**前端性能监测系统的设计与实现**

**摘要**

现代计算机技术和网络技术的快速发展促进了web应用的蓬勃发展，web应用在电子商务、娱乐、生活等方面起着越来越重要的作用。近年来Web应用数量激增，各大公司通过web应用这种媒介展示和传播自己的服务，由于市场需求，应用的用户规模和用户需求也日益庞大。在这种情况下，对前端的性能要求也越来越高，用户关注的不仅仅是网页本身所能呈现的内容，还对页面本身的加载速度有要求，漫长的网页加载等待很大程度上会导致用户失去耐心而直接关闭该网页，欠佳的用户体验对企业利益来说是极大的打击。所以，前端性能优化的工作势在必行。

本文首先研究了前端性能优化的相关理论和技术，包括浏览器工作原理、性能优化原理和性能优化方案。然后针对性能优化需求，提出前端性能检测系统Lighthouse的设计与实现方案。

在相关理论与技术方面，主要探讨了浏览器是如何工作，并根据浏览器的工作原理分析性能优化原理。同时还探究了本系统将采集的性能数据指标，并如何应用与可视化表达中。

在Lighthouse系统的前端架构方面，研究并分析了基于react+redux的开发模式、pageSpeed API接口、浏览器API、数据可视化技术，并通过这些技术开发出一个用于监测网页性能指标，并给出相应优化方案的前端性能监测系统（Lighthouse）。

**关键词**：React Flux 性能优化 RESTful API 数据可视化

**第一章 引言**

**1.1 研究背景**

近年来随着web技术的发展和用户数量的激增，web应用的性能成为各大企业关注的一个焦点，美观、流畅、交互性好的用户体验能帮助企业吸引更多的用户，带来更多的企业利润；然而卡顿、缓慢、过长等待的用户体验可能会造成用户量的流失，这对企业来讲是非常巨大的损失。

对于很多企业来讲，性能在一定程度上与利益直接相关。国外有相关的调研数据显示[1]：Google的加载延迟了400ms将会导致搜索量下降0.59%；Bing加载延迟2s将会导致收入下降4.3%；Yahoo延迟400ms，用户流量将会减少5-9%；Netflix开启Gzip压缩模式可将性能提升13.25%，带宽减少50%。美国研究生项目资讯网站的调查结果显示[2]，网页的加载时间超过4秒将会导致四分之一的人放弃打开该网页。调查机构KissMitrics研究发现：网页加载速度影响用户消费，如果电子商务每天收入为10万美元，那么1秒的延迟将会让该网站每年损失250万美元[3]。

由于网站的性能对企业的利益起着至关重要的作用，很多公司也相应开发出一些针对自己网站的监控系统，如阿里巴巴、美团、腾讯、百度等互联网大公司，网站的用户体验对他们来说尤为重要。与此同时，国内的网速与许多发达国家相比慢很多，中国大陆的网速在世界排名第71，平均网速为1.774Mbps，这个数字远远低于世界平均水平[5]。因此，在国内较低网速的环境下，对Web应用进行性能优化显得尤为重要。

**1.2 研究现状**

Google公司作为一个队速度有着极致追求的互联网巨头，对web前端的性能优化非常的重视，谷歌的Chrome浏览器的Chrome DevTools本身就是一个网站性能监测分析工具。在网络面板中可以查看网络请求资源的实时信息，明确和定位哪些比预期加载更耗时的请求以便针对性的进行请求优化。在Timeline面板中可以整体上看到web页面加载和被使用过程中时间消耗在哪里，所有的时间从加载资源到解析JavaScript都会被标记在时间线上。本系统使用的PageSpeed API 也是Google提供的一套性能分析接口，该工具是从页面的加载时间来衡量网页性能。

雅虎作为过去的互联网巨头，是Web前端优化方面的先驱。在多年以前便提出来Web性能优化的黄金法则：先优化前端程序的性能，因为这是80%或以上的最终用户响应时间的花费所在。同时雅虎还提出了在前端优化方面非常著名的“雅虎军规34法则”[4]，这些法则到现在还影响着Web开发人员。另外，雅虎的首席性能工程师Steve Souders根据性能优化经验编写了《High Performance Web Sites》[6]以及其他另外8位Web前端专家一起编写的《Even Faster Web Sites》[7]提供了提升网站性能的最佳实践和使用建议。

在网页性能分析工具方面，比较出名的有以下几个工具：PageSpeed和SpeedTracer，主要对页面进行评分，然后会显示各项指标的修改意见；Yslow，和PageSpeed类似，对各种影响网站性能的因素进行评分；WebpageTest，一个在线性能测试的网站，可以通过输入的URL生成一个URL加载的时间瀑布图，并针对所有资源（JS，CSS，Image等）列出优化清单。

国内许多互联网巨头在Web性能优化方面也非常关注，在工程实践中采取了许多有针对性的优化方案，如针对HTTP协议、浏览器工作方式、浏览器缓存机制、页面结构及Ajax请求等方案。

