文章编号:1001-5132（2009）02-0202-05

基于数据缓存策略 Ajax 的人机交互模型

陈　兵, 郜晓英*

（宁波大学 信息科学与工程学院, 浙江 宁波 315211）

摘要: 提出了如何在一个全文检索系统中开发基于数据缓存策略 Ajax 的人机交互模型. 通过在模型中设置数据缓存策略, 使 Ajax 应用程序在全文检索应用中改善了人机交互性能, 加快了用户交互过程, 使得用户界面更加友好. 经实验对比, 基于数据缓存的 Ajax 应用程序可以进一步缩减请求和响应的次数, 明显提高了 web 应用程序响应效率.

关键词: Ajax; 缓存策略; 输入前提示; 用户友好

中图分类号: TP311　　文献标识码: A

以 Google Suggest 为代表的输入前提示功能早已流行, 事实证明, Google Suggest 是快速和友好的, 并且能够高效率地完成它的工作. 然而国内大多数全文检索系统的开发却只停留在如何提高系统的检索能力, 鲜有改善人机交互方面的设计. 为此, 本文提出了一种如何在已有的全文检索系统上开发基于数据缓存设计的 Ajax 模型的方法, 实现输入前提示功能, 旨在改善检索系统的人机交互和用户的搜索体验.

1　输入前提示的 Ajax 模型

1.1　Ajax 技术

Ajax 是 Asynchronous JavaScript and XML 的缩写, 它可以理解为“增强的 JavaScript”. 它并不是单一的技术, 而是 JavaScript、CSS、DOM、XMLHttpRequest 4 种技术的集合[1].

JavaScript 在整个 Ajax 应用中就像胶水将各个部分粘合在一起, 它定义应用的工作流和业务逻辑. 通过使用 JavaScript 操作 DOM 来改变和刷新用户界面, 不断地重绘和重新组织显示给用户的数据, 并处理用户基于鼠标和键盘的交互[2]. CSS 为 Web 页面元素提供了一种可重用的可视化样式的定义方法, 它提供了简单而又强大的方法, 以一致的方式定义和使用可视化样式. 在 Ajax 应用中, 用户界面样式可通过 CSS 独立修改. DOM 以一组可以使用 JavaScript 操作的可编程对象展现出 Web 页面的结构. 通过使用 JavaScript 脚本修改 DOM, Ajax 应用程序可以在运行时改变用户界面, 或者高效地重绘页面的某些部分. XMLHttpRequest 对象则用来与服务器进行异步通信, 在用户工作时, 提交用户的请求并获取最新的数据[3]. 服务器返回的结果都是被格式化或者处理过的, 页面会直接把结果显示出来, 这样用户就可以在浏览器中看

收稿日期: 2008-10-16.　　宁波大学学报（理工版）网址: http://3xb.nbu.edu.cn
基金项目: 国家自然科学基金（60372026）.
第一作者: 陈　兵（1984－）, 男, 浙江黄岩人, 在读硕士研究生, 主要研究方向: 智能信息检索. E-mail: dabingdejizhi@yahoo.com.cn
*通讯作者: 郜晓英（1953－）, 女, 内蒙古扎赉特旗人, 博士/教授, 主要研究方向: 多媒体信息检索. E-mail: taixiaoying@nbu.edu.cn

到变化. 由于只有发生变化的那块区域会被重新渲染, 而不是整个页面进行刷新, 所以对于用户来说, 响应速度就变得更快.

1.2 输入前提示的 Ajax 模型及其问题

首先需对输入前提示的模型方法进行相应研究, 考察现有输入前提示实现中存在的缺陷, 并且在应用中找到解决这些缺陷的方法[4].

Ajax 实现输入前提示功能时, 它的操作主要是先获取用户在页面的输入字符, 使用 JavaScript 脚本发送请求到服务器, 接着服务器返回数据给客户端, 客户端获得数据并将结果显示在表格、文本框或者其他格式中.

不过, 有时可能一些 JavaScript 脚本程序的设计缺陷, 比如没有考虑缓存服务器返回的结果集或者与服务器频繁通讯等问题, 此时 Ajax 应用可能会损害用户体验, 而不是改善用户体验.

