Scaling; Hierarchical Resource Migration; KV Cache Optimization; SLO Guarantee

## 目 录

摘要.....	I
<b>Abstract</b> .....	III
符号和缩略语说明.....	VIII
<b>第1章 绪论</b> .....	1
1.1 研究背景 .....	1
1.2 研究意义 .....	2
1.3 国内外研究现状及分析.....	3
1.3.1 算法层面的推理优化.....	3
1.3.2 系统架构与内存管理优化.....	4
1.3.3 批处理与请求调度策略.....	5
1.3.4 资源配置策略.....	5
1.3.5 国内外研究现状对比.....	6
1.3.6 现有研究的局限性与启示.....	7
1.4 论文主要工作 .....	8
1.4.1 面向 LLM 推理任务的高效批式调度和部署方法 .....	8
1.4.2 面向 PD 分离架构的资源管理和 KV 缓存优化方法.....	9
1.4.3 主要创新点.....	10
1.5 论文组织架构 .....	10
<b>第2章 现有 LLM 的推理任务批式任务调度和 KV 缓存优化的方法</b> .....	12
2.1 现有的 LLM 推理任务批式调度方法 .....	12
2.1.1 S <sup>3</sup> : 输出长度感知调度.....	12
2.1.2 Continuous Batching: 动态请求合并 .....	13
2.1.3 Orca 中的 In-flight Batching 批处理机制 .....	14
2.1.4 传统调度策略.....	15
2.2 现有的 KV 缓存优化方法 .....	15
2.2.1 Mooncake: 全局 KV 缓存池 .....	15
2.2.2 MemServe: 弹性内存管理.....	16
2.2.3 vLLM: PagedAttention 显存优化.....	16
2.2.4 量化: 模型压缩与加速.....	17

2.3 现有方法的缺陷 .....	18
2.3.1 调度粒度与资源状态的紧耦合 .....	18
2.3.2 资源配置的静态化与刚性约束 .....	18
2.3.3 缺乏跨阶段的细粒度资源协同机制 .....	18
2.3.4 对异构性和动态性的适应性不足 .....	19
<b>第 3 章 数学符号和公式 .....</b>	<b>20</b>
3.1 数学符号 .....	20
3.2 数学公式 .....	21
3.3 数学定理 .....	22
3.4 数学字体 .....	22
<b>第 4 章 引用文献的标注 .....</b>	<b>24</b>
4.1 顺序编码制 .....	24
4.2 著者-出版年制 .....	24
4.2.1 其他引用注意事项 .....	24
<b>第 5 章 English and lower-case Example .....</b>	<b>25</b>
5.1 Reference guide .....	25
<b>结 论 .....</b>	<b>26</b>
<b>参考文献 .....</b>	<b>27</b>
<b>附录 A 补充内容 .....</b>	<b>31</b>
<b>致 谢 .....</b>	<b>34</b>
<b>个人简历、在学期间完成的相关学术成果 .....</b>	<b>35</b>

## 符号和缩略语说明

As-PPT	聚苯基不对称三嗪
DFT	密度泛函理论 (Density Functional Theory)
DMA <sub>s</sub> PPT	聚苯基不对称三嗪双模型化合物 (水解实验模型化合物)
$E_a$	化学反应的活化能 (Activation Energy)
HMA <sub>s</sub> PPT	聚苯基不对称三嗪模型化合物的质子化产物
HMPBI	聚苯并咪唑模型化合物的质子化产物
HMPI	聚酰亚胺模型化合物的质子化产物
HMPPQ	聚苯基喹噁啉模型化合物的质子化产物
HMPY	聚吡啶模型化合物的质子化产物
HMSPPT	聚苯基对称三嗪模型化合物的质子化产物
HPCE	高效毛细管电泳色谱 (High Performance Capillary lectrophoresis)
HPLC	高效液相色谱 (High Performance Liquid Chromatography)
IRC	内禀反应坐标 (Intrinsic Reaction Coordinates)
LC-MS	液相色谱-质谱联用 (Liquid chromatography-Mass Spectrum)
MAsPPT	聚苯基不对称三嗪单模型化合物, 3,5,6-三苯基-1,2,4-三嗪
MPBI	聚苯并咪唑模型化合物, N-苯基苯并咪唑
MPI	聚酰亚胺模型化合物, N-苯基邻苯酰亚胺
MPPQ	聚苯基喹噁啉模型化合物, 3,4-二苯基苯并二嗪
MPY	聚吡啶模型化合物
MSPPT	聚苯基对称三嗪模型化合物, 2,4,6-三苯基-1,3,5-三嗪
ONIOM	分层算法 (Our own N-layered Integrated molecular Orbital and molecular Mechanics)
PBI	聚苯并咪唑
PDT	热分解温度
PES	势能面 (Potential Energy Surface)
PI	聚酰亚胺
PMDA-BDA	均苯四酸二酐与联苯四胺合成的聚吡啶薄膜
PPQ	聚苯基喹噁啉
PY	聚吡啶
S-PPT	聚苯基对称三嗪
SCF	自洽场 (Self-Consistent Field)

SCRF	自洽反应场 (Self-Consistent Reaction Field)
TIC	总离子浓度 (Total Ion Content)
TS	过渡态 (Transition State)
TST	过渡态理论 (Transition State Theory)
ZPE	零点振动能 (Zero Vibration Energy)
<i>ab initio</i>	基于第一原理的量子化学计算方法，常称从头算法
$\Delta G^\ddagger$	活化自由能 (Activation Free Energy)
$\kappa$	传输系数 (Transmission Coefficient)
$\nu_i$	虚频 (Imaginary Frequency)



# 第1章 绪论

## 1.1 研究背景

近年来,以GPT-4<sup>[1]</sup>、LLaMA<sup>[2]</sup>、Claude<sup>[3]</sup>等为代表的大语言模型(Large Language Models, LLMs)在自然语言处理、代码生成、知识检索和内容创作等领域展现出卓越的性能,推动了人工智能技术的跨越式发展。这些模型通过数百亿乃至万亿级参数的规模化效应,涌现出强大的上下文学习与推理能力,已成为智能时代的关键基础设施。然而,随着模型参数规模的指数级增长,其训练和推理过程对计算资源提出了极高的要求,由此产生的资源密集型特征与高效部署需求之间的矛盾日益尖锐。

在机器学习即服务(MLaaS)的范式下,大模型的生命周期呈现出“训练集中,推理分散”的显著特征。据统计,云平台中约90%的人工智能计算资源被用于模型推理服务而非训练<sup>[4]</sup>,且推理成本随着模型规模呈超线性增长。以OpenAI的Chat-GPT为例,为维持其日均7000万次访问的服务规模,需要部署超过6万张NVIDIA A100 GPU,初始投资成本高达16亿美元,每日电费支出约10万美元<sup>[5]</sup>。这些数字清晰地揭示了大模型推理阶段对计算资源的巨额消耗及其带来的沉重经济负担。

大模型推理过程具有独特的两阶段执行特性:预填充(Prefill)阶段与解码(Decode)阶段,这种内在的计算模式差异构成了资源优化的核心挑战。Prefill阶段计算密集,需并行处理整个输入序列以计算注意力键值(KV Cache)并生成首个令牌(Time to First Token, TTFT),其延迟直接影响用户感知的响应速度。Decode阶段则受限于自回归生成机制,需逐令牌迭代输出,呈现显著的内存密集型特征,其关键指标为每输出令牌时间(Time Per Output Token, TPOT)。两个阶段在计算强度、内存访问模式和延迟敏感度上的固有不对称性,使得传统单一架构难以同时优化,往往导致严重的资源利用率低下和负载不均衡问题。

此外,大模型推理面临着严峻的显存墙(Memory Wall)挑战。以千亿级参数模型为例,其参数本身需占用数百GB显存,而KV Cache的动态增长进一步加剧了显存压力。由于单张GPU显存容量有限(通常为40GB或80GB),必须采用张量并行(Tensor Parallelism)或流水线并行(Pipeline Parallelism)等分布式部署策略。然而,随着部署GPU数量的增加,节点间通信开销和同步延迟显著上升。反之,若GPU资源配置不足,则会导致显存溢出(Out-of-Memory, OOM)错误或极长的推理延迟。这种资源配置的刚性约束与服务质量(Quality of Service, QoS)要求之间的张力,构

成了大模型部署的核心难点。

更为复杂的是,生产环境的推理负载具有高度动态性和不可预测性。请求到达率(Requests Per Second, RPS)随时间呈现剧烈波动,输入/输出序列长度分布呈现显著的重尾特性(Heavy-tailed),即少数长序列请求占据了大量资源。传统静态资源配置策略在负载低谷期造成严重的资源浪费(GPU利用率常仅20%-40%),而在突发流量(Bursty Traffic)下又因扩容滞后导致服务等级目标(Service Level Objective, SLO)违约甚至服务中断。现有系统如vLLM<sup>[6]</sup>通过PagedAttention优化显存管理,DistServe<sup>[7]</sup>采用PD分离架构消除阶段间干扰,但这些方案仍受限于粗粒度的资源配置(实例级或GPU池级)和缺乏前瞻性的静态调度策略,无法适应快速变化的负载模式。因此,如何在保障严格SLO的前提下,实现计算与内存资源的细粒度调度,成为大模型推理服务从实验室走向规模化产业应用必须攻克的关键瓶颈。

## 1.2 研究意义

本研究围绕大模型推理服务的批式调度与KV缓存优化展开,针对静态资源配置与动态负载需求之间的结构性矛盾,具有以下重要的理论价值和实践意义:

### (1) 理论价值: 构建分层优化的资源管理理论框架

从科学价值角度,本研究针对LLM推理中计算-内存协同优化这一基础科学问题,提出了从请求级批处理到层级资源分配的分层优化理论框架。通过建立考虑KV Cache动态增长、网络拓扑异构性和SLO约束的数学模型,深入探索了在离散配置空间中寻找最优部署策略的算法边界与计算复杂度。研究揭示了在Prefill与Decode阶段分离场景下,资源分配与延迟约束之间的权衡机理,为大规模分布式推理系统的资源管理提供了新的理论依据和分析工具,丰富了云计算与人工智能交叉领域的理论体系。

### (2) 经济效益: 显著降低推理成本与提升资源利用率

从工程实践角度,研究成果可直接应用于云原生AI基础设施,产生显著的经济效益。首先,通过精确的输出长度预测和SLO感知的动态批处理算法,可有效减少无效填充(Padding)和冗余计算,预计可降低推理成本30%以上,缓解企业运营压力。其次,层级资源分配技术通过考虑网络拓扑异构性,将资源配置时间从传统方法的数分钟降至秒级,显著提升系统对不同负载模式的适应性。再者,通过优化的批处理策略和部署配置,可将集群整体GPU利用率从当前的20%-40%提升至70%以上。这些技术对推动大模型在智能客服、自动驾驶、金融风控等延迟敏感场景的普及应用具有重要意义。

