LNMP 生产服务器技术改进

曾 棕 根
(宁波职业技术学院电子信息工程学院 浙江 宁波 315800)

摘 要 LNMP 服务器由于构造复杂、优化难度高,经常出现处理 PHP 网页时卡死、数据库崩溃和服务器被入侵的情况。针对上述问题,使用源代码现场编译、tmpfs 内存文件系统、网络压缩传输、多进程和线程池等多种技术手段,对 LNMP 生产服务器进行了深度技术改进。测试结果表明,LNMP 服务器处理 PHP-MariaDB 请求的速度提高了 1.8 倍,还具有安全可靠和抗拥塞特征,取得了很好的应用效果。

关键词 LNMP PHP 安全 高性能 抗拥塞 服务器

中图分类号 TP393.09 **文献标志码** A DOI:10.3969/j.issn.1000-386x.2021.06.005

TECHNICAL IMPROVEMENT OF LNMP PRODUCTION SERVER

Zeng Zonggen
(*School of Electronic Information Engineering, Ningbo Polytechnic, Ningbo 315800, Zhejiang, China*)

Abstract Due to the complexity of LNMP server structure and the difficulty of optimization, it often occurs that the server is stuck when processing PHP web pages, the database crashes and the server is invaded. This paper finds the key to the above problems. The LNMP production server was deeply improved by means of on-site source code compilation, tmpfs memory file system, network compression transmission, multi process and thread pool. The test results show that the speed of processing PHP-MariaDB requests by LNMP server is increased by 1.8 times, and it also has the characteristics of security, reliability and anti-congestion, and achieves good application effect.

Keywords LNMP PHP Security High-performance Anti-congestion Server

0 引 言

LNMP 架构免费开源、功能强大,是当前最流行的 Web 服务器架构,与大数据和人工智能服务器存在密切的关系。由于 LNMP 架构技术牵涉面广、安装与配置非常复杂且可借鉴的高端经验少,按照默认配置无法发挥该架构稳定、安全和高效的特征。LNMP 服务器出现安全漏洞、在访问高峰期网页卡死甚至数据库崩溃成为常态,因此对 LNMP 生产服务器进行技术改进成为当前亟待解决的问题。

历经七年对 LNMP 生产服务器的管理与深入研究,解决了 LNMP 架构安装、运行中遇到的各种问题,对该架构进行了全方位的技术改进,使得该架构安全、快速、抗拥塞。本文系统地论述了该架构在上述三方面所做的深度技术改进。

1 LNMP 架构的组成

LNMP 架构是指由 Linux 内核的操作系统、Nginx 服务器、MariaDB 数据库服务器和 PHP 脚本服务器组成的 PHP 动态网站运行架构,组成该架构的四个软件都是免费开源的[1]。生产环境下,推荐使用的 Linux 操作系统是 CentOS-7 (1908),其稳定、快速、安全;Nginx 是一款高性能低开销的 Web 服务器,目前最新的稳定版本是 1.16.2[2];MariaDB 是 MySQL 数据库的一个分支,由开源社区维护,代码更新速度最为快速,目前最新稳定版本为 10.3.11; PHP 网页服务器则是用来编译和运行 PHP 脚本,目前最新版本为 7.2.24。图 1 为 LNMP 架构数据流动示意图。

收稿日期:2019-09-24。**曾棕根**,副教授,主研领域:软件工程。

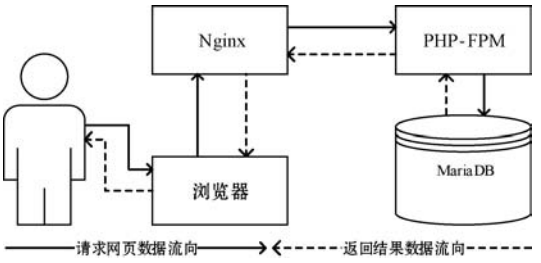

图 1 LNMP 架构数据流动示意图

用户总是在浏览器中发出 PHP 网页查看请求，Nginx 收到用户的 PHP 程序调用请求后，将此请求发送给 PHP-FPM 服务器，PHP-FPM 服务器则调用 PHP 进程去编译和运行 PHP 程序，PHP 程序则通过 SQL 指令从 MariaDB 数据库中去存取所需信息，MariaDB 将 SQL 指令运行后以二维表的形式将结果返回给 PHP 进程，PHP 进程则将记录以 HTML 语法进行包装，PHP-FPM 服务器会将此 HTML 文本传送给 Nginx 服务器，Nginx 服务器则将此 HTML 文本返回给客户端浏览器，客户端浏览器则解释此 HTML 文本中的 HTML 标签，用户即可从浏览器中看到此 PHP 程序的运行结果。

由图 1 可见，数据的流动是以单一、固定的路径传输的，数据总是从客户端浏览器中发出，经过 Nginx、PHP-FPM、MariaDB 三项服务，最后按原路返回，构成一个闭环。因此，要提高 LNMP 架构的安全性和性能，就必须对 LNMP 架构每一项服务进行技术改进。

