基于缓存的分布式统一身份认证优化机制研究

杨冬菊 冯 凯
(大规模流数据集成与分析技术北京市重点实验室 北京 100144)
(北方工业大学云计算研究中心 北京 100144)

摘 要 企业在进行应用系统集成时,普遍使用独立的身份认证系统来实现平台中身份信息的交换和共享。如何应对高并发、大用户流量的用户请求,是保障认证系统稳定、高效运行的重要问题。针对单认证中心负载过重,容易出现单点失效及系统响应慢的问题,提出了将认证服务器集群化的方案;将认证票据存储在缓存使得多个认证节点共享认证信息,并将重要且频繁使用的数据预存到缓存中以提高响应速度;结合复杂多样的用户行为提出了基于Hybrid的多因素缓存替换算法。实验结果表明,所采用的基于缓存的分布式认证架构能够保证系统的稳定性,提高系统的响应速度,改进的缓存替换算法提高了缓存命中率。

关键词 应用系统集成,身份认证,高可用性,缓存机制,缓存替换算法
中图法分类号 TP301 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.03.049

Distributed and Unified Authentication Optimization Mechanism Based on Cache

YANG Dong-ju FENG Kai
(Beijing Key Laboratory on Integration and Analysis of Large-scale Stream Data, Beijing 100144, China)
(Research Center for Cloud Computing, North China University of Technology, Beijing 100144, China)

Abstract When the enterprise integrates the application system, it is a common practice to use the independent authentication system to exchange and share the identity information of the platform. How to deal with user requests with high concurrency and large user traffic is an important issue to ensure the stable and efficient operation of the authentication system. In view of the overload of single authentication center, the single point failure and the slow response of the system, this paper proposed to cluster the authentication server. The authentication ticket is stored in the cache so that multiple nodes can share authentication information, and the important and frequently used data can be pre-fetched as cache to improve response speed. This paper proposed a multi-factor cache replacement algorithm based on Hybrid combining the complex and diversified user behavior to improve the effectiveness of data replacement. The experimental results show that the optimized distributed authentication architecture can guarantee system stability and improve system response speed, and the multi-factor cache replacement algorithm based on Hybrid can improve cache hit ratio.

Keywords Application system integration, Identity authentication, High availability, Cache mechanism, Cache replacement algorithm

1 引言

随着企业内部各类信息系统的增多[1],各个系统的访问机制和安全策略各异,以单点登录实现异构系统集成是目前比较常用的集成方式[2]。CAS系统[3]是耶鲁大学开发的一款开源单点登录系统,其由于开放性、可靠性、易用性等特点被广泛使用。采用CAS系统集成方案,能够实现各信息系统的跨域访问。但是,当大量用户并发访问时,如何保障系统可靠稳定以及其快速响应是目前亟需解决的问题。

以本课题组承担的某电子政务项目为例,需要集成的业务系统多达上百个,用户的总数量已达到百万,在访问高峰期间,瞬时的认证请求数量可达到上万,对认证中心造成了极高的请求压力,保证认证中心的稳定运行至关重要。利用Nginx建立CAS集群实现统一身份认证[4],通过多节点任务调度提高系统响应时间和应对高并发用户,从而保证认证系统的稳定运行,是目前比较常用的一种分布式解决方案。

但是直接使用Nginx代理,仅将认证用户与集群中的认证服务器进行点对点认证,将导致当该服务器失效后用户登录状态丢失。因此如何实现在集群中共享Session和票据(TGT)信息、解耦用户认证信息与认证节点的绑定是分布式统一身份认证系统中须解决的首要问题。另外,在认证服务器进行认证的过程中,须多次与数据库进行交互,所以磁盘的读写性能对系统响应时间的影响较大。因此,如何利用缓存来提高系统的响应速度和并发效果是需要解决的另一重要问题。

到稿日期:2017-06-18 返修日期:2017-08-01
杨冬菊(1975-),女,博士,副研究员,主要研究方向为服务计算、行业资源中心关键技术、创建模式及其在领域中的示范应用,E-mail:yangdongju@ncut.edu.cn(通信作者);冯 凯(1993-),男,硕士,主要研究方向为服务计算、行业资源中心关键技术,E-mail:457030770@qq.com。

