Web 服务器动态性能提升的策略研究

虞安骥[1]，陈旭瑶*[2]，胥志强[2]
（1.江西开放大学，江西 南昌 33001；2. 江西应用科技学院，江西 南昌 33000

摘要：由于经济全球一体化进程加剧，互联网+的全面铺开，互联网的利用率和普及率不断提高，使得网络用户和网络需求量不断增加，使得对 Web 服务器性能的要求越来越高。文章通过分析大规模 Web 服务器（IBM 奥运会网站）访问用户响应时间，演示了在不同请求流量强度下 Web 服务器性能的问题，并针对提升服务器性能提出了解决办法，寻找 Web 服务器性能提升的途径。

关键词：Web 服务器；动态性能提升；策略研究
中图分类号：TP309 TP302.2　　**文献标识码：**B　　**文章编号：**2096-9759(2023)06-0149-03

Research on Strategies for Improving Dynamic Performance of Web Server

YU Anji[1], CHEN Xuyao[2], XU Zhiqiang[2]
(1. Jiangxi Open University, Nanchang, Jiangxi 33000, China;
2 Jiangxi Institute of Applied Science and Technology, Nanchang, Jiangxi 33000, China

Abstract: Due to the intensified process of economic global integration. The full spread of internet plus, Because of the network users and network demand continue to increase, so making the performance of Web servers increasingly demanding. This paper demonstrates the performance problems of the Web server under different request traffic intensities by analyzing the response time of the mass access Web server (IBM Olympic website), and proposes a solution to improve the performance of the server, looking for ways to improve the performance of the Web server.

Keywords: Web server; Dynamic performance improvement; Strategy research

1 研究问题

互联网技术的极速发展彻底改变了人们的生产生活方式以及学习方式。首先，以阿里巴巴为代表的电商平台给人们带来了足不出户的购物体验。阿里创建的电商平台能够同时支撑亿万用户在线访问，面对如此巨大的流量冲击，服务器很容易出现因过载而宕机的现象（房俊华，2017）；其次，百度搜索引擎用户达到了 7.66 亿人，每天响应的搜索请求超过 60 亿次；再者，网络教育公开课代表平台：慕课平台，全称为“大规模开放网络课程”，中文翻译为“慕课”人们依托慕课平台可以更加方便地学习，所以用户数量快速增长，一门慕课课程动辄上万人，甚至最多可以达到 16 万人，像是慕课如此大的规模资源网站每天都会产生海量的用户请求，这对服务器来说是个巨大的挑战。

就目前为止，影响 Web 服务器请求响应速度的主要因素有两个：（1）从客户机到服务器网络链路端到端的传输延迟；（2）Web 服务系统（包括 Web 服务器、数据库服务器、应用服务器等）的处理性能。虽然，近年来的网络技术发展突飞猛进，甚至于 10G 以太网等纷纷涌现，但就目前而言，对于宽带 IP 主干网 Web 服务器中依然存在由于 Web 服务器端的网络带宽小、用户访问量巨大以及其他的一些原因（如跨网络运营商、跨越广域网等）而导致对网站的访问速度缓慢的状况，对于

收稿日期：2023-03-24
基金项目：2020 年江西省教育厅科学技术研究重点项目；项目名称：MOOC 远程教育中 Web 服务器集群负载均衡算法研究（项目编号：GJJ209917）。
作者简介：虞安骥（1985-），男，江西南昌人，博士，副教授，研究方向：教育学。
通信作者：陈旭瑶（1994-），女，江西南昌人，硕士，全国信息化工程师，研究方向：财务大数据。

