关于NodeJS的那些事

by 齐飞

UED分享・交流
http://cssrain.github.io

AsiaInfo 亚信 | UED分享 · 交流

NodeJS是什么？

AsiaInfo 亚信 | UED分享 · 交流


node JS
Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network
applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and
efficient, perfect for data-intensive real-time applications that run across distributed devices.
Current Version: v0.10.26
INSTALL
DOWNLOADS
API DOCS
AsiaInfo 亚信 | UED分享 · 交流

- 让JavaScript运行在服务器端
- 基于Google Chrome的V8引擎（高效）
- 单线程（模型简单）
- 异步式I/O（高效）事件驱动（编程思维）

NodeJS能做什么？

AsiaInfo 亚信 | UED分享 · 交流

- 大型网站(Paypal, LinkedIn, Walmart)
- Web中间层(Taobao)
- 提供Restful服务
- 命令行程序(Grunt、nico、SPM)
- 图形化应用(GitHub AtomEditor)
- 编译器(Uglify, Less)
- 操作系统(NodeOS)

NodeJS最大的亮点

异步式I/O ＆ 事件驱动

AsiaInfo 亚信 | UED分享 · 交流

传统的I/O是同步的

```
var result =
  db.query("select * from T");
// use result
```

想一想？　等待数据库响应的时候，程序在做什么？

大多数时候，我们的程序仅仅是在等待I/O完成。

I/O操作的代价

I/O latency

L1: 3 cycles