用户界面保持足够的响应能力是非常重要的, 用户界面控件呈现所需要的时间越长, 用户看到结果等待的时间就越长, 输入前提示的效率也就越低. 同样, 访问服务器过于频繁, 或者返回太多的数据, 用户界面的响应能力就会受到损害.

2 数据缓存策略人机交互模型

由于用户界面的响应能力对于用户搜索体验是至关重要的, 为了改善 Ajax 的响应能力, 笔者提出采取数据缓存策略的交互模型. 缓存策略是基于“以空间换时间”思想的典型应用模式, 是提高系统性能的一种重要方法. 数据缓存设计分为客户端的数据缓存设计和服务器端的数据缓存设计 2 个部分.

为减少浏览器与服务器通讯次数, JavaScript 可以使用 1 个 Cache 层, 将先前收到的某个关键字或短语缓存起来. 这样的做法是在输入某个短语, 然后回删掉 1 个或几个字的时候, 不必与服务器通讯, 因为旧的建议字段已经被缓存起来[5].

服务器端的数据缓存可以减少数据库的检索次数, 提高服务器的响应能力. 当 JavaScript 脚本发送请求到服务器时, 服务器并不是直接连接数据库进行检索, 而是先在服务器缓存页面中寻找, 当页面不存在时, 再去数据库检索, 最后返回结果. 缓存的使用在大访问量的情况下能够极大地减少对数据库操作的次数, 明显降低系统负荷提高系统性能.

在设计 Ajax 引擎前, 预先在数据库中建立 1 张表(create table suggest), 然后将 1 组关键字集合的每条作为 1 条记录存入此张数据表(suggest)中.

服务器端程序的设计思路是采用 php 脚本程序连接后台数据库. 当 php 脚本接收到 XMLHTTP Request 对象发送的关键字时, 先在数据表中执行 LIKE “keyword%”查询, 找出所有 keyword 开头的建议字段, 然后将结果包装成 XML 格式返回.

由于每次 Ajax 请求都要进行 1 次数据库检索, 这样的设计对于实际应用的系统来说, 数据库的开销是很庞大的. 为此, 有必要设计 1 套服务器端数据缓存系统. 在速度上, 静态页面要比动态页面的 php 快很多这是毫无疑问的. 基于这种想法,采用 PEAR (Php Extension and Add-on Repository)中的 Cache 模块. PEAR 的 Cache 模块提供了缓存动态内容、数据库查询和 php 函数调用的框架, 它包含 1 个总体的缓存类和几个特别的子类. 缓存类使用容器类来存贮和管理缓冲数据. 当基于缓存类的 php 脚本文件接到 Ajax request 后, 先判断是否第 1 次请求该 url, 如果是, 将该 url 所需的 php 脚本文件编译成字节代码后并缓存, 然后 redirect; 如果不是, 就是说该 url 的模板已经被编译过, 检查不需要重编译后可以马上 redirect, 而重编译条件由自己设定. 本文采用的 Cache 重编译时间和索引文件更新时间一致, 索引文件没有更新的话, 数据库中 suggest 表不需重建, 原来缓存类脚本文件生成的 php 字节代码缓存文件继续有效.

最后设计浏览器端脚本, JavaScript 脚本程序

是整个 ajax 应用的核心部分.

文本框中设置 onkeyup 属性，每当用户松开键盘按键时，将触发 JavaScript 脚本中 handleKeyUp 事件函数. 此函数先判断按键类型，如果是按了 Enter 键，则执行转向查询页面；如果只是按了上、下箭头键，则选择新的建议，并更新关键字，否则就直接获取文本框的字符串. Ajax 引擎在获取关键字列表信息时，先检查该字符串是否在 Cache 中存在，如果存在则直接取出；如果不存在，则新建 1 个 XMLHTTPRequest 实例，并向服务器发送异步调用请求，再将接收到的建议字段集合在用户页面展现，最后将这个新的关键字和对应的建议字段集合以键值对的形式加入到 Cache 层. 图 1 则是基于数据缓存的 Ajax 人机交互模型:

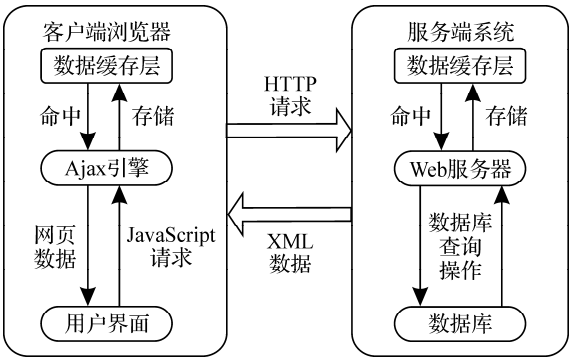

图 1 基于数据缓存的 Ajax 人机交互模型

在具体应用中，使用 1 个二维数组来保存 Cache 层的数据，而与 Cache 层相关的工作有 2 点.

(1) addToCache: 将关键字以及建议字段加入 Cache 层.

```
Function addToCache(keyword, suggestwords){
  Cache[keyword] = new Array();
  for (i=0; i<suggestwords.length; i++)
    Cache[keyword][i] = suggestwords[i];
  }
```

(2) checkCache: 检查关键字是否存在于 Cache 层中.

```
Function checkCache(keyword){
  if(Cache[keyword])
    return true;
  for(i=keyword.length-2; i>=0; i--){
    var currentKeyword = keyword.substring(0, i+1);
    if(Cache[currentKeyword]){
      var cacheResults = Cache[currentKeyword];
      var suggestwords = new Array();
      var suggestwordsSize = 0;
      for(j=0; j<cacheResults.length; j++){
        if(cacheResults[j].indexOf(keyword) = 0)
          suggestwords[suggestwordsSize++] = cacheResults[j];
        }
      addToCache(keyword, suggestwords);
      return true;
    }
  }
  return false;
}
```

在用户输入查询词的同时，输入前提示通过提供建议列表帮助用户使用正确的词汇和避免打字错误，用户只需输入少量查询词，想要的数据可以从列表获取. 比如: 用户想要检索与“黄岩”相关的信息，只要在文本框中输入“黄”，Ajax 引擎就开始发送异步请求到服务器，获取所有以“黄”字为前缀的关键字集合，显示在用户界面的下拉列表中.

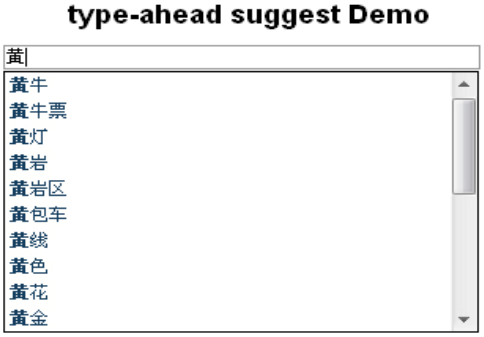

图 2 Ajax 输入前提示功能

具体提示功能如图 2 所示.

此功能加快了与用户交互过程, 使得用户界面更加友好, 改善用户的体验过程. 当然, 如果用户禁用浏览器的 JavaScript 的功能, 我们也能保证输入文本框能够正常工作, 只是少了输入前提示功能, 用户体验差些.

3　性能测试

将从浏览器的用户界面响应速度和并发访问下服务器的性能两方面来评测有基于缓存设计的 Ajax 应用与现有的 Ajax 应用的性能差别.

使用 FireBug 在 Firefox 浏览器上测试用户界面的响应速度, 并且在全文检索系统中随机选取了 212 个单字、250 个双元组和 123 个三元组进行测试. 在实验中, 使用单用户单线程访问服务器, 取这些测试时间的平均值, 所得响应速度上的对比数据见表 1.

表 1　无缓存 Ajax 和缓存 Ajax 的响应速度对比　ms

	单字	双元组	三元组
无缓存的 Ajax 应用	28	44	52
基于缓存的 Ajax 应用	29	34	37

从表 1 的测试时间可以看出, 基于缓存的 Ajax 应用在多元组查询的时候, 很明显地减少了用户界面的响应时间. 在单字查询的时候, 基于缓存的 Ajax 应用要将关键字和建议字段集合加入 Cache 层, 这部分操作消耗了一些时间, 不过在测试结果中也可看出, 在响应速度上和一般 Ajax 应用无明显差别, 所以这样的设计是可取的.

