中科院院史知识竞赛系统的优化实现①

杜义华, 陈 雄, 郭小龙

(中国科学院 计算机网络信息中心, 北京 100864)

摘 要: 根据竞赛系统访问量大、并发高、用户在线时间长的特点, 从业务流程、数据结构、程序设计、部署架构等各环节进行优化分析, 采用硬件负载和软件负载配合、动静资源分离、应用集群部署、数据实时备份的方式, 实现了一个高可用、高性能、可扩展的网上竞赛答题系统.

关键词: 系统设计; 架构优化; 知识竞赛系统; 集群

Completion and Optimization of Quiz System on History of the Chinese Academy of Sciences

DU Yi-Hua, CHEN Xiong, GUO Xiao-Long

(Computer Network Information Center, Chinese Academy of Sciences, Beijing 100864, China)

Abstract: According to high page-view, high concurrency and long utility time of the quiz system, we analyzed and optimized the work flow, data structure, program design and the deployment architecture. By using hardware and software load balance, static and dynamic resource separation, application clusters deployment and real-time data backup, we finally realized a high availability, high performance, and scalable online quiz system.

Key words: system design; structure optimization; quiz competition system; cluster

为彰显中科院的历史地位和伟大成就, 营造良好舆论氛围, 推进“率先行动”计划实施, 喜迎建院 65 周年, 2014 年 10 月中国科学院组织院史知识竞赛活动. 活动面向国内外所有关心中科院改革创新发展的人士, 以线上、线下结合的形式举行. 线上设立官方网站和在线答题, 线下进行相关专题报告和举办决赛.

1 系统功能需求与性能要求

1.1 功能性需求

网上竞赛共 7 天时间, 参赛者可随时注册, 注册成功后, 系统从题库中随机抽取 300 道试题并随机排序, 参赛者可以一次性或分多次提交答案进行答题; 系统实时展示赛况、难题排行、用户排行, 提供成绩分享、试题推荐、参与奖评选等功能.

1.2 非功能性需求

系统需适于不同终端、不同年龄段用户的友好操作, 活动期间不允许出现中断、无法访问、响应速度慢等情况. 根据活动组织力度, 预计活动期间有数十万人注册参赛, 每天有数万用户登录在线答题、查看得分排名等操作, 需重点解决安全稳定和高并发的性能问题.

2 常规实现方式与问题

2.1 运行环境与问题

开发语言采用 Java, 使用开源框架 SSH2, 数据库采用 Oracle, Web 容器采用 Tomcat, 按照普通应用系统, 采用一台应用服务器和一台数据库服务器的方式搭建运行环境.

此架构可以满足访问分布较均匀、并发峰值有限的常规需求, 面对访问集中、高并发、需确保高可用的需求, 就显得单薄和不可用, 单节点应用无法支持过高的会话数, Tomcat 解析静态资源性能不佳, 数据库压力大, 应用及数据库均缺少备用节点有意外故障风险.

2.2 实现方式与问题

从共 500 道题的题库中随机选取 300 道随机排列

① 基金项目: 中国科学院计算机网络信息中心一三五规划重点培育方向专项(CNIC_PY_1412)

收稿时间:2014-12-01;收到修改稿时间:2015-01-29

给用户，实现很容易，但支持用户不一次答完，则有不同方法.

基础工作: 先创建用户信息表、系统题库表等. 用户信息表中除参赛人员基本信息外另有已答题数、当前得分等字段，系统题库除题目、选项、答案外，另有被选中次数、被回答次数、被答错次数等字段.

实现方式一: 以用户题库表为主实现. 创建用户题库表，字段主要有: 明细编号、用户编号、题目编号、用户答案、是否正确、答题时间等.

在用户注册成功时系统一次性随机抽取 300 道题随机排序，并按顺序插入记录到用户题库表，用户答题提交时更新用户题库表中的相应字段(是否正确、答题时间)，未回答的题目可供继续答题或下次登录答题. 根据用户题库表中回答情况，系统统计答题数、得分排名、出错率高的题目等.

