关于Node.js

Node.js 是一个基于 Chrome V8 引擎的 JavaScript 运行环境。Node.js使用了事件驱动、无阻塞I/O的模型，使其轻量而且高效。因此Node.js被用来构造可伸缩的网络应用程序。

我们来看下面的例子，以下程序实现的功能是最简单的，即打印出“Hello world”

const http = require('http');

const hostname = '127.0.0.1';

const port = 1337;

http.createServer((req, res) => {

res.writeHead(200, { 'Content-Type': 'text/plain' });

res.end('Hello World\n');

}).listen(port, hostname, () => {

console.log(`Server running at http://${hostname}:${port}/`);

});

在上面的例子中，许多链接可以被同时处理。没来一个链接请求，回调函数会被调用一次，如果没有工作的话node将处在睡眠中。

基于以上一点，与现在那些通过操作系统线程实现并发的模型对比发现，基于线程的网络应用效率相对低下并且使用也相对比较复杂。进一步来说，Node用户不用担心进程死锁——因为Node没有使用锁。而且在Node中几乎没有一个方法是直接去操作I/O的，因此进程永远不会被阻塞。正是因为Node不会被阻塞，因此经验少的程序员同样也可以开发出可伸缩的系统。

受Ruby’s Event Machine 或者 Python’s Twisted 等系统的影响，Node 在设计上和这些系统很相似，Node采用事件模型，将事件轮询作为了语法结构而不是作为应用库。在其他的系统中是通过阻塞调用开始事件轮询，典型的一个应用就是在脚本开始的时候通过回调函数定义一个行为，并且在脚本结束的时候再次通过类似于EventMachine::run()的阻塞调用开启一个服务。在Node中没有start-the-event-loop 的调用，只是执行完输入脚本以后就开始进入事件轮询了，当再没有回调函数执行的时候退出轮询。这种方式就像浏览器的javascript——事件轮询对于用户来说是透明的。

以上翻译自Nodejs官网。原文网址：<https://nodejs.org/en/about/>