Node.js: 事件驱动并发的Web应用

Ganesh Iyer, BTech.(Computer Engineering), SVNIT

# 1. 简介

## 1.1 Nodejs简介

Node（也称为Node.js，Nodejs，NodeJS）是V8 JavaScript引擎的事件驱动I/O框架。它旨在编写可扩展的网络程序，如Web服务器。程序是用JavaScript编写的，使用事件驱动，异步I/O，以最小化开销并最大化可扩展性。它基于谷歌的运行时实现恰当命名的“V8”引擎。但是，虽然V8主要支持浏览器中的JavaScript（尤其是谷歌Chrome），但Node的目标是长期支持运行服务器进程消耗。

与其他现代环境不同，Node进程不依赖于多线程来支持并发执行；它基于异步事件驱动I/O模型。它的事件驱动，非阻塞I/O模型使其轻量且高效，是数据的理想选择密集的真实跨分布式服务器运行的时间应用程序。

## 1.2 背景

互联网不断发展，并为Web应用程序设计者带来了许多挑战。大型互联网服务必须以前所未有的规模处理并发问题。高流量需要服务以更好地处理并发会话。随着对Internet服务的需求的增长，必须使用新的系统设计技术来管理这种负载。

在Web服务器中处理并发性有两种流行的策略：线程和事件。Web应用程序通过将每个传入请求分配给单独的执行线程来实现线程。例如，Apache使用在其多处理模块工人MPM的方法。相比之下，事件为基础的系统，利用单个线程来处理事件，如到来的请求，从一个队列中。每种方法都有一些固有的优点和缺点。

构建高度并发的系统本身就很困难。构建代码以实现高吞吐量并不是很好由现有的编程模型支持。虽然线程区域常用于表示并发性的设备，但许多线程实现的高资源使用和可伸缩性限制已使开发人员更喜欢事件驱动方法。

Node由RyanDahl于2009年创建，其增长由Joyent赞助。Node的最初目标是创建使用推送功能的网站的能力，如在几个Web应用程序（Gmail等）中看到的那样。Node受到Ruby的EventMachine和Python的Twisted等系统的影响。Node通过将事件循环呈现为语言结构而不是库来进一步获取事件模型。

## 1.3 动机

Web服务器必须能够同时处理多个用户。传统的方法，每个请求使用一个线程，例如：Apache在其多处理模块WorkerMPM中使用这种方法，是有一些缺点的。无论随着系统中线程数量的增长而制作线程服务器的程度如何，操作系统开销（调度和内存占用量）都会增加。此方法限制了可以同时连接到服务器的客户端数量。C10K问题（并发10,000个连接），这是指优化Web服务器软件以同时处理大量客户端的问题，这是最值得注意的。

事件驱动系统往往对负载具有鲁棒性，吞吐量几乎没有降低，因为提供的负载增加超出了系统可以提供的负载。吞吐量超过其线程对应的吞吐量，但更重要的是不会随着并发性的增加而降低。并发在事件驱动方法中是明确的，程序员可以利用特定于应用程序的知识来重新排序事件处理，以确定优先级或效率。

Node有一个执行线程，无需等待I/O或上下文。相反，I/O调用设置了与事件循环工作分派事件时，一些事情变得可用的请求处理功能。程序执行应该快速返回到事件循环，以便分派下一次立即运行能干的任务。JavaScript非常适合这种方法，因为它支持事件回调。例如，当浏览器完全加载文档，用户单击按钮或完成AJAX请求时，事件会触发回调。JavaScript的功能性使得创建可以注册为事件处理程序的匿名函数对象变得非常容易。在服务器上也使用相同的语言方和客户方面有很大的好处。

## 1.4 小结

在第1节中，我们提供了Nodejs的概述。在第2节中，给出了对现代系统中处理并发性的两种最普遍的策略的调查：线程和事件。我们首先探索基于线程的服务器体系结构，在大规模应用程序中突出显示可伸缩性方面的问题，例如此方法的C10K问题。然后，我们提供该活动的概述驱动的服务器架构。

在第3节中，我们讨论JavaScript语言，使其适合事件的功能驱动I/O.提供了Node的架构概述。事件在Node中使用基于并发模型进行了探讨。我们还讨论了Node应用程序中最常用的基本中间件。此外，使用Node和服务器的优势列出了JavaScript，稍后突出显示了Node面临的挑战。

