文章编号：2096-1472(2021)-03-48-04
DOI:10.19644/j.cnki.issn2096-1472.2021.03.012

体育竞赛实时数据分享系统储存方案设计与优化

石 清

(武汉体育学院体育工程与信息技术学院，湖北 武汉 430079)
✉stoneclear@163.com

摘 要：在“互联网+”概念的影响下，越来越多的信息技术应用于体育产业。本文通过构建基于MEAN框架的体育竞赛实时数据管理系统，设计了一种结合本地存储与远端云数据库的分布式存储方案，既实现数据的实时分享，又保障数据的可靠性。并通过实验的方式比较了基于HTML5 Local Storage本地存储的两种方法与本地NoSQL数据库的性能差异，从而实现系统存储方案的优化。

关键词：实时数据共享；竞赛系统；MEAN框架；RESTful接口
中图分类号：TP311 **文献标识码**：A

Design and Optimization of Storage Scheme for Real-time Data Sharing System of Sports Competitions

SHI Qing

(*School of Sports Engineering and Information Technology*, *Wuhan Sports University*, *Wuhan* 430079, *China*)
✉stoneclear@163.com

Abstract: Under the influence of the Internet + concept, more and more information technology is used in sports industry. By constructing a real-time data management system for sports competitions based on the MEAN (MongoDB + Express + AngularJS + NodeJS) framework, this paper proposes to design a distributed storage scheme that combines local storage and remote cloud database, which realizes real-time data sharing and guarantees data reliability. Performance difference between the two methods based on HTML5 Local Storage and local NOSQL (Not Only SQL) database is compared through experiment, so to optimize the system storage scheme.

Keywords: real-time data sharing; competition system; MEAN framework; RESTful interface

1 引言(Introduction)

随着互联网信息技术的发展，云计算和云数据库以其可伸缩性、高可靠性等特点受到了开发者和企业的青睐，越来越多的企业将其服务和数据转移到云上。这些海量的数据如果可以通过接口的形式实现共享，将为大数据的分析和挖掘提供数据。然而冲突数据和时效错误数据相叠加而产生的错误数据将产生严重的后果，德国数据分析机构的调查显示：美国每年因为劣质数据而造成的损失高达6,000 亿美元[1]。因此，保证数据的可靠性乃是数据最重要的因素之一，建立在低质量数据基础之上的数据分析、数据挖掘将会变成一纸空谈，甚至会产生重大的错误。

有时为了确保数据的可靠性，往往会以牺牲数据的实时性为代价。而体育赛事信息的变动和不确定性，要求信息传播最大程度地追求时效性与接收的便捷性[2]。而传统竞赛服务系统多采用C/S构架，前期需要较大的硬件成本投入，同时开发周期长、成本高，除个别大型综合性赛事外，单项体育比赛的竞赛系统几乎不具备实时分享数据的能力。而基于B/S架构的系统除了开发周期端、成本低、系统可扩展性高之外，数据信息服务通过连接特定的数据接口，实现数据实时通讯，既能服务于电视直播、现场大屏幕的数据需求，还能

为所有对实时数据有需求的应用提供数据支持。赛后，所有的数据还可用于其他分享和数据挖掘，以达到数据价值利用最大化。本文研究的目的就是建立一套体育竞赛数据管理系统，既要满足实时传播的需要，又要建立在高可靠性的数据基础之上。

2　系统框架(System framework)

2.1　相关技术介绍

本文采用MEAN框架进行系统开发。MEAN框架是一个JavaScript平台下现代Web开发框架的总称，是MongoDB、Express、Angular、Node.js四个框架的第一个字母组合的简称。Node.js是一个基于Chrome JavaScript运行时建立的平台，用于搭建响应速度快、易于扩展的网络应用[3]。Node.js本身的特点非常适合在分布式设备上运行数据密集型的实时应用。它采用一系列“非阻塞”I/O模型库来支持事件循环方式，为文件系统、数据库等资源提供接口，通过异步的方式实现数据的非阻塞传输。Express是一款基于Node.js的Web应用开发框架。Express虽然规模小巧，却为Web和移动应用程序提供一组强大的功能。Angular是由Google公司开发和维护前端的应用框架，其核心特点包括MVVM、模块化、自动化双向数据绑定、语义化标签、依赖注入等。MongoDB是NoSQL的一种，可以方便地存储复杂的数据类型。其由于高性能、易部署、易使用，以及存储数据方便等特点，是目前应用最广泛的NoSQL数据库。