此外，针对复杂多变的用户行为，设计合适的缓存替换算法是保证系统稳定运行的关键。

针对以上问题，本文提出了一种基于缓存的分布式统一身份认证机制，使用 CAS 协议进行身份的集中认证；并在 CAS 协议上进行架构的修改，将认证服务器由单台扩展为多台；使用 Redis[5] 缓存数据库共享 Session 和 TGT 信息，提高系统的可用性。利用 Redis 将用户验证信息缓存到内存中，提高用户的查询速度，保证系统被高效使用。为了保证缓存的性能，应对复杂的用户种类，设计并实现了一套基于 Hybrid 的多因素缓存（Hybrid-MF）替换算法。实验表明，本文提出的基于缓存的分布式统一身份认证机制能够有效提高系统的响应时间、应对并发压力。

本文第 2 节对国内外在身份集中认证方面进行优化的相关工作进行概述；第 3 节介绍认证服务架构设计；第 4 节设计并实现缓存替换算法；第 5 节对本文所提方法进行实验测试；最后总结全文。

2　相关工作

随着云计算和面向服务的企业应用系统的发展，企业在集中管理多个信息系统时，单点登录系统的实现是重点[6]。如何减少服务器的负担以及提高响应时间成为了研究高并发问题的重点[7]。目前针对高并发用户访问时服务器负载过重的问题，集中对应用系统架构进行研究[7,11]；针对如何提高系统响应速度的问题，集中对缓存进行研究[12-17]。针对缓存的研究中，一方面是对缓存的实现方法和缓存内容的研究，另一方面是对缓存替换算法的研究。

实现单点登录有多种解决方案，文献[8]总结了目前实现单点登录的实现机制：1）基于口令的用户身份认证，其缺点是安全性较低，若使用加密技术会使系统增加大量的计算开销；2）基于 S/key 的认证，使用起来比较繁琐；3）Kerberos 身份认证，但 Kerberos 存在安全缺陷[9]，并且这种协议的使用需要复杂的设置。文献[10]重点分析了基于 CAS 协议的单点登录认证方案，该协议使用票据传递认证信息，传输时使用 https 协议传输，拥有良好的安全机制，较好地解决了集中式单点登录跨域的问题，只需要对接入的客户端做少量配置文件注入即可完成登录工作。

针对服务器负担过重的问题，文献[7]优化了 web 服务器的配置参数以及中间件的内存参数，但是没有考虑到单机失效会导致系统无法正常使用的问题。文献[11]分析了在用户大规模并发时系统集群化的必要性，使用负载均衡技术来提高系统集群的高可靠性、高伸缩性，比较全面地分析了目前的负载均衡技术，包括软/硬件的负载均衡、本地/全局的负载均衡以及网络层的负载均衡技术，使用负载均衡技术来提高系统的整体性能和吞吐量。

文献[12]分析了 Redis 内存数据库的特点及应用。Redis 支持多种数据结构，不同级别的磁盘持久性为缓存的实现提供了适用的工具。在缓存内容方面，文献[13]针对数据库中的数据使用了缓存技术，减少了数据库的访问次数，并使用页面级缓存技术缓存 Web 页面。

在缓存替换算法研究中，LRU 和 LFU 算法[14] 只考虑时间或访问频率，仅在特定的应用下表现出良好的性能；基于对象大小的 Size 算法[15]，可能会使小对象过久驻留缓存。文献[16]提出了一种最小驻留价值的缓存替换算法，该算法结合对象访问频率和大小缓存计算驻留价值，对缓存的价值进行了综合计算，但是其没有考虑缓存的使用时间和物理资源的使用情况。文献[17]提出了基于蚁群算法的缓存替换算法，该算法综合考虑了缓存数据的大小、被命中次数和其在缓存中存留的时间，考虑因素较全面，但计算量较大，影响替换效率。

综上所述，身份集中认证研究大多数只在实现层面，而没有考虑并发问题。综合上述文献，借鉴对 Web 网站并发问题的研究，本文在单点登录系统 CAS 的基础上，改进认证服务的架构，增加认证服务器的数量，并在系统查询时使用缓存技术。在缓存替换算法设计上提取上述文献的优点，规避缺点，应对类型复杂的用户登录，提出 Hybrid-MF 算法。

