**19-（实验）单机百万TCP连接**

**实验方法**

* 在一台机器上既当服务器端，也当客户端
* 用20个端口分别启动服务端
* 同样启动20个客户端进程，每个客户端向其中一个服务端建立5万条TCP连接

|  |
| --- |
| 如果遇到连接特别慢的问题，可以尝试使用40对儿进程，每对儿进程建立25000条连接。修改方法参见本文后面的问题汇总。 |

**准备工作**

**修改内核文件句柄限制**

查看当前配置

|  |
| --- |
| C # sysctl -a | grep nr\_open # sysctl -a | grep file-max |

修改配置（不推荐使用 echo xxx 的方式，只能临时生效）

* 每个客户端进程、服务端进程需要保持50000个TCP连接，nr\_open要比 hard nofile 大一点，所以nr\_open设置为60000
* 整个系统中由于同时保持客户端和服务端，100万条TCP连接需要200万个文件句柄，所以file-max设置为2200000

|  |
| --- |
| C # vi /etc/sysctl.conf fs.nr\_open=60000 // fs.file-max=2200000 //多留点buffer |

使之立即生效

|  |
| --- |
| C # sysctl -p |

**修改PAM文件句柄限制**

确保pam\_limits.so是被加载的

|  |
| --- |
| C #vi /etc/pam.d/login session required pam\_limits.so |

查看当前用户配置

|  |
| --- |
| C # ulimit -n |

修改配置（不推荐使用 ulimit -n 命令修改，只能临时生效）

|  |
| --- |
| C # vi /etc/security/limits.conf \* soft nofile 55000 \* hard nofile 55000 |

退出重新登录即可生效，再次查看

|  |
| --- |
| C # ulimit -n |

**再次提醒**：hard nofile 一定要比 fs.nr\_open 要小，否则可能导致用户无法登陆

**修改内核端口号范围**

查看当前配置

|  |
| --- |
| C # sysctl -a | grep ip\_local\_port\_range |

修改配置（不推荐使用 echo xxx 的方式，只能临时生效）

|  |
| --- |
| C #vi /etc/sysctl.conf net.ipv4.ip\_local\_port\_range = 1025 65500 |

使之立即生效

|  |
| --- |
| C # sysctl -p # sysctl -a | grep ip\_local\_port\_range |

**准备测试源码**

测试百万连接我用到的方案有两种，今天用一篇文章都给大家分享出来。

* 第一种是服务器端只开启一个进程，然后使用很多个客户端 ip 来连接
* 第二种是服务器开启多个进程，这样客户端就可以只使用一个 ip 即可

为了能让大部分同学都能用最低的时间成本达成百万连接结果，飞哥写了 c、java、php 三种版本的源码。两个方案对应的代码地址如下：

* 方案一：https://github.com/yanfeizhang/coder-kung-fu/tree/main/tests/network/test02
* 方案二：https://github.com/yanfeizhang/coder-kung-fu/tree/main/tests/network/test03

本文中，由于设置IP会影响内网其它服务器，所以采用方案二。

**下载源码**

|  |
| --- |
| C # git clone https://github.com/yanfeizhang/coder-kung-fu.git # cd tests/network/test03 |