最后，我们通过显示事件驱动，非结束报告阻塞I/O模型对于构建可伸缩的Web应用程序非常有效。

# 2. Web服务器中的并发性

传统上有两种主要的服务器体系结构，基于线程和事件。在本章中，我们从可伸缩性的角度介绍和评估这些体系结构。

## 2.1 基于线程的服务器架构

线程基于方法基本上将每个传入连接与单独的线程相关联。通过这种方式，同步阻塞I/O是处理I/O的自然方式。这是许多编程语言都很好支持的常用方法。它还导致直接的编程模型，因为请求处理所需的所有任务都可以按顺序编码。此外，它通过隔离请求和隐藏并发性来提供简单的抽象。通过同时使用多个线程或进程来实现真正的并发性。

流行的Apache Web服务器提供了强大的多功能实现混合多路的处理模块过程多线程服务器。通过使用线程来处理请求，它能够以比进程更少的系统资源提供大量请求基于服务器。但是它保留了过程的大部分稳定性通过保持多个进程，每个都有多个线程基于服务器。

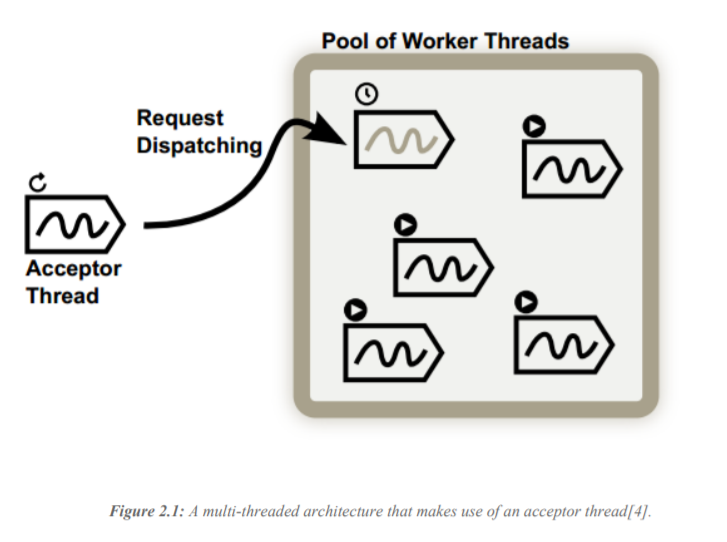
从概念上讲，多处理和多线程体系结构具有相同的原则：每个新连接都由专用活动处理（进程或线程）。

### 2.1.1 多进程架构

传统的网络服务器方法就是这个过程连接模型，使用专用进程处理连接。此模型用于第一个HTTP服务器CERN httpd。进程的创建是一个昂贵的操作和服务器经常采用的策略称为进程并发。进程的重量级结构限制了最大同时连接数。多流程体系结构仅为并发请求提供有限的可伸缩性。

### 2.1.2 多线程架构

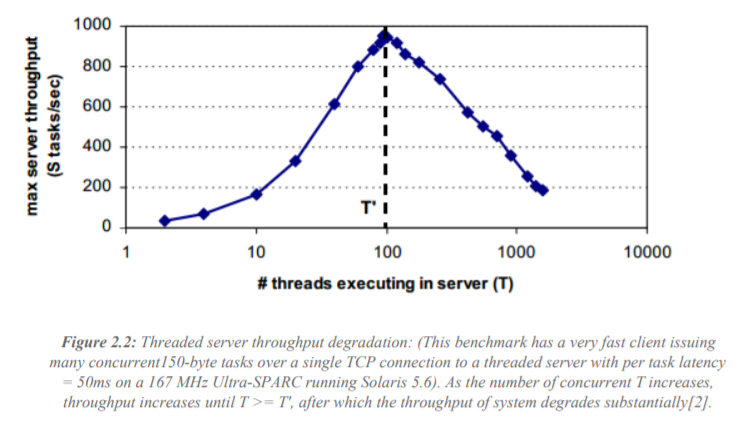
随着线程库的出现，已经出现了服务器架构，它们用更轻量级的线程取代了重量级进程。实际上，他们使用一个线程连接模型。多线程方法虽然遵循相同的原则，但在线程的轻量级结构方面是不同的。与整个进程的内存大小相比，线程占用的内存有限。此外，线程需要较少的资源来创建和终止。



