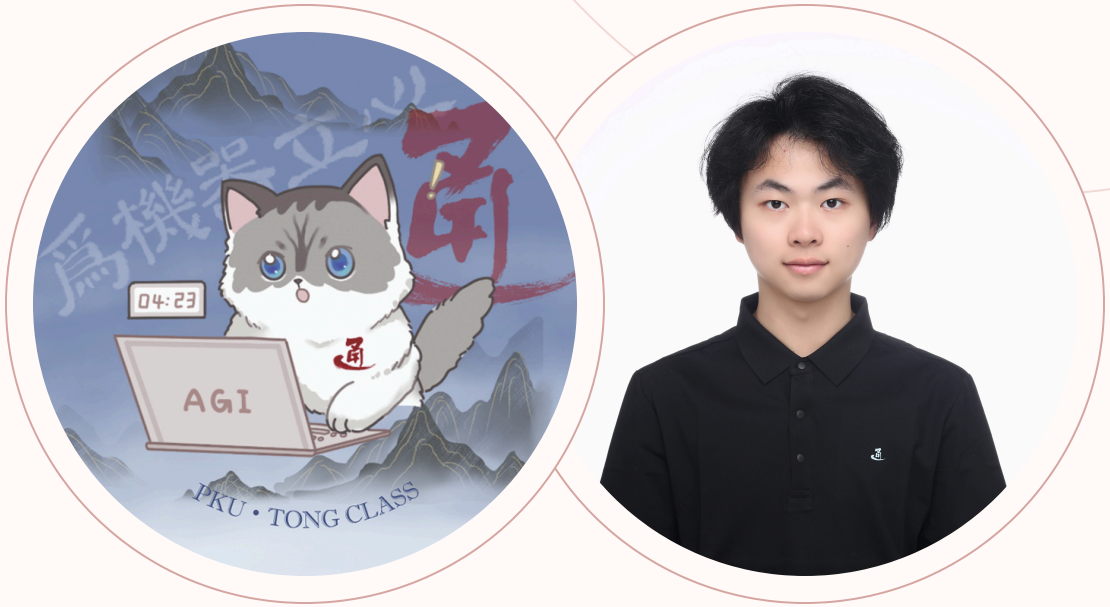
北京大学人工智能研究院 INSTITUTE FOR ARTIFICIAL INTELLIGENCE, PEKING UNIVERSITY
北京大学元培学院 YUANPEI COLLEGE PEKING UNIVERSITY

ToNG: Tutorials on Necessary Groundwork

通班人工智能科研先导课 · 第一讲

计算机的结构与硬件

Computer Architecture and Hardware



主讲单位
北京大学通用人工智能实验班

主讲人
张云琦

— 从一个日常动作开始

双击运行程序后，内部发生了什么？

程序不会直接“待在 CPU 里运行”。它要经过读取、装入和执行，结果再送往输出设备或写回存储。

01
输入
键盘、鼠标和文件把信息送入系统。

02
处理
CPU 组织流程，GPU 执行并行计算。

03
存储
RAM 暂存数据，SSD 长期保存。

04
输出
结果被显示、播放或写回文件。

输入 → 处理 ↔ 存储 → 输出

— 整体图景

计算机是一套协作系统

- **CPU** 执行指令、调度流程
- **RAM** 暂存当前使用的数据
- **SSD / HDD** 长期保存程序和文件
- **GPU** 执行适合并行的计算
- **主板、电源、散热与 I/O** 提供连接与运行条件