[2] 范亮，陈倩.人工智能在网络安全领域的最新发展[J].中国信息安全,2017(12):104-107.
[3] 张志勇,荆军昌,李斐,等.人工智能视角下的在线社交网络虚假信息检测、传播与控制研究综述[J].计算机学报,2021,44(11):2261-2282.
[4] 龚文全.人工智能在有害信息识别服务的应用和发展趋势[J].电信网技术,2018(2):10-14.
[5] 卢刚.面向网络社区的敏感信息语义计算方法研究[D].北京:北京邮电大学,2018.
[6] 吴珊,李跃新.智能设备网络虚假信息行为识别与控制技术研究[J].计算机测量与控制,2019,27(4):88-91,133.
[7] Devlin J, Chang M-W, Lee K, et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding [J]. North American Chapter of the Association for Computational Linguistics,2018:179-195.
[8] Vaswani A, Shazeer N, Parmar N, et al. Attention is All you Need[J].Advances in neural information processing systems, 2017,30:231-242.
[9] Tay Y,Dehghani M,Bahri D,et al.Efficient Transformers:A Survey[J].Learning,2020:1-39.
[10] Letarte G,Paradis F,Giguère P,et al.Importance of self-attention for sentiment analysis[C].In Proceedings of the 2018 EMNLP Workshop BlackboxNLP:Analyzing and Interpreting Neural Networks for NLP,2018:267-275.

Web 内容提供商来说，这是迫切需要解决的问题（武志鹏，2007）。由于 1998 年奥运会的 IBM 网站是一个特殊的大规模访问者重复大量访问的 Web 服务器环境的经典代表。所以本文将会以 1998 年奥运会的 IBM 网站的 Web 服务器作为研究对象，进行 Web 服务器响应问题讨论。

2 Web 服务器性能制约要素

2.1 研究方法

1998 年奥运会的 IBM 网站是一个特殊的大规模访问者重复大量访问的 Web 服务器环境的代表。本文基于长野冬季奥运会的 IBM 网站的 Web 服务器系统来分析用户登入响应速度及卡顿情况来分析 Web 服务器的响应问题。文章着重研究 Web 服务器响应时间与不同条件环境下的不同表现：(1)访问者所处地理位置的不同；(2)不同流量强度表现。通过两种类型的 Web 服务器反馈数据，对比响应时间数据，进行用户响应时间效能分析。

2.2 研究内容

本文基于 1998 年长野冬季奥运会的 IBM 网站的 Web 服务器系统来分析用户登入和地区的不同对 Web 服务器响应时间的影响，长野冬季奥运会的 IBM 网站系统主要由四个不同地点的多个 SP2 机器框架组成，每个 SP2 框架由 10 个单处理器节点和 1 个多处理器节点组成，特别是，在沙翁堡地点有 4 个框架，而在哥伦布、贝塞斯达和东京地点各使用 3 个框架，所有传入的请求都由一组网络调度器路由器（Dias，1996）路由到特定 SP2 机器的特定节点，每个节点都采用加权循环策略，节点的权重是其当前负荷的函数。这种可扩展的方法通过向负载较少的节点发送更多的流量来平衡 SP2 机器及其节点上的 Web 服务器负载，似乎每个 ND 都有平滑和均衡（在统计意义上）SP2 节点之间的请求到达过程的效果，这得到了对数据分析的支持，所有请求的文件都被直接发送给用户，而不需要再通过 ND（Dias，1996），选择这种系统结构是为了保证对不断变化的数据的访问具有一定的性能，确保在利用传统计算方式单线程运行时在奥运会期间提供服务保障。

1998 年 2 月 13 日奥林匹克网站访问日志中显示一天内有 5680 万次点击。在获取不同类型的文件中，所有的 gif、jpg、txt 和 class 文件以及大多数其他（数据/音频/等）文件都是静态的，大约 75%的 Dhtml/htm 文件是静态的，所有带有问题标记的页面、摄像文件和大约 25%的 html/htm 文件是动态的。总的来说，大约 32%的请求是向动态页面发出的，并且页面的平均大小为 10KB，这个数据表明，98 年长野冬季奥运会符合大规模访问者登入反应时间研究分析条件。

