计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2019,28(5):167−172 [doi: 10.15888/j.cnki.csa.006907]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

实时通信集群负载均衡策略研究与应用①

于　波[1,2], 李庆建[1,2], 王　卫[2]

[1](中国科学院大学, 北京 100049)
[2](中国科学院 沈阳计算技术研究所, 沈阳 110168)
通讯作者: 李庆建, E-mail: hydrogenlee@qq.com

摘　要: 实时通信主要传输实时音视频, 具有低延时和高带宽消耗的特点. 在用户量较大的场景下, 单服务器方案无法满足整体需求, 此时需搭建分布式集群对外提供服务, 而如何将这些访问合理的分配到不同服务器上, 均衡集群内服务器的负载就显得尤为重要. 本文首先分析单服务器场景下的实时通信流程, 然后研究和分析常见的负载均衡算法, 同时为满足同群组客户端需转发到相同服务器的一致性要求, 提出一种基于一致性哈希算法和遗传算法的自适应负载均衡算法, 并对该算法进行应用和实验验证.

关键词: WebRTC; 实时通信; 负载均衡; 一致性哈希; 遗传算法

引用格式: 于波,李庆建,王卫.实时通信集群负载均衡策略研究与应用.计算机系统应用,2019,28(5):167–172. http://www.c-s-a.org.cn/1003-3254/6907.html

Research and Application of Load Balancing Strategy on Real-Time Communication Cluster

YU Bo[1,2], LI Qing-Jian[1,2], WANG Wei[2]

[1](University of Chinese Academy of Sciences, Beijing 100049, China)
[2](Shenyang Institute of Computing Technology, Chinese Academy of Sciences, Shenyang 110168, China)

Abstract: Real-time communication mainly transmits real-time audio and video with low latency and high bandwidth consumption. In the scenario of large number of users, the single server solution cannot meet the overall needs, so it is necessary to build a distributed cluster to provide services, and how to properly distribute these requests to different servers, balance the load of servers in the cluster is particularly important. This paper first analyzes the real-time communication flow of single server. Then the common load balancing algorithms are analyzed. In order to meet the consistency requirements of forwarding from the same group of clients to the same server, an adaptive load balancing algorithm based on consistency Hash algorithm and genetic algorithm is proposed and verified.

Key words: WebRTC; real-time communication; load balancing; consistency Hash; genetic algorithm

实时通信与传统即时通信不同, 即时通信主要传输图片、文字和非实时音视频, 而实时通信主要传输实时音视频, 允许一定丢包率存在, 但对延时要求更高. 实时音视频可以使用 WebRTC 技术进行传输, WebRTC 技术与传统的 RTMP 直播技术不同, 可以在不使用任何插件的情况下进行实时通信[1].

WebRTC 技术本质是一种 P2P 技术, 不需要服务器进行转发混流, 但在用户较多的场景下, 会建立大量连接并过度消耗客户端带宽, 此时需要实时通信服务器的协助. 实时通信服务器按照功能可以分为信令服务器和媒体服务器, 信令服务器用于信令交互, 媒体服务器用于传输媒体流, 以及转码混流等操作, 其中本文研究的重点是媒体服务器集群的负载均衡.

本文首先分析单服务器场景下的实时通信流程,

① 收稿时间: 2018-11-29; 修改时间: 2018-12-26; 采用时间: 2018-12-28; csa 在线出版时间: 2019-05-01

然后研究和分析常见的负载均衡算法, 为满足同群组客户端需转发到相同服务器的一致性要求, 提出一种基于一致性哈希算法和遗传算法的自适应负载均衡算法, 并对该算法进行应用和实验验证.

1 相关技术简介

1.1 WebRTC 简介

WebRTC (Web Real-Time Communication, 网页实时通信)[1,2]是一种不使用任何插件就可以在客户端之间进行实时通信的开源技术. 目前大多数主流浏览器 (例如 Chrome、Firefox、Safari 等) 都已支持 WebRTC 功能, 开发人员只需要一些简单的 JavaScript 操作就可以快速搭建实时通信应用程序.