然后选择语言，如果选c的话就进入c目录即可

|  |
| --- |
| C # cd c |

不管使用任何语言，下面的操作方法都是一样的。因为本实验代码把操作都用Makefile封装起来了。

**修改IP**

修改 tool.sh 中的服务器 ip。端口会自动从 tool.sh 中加载。然后开始连接

|  |
| --- |
| C # SERVERIP="192.168.1.100" SERVERIP="127.0.0.1" |

**清理一下缓存**

实验开始前记录SLAB对象消耗。如果有干扰对象，可以清理一下pagecache dentries, inodes。

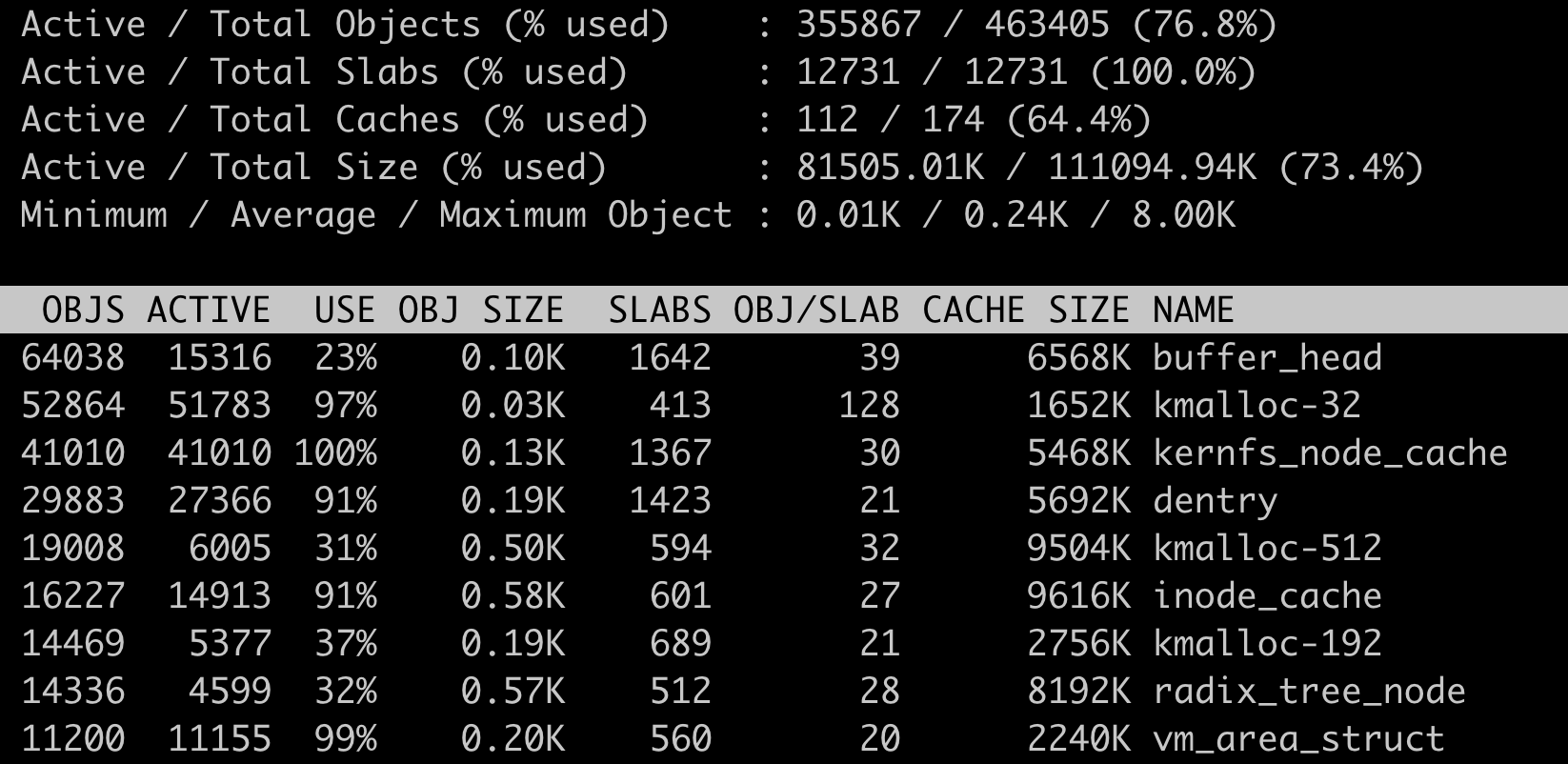
|  |
| --- |
| C # echo 3 > /proc/sys/vm/drop\_caches |

