

# 应用 Ajax 提高跨库检索系统 SRU 接口的检索速度

周 强, 许雁冬, 李 宇

(中国科学院 国家科学图书馆, 北京 100190)

**摘 要:**跨库检索系统 SRU 接口, 需要同时检索多个数据库。但是, 每个数据库检索的快慢程度不同, 原来是应用同步技术实现的, 当全部数据库检索完毕, 才能显示检索结果。用户通常需要等 8 秒左右, 才能显示检索结果, 等待时间较长。文中对检索过程进行了改进, 采用 Ajax 技术, 通过应用异步技术实现检索过程, 当有一个数据库检索完成后, 就显示检索结果。用户只需要等待 3 秒左右, 就能显示检索结果。文中论述了检索、显示结果异步处理的问题, 并且给出了解决问题的办法。现在, 跨库检索系统的 SRU 检索, 检索结果显示正确, 运行状态稳定, 得到了使用者的好评。

**关键词:**同步; 异步; Ajax

**中图分类号:** TP391.3

**文献标识码:** A

**文章编号:** 1673-629X(2010)10-0203-04

## Use Ajax to Improve Metasearch System SRU Interface to Retrieval Rate

ZHOU Qiang, XU Yan-dong, LI Yu

(National Science Library, Chinese Academy of Sciences, Beijing 100190, China)

**Abstract:** The metasearch system SRU interface to simultaneously search multiple databases. However, each database, different levels of speed of retrieval, the original is to apply synchronous technology, when all the database search is completed, to display the search results. The users usually have to wait 8 seconds or so, to display the search results, waiting for a long time. In this paper, the retrieval process has been improved, using Ajax technology, through the application of asynchronous technology to achieve the retrieval process, when a database search is complete, it displays search results. Users only need to wait for 3 seconds or so, can show search results. Discuss the problems for retrieval and display asynchronous processing result, and give the solution to the problem. At present, for metasearch system SRU interface, search results display correctly, running status is stable, so it has obtained user's praise.

**Key words:** synchronization; asynchronization; Ajax

### 0 引 言

跨库检索系统是中国科学院国家科学图书馆开发的集成检索系统, 集成了 30 多个全文数据库、5 个文摘数据库、4 个电子图书库和多个图书馆公共目录数据库, 约有 100 多个数据库。跨库检索系统的结果采用 SRU 接口进行封装, 链接在国家科学图书馆主页的“找文章”栏目上。

SRU 是一个面向网络环境的信息检索协议, 从 Z39.50 发展而来, 目的是通过提供通用的框架结构, 整合对各种网络资源的访问规范, 使分布式数据库之间能够协同工作。SRU 查询结果采用基于特定元数

据规范(如 DC)的 XML 编码。由于 SRU 实现了 Web 查询的标准化、查询结果的结构化, 因而具有良好的开放性、易用性, 近年来, 在集成检索服务领域得到了广泛应用<sup>[1-4]</sup>。

跨库检索系统 SRU 接口存在不足之处:

“找文章”, 需要同时检索多个数据库, 采用多线程技术, 一个线程检索一个数据库。当所有的数据库都检索完成后, 才能显示检索结果。各数据库检索的时间是不同的, 有的快些, 有的则慢些。例如通常检索的数据库是 Elsevier ScienceDirect、SpringerLink 电子期刊、维普中文科技期刊库, 其中 Elsevier ScienceDirect 数据库的检索比较慢, 而 SpringerLink 电子期刊、维普中文科技期刊库检索相对快些, 这就要等待 Elsevier ScienceDirect 数据库检索完成后, 一起显示检索结果。用户通常要等待 8 秒左右, 才能显示检索结果, 而且在

收稿日期: 2010-01-12; 修回日期: 2010-04-29

作者简介: 周 强(1971-), 男, 北京人, 软件设计师, 硕士, CCF 会员, 研究方向为网络信息系统、信息检索。

等待时,浏览器界面一片空白,用户体验不好。

threadinfo 表(见图 1)中记录每个线程的信息,包括线程标识、数据库标识、击中数、状态、下次取数据的记录号。每个用户的一个检索线程有一条记录。

下面对其中本文用到的字段进行解释:

数据库标识(Database)是检索数据库的标识。

击中数(Hits)表示该数据库该次检索到的记录总数。

下个记录号(nextRecordNo)记录该数据库下一次检索的起始的记录数。

列名	数据类型	长度	允许空
ThreadID	bigint	8	
UserSessionID	varchar	500	✓
DatabaseID	varchar	500	✓
Query	varchar	500	✓
Field	varchar	500	✓
Hits	int	4	✓
nextRecordNo	int	4	✓
sourceNextPageNo	int	4	✓
Status	int	4	✓

图 1 threadinfo 表的结构

线程标识(ThreadID)是表的主键。

状态(Staus)记录该数据库检索的状态:当为 0 时,表示检索中;当为 3 时,表示检索成功完成;当为 4 时,表示完成 1 次的检索,因为数据库有可能击中数比较多,如一次检索不完,还要进行多次检索。

result 表记录检索结果,包括线程标识、数据库标识、用户标识、标题、作者、摘要、关键字、issn、isbn、全文链接、详细链接等。下面对其中重要的字段进行解释:

线程标识(ThreadID)是检索线程的标识。

数据库标识(sourceid)是检索数据库的标识。

数据库名称(database)是检索数据库的名称。

用户标识(UserID)是检索用户的标识。

标题(title)是检索结果的标题。

作者(author)是检索结果的作者。

书、刊物名称和出版时间(source)是检索结果的书、刊物名称和出版时间。

摘要(abstract)是检索结果的摘要。

关键字(keywords)是检索结果的关键字。

issn 是检索结果的 issn 号。

isbn 是检索结果的 isbn 号。

link1 是检索结果的全文链接的说明。

link1\_href 是检索结果的全文链接的 url。

link2 是检索结果的详细链接的说明。

link2\_href 是检索结果的详细链接的 url。

图 2 表示同步显示检索结果的模式。

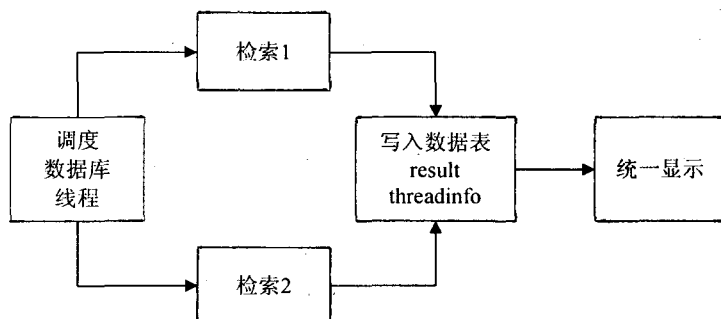


图 2 同步显示检索结果的模式

由此可见跨库检索系统 SRU 接口同步显示检索结果的检索过程的缺点是检索用户等待时间较长,用户体验不好。

## 1 跨库检索系统 SRU 接口快速显示的技术分析

是否可以加快显示结果的速度呢?“找文章”,是同时检索多个数据库,采用多线程技术,一个线程检索一个数据库,当有一个数据库完

成一次检索后,结果数据写入 result 表中。

当所有的数据库都检索完成后,从 result 表取出检索结果,显示检索结果,这是同步实现的,它的缺点是显示检索速度慢。为了提高显示速度,采用了异步处理技术。文中采用了 Ajax 技术,当有一个数据库一次检索完成后,从 result 表取出检索结果,来进行显示。Ajax 是异步的技术。Ajax 是 Asynchronous JavaScript and XML 的缩写,全称为异步 JavaScript 和 XML。它是几项技术组合而形成的新技术,包括 JavaScript、XML、XMLHttpRequest 对象等,其中 XMLHttpRequest、JavaScript 是 Ajax 技术的核心<sup>[5]</sup>。Ajax 技术是通过 XMLHttpRequest 来实现异步远程通信,利用 XML 实现数据的封装和更新,最终达到交互性更强的用户界面效果<sup>[5-10]</sup>。