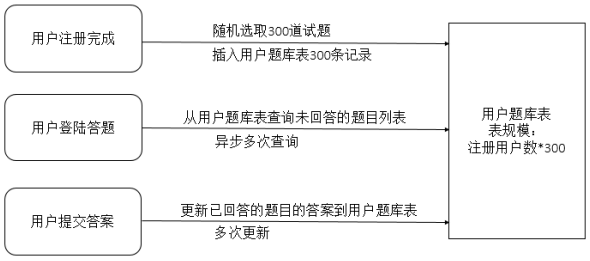

图 1 实现方式一的流程示意图

实现方式二: 以用户答题明细表为主实现. 创建用户答题明细表，字段主要有: 明细编号、用户编号、题目编号、用户答案、是否正确、答题时间、回答 ip 等.

每次从系统题库中随机取一定题目(如 30 题，通过内存运算/不入库)供用户回答，提交时只保存已做题目的答题情况(题号、是否正确)到答题明细表，空缺未答的按未做处理，继续答题或下次登录答题时，则从系统题库中过滤其已做题目后再随机选取，直到总共答题记录数到 300. 根据答题明细表再随机选题、统计答题数、得分排名.

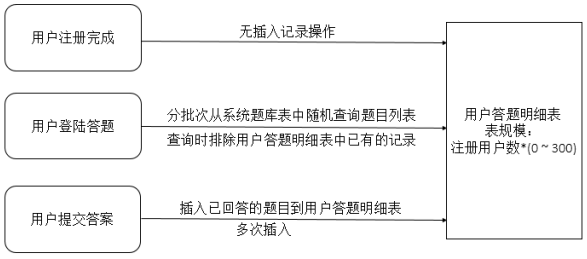

图 2 实现方式二的流程示意图

以上两种方式都是按常规思路进行的设计，均逻辑清晰，但当用户量大、并发高的情况下会存在性能问题.

方式一中，按注册用户数 100 万计算，用户题库表将有 3 亿多条记录，用户登录时查询出待做题目、答题提交时分别更新答题状态、实时从其中计算得分排名等，在访问量大的情况下会对数据库造成非常大的压力，响应速度将很慢.

方式二中，答题明细表中只记录用户已回答题目情况，避免产生部分用户注册后未全部答题而产生的无效记录，数据量相对较少，但在每次在继续答题、从系统题库表中随机抽取题目时，需与答题明细表中其已答题目进行比较排除，算法较复杂，在访问量大和答题明细表数据量大的情况下，应用服务器和数据库服务器压力均很大，将严重影响性能和用户体验.

3 系统优化的实现思路

考虑到按常规方式实现的性能问题，为了保障系统的稳定高效，需全面优化处理.

优化思路包括: 针对系统运行的各个环节进行瓶颈分析; 尽量简化实现流程; 避免产生数据量过大的表; 减少与数据库交互次数; 充分利用硬件资源和成熟集群解决方案[1,2].

3.1 优化业务逻辑模型，降低技术实现复杂性和难度

软件工程中，如果仅仅依据用户的需求描述去实施，而不去挖掘深层次的需求，常常会因对业务的机械或片面理解而采用并非最合理的实现方式，导致技术的复杂和困难，甚至系统应用中大量问题是在需求分析阶段就存在的隐患. 通过分析凝炼需求，在充分理解业务和不改变比赛规则的前提下，深入分析综合需求和数据要求，设计出最合适的系统逻辑模型[3]，优化开发流程.

3.2 优化设计数据表结构，提升数据读写效率

数据存储方式决定了数据应用效率. 在减少数据冗余、保持数据完整性和一致性的基本原则，进一步优化表结构，避免产生数据量过大的表，适当增加即时汇总列减少数据查询中的重复计算次数和复杂关联关系. 同时采用数据库连接池管理连接，如性能较好的开源 JDBC 连接池 c3p0，能提前将部分连接缓存在内存中，尽量减少在连接数据库过程中建立和销毁时的大量资源消耗.

3.3 动态资源静态化，减少数据库操作

动态资源指需实时从数据库中查询的数据，而静态资源指直接通过磁盘可以访问的文件，过多的数据库访问会让系统的响应速度变慢，且频繁访问查询时数据库压力变大. 将一些内容变化小的动态资源静态化，提前生成为静态文件，就能减少与数据库的交互次数，提高用户访问速度，改善系统整体性能；此外，通过开启 Apache 的缓存，将频繁访问的静态资源放入内存，可进一步提高访问静态资源的速度.

