分　数：	
评卷人：	

華中科技大學

研究生（ 数据中心 ）课程论文（报告）

题 目：PostMan: Rapidly Mitigating Bursty Traffic by Offloading Packet Processing

学　　号 M201973384

姓　　名 杨震

专　　业 计算机技术

课程指导教师 施展 曾令仿

院（系、所） 计算机科学与技术

2019 年 **10** 月 **28** 日

综述报告

1 背景

1.1 关于论文

本篇论文是由本校计算机学院“服务计算技术与系统”教育部重点实验室的金盼盼等人发表于 2019 年 USENIX ATC。该会议成立于 1975 年，并逐渐发展成为由计算机操作系统用户，开发者和研究者所组成的非盈利机构。其目标是促进技术创新、支持和传播带有实践偏见的研究、为技术讨论提供一个中立论坛以及鼓励计算机的发展。

该课题组曾在 2018 年提出了 eBA[1]：一种高效的带宽分配解决方案，可以在数据中心级别的大量短流量和突发流量的情况下，为虚拟机提供端到端的带宽保证。进而对小数据包的短流量和突发流量做进一步的研究，提出本篇论文。

项目课题主要来源于国家自然科学基金项目:“边缘计算基础理论与关键技术”、“网络资源共享与管理”和“高效能数据中心—数据中心资源的优化组织与调度”以及国家重点研发计划课题:“高效能云计算数据中心资源调度关键技术研究”。

1.2 研究机构与成果

现在有许多工作来研究如何通过更好的数据分区和布局策略来缓解负载不平衡问题，这些研究机构大多来自于公司。例如 Apache 公司的分布式数据库 HBase[2]，微软公司的 Windows Azure Storage[3]，谷歌公司的结构化数据管理的分布式存储系统 Bigtable[4]和高性能全球分布式同步备份数据库 Spanner[5]，亚马逊公司的高可靠数据存储技术 Dynamo[6]，Igor Sysoev 所开发高性能 Web 和反向代理服务器 Nginx[7]。

1.3 工作开展

这些负载均衡方案对于长期稳定的负载不平衡问题都很有效，但是对于突发流量导致的负载不平衡，此类系统需要进行限流或者在线热数据迁移来缓解负载不平衡。但是这两种方法都不能快速缓解服务器的负载。

论文提出了 PostMan，通过转移处理小数据包的开销来快速减轻服务器的负载。其关键思想是观察到处理小数据包的开销远大于处理大数据包的开销，所以将小包处理从服务器转移到 helper 节点，从而快速减轻服务器的负载。该方法具有快速、可扩展性

强以及高效处理小数据包等优点，但是该方法局限性明显，可用作数据迁移技术的补充。

2 发展概述

2.1 领域发展现状

本论文涉及到的领域为分布式系统的负载均衡。负载均衡是一种技术解决方案，用于在多个资源（例如服务器）之间分配负载，以达到资源的最优化使用。负载均衡一般用于解决分布式系统的大流量、高并发和高可用的问题，它的核心在于负载是否均匀分配。

现代分布式系统一般分为五层[8]：客户端层，如用户浏览器、APP 端；反向代理层，如 Nginx；Web 层，如 NodeJS、Vue 等框架；业务服务层，如 Spring Boot、Spring Cloud 等框架；数据存储层，如 MySQL、Redis 等。一个请求从客户端层到业务服务层，层层访问都需要负载均衡。即每个上游调用下游多个业务方的时候，需要均匀调用。各层间的负载均衡技术都已发展的比较成熟，如下所示：

1. 客户端层->反向代理层的负载均衡主要是利用 DNS 轮询，返回对应的 IP 地址，每个 IP 对应的反向代理层的服务实例。这样可以做到每一个反向代理层实例得到的请求分配是均衡的。
2. 反向代理层->Web 层的负载均衡主要是通过反向代理层的负载均衡模块实现。例如，Nginx 有多种负载均衡算法：*请求轮询*，请求按时间顺序，逐一分配到 web 层服务；*IP 哈希*，按照 IP 的哈希值，确定路由到对应的 Web 层。只要是用户的 IP 是均匀的，那么请求到 Web 层也是均匀的。
3. Web 层->业务服务层的负载均衡是通过“服务连接池”实现的。例如，Dubbo 服务治理方案，它的一个特性就是智能负载均衡：内置多种负载均衡策略，智能感知下游节点健康状况，显著减少调用延迟，提高系统吞吐量
4. 业务服务层->数据存储层的负载均衡主要通过数据分片进行负载均衡，数据分片的方式主要有按照数据范围进行切分，哈希水平切分。

2.2 主要挑战

目前负载均衡的发展比较成熟，但是还有一些方面存在挑战，我主要归纳了三点挑战：

1）基于硬件的负载均衡设备效率高，稳定性强，处理能力强；但是成本太高而且配置冗余。基于软件的负载均衡方式成本低，能够更好地根据系统与应用的状况来分配负载；但是负载能力受服务器本身性能的影响，性能越好，负载能力越大。如何将负载均衡的软硬方式相结合是一个挑战。

