第35卷 第12期 计 算 机 工 程 2009年6月
Vol.35 No.12 Computer Engineering June 2009
·开发研究与设计技术· 文章编号：1000—3428(2009)12—0275—04 文献标识码：A 中图分类号：TP393

基于Ajax技术的智能客户端流引擎

邵一川[1,2]，申德荣[2]，赵宏伟[1]，聂铁铮[2]
(1. 沈阳大学信息学院，沈阳 110044；2. 东北大学信息学院，沈阳 110004)

摘 要：为解决传统Ajax引擎缺乏智能及服务器端负载过重问题，设计一种基于浏览器缓存机制及Ajax技术的智能客户端流引擎(SCFE)。它将业务逻辑与数据访问分开，将业务逻辑移交到客户端执行，同时在客户端缓存业务逻辑涉及的数据，形成智能客户端体系。理论和实践证明SCFE可以有效地减少服务器访问次数，减轻服务器负载，减少网络流量，使Web应用更加智能化。

关键词：Ajax引擎；缓存；智能客户端

Smart Client Flow Engine Based on Ajax

SHAO Yi-chuan[1,2], SHEN De-rong[2], ZHAO Hong-wei[1], NIE Tie-zheng[2]
(1. School Department of Information, Shenyang University, Shenyang 110044;
2. School of Information, Northeastern University, Shenyang 110004)

【Abstract】In order to solve the problems of lacking intelligence and server overload in the traditional Ajax engine, a new Ajax engine based on the browser caching mechanism named Smart Client Flow Engine(SCFE) is proposed. It separates business logic layer from data access layer and transfers business logic layer and the relevant data to the client catch to implement. Theory and practice prove that SCFE can reduce the frequency of accessing server effectively, solve the server overload problem, and make the Web applications more intelligent.

【Key words】Ajax engine; cache; smart client

1 概述

随着互联网的普及，基于B/S结构的应用程序发展迅猛，互联网浏览器的作用与日俱增。但它所提供的人机交互界面和操作方式始终不能令人满意，如交互性差、功能简单、响应速度慢、有较大通信冗余、服务器端压力较大、不能利用客户端资源。而Ajax技术的出现可以弥补这些缺点。

在Ajax应用模型中，Ajax引擎作为Ajax模型中的核心，其设计与实现直接关系到整个Web系统应用的成败。因此，如何设计与实现一个好的Ajax引擎成为当前Ajax应用的重要研究对象。

传统的Ajax引擎[1]虽然采用了异步方式通信，具有更迅速的响应能力，但它在使Web应用更动态的同时，并没有减少服务器的负载。而基于RPC方式的Ajax引擎[2]虽然在一定程度上减少了服务器负载，但由于没有客户端缓存，因此始终不能降低访问数据持久层的频率，而且不具备脱机操作能力。

本文提出一种基于Ajax技术的智能客户端流引擎(Smart Client Flow Engine, SCFE)。它实现了在传统的Ajax引擎基础上，利用浏览器缓存机制[3]，建立智能的客户端体系这一目标。SCFE是将服务器端运行的业务逻辑下载到浏览器缓存中执行，并将业务逻辑所涉及的数据缓存在浏览器中。当再次运行此业务逻辑时，操作可以直接在浏览器缓存中进行，具有脱机运行能力。如果需要连接数据持久层，SCFE会利用“钩取”操作访问服务器，同时更新浏览器缓存中“享元库”。从而克服传统Ajax引擎的弊端，使Ajax引擎更智能化。

2 SCFE的工作原理

SCFE的设计思想是将业务流程以XML代码片段形式保存于服务器端，根据用户请求需要，由SCFE动态加载相应的业务流程(XML代码片段)，之后解析并执行，同时缓存相关数据。在该业务流程不再频繁使用时，SCFE会自动将其从客户端删除，同时更新缓存，减少冗余数据。SCFE实现了业务逻辑层与数据逻辑层的分离，将业务逻辑层前置到客户端，而服务器端相当于数据逻辑层，只提供数据的管理与支持。根据以上对SCFE的分析给出下面的形式化定义，并以一段业务流网的XML代码片段举例说明。