2.2　RESTful接口

在系统设计和实现的过程中，始终以RESTful接口的形式实行数据的互联共享。无论是现场比分、历史战绩，还是赛队或队员的相关信息都可以通过统一的数据接口实现访问，从而避免了运动项目或赛事因业务与功能的不同，建立数据彼此独立、相互封闭信息的“信息孤岛”[4]；以资源的方式提供数据服务可以提高体育信息数据的有效集成，扩大媒介的传播效果[5]。以信息服务为目标，提供体育赛事相关资料的综合性服务，必将成为大型体育赛事信息服务变革的方式，对于实现信息服务系统化、标准化以及提高信息利用率，具有重要意义[6]。

进行RESTful风格的API设计时，需要客户端和服务器之间的交互在请求之间是无状态的。所谓无状态即所有的资源都是通过URI进行定位的，而且该URI提供的资源与其他资源无关，也不会因为其他资源的变化而改变[7]。例如，查询某运动员的成绩，如果查询成绩时需要登录成绩管理系统，进入特定的成绩查询页面，执行相关操作后获取该名运动员的历史成绩，上述情况则属于有状态，因为查询运动员历史成绩的每一步操作都依赖于前一步操作的结果，只要前置操作失败，则后续操作就无法执行。如果在浏览器地址栏中输入一个URL地址即可返回某运动员的历史成绩，则该情况属于无状态，因为获取运动员成绩资源不依赖于其他资源或状态，而是与一个URL地址相对应，可以通过HTTP的GET方法得到该资源。

3　数据存储方案设计(Design of data storage)

高质量的数据信息服务，除了追求数据在分享方面的高效之外，对数据的准确性也有相当高的要求。不以数据质量为前提的数据服务，其后果必然是灾难性的。因此，系统的数据存储方案就显得格外重要，既要保证数据便于分享，也要保证数据准确。如果仅仅采用单一性的数据库将很难满足需求。

系统存储方案采用的是本地与云端同步存储的模式，如图1所示。系统前端通过AJAX发送数据到云数据库，并通过云数据库实现数据的实时分享，但由于不可预测的网络原因，存储过程有可能会出现长时间延迟，甚至失败。而竞技体育赛事往往无法重赛，如果出现数据丢失的情况将是非常严重的事故。这不仅无法满足数据分享的需求，更有可能因为竞赛数据的丢失导致赛事无法正常进行。本地存储由于没有网络依赖，能保证数据的完整性和准确性。系统通过同时向本地和云端存储数据，确保竞赛数据不会丢失。同时，将本地数据与云数据库进行数据一致性校对，避免由于网络原因造成云端数据丢失，确保云端数据的完整性。用户通过向云数据库发送请求，获取准确的竞赛实时数据。

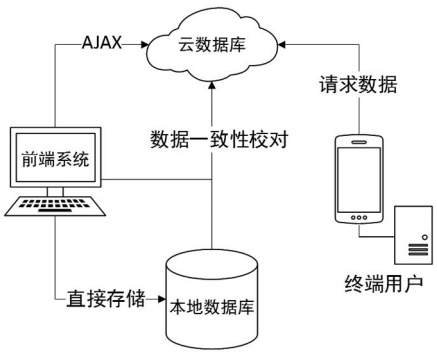

图1 存储方案示意图

Fig.1 Storage plan diagram

在数据库选择方面，以MySQL为代表的传统关系型数据库在处理海量数据文档时有其固有的局限性，很难满足海量数据的柔性管理需求[8]。而以MongoDB为代表的NoSQL数据库却能够很好地应对非结构化的数据存储，对于表结构需要临时调整、字段不固定、快速响应海量数据写入等方面都具有相当明显的优势。同时有研究表明，MongoDB数据库与MySQL数据库相比，多线程、高并发情况下的数据插入性能明显高得多[7]。最终，系统服务器端选择以MongoDB为存储

数据库。

本地存储的选择，同样将MongoDB作为主要的候选对象，同时，由于整个竞赛系统只需要存储与比赛相关的数据信息，而运动员信息、赛队信息、往期赛事成绩等相关数据无须进行存储，数据量并不大。在确认云端数据完整无误之后，本地数据可以删除。所以，HTML5 Local Storage也可作为候选对象。

4 存储性能优化(Storage performance optimization)