3.4 采用集群环境，保证系统高可用

集群环境可以解决单点故障问题，具有负载均衡和容错能力. Apache 和 tomcat[1,2]是较流行的开源集群软件解决方案，可保障系统的高可用和高扩展，并根据实际运行环境优化 web 容器的各项参数，以充分利用已有的硬件资源. Oracle 的 RAC(real application clusters)和 Data Guard[4]是较常用的保证数据库高可用的解决方案，其中 Data Guard(DG)是一种数据库级别的 HA 方案，具有冗灾、数据保护、故障恢复等功能.

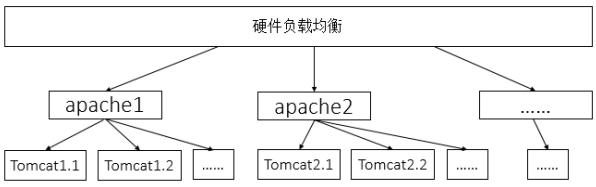

图 3 应用集群环境的系统架构

4 功能流程的优化设计

竞赛活动中，系统为每个用户从系统题库中随机抽取 300 道题目，用户可以分多次回答或重答. 根据对竞赛题目难度和用户答题所需时间等了解和分析，几乎不会有用户能一次性回答完所有题目，因此可对流程进行适当细化. 从更符合用户答题习惯和方便操作角度考虑，把 300 道题目分为 10 组，每组 30 道题目，用户可任意选择一组试题进行回答，这样回答 30 道题目的时间是在用户接受范围内，可以在不同时间段回答或重新回答完所有 10 组试题.

按分组方式设计后，整体逻辑流程上更清晰，对用户的操作更加友好，对各个环节的程序开发更可控，也便于统计查询功能和开发的优化实现.

系统整体分为三大功能模块：用户管理、答题管理、统计分享. 用户管理主要包括注册、登录、修改密码、找回密码等功能；答题管理主要包括用户查看题目分组、提交分组答案、查看当前成绩等；统计分享主要包括查看当前排名、查看系统各项统计数据等.

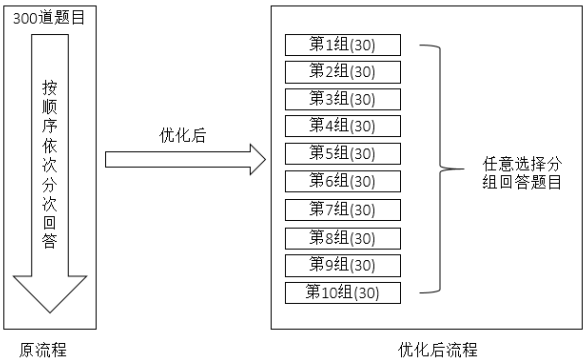

图 4 业务模型优化前后对比示意图

5 程序开发的优化实现

结合业务模型与操作流程，进一步针对性优化设计库表结构，并将部分资源静态化，尽量降低与数据库的交互次数.

5.1 数据库优化

记录每个参赛用户 300 道题及相关信息的库表数据量会很大，实现时采用按分组管理的方式， 优化设计用户题库表的表结构. 即将每组的 30 道题目对应库表中的一条记录，标识答题状态和记录答题详细信息，此种方式降低用户题库表的数据规模至原表 1/30，并在提交答题时减少更新数据表的次数.

将每组题目及答案精简为一条记录存入表，减小表数据行规模，为避免因此产生用户答题明细信息有缺失问题，系统同时将所有用户提交答题操作的详细情况以日志方式写入系统日志文件中，用于在需要时回溯分析.

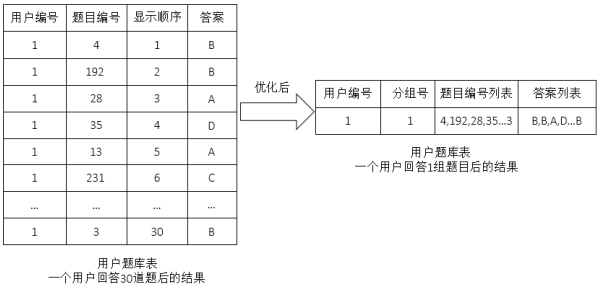

图 5 用户题库表结构的优化前后对比示意图

优化后的主要数据库表和字段如下：

系统题库表：题目编号、标题、选项 A、选项 B、选项 C、选项 D、答案、回答次数、答错次数、状态、

创建时间;