定义1 数据钩(Hook)为业务流程中处理某一特定任务的操作，每个数据钩都有固定任务，主要负责向数据库钩取或置入数据，如下文XML代码片段中的Hook。数据钩是构建定义4中业务流网的最小单元。

定义2 数据钩存根(HookStub)是数据钩向持久层钩取数据的媒介。

定义3 钩取(HookData)是数据钩索取数据的过程。

定义4 业务流网(Flownet)是若干数据钩通过协作组成的具有完整业务处理功能的XML代码片段，如XML代码片段举例所示。业务流网的执行由SCFE触发，业务流网具有一个网口(Entry)，网口负责连接持久层。数据钩则按照顺序(Number)挂在业务流网上。

定义5 享元(Shareware)是共享数据的集合，由键和共享数据组成。其中，键是共享数据在定义8(享元库)中全局唯一的标识。

基金项目：国家“863”计划基金资助项目(2003AA4142100)；国家自然科学基金资助项目(60673139)

作者简介：邵一川(1978－)，男，讲师、硕士，主研方向：Ajax引擎，数据网格；申德荣，教授、博士；赵宏伟，讲师、硕士；聂铁铮，助教、博士

收稿日期：2008-11-30 **E-mail：**yichuan_shao@sina.com

定义 6 键(Key)和享元组成二元组<Key,Shareware>。

定义 7 生成键(GenerateKey)是将数据钩存根及其相关参数抽象成键的过程称为生成键。

定义 8 享元库(Sharewarelib)是享元与键组成的二元组的集合，是 SCFE 的客户端数据缓存库。

定义 9 享元评估(SharewareValidate)是根据享元二元组的使用频率和用户的喜好程度等评价享元二元组，并根据结果更新享元库的过程。

定义 10 织网(Weaving)是将业务流网客户端实例化并将数据钩依附于业务流网的过程。

业务流网的 XML 代码片段举例如下:

```
<Flownet>
<Entry id="dataSource">
<property name="url">
jdbc:microsoft:sqlserver:
//192.168.35.22:1433;DatabaseName=login
</property></Entry>
<Hooks>
<Hook id="login"><Number=1/>
<Javascript>
<Code value="TLoginProxy.login()"/>
<Params>
<Param name="username" value="sa"/>
<Param name="password" value="sa"/>
</Params>
<Shareware key="login" isLocal="false">
</Javascript>
</Hook>
…</Hooks>
</Flownet>
```

3 SCFE功能模块划分

根据 SCFE 功能的不同分为客户端组件和服务器端组件，如图 1 所示。

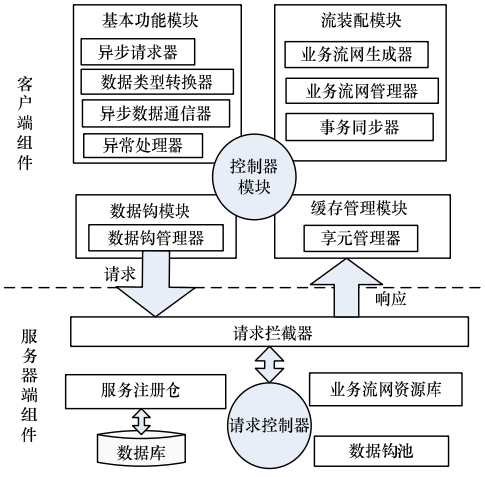

图 1 SCFE 的功能模块

3.1 客户端组件

客户端组件主要包括 5 个功能模块。

(1)基本功能模块。基本功能模块包括异步请求器、数据类型转换器、异步数据通信器、页面显示控制器、异常处理器等。主要实现发起请求、异步数据通信、结果显示、异常处理、特殊键的封闭等基本功能。基本功能模块在页面启动时就加载到内存中，并且常驻内存，是 SCFE 与用户的接口。