任何环节成为瓶颈，都可能拖慢整个系统。

程序如何运行

01

SECTION

— 程序如何运行

CPU、内存与存储

文件存在哪里，程序在哪里工作，指令由谁执行。

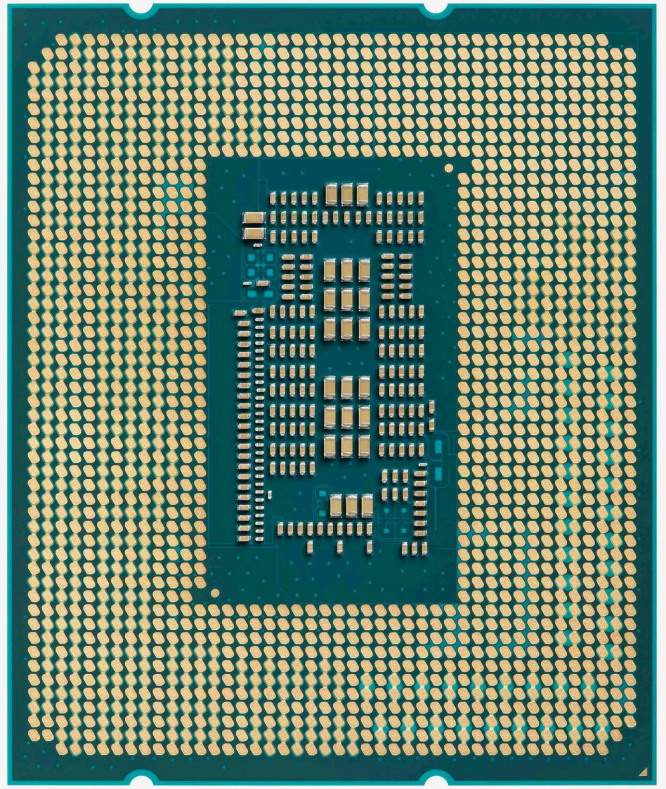
— 中央处理器

CPU 不断重复“取指—译码—执行”

CPU 擅长低延迟的通用计算、复杂控制和条件分支。

- 算术与逻辑运算
- 数据搬运
- 条件判断与跳转
- 组织程序流程

主频和核心数会影响性能，但必须结合架构与具体任务比较。



参数更大，不等于程序一定更快

主频

每个核心工作的节奏

GHz 描述时钟速率，但实际性能还受架构、缓存、功耗和工作负载影响。

核心数

可以并行推进的任务数量

只有程序能将工作并行拆分，更多核心才能充分发挥作用。

— 随机存取存储器

RAM 是程序运行时的工作空间

- 适合频繁读写
- 断电后数据消失
- 容量通常小于长期存储

16 GB / 32 GB 通常指 RAM；512 GB / 1 TB 通常指 SSD。

RAM 不足时，系统可能频繁换页并明显变慢，程序也可能因内存耗尽而终止。

运行程序与保存文件，分工不同

RAM：桌面

运行时工作区

- 取用方便，读写快
- 空间有限
- 断电后清空
- 保存当前处理的内容

SSD / HDD：书架

长期保存区

- 容量更大
- 长期保存
- 断电后保留
- 保存系统、软件、数据和结果

— 长期存储

HDD、SATA SSD 与 NVMe SSD：结构与接口不同

- **HDD**：机械盘片，容量大、成本低，随机访问较慢
- **SATA SSD**：使用闪存，无机械部件，响应更快
- **NVMe SSD**：通常通过 PCIe 通信，吞吐更高