2 安全性改进

确保在互联网环境下的访问安全是服务器建设中最重要的问题。LNMP 架构服务器安全措施包括磁盘规划、以编译方式安装最新源代码、开放有限端口、限定登录账户、规划文件读取权限和使用 HTTPS 访问协议等。

2.1 磁盘规划

服务器上应该使用多个物理磁盘，构建某种磁盘阵列，某个磁盘损坏时可以直接用新磁盘替换，从而不影响服务器的运行。例如 RAID 5，服务器上有 3 个以上硬盘即可做此模式，一份数据分成 3 份写，如果一个硬盘坏了，其他数据不会受损失，抽出坏的硬盘再插入新的硬盘即可[3]。

在安装 CentOS 操作系统时，应该将系统所需的不同的目录分别安装在不同的物理磁盘分区上，以避免某一个目录出现错误时殃及其他目录或整个磁盘。表 1 列出了不同分区和所需的磁盘空间，其中/boot 引导分区用来存储 Linux 内核，/根分区用来存储 CentOS 操作系统所需的其他文件，而服务器中所安装的其他软件如 Nginx、MariaDB、PHP 及其需要保存的数据则放在/opt 分区内。还可以进一步细化，将/根分区中的用户目录和临时目录等放于不同的分区中。这样确保了运行中出现的问题限定在某一个分区内，能最大限度地减少损失。

表 1 典型的 CentOS 磁盘分区规划

分区类型	文件系统	容量
swap 交换分区 1	N/A	2 GB
swap 交换分区 2	N/A	2 GB
/boot 引导分区	Ext4	200 MB
/根分区	xfs	40 GB
/opt 软件安装分区	xfs	其余空间

如果 LNMP 服务器配有专用的大容量磁盘阵列服务器时，可以使用网线将二者的网卡直连，两块直连网卡使用 192.168.0.x 地址组建成一个局域网，在 LNMP 服务器中采用 mount 命令将磁盘阵列服务器挂载到 CentOS 操作系统的一个目录中，如 mount 192.168.0.100:/vol1/mnt/array，最后将 CentOS 操作系统中 LNMP 架构的用户数据或者备份数据直接写在/mnt/array 即磁盘阵列设备上，这样就确保了 LNMP 架构在物理层面上的安全性。

2.2 编译安装最新源码

LNMP 架构中 Nginx、MariaDB 和 PHP 都是开源软件，官方一直在做功能、性能或安全性上的改进，因此不宜采用 LNMP 一键安装包或 RPM 安装包，而应去官网下载最新的适用于 Linux 操作系统的源代码包，直接在 CentOS 操作系统上现场配置、编译和安装。这样可以确保得到最新的稳定版本，而且可以生成最适合服务器 CPU 运行的二进制代码。同时，管理员也清楚每个软件都安装到哪个文件夹中，便于日后的升级和维护。

CentOS 操作系统安装好后，应该立即使用 yum-y update 命令对操作系统进行一次全面更新，而且要定期使用这个命令更新系统。系统更新后，采用 rpm-q kernel 查看是否升级了内核，若升级，则需重启操作系统，重启后会自动运行最新内核。这时，使用 uname-a 查看正在运行的内核，并用 rpm-e 将旧版本的内核删除，否则分配给内核的 200 MB 磁盘空间会不够用，下次将无法升级新版内核。

还要立即升级 CentOS 操作系统中的 gcc、cmake 和 libzip，这三个软件都是编译其他软件（如 Nginx、MariaDB 和 PHP）所必须的。

通过 g ++ - - version 命令，可以看到 CentOS 中

g ++ 的版本为 g ++ (GCC) 4.8.5 20150623 (Red Hat 4.8.5 - 39),由于该版本太老,因此须到官网(https://gcc.gnu.org/)上下载 gcc - 9.2.0.tar.gz 源代码编译安装。安装完成后,必须要把/usr/lib64/中的 libstdc ++.so.6 链接到新版本的/usr/local/lib64/libstdc ++.so.6.0.27 文件,否则 g ++ 编译器将无法使用 gcc - 9.2.0。

从 cmake 官方网站(https://cmake.org/download/)上下载最新的 cmake - 3.15.1.tar.gz 源代码包,编译安装完成后,调用 cmake-version 时,会出现"段错误 (core dumped)"而无法使用,执行 hash - r 命令可以解决此错误。

libzip 是编译 PHP 所必须的支持包,CentOS 默认没有安装 libzip,要去官网(https://libzip.org/)下载最新版本的 libzip - 1.5.2.tar.gz 源代码包,编译安装后,运行 libzip 时会出现找不到 zipconf.h 的错误,还需复制/usr/local/lib/libzip/include/zipconf.h 到/usr/local/include/zipconf.h 位置。

Nginx、MariaDB 和 PHP 都需要从官网上下载最新源代码,然后在本机上现场编译安装,Nginx 安装在/opt中,而 MariaDB 和 PHP 安装在/usr/local/文件夹中。要定期关注这三个软件的更新情况,并要随官方的修订及时更新它们,以确保最近发现的漏洞能被及时修复。

