2019-6-15 simplekvsvr 测试报告

小書匠

目录

一，无网络环境下

1.1 测试环境

序号	问题	解决方案
1	cpu	Intel Core i5-7200U 双核四线程
2	内存	8GB DDR4 2133MHz
3	存储	256GB PCIe * 4 NVMe SSD
4	操作系统	Deepin Linux 15.9

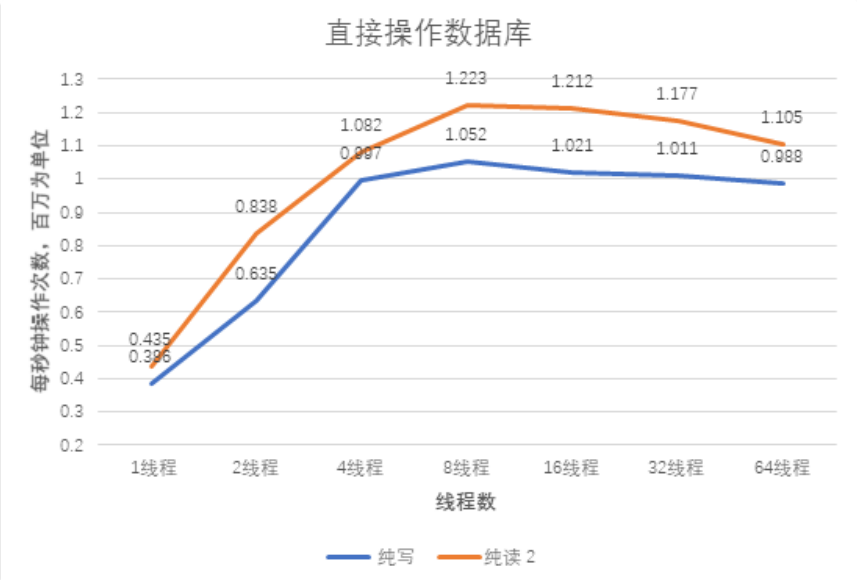
1.2测试场景

1.2.1 单纯插入与单纯查询

测试参数

如图所示，横轴为线程数，纵轴为每秒钟可以操作几百万次，写入数据key为10字节，value 为 10字节，请求分布为随机分布。

测试结果



读写压测表

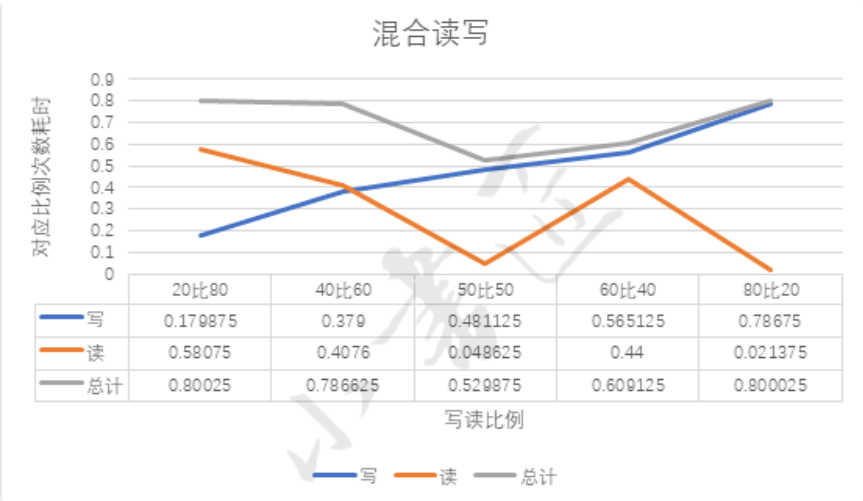
结果分析

程序在八线程读写时，整体性能表现最好，继续增加线程的数量，表现趋于平缓，有下降之势，这也符合预期，即线程数最好与核数保持一致。

1.2.2 读写混合

由上边的情况可以得知，本测试机线程数为8左右的情况下，程序性能表现最好，所以在本部分测试，线程数设置为8.每一个线程测试总量为100万次，即读写合计为100万次，即可得到一个曲线，这一步测试目的主要是检测其缓存能力，所以先写后读。
分为以下五种情况进行测试 数据分布均为随机

测试结果



	20比80	40比60	50比50	60比40	80比20
写	0.179875	0.379	0.481125	0.565125	0.78675
读	0.58075	0.4076	0.048625	0.44	0.021375
总计	0.80025	0.786625	0.529875	0.609125	0.800025

