# 一 Node.js介绍

## 1 简介

Node的V8引擎本身就是用于Chrome浏览器的JS解释部分，但是Ryan Dahl把这个运行环境抽V8离了出来，可以在服务端运行JS代码，而并非依赖浏览器去解析。

● **Node.js不是一种独立的语言**，与PHP、JSP、Python、Perl、Ruby的“既是语言，也是平台”不同，Node.js**使用JavaScript进行编程**，运行在JavaScript引擎上（V8）。

● 与PHP、JSP等相比（PHP、JSP都要运行在服务器程序上：Apache、Nginx、Tomcat），**Node.js跳过了Apache、Naginx等服务器，不用建设在任何服务器软件之上，**即：Node.js没有web容器。

## 2 特点

所谓的特点，就是Node.js是如何解决服务器高性能瓶颈问题的。

特点一：单线程

在Java、PHP或者.net等服务器端语言中，会为每一个客户端连接创建一个新的线程。而每个线程需要耗费大约2MB内存。也就是说，理论上，一个8GB内存的服务器可以同时连接的最大用户数为4000个左右。要让Web应用程序支持更多的用户，就需要增加服务器的数量，而Web应用程序的硬件成本当然就上升了。

Node.js不为每个客户连接创建一个新的线程，而仅仅使用一个线程。当有用户连接了，就触发一个内部事件，通过非阻塞I/O、事件驱动机制，让Node.js程序宏观上也是并行的。使用Node.js，一个8GB内存的服务器，可以同时处理超过4万用户的连接。

另外，带线程的带来的好处，还有操作系统完全不再有线程创建、销毁的时间开销。

坏处，就是一个用户造成了线程的崩溃，整个服务都崩溃了，其他人也崩溃了。

多线程、单线程的一个对比：单线程也能造成宏观上的“并发”。

特点二：非阻塞I/O non-blocking I/O

例如，当在访问数据库取得数据的时候，需要一段时间。在传统的单线程处理机制中，在执行了访问数据库代码之后，整个线程都将暂停下来，等待数据库返回结果，才能执行后面的代码。**也就是说，I/O阻塞了代码的执行，极大地降低了程序的执行效率。**

由于Node.js中采用了非阻塞型I/O机制，因此在执行了访问数据库的代码之后，将立即转而执行其后面的代码，把数据库返回结果的处理代码放在回调函数中，从而提高了程序的执行效率。

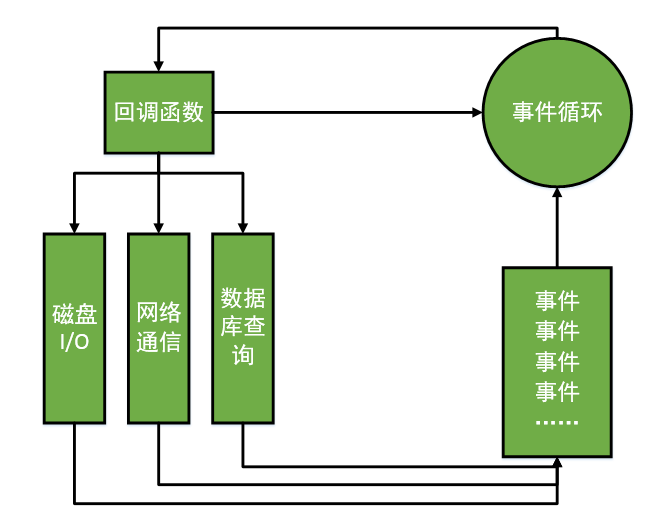
当某个I/O执行完毕时，将以事件的形式通知执行I/O操作的线程，线程执行这个事件的回调函数。为了处理异步I/O，线程必须有事件循环，不断的检查有没有未处理的事件，依次予以处理。

阻塞模式下，一个线程只能处理一项任务，要想提高吞吐量必须通过多线程。**而非阻塞模式下，一个线程永远在执行计算操作，这个线程的CPU核心利用率永远是100%。**所以，这是一种特别有哲理的解决方案：**与其人多，但是好多人闲着；还不如一个人玩命，往死里干活儿。**

特点三：事件驱动event-driven

在Node中，客户端请求建立连接，提交数据等行为，会触发相应的事件。在Node中，在一个时刻，只能执行一个事件回调函数，但是在执行一个事件回调函数的中途，可以转而处理其他事件（比如，又有新用户连接了），然后返回继续执行原事件的回调函数，这种处理机制，称为“事件环”机制。

Node.js底层是C++（V8也是C++写的）。**底层代码中，近半数都用于事件队列、回调函数队列的构建。**用事件驱动来完成服务器的任务调度，这是鬼才才能想到的。针尖上的舞蹈，用一个线程，担负起了处理非常多的任务的使命。