在 AI 工作流中，大量小文件读取、解码和预处理也可能让 GPU 等待。

— 存储分层

越靠近 CPU，通常越快、越小、越贵

寄存器 / CPU 缓存　最快 · KB–MB

RAM　较快 · GB 级

SSD / HDD　持久 · GB–TB 级

外置 / 网络 / 云存储　更大 · 更远

单一设备无法兼顾低延迟、大容量和低成本，因此系统采用分层存储。

— 完整运行链路

一个 Python 程序如何运行

读取

SSD

持久保存解释器、脚本和数据。

装入

RAM

提供程序运行所需的工作空间。

执行

CPU

执行指令、组织流程并调度任务。

写回

SSD

长期保存需要保留的结果。

SSD → RAM → CPU → RAM → SSD

遇到适合大规模并行的计算时，CPU 会组织任务并交给 GPU。

02

SECTION

— 支撑硬件

主板、电源、散热与 I/O

计算芯片还需要连接、供电和散热，并与用户及网络交换信息。

支撑硬件

— 连接平台

主板决定部件如何接入系统

主板提供 CPU 与内存插槽、PCIe 通道、存储接口和外部 I/O。

- 通常不直接承担主要计算
- 负责让部件互相通信
- 限定可安装的 CPU、RAM 和扩展设备

判断兼容性时，要同时检查插槽、接口标准、尺寸和供电。

稳定运行不只取决于算力

电源

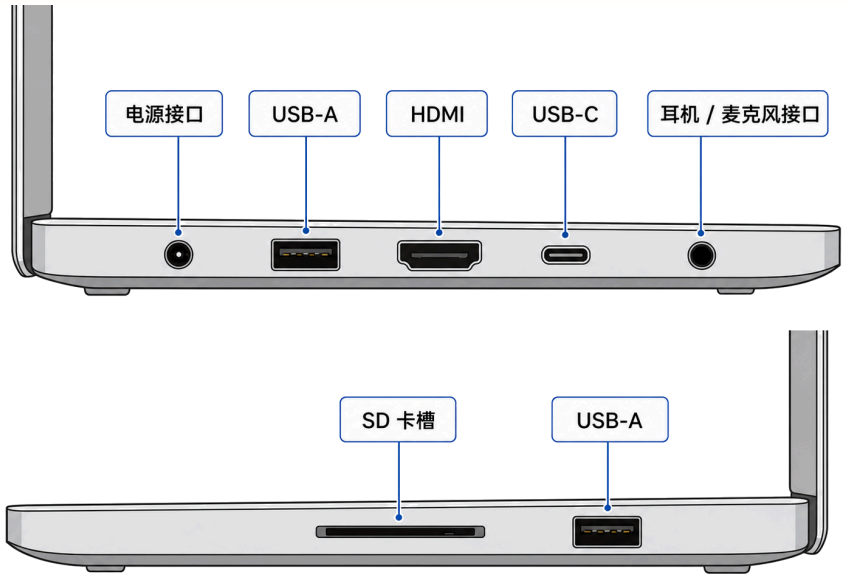
提供稳定电力

台式机 PSU 将交流电转换为各部件所需的直流电；功率、接口和质量都会影响稳定性。

散热

把热量带走

散热器、风扇和机箱风道维持芯片温度，避免过热降频和异常。



— 输入与输出

接口同名，能力未必相同

- USB-A：键盘、鼠标、U 盘等
- USB-C：数据传输；部分设备还支持供电或视频
- HDMI / DisplayPort：显示输出
- RJ45 / Wi-Fi：网络连接
- 3.5 mm：音频输入输出