**1.3 研究内容**

本文首先研究了浏览器的工作原理，从渲染引擎，解析和DOM树的构建，渲染树构建和布局与绘制四个过程进行探讨，通过了解浏览器渲染原理，再进行性能优化原理的探讨，进而得出性能优化的方案，然后针对性能优化需求，明确Lighthouse系统需要采集并分析的性能指标数据。

然后本文重点研究了Flux架构、基于react的组件化开发模式和Lighthouse系统的功能设计与实现三大方面。

在Flux架构方面，研究了Flux架构和传统的MVC模式的异同点，并引出并探究Flux架构中最常用的redux框架，然后结合react和react路由，提出了应用与本系统的react+redux+react-router开发模式。

在react组件化开发方面，本内容研究了组件化开发的起源及优势，接着提出容器和组件的概念，探究该如何对业务逻辑拆分成容器和组件。最后介绍了项目中使用到的蚂蚁金服的一套UI库antd及其具体使用方法。

在Lighthouse功能设计与实现方面，研究了整体的设计，入口页设计，可视化分析页设计以及路由设计，其中这些设计包括页面的响应式布局设计，容器和组件的设计，数据存储和分发的设计。

最后，在数据存储与可视化分析方面，研究了项目使用的本地存储技术，包括localStorage存储和sessionStorage；组件间数据传输方式，包括react特有的传递数据方式时HTML5下的传递方式；数据可视化分析。

**1.4 论文组织结构**

全文总共分为五个章节，每个章节的具体内容如下：

第一章，引言。分别概述了本课题的研究背景，研究现状，具体的研究内容以及研究的结构和安排。

第二章，相关理论和技术分析。分别探究了浏览器工作原理，Web性能优化原理，性能优化方案以及性能指标数据四个方面，为系统的设计和性能优化研究提供理论基础。

第三章，前端性能监测系统的设计与实现。分析Flux架构与传统MVC模式的异同，探究react+redux的开发模式，然后设计并实现前端性能监测系统。

第四章，数据的存储与可视化研究。分析项目中主要应用到的本地存储技术及其类型，探究组件间如何通信的问题，探究对数据可视化分析。

第五章，总结与展望。本章对论文全篇所研究和实现的内容进行了总结，提出研究成果所存在的不足之处，并对未来性能优化方面的学习和研究进行展望。

**第二章 相关理论和技术分析**

本章首先研究了浏览器的工作原理，然后基于浏览器的渲染原理，对Web性能优化原理进行探究，进而有针对性的研究了性能优化方案。通过以上的理论基础，探究本课题开发的项目所需采集的前端性能指标以及相关的采集方法。

**2.1 浏览器工作原理**

浏览器的主要功能是向服务器发出请求，在浏览器窗口中呈现相应的HTML文档、PDF、图片或其他类型的网络资源。资源的位置由用户使用URL（统一资源标示符）指定。

浏览器的主要组件为：

1. 用户界面：主要包括地址栏、前进/后退功能按钮、书签栏等。除了浏览器主窗口显示的请求资源外，其他显示的各个部分都属于用户界面。

2. 浏览器引擎：用于在用户界面和呈现引擎之间传送指令。

3. 渲染引擎：根据用户请求的内容进行解析相应的资源，并通过一定的规则显示在屏幕上。常见的渲染引擎[8]有Firefox的Gecko引擎，IE使用的Trident引擎和Chrome和Safari使用的Webkit引擎[8]。

4. 网络：用于网络调用，比如HTTP请求。其接口与平台无关，并为所有平台提供底层实现。

5. 用户界面后端：用户绘制基本的窗口部件，比如组合框和窗口。其公开了与平台无关的通用接口，而在底层使用操作系统的用户界面方法。

6. JavaScript引擎：用于解析和执行JavaScript代码。

7. 数据存储：这部分是持久层。浏览器需要在硬盘上保持各种数据，例如常见的Cookie和日志。新的HTML规范（HTML5）定义了“网络数据库”，这是一个完整的浏览器内数据库。

浏览器结构如图[9]：



图2-1 浏览器构造图

**参考文献**

[1] Impact of Web Latency on Conversion Rates[EB/OL]. <https://www.slideshare.net/bitcurrent/impact-of-web-latency-on-conversion-rates>

[2] Onlinegraduateprograms. Instant America Network search[EB/OL].

<http://www.onlinegraduateprograms.com/instant-america>

[3] KissMetrics. How Loading Time Affects Your Bottom Line[EB /OL].  
<http://blog.Kissmetrics.com/loading-time/?wide=1>

[4] Best Practices for Speeding Up Your Web Site[EB/OL], <http://developer.yahoo.com/performance/rules.html>

[5] 操秀英，唐婷. 中国互联网为何“跑”不出世界的网速[N].科技日报

2010,10(3):1.

[6] Steve Sounder. High Performance Web Sites[M]. 2007:1-170．

[7] Steve Sounder, et al. Even Faster Web Sites: Performance Best Practices for Web Developers[M]. O’Reily Media, 2009: 1-250.

[8] 常见的浏览器内核[EB/OL]. <https://www.jianshu.com/p/6efcccb5ed43>

[9] How Browsers Work:Behind the scenes of modern web browsers[EB/OL],

<https://www.html5rocks.com/en/tutorials/internals/howbrowserswork>