特别地,以前 PHP 使用 Oracle 官方发布的 MySQL 驱动 libmysql 与 MariaDB 服务进行通信,为了避免版权问题和程序的非可控性,PHP 已自行开发了 mysqlnd 本地驱动替代 libmysql,所以不再需要先安装 MariaDB 再安装 PHP 了,传统的安装 PHP 的方式中,在编译 PHP 时,需要指定以下几项:

- - with - mysql = /usr/local/mysql

- - with - mysqli = /usr/local/mysql/bin/mysql_config

- - with - pdo - mysql = /usr/local/mysql

现在使用 mysqlnd 驱动,编译 PHP 时,上述本条配置应该改为:

- - with - mysql = mysqlnd

- - with - mysqli = mysqlnd

- - with - pdo - mysql = mysqlnd

安装完成后,如果在 phpinfo 输出的 mysqli 项中发现 Client API library version 为 mysqlnd 5.0.12 - dev - 20150407,说明 mysqlnd 驱动 mysqlnd 5.0.12 - dev - 20150407 已经安装成功,PHP 脚本可以使用 pdo_mysql 和 mysqli 这两种 API Extensions 通过 mysqlnd 驱动与 MariaDB 数据库服务器通信。

CentOS 操作系统中各种软件都采用 OpenSSL 函数库进行加密,但 OpenSSL 在 2014 年 4 月 8 日曝光了一个名为 heartbleed 的漏洞。利用该漏洞,约 30% 以 https 开头网址的用户登录账号密码通过网络被窃取[4]。所以必须从 OpenSSL 官网(https://www.openssl.org)下载,将 OpenSSL 1.0.2k - fips 26 Jan 2017 升级为 OpenSSL 1.1.1d 10 Sep 2019。同时,使用 OpenSSL 加密函数库的 Nginx、PHP 和 OpenSSH 都必须采用新安装的 OpenSSL 加密函数库重新编译安装,以确保网站访问安全和服务器 SSH 远程访问安全。

2.3　服务器访问限制

LNMP 服务器编译安装好后,为确保服务器在互联网上的访问安全,必须在开放有限端口、限定登录账户、规划文件读取权限等方面作出限制。

通常情况下,服务器只开放 80 号 Web 服务端口、22 号 SSH 远程访问端口和 443 号 https 访问端口,同时关闭不需要的服务,这样可以减少暴露在互联网中的漏洞。同时,删除 firewalld.service 防火墙,安装 iptables-services 防火墙,这样易于使用。

同时,在 CentOS 操作系统中,要禁止在 ssh 中使用 root 直接登录,要求用户先用普通账号登录 CentOS 操作系统,再通过 su 命令输入 root 账号密码才可以登录 root 账号,这样可以防止针对 root 账号的远程暴力破解。因此,首先要在 CentOS 中创建一个普通账号,再在/etc/ssh/sshd_config 中设置 PermitRootLogin no 即可。

在 PHP-FPM 服务器的配置文件 php-fpm.conf 中,设置 user = www 和 group = www,使得只有 www 组的 www 用户才可能访问 PHP-FPM 服务。

MariaDB 数据库服务必须指定允许来自某台客户机的某个账号来连接自己。在 MySQL 数据库的 user 表中,只留下 Host 为 localhost 和 127.0.0.1 和 User 为 root 的这两条记录,以此只允许来自当前服务器的 root 账号连接 MariaDB 数据库服务,从而确保了 MariaDB 数据库服务的安全。

对 CentOS 操作系统中每个文件和文件夹,都可以通过 chgrp 命令设置其隶属于哪个组,通过 chown 命令来设置它隶属于哪个用户,并通过 chmod 命令来设置所有者、所属组其他用户和其他组对它的操作权限。操作权限用 4、2、1 数字来代表,4 表示读取权限,2 表示写入权限,1 表示执行权限[5]。如 775 这三个数字代表拥有者、组用户、其他用户的权限分别为 7 = 4 + 2 + 1、7 = 4 + 2 + 1、5 = 4 + 1,第一个数字 7 表示所有者具有读取、写入和执行权限,第二个数字 7 表示与所有者同组的用户具有读取、写入和执行权限,第三个数字

5 表示其他用户具有读取和执行权限。对访问用户、访问组和其他组的操作权限限定,使得 CentOS 操作系统中的文件访问得到有效地保护。

2.4 使用 HTTPS 访问协议

Nginx 默认采用 HTTP 协议与客户端浏览器通信,该协议采用明文方式传输网页和用户账号密码,容易受到网络中间人的窃密,所以极不安全。Nginx 服务器必须采用 HTTPS 安全传输协议与用户端浏览器通信。用户在首次访问 Nginx 服务器时,Nginx 服务器会将使用私钥加密的包含对应公钥的数字证书 CA 发送到客户端浏览器中,以后用户发送访问请求给 Nginx 服务器时,先用浏览器中的数字证书中的公钥对信息进行加密,再发送给 Nginx 服务器;Nginx 服务器在接收到用户的访问信息后,用服务器上的对应的私钥解开信息;在完成用户提交的请求后,将相应网页用服务器中对应的私钥加密后,再发送给客户端浏览器;客户端浏览器接收到信息后,会用浏览器中该 Nginx 服务器对应的数字证书中的公钥解密,完成双向加密通信。