使用 WebRTC 技术建立实时通信主要包括两个步骤: 信令交互和媒体传输. WebRTC 并没有指定具体的信令交互协议, 开发者可以根据需要选用合适的协议 (例如 JSON, SIP 等); 为了保证实时性, WebRTC 通过 RTP 和 RTCP 协议传输和控制实时媒体流.

1.2 Kurento 媒体服务器简介

Kurento 媒体服务器 (Kurento Media Server, KMS)[3]是一个开源的 WebRTC 媒体服务器, 提供了一系列 API 来简化实时通信应用程序开发流程. KMS 除了提供基本的转发和转码功能, 还提供了诸如计算机视觉和增强现实等高级功能, 这些功能都是基于模块化开发方式, 开发者可以自由选用合适功能. 根据 Goni Garcia 等人的研究[4]表明, 单台 Kurento 媒体服务器满足实时通信要求的最大并发用户量是 175, 当用户量超过 175 时, 就会出现明显的卡顿现象, 因此单台 Kurento 媒体服务器无法支持庞大的用户量.

2 单服务器场景下的实时通信流程

本部分首先分析基于 WebRTC 的点对点实时通信流程, 然后研究添加服务器的实时通信流程, 最后分析实时通信集群负载均衡策略需达到的目标.

使用 WebRTC 技术建立实时通信主要分为两个过程: 信令交互和媒体传输, 流程如图 1 所示. 客户端之间直接交换 SDP 信息 (Session Description Protocol, 主要包括客户端 IP 地址信息和音视频编码、码率等媒体相关信息). 在建立连接之后, 客户端将媒体流发送给其他客户端, 然后接收其他客户端发送的媒体流, 也就是说客户端分别发送和接收 N–1 路媒体流.

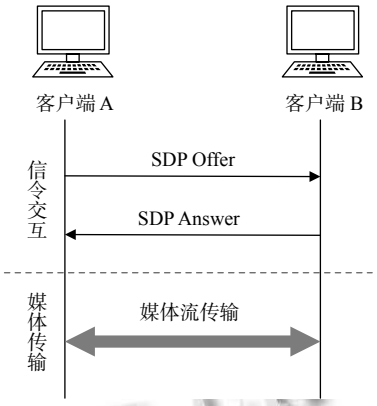

图 1 WebRTC 建立实时通信流程 (点对点)

当用户量过大时, 为减低带宽消耗和便于管理 (采用群组的概念管理一组相互通信的客户端), 需要服务器协助, 流程如图 2 所示. 为保证服务器功能的单一性, 将服务器分为信令服务器和媒体服务器. 信令服务器主要用于信令交互, 此时客户端之间不再直接发送 SDP 信息, 而是先将 SDP 发送给信令服务器, 然后再由信令服务器发送给其他客户端. 媒体服务器用于混合和转发媒体流, 客户端将媒体流发送给媒体服务器, 经过处理后, 发送给其他客户端.

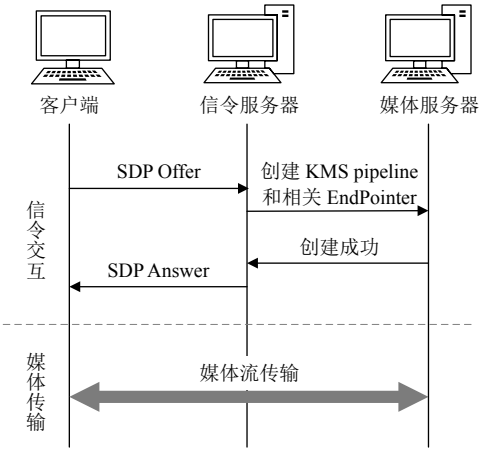

图 2 WebRTC 建立实时通信流程 (有服务器)

为提高系统性能和并发量, 无论信令服务器还是媒体服务器都需组建集群. 组建集群后, 如何保证集群内部服务器的负载均衡就显得尤为重要. 本文主要研究媒体服务器集群的负载均衡, 对于信令服务器集群, 本文末尾将会简单介绍. 根据需求, 媒体服务器集群的负载均衡策略, 需要达到以下目标:

① 保证集群整体负载均衡.