### (3) 绿色计算: 促进环境可持续性与社会责任

提高资源利用率对于全球环境可持续性具有重要的战略意义。当前大模型服务需要消耗巨量电力,不仅增加了企业的运营成本,也产生了显著的碳足迹。通过优化计算资源的调度与使用,减少闲置 GPU 的能源浪费,可有效降低数据中心整体功耗,符合全球倡导的绿色计算和可持续发展目标<sup>[4]</sup>。据估算,若全球云 AI 基础设施采用本研究提出的资源优化方法,每年可减少数亿千瓦时的能源消耗,为推动人工智能技术的绿色发展提供可行的技术路径。

#### (4) 技术生态: 推动边缘计算与异构计算发展

此外,本研究提出的统一优化框架不仅适用于云端数据中心,其核心理念还可延伸至边缘计算场景。通过高效的资源调度机制,使大模型能够在资源受限的边缘设备和嵌入式系统中高效运行,拓展了 LLM 的应用边界。研究还将促进 GPU、TPU、CPU 等异构计算资源的协同调度技术发展,为构建更加灵活、高效、经济的 AI 计算基础设施提供技术支撑,进一步推动人工智能技术在医疗、教育、制造等各行各业的深度落地与普惠应用。

综上所述,面向大模型推理任务的优化研究不仅是对现有系统性能的改进,更是推动大模型技术从理论走向规模化实践的关键一步。随着大模型的不断扩展及其推理需求的持续增长,本研究成果将在提升模型应用效率、节约计算资源及提高系统稳定性方面发挥愈加重要的作用,为人工智能产业的健康可持续发展奠定坚实基础。

### 1.3 国内外研究现状及分析

大语言模型推理优化的研究呈现出从算法创新到系统架构、从静态配置到动态调度、从单机优化到分布式协同的演进趋势。本节系统梳理了算法层面的解码优化、系统层面的内存管理与批处理技术、架构层面的资源配置策略,并对比分析国内外研究特点,最后指出现有研究的局限性与本文切入点。

#### 1.3.1 算法层面的推理优化

##### 1. 解码策略创新

为突破自回归生成的序列依赖瓶颈,学术界提出了多种非传统的解码范式。**推测性解码** (Speculative Decoding) 通过小规模草稿模型 (Draft Model) 快速生成候选序列,再由目标模型并行验证并修正,在保持输出质量不变的前提下实现 2-3 倍的加速比<sup>[8-9]</sup>。**非自回归解码** (Non-autoregressive Decoding) 则彻底打破逐 token 生成的限制,通过迭代精化 (Iterative Refinement) 或掩码预测 (Masked Prediction) 机制并行输出生成序列,虽然牺牲部分质量,但在实时性要求极高的场景 (如实时翻译) 展

现潜力。

针对生成过程的不确定动态性,早退出机制(Early Exiting)和级联推理(Cascade Inference)根据中间层置信度自适应决定计算深度。例如,CALM<sup>[10]</sup>通过动态层级退出减少简单查询的计算量。CascadeBERT<sup>[11]</sup>采用模型级联策略,对简单样本使用轻量级模型,仅将困难样本路由至大模型。这类方法通过计算量与任务难度的自适应匹配,显著提升了平均推理效率。

## 2. 模型压缩与量化

为缓解显存墙(Memory Wall)问题,模型压缩技术成为研究热点。低比特量化(Low-bit Quantization)将FP16/FP32权重压缩至INT8甚至INT4,配合权重量化(Weight-only Quantization)和激活量化(Activation Quantization)策略,使千亿级模型可在单卡消费级GPU上部署<sup>[12-13]</sup>。近期研究还探索了1-bit量化(如BitNet<sup>[14]</sup>)和混合精度量化,通过细粒度分组量化(Group-wise Quantization)减少精度损失。

知识蒸馏(Knowledge Distillation)通过将大模型(教师)的知识迁移至小模型(学生),在保持性能的同时显著降低推理成本。MiniLLM<sup>[15]</sup>和LaMini-GPT<sup>[16]</sup>等工作表明,经过针对性蒸馏的7B参数模型在特定任务上可逼近70B模型的性能。国内研究机构在此领域成果显著,清华大学的QLoRA<sup>[17]</sup>和Knowledge Fusion方法在低资源场景下的压缩效率达到国际领先水平。

### 1.3.2 系统架构与内存管理优化

#### 1. 注意力机制与显存优化

Transformer的注意力机制计算复杂度高,显存占用随序列长度平方增长。Page-dAttention<sup>[6]</sup>借鉴操作系统虚拟内存管理思想,将KV Cache划分为非连续的物理块(Block),通过块表(Block Table)映射实现动态分配,消除了传统连续存储导致的显存碎片,使GPU显存利用率从40-50%提升至90%以上。在此基础上,vLLM实现了连续批处理(Continuous Batching),通过动态合并新到达请求并移除已完成请求,最大化吞吐率。

针对长上下文(Long Context)场景,滑动窗口注意力(Sliding Window Attention)和稀疏注意力(Sparse Attention)通过限制注意力范围降低计算复杂度。StreamingLLM<sup>[18]</sup>发现注意力汇点(Attention Sinks)现象,仅需保留初始token的KV Cache即可维持长序列生成稳定性,将显存消耗从序列长度的二次方降至线性。

#### 2. 内核优化与硬件协同

为充分发挥GPU Tensor Core的计算能力,算子融合(Operator Fusion)和自动编译(Automatic Compilation)技术被广泛采用。FlashAttention<sup>[19]</sup>系列算法通过IO感知的精确计算调度和分块策略(Tiling),将注意力计算中的HBM访问次数从 $O(N)$

降至  $O(N^2/\text{block\_size})$ , 在 A100 GPU 上实现 2-4 倍的加速。TVM<sup>[20]</sup>和 MLIR 等编译器框架通过自动生成针对特定硬件的微内核 (Micro-kernel), 进一步优化了矩阵乘法和内存访问模式。

### 1.3.3 批处理与请求调度策略

批处理 (Batching) 是提升 LLM 推理吞吐量的核心手段, 但静态批处理 (Static Batching) 的填充浪费 (Padding Waste) 问题严重。近期研究聚焦于输出长度感知调度和 SLO 约束优化。

输出长度预测是优化批处理的关键。 $S^3$ <sup>[21]</sup>将请求调度建模为多维装箱问题, 利用轻量级预测器估计输出长度, 将相似长度的请求归入同一批次, 减少填充开销。现有方案多基于历史数据静态预测, 难以适应输出长度分布的实时变化。

前缀缓存与数据局部性是另一优化维度。SGLang<sup>[22]</sup>和 RadixAttention 通过前缀树 (Trie) 结构缓存共享系统提示 (System Prompt) 的 KV Cache, 对多轮对话场景实现显著加速。然而, 缓存感知路由 (Cache-aware Routing) 引入的负载不均衡问题尚未得到有效解决, 热点前缀可能导致特定 GPU 过载。

SLO 感知调度方面,ClockWork<sup>[23]</sup>针对传统模型提出基于完成时间预测的早期退出策略, 但难以适应 LLM 自回归生成的动态延迟特性。Splitwise<sup>[24]</sup>尝试在保证延迟 SLO 的前提下优化吞吐量, 但缺乏对 Prefill 与 Decode 阶段差异性的细粒度建模。

### 1.3.4 资源配置策略

#### 1. PD 分离架构的演进

大模型推理的 Prefill 阶段 (计算密集) 与 Decode 阶段 (内存密集) 具有截然不同的资源需求特征, 催生了预填充-解码分离 (Prefill-Decode Disaggregation) 架构。DistServe<sup>[7]</sup>和 Splitwise<sup>[24]</sup>将两个阶段部署在不同 GPU 实例上, 消除阶段间干扰, 使各自可独立优化。然而, 现有方案多采用静态配比 (如 1:1 或固定比例), 无法适应动态负载中 Prefill 与 Decode 需求的实时变化。

分布式 KV Cache 管理是支撑 PD 分离的关键。Mooncake<sup>[25]</sup>提出全局 KV Cache 池, 通过 RDMA 网络实现跨节点缓存迁移。MemServe<sup>[26]</sup>探索了缓存预取 (Prefetching) 和冗余消除技术。但这些方案引入了显著的跨节点通信开销, 且缺乏对网络拓扑异构性的感知。

#### 2. 部署配置优化

传统模型部署采用静态配置方式, 在部署时固定 GPU 数量和层到设备的映射关系, 无法在运行期间根据实际负载动态调整。Morphling<sup>[27]</sup>提出基于元学习的配

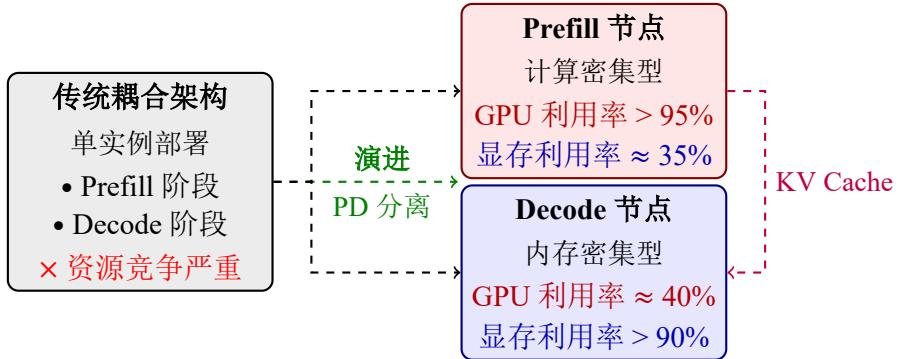


图 1-1 传统耦合架构与 PD 分离架构对比

置搜索框架,但需要对每个候选配置进行压力测试,产生额外计算负担。

在异构资源管理方面,AntMan<sup>[28]</sup>和 Allox<sup>[29]</sup>提出针对深度学习任务的 GPU 时间片调度,但主要针对训练场景,未考虑 LLM 推理中 KV Cache 的状态保持需求。NVIDIA MPS 和 MIG(Multi-Instance GPU) 支持硬件级的 GPU 切分,但配置刚性且缺乏软件层面的灵活性。

### 1.3.5 国内外研究现状对比

#### 1. 国外研究: 系统架构与硬件协同领先

以 OpenAI、Google、Meta、Stanford 等为代表的国外研究机构在系统架构创新和硬件-软件协同设计方面处于领先地位。OpenAI 的 GPT 系列和 Google 的 Gemini 推动了超大规模模型的工程化实践,其内部推理系统(如 Triton、Pathways)虽未完全开源,但通过 vLLM、TensorRT-LLM 等开源工具影响了行业标准。学术机构如 Stanford 的 FlexGen<sup>[30]</sup>和 Berkeley 的 SkyPilot<sup>[31]</sup>在资源受限场景下的推理优化提出系统性解决方案。

在新兴硬件适配方面,国外研究积极探索 TPU、AWS Inferentia、Groq LPU 等专用芯片的优化策略,以及 FPGA/ASIC 的定制化推理加速。PagedAttention、FlashAttention 等底层优化多源自国外顶尖实验室。

#### 2. 国内研究: 场景驱动与算法创新并重

国内以百度、阿里、华为、清华、中科院等为代表的研究力量在垂直场景优化和算法效率提升方面表现突出。百度的 ERNIE 系列、阿里的通义千问(Qwen)、华为的盘古(Pangu)模型在中文语境和多模态推理方面具有特色;清华的 FastTransformer 实验室在稀疏化、量化方面贡献显著。

