初学Node的小心得
之
Node是干嘛的

node.js是什么？

“它是一个Javascript运行环境，能让Javascript运行于服务器端。”

node.js是干嘛的？为什么我们要用node.js？

“Node.js 使用**事件驱动**， **非阻塞I/O 模型**而得以高效，能方便地搭建响应速度快的网络应用。”

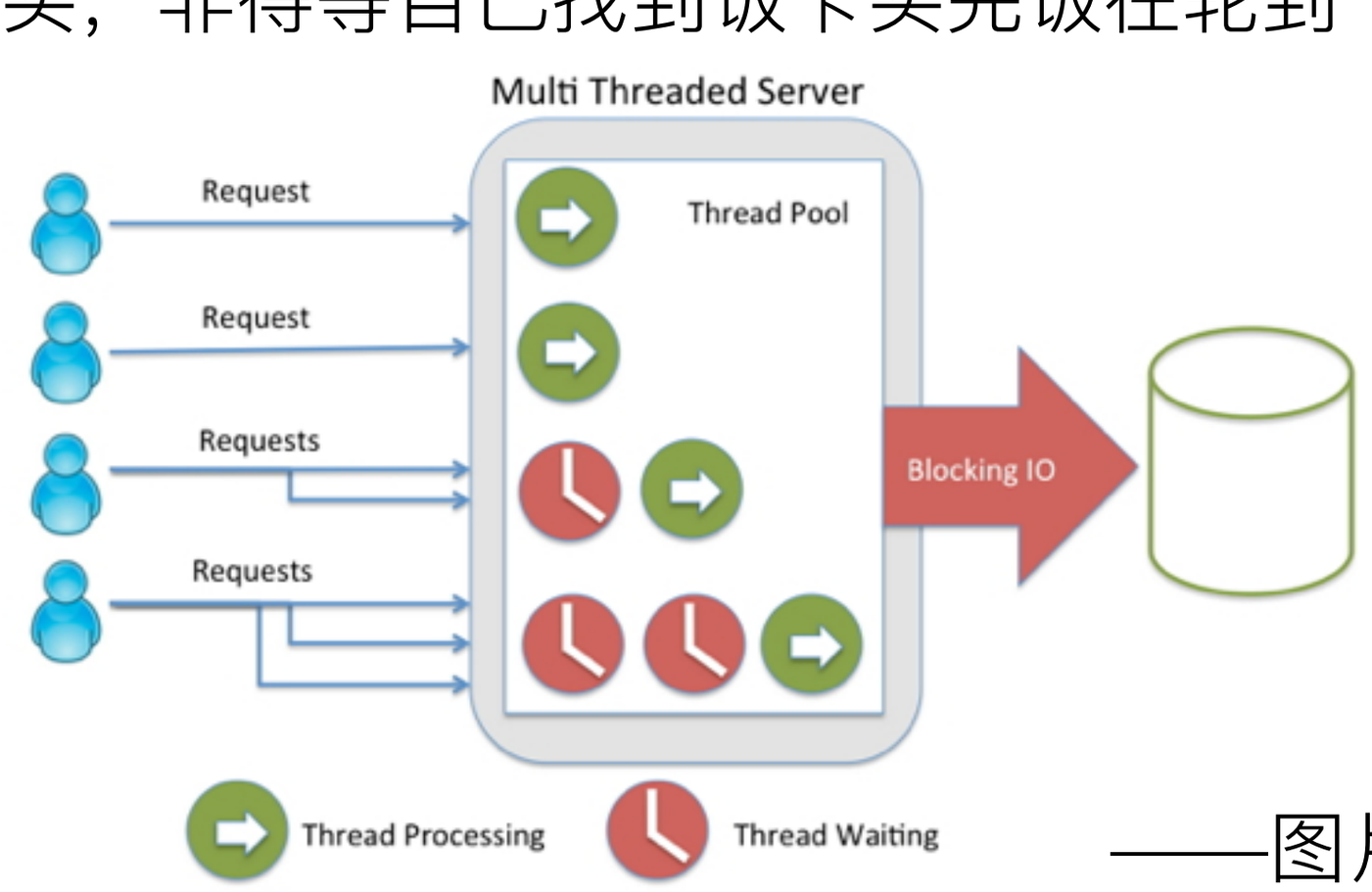
——from ~~百度~~ 谷歌

哦，不懂。

阻塞式IO

程序执行过程中必然要进行很多I/O操作，读写文件、输入输出、请求响应等等。I/O操作很费时。如果你要读一个文件，整个线程都暂停下来，等待文件读完后继续执行。换言之，I/O操作阻塞了代码的执行，降低了程序的效率。