将私钥和包含对应公钥的数字证书拷贝到 Nginx 服务器上后,只需在 Nginx 的配置文件中编写如下配置即可:

```
server {
listen 443 ssl;
server_name localhost;
ssl_certificate /etc/pki/tls/certs/server. cer;
ssl_certificate_key /etc/pki/tls/private/server. key;
```

重启 Nginx 服务器后,客户端浏览器就可以采用 https://方式来访问网站了,这样确保了通信安全。

3 处理速度改进

对软件的运行方式进行优化,才可以充分发挥服务器硬件性能,使得服务器处理 Web 请求的速度大大加快,从而避免服务器横向扩展,一方面降低了服务器购置成本,另一方面降低了软件管理上的复杂度。可以从如下几个方面进行技术改进来加快 LNMP 服务器的处理速度:用内存代替磁盘、使用 PHP 缓存、多进程并行处理、优化 InnoDB 存储引擎性能和压缩后再传输。

3.1 用内存代替磁盘

内存的存取速度远高于磁盘的存取速度,因此把网站的程序文件和数据库直接放在内存中,将会大大提高网站的访问速度。目前最大容量内存是单条 128 GB,一块 CPU 可以插 8 条内存,也就是 1 TB。如果是双路 CPU,最大内存可达 8 TB。网站代码磁盘占用量一般在 1 GB 以内,MariaDB 数据库的数据文件一般在 20 GB 以内,因此一台 64 GB 的服务器就可以将网站和数据库全部放入内存直接存取。

将磁盘文件放入内存的方法,就是使用 TMPFS 文件系统。TMPFS,即临时文件系统,是最好的基于 RAM 的文件系统,是由 Linux 内核的 VM 子系统管理的。TMPFS 初始容量默认是物理内存大小的一半[6]。指定的 TMPFS 内存空间并不会被锁定独占,只有挂载存储文件后才会占用相应大小的空间。

使用 TMPFS 文件系统最大的风险是意外掉电后该文件系统中的文件会全部消失。为了防止服务器意外掉电,确保数据安全,一方面定期自动备份数据库,另一方面,确保服务器上的双电源都有效。使用通信型 UPS 不间断电源通过联机 USB 线实时监测市电,一旦发现断电,立即会切换到电源供电,并自动将内存中的文件保存到磁盘中,再自动关闭 CentOS 服务器,最后自动关闭 UPS。整个处理过程只需 5 分钟,所以 UPS 只需 600 W 的功率,如 LADIS H1000 600 W 型 UPS。

3.2 使用 PHP 缓存

使用 PHP 缓存,可以大大减少同一网页再次被访问时 PHP 的编译时间开销,从而大大加快网页访问速度。

PHP 是解释型语言而非编译型语言,每次调用 PHP 文件时,翻译器先将 PHP 文件翻译成易于执行的中间代码(即 opcode 字节码)后再执行,执行完后所占用的内存马上被释放,基本上所有数据包括 opcode 字节码在此时都被销毁。opcode 字节码并非目标机器代码,不能直接在硬件上运行。该 PHP 文件第二次被调用时,同样还是会被重新转换为字节码,但很多时候文件内容几乎是一样的,比如静态 HTML 文件生成字节码后其内容很久都不会改变,这样就非常浪费时间。

为了避免上述问题,PHP 开发了 Opcache 组件。该组件的前身是 Optimizer + ,它是 PHP 的官方公司 Zend 开发的一款闭源但可以免费使用的 PHP 优化加速组件,2013 年 3 月中旬 Optimizer + 改名为 Opcache,在 PHP 源代码中已经包含了该组件。启用 PHP 的 Opcache 组件后,PHP 会缓存 PHP 字节码,使得再次调用相同的 PHP 文件时,无须编译,直接从 Opcache 缓存中获取字节码去运行;同时,Opcache 还应用了一些代码优化模式,使得代码执行更快,从而加速 PHP 的执行,使 CPU 消耗少了许多,PHP 网页的响应时间也大幅缩短。图 2 是 PHP 程序使用 Opcache 缓存后的生命周期示意图。

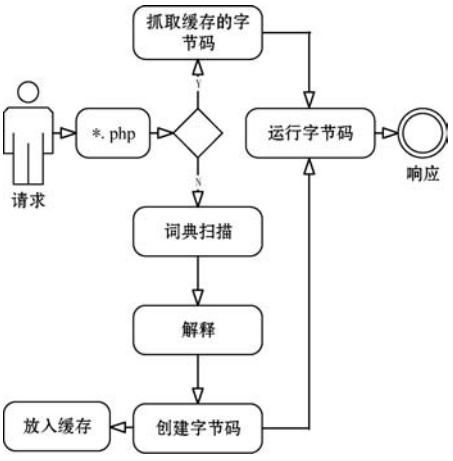

图 2　PHP 对象生命周期示意图

可以看到,传统的 PHP 的整个运行过程为:Request 请求(Nginx 等)→Zend 引擎读取. php 文件→扫描其词典和表达式→解析文件→创建要执行的计算机代码(称为 Opcode) →执行 Opcode→ Response 返回。使用了 Opcache 组件后,如果 PHP 网页没有变化,就直接从 Opcache 缓存中读取中间代码,再去执行,避免了 CPU 的重复计算。