国内研究更注重产业落地与边缘部署。针对国内云计算和边缘计算的特定需求,研究重点集中在移动端部署(如小米、OPPO 的端侧大模型)、多模态推理优化(文生图、文生视频的高效推理)以及国产 AI 芯片(昇腾、寒武纪、海光)的适配优

化。此外, 国内在长文本处理(如月之暗面的 Kimi 模型)和智能体(Agent)推理调度方面形成了特色研究方向。

然而, 在开源基础设施和底层系统软件方面, 国内仍以跟进和适配为主, 原创性的系统架构(如 vLLM 级别的创新)相对不足, 这反映了在系统软件栈积累上的差距。

表 1-1 国内外大模型推理优化研究特点对比

维度	国外研究	国内研究
核心优势	系统架构、硬件协同	场景驱动、算法创新
代表工作	vLLM <sup>[6]</sup> , DistServe <sup>[7]</sup>	QLoRA <sup>[17]</sup> , Mooncake <sup>[25]</sup>
技术重点	底层系统、内存管理	模型压缩、端侧部署
产业落地	云基础设施	端侧 AI、国产芯片适配

### 1.3.6 现有研究的局限性与启示

综合上述分析, 当前 LLM 推理优化研究在以下维度存在显著局限:

**(1) 调度粒度与资源状态的紧耦合:**现有系统(如 SGLang、Mooncake)的调度决策严重依赖 KV Cache 的物理位置, 前缀缓存感知路由导致负载热点倾斜。路由器被迫在计算负载均衡与缓存命中率之间做困难权衡(Cache-Load Balancing Trade-off), 缺乏计算-缓存联合优化的全局视角。

**(2) 资源配置的静态化与刚性约束:**现有 PD 分离系统(DistServe、Splitwise)在部署时固定 Prefill 与 Decode 实例比例, 无法在运行期间根据实际负载动态调整。传统部署方法的配置搜索开销巨大, 且模型加载时间较长, 难以处理突发流量下的资源重分配需求。

**(3) 缺乏跨阶段的细粒度资源协同机制:**Prefill 与 Decode 阶段的资源需求(计算 vs 内存)在时域和空域上互补, 但现有系统缺乏在两个阶段之间实时优化资源配置的细粒度机制, 导致严重的资源利用率失衡(Prefill 实例计算饱和但显存闲置, Decode 实例反之)。

**(4) 对异构性和动态性的适应性不足:**现有方案多假设同构的 GPU 集群和稳态负载, 对混合型号 GPU、网络拓扑差异以及重尾分布(Heavy-tailed)的动态负载缺乏有效建模, 导致实际部署中资源碎片化严重。

针对上述局限, 本文拟从请求级批处理优化与层级资源分配两个层面, 构建大模型推理服务的统一资源管理框架, 突破静态配置与动态需求间的结构性矛盾, 实现计算-内存资源的细粒度协同优化。

## 1.4 论文主要工作

针对大语言模型 (LLM) 推理服务中资源需求难以预测、静态配置效率低下、批处理策略粗放等关键挑战, 本文围绕大模型推理服务的批式调度与 KV 缓存优化这一核心主题, 构建“批处理优化-层级资源分配”的完整技术体系。如图 1-2 所示, 研究工作涵盖静态场景下的批处理与部署优化, 以及动态场景下的 PD 分离架构资源管理, 形成从任务接入到资源释放的全生命周期管理闭环。

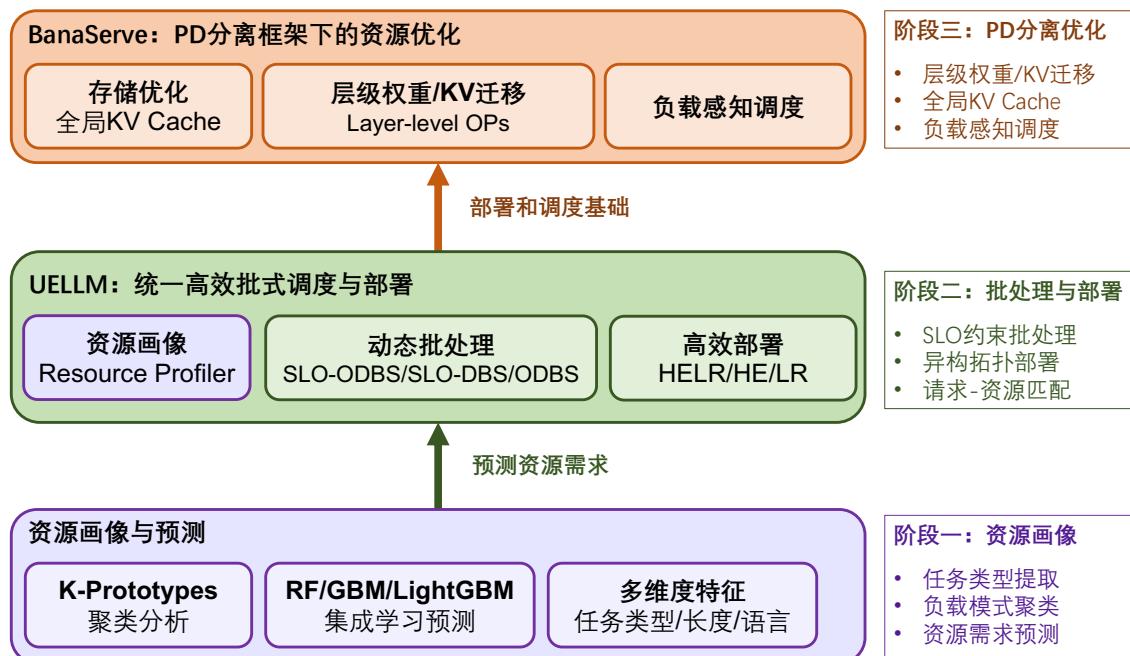


图 1-2 论文整体研究框架

### 1.4.1 面向 LLM 推理任务的高效批式调度和部署方法

针对传统系统批处理策略粗放 (FIFO)、部署配置静态化、缺乏服务质量 (SLO) 感知等缺陷, 本文提出 UELLM(Unified and Efficient LLM Inference Serving) 框架, 实现从资源画像到调度决策的端到端优化。

#### (1) 基于微调大模型的输出长度预测。

突破传统静态分析方法的局限, 认识到输出长度是决定 KV Cache 大小和计算量的关键变量。采用 ChatGLM3-6B 作为基础模型, 在 Alpaca<sup>[32]</sup> 和 Natural Questions 数据集上进行 LoRA 微调, 将输出长度预测建模为分类任务 (分桶策略: 8/16/32/64/128/.../2048 tokens)。微调后模型在测试集上达到 **99.51%** 的分桶准确率, 相比基于规则的方法显著提升了预测精度。

#### (2) SLO 与输出长度驱动的动态批处理算法 (SLO-ODBS)。

针对传统 FIFO 批处理策略导致的 KV Cache 冗余填充和 SLO 违约问题, 建立了综合考虑请求 SLO 约束和输出长度差异的批处理优化模型。算法通过权重化目

标函数平衡总延迟与总输出长度,采用贪心策略将长度相近的请求组合成批,避免为短输出请求不必要的填充计算。实验表明,SLO-ODBS 在保持低延迟的同时,将 SLO 违约率降低至接近 0%。

### (3) 面向异构拓扑的高效资源分配算法 (HELR)。

针对 LLM 分布式部署中设备映射 (Device Map) 搜索空间巨大、静态配置性能次优的问题,提出了基于动态规划的高效资源分配算法。算法综合考虑集群网络拓扑 (NVLink、PCIe 带宽异构)、GPU 计算能力差异和模型层间依赖关系,自动求解最优的层到设备映射策略。通过调整权重系数  $\alpha_1$  和  $\alpha_2$ ,可在高利用率模式 (HE) 和低延迟模式 (LR) 间灵活切换,适应不同服务等级需求。

UELLM 框架在真实 4-GPU 集群上的验证表明,相比 Morphling 和 S<sup>3</sup> 等先进方案,系统降低推理延迟 72.3% 至 90.3%,提升 GPU 利用率 1.2 倍至 4.1 倍,吞吐量提高 1.92 倍至 4.98 倍,实现了零 SLO 违约的高质量推理服务。

## 1.4.2 面向 PD 分离架构的资源管理和 KV 缓存优化方法

当面临动态变化的负载时,静态配置的系统难以维持高效运行。特别是 PD(Prefill-Decode) 分离架构中,Prefill 阶段计算密集而 Decode 阶段内存密集的固有特性导致资源利用率失衡。针对该问题,本文提出 BanaServe 优化框架,通过层级权重分配与全局 KV Cache 管理实现 PD 分离架构的性能优化。

### (1) 层级权重迁移机制。

针对 PD 分离架构中 Prefill 阶段计算密集与 Decode 阶段内存密集的资源需求差异,提出了层级 (Layer-level) 权重迁移机制。该机制支持将连续的 Transformer 层动态分配至不同的 GPU 实例,实现粗粒度负载重平衡。通过合理的层级划分,主 GPU 与辅助 GPU 可以并行处理不同的模型层,在不增加端到端延迟的情况下提升整体吞吐量。数学上证明了层级分解的正确性,确保分布式计算与单卡计算数值等价。

### (2) 注意力头级别的细粒度迁移。

为应对更细粒度的负载波动,进一步提出了注意力级别 (Attention-level) 的 KV Cache 迁移策略。该策略沿注意力头维度分割 KV Cache,选择性将部分注意力头的状态迁移至辅助 GPU,而无需传输模型权重。由于仅迁移中间激活值且传输量小,该方法可在毫秒级完成负载调整,适用于实时性要求高的动态场景。

### (3) 全局 KV Cache 存储与层间流水线传输。

针对传统前缀缓存感知路由导致的负载倾斜问题,设计了跨 Prefill 实例的全局 KV Cache 存储层,解耦缓存状态与计算位置,使调度决策无需受限于缓存局部性。针对全局存储引入的访问延迟,提出了层间流水线重叠传输机制,利用 Transformer

逐层计算特性, 将第  $i$  层计算与第  $i + 1$  层 KV Cache 预取重叠。理论分析与实验验证表明, 当 KV Cache 传输时间 (约 0.082ms) 远小于层计算时间 (约 4.22ms) 时, 可实现近透明的缓存访问。

#### (4) 负载感知调度算法。

基于全局 KV Cache 存储, 实现了完全基于实时负载的调度策略。算法周期性地测量各 Prefill 实例的综合负载 (计算 + 内存利用率), 将新请求分发至负载最轻的实例。通过消除缓存局部性对调度的约束, 系统能够实现更均衡的负载分布, 避免热点实例过载而其他实例空闲的情况。

BanaServe 在 13B 参数模型 (LLaMA-13B<sup>[2]</sup>、OPT-13B<sup>[33]</sup>) 和公开基准 (Alpaca<sup>[32]</sup>、LongBench<sup>[34]</sup>) 上的评估表明, 相比 vLLM, 系统吞吐量提升 1.2 倍至 3.9 倍, 总处理时间降低 3.9% 至 78.4%。相比 DistServe, 吞吐量提升 1.1 倍至 2.8 倍, 延迟降低 1.4% 至 70.1%, 并在突发流量下展现出卓越的鲁棒性。