② 群组内客户端的媒体传输需由同一台媒体服务

器处理, 即请求转发到同一台媒体服务器.

③ 能够根据集群实时状态进行调整 (自适应).

3 实时通信集群负载均衡策略研究与应用

在用户量较大的场景下, 单服务器方案无法满足整体需求, 此时需搭建分布式集群对外提供服务, 而如何将这些访问合理分配到不同服务器上, 均衡集群内各服务器负载就显得尤为重要. 下面将先比较常见的负载均衡算法, 然后提出一种基于一致性哈希和遗传算法的自适应负载均衡算法.

3.1 常见负载均衡策略比较

负载均衡策略根据能否及时响应节点状态, 分为静态负载均衡和动态负载均衡[5]. 静态负载均衡主要包括轮询算法、加权轮询算法和目标地址散列算法. 其中, 轮询算法只是将请求按照顺序转发给后端服务器, 没有考虑服务器性能差异; 加权轮询算法考虑到了后端服务器的个体差异, 但是是手动设置权值, 不能根据服务器的实时负载情况进行自动调整; 目标地址散列算法是将同一个 IP 地址的请求转发到同一台服务器上, 但可能会出现集群负载倾斜的现象, 一致性哈希算法通过使用虚拟节点技术, 可以在一定程度上解决负载偏移的问题.

动态负载均衡算法主要包括最小连接数算法和加权最小连接数算法. 其中, 最小连接数算法是将集群中各服务器当前任务连接数作为负载均衡指标, 每次都选择任务连接数最少的服务器, 但没有考虑服务器个体差异; 而加权最小连接数算法则根据集群中服务器的性能差异设置不同权值, 然后根据服务器当前任务连接数和权值的计算结果作为负载均衡指标, 但最小连接数算法只考虑了服务器的任务连接数, 没有考虑 CPU, 内存和带宽等其他因素.

国内外对于动态负载均衡算法研究较多, 在考虑任务连接数的基础上, 综合 CPU 消耗、内存利用率和带宽利用率等作为负载均衡的指标[6–11]. 由于使用场景的不同, 大多数的算法[6–9]没有考虑将同一请求转发到同一服务器节点上. 虽然算法[10,11]采用一致性哈希算法将相同请求转发到同一服务器节点上, 但是各个负载性能指标 (CPU、内存和带宽等) 的权值采用固定值, 没有根据集群具体情况进行调整.

3.2 基于一致性哈希和遗传算法的负载均衡算法研究

根据上述对常见负载均衡算法的分析和比较, 为满足同群组客户端需转发到相同服务器的一致性要求, 选用一致性哈希算法作为基本负载均衡策略, 并对其进行优化.

一致性哈希算法是将服务器节点通过哈希函数映射到一个值域为[0, 2^{32}–1]的环上, 同时将请求也映射到环上, 顺时针寻找最近的服务器节点. 但因服务器节点较少, 请求可能出现偏移, 产生雪崩效应[12], 通过使用虚拟节点, 可以在一定程度上解决雪崩效应. 图 3 是带有虚拟节点的一致性哈希示意图, 标号 S 表示物理服务器节点, 标记 V 表示虚拟服务器节点.