3.3　多进程并行处理

为了充分利用 CPU 多核特征,Nginx 和 PHP 都设计为多进程的工作方式,以提高处理效率。从图 3 所示的 LNMP 架构示意图中可以看出,客户端浏览器与 Nginx 管理进程交互后,Nginx 监控进程就会把具体的工作任务分配给一个空闲的 Nginx 工作进程,对于用户的 PHP 网页请求,Nginx 工作进程会把 PHP 的编译和运行工作交给 PHP-FPM 管理进程,PHP-FPM 则把具体的工作任务分配给一个空闲的 PHP FastCGI 工作进程,PHP FastCGI 会把要访问的 PHP 网页编译并运行,并存取 MariaDB 数据库,最后把结果组织成 HTML 超文本,回传给用户端浏览器。

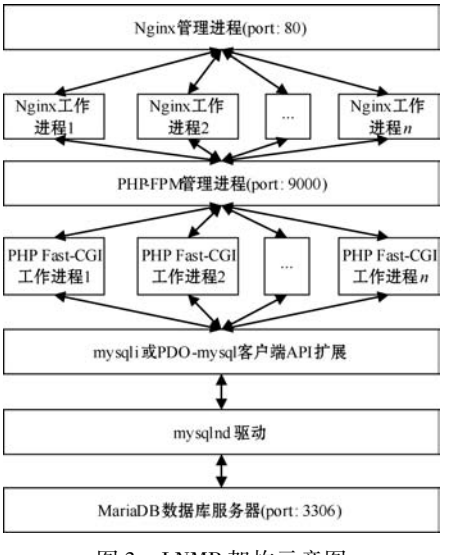

图 3　LNMP 架构示意图

因此,为 Nginx 和 PHP-FPM 设置合适的工作进程数量和工作方式,可以加快 LNMP 整体架构的处理速度。Nginx 通过 worker_processes 指令本配置开启多少个工作进程,其数量一般为 CPU 的逻辑核心数量(即 CPU 超线程总量)。例如,双 CPU 共 16 个物理核心,每个物理核心上有 2 个超线程,那么 worker_processes 应该设置为 32。

PHP-FPM 进程池(pm)有 static(静态)、dynamic(动态)和 ondemand(按需)三种 PHP FastCGI 工作进程启动方式。

如果将 pm 设置为 static,那么 PHP FastCGI 工作进程始终保持 pm. max_children 个数目。

如果将 pm 设置为 dynamic,那么 PHP-FPM 管理进程首先启动 pm. start_servers 个工作进程,遇到访问高峰时,会自动增加工作进程数量,确保任何时候都有 pm. min_spare_servers 个空闲进程存在。访问高峰过后,如果空闲进程多于 pm. max_spare_servers 个时,多余的空闲进程会被杀掉;而同时能存活的最大工作进程总量为 pm. max_children 个。

如果将 pm 设置为 ondemand,那么刚开始时 php-fpm 管理进程不会创建任何工作进程,当有请求时才会创建;当进程在 pm. process_idle_timeout 秒后还处于空闲状态就会被杀掉,默认时长为 10 秒。因此,在完全空闲时,只有一个 PHP-FPM 管理进程存在。在此模式下,能同时存活的工作进程最大数量为 pm. max_children 个。

要根据具体情况来选择这三种 PHP-FPM 的进程管理方式。如果内存足够大,可以选用 static 方式,按一个 PHP 进程占用 30 MB 内存的标准来确定 PHP 进程设置数量,由于不用在使用时临时创建 PHP 进程,因此这种方式效率最高,作为首选方式。如果服务器内存不宽裕,则可以选用 dynamic 方式,动态应付访问高峰,能兼顾内存与效率。ondemand 方式把节约内存放在首位,其缺点是遇到访问高峰或者如果 pm. process_idle_timeout 的值太短的话,服务器会频繁创建进程,用户会感受到明显的网络延迟。

3.4　优化 InnoDB 存储引擎性能

MariaDB 的存储引擎采用插件式工作方式,而 InnoDB 存储引擎作为事务型数据库的首选引擎,支持事务安全表(ACID),能与 MariaDB 数据库系统完美结合。MariaDB 系统接收的用户 SQL 请求和传输的数据最后都要通过 InnoDB 存储引擎与磁盘设备交互。LNMP 架构中用户请求数据链最后都要经过 InnoDB 存储引擎,因此 InnoDB 数据库的存取性能决定了

LNMP 架构的性能。MariaDB 中的 InnoDB 实际上是 XtraDB,它是 InnoDB 的增强版,但 MariaDB 的配置文件中仍然标记为 InnoDB。InnoDB 存储引擎采用缓冲池来缓存数据,最后按设定的时间间隔将数据写到磁盘中去持久化。