通常的做法是将单个接受器线程放在线程池前面以进行连接处理，如图4.2所示。线程池是绑定服务器内最大线程数的常用方法。接受器阻塞新的套接字连接，接受连接并将它们分派给工作池并继续。工作池提供了一组处理传入请求的线程。工作线程要么处理请求，要么等待处理新请求。

### 2.1.3 多可扩展性问题线程架构

无论线程服务器的精确程度如何，随着系统中线程数量的增加，操作系统开销也随之增加（调度和聚合内存占用量）增加，导致系统整体性能下降。通常存在给定系统可以支持的最多8个线程数T，超过该线程T将发生性能降级。



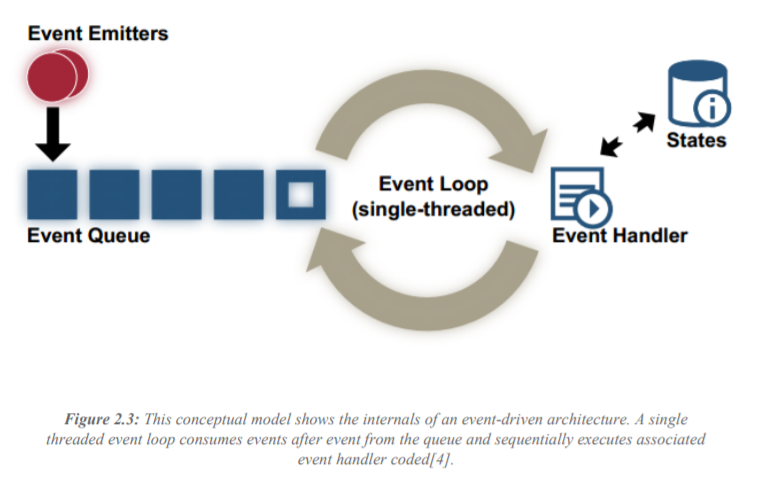
在重载下，多重线程服务器消耗大量内存（由于每个连接的单个线程堆栈），并且上下文切换导致相当大的CPU时间损失。其间接惩罚是CPU缓存未命中的可能性增加。减少线程的绝对数量增加线程性能但在最大同时连接方面限制了整体可伸缩性。

### 2.1.4 C10K问题

C10k问题是指优化Web服务器软件同时处理大量客户端的问题（因此名称为C10k并发一万个连接）。凯格尔在1999年的开创性文章中发表了这个问题，宣称“现在是网络服务器同时处理一万个客户的时候了”。观察到基于线程的方法具有为每个客户端使用整个堆栈帧的缺点，这会花费存储器。这可能会导致的记忆错误。然后调查了几种异步I/O方法，它们对缩放有影响，可以解决这个问题。

## 2.2 事件驱动的服务器架构

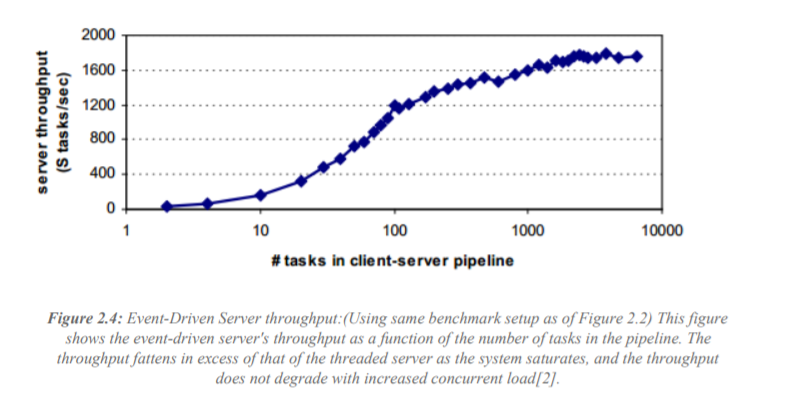
作为同步阻塞I/O的替代方案，事件驱动方法在服务器体系结构中也很常见。由于异步或非阻塞调用语义，其他模型比之前概述的线程每需要连接模型。常见的模型是单个线程到多个连接的映射。然后，该线程处理所有发生的事件，形成这些连接和请求的I/O操作。如图2.3所示，新事件排队，线程执行称为事件循环从队列中出列事件，处理事件，然后进行下一个事件或等待推送新事件。