在服务器负载方面, 测试使用 Microsoft Web Application Stress Tool 工具进行, 分别测试了服务器在并发连接数为 10、50 和 100 这些情况下的性能数据. 在实验中, 每次测试时间选择为固定 10 min, 使用 Number of Hits、TTFB Avg、TTLB Avg 3 个指标来评价 Web 应用的性能. 其中的 Number of hits 是测试间隔内虚拟用户点击页面的总次数; TTFB Avg 表示从第 1 个请求发出到测试工具接收到服务器应答数据的第 1 个字节之间的平均时间; 而 TTLB Avg 表示从第 1 个请求发出到测试工具接收到服务器应答数据的最后 1 个字节之间的平均时间. 实验得到的测试数据见表 2.

从表 2 的测试数据来看, 在并发连接数同为 10 的情况下, 有无缓存设计的服务器程序的性能差别不是很明显. 在并发连接数为 50 的情况下, 无缓存设计的服务器程序的 Web 应用响应延时比较明显, 数据库的瓶颈开始显现出来. 当并发连接数达到 100 的时候, 无缓存设计的服务器程序的 Web 应用响应延时愈加明显, 可见数据库是影响这个 Web 应用性能的最大阻碍. 对比而言, 并发连接数的增加, 对基于缓存设计的服务器端程序的响应能力影响不是很大, 由于数据库查询结果集已经被缓存, 此时不存在数据库瓶颈的问题.

4　总结

提出在一个全文检索系统中开发基于数据缓存策略 Ajax 的人机交互模型. 在设计的时候采取了双数据缓存策略, 在浏览器端加入了 Cache 层用来缓存建议字段数据, 在服务器端采用 PEAR 的 Cache 模块, 缓存数据库查询结果数据集合, 并且

表 2　服务器程序设计在并发连接数分别为 10、50、100 的情况下对比

WEB 性能指标	并发连接数为 10		并发连接数为 50		并发连接数为 100	
	无缓存	带缓存	无缓存	带缓存	无缓存	带缓存
Number of Hits/次	5 644	5 598	25 510	28 832	53 131	55 123
TTFB Avg/ms	23.33	23.43	748.59	43.30	1 655.89	63.28
TTLB Avg/ms	25.19	24.83	758.63	45.86	1 682.45	67.52

限制了服务器端返回结果的数量，避免延长了服务器的响应时间. 实验证明: 数据缓存策略使得基于 Ajax 模型的 web 应用程序的响应效率有了明显提高.

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(2):205-207.

[2] 赵永屹, 宿红毅, 胡韶辉. 基于 Ajax 与 J2EE 的新型 Web 应用的设计与实现[J]. 计算机工程与设计, 2007, 28(1):189-192.

[3] 胡振华, 周斌, 冷文浩. Ajax 在 J2EE 中数据交互的应用研究[J]. 计算机工程与设计, 2008, 29(12):3 102-3 105.

[4] Crane D, Pascarello E, James D. Ajax 实战[M]. 北京: 人民邮电出版社, 2006.

[5] Darie C, Filip C T, Brinzarea B, et al. Ajax 与 PHP Web 开发[M]. 北京: 人民邮电出版社, 2007.

Human-computer Interaction Model Based on Cache Strategy and Ajax

CHEN Bing, TAI Xiao-ying*

(Faculty of Information Science and Techonolgy, Ningbo University, Ningbo 315211, China)

Abstract: This paper proposes a method to develop Ajax application based on the cache strategy in a full text searching system. The Ajax application realizes the type-ahead suggestion function by setting cache strategy in the user searching process, thus performance of human-computer interaction is improved, and the man-machine conversation process is accelerated, making the user interface more friendly. Experimental results show that the Ajax application based on the cache strategy can reduce the time consumption of http request and response, and improve the efficiency of web response owing to application program’s speed-ups.

Key words: Ajax; cache strategy; type-ahead suggest; user-friendly

CLC number: TP311 **Document code:** A

（责任编辑 章践立）