JavaScript 是一种基于对象和事件驱动并且具有较高安全性能的脚本语言,是一种可以与 HTML 混合使用的脚本语言,其编写的程序可以直接在浏览器中解释执行<sup>[5]</sup>。

XML 是 Extensible Markup Language 的缩写,全称是可扩展标记语言。通过 XML,可以规范地定义结构化数据,使网上传输的数据和文档符合统一标准。XML 广泛应用在软件系统的相关文件中,日益成为因特网上的标准数据存储格式和交换格式<sup>[11,12]</sup>。

XMLHttpRequest 是 XMLHTTP 组件的对象,是浏览器中已经定义好的对象。JavaScript 通过它和服务端之间进行通信,并通过它来解析从服务器传回的 XML 文件<sup>[5]</sup>。

XMLHttpRequest 对象在 JavaScript 中创建并使用。通过该对象,Ajax 可以与桌面应用程序一样只与服务器进行数据层面的交换,而不需要每次都刷新界面<sup>[5]</sup>。

Ajax 沟通了 jsp(前台程序)、Servlet(后台程序),程序中最重要的是 Ajax 的代码,图 3 给出了异步显示结果的模式。

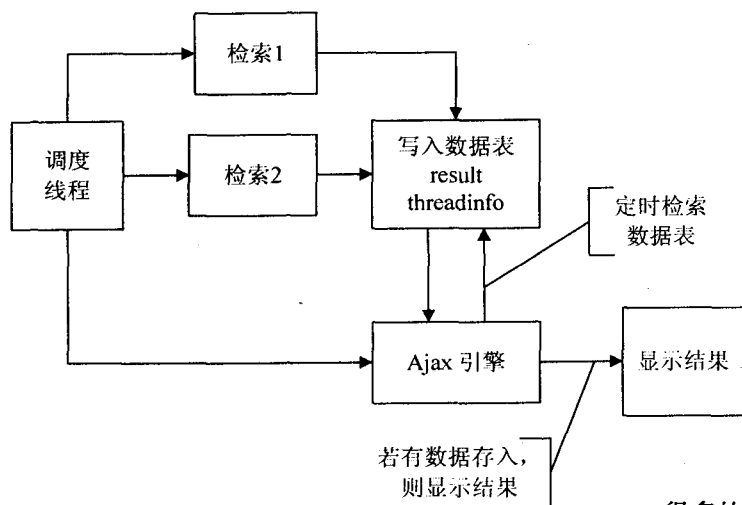


图 3 异步显示检索结果的模式

在检索中,要解决以下两个问题:

- (1) 如何判断有数据库检索完成了,可以显示了?
- (2) 如何判断所有数据库检索结束了?

如图 3 所示,Ajax 引擎在检索和显示结果之间,Ajax 引擎定时(时间可以设置,比如设为 1 秒)查询 result 表和 threadinfo 表,若有新的检索结果,就显示检索结果,否则就等待,直到下一次启动查询 result 表和 threadinfo 表。

Ajax 引擎定时启动一个判断是否可以显示结果的查询任务,新建 XMLHttpRequest 对象,调用一个 HttpServlet 类<sup>[13]</sup>(ReadDbAction.java),进行数据库查询,判断是否可以显示检索结果,若可以显示检索结果,则重定向到显示检索结果的 jsp(search-sruresult.jsp),进行显示,否则就等待,直到下一次启动任务。这就解决了第一个问题。

数据表 threadinfo 中,字段 Status 记录该数据库检索的状态:当为 0 时,表示检索中;当为 3 时,表示检索成功完成;当为 4 时,表示检索完成 1 页。当一个用户所有的数据库检索状态都为 3 时,数据库检索就完成了。这就解决了第二个问题。