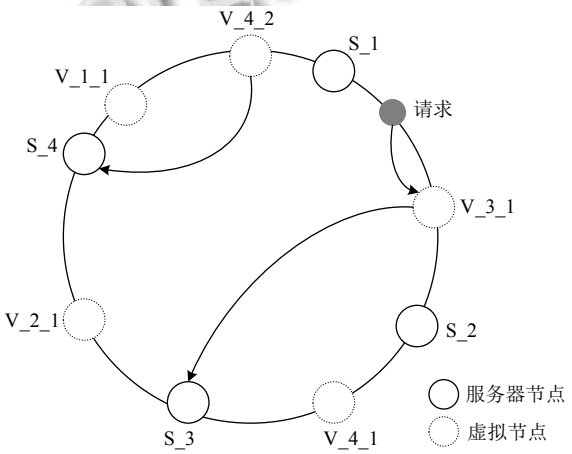

图 3 带有虚拟节点的一致性哈希示意图

在一致性哈希算法中, 集群内各服务器的性能差异, 可转变为每台物理服务器节点分配的虚拟节点数量. 在集群初始状态, 虚拟节点数量往往是根据经验人工进行配置. 为保证负载均衡算法的自适应性, 需要根据集群节点的负载状态进行调整, 将某些负载较高的服务器的权重降低, 把某些负载较低的服务器的权重提高, 而这些操作可通过调节物理服务器节点所对应的虚拟服务器节点数量来实现.

服务器的负载率可通过式 (1) 计算:

$$LB_i = \alpha C_i + \beta M_i + \gamma B_i + \delta L_i \tag{1}$$

式中, LB_i 代表第 i 台服务器的负载率, C_i、M_i、B_i、L_i 是负载状态指标, 分别代表 CPU 利用率、内存利用率、带宽利用率和连接数比率, 其中:

$$\alpha + \beta + \gamma + \delta = 1 \tag{2}$$

CPU 利用率 C_i 根据式 (3) 计算, M_i 和 B_i 同理.

$$C_i = \frac{C_{ti}}{C_{\text{Total}}} \tag{3}$$

不过, 连接数比率 L_i 的计算稍有不同, 当一个用户邀请群组内其他用户进行实时通信时, 其他用户不一定立即响应, 但这一部分连接数也需考虑在内, 因此乘以一个常数 k 来表示未响应的连接数中将会接受邀请的连接数, 如式 (4) 所示.

$$L_i = \frac{(L_{Ni} - L_{ti}) \times k + L_{ti}}{L_{\text{Total}}} \tag{4}$$

服务器定时向负载均衡器发送负载信息, 负载均衡器收到数据后计算各服务器节点的负载值 LB_i, 当出现媒体服务器节点的 $LB_i > LB_{\max}$, 其中 $LB_{\max}$ 代表集群中媒体服务器正常工作的最大负载率, 是一个经验值, 将根据式 (5)、式 (6) 计算出每个物理服务器对应的虚拟节点数, 其中 V_{Total} 代表集群所分配的虚拟节点总数.

$$p_i = \frac{1 - LB_i}{\sum (1 - LB_i)} \tag{5}$$

$$V_i = \lfloor p_i \times V_{\text{Total}} \rfloor \tag{6}$$

计算出每个物理服务器的虚拟节点数后, 将进行重新映射, 为保证群组中的所有用户仍能转发到同一台媒体服务器上, 需记录群组所对应的媒体服务器 (在信令服务器集群的 Redis 数据库中存储, 见图 4), 即当群组已存在时, 直接转发给指定的媒体服务器, 当不存在时, 按照一致性哈希算法分配媒体服务器.

但在上述算法中, $\alpha, \beta, \gamma, \delta$ 这几个参数是根据经验选取的固定值, 会对结果产生较大影响, 为解决这个问题, 可通过遗传算法对权值进行调整, 最终得到最优权值. 使用遗传算法解决问题的关键是选择合适的适应度函数以及选择、交叉和变异等遗传操作. 其中染色体的结构为 (α, β, γ, δ), 粗略的经验值为 (0.2, 0.25, 0.35, 0.2), 初始种群为 40, 每个染色体各个位置的值上下浮动不超 0.05, 通过随机算法生成.

适应度函数根据式 (7)~式 (9) 进行计算, f 的值越大, 代表适应度越高.

$$\overline{LB} = \frac{\sum LB_i}{N} \tag{7}$$

$$S^2 = \frac{\sum (LB_i - \overline{LB})^2}{N} \tag{8}$$

$$f = \frac{1}{S} \tag{9}$$

选择操作采用轮盘赌算法, 适应度高的个体更容易被选中进入下一代. 交叉操作, 通过随机方法选择染色体上的两个位置, 然后将其交换, 在本文算法中交叉操作执行的概率为 80%. 变异操作, 通过随机方法选择染色体的一个位置, 然后将其上下浮动不超过 0.05, 在本文算法中变异操作执行的概率为 5%. 为了保证集群的稳定性, 在集群启动的一定时间内, 不使用遗传算法改变权值.

