Node.js 究竟是什么？

一个“编码就绪”服务器

Michael Abernethy, 自由程序员, Freelancer

简介： Node 是一个服务器端 JavaScript 解释器，它将改变服务器应该如何工作的概念。它的目标是帮助程序员构建高度可伸缩的应用程序，编写能够处理数万条同时连接到一个（只有一个）物理机的连接代码。

发布日期： 2011 年 6 月 13 日
级别： 中级
原创语言： 英文
访问情况 15210 次浏览
建议：

内容

简介

如果您听说过 Node，或者阅读过一些文章，宣称 Node 是多么多么的棒，那么您可能会想：“Node 究竟是什么东西？”尽管不是针对所有人的，但 Node 可能是某些人的正确选择。

为试图解释什么是 Node.js，本文探究了它能解决的问题，它如何工作，如何运行一个简单应用程序，最后，Node 何时是和何时不是一个好的解决方案。本文不涉及如何编写一个复杂的 Node 应用程序，也不是一份全面的 Node 教程。阅读本文应该有助于您决定是否应该学习 Node，以便将其用于您的业务。

Node 旨在解决什么问题？

Node 公开宣称的目标是“旨在提供一种简单的构建可伸缩网络程序的方法”。当前的服务器程序有什么问题？我们来做个数学题。在 Java™ 和 PHP 这类语言中，每个连接都会生成一个新线程，每个新线程可能需要 2 MB 配套内存。在一个拥有 8 GB RAM 的系统上，理论上最大的并发连接数量是 4,000 个用户。随着您的客户端基础的增长，您希望您的 web 应用程序支持更多用户，这样，您必须添加更多服务器。当然，这会增加业务成本，尤其是服务器成本、运输成本和人工成本。除这些成本上升外，还有一个技术问题：用户可能针对每个请求使用不同的服务器，因此，任何共享资源都必须在所有服务器之间共享。例如，在 Java 中，静态变量和缓存需要在每个服务器上的 JVMs 之间共享。这就是整个 web 应用程序架构中的瓶颈：一个服务器能够处理的并发连接的最大数量。

Node 解决这个问题的方法是：更改连接连接到服务器的方式。每个连接都创建一个进程，该进程不需要配套内存块，而不是为每个连接生成一个新的 OS 线程（并向其分配一些配套内存）。Node 声称它绝不会死锁，因为它根本不允许使用锁，它不会直接阻塞 I/O 调用。Node 还宣称，运行它的服务器能支持数万个并发连接。事实上，Node 通过将整个系统中的瓶颈从最大连接数量更改到单个系统的流量来改变服务器面貌。

现在您有了一个能处理数万条并发连接的程序，那么您能通过 Node 实际构建什么呢？如果您有一个 web 应用程序需要处理这么多连接，那将是一件很“恐怖”的事！那是一种“如果您有这个问题，那么它根本不是问题”的问题。在回答上面的问题之前，我们先看看 Node 如何工作以及它被设计的如何运行。

回页首

Node 肯定不是什么

没错，Node 是一个服务器程序。但是，它肯定不 像 Apache 或 Tomcat。那些服务器是独立服务器产品，可以立即安装并部署应用程序。通过这些产品，您可以在一分钟内启动并运行一个服务器。Node 肯定不是这种产品。Apache 能添加一个 PHP 模块来允许开发人员创建动态 web 页，使用 Tomcat 的程序员能部署 JSPs 来创建动态 web 页。Node 肯定不是这种类型。

在 Node 的早期阶段（当前是 version 0.4.6），它还不是一个“运行就绪”的服务器程序，您还不能安装它，向其中放置文件，拥有一个功能齐全的 web 服务器。即使是要实现 web 服务器在安装完成后启动并运行这个基本功能，也还需要做大量工作。

回页首

Node 如何工作

Node 本身运行 V8 JavaScript。等等，服务器上的 JavaScript？没错，您没有看错。服务器端 JavaScript 是一个相对较新的概念，这个概念是大约两年前在 developerWorks 上讨论 Aptana Jaxer 产品时提到的（参见 参考资料）。尽管 Jaxer 一直没有真正流行，但这个理念本身并不是遥不可及的 — 为何不能在服务器上使用客户机上使用的编程语言？

什么使 V8？V8 JavaScript 引擎是 Google 用于他们的 Chrome 浏览器的底层 JavaScript 引擎。很少有人考虑 JavaScript 在客户机上实际做了些什么？实际上，JavaScript 引擎负责解释并执行代码。使用 V8，Google 创建了一个以 C++ 编写的超快解释器，该解释器拥有另一个独特特征；您可以下载该引擎并将其嵌入任何 应用程序。它不仅限于在一个浏览器中运行。因此，Node 实际上使用

Google 编写的 V8 JavaScript 引擎并将其重建为在服务器上使用。太完美了!既然已经有一个不错的解决方案可用,为何还要创建一种新语言呢?

