Node.js：一种新的Web应用构建技术

王金龙[1]，宋　斌[1]，丁　锐[2]
（1.南京理工大学，江苏 南京　210094；2.中国铁通集团有限公司 泰州分公司，江苏 泰州　225300）

摘　要：现如今，有很多种Web应用程序开发语言。在Web应用程序开发过程中，大部分语言都要解决多线程问题。而且这些Web应用程序都要部署在第三方Web服务器上，如：Apache，Tomcat，Nginx等。近期一种基于I/O事件驱动模型服务器端的JavaScript运行环境——Node.js得到了广泛的关注和应用。通过对同步阻塞语言PHP和异步非阻塞的Node.js构建的Web应用程序做一些性能上的比较，发现在高并发请求的情况下，Node.js构建的服务器比PHP构建的应用程序的响应时间短、吞吐率高。最终得出结论Node.js在构建快速、可扩展的Web应用程序方面的优势大于PHP。

关键词：Web服务器；异步非阻塞；事件驱动模型；Node.js
中图分类号：TN911-34　　**文献标识码：**A　　**文章编号：**1004-373X(2015)06-0070-04

Node.js: a new technology to build Web application

WANG Jin-long[1], SONG Bin[1], DING Rui[2]
(1. Nanjing University of Science &Technology, Nanjing 210094, China; 2. China Tietong, Taizhou 225300, China)

Abstract: Nowadays, there are many program languages to build Web application. During developing Web application, the most of languages must handle multithreading problem. In addition, these Web application programs must be deployed in a third-party Web server, such as Apache, Tomcat, Nginx and so on. Recently, Node.js, an event-driven server-side JavaScript environment based on I/O has been widely concerned and applied. In this paper, via comparing the performance of the Web application built by synchronous blocking PHP and asynchronous non-blocking Node.js, it is found that the response time of the Web server built by Node.js is shorter than PHP and the throughput of Node.js is higher than PHP in the case of high concurrent requests. A conclusion that Node.js is superior to PHP in building fast and scalable Web application program is obtained eventually.

Keywords: Web server; asynchronous non-blocking; event-driven model; Node.js

0　引　言

早期，PHP这门服务器端脚本语言一直受到Web应用程序的开发者们青睐，然而JavaScript一直被人们认为是前端的脚本开发语言，随着Node.js的出现，JavaScript得到了大家更多的关注。Node.js是建立在Chrome的JavaScript运行时之上的平台，它用于构建快速、可扩展的网络应用程序。Node.js使用一种事件驱动、非阻塞的I/O模型，这也使得跨分布式设备的数据密集型实时应用更加轻量、高效和完美[1]。

目前，国内外很多大公司都在将他们的部分产品的技术栈向Node.js转变。国外的有，知名团购网站Groupon将其站点从Ruby on Rails全面迁移到了Node.js. Groupon团队的开发人员表示，Rails非常适合小型团队的快速开发，可以让网站快速启动并运行起来，这对于初期功能不断变化的Groupon来说，是个不错的选择。但是，随着Groupon的发展和新产品的不断推出，这个代码库越来越大，有太多的开发者在同一个代码库工作，他们很难在本地运行并测试产品。因此，Groupon团队评估了不同的软件栈，想寻找一个能够解决这些问题的方案，有效处理大量传入的HTTP请求、使并行API请求服务于每一个HTTP请求、将结果渲染为HTML5，并可以有效实现监控、部署和支持。该团队使用不同的技术栈开发了原型，并做了测试，最终发现Node.js是个非常适合的解决方案。迁移之后，Groupon成为全球最大的Node.js部署产品之一，也为之带来下列好处：页面加载比之前快了50%；与之前相比，处理相同的流量所使用的硬件资源更少；团队可以独立地更改、部署各自负责的模块；网站功能和设计实现可以快速迭代。鉴于性能和可扩展性方面的原因，LinkedIn将其移动设施的后台Ruby on Rails替换成Node.js。

国内的Node.js应用主要有，淘宝的数据平台、网易

收稿日期：2014-09-24

开源的pomelo实时框架和有道词典等。MyFox是淘宝一个针对海量统计数据设计的高性能分布式MySQL集群中间层,负责从其中提取数据、计算并输出统计结果。起初MyFox使用PHP编写,但是遇到了很多问题。如PHP是单线程的,MySQL又需要阻塞查询,因此很难并发请求数据,最终项目组决定使用Node.js实现MyFox。pomelo是基于Node.js的高性能、分布式游戏服务器框架。它包括基础的开发框架和相关的扩展组件。网易选择Node.js的原因是,Node.js天生就是做多进程开发的,多个节点互相通讯交织在一起组成了分布式系统;单线程的应用模型处理游戏逻辑是最简单,最不容易出错的,而且不可能出现死锁、锁竞争的情况;游戏是非常I/O密集型的应用,而Node.js生来就是为I/O而生的。

