讲师介绍

Hash　　QQ：805921455

从事Java软件研发近十年。

前新浪支付核心成员、

咪咕视讯(中国移动)项目经理、

对分布式架构、高性能编程有深入的研究。

明天，你一定会感谢今天奋力拼搏的自己

追求极致性能之Nginx性能优化手段

分布式高并发—负载均衡

目录

课程安排

01

Nginx程序运行原理分析

Nginx的工作模式、整体架构、优化比懂工作模式

02

Worker与CPU的绑定

Worker工作的模式、如何与CPU绑定提供处理能力？

03

Linux服务器参数性能优化

Linux服务器可以在哪些地方对web服务提供有力的土壤？

04

常用nginx性能优化

Nginx的常规调整

目录

Nginx程序运行原理分析

Nginx工作模式

Nginx在启动时会以daemon形式在后台运行，采用多进程+异步非阻塞IO事件模型来处理各种连接请求。多进程模型包括一个master进程，多个worker进程，一般worker进程个数是根据服务器CPU核数来决定的。master进程负责管理Nginx本身和其他worker进程。

```
$ ps -ef|grep nginx
root     24574     1  0 17:41 ?        00:00:00 nginx: master process bin/nginx -c
conf/nginx.conf
nobody   24575 24574  0 17:41 ?        00:00:00 nginx: worker process
nobody   24576 24574  0 17:41 ?        00:00:00 nginx: worker process
nobody   24577 24574  0 17:41 ?        00:00:00 nginx: worker process
nobody   24578 24574  0 17:41 ?        00:00:00 nginx: worker process
```