用户信息表: 用户编号、姓名、手机号、邮箱、密码(加密存储)、所在单位、状态、注册时间、最后登录时间、最后登录 ip 地址、最后修改密码时间、修改密码总次数、答题分组数、总得分;

用户题库表: 明细编号、用户编号、分组号、题目编号字符串、提交答案字符串、分组得分、最后答题时间、最后答题 IP;

优化表结构的同时, 为了尽可能提高效率, 分散磁盘 I/O, 在建表时将各张表放在独立的表空间中, 为各个常用的查询字段添加索引, 索引放在单独的表空间中, 以最大程度的提高数据库的性能.

5.2 资源静态化

用户注册时系统即为其随机分配 300 道题并且随机排序. 当每个用户注册完成后, 随机分配的题目和顺序不会再发生变化. 技术实现时, 将其按组号分别生成 json 静态文件, 按一定目录规则存储, 当用户登录后选择某组题时, 采用在前端页面通过 ajax 异步请求方式去访问对应的静态文件, 由 apache 解析文件并返回结果. 这样既增加页面展示的友好性, 也不需要去查询数据库, 在高并发的情况下可极大减轻数据库压力, 大幅提高系统性能和访问速度.

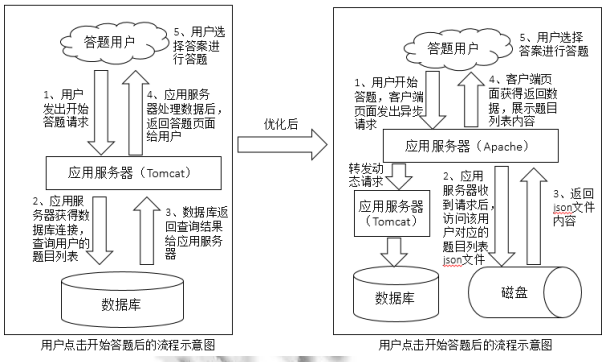

图 6　查询题目内容方式的优化前后对比示意图

5.3 查询统计轮询

大量用户高频度的访问或刷新查看一些统计信息时, 若每次直接连接数据库、即时关联运算会大量占用资源, 甚至导致系统不同用. 而部分统计数据如当前参赛人数、用户分析、得分排行、难题排行等, 并不需要绝对实时, 允许一定的滞后性.

技术实现时, 在应用中采用定时器触发统计查询, 将所有统计数据放在内存中, 每隔一定时间, 如 30 秒或 2 分钟查询数据库更新一次内存中各项数据, 所有应用均直接从内存中读取展示相应数值, 经过此轮询优化后, 各项统计功能不用再实时查询数据库, 可最大程度降低数据库访问频度和提高系统性能.

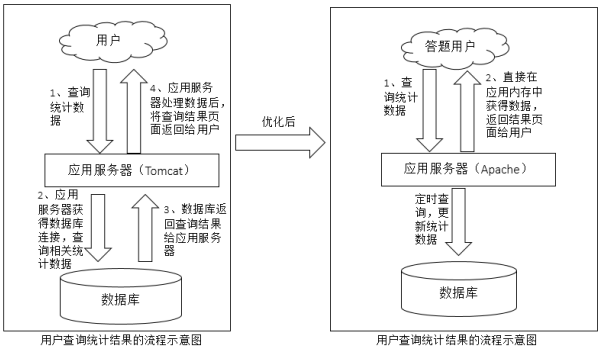

图 7　统计查询方式的优化前后对比示意图

对于在系统外其它网站, 如活动官网、中科院门户网站等的即时统计展示, 因跨域名, 系统采用开放数据查询的应用接口方式, 供通过jsonp方式异步调用和灵活展示.

6　系统架构部署

按照集群部署方案, 竞赛系统采用负载均衡设备, 搭建应用集群环境, 数据库采用 Oracle Data Guard, 实现动静资源分离、访问请求平均分配、单个应用节点故障不影响系统运行, 最大程度上保证系统的稳定性和高可用性.

