# Node.js不是单线程的吗？为什么又有child\_process、cluster等多线程模块？

NodeJS中的JavaScript也确实是在单线程上执行，但是作为宿主的NodeJS，它本身并非是单线程的，

child\_process模块可以创建子进程

cluster通过child\_process.fork的方式创建出多个子进程实例，这些实例会自动共享相同的侦听端口。它貌似是对child\_process的封装

对于cpu密集型运算，可以分解为小任务，通过process.netxtTick延迟，错开执行；或者使用child\_process、cluster等模块创建子进程

对于Node.js多线程这块还需要进一步研究！

NodeJS 封装了内部的异步实现后，导致程序员无法直接操作线程，也就造成所有的业务逻辑运算都会丢到JavaScript的执行线程上，这也就意味着，在高并发请求的时候，I/O的问题是很好的解决了，但是所有的业务逻辑运算积少成多地都运行在JavaScript线程上，形成了一条拥挤的JavaScript运算线程。NodeJS的弱点在这个时候会暴露出来，单线程执行运算形成的瓶颈，拖慢了I/O的效率。这大概可以算得上是密集运算情况下无法很好利用多核 CPU的缺点。这条拥挤的JavaScript线程，给I/O形成了性能上限。

但是，事情又并非绝对的。回到前端浏览器中，为了解决线程拥挤的情况，Web Worker应运而生。而同样，Node也提供了child\_process.fork来创建Node的子进程。在一个Node进程就能很好的解决密集 I/O的情况下，fork出来的其余Node子进程可以当作常驻服务来解决运算阻塞的问题（将运算分发到多个Node子进程中上去，与Apache创建多个子进程类似）。当然child\_process/Web Worker的机制永远只能解决单台机器的问题，大的Web应用是不可能一台服务器就能完成所有的请求服务的。拜NodeJS在I/O上的优势，跨OS的多Node之间通信的是不算什么问题的。解决NodeJS的运算密集问题的答案其实也是非常简单的，就是将运算分发到多个CPU上。请参考文章后的multi-node的性能测试，可以看到在多Node进程的情景下，响应请求的速度被大幅度提高（感谢CNode社区的snoopy友情测试）。

在文章的写作过程中，Node最新发布的0.6.0版本，新增了cluster模块。该模块的作用是可以通过fork的方式创建出多个子进程实例，这些实例会自动共享相同的侦听端口。你可以根据当前计算机上的CPU数量来创建相应的实例数，以此达到分发请求，充分利用CPU的目的。详情请参阅[官方文档](http://http/nodejs.org/docs/v0.5.10/api/cluster.html)。在之前的解决运算密集问题中，工程师需要multi-node这样的库或者其他方案去手动分发请求，在cluster模块的支持下，可以释放掉工程师在解决此问题上的大部分精力。