### 1.4.3 主要创新点

本文围绕大模型推理服务的资源管理问题, 从批处理优化到层级部署开展了系统性研究, 主要创新点可总结为:

**(1) 提出了 SLO 感知的动态批处理与异构部署联合优化方法:** 首次将输出长度预测与 SLO 约束联合建模, 设计了 SLO-ODBS 批处理算法与 HELR 部署优化算法, 突破了传统批处理仅优化吞吐量、传统部署忽视网络拓扑异构性的局限, 实现了延迟、利用率与违约率的联合优化。

**(2) 提出了 PD 分离架构下的层级资源分配新方法:** 针对 PD 分离架构, 提出了层级权重迁移机制, 实现了计算与内存资源的合理配置, 填补了 PD 分离架构下细粒度资源分配研究空白。

**(3) 提出了全局 KV Cache 共享与计算-通信重叠机制:** 通过全局 KV Cache 存储解耦缓存状态与计算位置, 消除了前缀缓存对调度的约束。结合层间流水线传输技术, 解决了全局存储的延迟瓶颈, 实现了 PD 分离架构下的高效负载均衡。

## 1.5 论文组织架构

本文共分为五章, 各章内容安排如下:

**第一章 绪论:** 介绍大语言模型推理服务的研究背景与意义, 系统梳理国内外在算法优化、系统架构、批处理调度等方面的研究现状, 深入分析现有研究的局限性。阐述本文“批处理优化-层级部署”的技术路线与创新点。

**第二章 现有 LLM 推理任务的批式调度与 KV 缓存优化方法:** 系统梳理大模

型推理优化领域的现有技术方案。在批式任务调度方面,介绍 S<sup>3</sup> 的输出长度感知调度、Continuous Batching 的动态请求合并、Orca<sup>[35]</sup>中的 In-flight Batching 批处理机制,以及 FIFO、短作业优先、长作业优先等传统调度策略。在 KV 缓存优化方面,分析 Mooncake 的全局缓存池架构、MemServe 的弹性内存管理、vLLM 的 PagedAttention 显存优化,以及模型量化等压缩技术。最后总结现有方法在调度粒度与资源状态紧耦合、资源配置静态化、缺乏跨阶段细粒度协同机制、对异构性和动态性适应性不足等方面的局限性,为后续章节提出 UELLM 和 BanaServe 两种优化方案提供技术背景与改进动机。

**第三章 面向 LLM 推理任务的高效批式调度和部署方法: UELLM:** 研究静态场景下的批处理与部署联合优化。介绍基于微调大模型 (ChatGLM3-6B) 的输出长度预测方法 (99.51% 分桶准确率)。详细阐述 SLO 与输出长度驱动的动态批处理算法 SLO-ODBS 的数学模型 (填充浪费建模) 与双阶段贪心策略。论述面向异构拓扑的高效资源分配算法 HELR,包括基于动态规划的层到 GPU 映射优化、HE/LR 双模式切换机制。介绍 UELLM 系统实现与 4-GPU 集群实验验证,展示相比 Morphling 和 S<sup>3</sup> 在延迟、利用率、吞吐量方面的性能提升。

**第四章 面向 PD 分离架构的资源管理和 KV 缓存优化方法: BanaServe:** 针对 PD 分离架构下的资源配置问题,研究层级资源分配与 KV 缓存优化机制。阐述层级 (Layer-level) 权重迁移的资源重平衡策略,给出层级分解的数学正确性证明。介绍全局 KV Cache 存储架构设计与层间流水线重叠传输机制,分析通信-计算重叠的条件与收益。论述负载感知请求调度算法,通过 LLaMA-13B 和 OPT-13B 模型在 Alpaca 和 LongBench 基准上的实验,验证 BanaServe 相比 vLLM 和 DistServe 的性能优势。

**第五章 总结与展望:** 总结本文的主要研究成果,讨论存在的问题与局限性,展望未来研究方向,包括异构硬件感知调度、端到端延迟优化、跨地域分布式推理等。

## 第 2 章 现有 LLM 的推理任务批式任务调度和 KV 缓存优化的方法

大语言模型推理优化是一个快速发展的研究领域，学术界和工业界提出了众多技术方案。本章系统梳理现有 LLM 推理任务的批式调度方法与 KV 缓存优化技术，分析各类方案的设计思想、核心机制与适用场景，并总结现有方法的局限性，为后续章节提出 UELLM 和 BanaServe 两种优化方案提供技术背景与改进动机。

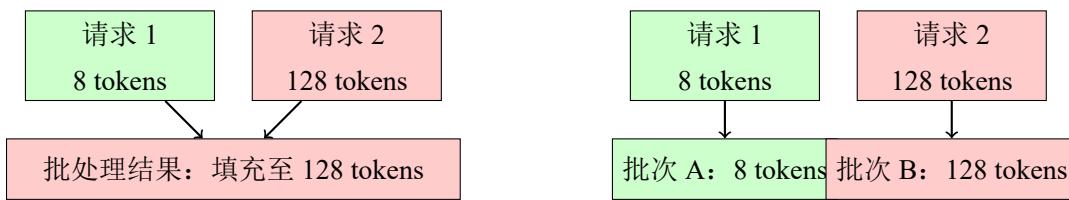
### 2.1 现有的 LLM 推理任务批式调度方法

批处理（Batching）是提升 LLM 推理吞吐量的核心手段。通过将多个请求合并处理，可以充分利用 GPU 的并行计算能力，提高硬件利用率。本节介绍从传统调度策略到面向 LLM 特性的先进批处理技术的演进过程。

#### 2.1.1 S<sup>3</sup>：输出长度感知调度

S<sup>3</sup>（SLO-aware, Size-based, Scheduling System）<sup>[21]</sup>是针对生成式 LLM 推理服务提出的批处理优化框架，其核心洞察是：**输出序列长度的差异是导致批处理效率低下的关键因素**。

传统批处理策略将不同输出长度的请求强制填充（Padding）到同一批次，导致大量无效计算。如图 2-1 所示，当短输出请求（如 8 tokens）与长输出请求（如 128 tokens）被批处理时，短请求必须等待长请求完成，造成严重的计算资源浪费。



传统方法：大量填充，资源浪费

S3 方法：按长度分组，减少填充

图 2-1 S3 与传统批处理策略对比

S3 的核心创新包括：

**(1) 输出长度预测模型。** S3 采用轻量级预测器（基于输入文本特征）估计每个请求的输出行长度，将长度相近的请求归入同一批次。预测模型在多种工作负载上达到 85% 以上的准确率。

**(2) 多维装箱调度算法。**将请求调度建模为多维装箱问题，综合考虑输出长度、SLO 约束和 GPU 显存容量，采用启发式算法优化批次组合。目标函数为：

$$\min \sum_{i=1}^N (\text{Padding}_i + \lambda \cdot \text{SLO\_Violation}_i) \quad (2-1)$$

其中  $\text{Padding}_i$  为第  $i$  个批次的填充开销， $\text{SLO\_Violation}_i$  为 SLO 违约惩罚， $\lambda$  为权衡系数。

**(3) 动态批次重构。**在请求到达和完成时动态调整批次组成，最大化吞吐量同时保证延迟约束。

S3 的局限性在于：仅优化批处理组合，未考虑模型部署配置对性能的影响；缺乏对网络拓扑异构性的感知；预测模型针对特定数据集训练，泛化能力有限。

### 2.1.2 Continuous Batching：动态请求合并

Continuous Batching（连续批处理）<sup>[6]</sup>是 vLLM 提出的动态批处理机制，打破了传统静态批处理“等待-处理-等待”的串行模式。

**核心机制：**在 Decode 阶段，每当有请求完成生成并释放资源时，系统立即从等待队列中选取新请求加入当前批次，保持 GPU 始终处于满负荷状态。如图 2-2 所示，与传统静态批处理相比，Continuous Batching 显著减少了 GPU 空闲时间。

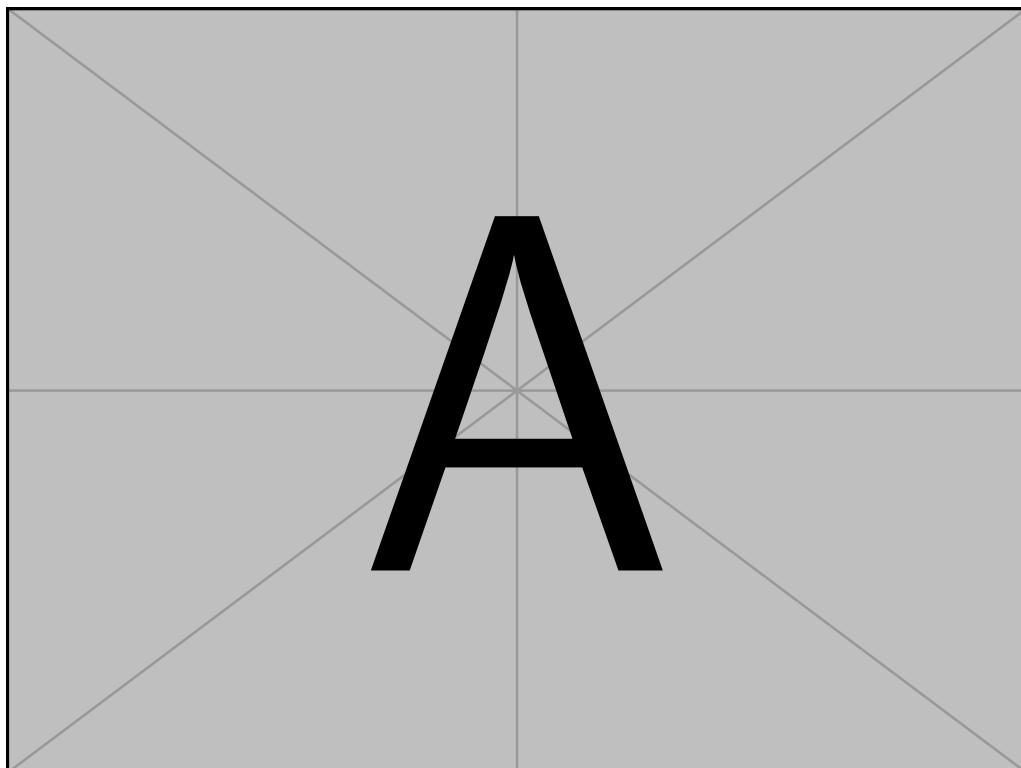


图 2-2 Static Batching 与 Continuous Batching 的时间线对比

Continuous Batching 的技术要点：

- (1) **迭代级调度**。以迭代（iteration）而非请求为粒度进行调度，每个 Decode 步骤完成后重新评估批次组成。
- (2) **显存动态管理**。配合 PagedAttention 技术（见第 2.2.3 节），动态分配和释放 KV Cache 显存，支持变长序列的高效批处理。
- (3) **抢占与恢复**。当高优先级请求到达时，可抢占低优先级请求的显存资源，被抢占请求稍后恢复执行。

Continuous Batching 的优势在于高吞吐量和低延迟，但其调度决策仅基于当前系统状态，缺乏对未来负载的预测能力；同时，未考虑 Prefill 与 Decode 阶段的资源需求差异，在混合负载场景下可能出现资源竞争。