处理事件要么需要为特定事件注册事件处理程序代码，要么基于事先与事件关联的回调的执行。线程处理的连接的不同状态以适当的数据结构组织要么明确地使用有限状态机，要么隐式地通过闭包回调。结果，事件之后的应用程序的控制流程驱动风格以某种方式倒置。一个事件而不是顺序操作驱动程序使用一系列异步调用和在事件上执行的回调。

让单个线程运行事件循环并等待I/O通知对可伸缩性的影响与线程不同基于方法。不关联连接和线程会大大减少服务器的线程数在极端情况下，直到单一事件循环加上一些用于I/O的OS内核线程。因此，我们摆脱了过度上下文切换的开销。

事件驱动系统往往对负载很稳健，吞吐量几乎没有降低，因为提供的负载增加超出了系统可以提供的负载。如果事件的处理和任务状态的捆绑有效，则峰值吞吐量可能很高。图2.4显示了事件的吞吐量作为负载的函数驱动网络服务的实现。吞吐量超过了线程服务器的吞吐量，但最重要的是不会因并发性增加而降级。



# 3. Nodejs的基础知识

Node是一个构建活动的平台驱动的网络程序，例如Web服务器。它为开发人员提供了JavaScript运行时和库来编写应用程序。JavaScript运行时是V8;与谷歌的网络浏览器Chrome使用的引擎相同。V8使用动态机器代码生成来实现高性能。清单3.1显示了在Node中实现的一个简单的helloworld服务器。



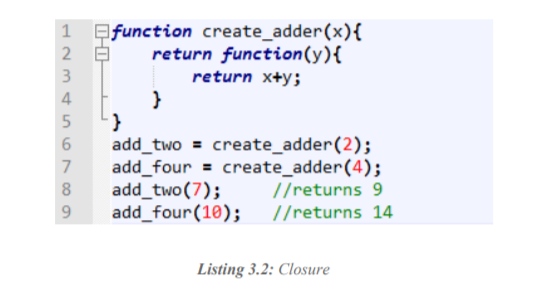
不像其他的现代环境中，一个节点的过程不依赖于多线程来支持业务逻辑的并发执行；它是基于一个异步I/O活动荷兰国际集团模型。节点服务器进程可以被认为是单个嵌入JavaScript引擎以支持自定义的线程守护程序。这与其他编程语言的大多数事件系统不同，后者以库的形式出现。Node支持语言级别的事件模型。

## 3.1 JavaScript

由BrendanEich于1995年在Netscape构建的JavaScript被认为是Netscape浏览器Navigator的脚本语言。JavaScript从此成为客户端网络的侧面脚本语言，是最广泛使用的语言之一语言，所有带JavaScript的现代浏览器引擎和几乎所有PC都安装了至少一个浏览器。JavaScript的语法是由C和Java的启发，但语言的内部工作是通过方案的启发。它具有原型基于继承的模型和函数是第一类对象。它还具有lambdas即匿名函数。

### 3.1.1 闭包

函数可能存在于函数内部，并作为“第一类对象”返回到它最初构造的函数之外的某个变量。当发生这种情况时，会创建一个新的闭包。闭包是一个包含函数和非引用环境的数据结构该函数的局部变量。



清单3.2显示了创建内部函数'create\_adder'（匿名函数）并返回它。当'create\_adder'返回并且它仍然可以访问x时，可以稍后调用内部函数。

### 3.1.2 服务器边JavaScript

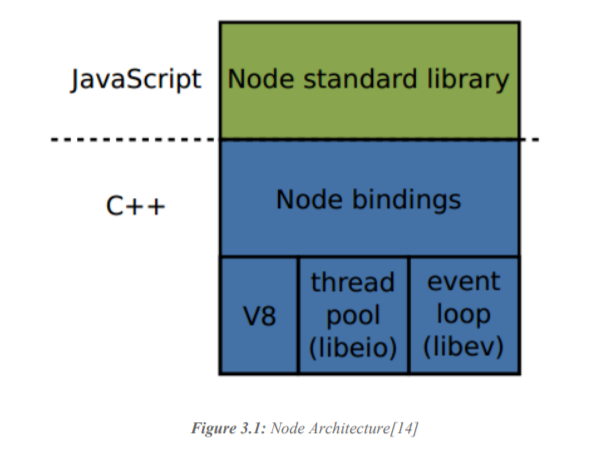
Netscape为服务器引入了语言的实现1994年12月使用NetscapeEnterpriseServer编写脚本。尽管JavaScript是客户端最流行的语言，但它在服务器端并不占优势。从中期开始2000年代，服务器出现了激增侧面JavaScript实现。Rhino是一个由MozillaFoundation管理的开源JavaScript引擎，旨在用于服务器副申请。存在其他几个实现，如10gen，ApacheSling等。Node是服务器最近值得注意的一个例子边JavaScript。

