Node 异步 I/O

李晟泽

并发时代的要求： 异步I/O

- 1. 用户体验：
  - 前端请求两个资源，消耗的时间分别为： M, N;
  - 同步，总的时间消耗为 M + N;
  - 异步， 时间消耗为 max(M + N);
- 2. 资源的分配:
  - 假设有一组互不相关的任务需要完成， 现行的主流方法分为两种：
  - 1. 单线程串行的执行;
    - 优点： 易于设计，符合人们顺序思考的思维方式。
    - 缺点： 性能低效，阻塞I/O导致硬件资源得不到更优的使用.
  - 2. 多线程并行的完成;
    - 优点：性能优秀，多线程并发的使用对资源的利用率更高。
    - 缺点：设计复杂，线程间死锁，状态同步，以及多线程通信的性能与安全。
      - 创建大量线程，线程间的上下文切换开销巨大。

Node 解决方案

- 利用单线程， 远离多线程死锁，状态同步问题。
  - Js 执行在单线程里， 内部完成IO另有 线程池。
- 利用异步I/O, 让单线程远离阻塞， 以更好地使用CPU。
  - 为了弥补单线程无法利用多核CPU的缺点， Node 提供了类似前端浏览器 Web Workers的子进程， 子进程可以通过工作进程高效的利用CPU和I/O.
- 异步： The application expressed interest at one point, then used the data at another point (in time and space).
- 非阻塞： The application process was free to do other tasks.
- 实际效果而言： 异步与非阻塞都达到了 并行 I/O的目的。

