架构师详解：Nginx 架构

Java架构分享 开源中国 2018-05-04

原文链接：https://my.oschina.net/u/3770281/blog/1802493

作者：Java架构分享

> 引言：众所周知，Nginx 服务器是一个高性能的 Web 和反向代理服务器。Nginx 在激烈的 Web 服务器竞争中依旧保持良好的发展势头，一度成为 Web 服务器市场的后期之秀，这一切跟 Nginx 的架构设计是分不开的。

一 Nginx 模块化设计

高度模块化的设计是 Nginx 的架构基础。Nginx 服务器被分解为多个模块，每个模块就是一个功能模块，只负责自身的功能，模块之间严格遵循“高内聚，低耦合”的原则。

Nginx 模块图

- 核心模块

核心模块是 Nginx 服务器正常运行必不可少的模块，提供错误日志记录、配置文件解析、事件驱动机制、进程管理等核心功能。

- 标准 HTTP 模块

标准 HTTP 模块提供 HTTP 协议解析相关的功能，如：端口配置、网页编码设置、HTTP 响应头设置等。

- 可选 HTTP 模块

可选 HTTP 模块主要用于扩展标准的 HTTP 功能，让 Nginx 能处理一些特殊的服务，如：Flash 多媒体传输、解析 GeoIP 请求、SSL 支持等。

- 邮件服务模块

邮件服务模块主要用于支持 Nginx 的邮件服务，包括对 POP3 协议、IMAP 协议和 SMTP 协议的支持。

- 第三方模块

第三方模块是为了扩展 Nginx 服务器应用，完成开发者自定义功能，如：Json 支持、Lua 支持等。

二 Nginx 请求处理方式

Nginx 是一个高性能的 Web 服务器，能够同时处理大量的并发请求。它结合多进程机制和异步机制，异步机制使用的是异步非阻塞方式，接下来就给大家介绍一下 Nginx 的多线程机制和异步非阻塞机制。

- 多进程

服务器每当收到一个客户端时。就有服务器主进程（master process）生成一个子进程（worker process）出来和客户端建立连接进行交互，直到连接断开，该子进程就结束了。

使用进程的**好处**是各个进程之间相互独立，不需要加锁，减少了使用锁对性能造成影响，同时降低编程的复杂度，降低开发成本。

其次，采用独立的进程，可以让进程互相之间不会影响，如果一个进程发生异常退出时，其它进程正常工作，master 进程则很快启动新的 worker 进程，确保服务部中断，将风险降到最低。

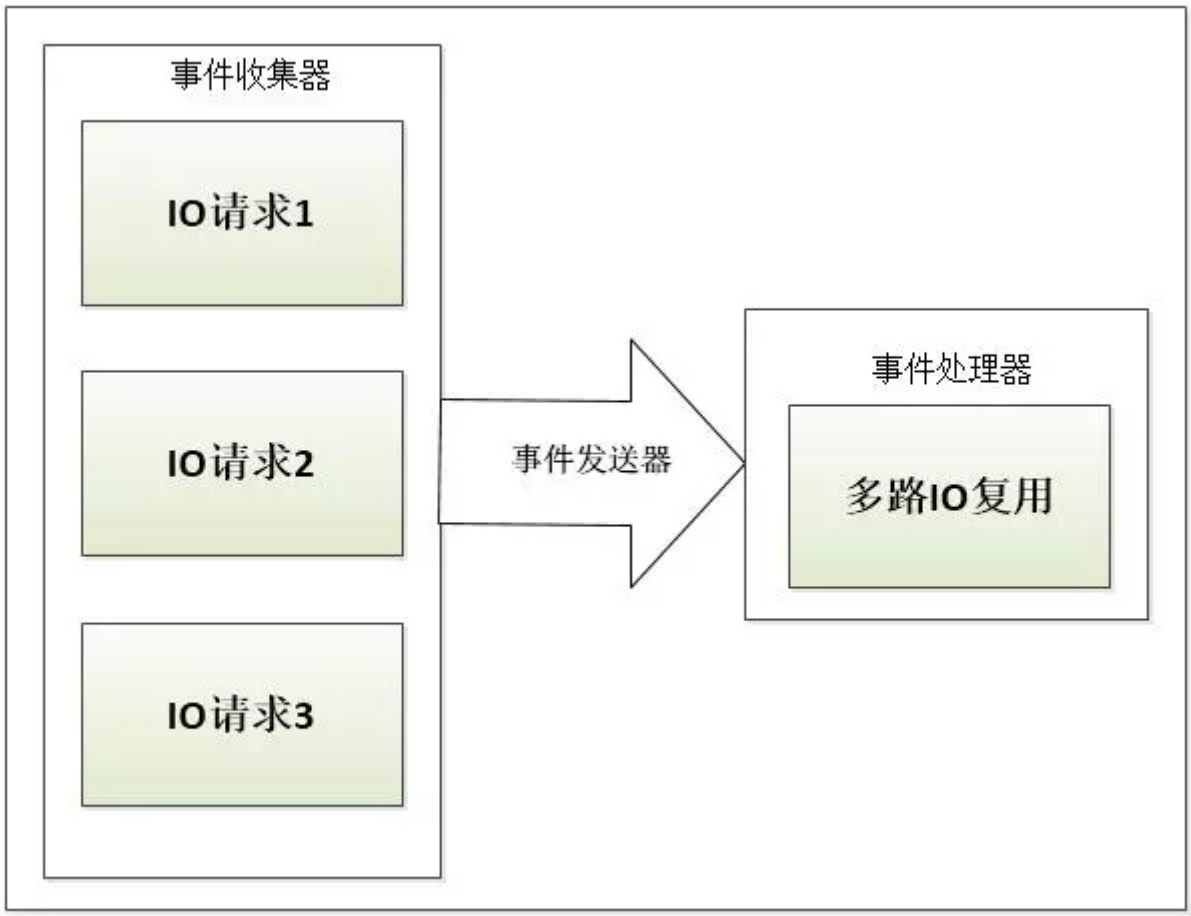
缺点是操作系统生成一个子进程需要进行内存复制等操作，在资源和时间上会产生一定的开销；当有大量请求时，会导致系统性能下降。

