08 如何选择事件模块？

更新时间：2019-12-17 09:16:01

> 虚心使人进步，骄傲使人落后。——毛泽东

回顾

我们在上一篇文章中详细的介绍了Linux中的事件模型，并且分析了每种模型的工作原理。

那么我们在工作当中要如何选择事件机制呢？本文会为你解开这个迷惑。

事件机制

为了最大限度的提高服务器的性能，我们显然不能使用阻塞I/O，因为这种效率实在是太太太低了……(大家用脚指头都可以想通~)

那么是否可以使用非阻塞I/O呢？这个可不会阻塞系统啊，

```
while (true) {
    int n1 = read(fd1);
    if (0 == n1 && EWOULDBLOCK == err){
        // fd1不可读
    }

    int n2 = read(fd2);
    if (0 == n2 && EWOULDBLOCK == err){
        // fd2不可读
    }
}
……
```

但是你想一下如果有几十万甚至上百万的连接同时和服务器进行连接的时候会是什么情形呢？

我们在程序里面要对每一个连接进行读写判断，这样的效率真的是无法让人接受啊

I/O多路复用

长话短说，Nginx选择使用I/O多路复用机制来提升性能。

如果我问大家为什么Nginx的效率这么高？可能大家都会说出来如下的两个个：

- Nginx使用了epoll(只针对Linux而言)
- Nginx使用了事件机制

但是大家知道为什么吗？哈哈，容我一一道来……

epoll是Linux系统上性能最高的基于事件机制的I/O多路复用模式。epoll的使用流程如下：

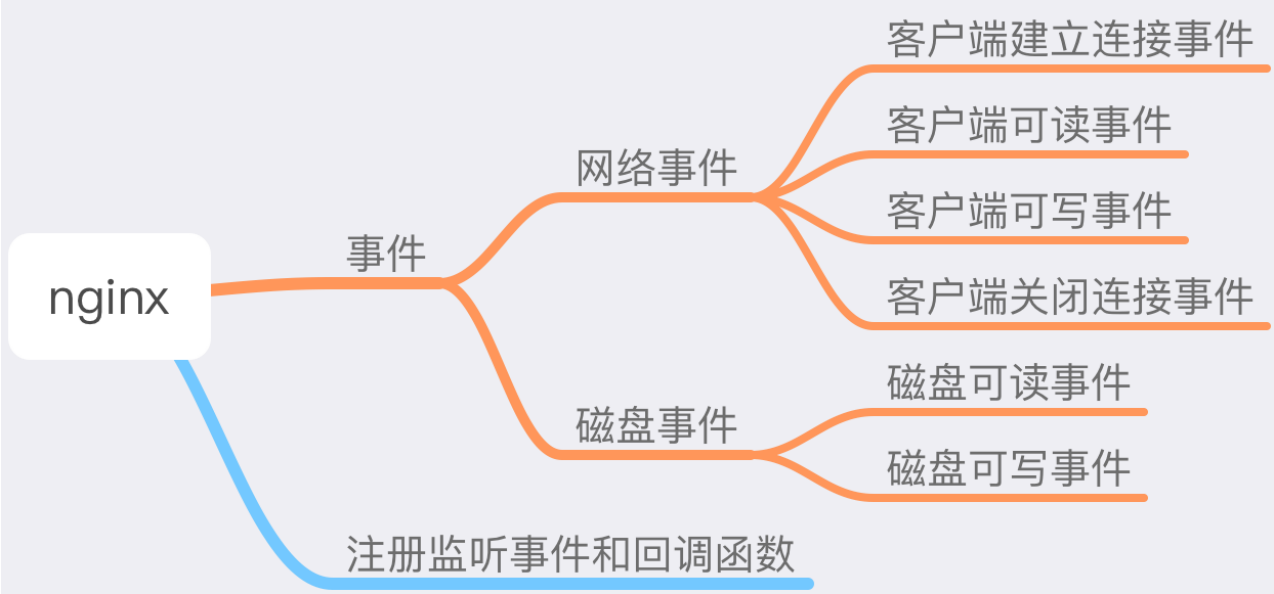
```
// 创建epoll句柄
int epoll_fd = epoll_create();
// 监听fd1的读事件
int rt1 = epoll_ctl(epoll_fd, fd1, READ);
// 监听fd2的写事件
int rt2 = epoll_ctl(epoll_fd, fd2, WRITE);
……
```

Linux在操作系统的内核中实现了epoll机制，这是一种类似订阅-发布设计模式。我们把自己关心的事件通过epoll函数注册到内核中，当指定的事件发生的时候内核会调用我们注册的回调函数进行处理。

那么nginx是如何使用epoll的呢？

nginx把事件分为了网络事件和磁盘事件，而网络事件包含了客户端建立连接，可读，可写，关闭连接事件。磁盘事件包含了平时遇到的可读事件和可写事件。

然后Nginx把这些事件都注册到epoll中，当相应的事件发生的时候，nginx就会自动调用我们注册的回调函数进行事件的处理。



为什么epoll高效

1、 epoll在Linux内核中开辟了一个cache缓冲区，使用红黑树结构，并且使用共享内核和应用层进行数据传递，非常的高效。

2、 eopll只对活跃的连接感兴趣，即使同时有几百万的客户端和Nginx相连，同时活跃的连接数量也不会很高（因为大多数的连接是处于不活跃状态的，这是客户端的一种特点），所以epoll可以把所有事件处于ready状态的连接全部返回给用户，避免了用户自己轮训所有连接的过程。

总结

本文简单的分析了epoll高效的原因，以及为什么Nginx选择epoll作为事件处理机制。

这部分内容大家要了解一下，对于理解Nginx工作机制非常有用。

}

07 打扮你的nginx：配置文件初识

09 必要的全局配置选项