3　系统架构

3.1　系统架构描述

本文将认证服务器进行了集群化改造，增加了 Nginx 反向代理服务器，将用户的认证请求通过轮训策略分发到 CasServer 集群中的服务器上，在查询过程中使用 Redis 进行缓存，缓存处理时使用 Hybrid-MF 算法进行缓存替换。系统架构如图 1 所示。

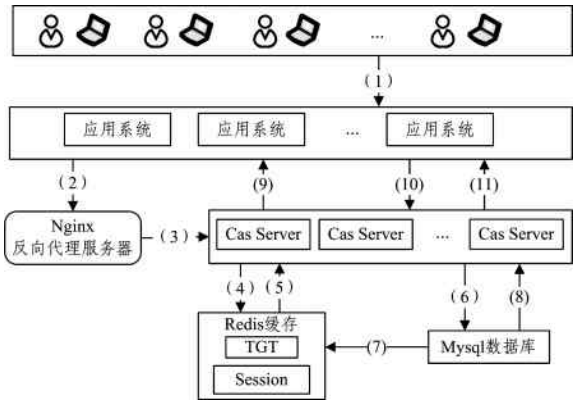

图 1　系统架构图

Fig. 1　System architecture diagram

认证流程如下：

1）用户访问应用系统，应用系统使用 Session 验证用户的登录状态，若成功验证则进入应用系统，否则执行步骤 2）。

2）重定向用户访问请求，对 CasServer 发起认证。

3）重定向用户的请求，通过 Nginx 服务器按照轮训策略分发到 CasServer 上。

4）通过 Redis 缓存查询用户信息。

5）Redis 返回用户的查询结果，若 Redis 中存在查询结果，则执行步骤 9），否则执行步骤 6）。

6）通过数据库查询用户信息，若查询到用户信息，则执行步骤 7），否则认证失败，结束。

7）将用户信息作为缓存对象写入 Redis 缓存中，并将对应的缓存价值信息写入缓存中，执行步骤 8）。

8）返回认证信息到 Cas Server 中。

9）认证成功后，Cas Server 会生成 Ticket Granting Ticket（TGT）票据，并生成 Service Ticket（ST）票据，请求带着

ST 信息重定向回到应用系统。

10)重定向用户请求,通过请求中的 ST 参数与 CasServer 中的 ST 进行验证,查看票据是否有效。

11)返回 ST 认证结果。若认证成功,则进入应用系统,否则认证失败。

3.2 系统架构的实现

系统架构的实现主要包括以下 3 部分:分布式认证集群的实现、反向代理的实现、缓存查询的实现。

3.2.1 分布式认证集群的实现

在实现分布式认证中心的过程中主要的工作有两点:Session 同步和验证 TGT 同步。

1)Session 同步

CAS 协议的登录过程使用 Session 进行登录信息的传递及认证。使用 Redis 修改 Session 的存储策略,将原先存储在 tomcat 的 Session 保存在 Redis 中,保证跨域共享 Session,即使用户的访问请求到不同认证服务器上也可以进行认证。将 Session 以 key-value 的形式存储在 Redis 中,以 jsessionID 作为 key,以 Session 对象为 value 存到缓存中。

2)票据同步

TGT 中记录了用户登录成功后的信息,将 CAS 认证时生成的 TGT 集中存储,以保证用户在各个应用系统的登录状态同步。实现 TGT 对象的序列化,以便 Redis 将缓存中的快照映射在硬盘文件中。存储时以 TGT 的 ID 为 key,以 TGT 对象为 value 存储在 Redis 中。

3.2.2 反向代理模块实现

增加反向代理服务器,将用户的认证请求分发到多个认证服务器上。Nginx 是一个 HTTP 和反向代理服务器,相比于 Apache,其占用更少的内存及资源,处理请求是异步非阻塞的,在高并发下能够实现低资源、低消耗、高性能。本文使用 Nginx 反向代理服务器配置轮询分发策略,将用户的认证请求按照时间顺序分发到不同的认证服务器。

3.2.3 缓存模块实现

使用 Redis 将缓存对象、缓存价值对象 CacheValueObject(CVO)和缓存价值对象集合 CacheValueObjectMap(CVOM)存储在内存中。缓存对象为用户认证时的用户信息;CVO 存储了缓存对象的价值信息,在缓存容量到达阈值时进行缓存清理。设定 CVOM 的存储容量,并在集合中保存了所有的 CVO。在插入缓存时,如果缓存容量达到阈值,使用 Hybrid-MF 算法替换掉价值最小的缓存对象。