### 2.1.3 Orca 中的 In-flight Batching 批处理机制

Orca<sup>[?]</sup>是 NVIDIA 提出的 LLM 推理服务系统，其 **In-flight Batching**（飞行中批处理）机制进一步优化了 Continuous Batching，实现了 Prefill 与 Decode 阶段的流水线并行。

**关键观察：**在 LLM 推理中，Prefill 阶段（计算密集）和 Decode 阶段（内存密集）对 GPU 资源的需求互补。传统系统串行执行两个阶段，导致资源利用率低下。

In-flight Batching 的核心设计：

(1) **阶段内批处理**。将 Prefill 和 Decode 分别批处理，但允许两个阶段在同一 GPU 上交替执行。当一批请求的 Prefill 完成后，立即开始 Decode，同时 GPU Tensor Core 可处理下一批 Prefill 请求。

(2) **细粒度同步**。采用迭代级同步而非请求级同步，减少流水线气泡（bubble）。如图 2-3 所示，通过精细的流水线调度，GPU 计算单元和内存带宽得到更充分利用。

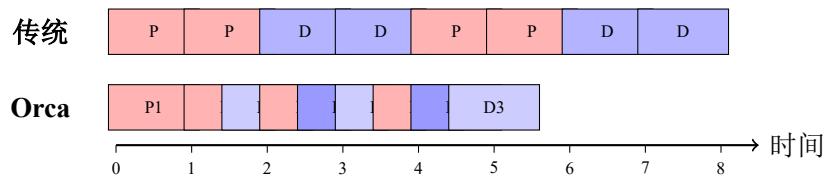


图 2-3 In-flight Batching 流水线调度示意 (P=Prefill, D=Decode)

(3) **选择性批处理**。对于 Prefill 阶段，仅对长度相近的请求批处理；对于 Decode 阶段，所有请求统一批处理，因为此时计算量与序列长度无关。

In-flight Batching 的局限在于：虽然实现了阶段内并行，但未完全解耦 Prefill 和 Decode 的资源分配；缺乏对 KV Cache 全局共享的支持，在多实例场景下缓存命中率受限。

### 2.1.4 传统调度策略

在LLM推理服务中，传统调度策略仍被广泛采用作为基准方法：

**(1) FIFO (First-In-First-Out, 先进先出)**。按请求到达顺序处理，实现简单、公平性好。但FIFO对请求特性不敏感，短请求可能被长请求阻塞，导致平均等待时间较长。

**(2) SJF (Shortest Job First, 短作业优先)**。优先处理预计执行时间短的请求。在LLM推理中，“短作业”对应输入长度短或输出长度短的请求。SJF可降低平均延迟，但可能导致长请求饥饿（starvation）。

**(3) LJF (Longest Job First, 长作业优先)**。优先处理长请求，适用于需要尽快完成大任务的场景，但会显著增加短请求的等待时间。

**(4) 优先级调度**。根据用户等级、请求类型等分配优先级，高优先级请求优先处理。需配合抢占机制避免低优先级请求长期等待。

传统策略的共性局限在于：**静态决策**，不考虑系统当前负载和资源状态；**缺乏预测**，无法利用输出长度等先验信息优化调度；**粗粒度**，以请求为调度单位，未探索更细粒度的资源分配。

## 2.2 现有的KV缓存优化方法

KV Cache（键值缓存）是LLM推理中的核心数据结构，用于存储注意力机制中的Key和Value张量，避免重复计算。然而，KV Cache的显存占用随批次大小和序列长度线性增长，成为推理扩展的主要瓶颈。本节介绍四种代表性的KV缓存优化技术。

### 2.2.1 Mooncake：全局KV缓存池

Mooncake<sup>[25]</sup>是月之暗面（Moonshot AI）提出的KV Cache-centric推理架构，其核心思想是将KV Cache从GPU显存解耦，构建全局共享的缓存池。

**系统架构：**Mooncake将推理集群划分为Prefill集群和Decode集群，两者之间通过高速RDMA网络连接。Prefill集群计算并产生KV Cache，Decode集群消费KV Cache进行自回归生成。关键创新在于引入**分布式KV Cache池**，作为独立存储层管理所有缓存数据。

**核心技术：**

**(1) 分层缓存存储。**KV Cache池采用分层设计：热数据驻留GPU显存，温数据存放CPU内存，冷数据持久化到SSD。通过LRU（Least Recently Used）策略动态迁移数据，平衡访问延迟和存储成本。

**(2) 前缀复用。**利用请求间的前缀共享（如系统提示、多轮对话历史），通过前缀树（Trie）索引快速匹配缓存。匹配成功后，仅需计算增量部分的KV Cache，显著减少Prefill计算量。

**(3) 异步预取。**基于请求特征预测未来访问模式，异步将所需KV Cache从远程节点或下层存储预取到本地GPU，隐藏传输延迟。

Mooncake的优势在于高缓存命中率和弹性扩展能力，但其调度决策与缓存位置紧耦合：路由器必须考虑KV Cache的物理位置进行请求分发，导致负载不均衡——热点缓存节点可能过载，而冷节点资源闲置。

## 2.2.2 MemServe：弹性内存管理

MemServe<sup>[26]</sup>聚焦于PD（Prefill-Decode）分离架构下的内存管理优化，提出弹性内存池（Elastic Memory Pool）概念。

**核心机制：**

**(1) 内存解耦。**将KV Cache存储与计算实例分离，构建独立的内存服务器集群。计算实例（GPU）通过高速网络访问远程内存，按需获取KV Cache。

**(2) 动态扩缩容。**根据负载变化动态调整内存池容量：高峰时段增加内存节点，低谷时段释放资源。结合Kubernetes等容器编排平台实现自动化运维。

**(3) 冗余消除。**识别并合并重复的KV Cache块（如共享前缀），采用引用计数管理生命周期，减少存储冗余。

MemServe的局限性在于：远程内存访问引入额外延迟，对网络带宽要求高；内存池的管理开销随规模增长，超大规模部署时可能成为瓶颈；未充分考虑Prefill与Decode实例间的负载均衡。

## 2.2.3 vLLM：PagedAttention显存优化

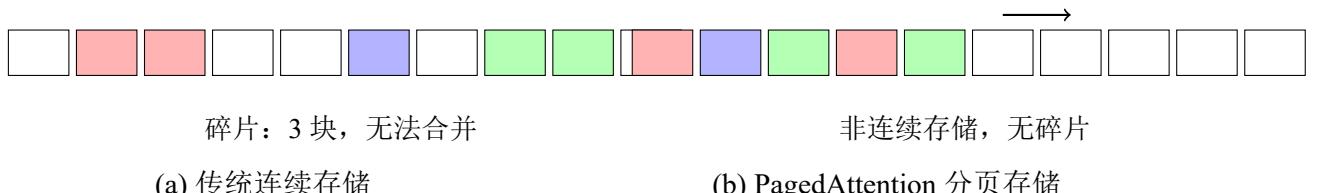
vLLM<sup>[6]</sup>是伯克利大学提出的开源LLM推理引擎，其**PagedAttention**技术从根本上解决了KV Cache的显存碎片问题。

**问题背景：**传统LLM推理系统将KV Cache存储为连续的显存块，如图2-4(a)所示。由于请求长度动态变化，频繁分配和释放导致严重的显存碎片，实际可用显存远低于物理容量。

**PagedAttention核心设计：**

**(1) 块表映射。**将KV Cache划分为固定大小的块（Block，通常为512 tokens），通过块表（Block Table）记录逻辑块到物理块的映射关系。逻辑上连续的KV Cache在物理上可以分散存储。

**(2) 动态分配。**按需分配物理块，请求完成时立即回收。由于块大小固定，回



收的块可立即被其他请求复用，消除外部碎片。

**(3) 共享机制。**通过块表引用计数实现 KV Cache 共享：多个请求共享同一前缀时，物理块只存储一份，块表记录多个引用。Copy-on-Write 机制确保写操作的安全性。

PagedAttention 使 vLLM 的显存利用率从传统系统的 40-50% 提升至 90% 以上，支持更大的批次和更长的上下文。但其设计针对单实例优化，在多实例分布式场景下缺乏全局缓存共享能力。

#### 2.2.4 量化：模型压缩与加速

量化（Quantization）通过降低模型权重和激活值的数值精度，减少显存占用和计算量，是 LLM 推理优化的基础技术。

**(1) 权重量化（Weight Quantization）。**将 FP16/BF16 精度的模型权重压缩至 INT8、INT4 甚至更低比特。代表性方法包括：

- **LLM.int80<sup>[12]</sup>**：对异常值（outliers）保持 FP16 精度，其余权重用 INT8 表示，减少精度损失。
- **GPTQ<sup>[13]</sup>**：基于近似二阶信息的逐层量化，支持 INT4 精度且精度损失可控。
- **AWQ<sup>[?]</sup>**：激活感知的权重量化，保护对激活值敏感的重要权重。

**(2) KV Cache 量化。**对 KV Cache 进行低比特量化，显著降低长上下文场景的显存压力。例如，将 KV Cache 从 FP16 量化为 INT8，显存占用减半，支持 2 倍长的上下文。

**(3) 混合精度量化。**根据层的重要性或激活分布动态选择精度，关键层用高精度，次要层用低精度，平衡效率和精度。

量化的优势在于通用性强，可与前述批处理、缓存优化技术正交叠加。但其局限在于：量化本身不解决资源调度和负载均衡问题；极低比特量化（如 INT4 以下）可能显著影响模型能力；需要硬件支持低比特运算才能发挥加速效果。

## 2.3 现有方法的缺陷

综合上述分析，当前LLM推理优化研究在以下维度存在显著局限，这些局限性构成了本文提出UELLM和BanaServe的改进动机。

### 2.3.1 调度粒度与资源状态的紧耦合

现有系统（如SGLang、Mooncake）的调度决策严重依赖KV Cache的物理位置。前缀缓存感知路由（Cache-aware Routing）虽然提高了缓存命中率，但引入了负载热点倾斜问题：路由器被迫在计算负载均衡与缓存命中率之间做困难权衡（Cache-Load Balancing Trade-off）。

具体表现为：高缓存命中率的节点持续接收新请求，计算负载迅速饱和；而低命中率节点虽有剩余算力，却因缺乏缓存数据无法分担负载。这种正反馈效应导致集群资源利用率严重失衡，部分节点过载而其他节点空闲。

### 2.3.2 资源配置的静态化与刚性约束

现有PD分离系统（DistServe、Splitwise、MemServe）在部署时固定Prefill与Decode实例比例，无法在运行期间根据实际负载动态调整。静态配置的问题在于：