设置 InnoDB 的缓存池的大小 innodb_buffer_pool_size 为所有内存的 80% 以上。当然,需要除去 CentOS 操作系统、PHP 进程和 Nginx 进程的内存开销,余下的都设置为 InnoDB 的缓存。在 64 GB 的内存下,一般 16 GB InnoDB 缓存就足够使用了,其中操作系统、TMPFS 文件系统、Nginx 进程和 PHP 进程分配 16 GB 内存,其余 32 GB 全部分配给 MariaDB 的线程。

innodb_log_files_in_group 日志文件分为 3 组就足够了[7]。此外,为了避免在日志文件覆盖时产生不必要的缓冲池刷新活动,每个日志文件的大小应该正好占缓存大小的 25%。这样,innodb_log_file_size 正好设置为 innodb_buffer_pool_size 的 1/12。

假如 InnoDB 缓冲的大小为 16 GB 时,每个 innodb_log_file_size 就是 1 365 MB 了。

InnoDB 内核中允许的线程数量依赖服务器硬件性能,设置太高会导致线程抖动。一般将 innodb_thread_concurrency 设置为 16 即可。

InnoDB 存储引擎中用来同步操作系统 I/O 的写线程 innodb_write_io_threads 和读线程 innodb_read_io_threads 在 UNIX/Linux 操作系统上都设置为 8 个。

innodb_flush_log_at_trx_commit 的值可以设置为 0、1 或 2。设置为 2 时,每个交易在提交时都会写日志到文件缓存中,且大约每秒将日志文件缓存刷新到磁盘中去持久化,这样使得 InnoDB 的日志存盘时间开销达到最少,又兼具数据安全。

3.5 压缩后再传输

经过前面的 LNMP 服务器性能优化后,处理 PHP 页面的速度得到了加快,访问 LNMP 架构的性能瓶颈就落在网页从 Nginx 服务器传输到用户浏览器的网络带宽上了。HTTP 协议使用 gzip 压缩网页内容,能极大减少传输内容的大小,从而使网页访问速度得到大幅提升。

gzip 是 GNUzip 的缩写,它是一个 GNU 自由软件的文件压缩程序,由 Jean-loup Gailly 和 Mark Adler 一起开发。其工作原理是:在一个文本文件中找出类似的字符串,并临时替换它们,使整个文件变小。这种形式的压缩对 Web 来说非常适合,因为 HTML 和 CSS 文件通常包含大量的重复的字符串,例如空格、标签。检测表明,Nginx 启用 gzip 压缩后,网页字节数缩小为原先的 20%,其压缩率高达 80%,有效减少了网络带宽开销,大幅提高了网页的浏览速度。

gzip 的压缩过程如下:

(1) 浏览器发送 HTTP Request 给 Web 服务器,Request 中有 Accept-Encoding: gzip、deflate 字符串,它告诉服务器,当前的浏览器支持 gzip 压缩。

(2) Web 服务器接到 Request 后,生成原始的 Response,其中有原始的 Content-Type 和 Content-Length。

(3) Web 服务器通过 gzip,来对 Response 进行编码,编码后 header 中有 Content-Type 和 Content-Length (压缩后的大小),并且增加了 Content-Encoding: gzip 字符串,然后把 Response 发送给浏览器。

(4) 浏览器接到 Response 后,根据 Content-Encoding:gzip 字符串来对 Response 进行解码,获取到原始 Response 后,显示出网页。

Nginx 服务器是通过 ngx_http_gzip_module、ngx_http_gzip_static_module、ngx_http_gunzip_module 三个模块对指令进行解析处理。其中:ngx_http_gzip_module 模块对 Response 进行压缩;ngx_http_gzip_static_module 负责搜索和发送经过 gzip 处理的数据;ngx_http_gunzip_module 用于对后端服务器或者预压缩数据解压,为了防止浏览器不能解压,用此模块解压。

Nginx 服务器中,gzip 的压缩级别 gzip_comp_level 可以设置为 1 ~9 级,级别越高,压缩率更高,但服务器的 CPU 开销也越大,访问高峰时,会影响 LNMP 架构的整体性能。同时,由于压缩级别越高,压缩比提高并不是非常明显,所以需要权衡压缩比与 CPU 开销增长之间的关系,当 gzip_comp_level 设置为 6 时,性价比最好。

Nginx 通过在 nginx. conf 配置文件中添加 gzip on 指令来启用 gzip 压缩功能。为了避免对于访问同一个文件时被重复压缩,可以开启静态压缩模块 ngx_http_gzip_static_module,在使用 gzip_static on 指令后,Nginx 会直接读取之前已经压缩好的文件(文件名为 *. gz),而不是现场动态压缩,这样再次加快了网页访问速度。

4 抗拥塞改进

上文对 LNMP 架构各个环节的优化最大可能地缩短了服务器处理延迟和线路处理延迟,使服务器能在 0.5 秒左右完成单个用户的 HTTP 请求处理的全部环节,用户获得了极快的网页响应体验。

但当 LNMP 架构遇到瞬间大量的 HTTP 请求时,服务器就会出现响应缓慢、卡死甚至数据库服务器崩溃的典型现象。因此,对服务器进行抗拥塞改进成为

整个 LNMP 架构服务器技术改进的重中之重。

实践表明,遇到访问高峰时,服务器出现拥塞的原因在于:(1) MariaDB 数据库默认采用了实验室模式的线程模型,使得访问高峰时响应缓慢;(2) PHP 工作进程数过多,导致 MariaDB 连接数同步变大,造成数据库服务崩溃。