事件驱动编程

许多程序员接受的教育使他们认为,面向对象编程是完美的编程设计,而对其他编程方法不屑一顾。Node 使用一个所谓的事件驱动编程模型。

清单 1. 客户端上使用 jQuery 的事件驱动编程

```
// jQuery code on the client-side showing how Event-Driven programming works

// When a button is pressed, an Event occurs - deal with it
// directly right here in an anonymous function, where all the
// necessary variables are present and can be referenced directly
$("#myButton").click(function(){
     if ($("#myTextField").val() != $(this).val())
         alert("Field must match button text");
});
```

实际上,服务器端和客户端没有任何区别。没错,这没有按钮点击操作,也没有向文本字段键入的操作,但在一个更高的层面上,事件正在 发生。一个连接被建立 — 事件!数据通过连接接收 — 事件!数据通过连接停止 — 事件!

为什么这种设置类型对 Node 很理想?JavaScript 是一种很棒的事件驱动编程语言,因为它允许匿名函数和闭包,更重要的是,任何写过代码的人都熟悉它的语法。事件发生时调用的回调函数可以在捕获事件处编写。这样,代码容易编写和维护,没有复杂的面向对象框架,没有接口,没有在上面架构任何内容的潜能。只需监听事件,编写一个回调函数,然后,事件驱动编程将照管好一切!

回页首

示例 Node 应用程序

最后,我们来看一些代码!让我们将讨论过的所有内容综合起来,创建我们的第一个 Node 应用程序。由于我们已经知道,Node 对于处理高流量应用程序很理想,我们就来创建一个非常简单的 web 应用程序 — 一个为实现最大速度而构建的应用程序。下面是 “老板” 交代的关于我们的样例应用程序的具体要求:创建一个随机数字生成器 RESTful API。这个应用程序应该接受一个输入:一个名为 “number” 的参数。然后,应用程序返回一个介于 0 和该参数之间的随机数字,并将生成的数字返回调用者。由于 “老板” 希望它成为一个广泛流行的应用程序,因此它应该能处理 50,000 个并发用户。我们来看看代码:

清单 2. Node 随机数字生成器

```
// these modules need to be imported in order to use them.
// Node has several modules.  They are like any #include
// or import statement in other languages
var http = require("http");
var url = require("url");

// The most important line in any Node file.  This function
// does the actual process of creating the server.  Technically,
// Node tells the underlying operating system that whenever a
// connection is made, this particular callback function should be
// executed.  Since we're creating a web service with REST API,
// we want an HTTP server, which requires the http variable
// we created in the lines above.
// Finally, you can see that the callback method receives a 'request'
// and 'response' object automatically.  This should be familiar
// to any PHP or Java programmer.
http.createServer(function(request, response) {

     // The response needs to handle all the headers, and the return codes
     // These types of things are handled automatically in server programs
     // like Apache and Tomcat, but Node requires everything to be done yourself
     response.writeHead(200, {"Content-Type": "text/plain"});

     // Here is some unique-looking code.  This is how Node retrieves
     // parameters passed in from client requests.  The url module
     // handles all these functions.  The parse function
     // deconstructs the URL, and places the query key-values in the
     // query object.  We can find the value for the "number" key
     // by referencing it directly - the beauty of JavaScript.
     var params = url.parse(request.url, true).query;
     var input = param.number;

     // These are the generic JavaScript methods that will create
     // our random number that gets passed back to the caller
     var numInput = new Number(input);
     var numOutput = new Number(Math.random() * numInput).toFixed(0);

     // Write the random number to response
     response.write(numOutput);
```

```
    // Node requires us to explicitly end this connection.  This is because
    // Node allows you to keep a connection open and pass data back and forth,
    // though that advanced topic isn't discussed in this article.
    response.end();

  // When we create the server, we have to explicitly connect the HTTP server to
  // a port.  Standard HTTP port is 80, so we'll connect it to that one.
}).listen(80);

// Output a String to the console once the server starts up, letting us know everything
// starts up correctly
console.log("Random Number Generator Running...");
```

启动应用程序

将上面的代码放到一个名为 “random.js” 的文件中。现在，要启动这个应用程序并运行它（进而创建 HTTP 服务器并监听端口 80 上的连接），只需在您的命令提示中输入以下命令：`% node random.js`。下面是服务器已经启动并运行时它看起来的样子：

```
root@ubuntu:/home/moilanen/ws/mike# node random.js
Random Number Generator Running...
```

访问应用程序

应用程序已经启动并运行。Node 正在监听任何连接，我们来测试一下。由于我们创建了一个简单的 RESTful API，我们可以使用我们的 web 浏览器来访问这个应用程序。键入以下地址（确保您完成了上面的步骤）：http://localhost/?number=27。

您的浏览器窗口将更改到一个介于 0 到 27 之间的随机数字。单击浏览器上的 “重新载入” 按钮，将得到另一个随机数字。就是这样，这就是您的第一个 Node 应用程序!

