# Nginx

## ngx\_cycle\_t结构体

void \*\*\*\*conf\_ctx为什么是四个指针？

## Nginx事件驱动

## Nginx定时事件

ngx每个子进程都会独立管理时间。时间更新是在ngx\_epoll\_process\_events中。当flag中设置了NGX\_UPDATE\_TIME或ngx\_timer\_alarm == 1时，调用ngx\_time\_update更新时间。

ngx配置项time\_resolution是设置时间最长更新间隔。通过系统调用setitimer，每隔time\_resolustion毫秒调用一次ngx\_timer\_singal\_handler，将ngx\_timer\_alarm设置为1。

设置了time\_resolution之后，epoll\_wait的timer参数将被设置为-1。这样如果没有时间发生，epoll\_wait会立即返回，不会一直等待。这样可以避免ngx\_epoll\_process\_events长时间得不到执行。但是如果某个handler模块回调执行时间过长，时间精度还是会受到影响。

保存定时事件的数据结构是红黑树。超时时间最短的时间在树的最左边。每次只需要将最左节点与当前时间比较，就可以知道是否有超时事件。调用ngx\_expire\_timer会执行所有已超时的事件。

## 三、“惊群”问题

所谓惊群问题，就是多个子进程都监听了某个端口。在没有请求接入的时候，子进程都在等待。当有一个请求接入的时候，所有的子进程都会被唤醒，但最终只有一个子进程能accept成功，其他子进程又重新进入等待。而系统唤醒子进程的代价是很大的。

ngx通过配置accept\_mutex锁来解决这个问题，默认是开启。通过ngx\_accept\_disabled变量控制子进程是否竞争accept\_mutex锁。

当前空闲连接数小于总连接的1/8的时候，ngx\_accpet\_disabled大于0，当前子进程不竞争accept\_mutex锁。accept\_mutex是一个共享内存锁，加锁之后其值为进程号。获得accept\_mutex锁的进程才能处理accept事件和已有的连接事件。而没有获得锁的进程只能处理已有的连接事件。

accept\_mutex锁何时释放？accept获得新连接的进程并不会马上处理新连接上的事件，而是将新连接加入ngx\_posted\_accept\_event队列中。已有连接的事件则加入ngx\_posted\_event队列中。然后先处理ngx\_posted\_accept\_event队列中的事件，处理完成之后立刻释放accept\_mutex锁，再去处理ngx\_posted\_event事件。

ngx\_process\_events\_and\_timers是ngx处理所有事件（post事件和定时事件）。