(2)流装配模块。流装配模块包括业务流网生成器、业务流网管理器等。负责业务流网装配与卸载。它为用户的请求提供相应的解决处理方案，并动态加载、卸载相关的业务流网。业务流网一般分为系统业务流网和自定义业务流网。系统业务流网主要指那些应用频繁的、不涉及与服务器端交互的、通用的业务，在加载 SCFE 时会随之自动启动，通常会长驻内存。自定义业务流网是指针对某种特殊应用而提供的业务，不具有通用性，所以，一般采用动态加载的方式由流装配模块进行管理。流装配模块的工作过程分为 3 步:

1)实例化业务流网并执行。解释由业务流网所定义的业务逻辑过程,根据提供的网口(Entry)和相应参数(Params)实例化数据源(DataSource),并且初始化业务流网所涉及的数据钩实例(Hooks)，将数据钩织入业务流网。

2)为业务逻辑过程及所涉及的数据钩的执行导航。以业务流网定义的业务逻辑和数据钩钩取的相关数据作为推动业务逻辑执行的依据,根据得到数据决定后续业务逻辑的走向。

3)完成与享元库的交互，维护数据钩数据。流装配模块不能直接调用数据持久层钩取数据，它要用数据钩模块完成数据钩取。

(3)数据钩模块。数据钩模块包括数据钩管理器，主要负责向服务端钩取数据及数据钩实例的生存周期管理。数据钩使用数据钩存根通过远程方法调用方式钩取远程数据。

(4)缓存管理模块。缓存管理模块包括享元管理器，主要负责客户端的享元库管理，包括享元评估、更新过期享元、注销无用享元。

(5)控制器模块。控制器模块主要负责 SCFE 的加载、各模块之间的调用。控制器模块相当于 SCFE 的“指挥官”，其他模块都是通过控制器模块的控制和处理才能最终为系统服务的。

3.2 服务器端组件

服务器端组件主要包括 5 个部件。

(1)请求拦截器。对客户端的请求进行解析，转交请求控制器，并返回客户端特定的结果。

(2)请求控制器。负责检查和抽取请求参数、调用服务、返回数据。

(3)服务注册仓。数据钩在服务端的钩点，是系统与数据持久层的接口。

(4)数据钩池。负责数据钩服务器端的管理，是数据钩的容器。利用数据钩池可以在线程间共享数据钩实例，它对请求控制器是完全透明的。服务器初始化时数据钩池被创建，停止时清除。由于数据钩池提供对数据钩的缓存机制，因此可以减少频繁地创建和回收数据钩实例产生的系统开销，节省存储资源，而且可以充分利用数据持久层的连接资源。

(5)业务流网资源库。负责向客户端请求提供业务流网资源下载。

4 SCFE工作流程

4.1 系统初始化

系统的初始化包括 3 方面内容: (1)服务器端的数据准备工作。(2)客户端 SCFE 的加载。服务器端仅仅是数据提供场所，它的数据准备工作在服务器容器初始化时进行，它包含 3 个方面: 1)由业务流网资源库初始化所有的业务流网资源，以待客户端发起请求下载; 2)服务注册仓注册所有的数据采集服务，以待客户端发起钩取数据请求; 3)创建数据钩池。(3)SCFE 的加载发生在客户端首次提出事件请求,此时 SCFE 的各个模块会被顺序加载到客户端。

4.2 用户响应处理

当页面加载完毕，用户开始在页面上进行各种操作，首先基本功能模块的异步请求器捕获到用户的操作，并且将与之相关的数据提交给控制器模块进行逻辑处理。图 2 为 SCFE 工作流程。

