**Nginx优化**

    1、IPv4内核参数优化

    2、处理器指令配置

    3、网络连接指令配置

    4、事件驱动相关指令配置

**针对IPv4的内核参数优化**

    我们将内核参数的值追加到linux系统的/etc/sysctl.conf文件中，然后使用命令使修改生效：

    #/sbin/sysctl -p

    1、net.core.netdev\_max\_backlog参数

    参数net.core.netdev\_max\_backlog，表示当每个网络接口接收数据包的速率比内核处理这些包的速率快时，允许发送队列的数据包的最大数目，一般默认值为128（不同的linux系统值也不同）。Nginx服务器中定义的net.core.netdev\_max\_backlog默认为511.我可以将它调整一下：

    net.core.netdev\_max\_backlog = 262144

    2、net.core.somaxconn参数

    该参数用于调节系统同时发起的TCP连接数，一般默认值为128.在客户端存在高并发请求的情况下，该默认值较小，可能导致链接超时或者重传问题，我们可以根据实际情况需要结合并发请求数来调节此值

    net.core.somaxconn = 262144

    3、net.ipv4.tcp\_max\_orphans参数

    该参数用于设定系统中最多允许存在多少TCP套接字不会被关联到任何一个用户文件句柄上。如果超过这个数字，没有与用户文件句柄关联的TCP套件字就会立即被复位，同时给出警告信息。这个限制只是为了防止简单的DDOS攻击，一般在系统内存比较充足的情况下，可以增大这个参数的赋值：

    net.ipv4.tcp\_max\_orphans = 262144

    4、net.ipv4.tcp\_max\_syn\_backlog 参数

    该参数用于记录尚未收到客户端确认信息的连接请求的最大值。对于拥有128MB内存的系统而言，此参数的默认值是1024，对小内存的系统则是128，一般在系统内存比较充足的情况下，可以增大这个参数的赋值：

    net.ipv4.tcp\_max\_syn\_backlog = 262144

    5、net.ipv4.tcp\_timestamps 参数

    该参数用于设置时间戳，这可以避免序列号的卷绕。在一个1Gb/s的链路上，遇到以前用过的序列号的概率很大。当此值赋值为0时，禁用对于TCP时间戳的支持。在默认情况下，TCP协议会让内核接收这种“异常”的数据包。针对Nginx服务器来说，建议将其关闭：

    net.ipv4.tcp\_timestamps = 0

    6、net.ipv4.tcp\_synack\_retries 参数

    该参数用于设置内核放弃TCP连接之前向客户端发送SYN+ACK包的数量。为了建立对端的连接服务，服务器和客户端需要进行三次握手，第二次握手期间，内核需要发送SYN并附带一个回应前一个SYN的ACK，这个参数主要影响这个过程，一般赋值为1.即内核放弃连接之前发送一次SYN+ACK包，可以设置其为：

    net.ipv4.tcp\_synack\_retries = 1

    7、net.ipv4.tcp\_syn\_retries 参数

    该参数的作用与上一个参数类似，设置内核放弃建立连接之前发送SYN包的数量，它的赋值和上一个参数一样即可：

    net.ipv4.tcp\_syn\_retries = 1

**针对CPU的Ngnix配置优化**

    1、worker\_processes指令

    worker\_processes指令用来设置Nginx服务的进程数。官方文件建议此值指令一般设置为1即可，赋值太多会影响系统的IO效率，降低Nginx服务器的性能。为了让多核CPU能够更好地并行处理任务，我们可以将worker\_processes指令的赋值适当地增大一些，最好是赋值为机器CPU的倍数。当然，这个值并不是越大越好，Nginx进程太多可能增加主进程调度负担，也可能影响系统的IO效率。针对双核CPU，建议设置为2或4.四核CPU设置为：

    worker\_processes 4;

    设置好worker\_processes指令之后，就很有必要设置worker\_cpu\_affinity指令

    2、worker\_cpu\_affinity指令

    worker\_cpu\_affinity指令用来为每个进程分配CPU的工作内核。这个指令的设置方法有些麻烦。worker\_cpu\_affinity指令的值是由几组二进制值表示的。其中，每一组代表一个进程，每组中的每一位表示该进程使用CPU的情况，1代表使用，0代表不使用。注意，二进制位排列顺序和CPU的顺序是相反的。建议将不同的进程平均分配到不同的CPU运行内核上