## 2 Ajax 引擎的处理过程

(1) 新建 XMLHttpRequest,调用类 ReadDbAction。

首先,新建一个 XMLHttpRequest 对象 XMLHttpRequest。

接着,从 HttpSession 中取出标识当前用户的参数 userID,每次用户检索时,都会生成一个用户标识,并放入 HttpSession 中。

类 ReadDbAction.java,在 web 应用部署目录被映射为 readnum。

```
//指定后台响应的 servlet (ReadDbAction.java)
surl = "/metasearch_ SRU/readnum? suserID = " +
userID;
// 打开连接,true 表示异步提交
XMLHttpRequest.open("GET",surl,true);
XMLHttpRequest.onreadystatechange = processResponse; //
指定响应函数
XMLHttpRequest.send(null); //发送请求
```

(2) 类 ReadDbAction 查询 threadinfo 表和 result 表,判断是否显示结果页面。

数据表 threadinfo 中,字段 Hits 记录该数据库已经击中的数量。数据表 threadinfo 中,字段 nextRecordNo 记录该数据库下一次检索的起始的记录数。

因为有的检索词,数据库击中的数据会很多的,一次检索不完,需要进行多次检索,为此,设定了字段 nextRecordNo,记录该数据库下一次检索的起始的记录数。

```
int haveNumber = 0; // 是否有新的数据可以展示,若有为 1,否则为 0
```

```
int save=0; //已经取得的总数据量,初始化为 0
```

```
int total_hit=0; //数据库的总击中数,初始化为 0
```

```
int getRecordNum = 0; // 当前已经取得的总数据量
```

```
int status-3 = 1; // 数据库的检索状态是否为 3
```

```
Statement stmt;
```

```
ResultSet rs;
```

```
stmt = conn. createStatement();
```

构造查询语句,从 threadinfo 表中查询数据,其中 userID、threadIDs 是传入的变量。

userID 就是前面介绍的当前用户标识。跨库检索时,会同时检索多个选中的数据库,每检索一个数据库,启动一个线程,对于每一个线程,会生成一个线程标识,多个线程标识以 SQL 语句形式组合起来,就形成了变量 threadIDs 的值,这个变量是在检索时生成的,写入了 HttpSession 的,这里是从 HttpSession 中取出的。

```
String ssl = "select * from threadinfo where (ThreadID IN
```

```

(SELECT ThreadID FROM Result WHERE UserID = '' +
userID + "" + threadIDs + "));";
rs = stmt.executeQuery(ssql);
while (rs.next()) {
    //取出 Status 字段的值
    String status = rs.getString("Status");
    int status_int = Integer.parseInt(status); //字符型转换为整型
    //判断数据库检索是否完成了,即成功完成
    // 这就是判断所有数据库检索完成的语句
    if (status_int==3) {
        status_3 = status_3 * 1;
    } else {
        status_3 = status_3 * 0;
    }
    //取出击中数
    s_hit = rs.getString("Hits");
    hit = Integer.parseInt(s_hit); // 字符型转换为整型
    // 取出下一次检索的起始的记录数
    nextRecordNo = rs.getString("nextRecordNo");
    //计算出本次检索已经取得的记录数
    theNumber = Integer.parseInt(nextRecordNo) - 1;
    save = save + theNumber;
    total_hit = total_hit + hit;
}
//计算出现在已经取得的所有的记录数
//若检索完成,该数就是总击中数,否则就是目前取得击中
数的总和
    if (status_3==1) {
        getRecordNum = total_hit;
    } else {
        getRecordNum = save;
    }
    // startRecord 是由 SRU 的 Url 传入参数,即检索结果的起
始记录号
    int theNextRec = Integer.parseInt(startRecord); //字符型转
换为整型
    //如果 startRecord 小于等于 已经取得的检索记录数,就可
以显示结果
    //这就是判断有一个数据库一次检索完成了,并且可以显
示结果的语句
    if (theNextRec<=getRecordNum) {
        haveNumber = 1;
    }
    return haveNumber;
}
把 haveNumber 写入 HttpServletResponse, 返回给对象 XML-
HttpRequest
(3)显示结果,或者继续等待。
XMLHttpRequest 对象 XMLHttpRequest 处理结果响