1 性能比较

Node.js是Web服务器技术的新宠儿。与此同时,PHP这个传统的Web应用程序使用的语言,自诞生以来,褒贬不一。大家可能会说对语言的评论是没有意义的,但是有些评论的确是得到权衡的。与PHP相比,你不必使用一个独立的HTTP服务器,把Node.js应用部署在Nginx下也是十分常见的,但不是必须的。因此,一个典型的Web应用程序的核心就是一个Web服务器的实现。比较Node.js和PHP。其实,真正比较的是Node.js和PHP+Apache2(或者其他的HTTP服务器)。所以为了论证Node.js构建的Web应用程序在高并发请求下的性能优势,本文的比较实验用的是Apache2和mod_php,因为它们目前是最流行的配置。

1.1 测试方案

为了使比较更加的有理可据,实验用PHP5和Node.js创建了一个简单的Web应用程序。为了对两种架构的I/O性能做比较,该应用程序从测试数据库的用户表中读出前50行数据,并且以JSON字符串的形式输出到页面。并通过开源的服务器性能测试工具Siege对两者分别进行50,100,150,200,250和300个并发情况下的压力测试。保持应用程序简单的好处一是不必过问两种语言的实现细节,更重要的是,并非测试代码的能力,真正测试的是两种结构的差别。实验服务器和实验环境配置如下:

硬件环境:Intel Core i5-3337U CPU@1.80 GHz,4 GB RAM。

软件环境:Ubuntu 12.10 x64,Apache 2.2.22,PHP 5.4.6,Node.js 0.10.28,MySQL 5.5.37、服务器性能测试工具Siege。

Node.js应用程序代码如下:

```
var http = require('http'),
mysql = require('mysql');
var connection = mysql.createConnection({
 host: 'localhost',
 user: 'root',
 password: 'root',
 database: 'test'
});
http.createServer(function(req, res){
 connection.query('select * from test_table limit 50', function
(err, rows, field){
        if(err) return err;
        res.end(JSON.stringify(rows));
});
}).listen(8090);
```

PHP代码如下:

```
<?php
$db = new PDO('mysql:host=localhost;dbname=test', 'root',
'root');
$rows =$db->query('select * from test_table limit 50;')->
fetchAll();
echo json_encode($rows);
?>
```

从上面的代码可以看出,Node.js是通过回调函数(即函数作为参数)实现异步操作,而PHP是完全的同步编程风格。而且,两段代码都能实现上述的功能。虽然PHP脚本明显比Node.js短,但是,PHP不必实现一个完整的HTTP服务器,而Node.js只需要一句代码(即http.createServer)就可以创建一个HTTP服务器。

1.2 测试结果及分析

图1显示在不同并发数下,Node.js构建的服务器每秒钟处理的事务数明显高于PHP应用。图2显示Node.js构建的服务器的吞吐率维持在0.7 MB/s上下波动,而部署在Apache2服务器上PHP应用的吞吐率则维持在0.2 MB/s上下波动。

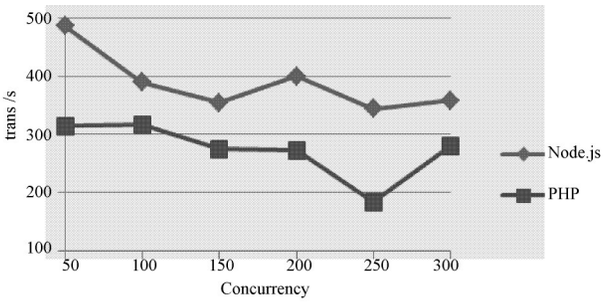

图1 不同并发数下的服务器每秒处理的事务数

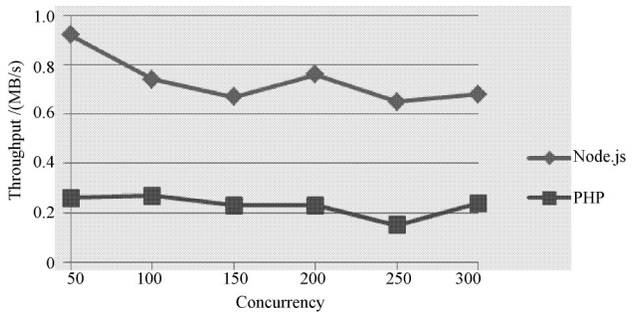

图2 不同并发数下的服务器的吞吐率

1.2.1 多线程与单进程

实际上,上面的实验结果并不奇怪,只是表明了两种解决方案之间的架构差异。PHP本身不是多线程的,但是Apache2支持多线程,一个请求对应一个Apache线程。每个请求都有一个独立的PHP线程运行,请求之间的PHP环境相互独立,互不影响。此外,每次启动一个线程,都会占用一定大小的内存,CPU上下文环境的切换也会带来一定的开销。