USB-C 尤其需要确认具体协议，不能只看接口外形。

03

SECTION

— AI 计算

GPU、显存与 CUDA

GPU 不是“更强的 CPU”，而是面向不同任务优化的处理器。

— 并行计算

GPU 擅长并行处理大量相似任务

CPU 擅长复杂控制与低延迟响应；GPU 擅长对大量数据执行同类运算。

深度学习中的矩阵和张量运算常能拆成大量结构相似的小任务，适合 GPU 并行执行。

能否加速，取决于任务能否有效并行，以及数据传输是否成为瓶颈。

CPU 与 GPU 协作，而非相互替代

CPU

低延迟的通用控制

- 少量功能完整的核心
- 复杂逻辑和条件分支
- 操作系统与程序调度
- 数据读取和预处理

GPU

高吞吐的并行计算

- 大量并行计算单元
- 结构相似的批量运算
- 图形渲染
- 矩阵和张量计算

— 显存约束

VRAM 决定 GPU 能容纳多少工作

显存中通常需要容纳：

- 模型参数
- 输入批次
- 中间激活值
- 梯度、优化器状态和通信缓冲区

显存不足时，可以减小 batch size、采用节省显存的算法、进行卸载或拆分到多张 GPU，但通常会牺牲速度或增加复杂度。

— 硬件与软件的桥梁

CUDA 连接 AI 框架与 NVIDIA GPU

上层

AI 框架

组织模型、数据和算子调用。

桥梁

CUDA 软件栈

提供编程模型、工具和计算库。

硬件

GPU

最终执行适合并行的计算。

AI 框架 → CUDA 软件栈 → GPU

CUDA 不是芯片，也不等同于显卡驱动；硬件、驱动、运行库和框架需要相互兼容。

— 规模扩展

多 GPU 扩展资源，也增加通信开销

多卡系统通常需要更强的：

- 电源与散热
- 系统内存容量
- PCIe / 高速互联
- 持续高负载能力

多 GPU 不会自动带来线性加速。 程序还要划分数据或模型，并同步各设备的计算。

应用与判断

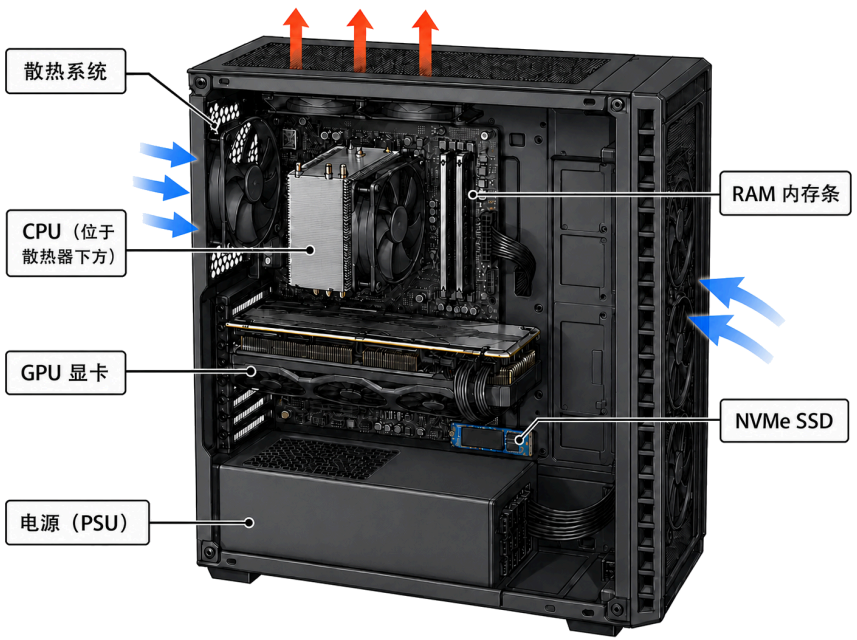
04

SECTION

— 应用与判断

认识自己的电脑

把术语转化为四项能力：认部件、读配置、看监控、描述问题。



— 辨认实物

按位置快速识别台式机部件

- 最大的电路板：主板
- CPU 附近的长条插槽：RAM
- 横向安装的大型扩展卡：GPU
- 主板上的 M.2 小板：NVMe SSD
- 机箱角落的金属盒：PSU
- 鳍片与风扇：散热系统

