STAT243 Lecture 6.3 Parallelization Strategies

1 并行化策略 (Parallelization Strategies)

1.1 影响并行化效率的主要因素

在评估一种并行计算方案是否有效时，需要考虑以下因素：

- **内存使用量**：不同进程所占用的总内存量是否超出机器容量
- **通信需求量**：进程之间需要交换多少数据
- **通信延迟 (latency)**：在进程之间传递数据或启动子进程时的延时
- **同步等待**：某些进程是否需要等待其他进程完成后才能继续执行

1.2 并行计算的基本原则与建议

1.2.1 单节点 vs 多节点

- 若能在单个节点上使用共享内存完成计算，通常比在多个节点上运行更快
- 即使多节点拥有更多核心，通信开销仍可能使速度下降
- 如果任务需要处理大量数据或占用高内存，单机拥有足够内存的情况下会比使用 Spark/Hadoop 等框架更快
- 但如果单节点内存不足，则必须采用分布式内存架构

> ⚠ **Remark: 总结**
>
> 单节点内存足够就用单节点, 否则用多节点

1.2.2 选择并行化的层次或维度

- 若存在嵌套循环，一般只在一个层次上进行并行化
- 可以考虑是否已经使用了线程化线性代数 (threaded linear algebra)，以避免多层并行导致过度开销
- 通常情况下，应并行化**外层循环**
- 目标是确保计算任务之间 **负载均衡 (load-balanced)**，同时 **减少通信**

1.2.3 平衡通信开销与核心利用率

- 若任务数量较少且执行时间不均，部分核心会空闲，导致负载不均衡
- 若任务数量过多且每个任务耗时极短，频繁启动与停止任务的开销会降低效率
- 因此需要在任务粒度和通信成本之间找到平衡

1.2.4 静态调度 (Prescheduling) vs 动态调度 (Dynamic Scheduling)

- **静态调度 (Prescheduling)**
  任务在开始时就预先分配给各个进程
  - 适用于任务数量多且耗时相似的情况

    - 优点：减少通信开销
    - 示例：6 个任务与 3 个 worker
        - worker1 → 任务 1 和 4
        - worker2 → 任务 2 和 5
        - worker3 → 任务 3 和 6
- **动态调度 (Dynamic Scheduling)**
  任务按执行进度动态分配
    - 适用于任务数量少或耗时差异较大的情况
    - 优点：提高负载均衡性
    - 示例：
        - 初始时：worker1 任务 1，worker2 任务 2，worker3 任务 3
        - 当某个 worker 率先完成后，再分配下一个任务 (如任务 4)

R 中的任务调度控制:

- R 的部分并行函数允许手动指定是否预分配任务
- **future 包**中的 `future_lapply()` 可通过以下参数控制调度方式
    - `future.scheduling`
    - `future.chunk.size`
- **parallel 包**中的 `mclapply()` 提供参数
    - `mc.preschedule`
      用于指定是否启用预调度