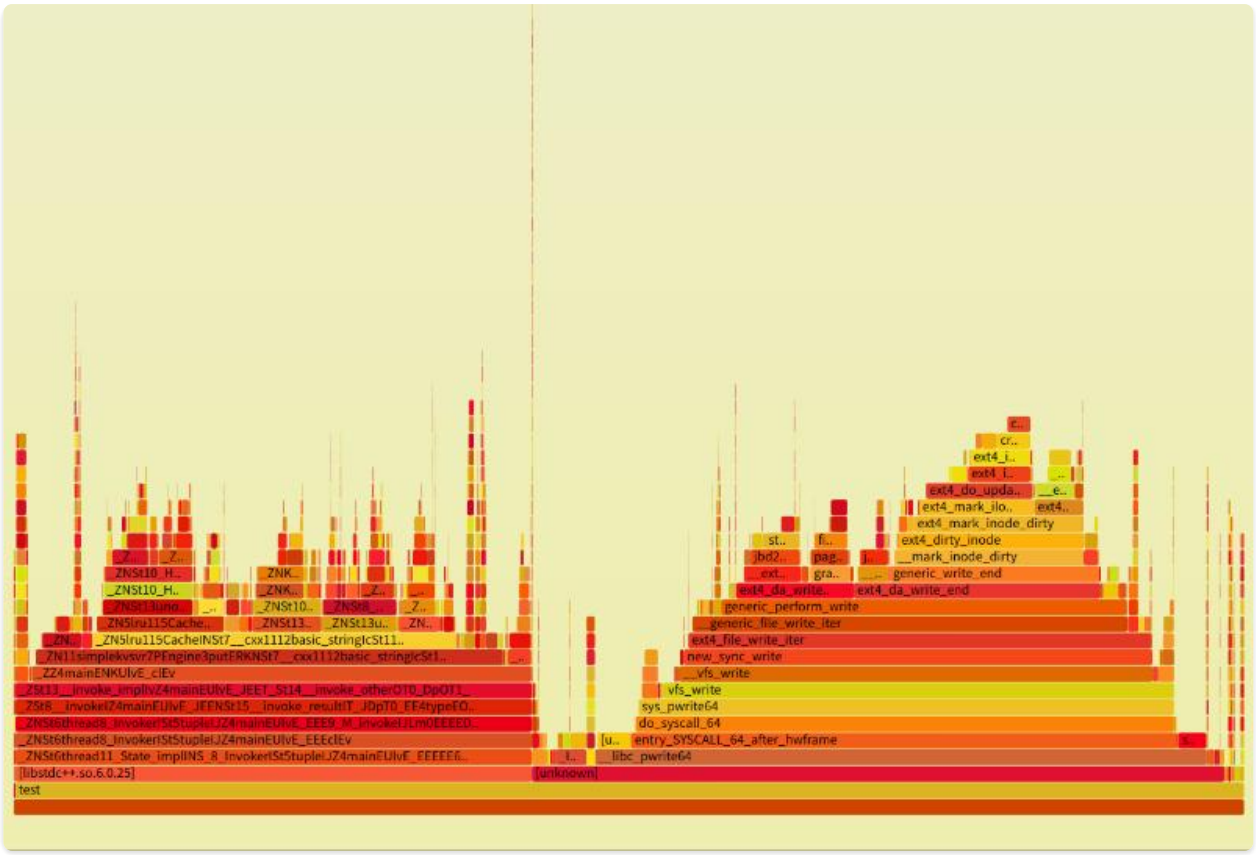
enter description here

结果分析

可以发现总耗时是呈U型的两边高中间低，比较符合预期，前部分写占了20%，数据读取大部分从磁盘获得，中间部分表现优异，缓存在这里居功至伟，后期写多读少，主要耗时在写上。

火焰图分析

八线程分别读写100万次性能分析



无大平顶

火焰图无大坪顶，整体表现良好，暂未找到高耗时代码。

valgrind内存分析

```
==27600==    by 0x10B030: main::{lambda()#1}::operator()() const (test.cc:39)
==27600==    by 0x10B5CB: void std::__invoke_impl<void, main::{lambda()#1}>(std::__invoke_o
==27600==    by 0x10B3FB: std::__invoke_result<main::{lambda()#1}>::type std::__invoke<main
#1}&&)...) (invoke.h:95)
==27600==
==27600== LEAK SUMMARY:
==27600==    definitely lost: 0 bytes in 0 blocks
==27600==    indirectly lost: 0 bytes in 0 blocks
==27600==      possibly lost: 1,152,952 bytes in 22,285 blocks
==27600==    still reachable: 0 bytes in 0 blocks
==27600==         suppressed: 0 bytes in 0 blocks
==27600==
==27600== For lists of detected and suppressed errors, rerun with: -s
==27600== ERROR SUMMARY: 10 errors from 10 contexts (suppressed: 0 from 0)
```