本文中缓存的存储内容是在认证过程中会频繁使用的 3 类信息:1)用户的账号和密码;2)用户的个人信息;3)用户的权限信息。

4 缓存替换算法的设计

本文在进行用户认证时使用 Redis 缓存进行信息查询。Redis 为了保证高可用性,在使用过程中会定期生成快照。随着存储在 Redis 中的内容增加,内存占用和生成快照的时间也会相应增加,内存中插入过多的缓存数据会降低缓存的使用效率,因此使用缓存替换算法来解决该问题。Redis 默认的替换算法为 LRU 近似算法,设计简单且随机性强,但会损失一定的缓存命中率。因此根据集成系统中复杂多变的用户行为,提出 Hybrid-MF 算法。

Hybrid-MF 算法的基本思想如下:给定有限的缓存对象集 U,每个对象 $t_i \in U$,每个对象的缓存价值为 v_i,设定集合 U 的最大长度为 L,使得对象个数的总和不超过 L。

当第 n 个对象需要插入到缓存中时,判断缓存中是否有剩余空间存储新的缓存对象。

$$\sum_{i \in n} t_i \leqslant L \tag{1}$$

当式(1)不满足时,说明缓存集合中已无剩余空间存储新的缓存,则选择满足式(2)的缓存对象进行替换。

$$\min\{v_i\} \tag{2}$$

4.1 算法设计

4.1.1 Hybrid 算法模型

Hybrid 缓存替换算法的基本思想是:在插入缓存对象时,根据缓存的大小、存储缓存的网络情况、缓存的使用频率来计算缓存对象的价值,在缓存容量达到设定的阈值时,替换掉价值最低的对象。缓存对象的价值公式如下:

$$\frac{(C_S + K_1/b_s) * (fr_f)^{K_2}}{size_f} \tag{3}$$

公式的符号描述如表 1 所列。

表 1 Hybrid 算法中公式的符号描述

Table 1 Formula description of Hybrid algorithm

符号	描述
C_s	客户端与服务器 s 的连接时间
b_s	服务器 s 的带宽
fr_f	缓存对象的访问频率
$size_f$	缓存对象 f 的占用大小
K_1	带宽的权值
K_2	缓存对象使用频率的权值

该算法考虑到了网络带宽、缓存的访问频率等多种因素。本文需要缓存的对象只有字符串对象,忽略对象的 Size 因素;该算法没有考虑缓存的最后一次使用时间,若缓存仅在某一时段频繁使用,将造成缓存垃圾且无法清除。基于以上原因,本文在该算法上进行了改进。

4.1.2 缓存替换算法的设计

本文在 Hybrid 缓存替换算法的基础上提出了一种 Hybrid-MF 算法。该算法综合了缓存在读取时的性能、缓存的访问频率以及缓存的访问时间因素,计算出了缓存的价值,在替换缓存时将缓存价值最小的替换出去。

结合本项目的研究内容,本文缓存的价值因子如表 2 所列。

表 2 Hybrid-MF 算法的符号描述

Table 2 Formula description of Hybrid-MF algorithm

符号	描述
b_s	取得缓存对象的服务器的带宽
fr_f	缓存对象的访问频率
T	缓存对象的最后一次访问时间
K_1	带宽的权值因子
K_2	缓存对象访问频率的权值因子
C	认证服务器的数量

本文缓存的内容都为字符串对象,缓存对象的 Size 相同,因此不考虑该因素。为了减少计算量,本文忽略客户端与服务器的连接时间 C_s,在网络性能上只考虑服务器带宽 b_s。服务器 s 的带宽越大,查询速度就越快,因此在数据库中进行

查询的速度就越快，它的价值就越小；C 为认证服务器的总数量，假设用户访问认证服务器的概率是相同的，若须将带宽的价值因素缩小，则须将访问的概率缩小为认证服务器数量 C 的倒数。

结合 LRU 算法的思想，式(4)中加入了最后一次的使用时间 T。如果是比较久远的缓存就认为它的价值比较低，如果时间与现在较近，则认为它的价值相对较高。直接使用 T，T 会随着时间直线增长，从而使整个公式更倾向于最后一次登录时间，当最后一次登录时间的差别较大时，公式的效果就会类似于 LRU。因此对 T 取 e 的对数，缩小 T 对整个公式的权重。