2）如何做到自适应选择负载均衡器，使系统不论处于何种状态，都能进行合理的流量分配也是一个很大的挑战。

3）目前的负载均衡的方案对于突发流量的处理效果不是很好，一般采用限流或热数据迁移方式，这两种方式会增加客户端的延迟，甚至会在一段时间内增加服务器的负载。

2.3 论文处理的问题

论文中主要讨论了遇到突发流量时的负载均衡。传统的负载均衡方案会启用限流机制，即通过排队或丢弃的方式，限制进入系统的请求数，防止系统崩溃。这种方法会降低服务器的吞吐量，增加延迟。另外还会通过热数据迁移方式缓解服务器负载，但是热数据迁移会加重服务器的负载，并且在疯狂网购季，迁移数据可能会消耗一天的时间[6]。

基于此，论文提出了 PostMan，该方法属于 Web 层到业务服务层的负载均衡，但是它与现有的方法不同，现有的方法主要是对服务的调用进行负载均衡。而 PostMan 针对底层处理数据包时，由于处理小包的开销远大于处理大包的开销，所以它提出用中间件 helper 节点来代替服务器组装小包，将小包开销从服务器转移到 helper 节点，从而快速缓解遭遇突发流量的服务器的负载。

3 PostMan 概览

论文中提出了 PostMan，一种用于减轻重负载服务器的数据包处理开销的分布式服务。PostMan 在网络中部署了许多 helper 节点来为重负载服务器组装小数据包，从而将每个包处理的恒定开销从服务器转移到了 helper 节点。服务器只需要处理大数据包，从而迅速降低开销，减小客户端延迟。

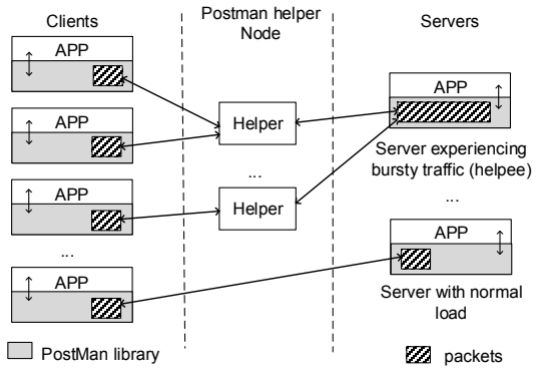

图 2.1 PostMan 概览

上图显示了 PostMan 的组织结构，其核心部分包括：

1）为服务器组装小数据包的 helper 节点，基于快速包处理的软件库 DPDK 和用户态批处理包的协议栈 mTCP 进行实现，可以高效的批处理数据包。将原包头部替换为较小的 PostMan 头部，降低有效负载。并且提供自适应批处理包的机制，有效提升高负载情况下的服务器吞吐量，降低客户端延迟。

2）为应用程序提供数据包重定向、组装和反汇编功能的 PostMan 库。此外，PostMan 库还提供了检测数据包顺序和在重新发送数据包的功能，这些功能保证了 helper 节点的容错机制与负载平衡。

4 PostMan 设计与实现

4.1 包的组装和分解

Helper 节点采用自适应批处理方法收集小数据包，即初始化一个最大批大小（S）和一个批时间间隔（T），当收集的包大小达到 S 或者等待时间达到 T 时，将收集的包传递给服务器。S 和 T 在循环的过程中自适应更新，通过收集上一循环中的包的大小（s）和等待时间（t），如果 s 和 t 位于下图中的蓝色区域，那么更新 S=s、T=t。

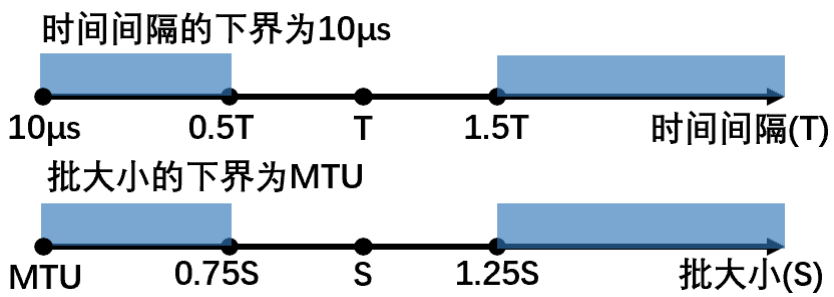

图 3.1 自适应批处理

在收集完包后，helper 节点会将收集到的包按指定格式组装成大数据包。主要是将小数据包冗余的 TCP/IP 头部替换成 PostMan 头部，该头部只有 7 个字节，包含类型、有效负载长度与源 IP 和端口字段，如下图所示。