内存分析图

没有直接与间接的内存泄漏，possibly lost有一些， 查询了一番，似乎是因为mmap的内存的原因，具体原因正在进一步定位中。

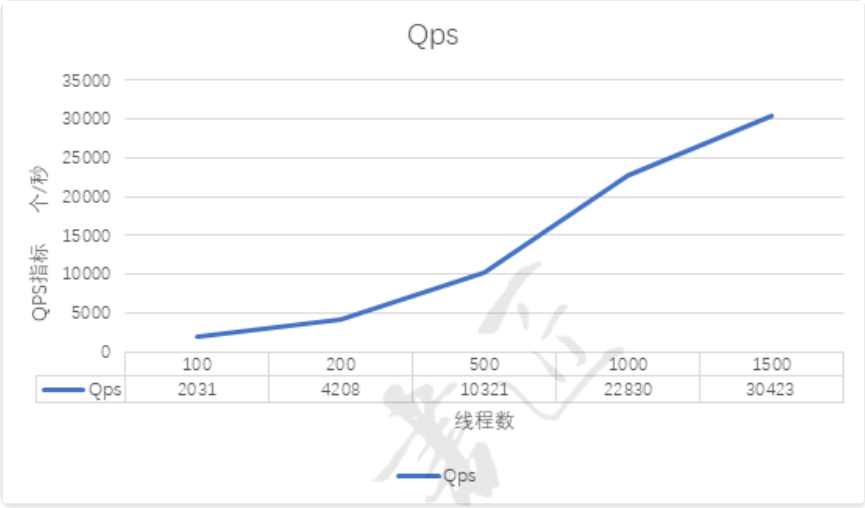
二， 网络环境下

2.1 测试场景

首先，根据我的程序预测，使用epoll多路复用，又加之线程池，所以线程池同一时刻内，同一fd最多只能有一个，所以这部分的压测参数，应该是客户端数量，由于网络部分比较耗时，此测试模块设定每个客户端发起1000次请求，多线程来模拟客户端，分别为100，200，500，1000 1500。 **计算方法**：取多次测试之平均值，由于是多线程来模拟，所以客户端为同时发起请求，总耗时为平均时间，所以QPS = 客户端数量*1000 / 平均耗时