K_1 和 K_2 也是公式中重要的一部分，通过调整 K_1 和 K_2 的大小，来平衡缓存的访问次数和带宽传输速率两个因素，防止局部条件变化而导致的价值失衡。通过多次实验，计算常数 K_1 和 K_2 的大小以及两个常数之间的关系，使缓存的命中率达到较好的效果。

最后经过对缓存的多个因素的考虑，缓存价值替换式如下：

$$(\frac{K_1}{b_S} * \frac{1}{C}) * (fr_f)^{K_2} * \ln(T) \tag{4}$$

4.2　缓存价值对象的设计

缓存价值对象 CVO 包含了缓存对象的信息，具体描述如下：

1）key：对应缓存对象的 key；

2）fr_f：缓存的访问次数；

3）T：缓存的最后一次访问时间；

4）b_s：上一次缓存访问服务器的带宽；

5）v：综合计算出的缓存价值。

将该对象的设计用于存储缓存对象的各项信息，在需要缓存替换时，对缓存价值进行排序，返回价值最小的缓存价值对象的 key，删除对应缓存对象 t。

4.3　算法实现

以用户登录验证账号密码为例，算法的实现过程如下：

1）在第一次使用 Redis 缓存时，由于缓存中没有任何用户信息的数据，在认证服务器启动前随机将数据库中的部分用户账号和密码通过替换算法写入到缓存中，进行缓存预热，以备登录验证时使用，从而提高正式使用时的缓存命中率。

2）用户在进行身份认证时，首先查询缓存中是否存在数据。

3）若缓存中存在用户名和密码信息，则更新缓存中的缓存价值对象 CVO 和缓存价值对象集合 $CVOM$，并将信息取回，否则执行步骤 4）。

4）若在缓存中没有找到数据，则在数据库中查询数据。

5）若在数据库中也没有查询到数据，则此次查询失败，否则执行步骤 6）。

6）查看缓存价值对象集合中的 $CVOM$ 是否已满，若已满则执行步骤 7）；否则执行步骤 8）。

7）根据缓存价值对象 CVO 的价值对缓存价值对象集合 $CVOM$ 进行排序，查找价值最小的缓存价值对象，并返回缓存价值对象的 key，删除对应的缓存对象及该缓存价值对象，更新缓存价值对象集合。返回步骤 6）。

8）将用户的信息插入到缓存中，创建新的缓存价值对象 CVO，并将其增加到缓存价值对象集合 $CVOM$ 中，返回用户信息。

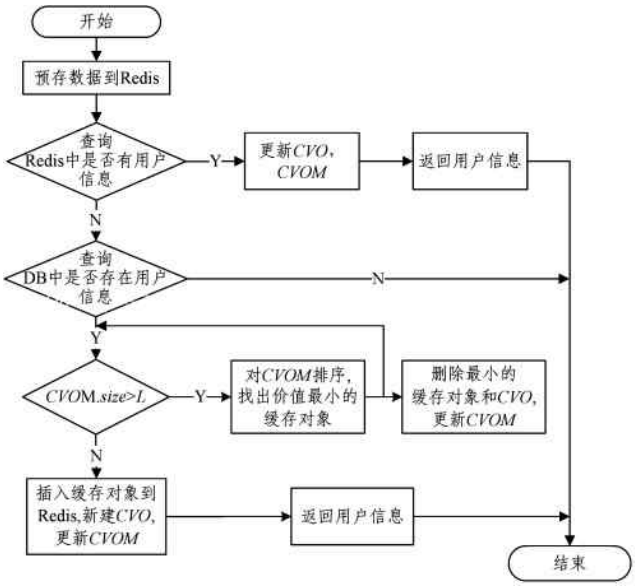

图 2　算法实现流程

Fig. 2　Flow chart of algorithm implementation

5　实验与分析

5.1　实验目的

为了验证集群化的系统稳定性，对比加入缓存前后的认证性能以及验证本文提出的缓存替换算法对缓存命中率的影响，设立了 4 组实验。实验 1：在认证集群正常运行时，关闭部分认证服务器，测试集群化系统架构认证服务的稳定性。实验 2：通过比较页面响应时间、单位时间内处理的 HTTP 事务数，来对比系统加入缓存前后对系统访问的性能差异。实验 3：针对 Hybrid-MF 算法的参数设置对照实验，找到 K_1 和 K_2 参数组合的最佳范围。实验 4：设置 Hybrid-MF 算法与传统 LRU 算法和 LFU 算法的对照实验，比较在两种算法下的缓存命中率。