然而,Node.js构建的Web服务器是单进程的,一个进程中保持一个活跃的请求处理线程。所以内部没有不同请求实例与父进程之间通信这种情况。相比PHP/Apache,Node.js的内存使用更加高效,Apache的每个并发请求都会占用一定的内存,但是,Node.js不同请求都是共享内存的。

由于Apache服务器上每个请求都是相互独立,而服务器的内存又是有限的,所以其必须在并发请求数和内存之间做一个权衡。然而,Node.js不同请求共享内存的优势,因此,Node.js构建的服务器每秒处理的请求数(即RPS)会高于Apache服务器,而RPS最终又会反应到服务器的吞吐率,如图2所示,Node.js构建的服务器的吞吐率明显高过PHP。

1.2.2 同步与异步

PHP语言是以同步阻塞的方式执行的。它的优点十分明显,利于程序员顺序编写业务逻辑;它的缺点在小规模站点中基本不存在,但是在复杂的网络应用中,阻塞导致它无法更好的并发。

然而Node.js是异步I/O的,Web应用已经不再是单台服务器就能胜任的时代了,在跨网络的结构下,并发已经是现代编程中的标准配备了[2]。具体到实处,则可以从用户体验和资源分配这两个方面说起。

在用户体验方面,Node.js的实质是运行在Chrome v8引擎上的JavaScript。在浏览器中JavaScript在单线程上执行,而且还与UI渲染共用一个线程。这意味着JavaScript在执行的时候UI渲染和响应是处于停滞状态的。如果网页获取一个网络资源,通过同步的方式获取,那么JavaScript则需要等待资源完全从服务器端获取后才能继续执行,这期间UI将停顿,不响应用户的交互行为。然而,如果采用异步请求,在下载资源期间,JavaScript和UI的执行都不会处于等待状态。但是,前端获取资源的速度也取决于后端的响应速度,所以Node.js中资源的I/O都是异步[3]。

从资源分配方面分析,在计算机资源中,通常I/O与CPU计算之间是可以并行进行的。但是同步的编程模型导致的问题是,I/O的进行会让后续的任务等待,这造成资源不能被更好的利用。从实验结果图1可以看出,PHP的同步阻塞I/O问题导致其每秒处理的事务数明显比异步非阻塞的Node.js低很多。而且,单线程同步编程模型会因阻塞I/O导致硬件资源得不到更优的使用,多线程编程模型也因为编程中的死锁、状态同步等问题让开发人员头疼。Node.js在两者之间给出了它的方案:利用单线程,远离多线程死锁、状态同步等问题;利用异步I/O,让单线程远离阻塞,以更好的使用CPU。

1.2.3 测试结论

通过上述实验,可以看出Node.js构建的单进程、异步非阻塞I/O的Web服务器在处理高并发请求上的优势。面对高并发的用户请求,Node.js构建的Web服务器能作出更快速的响应,而且能有效利用服务器的硬件资源。

2 Node.js机制及适用领域

以上分析了Node.js通过异步非阻塞I/O高效的处理并发请求的优势。但是Node.js如何在单线程的请求的局限下,处理并发请求。下面,本文将介绍Node.js处理并发请求机制及其适用的领域。

2.1 Node.js事件驱动模型

Node.js构建的服务器是通过事件驱动模型来处理并发请求的。在Node.js中,所有的磁盘I/O都对应一个事件,Node.js内部有一个事件循环进程,一直轮询是否有事件发生。如果某个事件发生了,就会执行相应事件的事件处理函数(又称回调函数)。用流程图表示,如图3所示。

2.2 Node.js编程模型

Node.js的I/O方法是严格的:异步的交互是它的规则。每个I/O操作都是通过高度嵌套的函数(即一个函数作为另一个函数的参数)处理的。在极少数的情况下,Node.js开发人员才会用到同步执行的函数。例如,为了删除或者重命名文件,如果这个操作可能需要网络或文件的I/O调用,控制逻辑会立即返回给调用者。但是,当有一些情况发生时,例如,如果数据变为可用于从网络Socket读取,输出流准备好写操作或者有错误发生

