大模型推理基础设施

第 13 讲研究选题与课程项目工作坊

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

本讲围绕课程项目的问题定义、最小实现计划与验证路径展开。

- 识别瓶颈对象和可控变量。
- 明确影响链路和验证方法。
- 为中期汇报和最终交付做收束。

项目定义的四个基本问题

核心结论

若项目无法清楚回答以下四个问题，通常说明题目尚未形成稳定研究问题。

1. 你要优化的系统对象是什么？
2. 你真正改变的控制点是什么？
3. 收益会沿着哪条链路传递到端到端指标？
4. 什么结果会直接否定当前假设？

常见项目方向

核心结论

课程项目优先围绕真实 runtime、真实 benchmark 和可复现系统问题展开。

- 调度：准入、尾时延控制、批处理组织。
- KV：块管理、复用、分层驻留。
- 状态：记忆对象、知识缓存与统一抽象。
- 执行：图执行、异构放置与数据路径优化。
- 工具：benchmark、可观测性与回归验证。

如何缩小问题范围

核心结论

研究生项目中的常见问题在于题目定义过大，从而难以形成可信交付。

- 先找一个最小可运行场景。
- 先做一个最小可验证机制。
- 先找到一个能够证伪自己的实验。

课程项目中的常见失控原因

核心结论

项目中途失控通常源于问题对象、验证路径或最小交付边界未能及时收紧。

- 问题太大，无法在一学期内完成。
- 假设太模糊，无法设计对照实验。
- 实现太重，挤占了分析和验证时间。

中期汇报建议结构

核心结论

中期汇报最重要的是暴露问题定义和验证计划，而不是展示已经做了多少代码。

- 问题背景与瓶颈对象
- 当前假设与拟议机制
- 实验设计与主要风险
- 需要老师和同学帮助判断的部分

从 tutorial 作业到课程项目

核心结论

课程项目不应从零起步，前期 tutorial 作业应被视为项目准备阶段的能力训练。

- Tutorial 1 到 5 用于建立 workload 指标判断、请求路径、调度、KV 与状态管理能力。
- Tutorial 6 到 10 用于建立架构、优化、平台、论文比较与实验方法能力。
- Tutorial 11 到 13 用于把开源系统实践、项目工作坊和最终汇报连接起来。

助教与学生的交付接口

核心结论

如果课程采用 nano-vllm-hust 统一作业包，学生提交与助教汇总可以保持同一结构。

- 学生根据各个 experiment_*/assignment_spring-2026/ 中的模板提交实现、报告与结果。
- 助教可使用 collect_benchmark_results.py 汇总 benchmark 结果。
- 统一目录结构有助于后续过渡到课程项目与最终汇报。

最终项目交付应该包含什么

核心结论

最终项目不是一堆脚本，而应是一份可被他人理解和继续推进的系统工作样本。

- 问题定义与机制设计说明。
- 最小实现或实验脚本。
- 结果分析与失败边界。
- 后续可扩展方向。

讲解：项目工作坊首先要把题目压缩成什么

核心结论

一个课程项目如果不能在一句话里同时说出对象、控制点和指标，通常说明题目还没成形。

- “对象”回答你到底在优化什么，例如调度、KV、状态搬运、观测路径。
- “控制点”回答你准备改哪里，而不是泛泛地说“优化系统”。
- “指标”回答你如何判断成功，以及什么结果会证明你判断错了。

讲解：为什么项目工作坊不鼓励一开始就做大而全

核心结论

课程项目最常见的失败，并不是学生不努力，而是题目边界从第一周起就没有收紧。

- 没有最小场景时，实验无法快速给出正反反馈。
- 没有最小机制时，实现工作会吞掉分析和验证时间。
- 所以工作坊的目标不是激励大家“做更大”，而是帮助大家“做得更真、更可证伪”。