5.2　实验环境

实验环境如表 3 所列。

表 3　CasServer 测试环境

Table 3　CasServer test environment

序号	内存/GB	CPU 参数	运行服务
1	8	2.0GHz 8 核	CasServer 服务器 1
2	8	2.0GHz 8 核	CasServer 服务器 2
3	8	2.0GHz 8 核	LoadRunner 测试软件
4	4	2.0GHz 4 核	Nginx-1.10.1
5	4	2.0GHz 4 核	Redis-64.3.0
6	8	2.0GHz 8 核	Mysql-5.7.14

5.3　实验描述及分析

实验 1　通过运行测试脚本，测试用户登录系统后认证集群中的部分认证服务器失效时，用户的登录状态是否正常。实验步骤如下：

1）在脚本中设置 10000 个用户进行登录操作，设置思考时间 Think Time 为 5min，然后进行用户退出操作。

2)用户登录成功后,将一台 Cas Server 服务器关闭,模拟用户继续访问不同的应用系统,然后退出应用系统,观察用户在此过程中是否可以正常访问,正常注销。

经过实验测试,当用户登录成功后,关闭集群中的某台认证服务器,由于用户的 Session 信息和登录票据信息存储在 Redis 服务器中,各认证服务器共享了认证信息,当用户到认证系统进行认证时,会根据存储在 Redis 中的票据验证身份,用户可以正常地进行系统访问和身份注销。

实验 2 使用 LoadRunner 测试软件进行压力测试,为录制脚本创建模拟场景。实验方法如下:

1)设置场景的运行状态。设置运行脚本的运行持续时间以执行完登录脚本为结束,在集合点释放 100 个虚拟用户。

2)设置用户的初始数量为 1000,当虚拟用户成功数量达到初始化用户数量 90%以上时,记录成功的并发数量。

3)以 1000 为步长增加虚拟用户的数量,记录在不同用户量下测试的情况。

实验结果如图 3—图 5 所示。

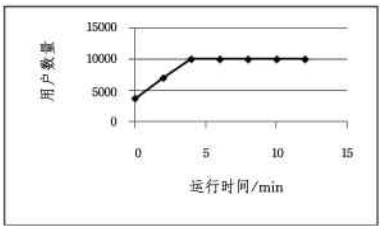

图 3 认证集群稳定性测试图

Fig. 3 Test chart of certified cluster's stability

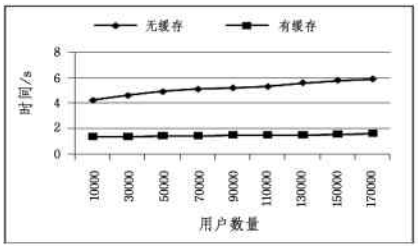

图 4 平均事务响应时间的对比

Fig. 4 Comparison of average transaction response time

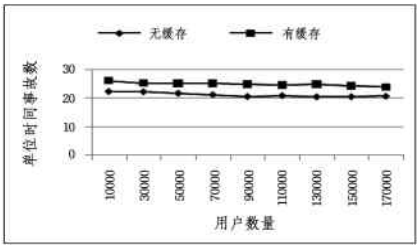

图 5 单位时间处理事务数量的对比

Fig. 5 Comparison of the number of transactions per unit time

实验结果表明,加入缓存的认证集群比未加入缓存的认证集群在单位时间处理的事务数量方面有所提高,显著降低了平均事务的响应时间。在无缓存版本的认证服务器的测试过程中,由于频繁访问数据库,消耗了大量的时间,增加了响应时间。在加入缓存版本的认证服务器中使用了缓存查询,大大减少了查询时间,提高了响应速度。

实验 3 设置 Hybrid-MF 算法的参数对照实验,通过实验测试来寻找参数的最佳组合数值范围,以保证经过算法计算后的缓存命中率最优。具体设置如下(设置算法参数初始值分别为 K_1 和 K_2($K_1=1$,$K_2=1$));1)保持 K_1 不变,K_2 从 1 开始,以 2 为步长递增,每次取值重复进行 5 次,记录实验的平均命中率;2)保持 K_2 不变,K_1 从 1 开始,以 2 为步长递增,每次取值重复进行 5 次,记录实验的平均命中率;3)对以上各参数组合情况的数据进行汇总和分析,总结各个参数的最佳取值范围以及各参数之间的组合优化设置。实验结果如表 4 所列。