时,回调函数将被调用[4]。也正是由于Node.js异步回调的编程风格,导致开发过程中到处都是callback,代码不优雅,但是因为Node.js社区的活跃,出现了许多第三方的模块来解决“回调陷阱”[5]问题。相比于Node.js,PHP是同步阻塞的编程模型,代码逻辑更容易理解。

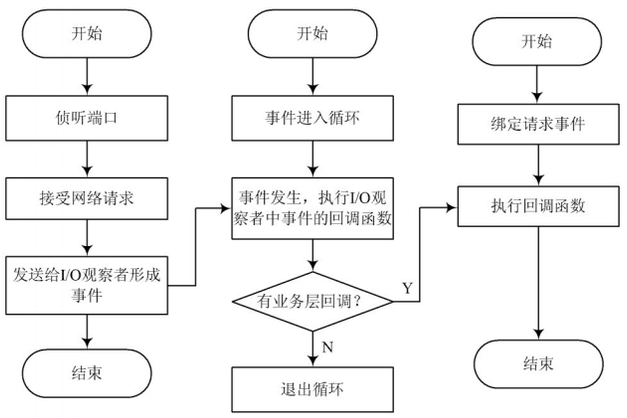

图3 事件驱动模型流程图

Node.js本身就支持构建HTTP服务器,而不需要借助第三方的Web服务器,所以用Node.js开发的Web应用程序部署非常方便、高效。

2.3 Node.js应用领域

正如本文上面提到的,Node.js非常适合以下情况:在响应客户端之前,预计可能有很高的流量,但所需的服务器端逻辑和处理不一定很多,Node.js发挥优势的典型示例如下:提供RESTful API的Web服务接收几个参数,解析它们,组合一个响应,并返回一个响应给用户。这是适合Node.js的理想情况,因为可以构建它来处理数万条连接,但是不需要处理大量逻辑,它本质上只是从某个数据库中查找一些值并将它们组成一个响应。由于响应是少量文本,请求也是少量的文本,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[6]。在游戏数据统计方面,Node.js也发挥了很好的优势。当生成很高级别的统计数据时,必须跟踪海量信息,如果有数百万玩家同时在线玩游戏,而且处于游戏中的不同位置,为了快速生成海量信息,Node.js是这种场景的一种很好的解决方案,因为它能采集游戏生成的数据,对数据进行最少的合并,然后对数据进行排队,以便将它们写入数据库。

但是,Node.js不适用在CPU密集型的应用,例如CPU使用率高而I/O操作少的情况[7]。因为Node.js是单线程的,对于多核CPU的服务器,Node.js不能有效地利用所有的核心。

3 结 语

本文通过对Node.js构建的Web服务器和PHP构建的应用程序做了性能上的比较,最终发现,异步非阻塞的Node.js构建的Web服务器在处理高并发请求方面的优势,但是它不适用于CPU密集型的应用。总的来说,Node.js完成了它提供快速可扩展服务器目标。Node.js使用了Google的一个非常快速的JavaScript引擎,即v8引擎[8]。同时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.js的理想目标,即编写一个快速可扩展的解决方案变得比较容易。与理解Node.js是什么同样重要的是,理解它不是什么。Node.js并不只是Apache的一个替代品,它旨在使Web应用程序更容易扩展。事实远非如此,尽管Node还处于初始阶段,但它发展得非常迅速,社区参与度非常高,社区成员创建了大量优秀模块,一年之内,这个不断发展的产品就有可能出现在您的企业中。

参 考 文 献

[1] MCLAUGHLIN Brett. What is node? [M]. California: O'Reilly Media, 2011.

[2] 朴灵.深入浅出Node.js[M].北京:人民邮电出版社,2013.

[3] TILKOY Stefan, VINOSKI Steve. Node.js: using JavaScript to build high-performance network programs [J]. IEEE Internet Computing, 2010, 14(6): 80-83.

[4] 赵昆.改变Web开发格局的新技术node.js[J].程序员,2011(7):124-125.

[5] CANTELON Mike, HOLOWAYCHUK T J. Node.js in action [M]. America: Manning Publications, 2013.

[6] RAUCH Guillermo. Smashing Node.js [M]. America: Wiley, 2012.

[7] LOUKIDES Mike. New directions in web architecture [EB/OL]. [2010-11-16]. http://radar.oreilly.com/2010/11/new-directions-in-web-architec.html.

[8] VOID B Y. Node.js开发指南[M].北京:人民邮电出版社,2012.

作者简介:王金龙(1989—),男,江苏淮安人,硕士研究生。研究方向为计算机应用技术。