大模型推理系统与实践

第 12 讲开源系统实践

张书豪

华中科技大学计算机学院

2027 · 研究生课程课件

教学目标

核心结论

本讲将前述概念落实到真实开源 runtime、benchmark 与复现实验流程之中。

- 学会进入真实代码库。
- 学会定位请求路径、KV 路径和指标输出。
- 学会记录最小复现实验。

进入代码库的最小路径

核心结论

进入系统的关键不在于覆盖全部代码，而在于准确定位入口路径。

- 请求入口在哪里。
- 调度循环在哪里。
- KV 分配与回收在哪里。
- benchmark 和结果输出在哪里。

代码导读的方法

核心结论

代码导读的关键在于将核心路径串联起来，而非简单扩大阅读文件数量。

- 先追请求入口。
- 再追调度循环和 batch 组织。
- 最后追 KV 与 metrics 输出点。

最小代码导读任务

核心结论

一次合格的代码导读，应至少能把请求、状态和指标三条路径串起来。

- 从 API 到 scheduler
- 从 scheduler 到 model runner
- 从 model runner 到 KV manager 和 metrics

最小复现实验任务

核心结论

复现实验的关键不在于跑出数字，而在于能解释数字是如何产生的。

- 记录模型、输入、参数和硬件环境。
- 记录 workload 构造方式和 warmup 过程。
- 保留日志、脚本和结果表格。

源码穿插：nano-vllm-hust 的代码分层

核心结论

nano-vllm-hust 现在采用“参考实现 + 学生练习”的双层结构，便于把代码导读、实验和作业包放到同一套代码树里。

- nanovllm/ 保留可运行参考实现，可用于源码导读和行为对照。
- student_exercises/ 提供带 TODO 的练习版本，对应课程实验中的补全任务。
- TEACHING.md 给出了 tutorial 与源码文件之间的映射关系。

源码穿插：一条实践主路径

核心结论

围绕这套代码库，实践路径可以按由浅入深的顺序推进。

- 第一步：从 `llm_engine.py` 跟踪请求进入与单步推进。
- 第二步：在 `scheduler.py` 中分析调度与回收逻辑。
- 第三步：在 `block_manager.py` 中理解 KV 分配、复用与哈希。
- 第四步：结合 `bench.py` 讨论如何形成可复现实验记录。

源码穿插：作业包如何组织

核心结论

‘nano-vllm-hust‘ 已提供面向 experiment 的作业打包脚本，可直接生成标准化作业目录。

- `scripts/teaching/prepare_assignment.py` 用于生成每次 experiment 的作业包。
- 生成结果位于对应 `experiment_X_.../assignment_spring-2027/` 目录中。
- 每个作业包同时包含说明文件、starter 文件与提交模板。

复现实验中的常见遗漏

核心结论

复现失败往往来自配置、默认参数与脚本细节记录不完整。

- 模型版本和推理精度。
- warmup、batch 限制和长度分布。
- benchmark 的默认请求生成逻辑。

实践环节的重要性

核心结论

进入真实系统后，才能观察到研究问题常常位于机制设计与工程边界的连接处。

- 论文中的一个框图，在系统里可能跨越多个模块。
- 实验结果往往依赖脚本、默认参数和日志解释。
- 这也是课程项目最有价值的训练部分。

讲解：开源系统实践课真正要避免的错误路径

核心结论

很多学生进入代码库时最大的问题不是看得太少，而是没有形成稳定阅读顺序。

- 如果一开始只在仓库里到处搜索，很容易收集到零散函数而不是主路径。
- 课程更强调从入口、调度、状态、结果四个问题依次往下走。
- 这样做的目标不是少看代码，而是先建立可复述的系统骨架，再扩展细节。

讲解：为什么 benchmark 也是课程实践的核心材料

核心结论

很多系统理解之所以断裂，就是因为代码导读只读 runtime，不读 workload 和结果脚本。

- benchmark 决定了请求如何被生成、指标如何被统计、图表如何被讲述。
- 因此它实际上连接了代码实现与实验结论，是课程闭环里不可缺的桥梁。
- 真正的开源系统实践，不是把程序跑通，而是同时读懂“怎么跑”和“怎么证明”。