表 4 命中率对比结果

Table 4 Comparison of hit rate

序号	K_1	K_2	命中率/%
1	1	1	40.08
2	1	3	40.12
3	1	5	40.30
4	1	7	39.59
5	3	1	40.02
6	3	3	40.12
7	3	5	40.12
8	3	7	39.92
9	5	1	40.18
10	5	3	40.07
11	5	5	40.30
12	5	7	40.30
13	7	1	40.10
14	7	3	40.00
15	7	5	40.10
16	7	7	39.93

由实验可得,通过调整网络带宽权值 K_1 可以提升缓存的价值,令访问带宽较高的缓存拥有更高的价值,但过分依赖带宽会导致一些无用的缓存,因此当 K_1 的取值为[1,5]时,缓存的平均命中率上升;当 K_1 的取值为(5,7]时,平均命中率开始下降;在 K_1 不同的情况下,K_2 取值为 5 时,缓存命中率达到最大值。因此当 Hybrid-MF 公式中的参数 K_1 取值为 5、K_2 取值为 5 时,可以使缓存命中率达到最优。

实验 4 设置 Hybrid-MF 算法与传统的 LRU 算法和 LFU 算法的对照实验,对比 Hybrid-MF 算法与传统 LRU 算法和 LFU 算法的缓存命中率,观察算法的性能。具体设置如下:设置账号数量总数为 50000,随机访问用户账号,累积查询缓存 100000 次;设置缓存初始容量大小为 15000,设置对照组实验,以 5000 的步长增加缓存容量大小。分别用 3 种算法进行 5 次测试,计算命中率的平均值。对比结果如图 6 所示。

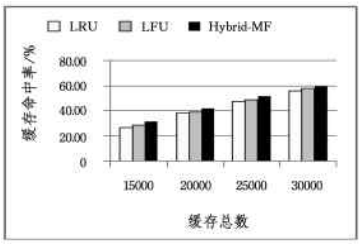

图 6 缓存命中率对比

Fig. 6 Comparison of cache hit rate

(下转第 310 页)

tion module for a fault tolerant NoC operation[C]// International Symposium on Quality Electronic Design. IEEE, 2015: 284-288.

[26] DAI L, SHANG D, XIA F, et al. Monitoring circuit based on threshold for fault-tolerant NoC[J]. Electronics Letters, 2010, 46(14): 984-985.

[27] YUAN W L. A Fault-Tolerant Design for TSV and Crossbar in 3D NoC [D]. Hefei: Hefei University of Technology, 2013. (in Chinese)
袁吴铃. 3D NoC 中 TSV 和交叉开关的容错设计[D]. 合肥: 合肥工业大学, 2013.

[28] OUYANG Y M, WANG Q, LIANG H G, et al. Link adaptive fault-tolerant method based on fault granularity partition in NoC [J]. Journal of Electronic Measurement and Instrumentation, 2015(8): 1102-1113. (in Chinese)
欧阳一鸣, 王悄, 梁华国, 等. 基于故障粒度划分的 NoC 链路自适应容错方法[J]. 电子测量与仪器学报, 2015(8): 1102-1113.

[29] OUYANG Y M, ZHANG Y D, LIANG H G, et al. Fault-Tolerant Router for 3D NoC Based on virtual channel Fault Granularity Partition[J]. Journal of Computer Research and Development, 2014, 3(9): 1073-1076. (in Chinese)
欧阳一鸣, 张一栋, 梁华国, 等. 基于虚通道故障粒度划分的 3D NoC 容错路由器设计[J]. 计算机研究与发展, 2014, 3(9): 1073-1076.

[30] OUYANG Y M, HE M, LIANG H G, et al. A Fault-Tolerant Architecture Design of Fault-Aware RVOQ in Three-Dimensional Network-on-Chip[J]. Journal of Computer-Aided Design & Computer Graphic, 2015, 27(1): 192-200. (in Chinese)
欧阳一鸣, 何敏, 梁华国, 等. 3D NoC 中故障感知的 RVOQ 容错架构设计[J]. 计算机辅助设计与图形学学报, 2015, 27(1): 192-200.