回页首

Node 对什么有好处?

到此为止，应该能够回答 “Node 是什么” 这个问题了，但您可能还不清楚什么时候应该使用它。这是一个需要提出的重要问题，因为 Node 对有一些东西有好处，但相反，对另一些东西而言，目前 Node 可能不是一个好的解决方案。您需要小心决定何时使用 Node，因为在错误的情况下使用它可能会导致一个多余编码的 LOT。

它对什么有好处?

正如您此前所看到的，Node 非常适合以下情况：您预计可能有很高的流量，而在响应客户端之前服务器端逻辑和处理所需不一定是巨大的。Node 表现出众的典型示例包括：

- RESTful API
  提供 RESTful API 的 web 服务接收几个参数，解析它们，组合一个响应，并返回一个响应（通常是较少的文本）给用户。这是适合 Node 的理想情况，因为您可以构建它来处理数万条连接。它还不需要大量逻辑；它只是从一个数据库查找一些值并组合一个响应。由于响应是少量文本，入站请求时少量文本，因此流量不高，一台机器甚至也可以处理最繁忙的公司的 API 需求。
- Twitter 队列
  想像一下像 Twitter 这样的公司，它必须接收 tweets 并将其写入一个数据库。实际上，每秒几乎有数千条 tweets 达到，数据库不可能及时处理高峰时段需要的写入数量。Node 成为这个问题的解决方案的重要一环。如您所见，Node 能处理数万条入站 tweets。它能迅速轻松地将它们写入一个内存排队机制（例如 memcached），另一个单独进程可以从那里将它们写入数据库。Node 在这里的角色是迅速收集 tweet 并将这个信息传递给另一个负责写入的进程。想象一下另一种设计 — 一个常规 PHP 服务器自己试图处理对数据库的写入 — 每个 tweet 将在写入数据库时导致一个短暂的延迟，这是因为数据库调用正在阻塞通道。由于数据库延迟，一台这样设计的机器每秒可能只能处理 2000 条入站 tweets。 每秒 100 万条 tweets 需要 500 个服务器。相反，Node 能处理每个连接而不会阻塞通道，从而能捕获尽可能多的 tweets。一个能处理 50,000 条 tweets 的 Node 机器只需要 20 个服务器。
- 映像文件服务器
  一个拥有大型分布式网站的公司（比如 Facebook 或 Flickr）可能会决定将所有机器只用于服务映像。Node 将是这个问题的一个不错的解决方案，因为该公司能使用它编写一个简单的文件检索器，然后处理数万条连接。Node 将查找映像文件，返回文件或一个 404 错误，然后什么也不用做。这种设置将允许这类分布式网站减少它们服务映像、.js 和 .css 文件等静态文件所需的服务器数量。

它对什么有坏处?

当然，在某些情况下，Node 并非理想选择。下面是 Node 不擅长的领域：

- 动态创建的页
  目前，Node 没有提供一种默认方法来创建动态页。例如，使用 JavaServer Pages (JSP) 技术时，可以创建一个在 `<% for (int i=0; i<20; i++) { } %>` 这样的 JSP 代码段中包含循环的 index.jsp 页。Node 不支持这类动态的、HTML 驱动的页面。同样，Node 不太适合作为 Apache 和 Tomcat 这样的网页服务器。因此，如果您想在 Node 中提供这样一个服务器端解决方案，必须自己编写整个解决方案。PHP 程序员不想在每次部署 web 应用程序时都编写一个针对 Apache 的 PHP 转换器，当目前为止，这正是 Node 要求您做的。
- 关系数据库重型应用程序

Node 的目的是快速、异步和非阻塞。数据库并不一定分享这些目标。它们是同步和阻塞的，因为读写时对数据库的调用在结果生成之前将一直阻塞通道。因此，一个每个请求都需要大量数据库调用、大量读取、大量写入的 web 应用程序非常不适合 Node，这是因为关系数据库本身就能抵销 Node 的众多优势。（新的 NoSQL 数据库更适合 Node，不过那完全是另一个主题了。）