- 异步非阻塞

每个工作进程使用异步非阻塞方式，可以处理多个客户端请求。当某个工作进程接收到客户端的请求以后，调用 IO 进行处理，如果不能立即得到结果，就去处理其他的请求（即为非阻塞）；而客户端在此期间也无需等待响应，可以去处理其他事情（即为异步）；当 IO 返回时，就会通知此工作进程；该进程得到通知，暂时挂起当前处理的事务去响应客户端请求。

三 Nginx 事件驱动模型

在 Nginx 的异步非阻塞机制中，工作进程在调用 IO 后，就去处理其他的请求，当 IO 调用返回后，会通知该工作进程。对于这样的系统调用，主要使用 Nginx 服务器的事件驱动模型来实现。



Nginx 事件驱动模型

如上图所示，Nginx 的事件驱动模型由事件收集器、事件发送器和事件处理器三部分基本单元组成。其中，事件收集器负责收集 worker 进程的各种 IO 请求，事件发送器负责将 IO 事件发送到事件处理器，而事件处理器负责各种事件的响应工作。

事件发送器将每个请求放入一个待处理事件的列表，使用非阻塞 I/O 方式调用“事件处理器”来处理该请求。其处理方式称为“多路 IO 复用方法”，常见的包括以下三种：select 模型、poll 模型、epoll 模型。

针对上面的技术我特意整理了一下，有很多技术不是靠几句话能讲清楚，所以干脆找朋友录制了一些视频，很多问题其实答案很简单，但是背后的思考和逻辑不简单，要做到知其然还要知其所以然。