（简单来说就是你排队买饭的时候找不到饭卡，但你坚决不让后面排队的人先买，非得等自己找到饭卡买完饭在轮到下一个。）

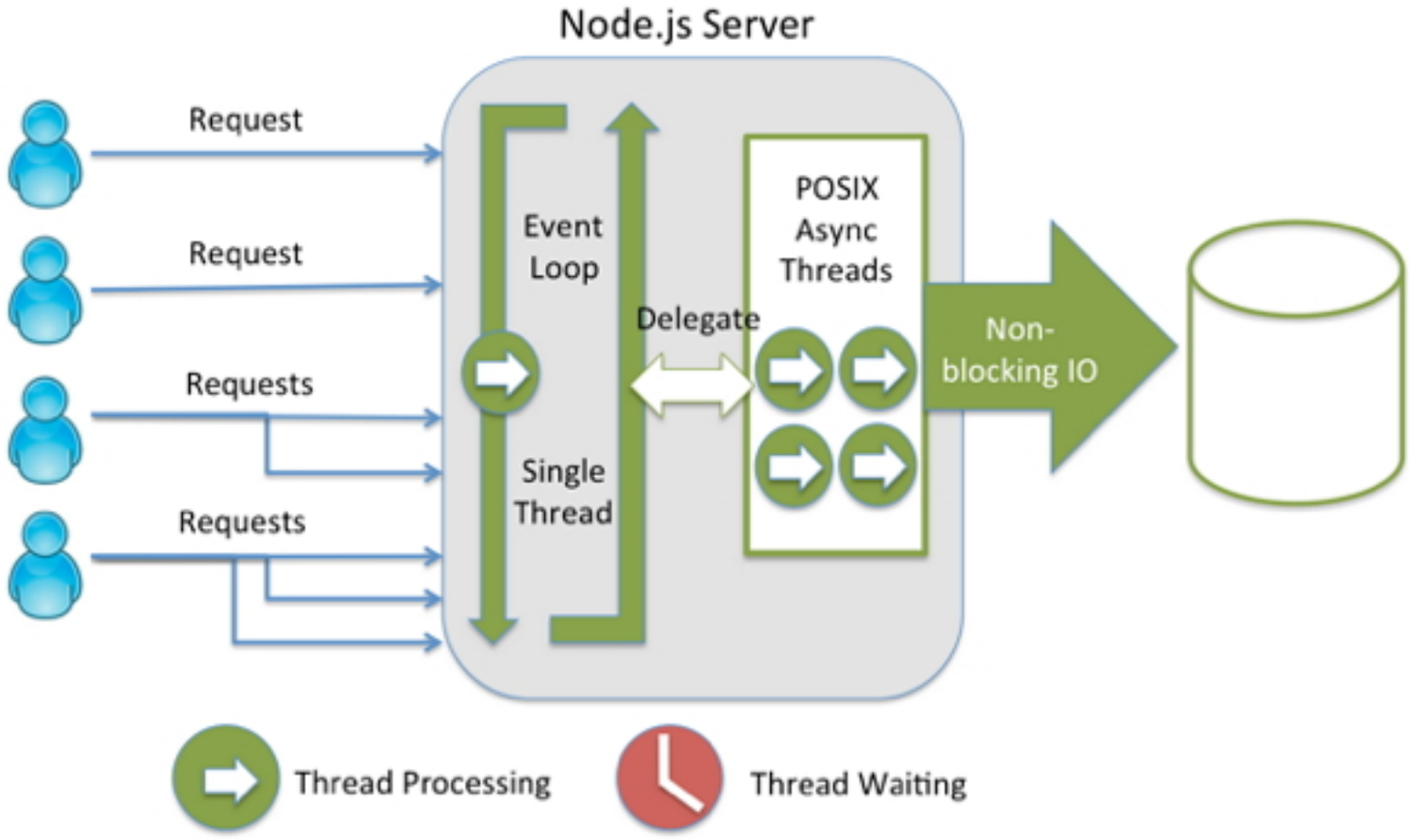


——图片来自百度 谷歌

非阻塞式IO

　　非阻塞I/O是程序执行过程中，I/O操作不会阻塞程序的执行，也就是在I/O操作的同时，继续执行其他代码。（通过事件驱动来处理。）

（简单来说就是你排队买饭的时候找不到饭卡，但是你退出来让别人先买饭，等自己找到饭卡了在进入队列中买饭。）

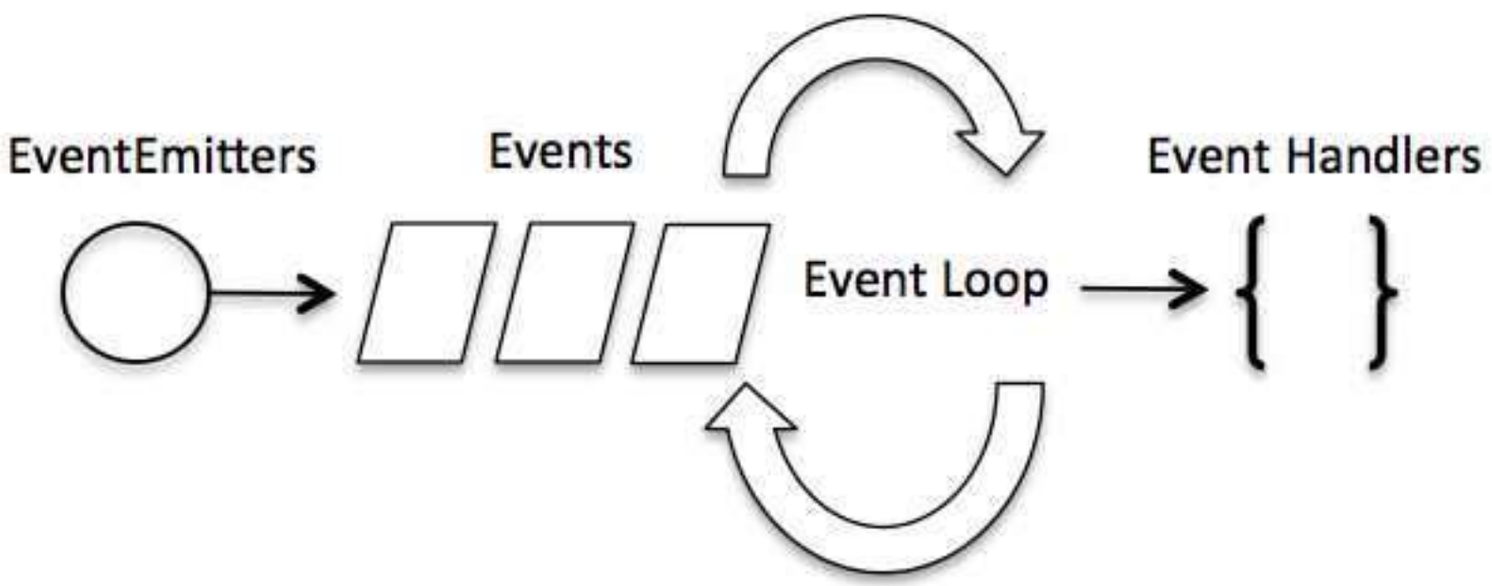


看个小栗子

```
//check whether it is blocked

var fs = require("fs");
var file = 'text.txt';

fs.readFile(file, function(error, data) {
    if (error) throw error;
    else {
                console.log("我读完文件了! ");
    }
});
console.log("我没有被阻塞! ");

/*
我没有被阻塞!
我读完文件了!
*/
```

事件驱动

Node通过事件驱动的方式处理请求时无需为每一个请求创建额外的线程。在事件驱动的模型当中，每一个IO工作被添加到事件队列中，线程循环地处理队列上的工作任务，当执行过程中遇到来堵塞(读取文件、查询数据库)时，线程不会停下来等待结果，而是留下一个处理结果的回调函数，转而继续执行队列中的下一个任务。这个传递到队列中的回调函数在堵塞任务运行结束后才被线程调用。



再看个小栗子

```
//check Node's event-driven

var http = require('http');

var server=http.createServer();

server.on('request',function(request,response){

    console.log('request event');

    response.writeHead(200,{'Content-Type':'text/plain'});
    response.end('Hello World\n');
});

server.on('connection',function(){
    console.log('connection event');
});

server.listen(8124,function(){
    console.log('listening event');
});

console.log('Server running on port 8124');

/*
result:
Server running on port 8124
listening event
connection event
request event
*/
```

所以node.js是干嘛的？为什么我们要用node.js?

“Node.js 使用**事件驱动**， **非阻塞I/O 模型**而得以高效，能方便地搭建响应速度快的网络应用。”

哦。