在明确存储方案之后，为进一步优化该方案，在系统设计过程中，对本地各存储备选方案进行了性能测试，以选出最优本地存储方案。

4.1 测试环境

测试环境使用的是Intel Core i7 2.5 GHz CPU，16 GB内存、AMD Radeon R9 M370X 2048 MB显存的GPU，操作系统为macOS Sierra 10.12.1，测试用的浏览器分别是Safari、Chrome和Opera。由于在macOS中不支持Windows操作系统下常用的IE浏览器，因此没有在IE浏览器中进行测试。

4.2 测试方法

分别调用HTML5原生Local Storage方法、自定义基于Angular Service规范的HTML5 Local Storage方法(以下简称Angular Local Storage Service)，以及本地MongoDB Service方法，比较它们在相同存储次数下所消耗的时间。存储的对象是一个不断增加的MongoDB ObjectId对象数组。设置这样的存储测试方法，一方面是因为摩托艇这类环圈竞速赛比赛规则的原因；另一方面这也是NoSQL数据库的优势所在，无须像SQL数据库一样，每条数据顺序插入数据库中，而只需要修改文档数据库的字段内容即可。同时，该数据结构也更有利于进行与云端的数据一致性校验。

测试的存储次数分别为100、1,000、2,000、3,000、4,000和5,000，最终判断Safari、Chrome和Opera三款浏览器的性能优劣。由于在真实比赛过程中，系统存储数据I/O的频次远低于实验设计，同时实验的主要目的是为了找出在较低频次下，各个储存方案的性能差异，因此本次实验的最大测试次数设置为5,000，没有继续增大循环测试次数。

实际测试过程中，通过在循环开始前后分别调用Date.now()方法获取系统时间，通过比较两次系统时间的时间差来确定循环运行所消耗的时间。以HTML5 Local Storage测试的实验代码为例：

```
saveByLocalStorage(data){
  let tempArray = [];
  let tempObject = {};
  let startTime = Date.now();
  for(let i=0;i<TEST_TIMES;i++){
    tempArray.push(data);
    tempObject['result'] = tempArray;
    localStorage.setItem(LOCAL_STORAGE_KEY,
JSON.stringify(tempObject));
  }
  let endTime = Date.now();
 console.log("LocalStorage Testing Duration Time :" +
(endTime - startTime));}
```

Angular的结构框架不推荐直接使用组件获取或者保存数据，虽然直接调用HTML5 Local Storage方法可以实现数据的存储和读取，但是与Angular的理念不一致。因此在开发的过程中独立编写了一个基于HTML5 Local Storage的Service，再通过Angular的依赖注入机制应用于系统之中。

```
class LocalStorageService {
    constructor(storage) {

    this.storage = storage; }
    get(key) {
    try {
    return JSON.parse(this.storage.getItem(key));
        }
    catch (error) { return null; }
        }
    set(key, value) {
    this.storage.setItem(key, JSON.stringify
(value));}
    remove(key) {
    this.storage.removeItem(key);
        }
    }
```

保存到本地MongoDB数据库时，利用Angular的Observable对象，借助RxJS库对数据流进行异步处理，在Angular中几乎都使用该方法来处理AJAX请求和响应。在Angular前端设置比赛成绩的Service的数据服务之前，需要在服务器端创建RESTful风格的数据接口用以满足前端的需要。由于是以竞赛成绩为测试内容，而竞赛成绩本身属于敏感信息，不能允许任意用户进行随意的创建、修改和删除，因此在接口设计时除GET方法外，其他所有方法都进行了权限管理，必须管理员用户才可以访问这些接口。

5 实验结果(Experimental results)

在实际测试中，分别在Safari、Chrome和Opera三款浏

览器运行测试程序，所有测试结果均在console面板中显示。从测试结果来看，Safari、Chrome和Opera三款浏览器在HTML5 Local Storage、Angular Local Storage Service和本地MongoDB Service这三项测试中，所表现出的性能没有显著差异。这三款浏览器在HTML5 Local Storage和Angular Local Storage Service的测试对比中，均表现出用Angular Service标准写出的方法比直接调用HTML5 Local Storage有5%左右的性能优势。用Angular Service标准编写的服务本身调用的就是HTML5 Local Storage，虽然只是用Angular Service的规范将HTML5 Local Storage进行封装，但Angular Service服务由于在构建Component时已经加载成功，因此会表现出5%左右的性能优势。