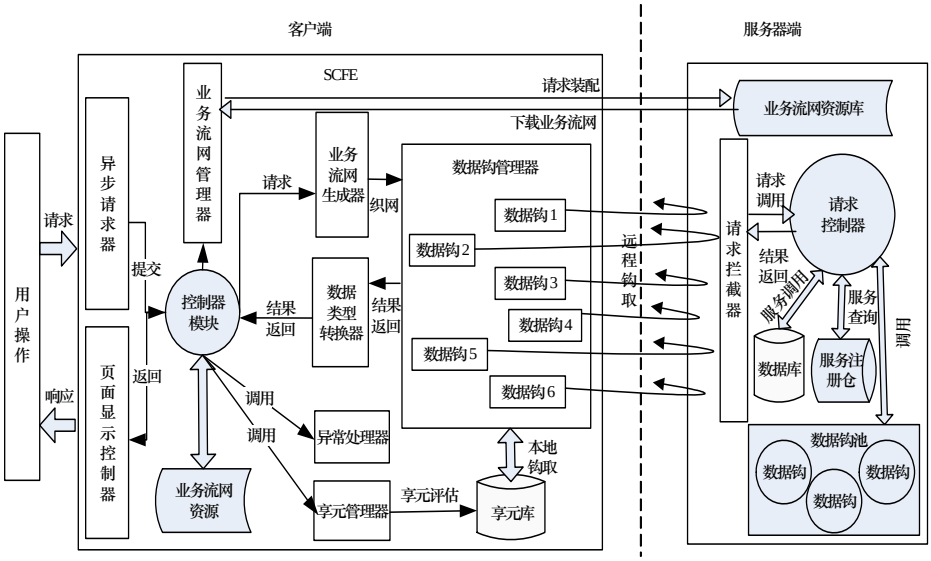

图 2 SCFE 的工作流程

4.3 逻辑处理

逻辑处理分 4 步进行:

(1)调用流装配模块的业务流网管理器进行业务流网的装配(在需要的情况下)。

(2)使用业务流网生成器对装配的业务流网织网。

(3)织网后流装配模块根据业务流网所涉及的数据钩向数据采集处理模块发出钩取任务。

(4)控制器模块继续进行其他的业务流网逻辑处理，而不必等待任务指令的执行结果。

4.4 数据处理

当数据采集处理模块接到流装配模块所发出的钩取任务后，调用相关的数据钩存根对数据进行钩取，钩取过程分为:

(1)生成键。将数据钩存根及其相关参数抽象成键。

(2)本地享元库钩取，在本地客户端享元库中进行键匹配，如果存在则直接返回享元。

(3)享元库无匹配享元时调用数据钩进行远程钩取数据享元。远程钩取的过程采用 HTTP 协议异步发送到服务器，由于采用异步方式，因此同时可能有多个 HTTP 链接存在，即有很多数据钩实例同时存在。

(4)在发送远程钩取请求之后，数据钩管理器需要监视每个数据钩的通信情况，对其进行管理，并等待服务器返回数据享元。

(5)服务器的请求拦截器接到钩取请求后会转发给请求控制器对数据钩进行解析，并根据其中的钩取函数描述信息调用服务注册仓中相对应的服务，然后将执行结果返回给数据钩。

(6)数据钩钩取成功后，数据采集处理模块会将结果与生成键形成新的享元二元组。

(7)调用缓存管理模块的享元管理器对享元二元组进行享元评估，包括注销享元、更新享元、删除过期享元等。

4.5 数据类型转换处理

当控制器模块接到数据采集处理模块返回的享元二元组之后，将其提交给基本功能模块的数据类型转换器，它负责将享元二元组转换为控制器模块可以识别的格式。

4.6 接收数据的处理

控制器模块将最终数据转发页面显示控制器，将结果显示在页面上。如果某个过程产生异常，控制器模块都会交给异常处理器处理，并将处理结果同样转发给页面显示控制器，显示给客户。