**(1) 负载失配。**实际生产环境中，Prefill和Decode的负载比例随时间动态变化（如白天交互式应用Decode密集，夜间批处理任务Prefill密集）。固定比例导致某一阶段资源过剩而另一阶段成为瓶颈。

**(2) 扩容滞后。**突发流量（Bursty Traffic）下，静态系统无法快速重新分配资源，导致队列堆积和SLO违约。

**(3) 配置搜索开销。**Morphling等配置优化方法需要对每个候选配置进行压力测试，产生巨大时间和计算开销，难以适应快速变化的负载模式。

### 2.3.3 缺乏跨阶段的细粒度资源协同机制

Prefill与Decode阶段的资源需求（计算vs内存）在时域和空域上互补，但现有系统缺乏在两个阶段之间实时优化资源配置的细粒度机制。

如图2-5所示，Prefill实例通常计算利用率 $>95\%$ 但显存利用率 $<40\%$ ，Decode实例则相反（计算 $<40\%$ ，显存 $>90\%$ ）。这种资源利用率失衡导致严重的资源浪费：Prefill实例的大量显存闲置无法被Decode利用，而Decode实例的计算能力未被充分利用。

现有系统缺乏在实例间动态迁移计算任务或内存状态的能力，无法根据实时负载重新平衡资源。

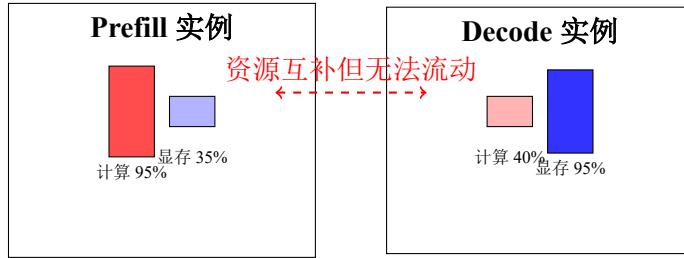


图 2-5 Prefill 与 Decode 实例资源利用率失衡

### 2.3.4 对异构性和动态性的适应性不足

现有方案多假设同构的 GPU 集群和稳态负载，对以下实际场景缺乏有效支持：

**(1) 硬件异构性。**实际数据中心常混合部署不同型号 GPU（如 A100、H100、RTX 3090），其计算能力、显存容量、互联带宽差异显著。现有方法未充分考虑这些差异，导致资源碎片化严重。

**(2) 网络拓扑异构性。**GPU 间通过 NVLink、PCIe、InfiniBand 等多种互联方式连接，带宽差异可达 10 倍以上。现有部署策略忽视拓扑结构，可能将频繁通信的层分配到高延迟链路上。

**(3) 负载动态性。**生产环境负载呈现重尾分布（Heavy-tailed）和突发特性，现有静态或准静态策略无法及时响应，导致服务质量波动。

针对上述局限，本文后续章节提出两种优化方案：第三章的 **UELLM** 聚焦于静态场景下的批处理优化与异构部署，通过输出长度预测和动态规划实现高效资源分配；第四章的 **BanaServe** 针对动态 PD 分离架构，通过层级资源迁移和全局 KV Cache 共享实现细粒度负载均衡。

## 第3章 数学符号和公式

### 3.1 数学符号

模板中使用 `unicode-math` 宏包来配置数学符号，

研究生《写作指南》要求量及其单位所使用的符号应符合国家标准《国际单位制及其应用》(GB 3100—1993)、《有关量、单位和符号的一般原则》(GB/T 3101—1993) 的规定，但是与 `TeX` 默认的美国数学学会 (AMS) 的符号习惯有所区别。

英文论文的数学符号使用 `TeX` 默认的样式。论文以中文为主要撰写语言按照国标建议的配置数学字体格式：

- (1) 大写希腊字母默认为斜体，如

$$\Gamma \Delta \Theta \Lambda \Xi \Pi \Sigma \Upsilon \Phi \Psi \Omega.$$

注意有限增量符号  $\Delta$  固定使用正体，模板提供了 `\increment` 命令。

- (2) 小于等于号和大于等于号使用倾斜的字形  $\leqslant$ 、 $\geqslant$ 。
- (3) 积分号使用正体，比如  $\int$ 、 $\oint$ 。
- (4) 行间公式积分号的上下限位于积分号的上下两端，比如

$$\int_a^b f(x) dx.$$

行内公式为了版面的美观，统一居右侧，如  $\int_a^b f(x) dx$ 。

- (5) 偏微分符号  $\partial$  使用正体。
- (6) 省略号 `\dots` 按照中文的习惯固定居中，比如

$$1, 2, \dots, n \quad 1 + 2 + \dots + n.$$

- (7) 实部  $\text{Re}$  和虚部  $\text{Im}$  的字体使用罗马体。

以上数学符号样式的差异可以在模板中统一设置。另外国标还有一些与 AMS 不同的符号使用习惯，需要用户在写作时进行处理：

- (1) 数学常数和特殊函数名用正体，如

$$\pi = 3.14 \dots; \quad i^2 = -1; \quad e = \lim_{n \rightarrow \infty} \left(1 + \frac{1}{n}\right)^n.$$

- (2) 微分号使用正体，比如  $dy/dx$ 。
- (3) 向量、矩阵和张量用粗斜体 (`\mathbf`)，如  $\mathbf{x}$ 、 $\mathbf{\Sigma}$ 、 $\mathbf{T}$ 。

(4) 自然对数用  $\ln x$  不用  $\log x$ 。

关于数学符号更多的用法，参考 `unicode-math` 宏包的使用说明，全部数学符号命的令参考 `unimath-symbols`，也可以参考 Stack Overflow 上的答案 What are all the font styles I can use in math mode?。

关于量和单位推荐使用 `siunitx` 宏包，可以方便地处理希腊字母以及数字与单位之间的空白，比如： $6.4 \times 10^6 \text{ m}$ ,  $9 \mu\text{m}$ ,  $\text{kg} \cdot \text{m} \cdot \text{s}^{-1}$ ,  $10^\circ\text{C} \sim 20^\circ\text{C}$ 。

## 3.2 数学公式

数学公式可以使用 `equation` 和 `equation*` 环境。注意数学公式的引用应前后带括号，建议使用 `\eqref` 命令，比如式(3-1)。

$$\frac{1}{2\pi i} \int_{\gamma} f = \sum_{k=1}^m n(\gamma; a_k) \mathcal{R}(f; a_k) \quad (3-1)$$

注意公式编号的引用应含有圆括号，可以使用 `\eqref` 命令。

晶体衍射基础的著名公式——布拉格方程：

$$2d \sin \theta = k\lambda, \quad k = 1, 2, 3 \dots \quad (3-2)$$

式中  $d$  —— 晶面间距 (nm);

$\theta$  —— 入射线与晶面的夹角 (rad);

$\lambda$  —— X 射线波长 (nm)。

$k$  —— 公式中第一次出现的物理量代号应给予注释，注释的转行应与破折号“——”后第一个字对齐。

多行公式尽可能在 “=” 处对齐，推荐使用 `align` 环境。

$$a = b + c + d + e \quad (3-3)$$

$$= f + g \quad (3-4)$$

此外需要注意：公式需紧挨段前文字，不可空行，不然会导致公式独立成段，如下错误效果。公式前文字公式前文字公式前文字公式前文字公式前文字。

$$\frac{1}{2\pi i} \int_{\gamma} f = \sum_{k=1}^m n(\gamma; a_k) \mathcal{R}(f; a_k) \quad (3-5)$$

公式后文字公式后文字公式后文字公式后文字公式后文字公式后

文字公式后文字公式后文字公式后文字。正确效果，如下：

$$\frac{1}{2\pi i} \int_{\gamma} f = \sum_{k=1}^m n(\gamma; a_k) \mathcal{R}(f; a_k) \quad (3-6)$$

公式后文字公式后文字公式后文字公式后文字公式后文字公式后文字公式后。

### 3.3 数学定理

定理环境的格式可以使用 `amsthm` 或者 `ntheorem` 宏包配置。用户在导言区载入这两者之一后，模板会自动配置 `thoerem`、`proof` 等环境。

**定理 3.1 (Lindeberg–Lévy 中心极限定理):** 设随机变量  $X_1, X_2, \dots, X_n$  独立同分布，且具有期望  $\mu$  和有限的方差  $\sigma^2 \neq 0$ ，记  $\bar{X}_n = \frac{1}{n} \sum_{i=1}^n X_i$ ，则

$$\lim_{n \rightarrow \infty} P\left(\frac{\sqrt{n}(\bar{X}_n - \mu)}{\sigma} \leq z\right) = \Phi(z), \quad (3-7)$$

其中  $\Phi(z)$  是标准正态分布的分布函数。

证明：Trivial. ■

同时模板还提供了 `assumption`、`definition`、`proposition`、`lemma`、`theorem`、`axiom`、`corollary`、`exercise`、`example`、`remar`、`problem`、`conjecture` 这些相关的环境。

### 3.4 数学字体

按照《撰写规范》表达式字体可以采用 Times New Roman、Xits Math 或 Cambria Math (MS Word 默认字体)。Cambria Math 缺少部分样式，例如：积分符号设定为 upright 也看起来没有变化。

TeX Gyre Termes Math 字体 (Times New Roman 的 TeX 克隆版) 样例：

$$\frac{1}{2\pi i} \int_{\gamma} f = \sum_{k=1}^m n(\gamma; a_k) \mathcal{R}(f; a_k) \quad (3-8)$$

Cambria Math 字体样例：

$$\frac{1}{2\pi i} \int_{\gamma} f = \sum_{k=1}^m n(\gamma; a_k) \mathcal{R}(f; a_k) \quad (3-9)$$

Xits Math 字体样例：

$$\frac{1}{2\pi i} \int_{\gamma} f = \sum_{k=1}^m n(\gamma; a_k) \mathcal{R}(f; a_k) \quad (3-10)$$

STIX Math 字体样例：

$$\frac{1}{2\pi i} \int_{\gamma} f = \sum_{k=1}^m n(\gamma; a_k) \mathcal{R}(f; a_k) \quad (3-11)$$

## 第 4 章 引用文献的标注

模板支持 BibTeX 和 BibLaTeX 两种方式处理参考文献。下文主要介绍 BibTeX 配合 `natbib` 宏包的主要使用方法。

### 4.1 顺序编码制

在顺序编码制下，默认的 `\cite` 命令同 `\citep` 一样，即序号置于方括号中，引文页码会放在括号外。统一处引用的连续序号会自动用短横线连接。如多次引用同一文献，可能需要标注页码，例如：引用第二页<sup>[36]2</sup>，引用第五页<sup>[36]5</sup>。

<code>\cite{zhangkun1994}</code>	⇒ [36] 不带页码的上标引用
<code>\citet{zhangkun1994}</code>	⇒ 张昆等 <sup>[36]</sup>
<code>\citep{zhangkun1994}</code>	⇒ [36]
<code>\cite[42]{zhangkun1994}</code>	⇒ [36] <sup>42</sup> 手动带页码的上标引用
<code>\cite{zhangkun1994, zhukezhen1973}</code>	⇒ [36-37] 一次多篇文献的上标引用

### 4.2 著者-出版年制

著者-出版年制下的 `\cite` 跟 `\citet` 一样。

<code>\cite{zhangkun1994}</code>	⇒ 张昆等(1994)
<code>\citet{zhangkun1994}</code>	⇒ 张昆等(1994)
<code>\citep{zhangkun1994}</code>	⇒ (张昆等, 1994)
<code>\cite[42]{zhangkun1994}</code>	⇒ (张昆等, 1994) <sup>42</sup>
<code>\citep{zhangkun1994, zhukezhen1973}</code>	⇒ (张昆等, 1994; 竺可桢, 1973)

#### 4.2.1 其他引用注意事项

注意，引文参考文献的每条都要在正文中标注<sup>[36-69]</sup>。

引用测试：2 个连续引用<sup>[36-37]</sup>，2 个间隔<sup>[36,38]</sup>，3 个连续<sup>[36-38]</sup>。

如参考文献中需要使用上标或者下标，使用数学环境书写 $\mathrm{Ba}_{\{3\}}\mathrm{CoSb}_{\{2\}}\mathrm{O}_{\{9\}}$ ，例如该文献<sup>[70]</sup>。根据 gbt7714 规定著者姓名自动转为大写。西文的题名、期刊名的小写不自动处理，需要自行处理以符合信息资源本身文种的习惯用法。

## 第 5 章 English and lower-case Example

If your supervisor is a foreign resident, or if your supervisor or defense committee specifically allows writing in English, the thesis may be written in English as the primary language. Please check with your supervisor or department secretary to confirm if you can write in English.