4个worker进程的父进程都是master进程，表明worker进程都是从父进程fork出来的，并且父进程的ppid为1，表示其为daemon进程。

Nginx工作模式

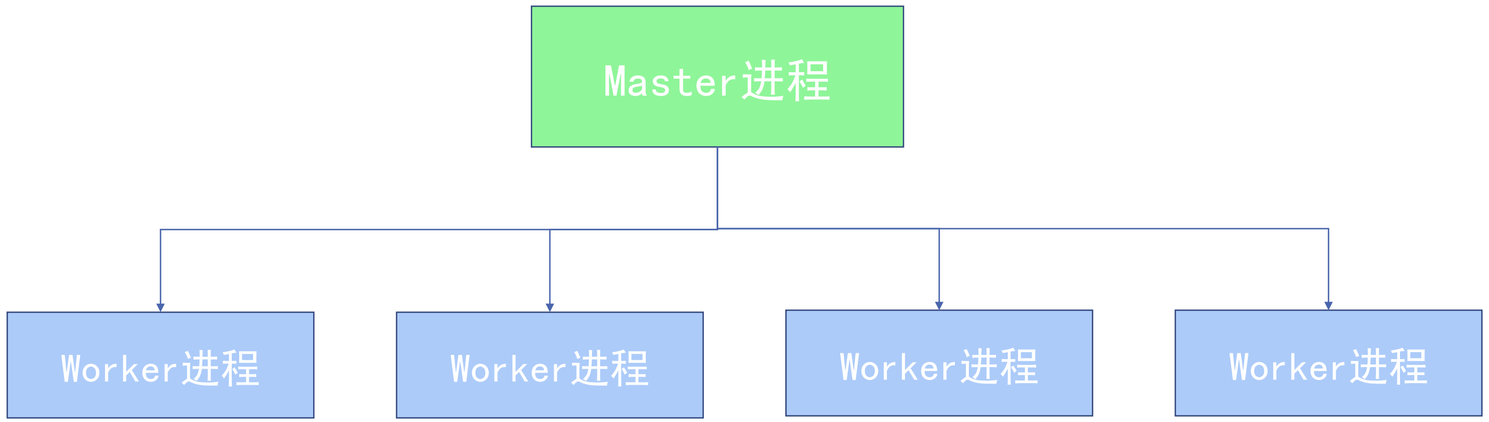
Nginx中的Master、Worker两种进程。

Master进程

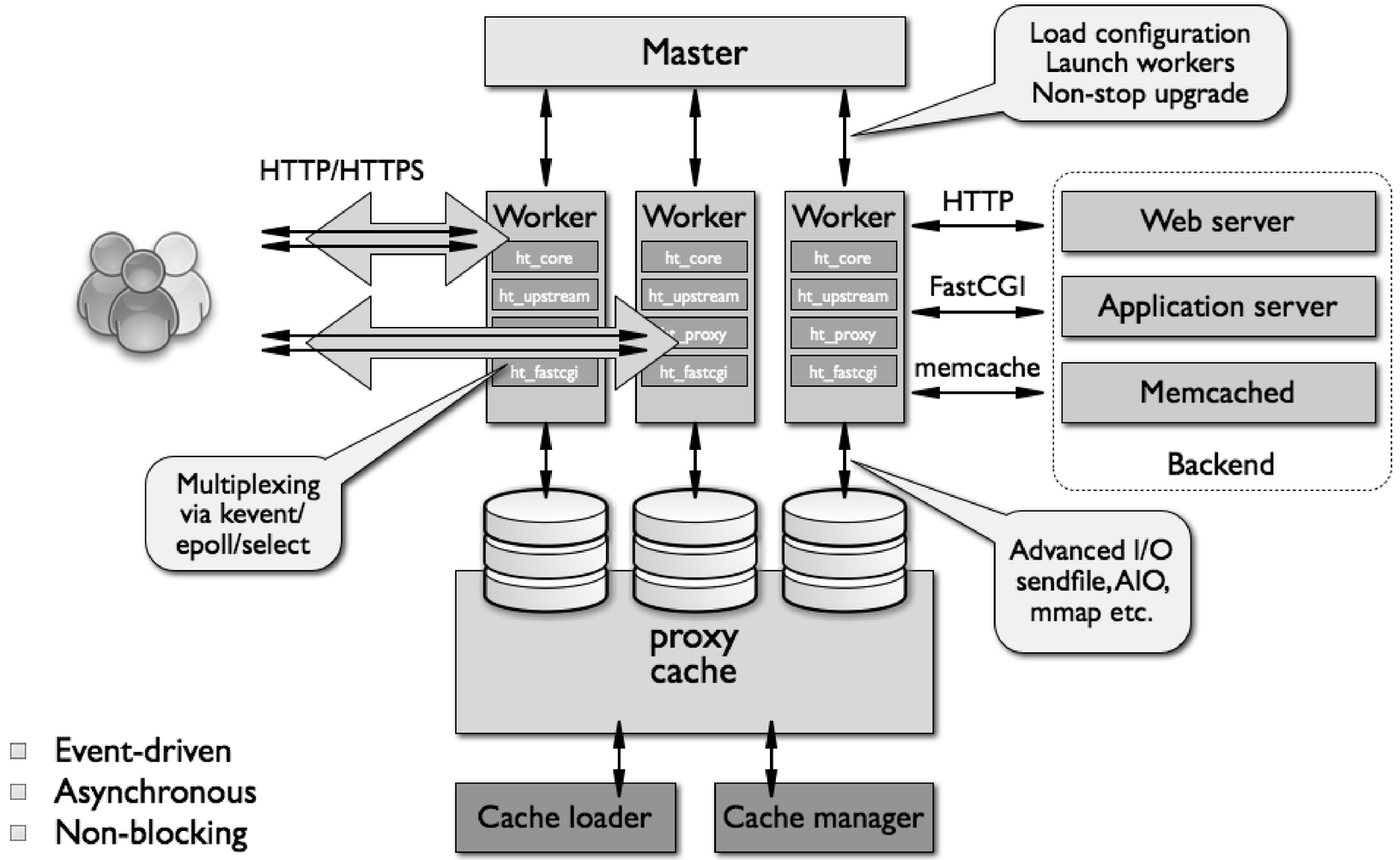
负责加载配置、接收命令、监控子进程。Master进程也是可以关闭的

Worker进程

负责处理网络请求。Worker进程的个数由配置文件决定，一般和CPU个数相关（有利于进程切换），配置几个就有几个Worker进程。需要说明的是，在nginx多进程中，每个worker都是平等的，因此每个进程处理外部请求的机会权重都是一致的。