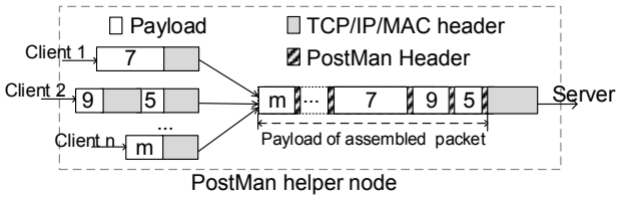

图 3.2 组装包格式

服务器收到组装后的包时，要先拆解然后处理。当服务器发送回复给 helper 节点时，会先把回复组装起来发送给节点。Helper 节点收到包后，将包进行拆解发送给不同的客户端。Helper 节点基于高速包处理软件库 DPDK 和用户态批量包处理协议栈 mTcp 进行实现，降低了每个包的处理中频繁中断所带来的开销，而且满足了 helper 节点的组装要求，确保了 PostMan 的效率和高扩展性。

4.2 PostMan 库

除了传统网络库中提供的 bind、connect、send 等 API，PostMan 库还提供了一些附加 API，客户端主要有：pm_connect：建立客户端到 helper 节点的连接；get_info：允许应用程序检索连接信息，例如已发送和接收的数据包。服务器主要有：decompose：识别请求连接的数据包以及拆解 helper 的大数据包；compose：组装多个回复成大数据包。

4.3 容错机制

当 helper 节点出错或者自动中断连接时，PostMan 提供了一套容错机制，以便客户端有足够的信息连接到其它 helper 节点。首先，PostMan 提供了一套确保包顺序的机制，客户和服务器分别维护了自己收发包的原信息，客户端标识已经发出的包。服务器标识已经收到的包。当与 helper 节点失去连接后，客户端会连接到其它的 helper 节点，然后发送重连信息。服务器收到信息后返回给客户端自己当前收到的包个数，然后客户端选择重新发送的包进行通信。具体流程如下图所示。

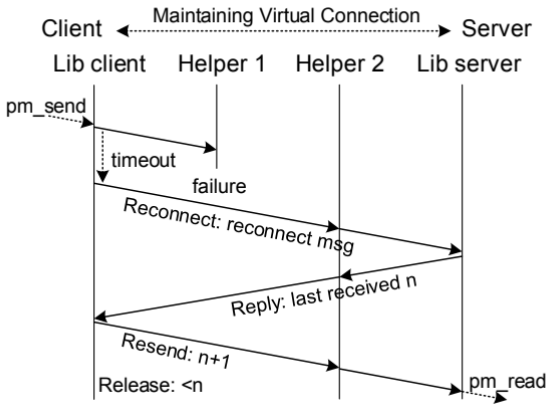

图 3.3 容错流程图

4.4 helper 节点的负载均衡

Helper 节点会监控自己的 CPU 占用率，当客户端请求连接到 helper 节点时，会随机选取一组节点，然后节点返回自己的 CPU 占用率，客户端选择占用比较小的节点进行连接。当该 helper 节点负载过高时，会断开与客户端的连接，然后客户端自动连接到其它 helper 节点。这种机制使得 PostMan 没有扩展瓶颈，可无限扩充 helper 节点。

5 性能测试

为了评估 postman 的性能，论文中主要测试了 PostMan 降低负载的效率、包大小对 PostMan 的影响、PostMan 对哪些负载有作用以及 helper 节点失败重连的时间。结果表明，PostMan 消耗的时间远小于数据迁移所消耗的时间；PostMan 可以减轻包处理开销产生的负载，但是不能减轻由锁争夺产生的负载，这也是它的缺点所在；PostMan 会提高对小包的吞吐量，但是会降低对大包的吞吐量；它的容错机制的效率很高，可以在短时间内修复大量连接；自适应批处理机制可以在高负载的情况下提高吞吐量，降低延迟时间。

6 论文改进思路

1. helper 节点的负载均衡策略有改进空间，目前的方法是客户端随机选择一组 helper 节点，询问它们的资源占用情况。但是如果选取的这一组节点资源占用都很高，而有另外一组节点的资源占用很低的话，会增加系统的延迟。
2. 启用 helper 节点的策略可以对不同的系统设置不同的方法，对一些突发流量可

预测的情况可以定时启用 helper 节点，节省系统资源。例如，电商类型的流量爆发点可以用机器学习算法进行预测，定时启用 helper 节点。

7 参考论文

[1] Liu F , Guo J , Huang X , et al. eBA: Efficient Bandwidth Guarantee Under Traffic Variability in Datacenters[J]. IEEE/ACM Transactions on Networking, 2016, 25(1):1-14.
[2] Apache HBASE. http://hbase.apache.org/.
[3] Brad Calder, Ju Wang, Aaron Ogus, Niranjan Nilakantan, Arild Skjolsvold, et al. Windows Azure Storage: a highly available cloud storage service with strong consistency. In Proceedings of SOSP, 2011.
[4] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, et al. Bigtable: A Distributed Storage System for Structured Data. In Proceedings of OSDI, 2006.
[5] James C. Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, et al. Spanner: Google's Globally-Distributed Database. In Proceedings of OSDI, 2012
[6] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, et al. Dynamo: Amazon's Highly Available Key-value Store. In Proceedings of SOSP, 2007.
[7] Nginx. http://nginx.org/.
[8] 分布式系统的负载均衡. http://news.west.cn/59317.html

研　究　生　签　字_____