L2: 14 cycles

RAM: 250 cycles

DISK: 41,000,000 cycles

NETWORK: 240,000,000 cycles

多线程也可以解决I/O阻塞

对于阻塞式的I/O操作，开启独立线程在后台运行。

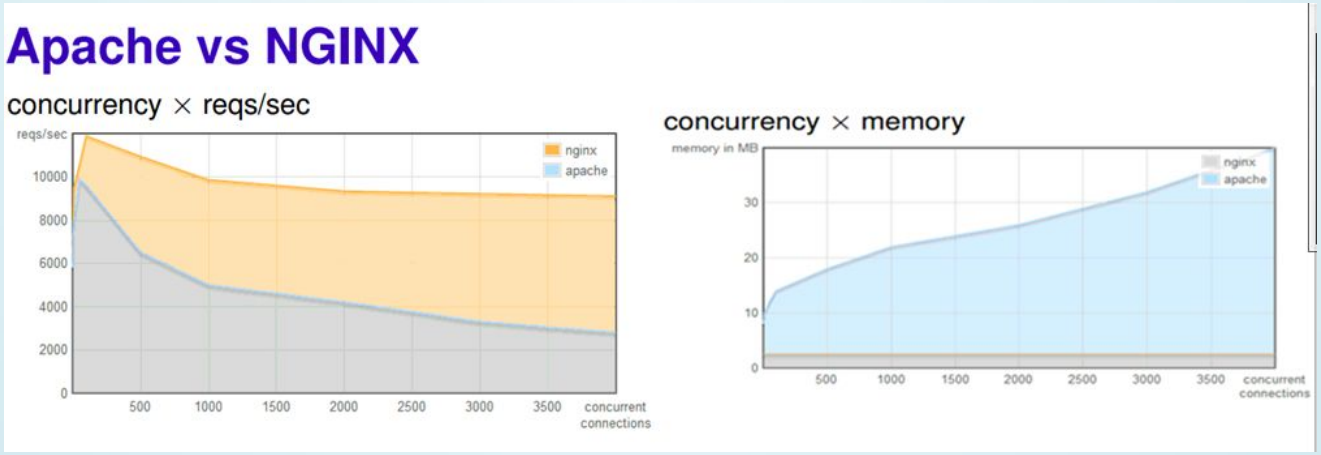
多线程是昂贵的

AsiaInfo 亚信 | UED分享 · 交流

Apache vs Nginx

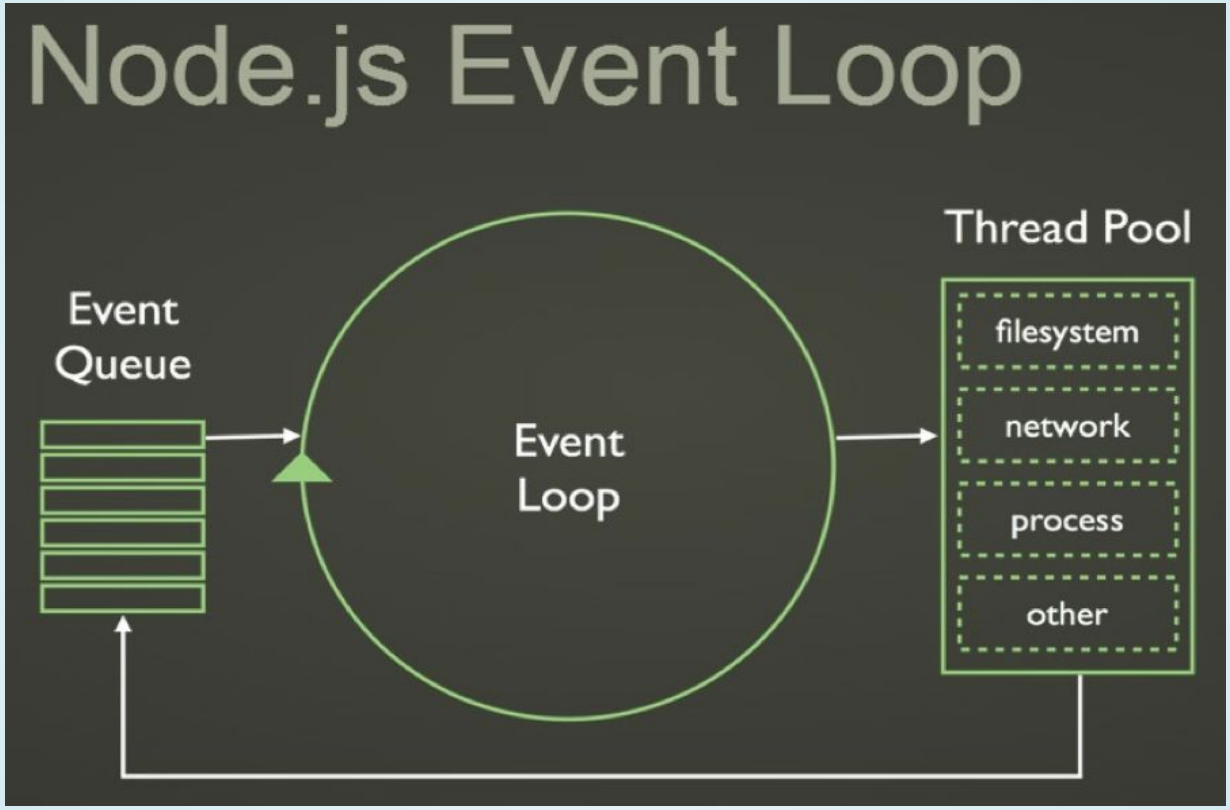
Apache - 多线程，为每个连接开一个线程

Nginx - 单线程，异步、事件循环



AsiaInfo 亚信 | UED分享 · 交流

NodeJS事件循环机制



异步式I/O

```
db.query("select..", function (result) {
  // use result
});
```

- 发起I/O操作后，允许主线程立即返回事件循环。
- 当I/O操作技术后，在回调函数中处理结果

异步式I/O是实现高性能Web运用的关键

事件循环机制是实现异步式I/O的基础

AsiaInfo 亚信 | UED分享 · 交流

NodeJS是模块化的

AsiaInfo 亚信 | UED分享 · 交流

- 模块化是使用JS开发大型程序的基础

- CommonJS 是为了统一服务器端JS API而诞生的规范，它包含了modules, packages, encodings, filesystems, sockets等多个部分

- NodeJS遵循CommonJS中的Modules模块化标准

NodeJS核心模块

AsiaInfo 亚信 | UED分享 · 交流

- global - 全局对象
- fs - 文件系统
- util - 常用工具
- events - 事件驱动
- http - http服务器模块

JavaScript高级特性

AsiaInfo 亚信 | UED分享 · 交流

NodeJS的编程中会涉及到的JS语法特性：

作用域、闭包、对象、原型 ...