**观察内存开销**

实验开始前记录内存消耗

|  |
| --- |
| C # cat /proc/meminfo MemTotal: 16152104 kB MemFree: 14919328 kB MemAvailable: 14906940 kB Buffers: 10608 kB Cached: 368192 kB ... Slab: 117784 kB |

**观察SLAB对象消耗**



**开始动手测试**

**启动Server**

启动 server

|  |
| --- |
| C # make run-srv |

使用 netstat 命令确保 server 监听成功。

|  |
| --- |
| C netstat -nlt | grep 81 tcp 0 0 0.0.0.0:8100 0.0.0.0:\* LISTEN tcp 0 0 0.0.0.0:8101 0.0.0.0:\* LISTEN ...... tcp 0 0 0.0.0.0:8119 0.0.0.0:\* LISTEN |

**启动Client**

|  |
| --- |
| C # make run-cli |

**观察连接数**

连接数

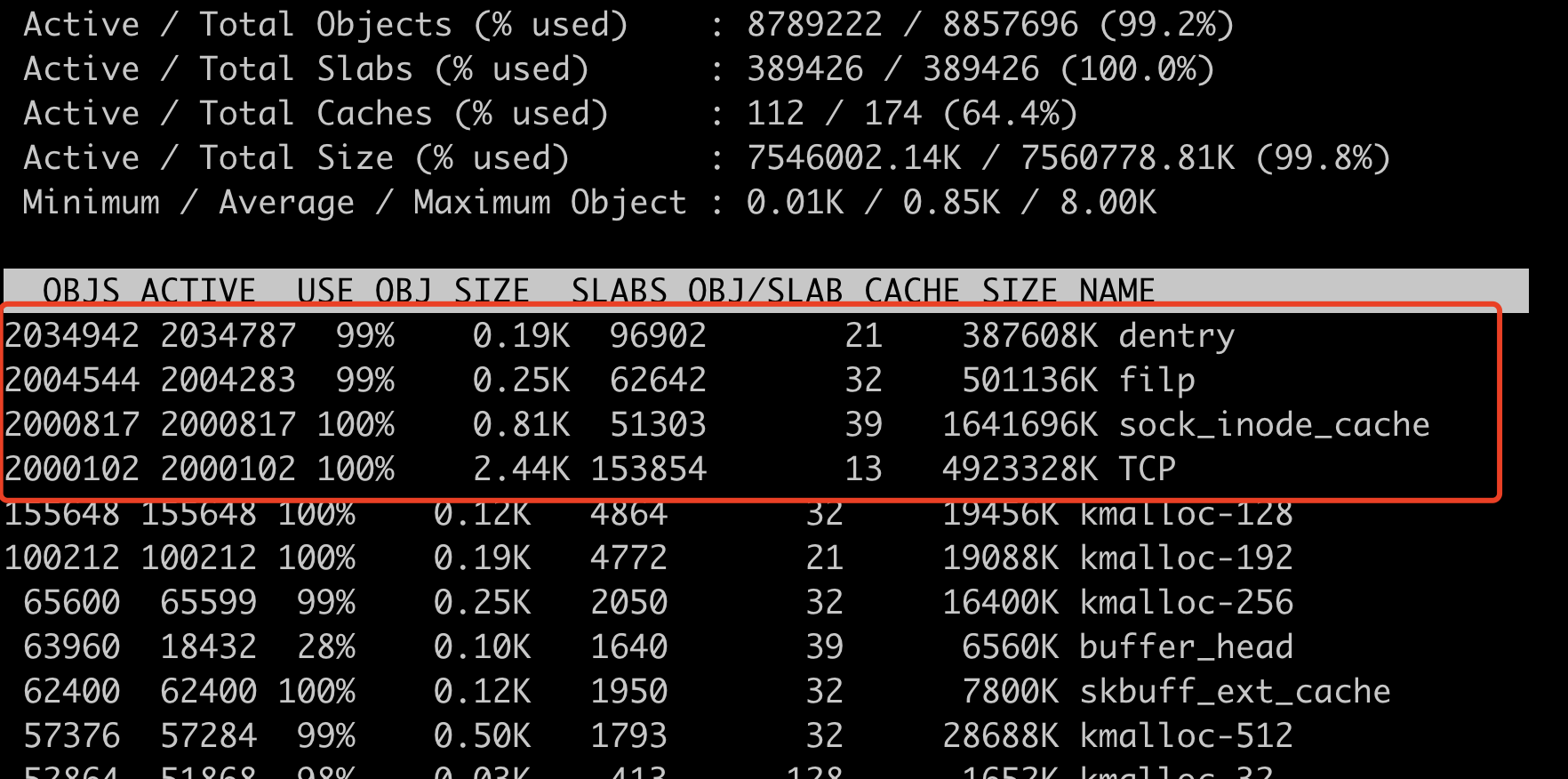
|  |
| --- |
| C # watch "ss -ant | grep ESTAB | wc -l" 2000039 |