Nginx的架构及工作流程



Master与Worker是如何分工协作的？与Netty的线程模型有什么异同？

多进程处理模型

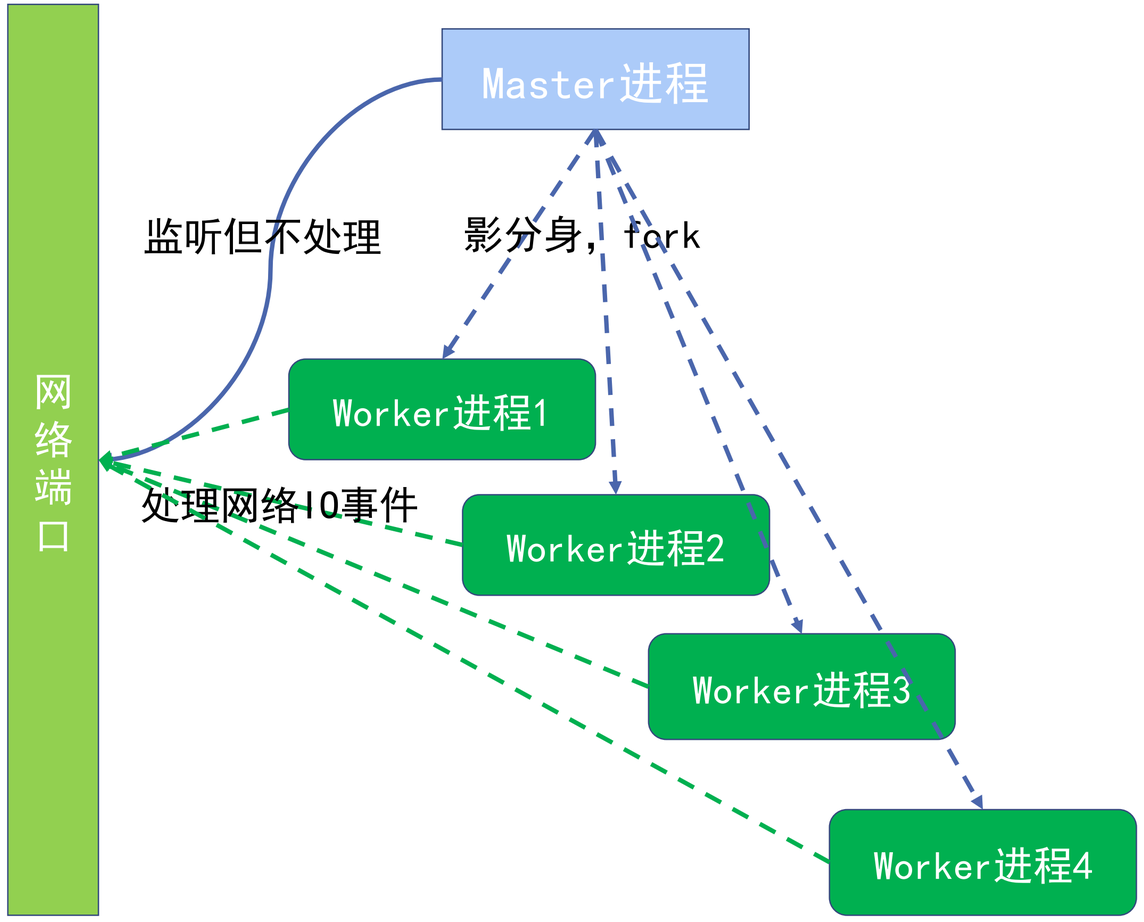
多进程模型的**处理方式**

首先，master进程一开始就会根据我们的配置，来建立需要listen的网络socket fd，然后fork出多个worker进程。

其次，根据进程的特性，新建立的worker进程，也会和master进程一样，具有相同的设置。因此，其也会去监听相同ip端口的套接字socket fd。

然后，多个worker进程监听同样设置的socket fd，当有一个请求进来，所有的worker都会感知。

最后，监听成功的worker进程，读取请求，解析处理，响应数据返回给客户端，断开连接，结束。因此，一个request请求，只需要worker进程就可以完成。



多进程处理模型优点

Nginx为何要采用多进程模式？

- ✓ 进程之间是独立的，当一个worker进程出现异常退出，其他worker进程不会受到影响；
- ✓ 独立进程也会避免一些不需要的锁操作，这样能提高处理效率，开发调试也更容易。