AsiaInfo 亚信 | UED分享 · 交流

例如，变量作用域带来的陷阱 ...

```
// 变量提升

var scope = 'global';

var f = function() {
  console.log(scope);
  // undefined
  var scope = 'f';
};

f();
```

```
// 静态作用域

var scope = 'global';

var f1 = function() {
  console.log(scope);
};
f1(); // global

var f2 = function() {
  var scope = 'f2';
  f1();
};
f2(); // global
```

由于这些JS特性，在传统Web前端开发中不常用，大部分开发者对其并不熟悉。

了解这些JS特性，理解其背后的机制，将会帮助我们在NodeJS开发中避免许多编程陷阱。

JSConf分享 - 淘宝前后端分离实战

淘宝的前后端分离实战PPT：

http://2014.jsconf.cn/slides/herman-taobaoweb/index.html#/

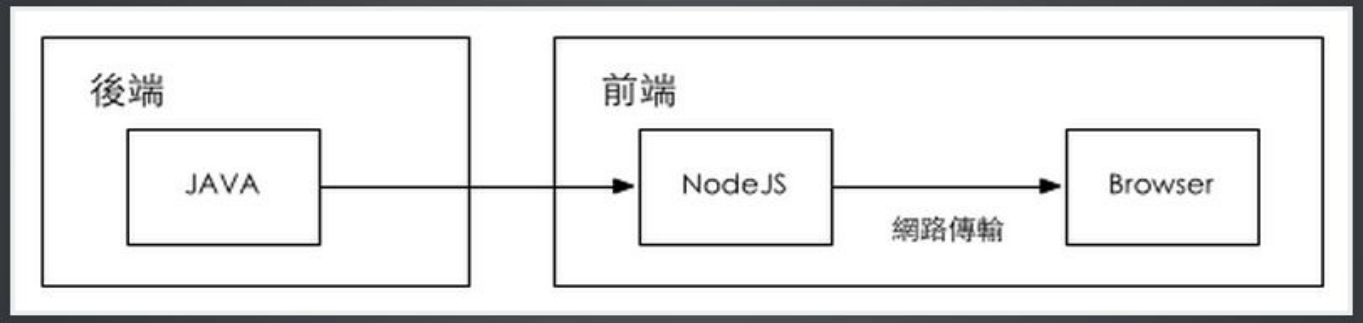
点击这里查看关键内容：

http://2014.jsconf.cn/slides/herman-taobaoweb/index.html#/57

AsiaInfo 亚信 | UED分享 · 交流

传统认知的前后端

重新定义的前后端



在服务器(JAVA) 与 浏览器(JS)的中间架了一个中间层(NODEJS)

AsiaInfo 亚信 | UED分享 · 交流

Why NodeJS

- 大家都熟悉的语言，学习成本低
- 都是JS，可以前后端复用
- 体质适合：事件驱动、非阻塞I/O
- 适合IO密集型业务
- 执行速度也不差

后端		前端
服务器		浏览器
JAVA	NodeJS	JS + HTML + CSS
• 服务层 • 提供数据接口 • 维持数据稳定 • 封装业务逻辑	• 跑在服務器上的JS • 转发数据，串接服务 • 路由设计，控制逻辑 • 渲染页面，体验优化 • 更多的可能	• 跑在浏览器上的JS • CSS、JS加載與運行 • DOM操作 • 任何的前端框架與工具 • 共用模版、路由