**观察内存开销**

|  |
| --- |
| C # cat /proc/meminfo MemTotal: 16152104 kB MemFree: 7224188 kB MemAvailable: 9273624 kB Buffers: 23228 kB Cached: 477420 kB ... Slab: 7659376 kB |

**观察SLAB对象消耗**

|  |
| --- |
| C #slabtop |



**结束实验**

|  |
| --- |
| C # make stop-cli # make stop-srv |

**可能遇到的问题**

遇到问题不要气馁，我自己当年做这个实验的时候断断续续持续了半个月才做完。找到问题把问题解决掉，你的体会会更深。

**连接建立速度变慢**

**可能原因1：连接队列溢出了**

这种问题是因为服务端的连接队列设置的太小了。因为是127.0.0.1网络延迟极低，所以客户端的连接请求可能会把连接队列打满。一旦发生连接队列发生溢出丢包，那重试的超时计时器就是秒级别的延迟。解决办法是可以加大全连接队列，或者修改客户端代码，降低连接速度。

全连接队列长度是内核参数net.core.somaxconn和服务端listen系统调用时传入的backlog中较小的那个值，可能二者都需要调整。

**可能原因2：端口号范围设置小了**

如果端口号不足，会导致选择端口号时的耗时变长，端口号范围开的尽量要大一些。我是直接设置成[1025,65500]

|  |
| --- |
| C #vi /etc/sysctl.conf net.ipv4.ip\_local\_port\_range = 1025 65500 |

**可能原因3：在某些版本的内核下即使端口范围很充足，也会变慢**

解决办法，使用40对儿进程来连接，这样每个client只需要25000条连接就可以了。

修改client.c中的连接数，以前是50000，把它改成25000.

|  |
| --- |
| C #define MAX\_CONNECTION\_NUM 25000 |

修改tool.sh中的端口号，开启上40个server进程。

|  |
| --- |
| C SERVERPORTS=(  "8100"  "8101"  "8102"  "8103"  "8104"  "8105"  "8106"  "8107"  "8108"  "8109"  "8110"  "8111"  "8112"  "8113"  "8114"  "8115"  "8116"  "8117"  "8118"  "8119"  "8120"  "8121"  "8122"  "8123"  "8124"  "8125"  "8126"  "8127"  "8128"  "8129"  "8130"  "8131"  "8132"  "8133"  "8134"  "8135"  "8136"  "8137"  "8138"  "8139" ) |

**报错socket创建失败**

这种问题一般是因为没有正确设置文件句柄相关参数，导致创建socket时不能成功。按照教程正确修改内核参数即可。

**shell脚本报错Syntax error: "(" unexpected**

这是因为某些Linux下的sh使用的是dash的原因

|  |
| --- |
| C #ll /bin/sh lrwxrwxrwx 1 root root 4 Jul 11 2022 /bin/sh -> dash |

把Makefile文件中的 sh 都替换成 bash 命令即可。注意是改成 bash，不是dash。