大模型推理基础设施

第 1 讲课程导论

张书豪

华中科技大学计算机学院

研究生课程课件

教学目标

核心结论

本讲建立课程导论的基本框架，包括研究对象、课程定位与后续 14 讲主线。

- 区分模型能力、服务能力与系统能力三个分析层次。
- 建立“请求 - 状态 - 调度 - 指标”的基本观察框架。
- 说明课程讲义、实验与教学代码之间的联动关系。
- 给出后续讲次共享的问题地图。

教学样例与课程落点

核心结论

nano-vllm-hust 在课程中主要作为最小教学样例使用；课程最终目标是支持学生进入 vllm-hust 的优化与开发场景。

- 前几讲以 nano-vllm-hust 为载体建立请求路径、调度控制点与 KV 状态对象的基本认识。
- 该样例用于说明系统主线，而不作为课程长期停留的唯一代码对象。
- 后续代码导读、实验扩展与课程项目将逐步过渡到更真实的 vllm-hust 开发语境。

为什么值得独立成课

核心结论

大模型进入真实在线服务后，研究重点会从“能否生成”转向“能否稳定、经济、可验证地提供服务”。

- 长上下文、RAG、智能体工作流与多租户负载共同改变了请求形态。
- TTFT、P99、goodput、成本与稳定性构成核心服务指标集合。
- 系统收益主要取决于全链路组织，而非局部 kernel 或单层算子的单点提速。

从模型问题到系统问题

核心结论

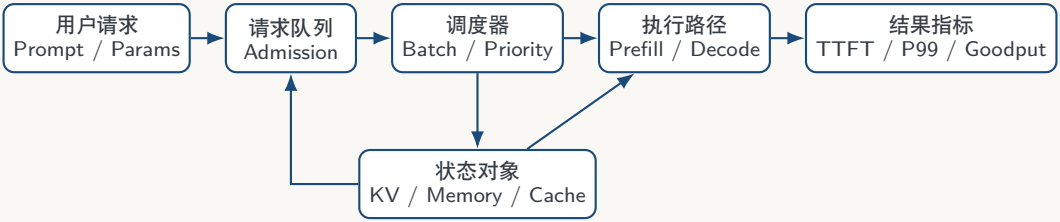
同一模型在不同运行时、调度策略和状态管理机制下，可能呈现出显著不同的服务质量。

- 模型层主要关注能力边界、精度与泛化表现。
- 系统层主要关注排队、缓存、执行路径、资源治理与回归稳定性。
- 因而课程的分析单位是服务系统，而非单一网络层或单篇算法论文。

课程分析对象示意

核心结论

本课程的分析对象是由请求、状态、控制点与结果指标共同构成的在线推理闭环。



课程重点是这些对象之间的约束关系，而不是把推理问题缩成一次前向计算。

课程边界

核心结论

本课程关注在线推理服务系统，而不是训练系统或纯算法优化。

- 分析请求进入、调度、执行与回收的完整路径。
- 分析 KV Cache、知识缓存与会话记忆等状态对象。
- 分析执行路径、异构平台与实验验证方法。

课程不重点展开的内容

核心结论

为保持系统问题定义的清晰性，课程将若干相关但非核心议题置于讲授边界之外。

- 不系统展开预训练、RLHF 与参数高效微调。
- 不将提示工程或应用层工作流设计作为主体内容。
- 不将单个 kernel 调优等同于整门系统课程。

统一视角：状态化系统

核心结论

本课程将大模型推理系统表述为“持续携带状态的在线服务系统”。

- 模型参数之外，还存在 KV Cache、知识缓存、会话记忆与调度元数据。
- 状态会跨时间、跨请求与跨设备持续传播。
- 调度、缓存与执行路径因此形成连续耦合关系。

状态化系统的含义

核心结论

状态化系统是指其当前执行决策同时受到当前请求与历史遗留状态的共同约束。

$$\text{Service Behavior}_t = f(\text{Request}_t, \text{State}_{t-1}, \text{Policy}_t)$$

- KV 会影响后续 decode 路径与显存占用。
- 会话历史会改变 prompt 长度、状态规模与排队压力。
- 运行时统计信息又会反向影响调度与资源治理。

时延分解及其系统含义

核心结论

首 token 时延与端到端生成时间均由多个系统环节共同决定，而非单一算子性能的直接映射。

$$\mathrm{TTFT} = T_{\mathrm{queue}} + T_{\mathrm{schedule}} + T_{\mathrm{prefill}} + T_{\mathrm{return}}$$

$$T_{\text{end-to-end}} = \mathrm{TTFT} + N_{\mathrm{out}} \cdot T_{\text{decode-step}} + T_{\mathrm{stream}}$$