```

应的函数,在前面定义为 processResponse,以下的代码就是 processResponse 中的。

```

// 变量 datanum 的值就是前面的变量 haveNumber 的值
var datanum = XMLHttpRequest.responseText;
datanum = trim(datanum); //去掉 datanum 左右的空格
var intDatanum = parseInt(datanum); //字符型转换为整型
if (datanum==1) {
    //重定向到 search_sruresult.jsp,显示结果
    var hurl = "search_sruresult.jsp";
    window.location = hurl;
} else {
    //过一段时间(这个值可以设定的),重新调用类 Read-
DbAction,进行查询
    setTimeout("sendDataRequest()",1000);
}

```

### 3 结束语

中国科学院国家科学图书馆开发了跨库检索系统,检索的数据库有 100 多个。跨库检索系统的结果采用 SRU 接口进行封装,链接在国家科学图书馆主页的“找文章”栏目上。

“找文章”需要同时检索多个数据库。但是,每个数据库检索需要的时间是不同的。由于各数据库检索的快慢程度不同,如应用同步检索,检索快的需要等检索慢的,直到全部数据库检索完毕,才能显示检索结果。采用同步检索,检索用户通常需要等 8 秒左右,才能显示检索结果,用户体验不好。

文中采用 Ajax 技术,实现异步检索。当有一个数据库一次检索完成后,从 result 表取出检索结果,来进行显示。这里需要解决两个问题:

- (1)如何判断有数据库检索完成了,可以显示了?
- (2)如何判断所有数据库检索结束了?

文中应用 Ajax 技术解决了这两个问题。通过新建 XMLHttpRequest 对象,调用一个 HttpServlet 类 (ReadDbAction.java),进行数据库查询,判断是否可以显示检索结果,这就解决了第一个问题。

数据表 threadinfo 中,字段 Status 记录该数据库检索的状态:当为 0 时,表示检索中;当为 3 时,表示检索成功完成;当为 4 时,表示检索完成 1 页。当一个用户所有的数据库检索状态都为 3 时,数据库检索就完成了。这就解决了第二个问题。

通过采用 Ajax 技术,实现跨库检索系统 SRU 接口异步处理后,使检索速度加快了。现在,跨库检索系统的 SRU 检索,检索结果显示正确、运行状态稳定,检索用户等待时间只有 3 秒左右,用户体验好多了,得到了使用者的好评。

(下转第 210 页)

字识别准确率之间的关系如图 5 所示,当  $k$  取值 6000 (约为描述子总数的 10%) 左右时,汉字的正确识别即接近 90%,因此该模型在汉字识别领域具有广阔的应用前景。

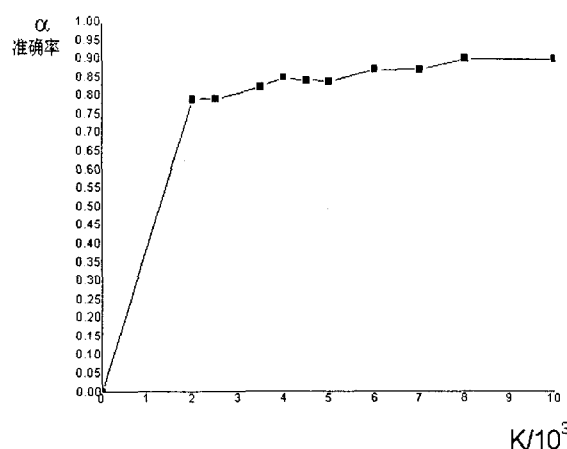


图 5 不同的  $k$  值与汉字识别正确率之间的关系示意图

### 3 结束语

文中借鉴了视频检索模型思想,并引入 KMEANS 聚类操作,有效地解决了复杂背景下的旋转汉字难以识别的问题。文中的汉字识别模型对噪声、复杂背景和旋转等字体的识别具有较强的鲁棒性。实验结果表明,在实际应用中该模型的综合性能优于传统的 OCR 汉字识别软件,其研究和应用价值不可忽视。

#### 参考文献:

- [1] Casey R, Nagy G. Recognition of Printed Chinese Characters [J]. IEEE Trans. Electronic Computers, 1996, 15(1): 91 -

101.

