A C++ Short URL Server

项目功能

- 使用 `makefile` 文件进行自动化编译
  - 自动生成文件依赖关系
- 配置文件读取及log输出
- 使用守护进程模式
- 根据监听端口号的不同来区分不同的业务，从而执行不同的业务逻辑
  - 短链及web服务
  - TCP回射服务
- 使用TCP连接池来提升系统响应速度
- I/O复用方式使用 `epoll` + `EPOLLONESHOT`
- 使用单 `Reactor` +多线程的模型来处理业务逻辑
  - `Reactor` 负责监听和分发事件
  - 线程池负责处理具体业务的处理
- 结合 `murmurhash` 实现了一个简单的布隆过滤器，主要用于过滤URL以减少恶意请求带来的对数据库的频繁访问

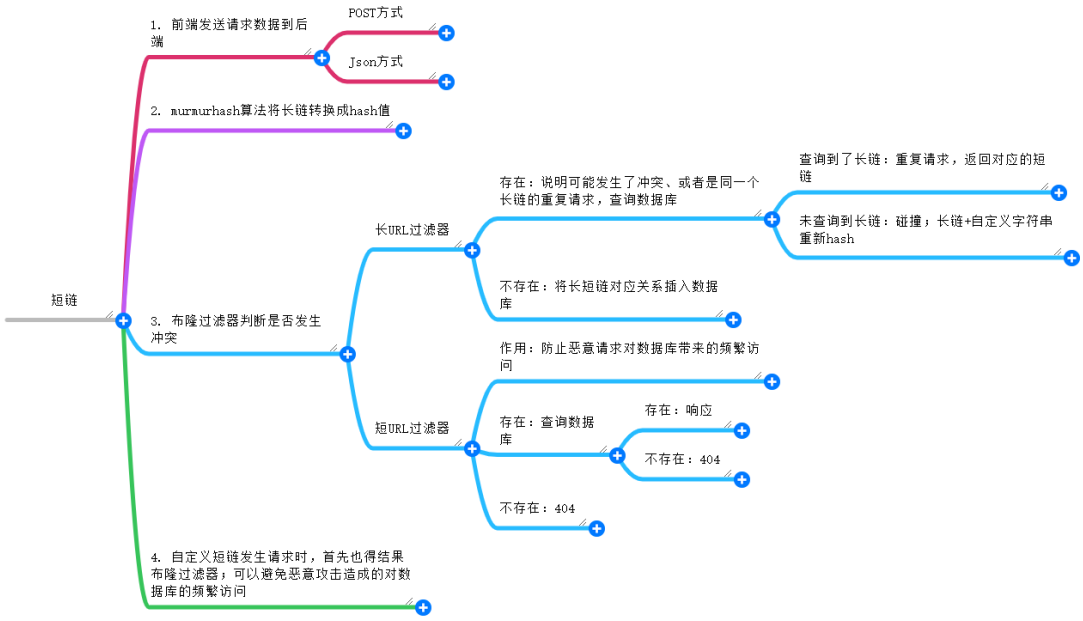
使用

`make` then `./urls`

测试地址

http://www.applestar.xyz/

短链处理流程

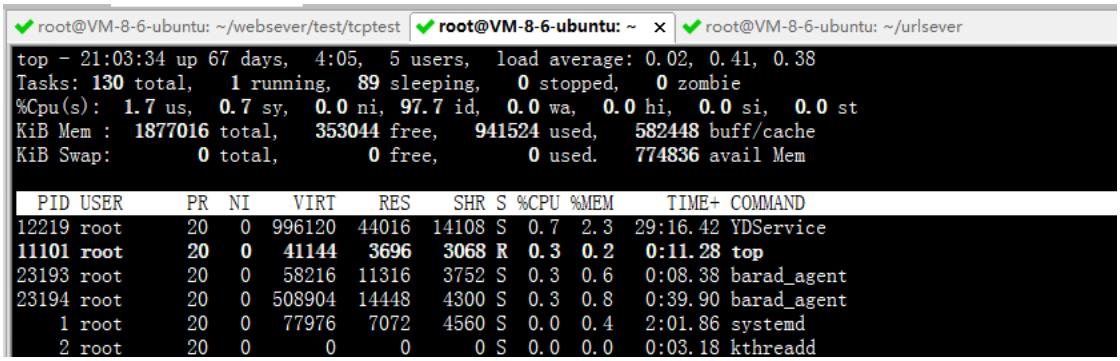


布隆过滤器

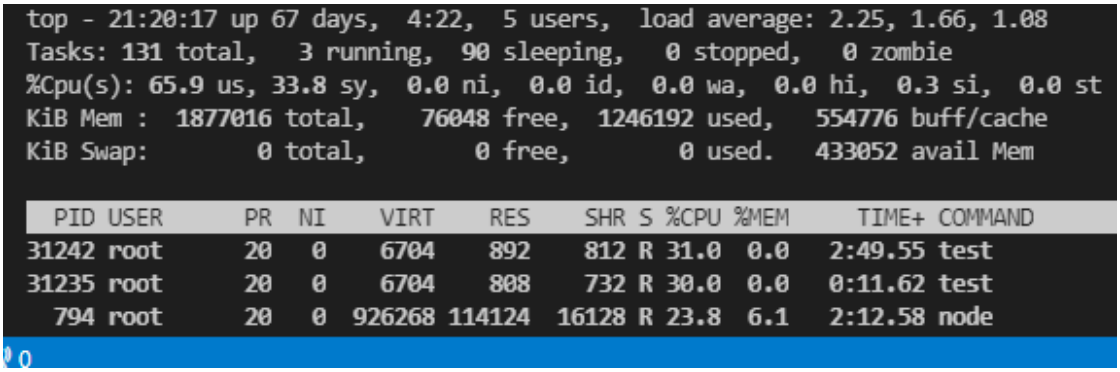
1. 布隆过滤器Bloom的使用
    - 定义：实际上时一个很长的二进制向量和一系列随机散列函数，可以用来教唆一个元素是否在一个集合中；优点是：**时空复杂度都比较低**，不需要key；缺点是 **有一定的误识别率而且删除困难**
    - 应用场景：
        - 邮件黑名单过滤器，判断一个邮件地址是否在黑名单中
        - 网络爬虫，判断该URL是否以及被爬取过
        - K-V系统查询之前进行判断，对于key不在的情况可以节省后续查询的时间，同时防止恶意攻击
2. **BloomFilter**的实现：
    - 首先需要k个hash函数，每个函数可以把输入变成一个整数
    - 需要一个bit容器，每个bit初始化成0
    - 当某个 `key` 加入时，k个hash函数分别对其进行计算并将hash得到的整数的那个位置置为1
    - 当查询某个 `key` 时，将hash得到的k个值的位置进行查询，如某个位置不为0，说明不存在
3. 如何解决删除的问题，可以将 `bit` 变成count进行计数，删除的时候减1即可
4. 如何对其进行设计呢？
    - 首先是hash的选取，hash计算的数值与 `bit` 数组的大小是相关联的
    - bit数组用什么容器来存储效率高？使用bitset局限性有点大，不仅不分辨调整大小，而且使用栈的空间，不合理；所以使用
    - bit数组如何进行存取
5. 当bloom过滤器发现可能发生hash冲突的时候，就采用**长链+自定义字符串**的形式重新hash；同时在返回查询的时候，需要需要进行判断

