STAT243 Lecture 6.1 Some scenarios for parallelization

1 并行计算概述（Overview of Parallelization）

> **Logic**
>
> 参考资料:
>
> - Tutorial on parallel processing using Python's Dask and R's future packages
> - Tutorial on parallelization in various languages, including use of PyTorch and JAX in Python

---

1.1 Context: 并行计算与操作系统

- **Linux 环境为主**:
  - 绝大多数高性能并行计算系统运行于 Linux 或其变体。
  - 这里主要讨论单机并行化（single-machine parallelization），在 Mac 与 Windows 上也均可运行，但底层机制有所不同。
- **提升计算速度的几种途径:**
  1. **改进算法（better algorithms）**
     - 在科学计算中非常关键，但本单元不聚焦此方向。
  2. **更高效的算法实现（efficient implementation）**
     - 如优化代码向量化、减少冗余运算（Unit 5 讨论）。
  3. **更快的硬件（faster computers）**
     - CPU 发展趋缓（摩尔定律放缓），但 GPU 技术发展迅猛。
  4. **更多计算资源（more computers/processors）**
     - 利用更多 CPU、GPU 线程或计算节点进行任务分解并行执行（本单元重点）。

1.2 并行化的常见应用场景

- **模型拟合任务**
  - 对数据拟合单一模型（如随机森林、回归模型）。
  - 对同一数据拟合多种不同模型。
- **交叉验证与集成预测**
  - 在 10 折交叉验证中并行运行多种模型的预测。
  - 使用集成方法（如 SuperLearner、贝叶斯模型平均）对每一折的多模型进行组合。
- **分层分析（Stratified Analysis）**
  - 对大型数据集的不同子组分别进行回归分析。
- **模拟研究（Simulation Studies）**
  - 例如运行 1000 次重复模拟（每次需拟合多个模型）。

1.3 并行化的可行性

- 需要思考以下问题:
  - 能否并行化?
  - 能否在笔记本或单台机器上运行?

- 是否需要多台机器（集群）以加快速度或扩大内存容量?
- 若不同任务 **相互独立、无需通信**，即可并行化执行。
  - 这种场景称为 **“尴尬并行”（Embarrassingly Parallel, EP）** 计算。

> **Logic**
> 之所以叫 "Embarrassingly Parallel", 是因为这种情形 "简单到令人尴尬, 没有什么好研究的"

---

2 Embarrassingly Parallel Problem

2.1 尴尬并行（Embarrassingly Parallel, EP）问题

- **定义**
  - 各任务可独立运行、互不依赖，也无需进程间通信。
  - 可将总任务拆分为多个独立子任务并行执行，最后合并结果。
- **典型统计应用**
  - 模拟实验中大量独立重复（replicates）
  - 自助法（Bootstrapping）
  - 分层分析（Stratified Analyses）
  - 随机森林（Random Forests）
  - 交叉验证（Cross-Validation）
- **特征**
  - 同一段代码作用于不同数据子集。
  - 各进程可能需要独立的随机数流（将在 Simulation 单元讨论）。

2.2 实现 EP 并行计算的条件

- 需要对多个**进程（processes）** 拥有控制权。
- 在共享系统（如集群）中，通常需通过队列/调度系统：
  - 申请一定数量的 CPU；
  - 按多进程方式提交作业。

2.3 理想加速效果：线性加速（Linear Speedup）

- 若一个任务原本需时间 (T)，并行化后使用 (p) 个 CPU：
  - 理论执行时间 ≈ (T / p)（忽略少量通信与管理开销）。
  - 加速比（speedup） ≈ (p)，即所谓 **线性加速（linear speedup）**。
  - 当加速比与 CPU 数量成比例时，说明并行化效率极高。