2.3 制约要素

2.3.1 访问者所处地理位置

根据测试不同地区，不同时间段对 Web 服务器访问流量也有些许不同。在 1998 年 2 月 13 日，整个网站以及来自三个地理区域的每 5 分钟收到的请求数，总体流量的三个峰值主要是来自三个不同地理区域的突发请求。第一个峰值主要是来自亚洲的突发请求，欧洲和美洲的流量较少；第二个，也是最大的一个峰值主要是来自亚洲和欧洲的突发流量，而美洲的请求较少；第三个，也是最宽的一个峰值，主要是来自欧洲和美洲的突发请求，而亚洲的流量要少得多。

不同地理位置，峰值的突发请求时间也不同。综合来看，美洲的峰值出现与其他区域出现的时间不同，但基本上所有区域都在 7 点-8 点出现了峰值，最高峰值为 319472。访问峰值或请求超过服务器的承受力会导致服务器瘫痪，网站访问超时，即为超载。

2.2.2 访问时间

本文对来自 1998 年奥运会网站的访问日志的分析为请求模式中的自相似性提供了一些证据，为了更详细地研究这种行为的一些原因，从不同的方式中提取了趋势，在去除趋势后，得到了原始时间序列和同一时间序列对应的自相关函数（即残差）。 整个网站收到的请求数量有相当大的季节性变化，对应于每天的周期，其中每一天都反映了上面显示的三个高峰的行为，在一周的不同日子里表现出不同模式的趋势，从周一（2/9）到周五（2/13），访问强度的增加趋势，而访问强度在周末下降，只是在下周一（2/16）重复，这种行为在其他星期也有类似的表现，显示出与每周周期相对应的季节性变化。

在峰值阶段大量访问者进入导致 Web 服务器性能降低，特别是在周五（2/13-2/14）这天中达到了最高值，大部分 Web 请求都是阻塞性质的，当一个请求被处理时，进程就会被挂起（占用 CPU）直至请求完成（Meng M，2018）。所以，无论每个访问请求自身返回多快，服务器对远端的访问请求往返都会产生滞后。

3 Web 服务器性能提升路径分析

Web 服务器集群系统架构实现服务器负载均衡，从而试验随机策略以及最少连接策略两种提升路径进行分析。

3.1 随机策略

随机策略指的是在可用服务中随机找一个，如果随机找到的服务器为 null 或者不可用，就会 while 不停的循环选取，直到选取可用的服务器。

用 python 简单表示就是：

```
public class RandomRuleTest {
    private Random random;
    /**
    *负载均衡随机策略
    *1. 获取注册到注册中心的服务的总列表，服务状态为 UP 的服务列表
    *2.判断服务总列表，服务状态为 UP 的服务列表，若为空或者数量为 0，则返回 null
    *3.根据总服务的数量，获取随机数
    *4.根据随机数获取服务器
    *5.判断服务器是否为 null，若为 null，则重新 1，2，3，4 步骤
    *6.若服务器不为空，判断服务器是否为存活状态，若为存活状态，为返回服务器信息；若不为存活状态，重新 1，2，3，4，5 步骤
    * @param loadBalancer
    * @return
    */
    public Server chooseServer(ILoadBalancer loadBalancer) {
        if(null == loadBalancer) {
            return null;
        }
        Server server = null;
        while (server == null) {
            // 当前线程被打断时，返回 null
            if(Thread.interrupted()) {
```

```
            return null;
        }
        // 获取此时此刻可以访问的服务列表和全部服务器列表
        List<Server> reachableServers = loadBalancer.getReachableServers();
        List<Server> allServers = loadBalancer.getAllServers();
        // 此时此刻总服务器列表为 null，则返回 null
        if(allServers == null || allServers.isEmpty() || reachableServers == null || reachableServers.isEmpty()) {
            return null;
        }
        // 服务器总数量
        int serverCount = allServers.size();
        // 随机获取一个索引
        int index = random.nextInt(serverCount);
        // 根据随机索引，获取服务器
        // 这里源码有一个 bug，应该通过 allServers 来获取服务器
        server = allServers.get(index);
        // 判断 server 是否为 null，若为 null，则需要再次获取，直到获取能够使用的服务器
        if(null == server) {
            // 当 server 为 null 时，将当前线程设置成挂起状态，然后和其他线程重新争抢 CPU 资源
            Thread.yield();
            continue;
        }
        if(server.isAlive()) {
            return server;
        }
        server = null;
        Thread.yield();
    }
    return server;
  }
}
```