单机最大能支持的TCP连接数目测试

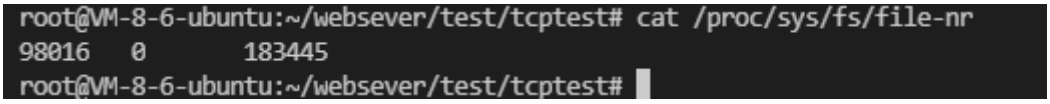
1. 测试前的状态：**350MB**free



测试后的状态：**76MB**free



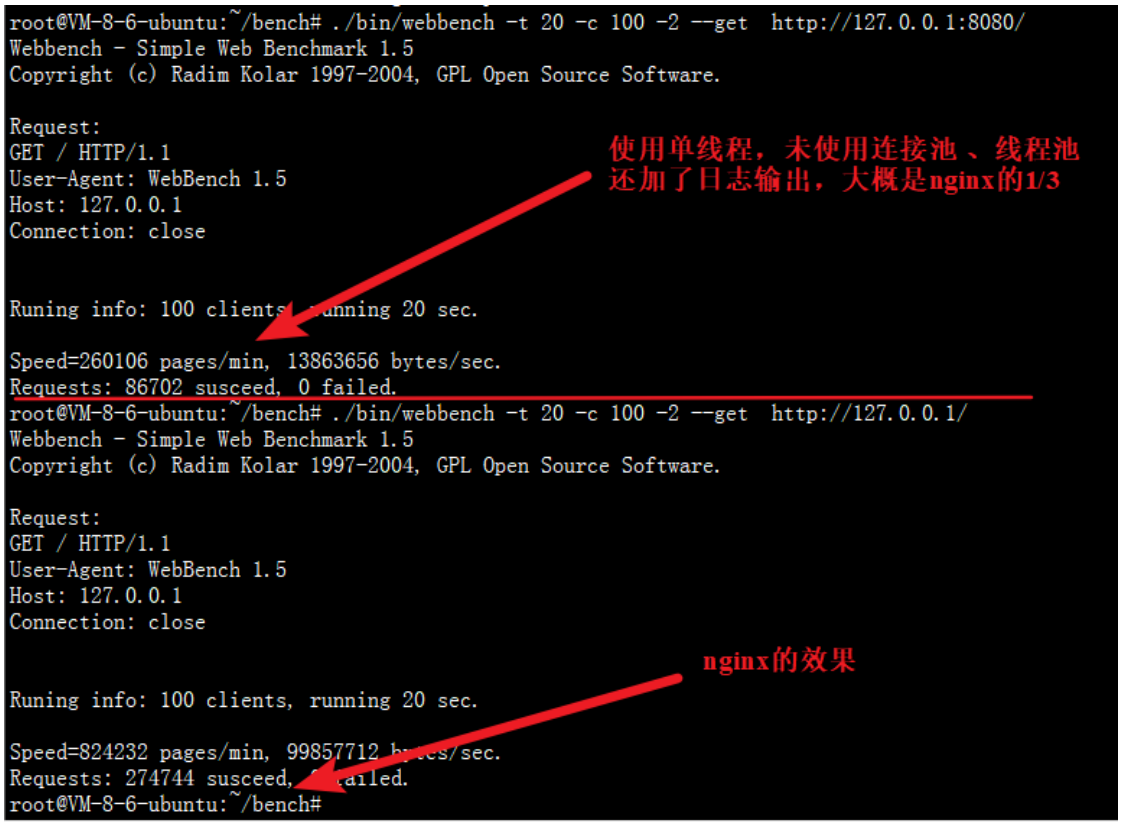
文件描述符使用个数：**10w**左右，操作系统还在慢慢寻找端口进行连接



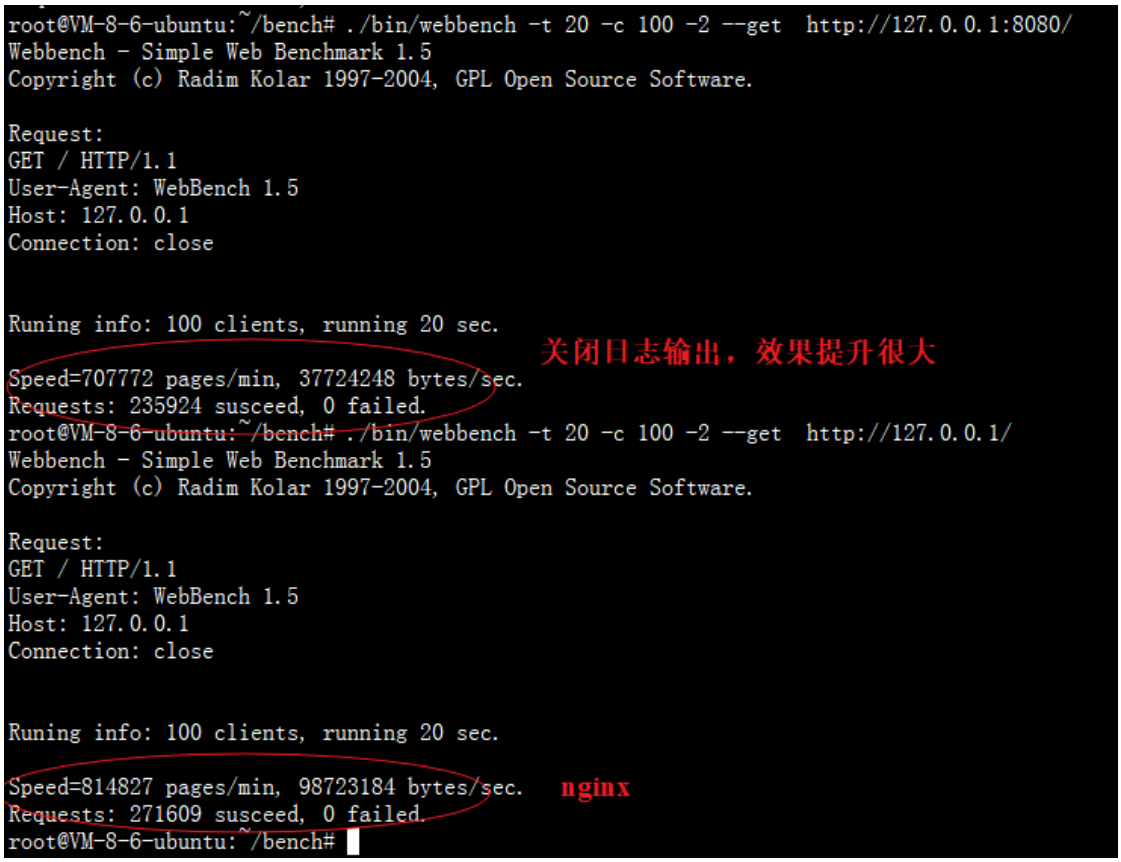
2. 测试结果表明：
   - 使用了本地机器的两个IP地址对进行测试，测试结果大约在**10万个左右**，而数目还在缓慢增加
   - 最大链接数目受限于端口和内存资源
   - 操作系统查找可用端口是需要时间的，前期查找特别快，可用端口数目越少，查找所需的时间就越长

压力测试

1. 只有一个主线程，主线程监听并处理所有事件，**未使用线程池也未使用连接池**，当即创建当即销毁（使用智能指针）



然后关闭了提升了日志输出等级（**无日志输出**）的效果，比较接近 `nginx` 了



2. 加了连接池的效果：

3. 加入了线程池的效果，单核CPU：与上述的结果相对比，效果不升反降；这是因为在单核CPU里面使用多线程额外带来的 **上下文线程间切换**的开销