[31] ZHANG S J, HAN G D, SHEN J L, et al. Fault-tolerant Routing Algorithm of NoC Based on Buffer Reuse of Faulty Links [J]. Journal of Computer-Aided Design & Computer Graphics, 2014, 26(1): 131-137. (in Chinese)
张士鉴, 韩国栋, 沈剑良, 等. 基于故障链路缓存再利用的 NoC 容错路由算法[J]. 计算机辅助设计与图形学学报, 2014, 26(1): 131-137.

(上接第 304 页)

由图 6 可知, 随着缓存容量的增加, 缓存的命中率也随之增加; 并且 Hybrid-MF 算法的缓存命中率与传统的 LRU 算法和 LFU 算法相比, 在缓存容量不同的情况下, 其缓存命中率均有一定的提升, 针对复杂的用户种类拥有更优的性能。

结束语 本文研究了多应用系统中身份认证的优化问题, 提出了认证中心集群化的架构优化方案; 使用数据缓存技术和缓存替换算法方案, 解决了大规模访问量下的高可用和高效问题。经过实验结果验证表明, 改造后的认证系统提高了事务的响应时间, 实现了高性能认证; 在认证集群中共享认证信息, 增加了系统的稳定性; 针对复杂的用户种类, 本文提出的 Hybrid-MF 算法保证了其在查询时拥有较好的缓存命中率。

参 考 文 献

[1] WANG Z R, LIU Z T, CAO Y. A Lightweight solution forcross-domain single sign-on [J]. Computer Applications and Software, 2013, 30(7): 268-270.

[2] CHEN M. Research and Application of Heterogeneous Systems Integration Technologies in Education Systems [D]. South China University of Technology, 2016.

[3] ZHAO V F. Design and implementation of the unified authentication platform based on CAS[J]. Journal of Yunnan University, 2013, 35(22): 165-168.

[4] QI F S, TIAN C Y, WEI H. The Design of High Available Single Sign-On Server of Nginx-Based[J]. Applied Mechanics and Materials, 2013, 241-244: 2411-2416.

[5] ZHU J, HU B, SHAO H, et al. Research of Lightweight Vector Geographic Data Management Based on Main Memory Database Redis[J]. Journal of Geo-Information Science, 2014, 16(2): 165-172.

[6] XU F H, CHEN X, LONG D. New distributed multi-user single sign-on system model [J]. Application Research of Computers, 2012, 29(9): 3355-3357, 3364.

[7] WANG Y N, WU H R, HUANG F. Optimization analysis and research of high concurrency Web application system performance[J]. Computer Engineering and Design, 2014, 35(8): 2976-2981.

[8] REN C L, LI Z X, NIU X X, et al. Single Sign-On Model in Distributed Network [J]. Computer Systems & Applications, 2011, 4(2): 138-145.

[9] BELLOVIN S M, MERRITT M. Limitation of Kerberos authentication system[J]. ACM Computer Communications Review, 1990, 20(5): 119-132.

[10] LIU F, WANG Z, CAO H P, et al. Portal Single Sign-on Scheme Based on CAS[J]. Computer Systems & Applications, 2011, 20(6): 77-80, 102.

[11] MEI H W, ZHANG M Q, LI T. Research of Website architecture with high load and high concurrency[J]. Computer & Network, 2009, 35(14): 54-57.

[12] ZENG C Y, LI J X. Redis application in cache system [J]. Micro Computer & Its Applications, 2013, 32(12): 11-13.

[13] BAO L H, HUANG Y F. Research on Architecture of High Concurrent Website and Its Solution [J]. Computer Science, 2012, 39(S2): 184-187.

[14] ANDREW S. TANENBAUM. Modern operating systems[M]. USA: Pearson. 2007: 258-271.

[15] WILLIAMS S, ABRAMS M, STANDRIDGE C R, et al. Removal policies in network caches for World-Wide Web documents [C]// Conference Proceedings on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'96). New York: ACM, 1996: 293-305.

[16] LIU L, XIONG X P. Least cache value replacement algorithm [J]. Journal of Computer Applications, 2013, 33(4): 1018-1022.

[17] JIA L, ZHANG X Y. ACACRA: a Novel Cache Replacement Algrithm[J]. Journal of Chinese Computer Systems, 2011, 32(7): 1293-1297.