### 4.4.1 并行化模型

正如 4.3.1 小节所介绍，进程是计算机系统中正在运行的应用程序实例。同一个应用程序可同时运行多个实例，一个实例也可包含多个进程，为完成一个任务同时进行运算，从应用程序设计的角度来看，这涉及并行化程序的概念。在现代操作系统中，通常存在进程、线程、协程这 3 种不同的并行化手段。基于进程的并行化程序具有较高的安全性，但由于每个进程拥有独立的虚拟地址空间等一系列独立的资源，维护这些资源需要消耗相对较多的计算时间和内存空间。操作系统研究者设计了更为轻量级的并行化程序支持机制——线程（thread）和协程（coroutine），这两种机制虽然相对复杂，但它们卓越的性能受到了HPC 社区的广泛认可。

要了解并行计算的相关概念，可先了解非常典型的并行处理单元的工作模式。并行计算是指同时使用多种计算资源来快速解决计算问题，可分为时间上的并行和空间上的并行。时间上的并行是指流水线技术，而空间上的并行则是指用多个处理器并发地执行计算。

- 串行计算：就像一个人炒菜，按步骤一道又一道地完成。
- 并行计算：就像几个人分工，同时炒菜，一个人洗菜，一个人备菜，一个人掌勺…各司其职，相互配合。

计算机的 CPU 核就像多个“厨师”，它们可同时处理不同任务的不同部分。一个内核处理任务 A，一个内核处理任务 B，于是任务的总完成时间会比通过串行计算实现时少很多。总之，并行计算利用计算机多个处理单元同时工作的优势，通过划分任务并发执行各个部分的工作，从而提高整体效率。这在多核时代尤为重要，它让计算机的运算能力更好地发挥出来，如图4.8所示。

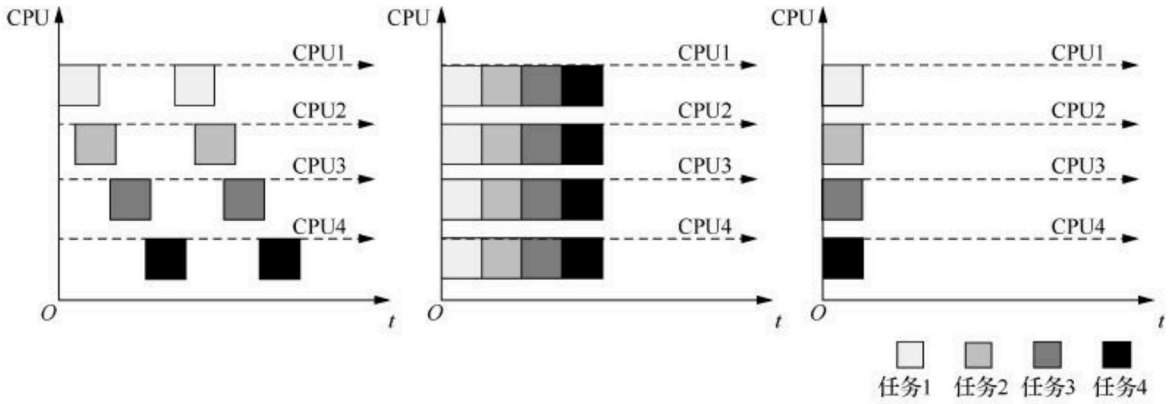


图 4.8 不同并行计算调度方法示意

UNIX 设计之初就是分时系统，通过时间分片技术同时为多个用户的进程提供计算资源，实现了时间上的并行；随着多核处理器的发展，Linux 需要针对多核进行深度优化的并行计算方法，openEuler在多核调度器和系统内核支持协程方面做了创新贡献。

### 4.4.2 进程创建

通过 fork 系统调用（简称fork 调用）即可简单地创建一个新进程。当一个应用程序执行完 fork 调用后，当前进程就会分裂为两个进程，且这两个进程都会继续执行fork 调用后的程序。这两个进程中一个是原有进程，会继续向后执行，被称为父进程；另一个是新创建的进程，被