单纯的**多进程模型会导致连接并发数量降低，异步非阻塞IO模型**能解决这个问题，还能避免**多线程上下文切换导致**的性能损失。

多进程模型+异步非阻塞模型是一个非常好的选择。

跟NIO、Netty线程模型思想一致，只不过一个是线程级、一个是进程级。

多worker进程如何协作？

Worker进程遇到的问题

出现惊群效应

多个worker子进程监听相同端口的设计，这样多个子进程在accept建立新连接时会有争抢，这会带来著名的“惊群”问题，子进程数量越多问题越明显，这会造成系统性能下降。建立新连接时Nginx是如何避免出现“惊群”现象的呢？

Worker进程负载不均衡

多个子进程争抢处理一个新连接事件，最终只有一个worker子进程成功建立连接，随后，它会一直处理这个连接直到连接关闭。

如果有的子进程很“勤奋”，它们抢着建立并处理了大部分连接，而有的子进程则“运气不好”，只处理了少量连接。

子进程间负载不均衡，必然影响整个服务的性能。怎么办？

问题的解决之道

Nginx的Post队列

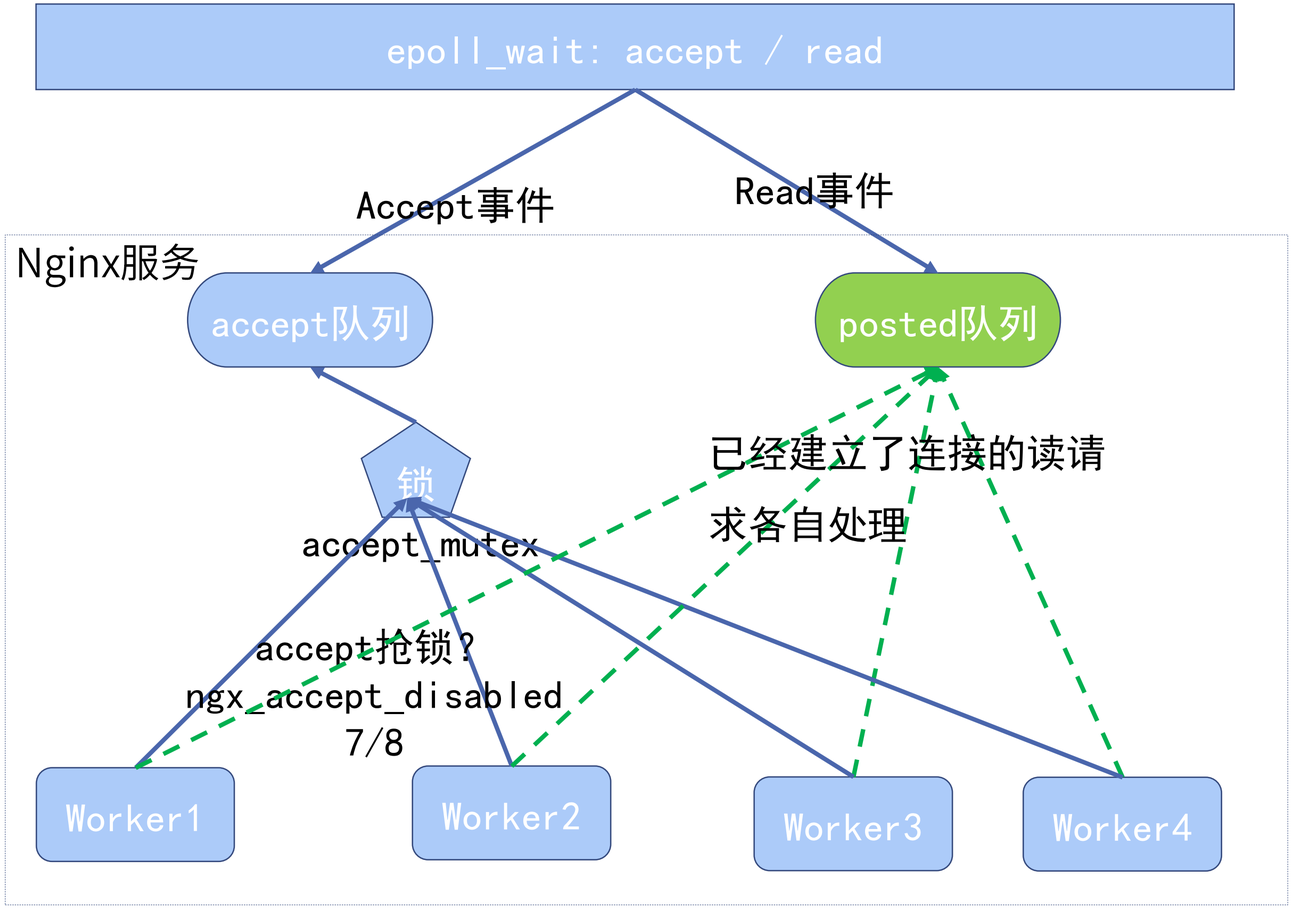
将epoll产生的事件分开，新连接事件的ngx_posted_accept_events队列优先执行，普通事件的ngx_posted_events队列最后执行