### 5.1 Reference guide

Writing in English still requires the Chinese reference standard GB/T 7714-2015.

## 结 论

学位论文的结论作为论文正文的最后一章单独排写，但不加章标题序号。

结论应是作者在学位论文研究过程中所取得的创新性成果的概要总结，不能与摘要混为一谈。博士学位论文结论应包括论文的主要结果、创新点、展望三部分，在结论中应概括论文的核心观点，明确、客观地指出本研究内容的创新性成果（含新见解、新观点、方法创新、技术创新、理论创新），并指出今后进一步在本研究方向进行研究工作的展望与设想。对所取得的创新性成果应注意从定性和定量两方面给出科学、准确的评价，分（1）、（2）、（3）…条列出，宜用“提出了”、“建立了”等词叙述。

在评价自己的研究成果时，要实事求是，除非有足够的证据表明自己的研究是“首次”、“领先”、“填补空白”的，否则应避免使用这些或类似词语

## 参考文献

- [1] ACHIAM J, ADLER S, AGARWAL S, et al. GPT-4 Technical Report[A]. 2023.
- [2] TOUVRON H, LAVRIL T, IZACARD G, et al. LLaMA: Open and Efficient Foundation Language Models[A]. 2023.
- [3] Anthropic. Claude[EB/OL]. 2025. <https://claude.ai>.
- [4] WU C J, RAGHAVENDRA R, GUPTA U, et al. Sustainable AI: Environmental Implications, Challenges and Opportunities[J]. Proceedings of Machine Learning and Systems, 2022, 4: 795-813.
- [5] HE Y, XU M, WU J, et al. UELLM: A Unified and Efficient Approach for Large Language Model Inference Serving[C]//Proceedings of the 22nd International Conference on Service-Oriented Computing (ICSOC). Springer, 2024: 218-235.
- [6] KWON W, LI Z, ZHUANG S, et al. Efficient Memory Management for Large Language Model Serving with PagedAttention[C]//Proceedings of the 29th ACM Symposium on Operating Systems Principles (SOSP). 2023: 611-626.
- [7] ZHONG Y, LIU S, CHEN J, et al. DistServe: Disaggregating Prefill and Decoding for Goodput-optimized Large Language Model Serving[C]//Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI). 2024: 193-210.
- [8] LEVIATHAN Y, KALMAN M, MATIAS Y. Fast Inference from Transformers via Speculative Decoding[J]. International Conference on Machine Learning (ICML), 2023: 19274-19286.
- [9] CHEN C, BORGEAUD S, IRVING G, et al. Accelerating Large Language Model Decoding with Speculative Decoding[J]. International Conference on Machine Learning (ICML), 2023: 6159-6181.
- [10] SCHUSTER T, FISCH A, GUPTA J, et al. Confident Adaptive Language Modeling[J]. Advances in Neural Information Processing Systems (NeurIPS), 2022, 35: 17456-17472.
- [11] LI Z, LI S, ZHANG M, et al. CascadeBERT: Accelerating Inference of Pre-trained Language Models via Cascade Breaking[C]//Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL). 2020: 3876-3886.
- [12] DETTMERS T, LEWIS M, BELKADA Y, et al. LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale[J]. Advances in Neural Information Processing Systems (NeurIPS), 2022, 35: 30318-30332.
- [13] FRANTAR E, ASHKBOOS S, HOEFLER T, et al. GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers[J]. International Conference on Learning Representations (ICLR), 2023.
- [14] WANG H, MA S, DONG L, et al. BitNet: Scaling 1-bit Transformers for Large Language Models[A]. 2023.
- [15] GU Y, DONG L, WEI F, et al. Knowledge Distillation of Large Language Models[A]. 2023.

- 
- [16] WU C, ZHAO W, LI B, et al. LaMini-GPT: Large Language Models with Minimal Data and Parameters[A]. 2023.
  - [17] DETTMERS T, PAGNONI A, HOLTMAN A, et al. QLoRA: Efficient Finetuning of Quantized LLMs[J]. Advances in Neural Information Processing Systems (NeurIPS), 2023, 36.
  - [18] XIAO G, TIAN Y, CHEN B, et al. StreamingLLM: Efficient Streaming Language Models with Attention Sinks[A]. 2023.
  - [19] DAO T, FU D Y, ERMON S, et al. FlashAttention: Fast and Memory-efficient Exact Attention with IO-awareness[J]. Advances in Neural Information Processing Systems (NeurIPS), 2022, 35: 16344-16359.
  - [20] CHEN T, MOREAUT J, JIANG Z, et al. TVM: An Automated End-to-End Optimizing Compiler for Deep Learning[C]//Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI). 2018: 578-594.
  - [21] JIN C, ZHANG Z, JIANG X, et al. S<sup>3</sup>: Increasing GPU Utilization during Generative Inference for Higher Throughput[A]. 2023.
  - [22] ZHENG L, CHIANG W L, SHENG Y, et al. SGLang: Efficient Execution of Structured Language Model Programs[C]//Advances in Neural Information Processing Systems (NeurIPS): Vol. 37. 2024.
  - [23] GUJARATI A, SHEKHAR S, GARG T, et al. Serving DNNs like Clockwork: Performance Predictability from the Bottom Up[C]//Proceedings of the 14th USENIX Conference on Operating Systems Design and Implementation (OSDI). 2020: 443-462.
  - [24] PATEL P, CHOUKSE E, ZHANG C, et al. Splitwise: Efficient Generative LLM Inference Using Phase Splitting[C]//Proceedings of the 51st Annual International Symposium on Computer Architecture (ISCA). 2024: 118-132.
  - [25] QIN R, LI Z, HE W, et al. Mooncake: A KVCache-centric Disaggregated Architecture for LLM Serving[A]. 2024.
  - [26] HU C, HUANG H, HU J, et al. MemServe: Context Caching for Disaggregated LLM Serving with Elastic Memory Pool[A]. 2024.
  - [27] WANG L, YANG L, YU Y, et al. Morphling: Fast, Near-Optimal Auto-Configuration for Cloud-Native Model Serving[C]//Proceedings of the ACM Symposium on Cloud Computing (SoCC). 2021: 639-653.
  - [28] XIAO W, REN S, LI Y, et al. AntMan: Dynamic Scaling on GPU Clusters for Deep Learning [C]//Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI). 2022: 533-549.
  - [29] QIN R, YANG Y, ZHENG Q, et al. Allox: Compute Allocation in Hybrid Clusters[C]// Proceedings of the Eighteenth European Conference on Computer Systems (EuroSys). 2023: 587-603.
  - [30] SHENG Y, ZHENG L, YUAN B, et al. FlexGen: High-throughput Generative Inference of Large Language Models with a Single GPU[C]//Proceedings of the 40th International Conference on Machine Learning (ICML). PMLR, 2023: 31094-31116.

- [31] YANG Z, LUAN Z, LI B, et al. SkyPilot: An Intercloud Broker for Sky Computing[C]// Proceedings of the 17th USENIX Symposium on Operating Systems Design and Implementation (OSDI). 2023: 577-596.
- [32] TAORI R, GULRAJANI I, ZHANG T, et al. Stanford Alpaca: An Instruction-following LLaMA model[J/OL]. GitHub repository, 2023. [https://github.com/tatsu-lab/stanford\\_alpaca](https://github.com/tatsu-lab/stanford_alpaca).
- [33] ZHANG S, ROLLER S, GOYAL N, et al. OPT: Open Pre-trained Transformer Language Models[A]. 2022. arXiv: 2205.01068.
- [34] BAI Y, LV X, ZHANG J, et al. Longbench: A bilingual, multitask benchmark for long context understanding[A]. 2023.
- [35] YU G I, JEONG J S, KIM G W, et al. Orca: A Distributed Serving System for Transformer-Based Generative Models[C/OL]//16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22). Carlsbad, CA: USENIX Association, 2022: 521-538. <https://www.usenix.org/conference/osdi22/presentation/yu>.
- [36] 张昆, 冯立群, 余昌钰, 等. 机器人柔性手腕的球面齿轮设计研究[J]. 清华大学学报: 自然科学版, 1994, 34(2): 1-7.
- [37] 竺可桢. 物理学论[M]. 北京: 科学出版社, 1973: 56-60.
- [38] DUPONT B. Bone marrow transplantation in severe combined immunodeficiency with an unrelated MLC compatible donor[C]//WHITE H J, SMITH R. Proceedings of the third annual meeting of the International Society for Experimental Hematology. Houston: International Society for Experimental Hematology, 1974: 44-46.
- [39] 郑开青. 通讯系统模拟及软件[D]. 北京: 清华大学无线电系, 1987.
- [40] 姜锡洲. 一种温热外敷药制备方案: 中国, 88105607.3[P]. 1980-07-26.
- [41] 中华人民共和国国家技术监督局. GB3100-3102. 中华人民共和国国家标准-量与单位[S]. 北京: 中国标准出版社, 1994.
- [42] MERKT F, MACKENZIE S R, SOFTLEY T P. Rotational Autoionization Dynamics in High Rydberg States of Nitrogen[J]. J Chem Phys, 1995, 103: 4509-4518.
- [43] MELLINGER A, VIDAL C R, JUNGEN C. Laser reduced fluorescence study of the carbon monoxide nd triplet Rydberg series - Experimental results and multichannel quantum defect analysis[J]. J Chem Phys, 1996, 104: 8913-8921.
- [44] BIXON M, JORTNER J. The dynamics of predissociating high Rydberg states of NO[J]. J Chem Phys, 1996, 105: 1363-1382.
- [45] 马辉, 李俭, 刘耀明, 等. 利用 REMPI 方法测量 BaF 高里德堡系列光谱[J]. 化学物理学报, 1995, 8: 308-311.
- [46] CARLSON N W, TAYLOR A J, JONES K M, et al. Two-step polarization-labeling spectroscopy of excited states of Na2[J]. Phys Rev A, 1981, 24: 822-834.
- [47] TAYLOR A J, JONES K M, SCHAWLOW A L. Scanning pulsed-polarization spectrometer applied to Na2[J]. J Opt Soc Am, 1983, 73: 994-998.
- [48] TAYLOR A J, JONES K M, SCHAWLOW A L. A study of the excited  $1\Sigma^+$  states in Na2[J]. Opt Commun, 1981, 39: 47-50.

