并发编程

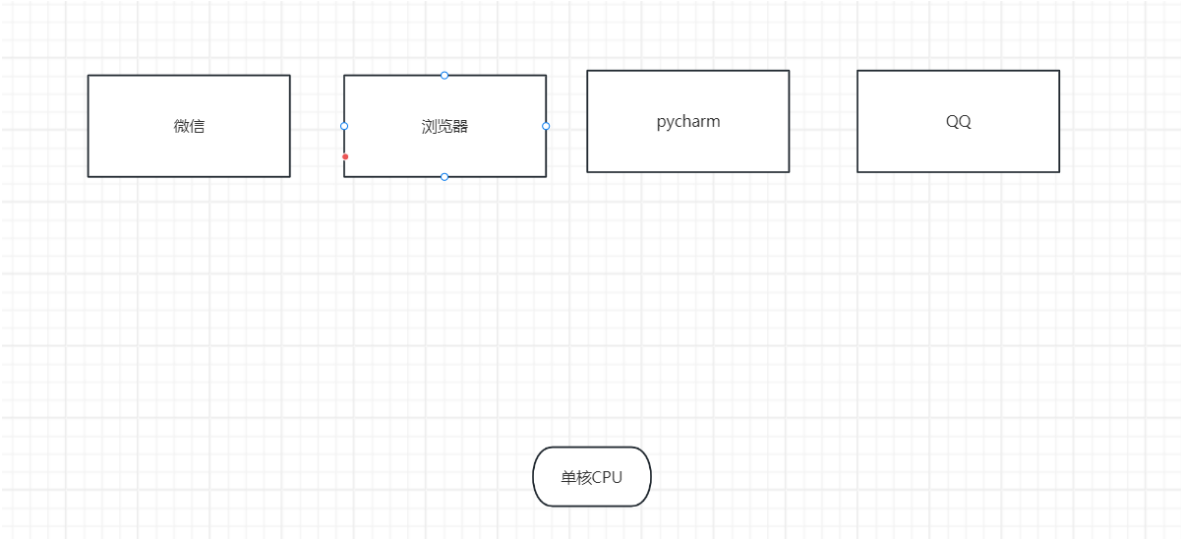
> 实现多任务的爬虫的目的：提高效率

并发与并行

> 并发与并行，他是用来描述在计算机系统中同时处理多个任务的两种不同方式

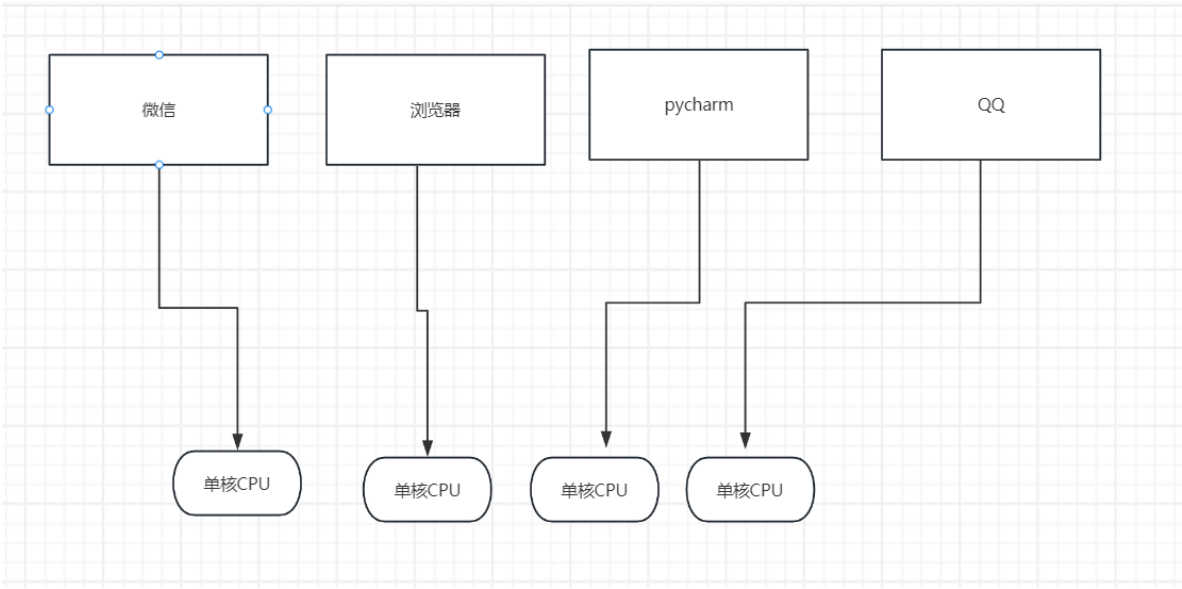
1. 多任务：在同一时间段内执行多个任务。电脑当中可以同时运行过个任务
2. 并发：系统可以处理多个任务的能力，但并不代表这些任务是同一时刻执行的。单核与多核。利用时间分片快速的不同任务之间切换。适用场景：大量等待，I/O.