- 若排队或状态驻留构成主导项，单个算子优化通常无法直接改善用户体验。
- 这也是后续课程分别讨论工作负载、调度、KV 与验证方法的原因。

课程主线

核心结论

课程沿着工作负载、调度、状态管理、执行路径和验证方法五条线展开。

1. 工作负载与评价指标视角
2. 请求生命周期分解
3. 调度、KV 与状态管理机制
4. 异构执行、平台适配与实验方法
5. 开源系统实践与课程项目

课程问题地图

核心结论

整门课都围绕一个总问题展开：如何在动态负载、持续状态和异构约束下构建稳定可验证的推理服务。

- 工作负载定义问题边界。
- 调度决定系统控制点。
- 状态组织决定容量与复用空间。
- 执行路径决定收益能否兑现。

课程组织方式

核心结论

课程组织采用“概念讲解 + 论文精读 + 代码导读 + 实验验证 + 课程项目”的闭环结构。

- 平时阅读与课堂讨论
- 论文精读报告
- 最小复现实验或代码导读
- 课程项目中期与最终汇报

四层材料如何配合

核心结论

整门课按“讲义定问题、tutorial 定路径、实验定证据、代码定控制点”的顺序推进。

- 讲义与课件首先建立工作负载、调度、状态与执行路径四条主线。
- Tutorials 将同一问题先落到 nano-vLLM-HUST 的最小代码对象和日志观察点，再为进入 vllm-hust 做准备。
- 课程实验要求学生围绕统一模板提交调用链、表格、日志与可否定解释。
- 教学代码同时提供参考实现、student_exercises、assignment 包与 TA 汇总脚本，以保持讲授、实践与提交的一致性。

课程产出要求

核心结论

课程产出强调三点：可复查、可复用、可继续推进。

- 系统论文精读报告一份。
- 代码导读或最小复现实验记录一份。
- 课程项目及最终汇报材料一套。

课程对象与导论课的统一判断框架

核心结论

导论部分将研究对象界定为在线推理服务系统，并用对象、控制点、指标和证据构成基本分析框架。

课程对象

- 课堂观察单位不再局限于网络层，而是请求对象、控制点、状态对象与结果指标。
- 同一模型在不同 runtime、调度器与 KV 管理策略下，会呈现不同的服务质量。

分析框架

1. 长期存在的系统对象。
2. 真正改变行为的控制点。
3. 端到端指标的变化方向。
4. 支撑解释的日志、表格和代码证据。

例子：传统部署经验的局限

核心结论

从分类模型到大模型在线服务，系统主导矛盾已由“单次前向吞吐”转向“跨轮状态组织”。

场景	主要特征	主导系统问题
传统 CV / NLP 服务	一次前向、输入输出短	吞吐、批处理、资源利用率
早期小模型对话	有生成过程，但状态较小	短任务排队、基础流式返回
大模型在线服务	长上下文、连续 decode、状态驻留	调度、状态管理、尾时延与稳定性

例子：真实服务中的典型问题

核心结论

许多服务失效并非源于模型无法生成，而是源于系统对状态与并发的承载能力不足。

现象	直接表现	更可能追查的系统对象
晚高峰突然超时	TTFT 和 P99 同时恶化	队列、准入、调度策略
显存未满却无法扩容	理论容量未用尽先失去弹性	KV 块管理、碎片、预留策略
换一套 runtime 行为大变	同一模型服务质量差异明显	请求组织、状态路径、执行栈

代码例子：从 nano-vllm 到 vllm-hust

核心结论

nano-vllm-hust 用于建立最小控制路径的理解；相同的分析方法将进一步迁移到 vllm-hust 的优化与开发任务中。

入口代码

```
from nanovllm import LLM,
SamplingParams
llm = LLM(model_path,
enforce_eager=True)
outputs = llm.generate(prompts,
sampling_params)
```

代码观察点与迁移方向

- `LLM.generate` 对应接口封装还是更深层 engine 入口。
- 请求状态首次被持有的对象位置。
- 从 nano-vllm 到 vllm-hust 的方法迁移位置。

代码例子：最小日志的观察价值

核心结论

导论阶段的插桩应以支撑主路径判断为主要目标，而不以初始阶段覆盖全部细节为目标。

最小日志样式

```
print(request_id, prompt_len,
stage)
print(batch_size, running_size)
```

观察价值

- 请求进入、执行与返回这条主路径可以被直接确认。
- 阶段日志、状态探针与 benchmark 记录可以在后续实验中逐步增加。
- 这也是课程实验组织观察点的基本方式。

常见误读与讨论任务

核心结论

本页用于区分常见误读，并将讨论收束到对象、控制点与证据三个层面。

三种常见误读

- 输出慢说明模型不够强。
- 换更快 kernel 就能解决服务问题。
- 设备利用率低就一定是算力浪费。

讨论任务

1. 判断一次“卡顿”更接近排队、状态还是执行路径问题。
2. 在 `example.py` 到 `llm_engine.py` 之间指出最值得优先阅读的两个函数。

讨论收束问题

课程为何先从系统边界切入，而不是直接从单篇机制论文切入。

课堂讨论

核心结论

课堂讨论围绕系统边界、用户感知与状态瓶颈三个维度展开。

- 大模型服务问题为何不能简化为“将模型部署出去”。
- 用户感知到的“卡顿”通常对应哪些系统现象。
- 哪些状态对象更容易成为瓶颈放大器。

本讲总结

核心结论

本讲完成了研究对象由“模型”向“服务系统”的转换，并给出了后续讲次共享的分析框架。

- 研究对象是完整服务链路，而非单一模型。
- 研究目标是稳定且可验证的系统收益。
- 后续各讲均将回到“状态对象 + 控制点 + 收益链路”的分析框架。