4.1 MariaDB 采用线程池模型

PHP 与 MariaDB 建立连接后,MariaDB 通过工作线程来处理来自 PHP 的请求。由于 MariaDB 数据库默认的线程模型是 one-thread-per-connection,即会为每个 PHP-MariaDB 连接创建一个独立的线程,处理 PHP 请求的任务完成后,该线程的生命周期就结束了。遇访问高峰时,MariaDB 创建的线程数量激增,导致操作系统频繁的上下文切换和对系统资源的竞争,造成访问阻塞,LNMP 架构吞吐量急剧下降,直到无法正常提供 HTTP 服务。

Oracle 公司的 MySQL 5.5 商用版采用线程池插件(Thread Pool Plugin)有效地解决了这个问题。由开源社区维护的 MariaDB 从 5.5 版本开始内建了线程池(Thread Pool),先将连接分配到不同的组的队列里,以语句(statement)为最小单位,排队处理用户的 SQL 请求,不光减少了同时运行的线程的数量,还减少了上下文切换的次数;尤其是通过优先队列和普通队列运行机制,保证了最短时间内完成对一个事物(transaction)的处理,减少了并行处理事务的机率,最大可能地防止了对热点资源的竞争产生死锁;同时,线程的复用降低了创建和销毁线程的 CPU 开销,使得在访问高峰时 MariaDB 数据库能保持恒定的最高的吞吐量,有效解决了阻塞问题[8]。

为了适应不同的运行场景,MariaDB 线程池提供了以下 8 个可调节参数:

thread_handling = pool-of-threads
thread_pool_size = 32
thread_pool_priority = auto
thread_pool_prio_kickup_timer = 1 000
thread_pool_stall_limit = 60 ms
thread_pool_oversubscribe = 10
thread_pool_max_threads = 65 536
thread_pool_idle_timeout = 500

通常只需要在 MariaDB 的配置文件 my. cnf 中设置 thread_handling = pool-of-threads 线程池就能很好地运行了,其余 7 个参数 MariaDB 默认会直接复制 CentOS 操作系统的线程池运行参数。

线程池通过使用分组的方法来限制和平衡并发,隔离了用户连接和 SQL 请求语句,高优先级队列使一个完整的事务能尽快完成,减少了并行事务的数量,最大可能降低事务对热点资源的抢占,同时使用 Timer 定时线程防止线程组运行停滞,实现了高并发下平稳的高吞吐量,防止 LNMP 架构访问高峰时的阻塞现象。

Thread pool 取得了良好的应用效果,根据 Oracle MySQL 官方的性能测试:在并发达到 128 个连接以后,没有线程池的 MySQL 性能会迅速降低。使用线程池以后,性能不会出现波动,会一直保持在较好的状态运行。在读写模式下,128 个连接以后,有线程池的 MySQL 比没有线程池的 MySQL 性能高出 60 倍。在只读模式下,512 个连接以后,有线程池的 MySQL 比没有线程池的 MySQL 性能高出 18 倍。

4.2 固定 PHP 进程数量

在 LNMP 架构中,来自客户端的请求通过 Nginx 传达到 PHP-FPM 服务器中,PHP-FPM 服务器将这些用户请求排队发送给 PHP 工作进程去处理,每个 PHP 工作进程都会与 MariaDB 建立一个连接,MariaDB 则通过某种线程模型去处理来自 PHP 连接的 SQL 请求。通过观察发现,PHP 工作进程的数量与 MariaDB 所创建的连接 connection 数量是相等。也就是说,如果 PHP 进程模型采用的是 dynamic(动态)方式的话,同时有 1 000 个用户来连接 LNMP 服务器,那么 PHP-FPM 服务器将创建 1 024 个 PHP 工作进程,此时,MariaDB 数据库服务器也将创建 1 024 个 connection 连接,并同时将这 1 000 个连接交给自己的工作线程去处理。在如此大的连接处理的压力下,MariaDB 的工作线程,无论是 one-thread-per-connection 每个连接一个线程的模式还是 pool-of-threads 进程池处理模式,都会无力运转。因此,固定 PHP 进程数量,成为解决问题的关键。

所以,PHP 应该采用 static(静态)工作方式,设置为 64 个固定的静态进程并行处理就可以了,MariaDB 同时建立的 connection 连接数最大就是 64,这很好地保护了后端的 MariaDB 数据库服务器的正常运行。

通常,可以根据先 MariaDB 最大允许的连接数量反过来确定 PHP 工作进程设置的最大数量。例如,MariaDB 以每个连接 42.2 MB 的最大内存用量来计算的话,如果分配 32 GB 的内存用量给 MariaDB 连接使用,那么 MariaDB 的最大连接数 max_connections 可以设置为 776,也就是说 PHP 工作进程数量也可以设置为 776。但考虑到每个 PHP 工作进程最大 30 MB 的内存开销和 LNMP 架构整体 CPU 计算、内存开销情况,一般 PHP 设置为 64 个进程同时运行就足够了。以此