3.3 实时通信集群负载均衡策略应用

根据上述对负载均衡策略的研究, 整个实时通信集群的架构如图 4 所示, 主要分为信令服务器集群和媒体服务器集群. 媒体服务器集群, 主要用于转发和转码视频流, 通过基于一致性哈希和遗传算法的负载均衡算法, 在集群整体均衡的基础上, 将同一个群组中的客户端的视频流转发到同一台媒体服务器上进行处理. 由于 Kurento 媒体服务器有良好的扩展性[3], 因此影响集群最大用户量的关键因素是集群中媒体服务器的数量, 即可以通过增加媒体服务器的数量来提高集群服务的最大用户量.

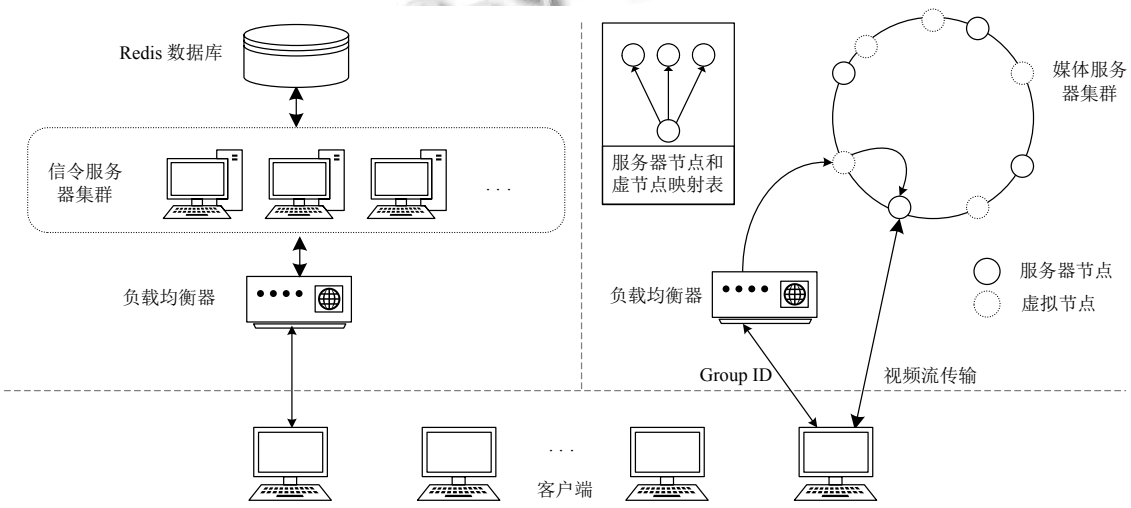

图 4 实时通信集群整体架构图

对于信令服务器集群, 通过将群组的信息存放在 Redis 数据库中, 可以防止多台信令服务器信息不一致的情况, 因此客户端的请求可以转发到任意一台信令服务器上, 不需要使用一致性哈希算法进行负载均衡. 本文虽然没有对信令服务器集群的负载均衡进行详细介绍, 其实可以按照式 (1)、式 (2)、式 (3)、式 (7)、式 (8)(L_i 也需按照式 (3) 进行计算) 计算出每台服务器的负载, 然后每次都选择负载率最低的服务器处理客户端请求.

4 性能测试

本文采用修改后的 NUBOMEDIA benchmark[4]工具对文中提出的负载均衡算法进行测试, 并将测试结果与传统的一致性哈希算法进行比较, 其中视频流采用 H.264 编码, 帧率为 30 fps, 分辨率为 600×800; 音频流采用 Opus 编码, 采样率为 48 kHz. 为便于测试, 客户端连接媒体服务器后不断开连接, 实验在局域网环境下进行. 测试所用设备如表 1 所示.