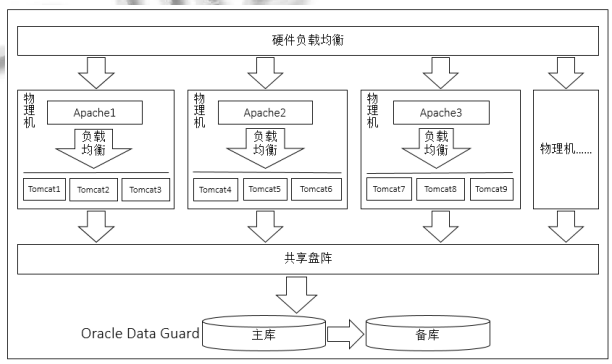

图 8　系统架构示意图

1) 采用硬件七层负载均衡设备. 设置透明代理, 开启会话粘性功能, 将所有请求根据 ip 或 cookie 平均分配到各个 apache 应用上, 并根据实际应用情况设置分配权重.

2) 搭建 apache 和 tomcat 应用集群环境. 将 apache 作反向代理和负载均衡, 由 apache 解析 html、js、json 等静态资源, 由 tomcat 解析 jsp、servlet、action 等动态请求[5]; 最大程度利用硬件资源, 在每台服务器上安装一至多个 tomcat 容器; 合理配置 apache 和 tomcat 参数, 设置较大连接数、JVM 内存等.

3) 按需增加设备扩充承载能力. 集群环境中当系统中任意应用节点出现故障, 都不影响系统的正常使用, 并可随时恢复故障节点. 部署时采用共享盘阵和三台较高性能服务器, 共搭建了 9 个应用, 根据需要可以动态增加服务器, 进一步增强承载能力.

4) 采用 Oracle Data Guard 进行数据库容灾. 将生产库作为主库, 复制一份数据库作为备库, 主库将实时生成的数据以 REDOLOG 的形式传输至备库, 保证主库和备库的数据实时性和一致性, 若主库出现问题, 能平滑切换至备库.

7 系统性能测试

为了验证最终实现系统的性能效果, 在正式上线前对系统进行全面压力测试, 测试工具采用 LoadRunner[6], 测试指标包括平均响应时间、每秒事务总数、事务通过率. 测试项目为针对用户注册、用户登录、用户提交答题三个核心功能点.

用户提交答案的测试结果如图 9, 当并发数为 500 时, 平均响应时间都在 3 秒左右, 系统具有良好的性能表现; 当并发数为 3000 时, 仍然可以保证较高的事务通过率(90%以上), 系统最少可支撑 3000 并发.

依据并发数公式 C=(nl)/T(C 为并发数, n 为同时在线人数, l 为平均在线时长, T 为提供服务时长), 预估竞赛系统的用户平均在线时长为 30 分钟, 大部分都在白天 8 小时访问系统, 可以算出同时在线人数为: n=CT/l=3000*8/0.5=48000(人); 根据巴莱多定律(即二八定律)可以推算出同时在线人数: (n*80%) /20%= 48000*4= 192000(人).

并发用户	平均响应时间(S)	每秒事务总数(个)	事务通过率(%)
100	1.150	24.753	100%
500	3.010	29.386	99.85%
1000	5.502	30.519	99.19%
3000	8.727	28.576	96.23%

图 9 用户提交答题的测试结果

当系统有 19 万人同时在线时, 并发数可能达到 3000, 此时系统仍可以提供较稳定的服务.

8 结语

中科院院史知识竞赛活动(http://yszsjs.cas.cn)已在 2014 年 10 月进行完毕, 实际上线时根据需要调整为每人回答 210 道题分为 7 组, 活动得到社会公众的积极参与, 系统有效支撑和顺利保障了活动进行.

参考文献

1 The Apache Software Foundation.The Number One HTTP Server on the Internet. http://httpd.apache.org. [2014-10-01].

2 The Apache Software Foundation.Apache Tomcat 7. http://tomcat.apache.org/. [2014-10-01].

3 王继成,高珍.软件需求分析的研究.计算机工程与设计, 2002,23(8):18–21.

4 Oracle Corporation. Data Guard Concepts and Administration. http://docs.oracle.com/cd/B19306_01/server.102/b14239/concepts.htm. [2014-10-01].

5 边清刚,潘东华.Tomcat 和 Apache 集成支持 JSP 技术探讨. 计算机应用研究,2003,20(6):12–14.

6 桑圣洪,胡飞.性能测试工具LoadRunner的工作机理及关键技术研究.科学技术与工程,2007,7(6):1019–1022.