很好地保护了 MariaDB 服务器,即使遇到访问高峰,MariaDB 数据库服务器承受的计算压力也稳定在一个可承受水平上,不至于被压垮。

5 LNMP 服务器测试

5.1 服务器配置

被检测的 LNMP 服务器是闪电 Moodle 服务器 https://mood.nbpt.edu.cn/;服务器 CPU 型号及数量是 Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60 GHz,2 块,共 16 核心 32 线程;内存型号及容量是 DDR 3,64 GB;硬盘是 SCSI 硬盘,转速 15 000 r/min,共 4 块 300 GB 的硬盘,建了 Raid 5 磁盘阵列。

5.2 压缩比测试

Nginx 中网页传输采用 gzip 压缩后,经过 http://tool.chinaz.com/Gzips/Default.aspx? q = mood.nbpt.edu.cn 网站统计,压缩率(估计值)达到了 73.07%,即减少了 73.07% 的数据传输的体积。

5.3 安全性检测

经国内某知名的网络安全公司对该 LNMP 服务器进行的权威安全性检测,没有发现紧急、高危和中危漏洞,通过了 ITSS(信息技术服务标准)、ITIL v3(信息技术基础设施库)、ISO/IEC 20000(IT 服务管理国际标准体系)、ISO/IEC 27000(信息技术-安全技术-信息安全管理体系-要求)系列服务标准下的安全评测。使用的检测软件是 WebScan(V6.0.1.9),Engine Version 是 V6.1.79, Policy Version 是 V6.1.109。

5.4 性能测试

对 LNMP 服务器整体性能测试,采用 Chrome 浏览器自带的 Network 网页加载计时工具,从浏览器发出 PHP 读取 MariaDB 数据库的请求,到浏览器收到运行结果,即针对一个完整的 Request 和 Response 的 PHP 存读取数据库操作计时,发现优化前 https://mood.nbpt.edu.cn/index.php 主页中 PHP 访问 MariaDB 加载耗时稳定在 352 ms 左右;而优化后,主页加载耗时稳定在 124 ms 左右,处理速度达到优化前的 2.8 倍,优化效果明显。

5.5 抗拥塞测试

在整个 LNMP 架构中,拥塞出现在 MariaDB 服务这个节点中。因此,采用 sysbench OLTP RO 对 MariaDB数据库进行了只读压力测试,考察其在高线程时的吞吐量 TPS,即每秒钟系统能够处理的交易或事务的数量,如图 4 所示。

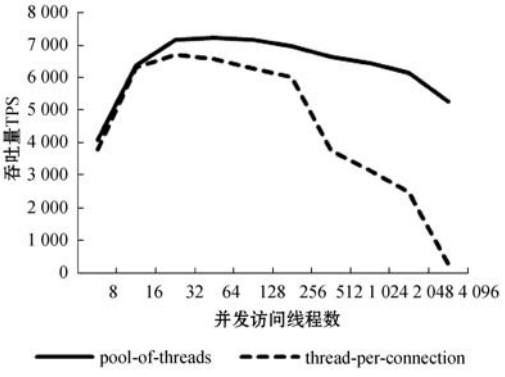

图 4 两种线程调度器吞吐量对比

可以看出,在高达 4 096 个线程并发读取的情况下,新的 pool-of-threads 线程池吞吐量 TPS 能稳定在 5 250;而传统的 thread-per-connection 调度器 TPS 则急速下降到 268,几乎被压垮。测试表明,使用线程池设计的 MariaDB 数据库服务器具有抗拥塞特征,能应付网络访问高峰。

6 结 语

本文详细论述了 LNMP 服务器在安全性、处理速度和抗拥塞性上的深度技术改进,应用在闪电 moodle 服务器(网址 https://mood.nbpt.edu.cn/)中,该服务器得到了多年的应用检验,用户体验度高,取得了很好的应用效果。

本文提出的技术改进对构建于 Linux 操作系统上的各类系统如 Docker 等的性能优化都具有重要的参考价值。随着技术进步,人们对 LNMP 服务器不断深入研究,LNMP 服务器也必将更加安全、快速和稳固。

参 考 文 献

[1] 曾棕根. 便携式高性能 WNMP 架构的研制[J]. 计算机应用与软件,2018,35(4):168 - 171.

[2] 苗泽. Nginx 高性能 Web 服务器详解[M]. 北京:电子工业出版社,2013.

[3] 徐伟. 基于备份的 RAID6 在线重构框架[J]. 计算机应用与软件,2018,35(5):48 - 54,101.

[4] 杨勇. OpenSSL Heartbleed 漏洞研究及启示[J]. 信息安全与通信保密,2014(5):99 - 102.

[5] 叶和平. Linux 特殊权限位及其在提权攻击中的应用[J]. 网络安全技术与应用,2017(4):28 - 29.

[6] 周狄波,王迪峰. 基于 Linux 的内存式 WebGIS 设计与实现[J]. 计算机应用,2009,29(7):1974 - 1977.

[7] 辛锋,袁荣喜,文如泉. InnoDB 重做日志深入分析[J]. 萍乡学院学报,2015,32(6):33 - 38.

[8] 李涛,董前琨,张帅,等. 基于线程池的 GPU 任务并行计算模式研究[J]. 计算机学报,2018,41(10):2175 - 2192.