表 1 测试设备表

设备	CPU(核)	内存 (GB)	带宽	硬盘 (GB)
服务器 1、2	8	8	1 G	500
服务器 3、4	4	8	1 G	500
服务器 5	4	4	512 M	500

实验结果如下, 当集群连接数较少时, 本文算法和传统的一致性哈希算法在平均延时方面相差不大, 但随着连接数增多, 本文算法平均延时更短, 如图 5 所示. 平均延时在一定程度上能够反映负载均衡的效率, 但单纯使用平均延时来评价算法的负载均衡效率并不精确, 本文通过计算集群内各台媒体服务器负载率的方差, 来更准确的度量算法的负载均衡效率, 如图 6 所示 (数据经过预处理, 各媒体服务器的负载率先乘以 100 后再计算方差). 通过图 6 可以看出本文算法的负载率的方差相对较低, 即本文算法相对于传统的一致性哈希算法更能均衡集群内各服务器的负载.

因客户端连接服务器后不断开连接, 各服务器的负载率将不断上升, 但集群中各服务器负载整体比较均衡 (不超过 15%), 如图 7 所示; 另外通过图 8 可看出, 虽然服务器负载整体比较均衡, 但由于服务器性能差异导致服务器分配的连接数不同, 即本文算法能够根据服务器之间的性能差异, 合理的均衡负载.

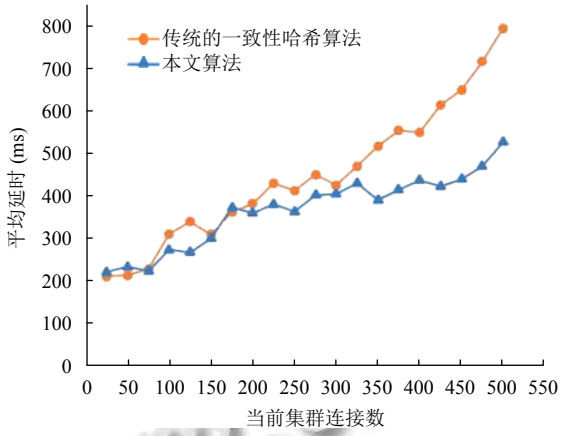

图 5 本文算法和传统一致性哈希算法延时比较图

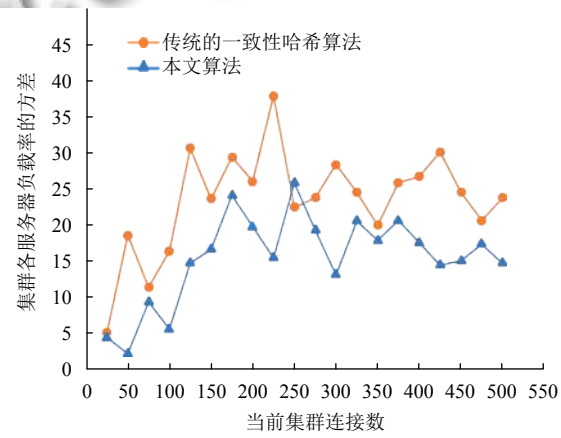

图 6 本文算法和一致性哈希算法负载率方差比较图

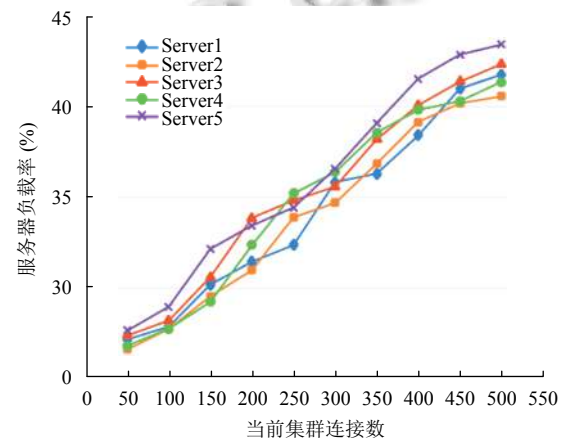

图 7 服务器负载率与集群连接数关系图

5 结语

本文通过分析和比较常见的负载均衡算法, 并根据 WebRTC 实时通信的特点, 对传统的一致性哈希算法进行优化, 提出了一种基于一致性哈希算法和遗传

算法的自适应负载均衡算法, 算法能够保证系统整体的负载均衡并且能够将一个群组内的客户端请求转发到同一台媒体服务器上进行处理. 但是本文对负载状态指标的选取和集群发生问题后, 如何快速将系统的负载调节到均衡状态等方面的研究仍有不足, 需进一步研究并优化算法.