3.2 最少连接策略

最少连接策略指的是从已有的后端列表中选择正在处理的连接数或请求数最少的节点出来提供服务。

用 python 简单表示就是：

```
class Node:
def __init__(self, name):
    self.name = name
    self.connections = 0
    def __repr__(self):
        return '<Node:  {}, conn: {}>'.format (self.name, self.connections)
class LC:
    def __init__(self, nodes):
        self.nodes = nodes
    def select(self ):
        best =None
        for node in self.nodes:
            if best is None or node.connections < best.connections: best = node
            best.connections += 1
            return best
    def release(self, node):
    node.connections－=1
```

通过对随机策略和最少连接策略进行 Web 服务器实战模拟，得出图 1。图 1 为模拟用 3000 名用户登录西安电子科技大学网络教育学院学习平台，以平均连接数和成功率作为基础比较来进行测试，比较测试期间系统的访问成功率。

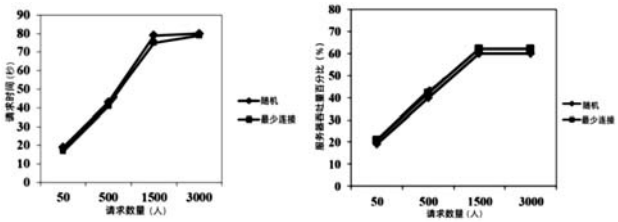

图 1 集群服务器系统吞吐量及响应时间对比

图 1 的大致趋势并无明显区别，但是通过对比，最少连接策略和随机策略在介质负载时，还是有细微的不同表现。根据吞吐量测试数据输出结果可以明显得出不管介质是否负载，随机策略的服务器吞入量占比一直都大致平衡于最少连接策略。然而根据请求时间的表现来看，最少连接策略在越接近介质负载的情况下，相较于随机策略的请求时间就越趋近，在访问人数在 750-3000 人之间时，最少连接策略的反应时间都是优于随机策略的，随着服务器负载提高，随机策略与最少连接策略都趋于疲软。总而言之，在高负载的情况下，随机策略与最少连接策略对提升服务器反应速率的效果都作用不大。

4 结语

从用户角度讲，用户调用 Web 服务时，请求等待反应时间越短，用户体验越好；从服务端角度讲，同一时间能负载用户请求量越大，服务端性能就越强。综合两方面，总结性能优化的三个方向：(1)增加服务端所能负载请求的最大数量；(2)提高每个请求处理速度；(3)设计最优服务器选择策略。效果最好的还是基于服务器状态和请求信息的新策略，以减少响应时间，提高负载均衡效果，该算法的机制是基于客户端的请求内容对服务器的影响程度进行分类，并设置相应的权重，同时结合服务器性能参数和负载情况，将请求分配给最轻负载的服务器。

参考文献：

[1] 房俊华,分布式数据流系统中负载均衡技术研究[D].华东师范大学博士论文,2017.

[2] 武志鹏,Web 服务的性能优化研究[D].厦门大学,2007.

[3] Dias D M,Kish W,Mukherjee R,et al.A scalable and highly available Web server [C]// Compcon '96.'Technologies for the Information Superhighway' Digest of Papers.IEEE, 1996.

[4] Meng W, Qian C, Hao S,Rampart: Protecting web applications from CPU-exhaustion denial-of-service attacks[C]//27th [USENIX] Security Symposium ([USENIX] Security 18), 2018: 393-410.