    例如Nginx服务的进程数为4，CPU是四核，因此会有四组值，并且每组有四位，所以，此指令的设置为：

    worker\_cpu\_affinity 0001 0100 1000 0010;

    四组二进制数值分别对应4个进程，第一个进程对应0001，表示使用第一个CPU内核;第二个进程对应0010.表示使用第二个CPU内核，以此类推

    如果笔者将worker\_processes指令的值赋值为8，即赋值为CPU内核个数的两倍，则worker\_cpu\_affinity指令的设置可以是：

    worker\_cpu\_affinity 0001 0010 0100 1000 0001 0010 0100 1000；

    如果一台机器的CPU是八核CPU，并且worker\_processes指令的值赋值为8，那么worker\_cpu\_affinity指令的设置可以是：

    worker\_cpu\_affinity 00000001 00000010 00000100 00001000 00010000 00100000 01000000 10000000

**与网络连接相关的配置**

    1、keepalive\_timeout指令

    该指令用于设置Nginx服务器与客户端保持连接的超时时间

    keepalive\_timeout 60 50；

    该设置表示Nginx服务器与客户端连接保持活动的时间是60S，60S后服务器与客户端断开连接；使用Keep-Alive消息头保持与客户端某些浏览器的连接时间为50S，50S后浏览器主动与服务器断开连接

    2、send\_timeout指令

    该指令用于设置Nginx服务器响应客户端的超时时间，这个超时时间仅针对两个客户端和服务器之间建立连接后，某次活动之间的时间。如果这个时间后客户端没有任何活动，Nginx服务器将会关闭连接。此指令的设置需要考虑服务器访问数量和网络状况封方面

    send\_timeout 10s;

    该设置表示Nginx服务器和客户端连接建立后，某次会话中服务器等待客户端响应超过10S，就自动关闭连接。

    3、client\_header\_buffer\_size指令

    该指令用于设置Nginx服务器允许的客户端请求头部的缓存区大小，默认为1KB。此指令的赋值可以根据系统分页大小来设置。分页大小可以用以下命令获取：

    #getconf PAGESIZE

    Nginx服务器的400错误，有很大一部分情况是客户端的请求头部过大造成的。请求头部过大，跳槽是客户端Cookie中写入了较大的值引起的。于是适当增大此指令的辅助，允许Nginx服务器接收较大的请求头部，可以改善服务器对客户端的支持能力，一般将此值赋值为4KB大小，即：

    client\_header\_buffer\_size 4K；

    4、mutil\_accept

    该指令用于配置Nginx服务器是否尽可能多地接收客户端的网络连接请求，默认值为off

**与事件驱动模型相关的配置**

    1、use指令

    use指令用于指定Nginx服务器使用的事件驱动模型

    2、worker\_connections指令

    该指令用于设置Nginx服务器的每个工作进程允许同时连接客户端的最大数量，语法为：

    worker\_connections number

    其中，number为设置的最大数量。结合worker\_processes指令，我们可以计算出Nginx服务器允许同时连接的客户端最大数量Client=worker\_processes\*worker\_connections/2

    在使用Nginx服务器的过程中，此指令一般设置为65535:

    worker\_connections 65535;

    此指令的赋值与Linux操作系统中进程可以打开的文件句柄数量有关系。linux默认允许打开的文件数限制为1024个。

    3、worker\_rlimit\_sigpending指令

    该值指令用于设置linux2.6.6-mm2版本之后linux平台的事件信号队列长度上线。其语法结构为：

    worker\_rlimit\_sigpending limit

    其中，limit为Linux平台事件信号队列的长度上限值

    该指令朱玥影响事件驱动模型中rtsig模型可以保存的最大信号数。Nginx服务器的每一个工作进程有自己的事件信号队列用于暂存客户端请求发生信号，如果超过长度上限，Nginx服务器自动转用poll模型处理未处理的客户端请求。为了保证Nginx服务器对客户端请求的高效处理，请大家根据实际的客户端并发请求数量和服务器运行环境的处理能力设定该值。设置示例为：

    worker\_rlimit\_sigpending 1024;