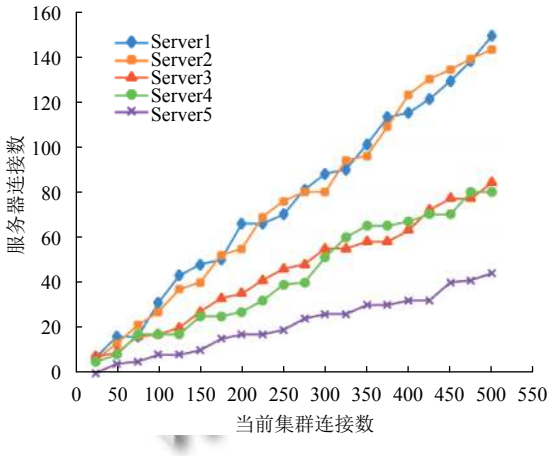

图 8 服务器连接数与集群连接数关系图

参考文献

1 Loreto S, Romano SP. Real-time communications in the web: Issues, achievements, and ongoing standardization efforts. IEEE Internet Computing, 2012, 16(5): 68–73. [doi: 10.1109/MIC.2012.115]

2 张向辉, 黄佳庆, 吴康恒, 等. 基于 WebRTC 的实时视音频通信研究综述. 计算机科学, 2015, 42(2): 1–6, 32.

3 Garcia B, Lopez-Fernandez L, Gallego M, *et al*. Kurento: The swiss army knife of webRTC media servers. IEEE Communications Standards Magazine, 2017, 1(2): 44–51. [doi: 10.1109/MCOMSTD.2017.1700006]

4 Garcia B, Gortazar F, Lopez-Fernandez L, *et al*. WebRTC testing: Challenges and practical solutions. IEEE Communications Standards Magazine, 2017, 1(2): 36–42. [doi: 10.1109/MCOMSTD.2017.1700005]

5 Deepa T, Cheelu D. A comparative study of static and dynamic load balancing algorithms in cloud computing. Proceedings of 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing. Chennai, India. 2017. 3375–3378. [doi: 10.1109/ICECDS.2017.8390086]

6 王钊. 流媒体服务器集群负载均衡策略的研究[硕士学位论文]. 西安: 西安邮电大学, 2017.

7 Rajput SS, Kushwah VS. A genetic based improved load balanced min-min task scheduling algorithm for load balancing in cloud computing. Proceedings of the 2016 8th International Conference on Computational Intelligence and Communication Networks. Tehri, India. 2016. 677–681. [doi: 10.1109/CICN.2016.139]

8 魏雪. 基于遗传算法的 Web 服务器集群负载均衡的研究[硕士学位论文]. 杭州: 浙江理工大学, 2017.

9 高田. 基于服务器集群的负载均衡策略的研究[硕士学位论文]. 天津: 天津理工大学, 2016.

10 Li J, Nie YF, Zhou SJ. A dynamic load balancing algorithm based on consistent hash. Proceedings of the 2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference. Xi'an, China.2018.2387–2391.[doi:10.1109/IMCEC.2018.8469341]

11 胡丽聪, 徐雅静, 徐惠民. 基于动态反馈的一致性哈希负载均衡算法. 微电子学与计算机, 2012, 29(1): 177–180. [doi: 10.19304/j.cnki.issn1000-7180.2012.01.043]

12 Lin P, Nie HM, Ding G. Load balancing framework based on consistency Hashing algorithm. Proceedings of 2014 International Conference on Mechatronics and Control. Jinzhou, China. 2014. 1504–1507. [doi: 10.1109/ICMC.2014.7231808]