实验 1：最小运行与代码地图

1. 本节定位

本实验是课程的第一份实验单。目标不是立即优化系统，而是建立后续全部实验都会反复复用的三个锚点：

1. 一次最小推理请求能够稳定跑通。
2. 请求进入、执行、返回的主路径能够被学生自己复述。
3. 代码地图能够作为后续生命周期、调度、状态管理实验的共同起点。

本节与第 2 讲 “工作负载与评价指标” 及对应 tutorial 配套使用。lecture 和 tutorial 负责讲清 workload 与指标，本实验负责把这些讨论落到真实代码入口和最小运行证据上。

2. 核心问题

一次请求从哪里进入系统，经过哪些关键对象，又在哪里返回结果？

本节只回答结构问题，不要求解释为什么性能更好，也不要求提出优化方案。

3. 学习目标

完成本节后，你应能够：

- 说清一次最小请求从哪里进入系统。
- 指出哪个对象首先真正持有请求状态。
- 区分请求入口、调度循环、模型执行、结果返回四类职责。
- 用最小日志而不是主观猜测支撑自己的代码路径判断。

4. 开始前

4.1 先修要求

- 已了解 Transformer 推理的基本过程。
- 知道 prompt、token、max tokens、TTFT 等基本术语。
- 具备最基本的 Python 运行能力。

4.2 本节材料

- 理论配套：build/tutorials/Tutorial_01_工作负载与评价指标.pdf
- 教学代码：src/code/nano-vllm-hust/
- 参考入口：src/code/nano-vllm-hust/example.py
- 封装入口：src/code/nano-vllm-hust/nanovllm/llm.py
- 主控制路径：src/code/nano-vllm-hust/nanovllm/engine/llm_engine.py

- 作业包：src/experiments/nano-vLLM实验课/experiment_1_最小运行与代码地图/assignment_spring-2027/

4.3 环境检查

开始前请确认以下项目：

- 能访问课程仓库。
- 运行环境与模型路径可用。
- `example.py` 至少存在一条可直接执行的最小命令。
- 你知道输出保存在哪里，便于后续摘录日志。

5. 提交内容

本节提交物必须包含四项：

1. 一张代码地图表。
2. 一条 6-10 步的最短调用链。
3. 一段最小日志摘录。
4. 一段 300-500 字说明，回答“当前看起来第一个控制点在哪里”。

6. 时间安排建议

- 10 分钟：确认本节问题与运行入口。
- 15 分钟：跑通最小请求。
- 25 分钟：建立代码地图。
- 20 分钟：追踪最短调用链并加最小日志。
- 15 分钟：完成简短说明与课堂讨论。

7. 实验任务

任务 1：完成一次最小运行

完成以下动作：

1. 打开 src/code/nano-vllm-hust/example.py。
2. 检查模型路径、运行环境与依赖。
3. 运行一次单请求或双请求的最小生成示例。
4. 记录输入 prompt、输出长度和是否成功返回。

请填写以下记录：

运行命令：
模型路径：
输入 `prompt`：
输出 `token` 数：

是否成功返回：
异常信息（如有）：

完成标准：给出一次可复现的成功运行记录，而不是只保留终端末尾截图。

任务 2：整理最小代码地图

不要泛读整个仓库。只定位以下五类对象：

1. 请求入口。
2. 请求对象或状态对象。
3. 调度器或主循环。
4. 模型执行器。
5. 结果返回位置。

请填写下表：

系统职责	文件 / 类 / 函数	你的判断依据
请求入口		
请求状态		
调度循环		
模型执行		
输出返回		

填写要求：每一行都必须写明“判断依据”，不能只列文件名。

任务 3：追踪最短调用链

从 `example.py` 出发，写出一条 6-10 步的最短调用链。函数名允许不完全一致，但职责必须闭合。

建议格式：

```
example.main -> LLM.generate -> ... -> response
```

填写要求：调用链必须能解释“请求如何进入”和“结果如何返回”，不能只罗列目录。

任务 4：加入最小日志记录

只允许在三个位置加日志：

- 请求进入点。
- 执行开始点。
- 结果返回点。

日志建议至少包含：

- `request_id`
- `prompt_len`
- `max_tokens` 或 `output_len`

- `timestamp`

目标是证明调用链，而不是把运行时输出打满。

任务 5：回答核心问题

基于前面四个任务，回答以下问题：

1. 系统里第一个真正拥有请求的对象是谁。
2. `LLM` 更像控制点，还是对更深层对象的封装入口。
3. 输出返回之前，最后一次关键状态更新大概发生在哪里。

要求：你的回答必须引用前面的代码地图或日志证据。

8. 证据要求

以下内容不能视为充分证据：

- 只给目录截图，不写职责判断。
- 只说 “请求进入 engine”，但说不清进入前后对象如何变化。
- 一上来就浏览大量文件，却没有收束出主路径。
- 日志插桩过多，最后无法提取关键路径。

本节更看重 “判断是否可复核”，而不是 “看过多少文件”。

9. 提交核对

提交前请逐项检查：

- 已附代码地图表。
- 已附最短调用链。
- 已附最小日志摘录。
- 已附 300-500 字结构化说明。
- 说明中明确回答了 “第一个控制点在哪里”。

10. 评分关注点

- 是否真的定位了五类核心职责。
- 最短调用链是否闭合、自洽。
- 日志是否能支撑你的路径判断。
- 说明文字是否回到 “控制点”，而不是停在目录描述。

11. 与后续实验的衔接

本节交付物会直接复用到后续实验：

- 实验二：请求生命周期时序图。
- 实验三：调度控制点定位。
- 实验四：状态对象定位。
- 实验五：实验记录结构。

如果本节的代码地图没有建立好，后续实验会很容易退化成盲查代码。

12. 提交模板

请直接填写或据此整理你的提交材料。

12.1 基本信息

姓名 / 组员:
学号:
实验日期:
运行环境:
模型路径:

12.2 最小运行记录

运行命令:
输入 prompt:
输出 token 数:
是否成功返回:
异常信息 (如有):

12.3 代码地图

系统职责	文件 / 类 / 函数	判断依据
请求入口		
请求状态		
调度循环		
模型执行		
输出返回		

12.4 最短调用链

```
example.main ->
```

12.5 最小日志摘录

```
[timestamp] ...
```

12.6 控制点说明

请用 300-500 字回答：当前看起来第一个控制点在哪里？你的证据是什么？