accept_mutex

Nginx采用了一个是否打开accept_mutex选项的值，控制worker进程是否需要去竞争锁，打开accept_mutex锁，能很好解决accept惊群问题。

ngx_accept_disabled

worker进程进行负载均衡阈的判断值，处理的连接总数没有达到7/8，不会抢accept锁，达到时，则需要抢锁，它解决了负载均衡的问题。



目录

Worker与CPU的绑定

worker与CPU的绑定

Worker进程跟CPU个数有关，但是如果多个Worker进程在同一个CPU上面争夺资源也非常消耗性能。Nginx中有一个worker_cpu_affinity 配置，能够将worker绑定到对应的CPU上面。

```
worker_processes    4;
worker_cpu_affinity 0001 0010 0100 1000;
```

单个Worker进程处理的并发连接数配置

```
worker_connections 1024;
```

目录

Linux服务器参数性能优化

linux服务器参数调整

当TCP TIME_WAIT套接字数量经常达到两、三万，服务器很容易被拖死，需要减少Nginx服务器的TIME_WAIT套接字数量。

具体参考《Nginx的Linux服务器参数性能优化.md》文件

目录

常用nginx性能优化

Nginx常规调整

常见指令	作用	默认值
worker_processes	定义工作进程的数量，最佳值取决于许多因素，包括（但不限于）CPU核心数，存储数据的硬盘驱动器数和负载模式。设置为可用CPU核心数将是一个良好的开端（值“ auto”将尝试自动检测它）。	
worker_rlimit_core	工作进程的核心文件的最大大小限制，在不重新启动主进程的情况下增加限制。	
worker_rlimit_nofile	工作进程允许打开的最大文件数，在不重新启动主进程的情况下增加限制。	
worker_connections	工作进程可以打开的最大并发连接数。这个数字包括所有连接（例如与代理服务器的连接等），而不仅仅是与客户端的连接。另一个考虑因素是实际的并发连接数不能超过最大打开文件数的当前限制，可以通过worker_rlimit_nofile更改	
worker_cpu_affinity	将worker进程绑定到CPU组，每个CPU集由允许的CPU的位掩码表示。应该为每个worker进程定义一个单独的集合。默认情况下，工作进程不绑定到任何特定的CPU。	
ssl_engine	硬件SSL加速器的名称。可以通过openssl engine -t来检查是否有硬件加密设备。	-
worker_priority	定义工作进程的调度优先级，就像nice命令一样：负数 number 表示更高的优先级，允许范围通常在-20到20之间变化。linux系统安装进程优先级来调度进程，优先级越高时间片越长。	

总结

Nginx工作原理

- Nginx的工作模式
- Nginx总体架构及工作流程
- 多进程处理模式
- Worker进程间的协作
- accept_mutex开关

Nginx与CPU绑定

- worker_cpu_affinity

Linux服务器参数调整

Nginx性能优化常规调整

谢谢观看