5 实验

为了与传统 Ajax 引擎进行对比，笔者分别使用 SCFE 和传统 Ajax 引擎，通过 LoadRunner[4](一种基于 B/S 的预测系统行为和性能的负载测试工具)模拟 800 个并发用户，同时向服务器发送一些相同的业务操作请求，并记录和分析测试结果。测试比较的是 3 组数据，即服务器负载情况、访问次数以及网络流量。从图 3~图 5 中可以看出:

(1)访问进程较少时，服务器负载基本相同，访问进程在 10 个以上时，采用 SCFE 的服务器负载明显较轻。

(2)在相同的访问进程情况下， SCFE 的访问次数开始较多，随着时间的增加 SCFE 访问服务器次数明显减少。

(3)开始 SCFE 的网络流量较大,但随着时间的增加 SCFE 的访问总量明显少于传统 Ajax 引擎。

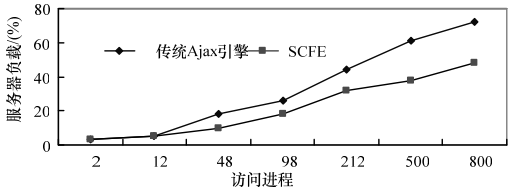

图 3 服务器负载情况对比

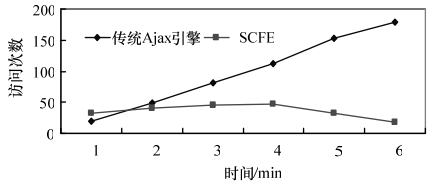

图 4 访问服务器次数对比

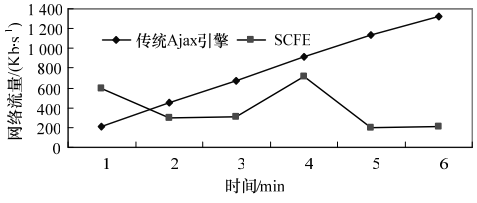

图 5 网络流量对比

6 结束语

传统 Web 模型和 Ajax 应用模型最重要的区别就是，在 Ajax 应用模型的客户端增加了一个 Ajax 引擎。而本文改进了 Ajax 引擎，为其披上了智能的“外层”——SCFE。这一智能的“外层”作为用户与系统连接的中间层下载到客户端

浏览器上运行。SCFE 的思想是将业务逻辑层前置[5]，而服务器端只相当于数据管理中心在后面提供数据支持。被披上“外层”的 Ajax 引擎具有如下优点：

(1)充分利用客户端资源。利用客户端驻留业务逻辑代码与数据，可最大程度地利用客户端的软件资源。

(2)减少访问次数。利用客户端的享元库存储数据可大大减少服务器的访问次数。

(3)减少网络流量。有些操作必须由服务端完成，尤其是对需要数据持久层访问的操作，对于这样的操作采用 Ajax 局部刷新技术减少网络流量。

(4)减少服务器负载。把业务逻辑处理移到客户端进行，减少服务器的负载。

(5)使 Web 应用更加智能。SCFE 可以动态加载、卸载业务流网，以减少冗余数据。

(6)具有脱机操作能力。由于 SCFE 装载在客户端，因此即使不与服务器端交互，SCFE 也可以利用本地享元库对一些请求提供正常的响应。

(7)提升系统性能和易用性。SCFE 是操作的载体，也是用户接口。用户的数量是无限定的，而服务器的性能却是一定的，SCFE 改变了软件的部署结构，把原本在服务器方面的业务逻辑和数据运算转移到客户端，这样可以在同样硬件成本的前提下，提升系统性能和易用性。

但是，在客户端缓存建立 SCFE 必将加重客户端负担，同时对于访问次数与访问进程较少的 Web 应用来说，SCFE 并没有提升 Web 应用性能。而且当持久层的数据被频繁地改动时，会产生持久层数据与享元库的不一致性，如何灵活有效地解决这一问题需要进一步的研究。

综上所述，SCFE 适用于以下 3 种情况：(1)系统业务逻辑和数据访问方式比较固定；(2)持久层数据更新不太频繁；(3)大量进程多次访问的系统。

参考文献

[1] 游丽贞，郭宇春，李纯喜. Ajax引擎的原理和应用[J]. 微计算机信息, 2006, 20(6): 73-75.