例子：进入一个开源 runtime 的正确姿势是什么（1）

核心结论

开源系统实践课最怕学生一上来就“全仓库全局搜索”，最后只积累了碎片印象。

错误路径

先把目录全翻一遍，却没有确定任何主调用链和控制点。

例子：进入一个开源 runtime 的正确姿势是什么（2）

核心结论

开源系统实践课最怕学生一上来就“全仓库全局搜索”，最后只积累了碎片印象。

正确路径

先找入口，再找调度循环，再找状态对象，最后找指标与脚本出口。

例子：进入一个开源 runtime 的正确姿势是什么（3）

核心结论

开源系统实践课最怕学生一上来就“全仓库全局搜索”，最后只积累了碎片印象。

课程结论

代码实践的第一原则不是多看，而是先抓住最短的真实路径。

例子：为什么 benchmark 脚本也必须纳入源码导读（1）

核心结论

系统知识不只在 runtime 本体里，benchmark 脚本经常在定义 workload、指标和报告方式。

runtime 负责执行

它告诉你请求如何被调度、状态如何被组织、结果如何被产生。

例子：为什么 benchmark 脚本也必须纳入源码导读（2）

核心结论

系统知识不只在 runtime 本体里，benchmark 脚本经常在定义 workload、指标和报告方式。

benchmark 负责解释

它告诉你输入是什么、统计口径是什么、图表为什么长成这样。

例子：为什么 benchmark 脚本也必须纳入源码导读（3）

核心结论

系统知识不只在 runtime 本体里，benchmark 脚本经常在定义 workload、指标和报告方式。

教学意义

不读 benchmark，就很容易误把“脚本默认值”当成“系统本身结论”。

代码例子：最小代码导读清单（1）

核心结论

开源系统实践课适合给学生一份很短但很硬的导读清单。

导读顺序

1. `LLMEngine.generate()`
2. `Scheduler.schedule()`
3. `BlockManager.allocate()`
4. `bench.py / result writer`

代码例子：最小代码导读清单（2）

核心结论

开源系统实践课适合给学生一份很短但很硬的导读清单。

课堂应追问

为什么这四步足以构成一个最小系统理解闭环？

代码例子：最小代码导读清单（3）

核心结论

开源系统实践课适合给学生一份很短但很硬的导读清单。

结论

好的课堂实践不会让学生读完所有代码，而是让他们先得到一条可复述的主路径。

代码例子：一次最小复现实验记录应该写什么（1）

核心结论

“我跑通了”不构成复现；复现必须留下别人可重复的上下文。

记录模板

```
model=..., precision=..., max_model_len=...
workload=..., warmup=..., num_requests=...
command=..., logs=..., result_table=...
```

代码例子：一次最小复现实验记录应该写什么（2）

核心结论

“我跑通了”不构成复现；复现必须留下别人可重复的上下文。

课堂应追问

哪些字段缺失时，哪怕数字是对的，也不能算可复现？

代码例子：一次最小复现实验记录应该写什么（3）

核心结论

“我跑通了”不构成复现；复现必须留下别人可重复的上下文。

课程结论

复现实验文档是工程记录，也是研究证据的一部分。

例子：为什么学生最容易在默认参数上踩坑（1）

核心结论

很多所谓“复现失败”并不是机制没实现，而是默认值、环境与隐藏脚本分支没有被看见。

常见遗漏

温启动、随机种子、batch 上限、长度分布和精度模式。

例子：为什么学生最容易在默认参数上踩坑（2）

核心结论

很多所谓“复现失败”并不是机制没实现，而是默认值、环境与隐藏脚本分支没有被看见。

常见误判

看到不同数字就直接怀疑代码逻辑，而不是先检查实验边界是否一致。

例子：为什么学生最容易在默认参数上踩坑（3）

核心结论

很多所谓“复现失败”并不是机制没实现，而是默认值、环境与隐藏脚本分支没有被看见。

教学重点

进入真实开源系统后，最需要养成的是“先核对边界，再解释数字”的习惯。

课堂练习：给自己设计一条 30 分钟导读路线（1）

核心结论

实践课的目标不是一口吃掉系统，而是能在短时间内稳定定位关键对象。

练习一

规定 30 分钟时间，写出你会先打开的 4 个文件和顺序。

课堂练习：给自己设计一条 30 分钟导读路线（2）

核心结论

实践课的目标不是一口吃掉系统，而是能在短时间内稳定定位关键对象。

练习二

说明每个文件分别是为了解决“入口、控制、状态、证据”中的哪一个问题。

课堂练习：给自己设计一条 30 分钟导读路线（3）

核心结论

实践课的目标不是一口吃掉系统，而是能在短时间内稳定定位关键对象。

练习三

用一句话回答：为什么 benchmark 脚本必须算作系统知识的一部分？

课堂讨论

核心结论

实践环节的目标在于建立可解释、可修改、可验证的系统理解。

- 什么是进入一个 runtime 的正确起点？
- 为什么 benchmark 脚本本身就是系统知识的一部分？
- 复现实验最容易遗漏哪些导致结论失真的细节？

本讲总结

核心结论

通过代码与实验进入真实系统，是将课程知识转化为研究能力的关键步骤。

- 项目后续将以这一实践基础继续推进。
- 下一讲将进入项目问题定义和工作坊环节。

本讲参考资料

- **nano-vllm-hust**（课程教学代码） 原文链接
- **vLLM-HUST**（HUST 维护的 vLLM fork） 原文链接
- **上游 vLLM**（开源项目） 原文链接

课件中的论文数字均指作者在特定模型、硬件和工作负载下报告的结果，不代表无条件性能。

对应 Tutorial

核心结论

本讲对应 Tutorial 11。先独立作答，再使用对应 Tutorial 的教师讨论要点核对。

- 文件：`build/tutorials/Tutorial_11_开源系统实践.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-2027/`。