回页首

结束语

问题是 “什么是 Node.js?” 应该已经得到解答。阅读本文之后，您应该能通过几个清晰简洁的句子回答这个问题。如果这样，那么您已经走到了许多编码员和程序员的前面。我和许多人都谈论过 Node，但它们对 Node 究竟是什么一直很迷惑。可以理解，他们具有的是 Apache 的思维方式 — 服务器是一个应用程序，将 HTML 文件放入其中，一切就会正常运转。而 Node 是目的驱动的。它是一个软件程序，使用 JavaScript 来允许程序员轻松快速地创建快速、可伸缩的 web 服务器。Apache 是运行就绪的，而 Node 是编码就绪的。

Node 完成了它提供高度可伸缩服务器的目标。它并不分配一个 “每个连接一个线程” 模型，而是使用一个 “每个连接一个流程” 模型，只创建每个连接需要的内存。它使用 Google 的一个非常快速的 JavaScript 引擎：V8 引擎。它使用一个事件驱动设计来保持代码最小且易于阅读。所有这些因素促成了 Node 的理想目标 — 编写一个高度可伸缩的解决方案变得比较容易。

与理解 Node 是 什么同样重要的是，理解它不是 什么。Node 并不是 Apache 的一个替代品，后者旨在使 PHP web 应用程序更容易伸缩。事实确实如此。在 Node 的这个初始阶段，大量程序员使用它的可能性不大，但在它能发挥作用的场景中，它的表现非常好。

将来应该期望从 Node 得到什么呢？这也许是本文引出的最重要的问题。既然您知道了它现在的作用，您应该会想知道它下一步将做什么。在接下来的一年中，我期待着 Node 提供与现有的第三方支持库更好地集成。现在，许多第三方程序员已经研发了用于 Node 的插件，包括添加文件服务器支持和 MySQL 支持。希望 Node 开始将它们集成到其核心功能中。最后，我还希望 Node 支持某种动态页面模块，这样，您就可以在 HTML 文件中执行在 PHP 和 JSP（也许是一个 NSP，一个 Node 服务器页）中所做的操作。最后，希望有一天会出现一个 “部署就绪” 的 Node 服务器，可以下载和安装，只需将您的 HTML 文件放到其中，就像使用 Apache 或 Tomcat 那样。Node 现在还处于初始阶段，但它发展得很快，可能不久就会出现在您的视野中。

参考资料

学习

- Node.js 主页 是了解这个应用程序的切入点。
- 浏览 Node.js API 页。注意，不同发布的语法可能不同，因此，请根据您正在浏览的 API 检查您已下载的版本。
- 了解 Jaxer，这是创建一个服务器端 JavaScript 环境的首次重要尝试。
- 阅读 Michael Galpin 撰写的这篇 developerWorks 文章，了解 现实中的开放源码云计算，第 1 部分：并不是所有云都相同。
- 查看免费的 developerWorks 演示中心，观看并了解 IBM 及开源技术和产品功能。
- 随时关注 developerWorks 技术活动和网络广播。
- 访问 developerWorks Open source 专区获得丰富的 how-to 信息、工具和项目更新以及最受欢迎的文章和教程，帮助您用开放源码技术进行开发，并将它们与 IBM 产品结合使用。

获得产品和技术

- 下载 Node.js 和 Python，后者您也会用到。
- 参见 Wikipedia 上不断变化的 开源软件包列表。
- 使用 IBM 产品评估试用版软件 改进您的下一个开源开发项目，这些软件可以通过下载 获得。

讨论

- 参与讨论论坛。
- 帮助构建 developerWorks 社区中的 真实世界开源 讨论组。
- 加入 developerWorks 中文社区，developerWorks 社区是一个面向全球 IT 专业人员，可以提供博客、书签、wiki、群组、联系、共享和协作等社区功能的专业社交网络社区。
- 加入 IBM 软件下载与技术交流群组，参与在线交流。

关于作者

在 Michael Abernethy 的 13 年技术生涯中，他与各种不同的技术和客户打交道。他目前是一名自由程序员，擅长 Java 高可用性和 jQuery。他现在专注于富 Internet 应用程序，试图同时实现应用程序的复杂性和简单性。他空闲时常常去打高尔夫球，更确切地说，是在灌木丛中寻找他打飞的高尔夫球。

建议

回页首

打印此页面　分享此页面　关注 developerWorks

技术主题
AIX and UNIX
IBM i
Information Management
Lotus
Rational
WebSphere
Cloud computing
Java technology
Linux
Open source
SOA and web services
Web development
XML
更多...

查找软件
IBM 产品
评估方式（下载，在线试用，Beta 版，云）
行业
技术讲座

社区
群组
博客
Wiki
文件
使用条款
报告滥用
更多...

关于 developerWorks
反馈意见
在线投稿
投稿指南
网站导航
请求转载内容
相关资源
ISV 资源 (英语)
IBM 教育学院教育培养计划

IBM
解决方案
软件
支持门户
产品文档
红皮书 (英语)
隐私条约
浏览辅助