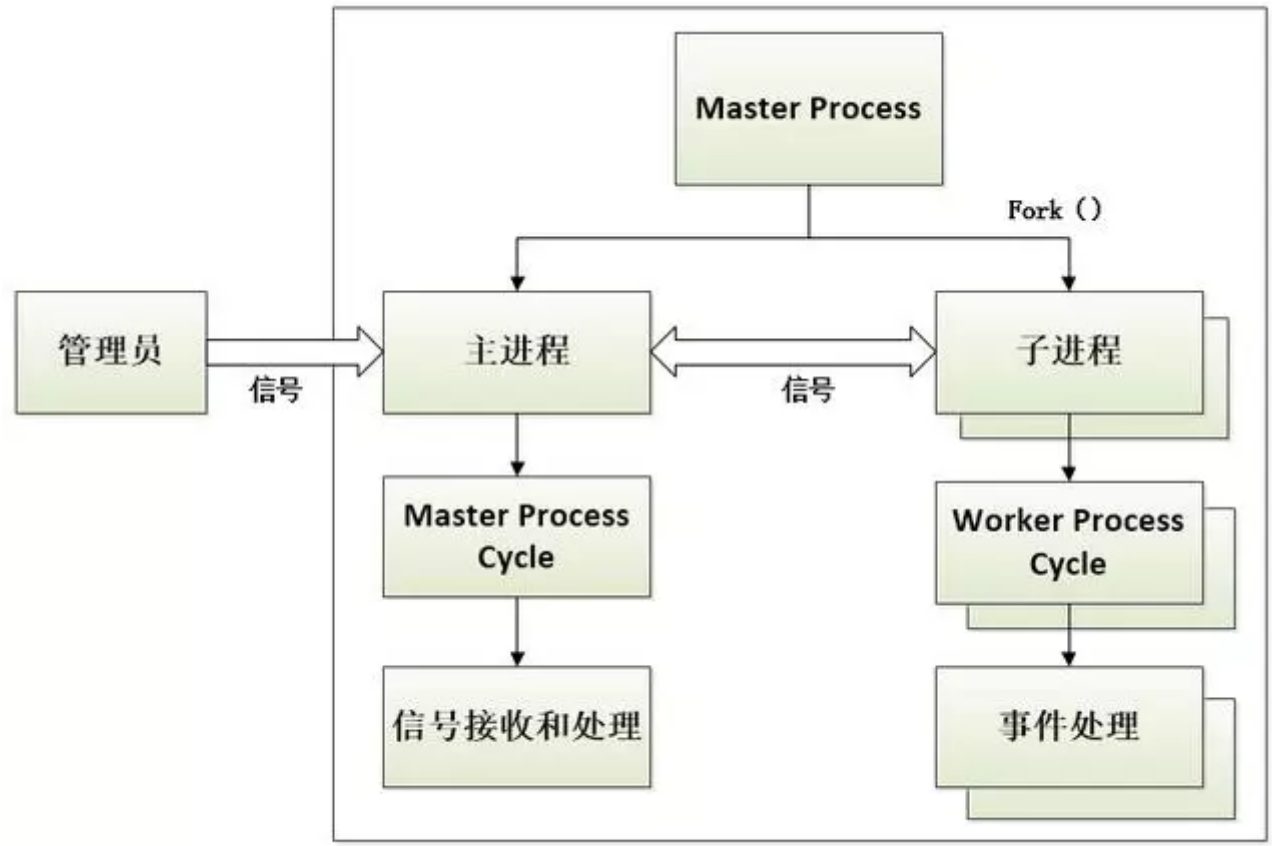
四 Nginx 设计架构

Nginx 服务器使用 master/worker 多进程模式。多线程启动和执行的流程如下：主程序 Master process 启动后，通过一个 for 循环来接收和处理外部信号；主进程通过 fork() 函数产生子进程，每个子进程执行一个 for 循环来实现 Nginx 服务器对事件的接收和处理。

一般推荐 worker 进程数与 cpu 内核数一致，这样一来不存在大量的子进程生成和管理任务，避免了进程之间竞争 CPU 资源和进程切换的开销。而且 Nginx 为了更好的利用多核特性，提供了 cpu 亲缘性的绑定选项，我们可以将某一个进程绑定在某一个核上，这样就不会因为进程的切换带来 cache 的失效。

对于每个请求，有且只有一个工作进程对其处理。首先，每个 worker 进程都是从 master 进程 fork 过来，在 master 进程里面，先建立好需要 listen 的 socket（listenfd）之后，然后再 fork 出多个 worker 进程。所有 worker 进程的 listenfd 会在新连接到来时变得可读，为保证只有一个进程处理该连接，所有 worker 进程在注册 listenfd 读事件前抢 accept_mutex，抢到互斥锁的那个进程注册 listenfd 读事件，在读事件里调用 accept 接受该连接。

当一个 worker 进程在 accept 这个连接之后，就开始读取请求，解析请求，处理请求，产生数据后，再返回给客户端，最后才断开连接，这样一个完整的请求就是这样的了。我们可以看到，一个请求，完全由 worker 进程来处理，而且只在一个 worker 进程中处理。



Nginx 架构图

在 Nginx 服务器的运行过程中，主进程和工作进程需要进程交互。交互依赖于 Socket 实现的管道来实现。

- Master-Worker 交互

这条管道与普通的管道不同，它是由主进程指向工作进程的单向管道，包含主进程向工作进程发出的指令，工作进程 ID 等；同时主进程与外界通过信号通信；每个子进程具备接收信号，并处理相应的事件的能力。

- worker-worker 交互

这种交互是和 Master-Worker 交互是基本一致的，但是会通过主进程。工作进程之间是相互隔离的，所以当工作进程 W1 需要向工作进程 W2 发指令时，首先找到 W2 的进程 ID，然后将正确的指令写入指向 W2 的通道。W2 收到信号采取相应的措施。

五 总结

通过这篇文章，我们对 Nginx 服务器的整体架构有了一个整体的认识。包括其模块化的设计、多进程和异步非阻塞的请求处理方式、事件驱动模型等。通过这些理论知识，对于我们以后学习 Nginx 的源码有很大的帮助；也推荐大家多看看 Nginx 的源码，才能更好地领悟 Nginx 的设计思想。

推荐阅读

谷歌终于开放 ".app" 顶级域名注册

可以抛弃 Python 了？Google 开源 Swift for TensorFlow 意味着什么

宣布 Java 8 停止维护后，Oracle 又毙掉了 JavaOne

期待已久的 Ubuntu 18.04 LTS 终于正式发布

图文详解 Java 字节码，想不懂都难

点击**“阅读原文”**查看更多精彩内容

阅读原文