> 任务数量大于cpu核数



3. 并行：真正多个任务同时执行。这通常要求的多个处理器。每一个处理器核心可以同时执行一个不同的任务

> 任务数量小于cpu核数



> 总结：并发与并行关键：并发是关于如果在单核或者多核cpu上有效的处理多任务。并行同时执行多个任务

python对并发编程

1. 多线程：threading，利用cpu和IO(输入/输出）可以同时执行的原理，cpu不会干巴巴等待IO完成（适用于i/o密集型计算)
2. 多进程：利用多核CPU，真正的执行并行任务（CPU密集型计算)
3. 多协程：在但线程当中利用CPU和IO同时执行的原理
4. 多进程里面包含了多个线程，线程包含多个协程

- 判断一个很大的数是否是素数，

> CPU密集型计算，IO密集型计算
>
> CPU密集型计算
>
> 核心资源：cpu
>
> 特点：大量的数学计算，科学计算，图像处理，大数据处理
>
> IO密集型计算：
>
> 核心资源：输入/输出设备，硬盘，网络等
>
> 特点：等待数据的读取写入，文件处理，数据库处理，网络通信，cpu经常处于等待的状态，因为他需要将数据从硬盘或者是网络传输过来

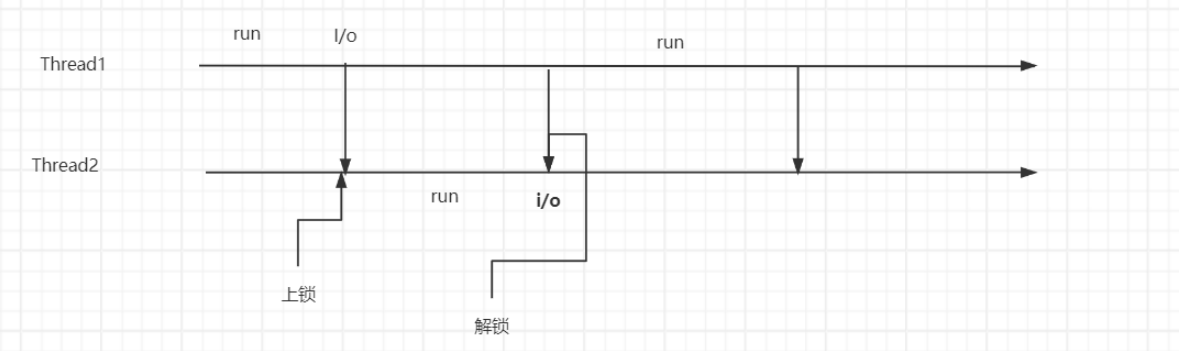
总结：

> CPU密集型计算：依赖cpu计算能力，采用多进程来提高效率
>
> IO：依赖与数据的读写速度，适合用多协程(aiohttp)或者多线程来减少等待时间

Cpython 当中是没有真正意义上的多线程

> 默认解释器都是使用cpython（jpython。。）
>
> 只允许同一个时间执行一个线程

> 全局解释器锁(GIL)。即使多核处理器，使用了GIL,同一个时间，只能执行一个线程
>
> 开启了三个线程
>
> 4核，开启3个线程



程序主入口

>

```
def demo1():
    print('demo1')


def demo2():
    print('demo2')


'''
区分：直接运行该代码，被导入运行
程序主入口
if 条件判断
__name__ ：值？
==：运算符，等于
```

```
'__main__': 字符串类型的值
'''
print(__name__)  # 直接运行：__main__  别导入执行：test1
if __name__ == '__main__':  # 'test1' == '__main__'
    demo1()  # 调用1次
    demo2()
```