- [2] 丁晓青. 汉字识别研究的回顾[J]. 电子学报, 2002(9): 64 - 68.
- [3] Mori S, Suen C Y, Yamamoto K. Historical review of OCR research and development[J]. Proceedings of IEEE, 1992, 80 (7): 1029 - 1058.
- [4] 王 勇, 郑 辉, 胡德文. 图像和视频中的文字获取技术[J]. 中国图象图形学报, 2004, 9(5): 532 - 538.
- [5] 卢汉洁, 孔维新, 廖 明, 等. 基于内容的视频信号与图像库检索中的图像技术[J]. 自动化学报, 2001, 21(1): 56 - 69.
- [6] Sivic J, Zisserman A. Video Google: a text retrieval approach to object matching in videos[C]// Proceedings of International Conference on Computer Vision. Washington, DC: [s. n.], 2003: 1470 - 1477.
- [7] Lowe D G. Distinctive image features from scale - invariant keypoints [J]. International Journal of Computer Vision, 2004, 60 (2): 91 - 110.
- [8] Ke Y, Sukthankar R. PCA - SIFT: A more distinctive representation for local image descriptors[C]// IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR' 2004). Washington, DC: [s. n.], 2004: 506 - 513.
- [9] Mikolajczyk K, Schmid C. A performance evaluation of local descriptors[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2005, 27(10): 1615 - 1630.
- [10] 孙吉贵, 刘 杰, 赵连宇. 聚类算法研究[J]. 软件学报, 2008, 19(1): 48 - 61.
- [11] 彭敦陆, 周傲英. 基于方法聚类的 Web 服务检索技术[J]. 计算机应用, 2007, 27(10): 2365 - 2368.
- [12] 施聪莺, 徐朝军, 杨晓江. TFIDF 算法研究综述[J]. 计算机应用, 2009, 29(6): 167 - 170.

(上接第 206 页)

#### 参考文献:

- [1] SRU: Search and Retrieve via URL (standards, Library of Congress)[EB/OL]. 2007 - 08 - 23. <http://www.loc.gov/standards/sru/index.html>.
- [2] SRW/U (OCLC - Software) [EB/OL]. 2007 - 08 - 23. <http://www.oclc.org/asiapacific/zcn/research/software/srw/default.htm>.
- [3] Eric L M. An introduction to the search/retrieve URL service (SRU)[EB/OL]. 2007 - 08 - 23. <http://www.ariadne.ac.uk/issue40/morgan/>.
- [4] 李春旺, 王小梅, 王 昉, 等. 基于 SRU 的集成服务平台设计与实现[J]. 现代图书情报技术, 2007, 23(10): 12 - 15.
- [5] 王 沛, 冯曼菲. 征服 AJAX - Web2.0 开发技术详解[M]. 北京: 人民邮电出版社, 2006.
- [6] 徐 驰. Ajax 模式在异步交互 Web 环境中的应用[J]. 计算机技术与发展, 2006, 16(11): 228 - 230.
- [7] 杨晓俊. Web2.0 下的 Ajax 及其应用[J]. 计算机与数字工程, 2007, 35(8): 157 - 160.
- [8] 柯昌正, 黄厚宽. Ajax 技术的原理与应用[J]. 铁路计算机应用, 2007(1): 47 - 49.
- [9] 王 东, 孙 彬. 基于 Ajax 的 MVC 框架的改造分析[J]. 计算机应用, 2007(s1): 301 - 303.
- [10] 冉春玉, 童 莹. Ajax 技术及其 Web 开发[J]. 福建电脑, 2007(3): 100 - 101.
- [11] 左伟明. 即用即查 - XML 数据标记语言参考手册[M]. 北京: 人民邮电出版社, 2007.
- [12] 侯要红, 栗松涛. Java XML 应用程序设计[M]. 北京: 机械工业出版社, 2007.
- [13] 蔡 剑, 景 楠. Java 网络程序设计: J2EE (含 1.4 最新功能)[M]. 北京: 清华大学出版社, 2003.