理想异步IO

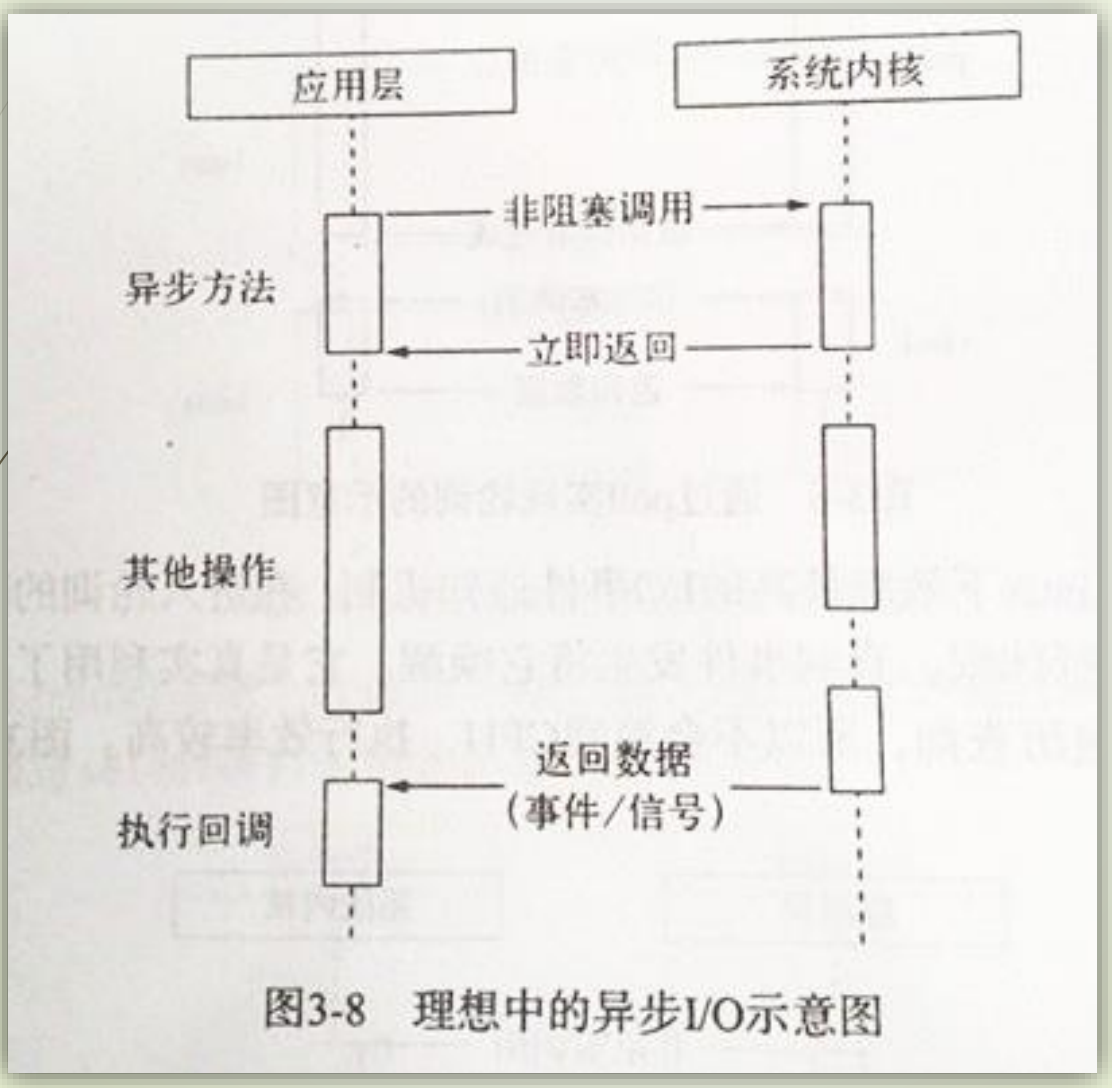

图3-8 理想中的异步I/O示意图

多线程的异步IO

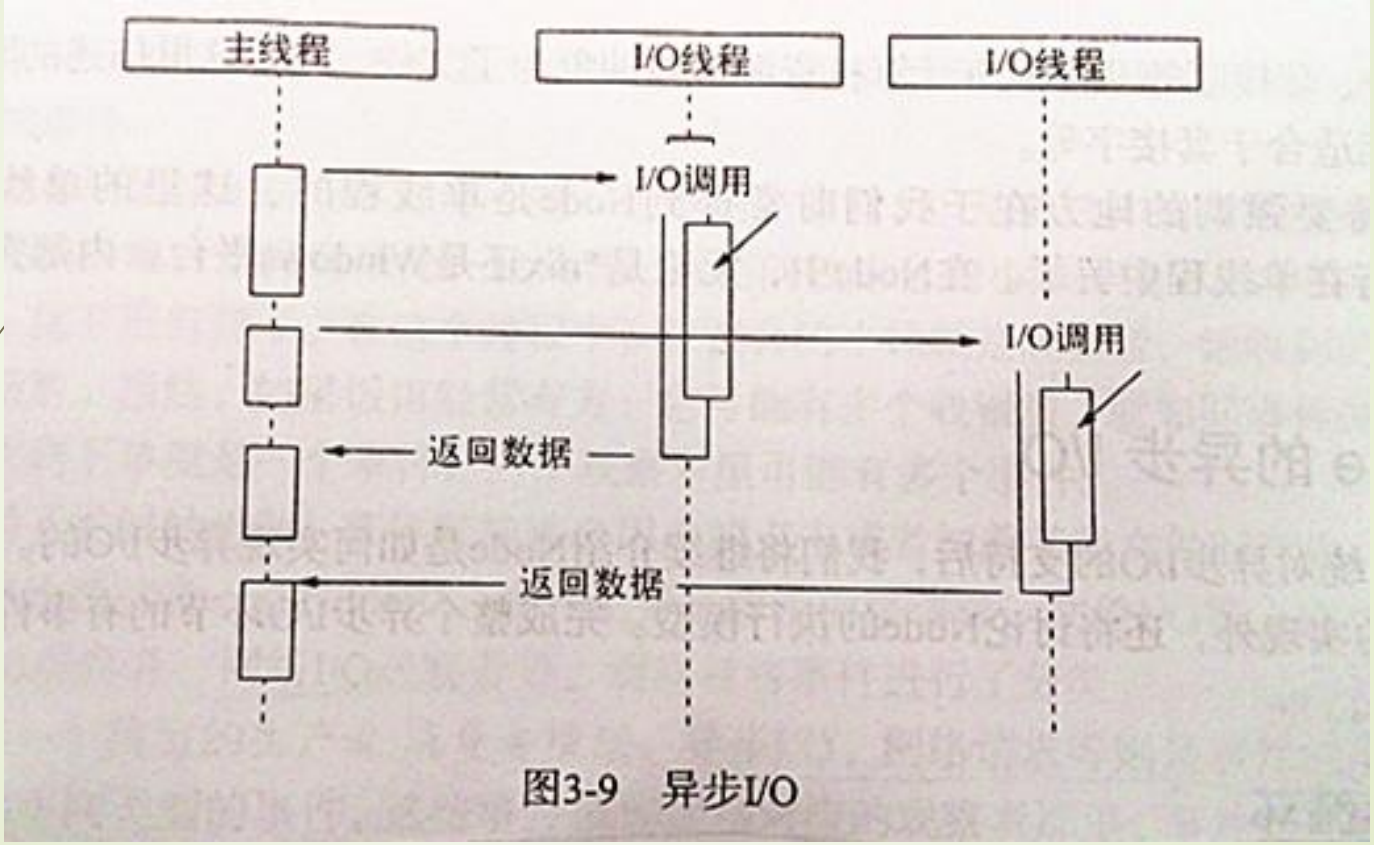

图3-9 异步I/O

Node 异步事件循环机制

因为现代操作系统大多都提供事件驱动相应的子系统， 所以应用可以要求操作系统监视socket并在队列里放置一个事件提醒器。