## 3.2 Nodejs架构

图3.1提供了Node组件的高级概述。V8是JavaScript引擎，libeio处理内部线程池，用于生成同步POSIX异步调用事件循环。不使用异步文件系统，而是在其自己的线程中执行阻塞I/O，因此它对Node中的事件循环是非阻塞的。libev是事件循环。节点绑定是一些简单的C++绑定，它们将底层组件的API暴露给JavaScript。

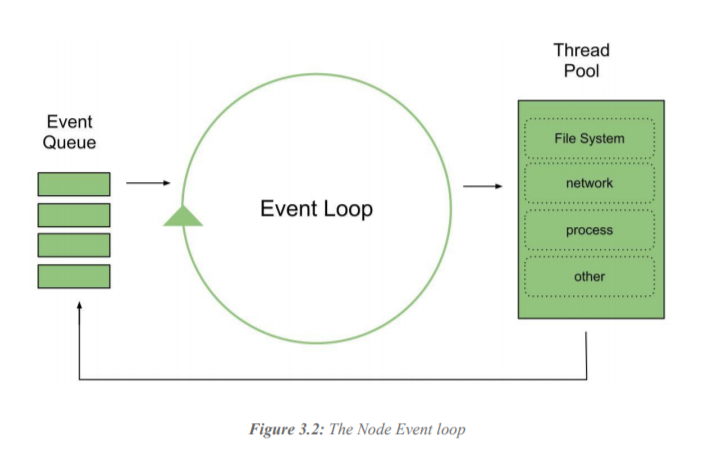
Node的标准库暴露了操作系统功能，例如处理文件系统，套接字，进程和定时器。标准库还具有事件，缓冲区和简单DNS和HTTP请求的功能。

Node使用'kqueue'，'select'，'epollto'等技术在新连接进入时从操作系统获取通知，然后将新事件分派给程序员以处理处理程序。



## 3.3 与Nodejs的并发编程

节点服务器进程，运行单线程，但可以同时为多个客户端提供服务。如图3.2所示，代码周围有一个隐式的主循环。没有实际的I/O，更不用说业务了逻辑处理，发生在循环体中。I/O相关事件触发实际处理，例如正在建立的连接，或从套接字，文件或外部系统发送或接收的字节。



随着节点开发人员可以轻松建立一个高性能，异步，事件驱动适度的资源要求网络服务器。程序的主要流程由显式调用的函数决定。这些函数永远不会阻塞I/O，而是注册适当的处理程序。没有显式的阻塞调用来调用事件循环。事件循环概念是Node的行为的核心，它隐藏在实现中。此外，使异步I/O成为默认情况，迫使开发人员从一开始就采用异步模型。这是Node与其他编程环境中使用异步I/O之间的主要区别之一。



清单3.3是简单HTTP服务器的稍微复杂的变体。它从HTTP请求中解析URI，并将URI的路径组件映射到服务器上的文件名。该文件以小块而不是一次性读取。在某些情况下，将调用为场景提供的函数作为回调。示例情况包括文件系统层何时准备将多个字节传递给应用程序，何时完全读取文件，或何时发生某种错误。

## 3.4 Nodejs生态系统：基本包

节点是更好的之一已知的支持服务器的框架和环境侧面JavaScript开发。社区已经为Node创建了一个完整的库生态系统，或者与Node兼容。

### 3.4.1 npm Nodejs的包管理器

npm是Node的包管理器，它通过命令行运行并管理应用程序的依赖关系。它是Node的主要包管理器。Node的模块可以作为源下载并手动组装使用。npm提供了一个更简单的替代方案，它是Node的标准包管理器，大大简化了这些模块的管理。