- [49] SHIMIZU K, SHIMIZU F. Laser induced fluorescence spectra of the a  $3\Pi_u$ -X  $1\Sigma_g^+$  band of Na<sub>2</sub> by molecular beam[J]. J Chem Phys, 1983, 78: 1126-1131.
- [50] ATKINSON J B, BECKER J, DEMTRÖDER W. Experimental observation of the a  $3\Pi_u$  state of Na<sub>2</sub>[J]. Chem Phys Lett, 1982, 87: 92-97.
- [51] KUSCH P, HESSEL M M. Perturbations in the A  $1\Sigma_u^+$  state of Na<sub>2</sub>[J]. J Chem Phys, 1975, 63: 4087-4088.
- [52] 广西壮族自治区林业厅. 广西自然保护区[M]. 北京: 中国林业出版社, 1993.
- [53] 霍斯尼. 谷物科学与工艺学原理[M]. 李庆龙, 译. 2 版. 北京: 中国食品出版社, 1989: 15-20.
- [54] 王夫之. 宋论[M]. 刻本. 金陵: 曾氏, 1865 (清同治四年) .
- [55] 赵耀东. 新时代的工业工程师[M/OL]. 台北: 天下文化出版社, 1998[1998-09-26]. <http://www.ie.nthu.edu.tw/info/ie.newie.htm>.
- [56] 全国信息与文献工作标准化技术委员会出版物格式分委员会. GB/T 12450-2001 图书书名页[S]. 北京: 中国标准出版社, 2002.
- [57] 全国出版专业职业资格考试办公室. 全国出版专业职业资格考试辅导教材: 出版专业理论与实务·中级[M]. 2014 版. 上海: 上海辞书出版社, 2004: 299-307.
- [58] World Health Organization. Factors Regulating the Immune Response: Report of WHO Scientific Group[R]. Geneva: WHO, 1970.
- [59] PEEBLES P Z, Jr. Probability, Random Variables, and Random Signal Principles[M]. 4th ed. New York: McGraw Hill, 2001.
- [60] 白书农. 植物开花研究[M]/李承森. 植物科学进展. 北京: 高等教育出版社, 1998: 146-163.
- [61] WEINSTEIN L, SWERTZ M N. Pathogenic Properties of Invading Microorganism[M]// SODEMAN W A, Jr, SODEMAN W A. Pathologic physiology: mechanisms of disease. Philadelphia: Saunders, 1974: 745-772.
- [62] 韩吉人. 论职工教育的特点[C]//中国职工教育研究会. 职工教育研究论文集. 北京: 人民教育出版社, 1985: 90-99.
- [63] 中国地质学会. 地质评论[J]. 1936, 1(1)-. 北京: 地质出版社, 1936-.
- [64] 中国图书馆学会. 图书馆学通讯[J]. 1957(1)-1990(4). 北京: 北京图书馆, 1957-1990.
- [65] American Association for the Advancement of Science. Science[J]. 1883, 1(1)-. Washington, D.C.: American Association for the Advancement of Science, 1883-.
- [66] 傅刚, 赵承, 李佳路. 大风沙过后的思考[N/OL]. 北京青年报, 2000-04-12(14)[2002-03-06]. <http://www.bjyouth.com.cn/Bqb/20000412/B/4216%5ED0412B1401.htm>.
- [67] 萧钰. 出版业信息化迈入快车道[EB/OL]. (2001-12-19)[2002-04-15]. <http://www.creader.com/news/20011219/200112190019.htm>.
- [68] Online Computer Library Center, Inc. About OCLC: History of Cooperation[EB/OL]. 2000 [2000-01-08]. <http://www.oclc.org/about/cooperation.en.htm>.
- [69] Scitor Corporation. Project scheduler[CP/DK]. Sunnyvale, Calif.: Scitor Corporation, 1983.
- [70] KAMIYA Y, GE L, HONG T, et al. The nature of spin excitations in the one-third magnetization plateau phase of Ba<sub>3</sub>CoSb<sub>2</sub>O<sub>9</sub>[J]. Nature communications, 2018, 9(1): 1-11.

## 附录 A 补充内容

附录是与论文内容密切相关、但编入正文又影响整篇论文编排的条理和逻辑性的资料，例如某些重要的数据表格、计算程序、统计表等，是论文主体的补充内容，可根据需要设置。

### A.1 图表示例

#### A.1.1 图

附录中的图片示例（图 A-1）。

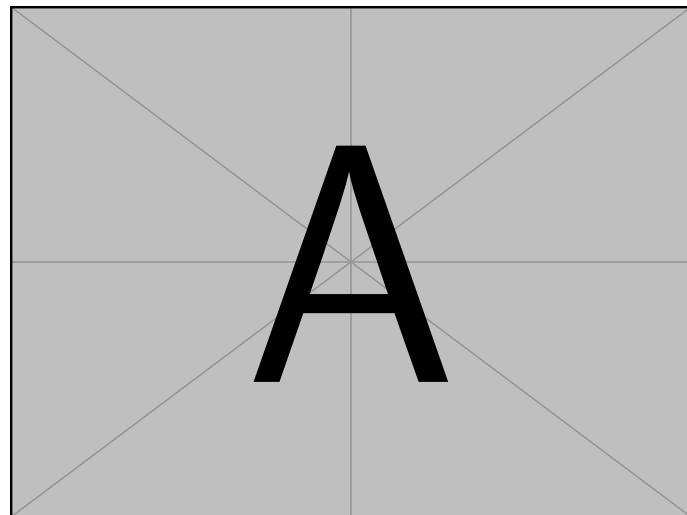


图 A-1 附录中的图片示例

#### A.1.2 表格

附录中的表格示例（表 A-1）。

### A.2 数学公式

附录中的数学公式示例（公式(A-1)）。

$$\frac{1}{2\pi i} \int_{\gamma} f = \sum_{k=1}^m n(\gamma; a_k) \mathcal{R}(f; a_k) \quad (\text{A-1})$$

表 A-1 附录中的表格示例

---

文件名	描述
sustechthesis.dtx	模板的源文件，包括文档和注释
sustechthesis.cls	模板文件
thuthesis-*.bst	BibTeX 参考文献表样式文件
thuthesis-*.bbx	BibLaTeX 参考文献表样式文件
thuthesis-*.cbx	BibLaTeX 引用样式文件

---

### A.3 源代码

附录中的代码示例：代码A-1。

```

1 class HelloWorldApp {
2     public static void main(String[] args) {
3         System.out.println("Hello World!"); // Display the
4             string.
5         for (int i = 0; i < 100; ++i) {
6             System.out.println(i);
7         }
8     }

```

代码 A-1 Java 代码示例（使用 `listings` 高亮）

### A.4 伪代码

附录中的伪代码示例（算法A-1）。

---

算法 A-1 Simulation-optimization heuristic

---

**Data:** current period  $t$ , initial inventory  $I_{t-1}$ , initial capital  $B_{t-1}$ , demand samples

**Result:** Optimal order quantity  $Q_t^*$

```

1  $r \leftarrow t;$ 
2  $\Delta B^* \leftarrow -\infty;$ 
3 while  $\Delta B \leq \Delta B^*$  and  $r \leq T$  do
4    $Q \leftarrow \arg \max_{Q \geq 0} \Delta B_{t,r}^Q(I_{t-1}, B_{t-1});$ 
5    $\Delta B \leftarrow \Delta B_{t,r}^Q(I_{t-1}, B_{t-1})/(r - t + 1);$ 
6   if  $\Delta B \geq \Delta B^*$  then
7      $Q^* \leftarrow Q;$ 
8      $\Delta B^* \leftarrow \Delta B;$ 
9   end
10   $r \leftarrow r + 1;$ 
11 end

```

---

## 致 谢

衷心感谢导师 ××× 教授对本人的精心指导。他的言传身教将使我终生受益。

感谢 ××× 教授，以及实验室全体老师和同窗们的热情帮助和支持！

本课题承蒙 ×××× 基金资助，特此致谢。

**以下内容为提示，仔细阅读后删除。**

致谢应另起页，放置在参考文献、附录之后，标题和页眉均为“致谢”。语言要诚恳、恰当、简短。

致谢对象可以包括指导教师，在研究工作中提出建议和提供帮助的人，给予转载和引用权的资料、图片、文献、研究和调查的所有者，其他应感谢的组织和个人，资助研究工作的项目基金、奖学金基金、合同单位、资助或支持的企业、组织或个人，协助完成研究工作和提供便利条件的组织或个人。致谢字数以不超过一页纸为宜。

学位论文应由学生在导师（组）的指导下独立完成；**若涉及团队工作，应注明属于团队成果，并明确个人独立完成的内容**，科学严谨，恪守规范。

#### 个人简历、在学期间完成的相关学术成果

## 个人简历

××××年××月××日出生于××××。

xxxx年xx月考入xx大学xx院(系)xx专业,xxxx年xx月本科毕业并获得xx学学士学位。

xxxx年xx月——xxxx年xx月，在xx大学xx院（系）xx学科学习并攻读（获得）xx学硕士学位。【注：博士生已获得硕士学位写“获得”，硕士生申请硕士学位应写“攻读”，本括号在使用时请删除】

获奖情况：如获三好学生、优秀团干部、×奖学金等（不含科研学术获奖）。

工作经历: .....

#### 在学期间完成的相关学术成果

特别注意，下面的引用文献部分需要使用半角括号，例如 [J]，(已被 xxxx 录用)。(本行在使用时请删除)。

学术论文

- [1] Pei S, Huang L L, Li G, et al. Magnetic Raman continuum in single-crystalline  $H_3LiIr_2O_6$ [J]. Physical Review B, 2020, 101(20): 201101. (SCI 收录, IDS 号为 LJ4UN, IF=3. 575, 对应学位论文 2.2 节和第 5 章.)
  - [2] Pei S, Tang J, Liu C, et al. Orbital-fluctuation freezing and magnetic-nonmagnetic phase transition in  $\alpha - TiBr_3$ [J]. Applied Physics Letters, 2020, 117(13): 133103. (SCI 收录, IDS 号为 NY3GK, IF=3, 597, 对应学位论文 2.2 节和第 3 章.)

申请及已获得的专利（无专利时此项不必列出）

- [3] 任天令, 杨轶, 朱一平, 等. 硅基铁电微声学传感器畴极化区域控制和电极连接的方法: 中国, CN1602118A[P]. 2005-03-30.
  - [4] Ren T L, Yang Y, Zhu Y P, et al. Piezoelectric micro acoustic sensor based on ferroelectric materials: USA, No.11/215, 102[P]. (美国发明专利申请号.)

参与的科研项目及获奖情况（无获奖时此项不必列出）

- [5] 姜锡洲, ××××× 研究, ×× 省自然科学基金项目。课题编号: ××××, 长长长长长长长  
长长长长长长长长长长长长长长长长长长长长长长长长长长长长长长长长。

[6] ×××, ××××× 研究, ×× 省自然科学基金项目。课题编号: ××××。

个人简历、在学期间完成的相关学术成果

---

[7] ×××, ××××× 研究, ××省自然科学基金项目。课题编号: ××××。