应用会在它方便的时候检查事件然后收集数据

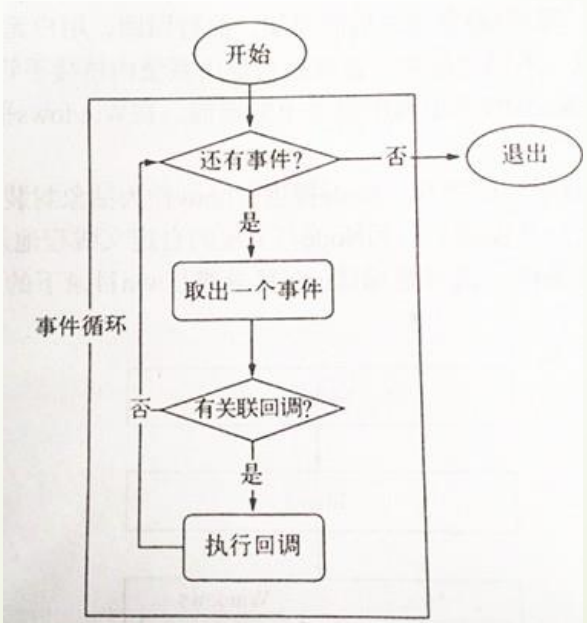

图3-11 Tick流程图

异步I/O的流程

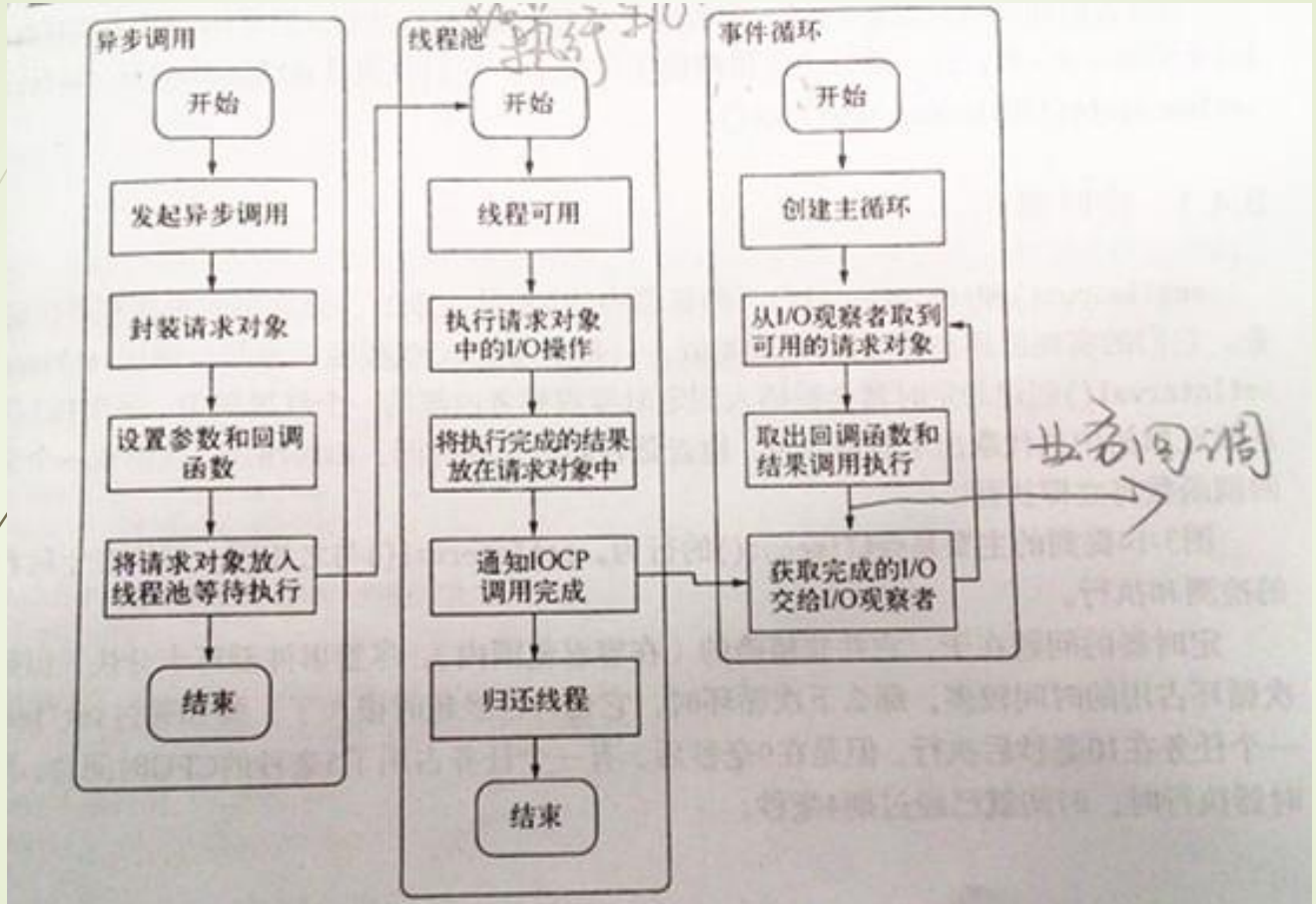

图3-13　整个异步I/O的流程

利用Node 构建web 服务器流程