**单线程**，单线程的好处，减少了内存开销，操作系统的内存换页。

如果某一个事情，进入了，但是被I/O阻塞了，所以这个线程就阻塞了。

**非阻塞I/O**， 不会傻等I/O语句结束，而会执行后面的语句。

非阻塞就能解决问题了么？比如执行着小红的业务，执行过程中，小刚的I/O回调完成了，此时怎么办？？

**事件机制，事件环**，不管是新用户的请求，还是老用户的I/O完成，都将以事件方式加入事件环，等待调度。

注意：

Node.js中所有的I/O都是异步的，回调函数，套回调函数。

同步处理方式中，每个请求用一个线程或者进程处理，一次请求处理完毕后，线程被回收。

Node使用一个主线程处理请求，遇到IO后执行异步，并立即返回，基于事件模型得到异步结果（并不是轮询）。相对于等待IO、开启线程等处理方式，Node有极大优势：即避免了等待和让CPU频繁在上下文切换。

## 3 适合开发什么？

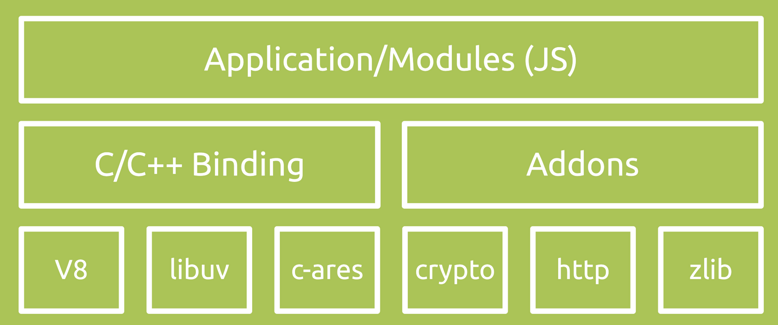
Node善于I/O，不善于计算。即CPU密集型Node不擅长，Node擅长的是IO密集型。

因为Node.js最擅长的就是任务调度，如果你的业务有很多的CPU计算，实际上也相当于这个计算阻塞了这个单线程，就不适合Node开发。

**当应用程序需要处理大量并发的I/O，而在向客户端发出响应之前，应用程序内部并不需要进行非常复杂的处理的时候，Node.js非常适合。Node.js也非常适合与web socket配合，开发长连接的实时交互应用程序。**

比如：用户表单收集、聊天室、考试系统、图文直播、提供jsonAPI。

## 4 Node结构



如图所示，binging一层是JS与底层C++沟通的关键，前者通过bindings调用后者，相互交换数据。

libuv为Node提供了跨平台、线程池、事件池、异步IO能力，是Node核心。

## 5 异步费阻塞与单线程

一个异步IO的流程大致如下：

A 发起异步IO调用

1 用户通过JS代码调用Node核心模块，将参数和回调函数传入核心模块；

2 Node核心模块会将传入的参数和回调函数封装成一个请求对象；

3 将这个请求对象推入IO线程池等待执行；

4 JS发起异步调用结束后，JS线程池继续执行后续操作

B 执行回调

1 IO操作完成后将结果储存到请求对象的result属性上，并发出操作完成的通知；

2 每次事件循环时会检查是否有完成的IO操作，如果有就将请求对象加入IO观察者，并将result当做参数，以完成JS回调的目的。

3 处理IO观察者事件时会取出之前封装在请求对象中的回调函数，执行这个回调函数，并将result当做参数，以完成JS回调的目的。

单线程是指主线程为单线程，所有阻塞的部分交给一个线程池处理，然后这个主线程通过一个队列跟线程池协作，我们写JS代码不再关心线程问题，代码也主要由一堆callback回调构成，然后主线程在循环过程中适时调用这些代码。

# 二 Node安装

## 2.0 Node版本问题

偶数位版本为稳定版，奇数位版本为非稳定版。

## 2.1 学习用安装

初学者可以在Node官网下载安装包，下一步下一步安装即可，并（可能）需要配置环境变量。

## 2.2 nvm介绍

nvm是一款可以管理node版本的工具，所以在企业级开发中，我们可以使用nvm来安装node，这样可以方便我们控制node的版本。

## 2.3 Win安装Node（使用nvm）

### 2.3.1安装nvm

<https://github.com/coreybutler/nvm-windows/releases>

安装完毕后使用命令：

查看安装情况 nvm

查看版本 nvm version

### 2.3.2设置nvm安装镜像

nvm的Node包存放在 <http://nodejs.org/dist/> 。由于是国外的服务器.......