例子：一个好题目和一个坏题目的差别（1）

核心结论

项目工作坊最先要解决的不是“做什么酷”，而是“什么问题有清晰对象、控制点和验证路径”。

坏题目

“优化大模型推理性能”过大，既没有对象边界，也没有明确失败条件。

例子：一个好题目和一个坏题目的差别（2）

核心结论

项目工作坊最先要解决的不是“做什么酷”，而是“什么问题有清晰对象、控制点和验证路径”。

好题目

“在长短请求混部场景下，验证某种 admission 策略能否改善短请求 TTFT 且不损伤 goodput”。

例子：一个好题目和一个坏题目的差别（3）

核心结论

项目工作坊最先要解决的不是“做什么酷”，而是“什么问题有清晰对象、控制点和验证路径”。

课程结论

题目一旦能被写成“对象 + 控制点 + 指标 + 失败条件”，项目才真正开始成立。

例子：为什么中期汇报最该暴露风险而不是展示进度（1）

核心结论

中期汇报的价值在于及时暴露错误方向，而不是把已经投入的工程量包装成确定性成果。

好汇报

说明当前假设最危险的地方、最可能失败的实验和最需要外界判断的路径。

例子：为什么中期汇报最该暴露风险而不是展示进度（2）

核心结论

中期汇报的价值在于及时暴露错误方向，而不是把已经投入的工程量包装成确定性成果。

坏汇报

只列做了多少代码、跑了多少图，却回避核心假设是否站得住。

例子：为什么中期汇报最该暴露风险而不是展示进度（3）

核心结论

中期汇报的价值在于及时暴露错误方向，而不是把已经投入的工程量包装成确定性成果。

教学意义

中期不是“半成品展示会”，而是“研究方向纠偏会”。

代码例子：项目提案的一页式模板（1）

核心结论

工作坊里最好强制学生先写一页提案，再决定是否进入较重实现。

模板片段

```
Problem object: ...
Changed control point: ...
Expected impact path: ...
Falsifying result: ...
```

代码例子：项目提案的一页式模板（2）

核心结论

工作坊里最好强制学生先写一页提案，再决定是否进入较重实现。

课堂应追问

如果其中一项写不出来，说明项目还缺什么？

代码例子：项目提案的一页式模板（3）

核心结论

工作坊里最好强制学生先写一页提案，再决定是否进入较重实现。

课程结论

一页提案的作用不是形式化，而是逼迫项目从“想法”进入“可检验命题”。

代码例子：最小实验矩阵如何设计（1）

核心结论

课程项目不需要一开始就大规模扫参，更需要一个能快速否定错误方向的实验矩阵。

实验矩阵草图

```
baseline vs method
short-heavy / mixed / long-heavy workload
ttft / p99 / goodput / probe metrics
```

代码例子：最小实验矩阵如何设计（2）

核心结论

课程项目不需要一开始就大规模扫参，更需要一个能快速否定错误方向的实验矩阵。

课堂应追问

为什么先做三种 workload 的小矩阵，往往比直接追求“大而全”更有效？

代码例子：最小实验矩阵如何设计（3）

核心结论

课程项目不需要一开始就大规模扫参，更需要一个能快速否定错误方向的实验矩阵。

结论

研究初期最宝贵的不是更多图，而是更快得到可否定反馈。

例子：项目失控往往从哪一步开始（1）

核心结论

大多数课程项目不是在最后失败，而是在最初没有把边界收紧时就已经埋下失败。

失控点一

先承诺一个“大一统系统”，没有最小场景和最小机制。

例子：项目失控往往从哪一步开始（2）

核心结论

大多数课程项目不是在最后失败，而是在最初没有把边界收紧时就已经埋下失败。

失控点二

先写很多代码，后补问题定义和实验设计。

例子：项目失控往往从哪一步开始（3）

核心结论

大多数课程项目不是在最后失败，而是在最初没有把边界收紧时就已经埋下失败。

失控点三

只追求正结果，没有提前设计失败也有价值的解释路径。

课堂练习：把你的题目压缩到一句话（1）

核心结论

一个能做完的课程项目，往往都能被压缩成一句短而硬的问题句。

练习一

用一句话写出你的项目对象、控制点和主指标，不超过 35 个字。

课堂练习：把你的题目压缩到一句话（2）

核心结论

一个能做完的课程项目，往往都能被压缩成一句短而硬的问题句。

练习二

再用一句话写出最危险的失败条件。

课堂练习：把你的题目压缩到一句话（3）

核心结论

一个能做完的课程项目，往往都能被压缩成一句短而硬的问题句。

练习三

用一句话回答：如果只能保留一个实验，你会做哪个，为什么？

课堂讨论

核心结论

课程项目首先应具备可证伪性，其次才适合继续优化。

- 什么样的项目题目看起来大，但其实不具备验证路径？
- 为什么最小可运行场景对课程项目尤其重要？
- 中期汇报最应该暴露哪些不确定性？

本讲总结

核心结论

工作坊的目标是将题目压缩为可落地、可验证的系统问题。

- 项目成功的关键是问题定义和实验设计。
- 最后一讲将回到课程主线并完成最终汇报。

对应 Tutorial

核心结论

本讲对应 Tutorial 12。建议先独立作答，再翻到下一页查看参考答案。

- 文件：build/tutorials/Tutorial_12_ 课程项目工作坊.pdf
- 动手运行、日志记录与提交要求统一查看 build/experiments/ 和对应 experiment 源目录下的 assignment_spring-2026/。