内部检查或插拔前，先断电并确认安全规范。

— 阅读配置单

先识别类别，再比较具体型号

```
i7-13700K / RTX 4070 12 GB / 32 GB DDR5 / 1 TB NVMe SSD
```

CPU	GPU / VRAM	RAM	存储
i7-13700K	RTX 4070 / 12 GB	32 GB DDR5	1 TB NVMe SSD

不必记住所有型号；先判断硬件类别、容量和用途。

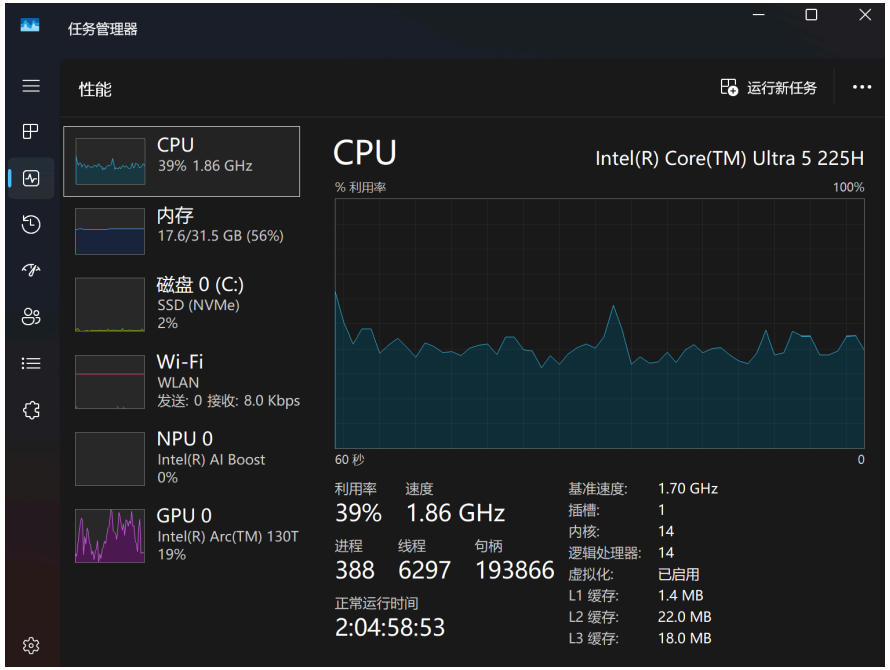
— 性能诊断

高占用本身不一定是故障

任务管理器或系统监控工具可观察：

- CPU 利用率
- RAM 使用量
- 磁盘读写活动
- GPU 利用率、VRAM 与温度

诊断时要结合症状、持续时间和工作负载；满载也可能只是硬件正在正常工作。



观察 · 记录 · 解释

5 分钟

Q01

现场实践：给电脑做一次“硬件体检”

记录 CPU 型号、RAM 容量、存储类型和容量、GPU 型号与显存。

启动一个常用程序，观察 CPU、RAM、磁盘与 GPU 指标的变化。

用一句话解释：哪个指标变化最明显？为什么？

求助时说明操作系统、具体症状、关键占用率和已尝试步骤；发送截图前检查隐私信息。

画图 · 对比 · 解释

Q02

课后练习：用数据流检验理解

- 画出 **SSD → RAM → CPU → RAM → SSD** 的基本程序运行链路。
- 在图中补充 **GPU 与 VRAM**，标明谁准备数据、在哪里计算、结果如何保存。
- 解释 RAM 与 SSD 为什么不能互相替代。
- 说明 AI 训练中 RAM 和 VRAM 各自可能怎样成为瓶颈。

检查标准：能说明每条箭头“传递什么、为何传递、由谁继续处理”。

估算 · 调研 · 延伸

Q03

挑战练习：从容量估算走向系统判断

一个任务需要容纳 **200 GB** 模型状态，每块 GPU 有 **24 GB VRAM**。

暂不考虑其他开销，且模型可均匀分片，至少需要多少块 GPU?

为什么实际训练还要为激活值、梯度、通信缓冲区与框架开销预留显存?

搜索 **Von Neumann Architecture**，它与本讲的数据流模型有什么联系?

提示：容量估算只是下限；设备增多后，还要考虑通信和同步成本。

本讲要点

把部件放回完整的数据链路

SSD	↔	RAM	↔	CPU	↔	VRAM	↔	GPU
读取与保存		程序工作区		执行与调度		GPU 工作区		并行计算

看系统

性能来自整条链路，瓶颈可能出现在任何环节。

看任务

CPU 与 GPU 分工不同，参数必须结合负载理解。

看条件

连接、供电、散热、I/O 和软件栈共同影响稳定运行。

下一讲：多个程序同时运行时，**操作系统如何统一管理硬件资源?**