    4、devpoll\_changes和devpoll\_events指令

    这个两个指令用于设置在/dev/poll事件驱动模式下Nginx服务器可以与内核之间传递事件的数量，前者设置传递给内核的事件数量，后者设置从内核获取的事件数量，语法结构为：

    devpoll\_changes number

    devpoll\_events number

    其中，number为要设置的数量，默认值均为32

    5、kqueue\_changes和kqueue\_events指令

    这个两个指令用于设置在kqueue事件驱动模式下Nginx服务器可以与内核之间传递事件的数量，前者设置传递给内核的事件数量，后者设置从内核获取的事件数量，其语法结构为：

    kqueue\_changes number

    kqueue\_events number

    其中，number为要设置的数量，默认值均为512

    使用kequeue\_changes方式，可以设置与内核之间传递事件的数量。

    6、epoll\_events指令

    该指令用于设置在epoll事件驱动模式下Nginx服务器可以与内核之间传递事件的数量，其语法结构为：

    epoll\_changes number

    其中，number为要设置的数量，默认值均为512

    注意：与其他数据驱动模型不同，在epoll事件驱动模式下Nginx服务器向内核传递事件的数量和从内核传递的事件数量是相等的，因此没有类似epoll\_changes这样的指令

    7、rtsig\_signo指令

    该指令用于设置rtsig模式使用的两个信号中的第一个，第二个信号是在第一个信号的变化上加1.语法为：

    rtsig\_signo signo

    默认的第一个信号设置为SIGRTMIN+10

    8、rtsig\_overflow\_\*指令

    该指令代表三个具体的指令，分别为rtsig\_overflow\_events指令、rtsig\_overflow\_test指令和rtsig\_overflow\_threshold指令。这些指令用来控制当rtsig模式中信号队列溢出时Nginx服务器处理方法。其语法结构为：

    rtsig\_overflow\_\* number

    其中，number是要设置的值

    rtsig\_overflow\_events指令指定队列溢出时使用poll库处理的事件数，默认值为15

    rtsig\_overflow\_test指令指定poll库处理完第几件事件后将清空rtsig模型使用的信号队列，默认值为32

    rtsig\_overflow\_threshold指令指定rtsig模式使用的信号队列中的事件超过多少时就需要清空队列了。该指令只对linux2.4.X及以下版本有效。在这些版本中包含两个参数：分别是proc/sys/kernel/rtsig-nr和/proc/sys/kernel/rtsig-max/rtsig\_overflow\_threshold，后者就是该指令设定的值。当Nginx服务器检测到前者大于后者时，将清空队列。该指令默认值为10，代表rtsig-max的1/10

**Nginx调优**

Worker进程数

同一时间内每个Worker可以维护的连接数

CPU绑定

保持会话

**nginx日志文件删除后空间不能释放**

在Linux或者Unix系统中，通过rm或者文件管理器删除文件将会从文件系统的目录结构上解除链接(unlink).然而如果文件是被打开的（有一个进程正在使用），那么进程将仍然可以读取该文件，磁盘空间也一直被占用。而我删除的是nginx的log文件删除的时候文件应该正在被使用

**nginx 404和50x页面的配置**

一、要在nginx.conf  文件里开启错误访问相关的配置

fastcgi\_intercept\_errors on;

将代码添加到 http  大块里

二、在自己的虚拟站点配置里定义错误页面，

error\_page 404 = http://www.aibaitou.com/error/404.html;

error\_page 500 502 503 504=http://www.aibaitou.com/error/50x.html;

include /usr/local/nginx/conf/rewrite/aibaitou.conf;

将代码添加到 server 大块里

三、修改重定向url的配置文件 aibaitou.conf ，添加错误页面的转发请求解析

rewrite ^/error/(.\*).html/?$ /index.php?app=errpage&id=$1 last;

四、nginx测试下

/usr/local/nginx/sbin/nginx -t

**502故障分析**

502故障出现的原因一般都不是负载均衡的问题，可能是后端程序进程出错、程序执行超时。后端服务器IO故障、CPU故障、网络故障、数据库故障，强依赖的第三方调用故障

**Select epoll 区别**

　 select，poll，epoll都是IO多路复用的机制。I/O多路复用就通过一种机制，可以监视多个描述符，一旦某个描述符就绪（一般是读就绪或者写就绪），能够通知程序进行相应的读写操作。但select，poll，epoll本质上都是同步I/O，因为他们都需要在读写事件就绪后自己负责进行读写，也就是说这个读写过程是阻塞的，而异步I/O则无需自己负责进行读写，异步I/O的实现会负责把数据从内核拷贝到用户空间

epoll跟select都能提供多路I/O复用的解决方案。在现在的Linux内核里有都能够支持，其中epoll是Linux所特有，而select则应该是POSIX所规定，一般操作系统均有实现