[2] 王　强，黄丽娟，喻国平. DWR在Struts架构的应用[J]. 微计算机信息, 2008, 24(15): 234-235.

[3] 王升平，李　青. 数据复制管理中共享缓存数据流的实现[J]. 计算机工程, 2007, 33(20): 83-85.

[4] 陈绍英，刘建华，金成姬. LoadRunner 性能测试实战[M]. 北京: 电子工业出版社, 2007.

[5] 黄　晶. 基于中间件的Web智能系统集成开发平台[J]. 吉林大学学报, 2008, 33(1): 116-122.

编辑　张正兴

(上接第 271 页)

参考文献

[1] Cardoso J F. Blind Signal Separation: Statistical Principles[J]//Proc. of the IEEE, 1998, 86(10): 2009-2026.

[2] Bofill P, Zibulevsky M. Underdetermined Blind Source Separation Using Sparse Representations[J]. Signal Processing, 2001, 81(10): 2353-2362 .

[3] Bobin J, Starck J L, Fadili J, et al. Sparsity and Morphological Diversity in Blind Source Separation[J]. IEEE Trans. on Image Processing, 2007, 16(11): 2662-2674.

[4] Abrard F, Deville Y A. Time-frequency Blind Signal Separation Method Applicable to Underdetermined Mixtures of Dependent Sources[J]. Signal Processing, 2005, 85(6): 1389-1403.

[5] Yilmaz O, Rickard S. Blind Separation of Speech Mixtures via Time-frequency Masking[J]. IEEE Trans. on Signal Processing, 2004, 52(7): 1830-1847.

[6] Li Yi, Amari S, Cichocki A, et al. Underdetermined Blind Source Separation Based on Sparse Representation[J]. IEEE Transactions on Signal Processing, 2006, 54(2): 423-437.

[7] 肖　明，谢胜利，傅予力. 欠定情形下语音信号盲分离的时域检索平均法[J]. 中国科学(E 辑): 信息科学，2007，37(12): 1564-1575.

编辑　张正兴

(上接第 274 页)

5　结束语

本文以 LDPC 译码器硬件结构实现条件为约束设计基于层调度的校验矩阵以简化硬件实现结构，采用交换校验矩阵的行列提升译码性能。定点仿真及硬件实现结果表明，这是一种低复杂度的 LDPC 译码器的校验矩阵设计方法。

参考文献

[1] Gallager R G. Low-density Parity-check Codes[M]. Cambridge, USA: MIT Press, 1963.

[2] Hocevar D. A Reduced Complexity Decoder Architecture via Layered Decoding of LDPC Codes[C]//Proc. of IEEE Workshop on Signal Processing Systems. Austin, USA: [s. n.], 2004: 107-112.

[3] Guilloud F, Boutillon E, Danger J. A-Min Decoding Algorithm of Regular and Irregular LDPC Codes[C]//Proc. of the 3rd International Symposium on Turbo Codes & Related Topics. Brest, France: [s. n.], 2003: 451-454.

[3] Brack T, Alles M. A Survey on LDPC Codes and Decoders for OFDM-based UWB Systems[C]//Proc. of Conference on Vehicular Technology. [S. l.]: Spring, 2007: 1549-1553.

[4] Kienel F, Brack T. Design of Irregular LDPC Codes for Flexible Encoder and Decoder Hardware Realization[C]//Proc. of Conf. on Software in Telecommunications and Computer Networks. [S. l.]: IEEE Press, 2006: 296-300.

编辑　张正兴