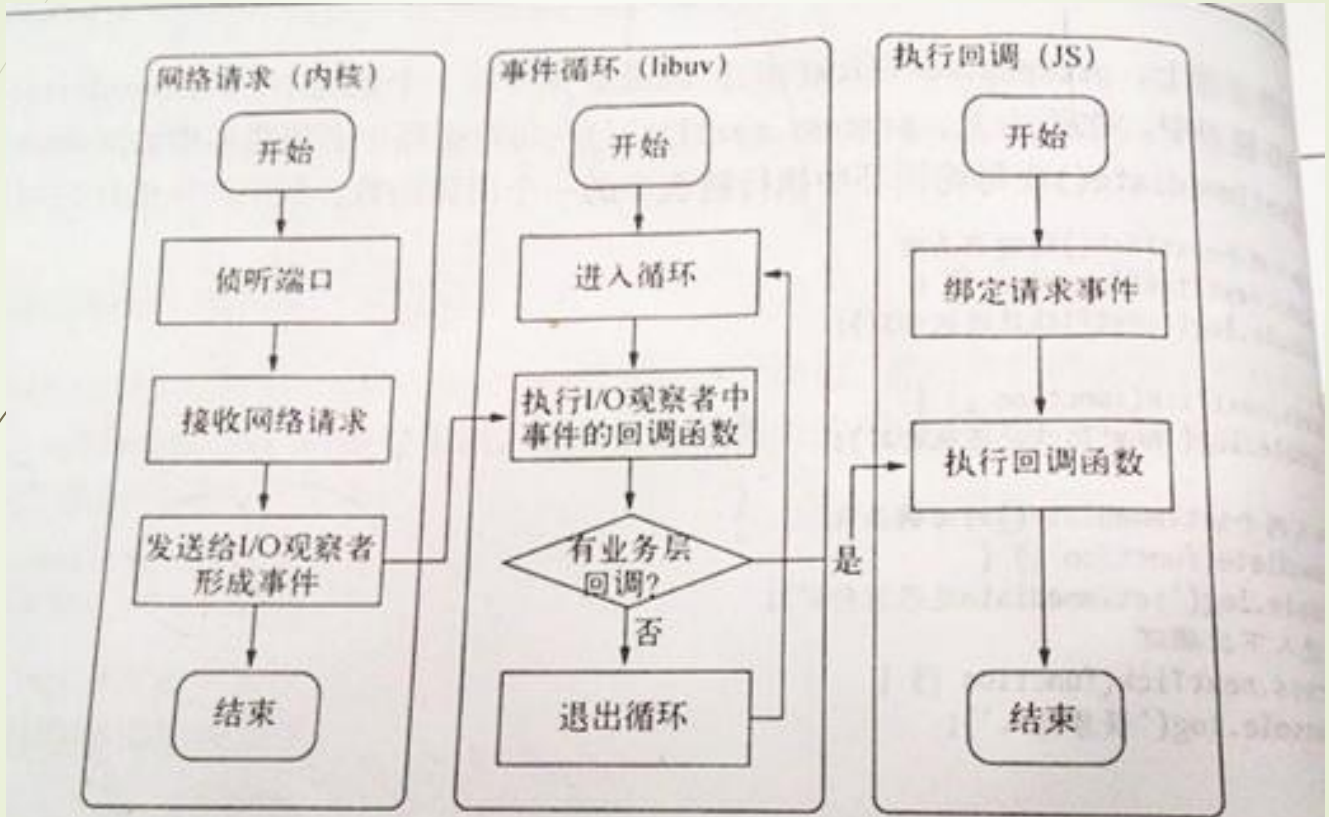

图3-15　利用Node构建Web服务器的流程图

与每请求每线程对比

- 监听I/O 完成的机制是不一样的， 一个是通过单独的线程来实现， 一个是通过事件循环机制来实现。
- 优点：
- 1. 减少了创建大量多线程的开销， 这在高并发大量的请求中尤为重要。
- 2. 避免了多线程数据间的通信， 数据是直接传送到事件循环所在的线程里！