而无论是HTML5 Local Storage还是Angular Local Storage Service，在测试次数在3,000以下时，与本地MongoDB Service都存在着数量级上的优势。随着测试次数增多，它们之间的差异虽然缩小，性能差距依然在三倍以上。

具体测试结果如表1—表3所示。

表1 Safari测试结果

Tab.1 Safari experimental results

测试次数	HTML5 LOCAL STORAGE	ANGULAR LOCAL STORAGE SERVICE	LOCAL MONGODB SERVICE
100	3 ms	2 ms	117 ms
1,000	101 ms	98 ms	1,049 ms
2,000	349 ms	328 ms	2,223 ms
3,000	789 ms	731 ms	3,546 ms
4,000	1,401 ms	1,352 ms	5,003 ms
5,000	2,247 ms	2,139 ms	6,465 ms

表2 Chrome测试结果

Tab.2 Chrome experimental results

测试次数	HTML5 LOCAL STORAGE	ANGULAR LOCAL STORAGE SERVICE	LOCAL MONGODB SERVICE
100	3 ms	3 ms	126 ms
1,000	109 ms	103 ms	1,128 ms
2,000	375 ms	326 ms	2,323 ms
3,000	848 ms	772 ms	3,548 ms
4,000	1,494 ms	1,367 ms	4,983 ms
5,000	2,301 ms	2,147 ms	6,668 ms

表3 Opera测试结果

Tab.3 Opera experimental results

测试次数	HTML5 LOCAL STORAGE	ANGULAR LOCAL STORAGE SERVICE	LOCAL MONGODB SERVICE
100	4 ms	3 ms	123 ms
1,000	102 ms	101 ms	1,034 ms
2,000	360 ms	348 ms	2,055 ms
3,000	823 ms	770 ms	3,234 ms
4,000	1,400 ms	1,370 ms	4,621 ms
5,000	2,209 ms	2,142 ms	6,055 ms

从整体的测试结果来看，使用Local Storage的存储方案明显会更加具有优势。Angular Local Storage Service的方法虽然在开发过程中比直接调用HTML5 Local Storage要复杂一些，但是其兼顾Angular Service的模式，同时还具备5%左右的性能优势，所以在最终开发过程中选用的是Angular Local Storage Service的开发方式。

6 结论(Conclusion)

对于追求时效性的B/S架构系统来说，以RESTful风格进行API设计将极大地方便数据分享和协同开发的效率，降低开发过程中出错的概率。同时，将所有的数据以资源的形式为外界提供服务时，将最大程度消除信息化障碍，确保所有人能够畅通地获取体育服务信息。

数据存储方案采用本地与云端同步存储的模式设计与开发，提高了系统的容错率，增加了数据的可靠性。在本地存储方案选择方面，采用Web浏览器提供的Local Storage足以满足临时性的需求，且性能与本地MongoDB数据库相比，在低频次的使用时具有数量级上的优势，可优先考虑以HTML5 Local Storage作为临时存储数据库对象。在以Angular为前端开发框架时，基于Angular标准的service存储性能比直接调用HTML5 Local Storage具有5%左右的性能优势，可优先考虑。

参考文献(References)

[1] 杜岳峰,申德荣,聂铁铮.基于关联数据的一致性和时效性清洗方法[J].计算机学报,2017,40(1):92-105.

[2] 冉荣.自媒体背景下体育微博的发展及价值[J].新媒在线,2017(11):147-148.

[3] 王伶俐.基于NodeJS + Express框架的轻应用定制平台的设计与实现[J].计算机科学,2017(11):596-599.

[4] 王家宏,孙晋海,伊超.基于数据集成的水上项目国家队数据库网络管理平台的设计与开发[J].山东体育学院学报,2015(1):1-7.

[5] 林宁波.大数据时代体育信息传播改革路径[J].大统计,2018(3):31-33.

[6] 王广田,张文莲.大型体育赛事档案信息服务的困境与变革[J].山西档案,2018(7):130-132.

[7] Haviv, Amos Q. MEAN Web Development: Master real-time web application development using a mean combination of MongoDB, Express, Angular JS, and Node.js[M]. Birmingham, UK: Packet Publishing, 2014:2-8.

[8] 胡小春,李陶深,王乐.基于NoSQL的大数据应用设计与性能保障方案研究[J].广西大学学报(自然科学版),2014(6):633-640.

作者简介：

石　清(1981-)，男，硕士，讲师.研究领域：体育信息技术，人机工程学.