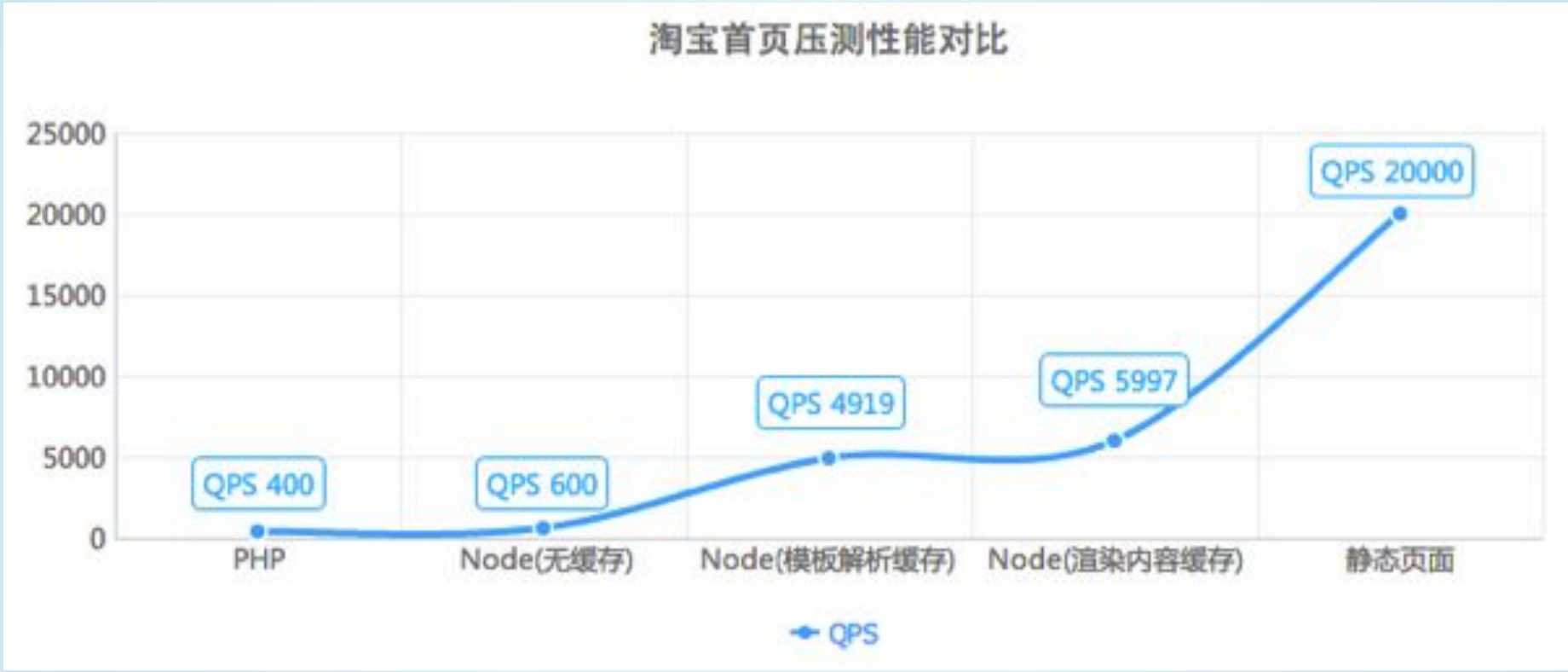
实际示例1 - 淘宝首页优化

需求

- 静态资料展示
- 方便运营管理
- 更好的承载密集庞大的流量

解决方案

- 页面缓存与Java后端定时刷新，返回缓存资料
- NodeJS产出静态页面到CDN，定时刷新


淘宝首页压测性能对比
25000
20000
15000
10000
5000
0
QPS 400
QPS 600
QPS 4919
QPS 5997
QPS 20000
PHP
Node(无缓存)
Node(模板解析缓存)
Node(渲染内容缓存)
静态页面
QPS

实际示例2 - 淘宝详情页优化

需求

- 页面数据来自各个不同接口
- 为了不影响体验，先产生页面框架后发起多个异步请求取数据更新页面
- 这些多出来的请求带来的影响不小，尤其在无线端

解决方案

- 在NodeJS端使用 BigPiper 技术合并请求，降低负担
- 分批输出，不影响体验

NodeJS带来的解决方案

页面渲染优化

- 前后端共享模版
- 首屏服务器渲染
- 次屏浏览器渲染
- 局部刷新浏览器渲染

在NodeJS端使用 BigPiper 技术合并请求，降低负担

分批输出，不影响体验

一台Node对应多台JAVA服务器
合理的分配服务器带来最大的产出

单页面应用优化

- 前后端共享路由与模版
- 前端换页，浏览器端渲染
- 直接输入URL，服务器渲染
- SEO问题迎刃而解

感谢聆听！

UED分享 • 交流
http://cssrain.github.io

AsiaInfo 亚信 | UED分享 · 交流