### 3.4.2 一些受欢迎的包裹概述

|  |  |
| --- | --- |
| Cluster | 它是一个可扩展的多Node的核心服务器管理器。它启动一组可配置的子进程，在发生崩溃时重新启动它们，并具有广泛的日志记录命令线路控制实用程序和统计信息 |
| Connect | Connect是节点的中间件框架。它包括超过18个必要的中间件和丰富的第3个中间件党中间件。一些重要的中间件是  ●记录器请求记录器，支持自定义格式  ●基本验证基本的http认证  ●jsonapplication/json解析器  ●会话会话管理支持。 |
| Express | Express是Node的Web应用程序框架。它提供了构建的框架，处理许多普通和无处不在的开发方面，以便专注于应用程序独有的功能。它内置了Connect中间件。 |
| Socket.IO | WebSocket是一种相对较新的技术，可实现双向，真实直接从客户端到服务器应用程序的时间通信，然后再返回。Socket.IO旨在让实时应用的移动，模糊了不同的传输机制之间的差异。 |
| Underscore | Underscore为Node提供实用程序功能。它提供了许多扩展程序JavaScript功能，可以在其他库中找到，例如jQuery。Underscore提供大约80个支持'map'，'select'，'invoke'的功能以及专业的帮助者。 |

## 3.5 Nodejs的优点

### 3.5.1 JavaScript

JSON（JavaScriptObjectNotation）是JavaScript中对象的本机表示，是通过AJAX发送数据时最常用的格式之一。

在前端和后端工作的程序员不必在两种语言之间切换。对于程序员而言，在不同任务或语言之间切换被认为是高成本并且可能降低生产率。

使用在服务器上运行的相同代码重用代码在双方语言相同的情况下，可以在浏览器中进行操作。

JavaScript附带了一些语言功能，使其适合编写事件驱动应用程序。第一个是闭包，它可以自动保存堆栈中的状态。JavaScript可以进行回调，因为它具有匿名功能

### 3.5.2 Nodejs

Node优于其他事件的优势之一驱动框架是Node有一个内置的事件循环。所有可用的模块和库都使用它。这使得在异步程序中处理多个部分变得更加容易。在事件中使用Node时驱动范式是自然的做事方式。只要作为程序员的程序员不做任何阻止性能和扩展的事情就会来自平台。

## 3.6 挑战

在撰写本文时，Node是一个年轻的平台，当前版本为v.8.8.14。因此，它在不断发展。部署是一个有许多替代方案的领域，但它们都没有经过充分测试。它在大型环境中是新的且未经测试的事实也可能是企业的风险因素。

在Node中，Web服务器和Web应用程序之间没有松耦合。松散耦合的架构已被证明是可维护的。

异步事件驱动编程对于许多开发人员来说是一个新概念，它可以用来习惯并获得充分的工作效率。调试变得困难，因为堆栈跟踪不再代表处理特定任务的控制流。

# 4. 结论

我们声称并发性对于可伸缩性至关重要，而可扩展性对于大型可扩展性本质上是至关重要的规模体系结构。网络流行的增长基于应用程序，因此需要双方可扩展的架构和相应的并发编程。

我们提供了用于并发的技术的详细概述，即线程和事件。我们通过引入一般体系结构和使用的相应模式来评估不同的方法。多使用线程的线程服务器每连接模型易于实施，遵循简单的策略。在重载下，多重线程Web服务器消耗大量内存（由于每个连接的单个线程堆栈），并且恒定的上下文切换相当大的CPU时间损失。事件，允许程序员通过将代码结构化为单一来明确地管理并发对事件作出反应的线程处理程序。事件驱动系统往往对负载很稳健，随着负载的增加，吞吐量几乎没有降低。

我们介绍了Node，这是一个基于GoogleV8引擎的平台，用于构建快速，可扩展的网络应用程序。它使用JavaScript编写服务器侧码，我们认为服务器JavaScriptJavaScript是一个合乎逻辑的步骤，可以在Web的所有方面使用单一编程语言基于分布式应用。我们讨论的事件驱动，非阻止I/O模型，使其轻量级和高效，是数据的理想选择密集的真实跨分布式设备运行的时间应用程序。Node不断发展，拥有一个庞大的社区，可以提供基本的包和模块。