2.2 写请求能力

测试结果



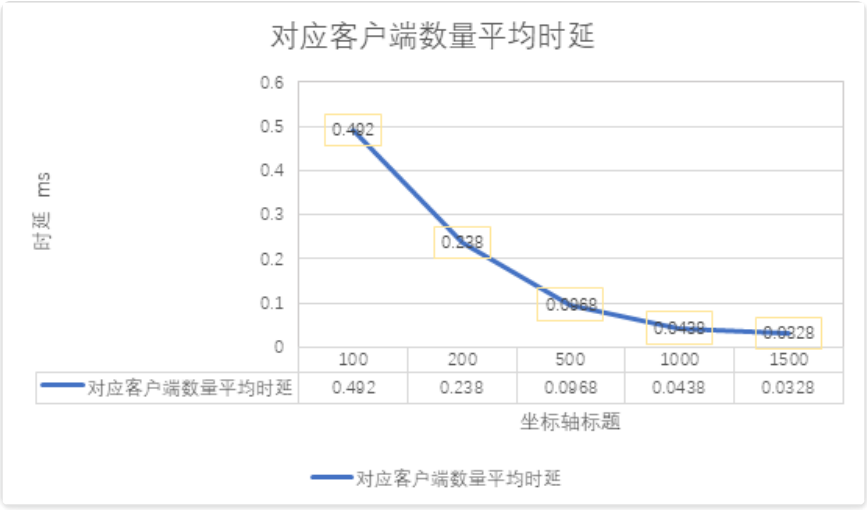
写QPS

结果分析

这部分测试，没有测到程序瓶颈，反而是因为我的机器原因，不能再开更多线程，且本地回环，也非常影响服务端的性能。相信随着硬件的提升，可以带来更好的表现。

时延测试

测试结果



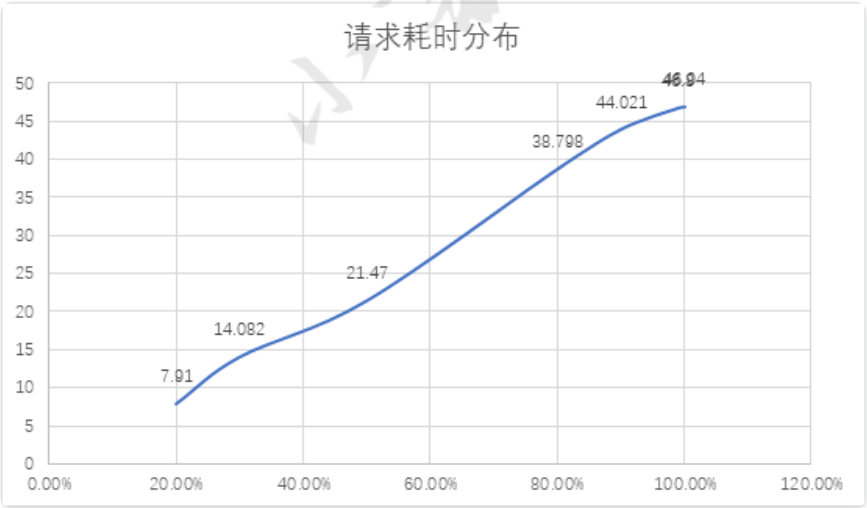
时延测试

结果分析

因为走本地回环，不走网卡，时延整体偏小，没有遇到瓶颈，这样的表现符合预期，

耗时分布

测试结果



enter description here

结果分析

整体耗时初期一直递增较快，末期放缓，符合预期。

程序整体优化分析

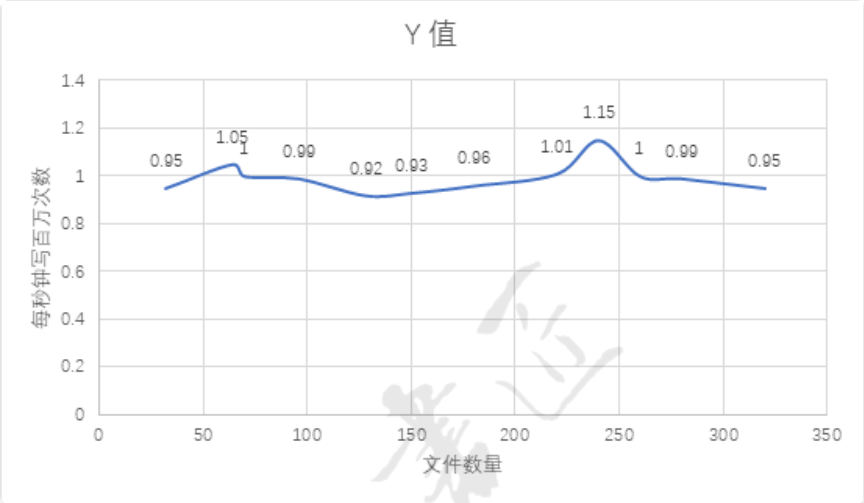
总体看来表现中等，数据库部分表现良好，网络模块的测试还算过的去，已测得QPS也达到了三万多，待相关问题

解决后，应该会有更高的表现，同时系统还有一下可见的不足之处，下边是我目前想到的：

1. 由于线程池队列里是一个有锁队列，且所有线程共享这一份队列，这个部分并发度不高 **正在解决**
   解决方案为替换为无锁队列，具体正在研究。

2. 由于客户端的套接字是唯一标识的，所以任务队列里某一时刻最多只能有一个此fd的待处理任务，这从根本上限制了QPS的提升 **正在解决**
   没有想到什么好的办法，或许one loop per thread 会是一个更好的解决方案。

3. 选择的文件个数没有测试其合理性，应该有一个更好的参数。 **已解决**
   解决方案： 进行了相关测试，文件数分别设置了32,64,70,96,128, 160,180 ,200,220 , 240, 260, 280, 560 如下图所示
   当文件数为240时，性能最高，八线程写100万次耗时7080945微秒，即7.08s,平均写次数达到1.130百万次/s 较64文件提升了10万次每秒。 读的提升倒不大，与之前基本持平。



写入能力与文件个数的关系

原因分析

这个问题之前有遇到过，当时没有在意，本次编程，又遇到这样的情况，网上也没有找到相关资料，和jim讨论一番，认为是因为超过阈值之后，更多文件不利与磁盘调度以及磁盘cache导致写性能下降。