windows设置镜像：

nvm node\_mirror https://npm.taobao.org/mirrors/node/

nvm npm\_mirror https://npm.taobao.org/mirrors/npm/

（当然也可以在nvm安装路径下，打开settings.txt，直接追加）

### 2.3.3 使用nvm安装node

# 安装最新版

nvm install latest

# 安装指定版本

nvm install 8.5.0

# 切换版本

nvm use 8.5.0

# 查看当前使用版本

nvm use ls

### 2.3.4 设置npm镜像

npm是node的包管理工具，用来下载各种node模块、第三方文件，由于npm也在国外，So。。。

npm install jQuery

上图用npm安装可能会很慢，较便捷的做法是放弃npm，使用国内的一款软件 cnpm 来下载Node包，然后再使用 cnpm安装需要的包

npm install cnpm -g

cnpm install jQuery

如果你不想使用cnpm，还是想用npm，可以设置npm的镜像：删除代理，设置镜像

npm config rm proxy

npm config rm https-proxy

npm config set registry https://registry.npm.taobao.org

虽然上述步骤能实现用npm安装大部分软件，但是有些地方仍然是一坨shit，比如安装Angular cli，仍然会出错，还要把镜像设置回来。。。

npm config set proxy=http://127.0.0.1:1080

npm config set registry=http://registry.npmjs.org

其他，git代理方法解决：

git config --global http.proxy 'socks5://127.0.0.1:1080'

git config --global https.proxy 'socks5://127.0.0.1:1080'

git config --global --unset http.proxy

git config --global --unset https.proxy

npm config delete proxy

## 2.4 CentOS安装Node（使用nvm）

### 2.4.1安装nvm

切换管理员用户： su root

安装git: yum install git

然后执行：

git clone https:*//github.com/cnpm/nvm.git ~/.nvm && cd ~/.nvm && git checkout `git describe --abbrev=0 --tags`*

### 2.4.2 激活nvm

echo ". ~/.nvm/nvm.sh" >> /etc/profile  
source /etc/profile

### 2.4.2设置nvm安装镜像

NVM\_NODEJS\_ORG\_MIRROR=https://npm.taobao.org/mirrors/node

如果你不想每次都输入环境变量，可以将其配置在.bashrc文件中，

export NVM\_NODEJS\_ORG\_MIRROR=https://npm.taobao.org/mirrors/node

source ~/git/nvm/nvm.sh

### 2.4.3 使用nvm安装node

# 安装指定版本

# 切换版本

nvm use 8.5.0

# 查看当前使用版本

nvm use ls

### 2.4.4设置npm镜像

npm是node的包管理工具，用来下载各种node模块、第三方文件，由于npm也在国外，So。。。

npm install jQuery

上图用npm安装可能会很慢，较便捷的做法是放弃npm，使用国内的一款软件 cnpm 来下载Node包，然后再使用 cnpm安装需要的包

npm install cnpm -g

cnpm install jQuery

如果你不想使用cnpm，还是想用npm，可以设置npm的镜像：删除代理，设置镜像

npm config rm proxy

npm config rm https-proxy

npm config set registry https://registry.npm.taobao.org

虽然上述步骤能实现用npm安装大部分软件，但是有些地方仍然是一坨shit，比如安装Angular cli，仍然会出错，还要把镜像设置回来。。。

npm config set proxy=http://127.0.0.1:1080

npm config set registry=http://registry.npmjs.org

其他，git代理方法解决：

git config --global http.proxy 'socks5://127.0.0.1:1080'

git config --global https.proxy 'socks5://127.0.0.1:1080'

git config --global --unset http.proxy

git config --global --unset https.proxy

npm config delete proxy

### 2.4.5 开机自动运行 解决多窗口无法运行node

echo "nvm use v8.5.0" >> /etc/profile  
source /etc/profile

### 2.4.6 设置默认版本

nvm alias default v8.5.0

## 2.5 Mac安装Node

## 2.6 node 查看版本

安装完node后，可以使用node -v 查看node版本；

## 2.7 webstorme配置

安装完node后，打开webstorme，会自动识别node路径，如果没有识别：

可以打开Settings-搜索node-配置；

但是此时webstorme 仍然没Node的代码提示功能，解决步骤：

File-setting-Languages&Frameworks-Node-



在这里配置源码地址，可以手动选择自己下载的或者让webstorme自己下载。

# 三 PM2管理工具

由于Node是单进程，很容易服务器死去，无法重启，pm2可以有效的监控服务器状况，并让node服务器自动重启。

npm install pm2 -g

开启：pm2 start \*\*\*.js （node关闭窗口仍然运行，且会自动重启）

关闭：pm2 stop \*\*.js

重启：pm2 restart \*\*.js

# 四 REPL

控制台直接输入node即可进入Node的REPL环境，使用ctrl+c或者输入 .exit退出。

chrome也提供了类似的工具，使用F12直接打开

# 五 NPM

npm -v 查看版本

npm init 初始化一个node项目描述文件

npm init -y或者-yes 初始化一个node项目描述文件，且使用默认值直接生成。

npm install 安装依赖

npm install --save 或者-S 安装依赖并记录到json中

npm install -dev 安装依赖并记录到json的开发依赖中