大模型推理基础设施

第 9 讲异构平台与国产算力适配

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

本讲讨论平台差异如何系统性改变瓶颈位置、最优策略与验证方法。

- 理解能运行、能优化、能验证三个层次。
- 理解国产异构平台的系统性差异。
- 讨论跨平台收益如何公平验证。

为什么平台差异是系统差异

核心结论

不同平台在图执行、算子栈、互联和调试环境上的差异，会直接改写系统设计空间。

- 软件栈成熟度不同。
- 互联协议与带宽比例不同。
- host-device 数据路径代价不同。

平台差异会改写哪些瓶颈

核心结论

同一机制在不同平台上可能卡在完全不同的地方，因此不能直接平移 GPU 经验。

- 有的平台先卡在图执行与编译稳定性。
- 有的平台先卡在通信与互联约束。
- 有的平台先卡在调试工具和观测能力不足。

适配的三个层次

核心结论

平台适配至少分为三个层次，而真正的研究通常从第二层开始。

1. 能运行：基本功能打通
2. 能优化：根据平台特性重构路径
3. 能验证：建立公平可复现的基线和对照

从可运行到可优化的差异

核心结论

适配工作常停留在基本可运行阶段，而研究价值主要体现在可优化与可验证两个层次。

- 跑通只说明接口存在。
- 稳定收益需要重新建模瓶颈。
- 可验证结论需要统一实验设计与对照基线。

常见误区

核心结论

把问题只定义成“移植工程”，往往会掩盖真正有研究价值的系统差异。

- 只报告局部跑通，不说明收益边界。
- 直接沿用 GPU 经验，不重新验证假设。
- 用不一致 workload 比较不同平台。

跨平台实验如何更可信

核心结论

跨平台比较最关键的是控制变量，而不是堆更多数字。

- 统一模型、输入分布和服务目标。
- 显式说明软件栈、精度和并行配置。
- 同时报告端到端指标和关键探针指标。

跨平台比较中的常见问题

核心结论

跨平台比较中的主要问题通常来自对照不公平、工作负载不一致或边界说明不足。

- 输入分布不同会直接破坏比较意义。
- 精度和默认优化不一致会导致结论失真。
- 没有端到端指标时很难说明真实服务价值。

讲解：平台适配课的核心不是移植，而是重建假设

核心结论

GPU 上形成的经验并不能直接平移到国产 NPU 或其他异构平台，因为假设条件一起变了。

- 图执行、编译路径、通信比例、调试工具和默认优化都可能不同。
- 因此适配不是“接口替换”，而是重新确认瓶颈在哪、哪些机制仍然成立。
- 这也是为什么平台适配有研究价值，它迫使我们暴露原本被默认隐藏的系统前提。

讲解：第九讲应建立的验证边界

核心结论

跨平台比较最容易失真，所以课程里必须先讲什么叫“可比”。

- 模型、精度、workload、warmup 和并行配置如果不统一，数字很难解释。
- “能跑起来”只对应功能层，真正可研究的是“能优化”和“能验证”两层。
- 因此平台课的重点不是比谁更快，而是学会设计公平、可追责的比较实验。

例子：为什么同一机制在 GPU 和国产 NPU 上经常表现不同（1）

核心结论

平台适配课的出发点不是“换了芯片”，而是执行语义、工具链和瓶颈位置一起变了。

GPU 路径

开发者往往更熟悉 CUDA、Nsight 和主流 kernel 栈，性能问题定位更直接。

例子：为什么同一机制在 GPU 和国产 NPU 上经常表现不同（2）

核心结论

平台适配课的出发点不是“换了芯片”，而是执行语义、工具链和瓶颈位置一起变了。

NPU 路径

图执行、编译栈、通信路径和 profiler 语义可能完全不同，原有直觉经常失效。

例子：为什么同一机制在 GPU 和国产 NPU 上经常表现不同（3）

核心结论

平台适配课的出发点不是“换了芯片”，而是执行语义、工具链和瓶颈位置一起变了。

课程结论

平台迁移不是简单重编译，而是一次系统假设重审。

例子：能跑、能优化、能验证为什么要分成三层（1）

核心结论

很多适配工作停留在“能跑起来”，但课程必须让学生看到这和“能做研究”之间还有距离。

能跑

模型能返回结果，但不代表路径正确、效率合理或结果稳定。

例子：能跑、能优化、能验证为什么要分成三层（2）

核心结论

很多适配工作停留在“能跑起来”，但课程必须让学生看到这和“能做研究”之间还有距离。

能优化

说明热点路径、编译行为和资源瓶颈已经可观察、可改动。

例子：能跑、能优化、能验证为什么要分成三层（3）

核心结论

很多适配工作停留在“能跑起来”，但课程必须让学生看到这和“能做研究”之间还有距离。

能验证

说明你可以用公平实验解释收益，而不是把所有差异都推给平台黑盒。

代码例子：平台适配首先会动到哪些运行时入口（1）

核心结论

平台适配往往不是直接改模型定义，而是先从 runner、worker、编译与内存路径切入。

典型入口

```
model_runner.py
worker / executor backend
graph capture / compile policy
```

代码例子：平台适配首先会动到哪些运行时入口（2）

核心结论

平台适配往往不是直接改模型定义，而是先从 runner、worker、编译与内存路径切入。

课堂应追问

这些位置分别更接近功能兼容、性能优化，还是验证与观测能力？

代码例子：平台适配首先会动到哪些运行时入口（3）

核心结论

平台适配往往不是直接改模型定义，而是先从 runner、worker、编译与内存路径切入。

系统含义

真正的适配路径通常要先建立“哪个层面在出问题”的分层诊断。

代码例子：最小平台差异观察日志（1）

核心结论

平台适配课里的代码页，重点是教学生如何建立平台差异的第一手证据。

最小日志样式

```
log(platform, batch_size, prompt_len, stage, latency_ms)
log(graph_mode, compile_hit, memory_allocated)
```

代码例子：最小平台差异观察日志（2）

核心结论

平台适配课里的代码页，重点是教学生如何建立平台差异的第一手证据。

课堂应追问

哪些量说明“路径不同”，哪些量说明“只是配置不同”？

代码例子：最小平台差异观察日志（3）

核心结论

平台适配课里的代码页，重点是教学生如何建立平台差异的第一手证据。

课程结论

没有分层日志，平台差异就会退化成模糊叙述，无法进入系统研究语境。

例子：公平实验为什么在异构平台上更难（1）

核心结论

平台比较最容易犯的错误，是把环境差异、软件成熟度差异和机制差异混在一起。

不公平一

比较时使用了不同精度、不同 batch 上限或不同 warmup 策略。

例子：公平实验为什么在异构平台上更难（2）

核心结论

平台比较最容易犯的错误，是把环境差异、软件成熟度差异和机制差异混在一起。

不公平二

一方用了成熟 profiler 和手工优化，另一方仍停留在默认配置。

例子：公平实验为什么在异构平台上更难（3）

核心结论

平台比较最容易犯的错误，是把环境差异、软件成熟度差异和机制差异混在一起。

课程要求

异构平台实验必须先交代可比边界，再谈数字输赢。

例子：迁移一条 decode 热路径时会遇到什么（1）

核心结论

decode 热路径对平台差异特别敏感，因为它既要求低抖动，又持续依赖状态与调度稳定性。

算子层

kernel、算子融合和内存布局可能都需要重新调优。

例子：迁移一条 decode 热路径时会遇到什么（2）

核心结论

decode 热路径对平台差异特别敏感，因为它既要求低抖动，又持续依赖状态与调度稳定性。

运行时层

graph capture、同步语义和调度节拍可能与原平台不同。

例子：迁移一条 decode 热路径时会遇到什么（3）

核心结论

decode 热路径对平台差异特别敏感，因为它既要求低抖动，又持续依赖状态与调度稳定性。

研究层

如果不能解释是哪个层面出问题，适配工作就很难形成可发表的系统结论。

课堂练习：为跨平台实验写一页计划（1）

核心结论

平台适配课最适合训练的，是写出一份短而硬的验证计划。

练习一

指定两个平台、一个共同 workload 和三项必须控制的变量。

课堂练习：为跨平台实验写一页计划（2）

核心结论

平台适配课最适合训练的，是写出一份短而硬的验证计划。

练习二

指出你会优先观测的两个局部探针，以及它们为什么能帮助解释性能差异。

课堂练习：为跨平台实验写一页计划（3）

核心结论

平台适配课最适合训练的，是写出一份短而硬的验证计划。

练习三

用一句话回答：为什么“能跑起来”还不算完成平台适配？

课堂讨论

核心结论

跨平台研究需要明确平台差异究竟改写了哪些系统假设。

- 为什么“在平台 A 上有效”不自动说明方法具有普适性？
- 怎样设计一个最小跨平台验证计划？
- 什么结论最容易因为实验不公平而失真？

本讲总结

核心结论

平台适配本身构成系统研究的重要组成部分。

- 平台差异会重写瓶颈与最优策略。
- 只有能验证，平台适配结论才有说服力。

对应 Tutorial

核心结论

本讲对应 Tutorial 8。建议先独立作答，再翻到下一页查看参考答案。

- 文件：build/tutorials/Tutorial_08_ 异构平台适配.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/。