

# 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

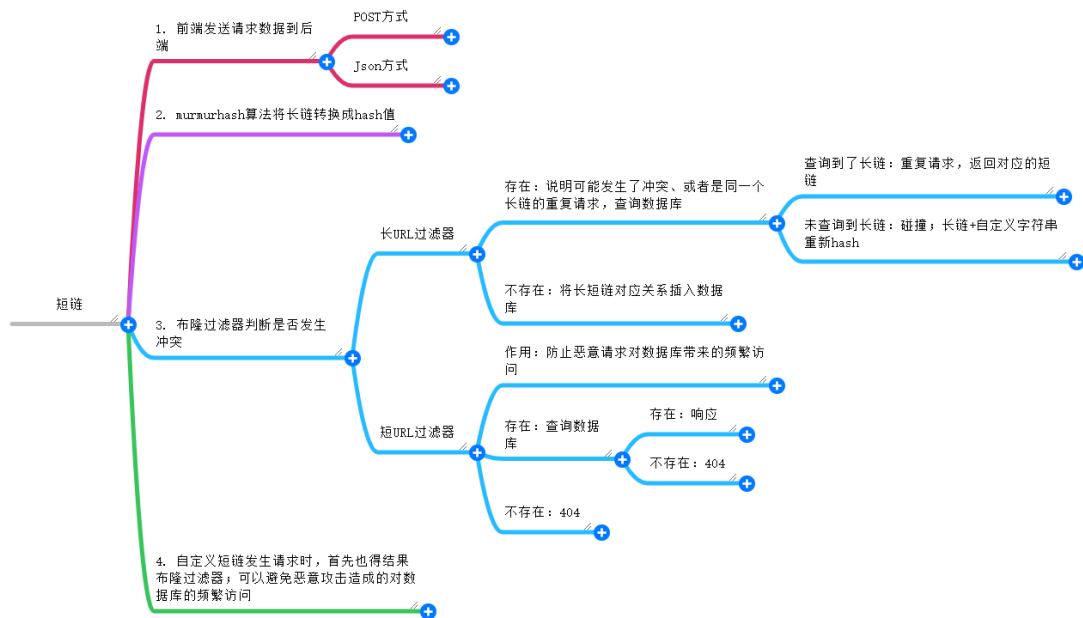
## 使用

```
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：实际上是一个很长的二进制向量和一系列随机散列函数，可以用来判断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：350MBfree

```
root@VM-8-6-ubuntu: ~/webserver/test/tcptest root@VM-8-6-ubuntu: ~ x root@VM-8-6-ubuntu: ~/urlserver
top - 21:03:34 up 67 days, 4:05, 5 users, load average: 0.02, 0.41, 0.38
Tasks: 130 total, 1 running, 89 sleeping, 0 stopped, 0 zombie
%Cpu(s): 1.7 us, 0.7 sy, 0.0 ni, 97.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 1877016 total, 353044 free, 941524 used, 582448 buff/cache
KiB Swap: 0 total, 0 free, 0 used. 774836 avail Mem

  PID USER      PR  NI   VIRT    RES    SHR S  %CPU  %MEM    TIME+  COMMAND
 12219 root        20   0  996120  44016  14108 S   0.7   2.3   29:16.42 YDService
 11101 root        20   0  41144    3696   3068 R   0.3   0.2    0:11.28 top
 23193 root        20   0   58216  11316   3752 S   0.3   0.6    0:08.38 barad_agent
 23194 root        20   0  508904  14448   4300 S   0.3   0.8    0:39.90 barad_agent
      1 root        20   0   77976   7072   4560 S   0.0   0.4    2:01.86 systemd
      2 root        20   0         0        0        0 S   0.0   0.0    0:03.18 kthreadd
```

测试后的状态：76MBfree

```
top - 21:20:17 up 67 days, 4:22, 5 users, load average: 2.25, 1.66, 1.08
Tasks: 131 total, 3 running, 90 sleeping, 0 stopped, 0 zombie
%Cpu(s): 65.9 us, 33.8 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 0.3 si, 0.0 st
KiB Mem : 1877016 total, 76048 free, 1246192 used, 554776 buff/cache
KiB Swap: 0 total, 0 free, 0 used. 433052 avail Mem

  PID USER      PR  NI   VIRT    RES    SHR S  %CPU  %MEM    TIME+  COMMAND
 31242 root        20   0    6704     892    812 R  31.0   0.0   2:49.55 test
 31235 root        20   0    6704     808    732 R  30.0   0.0    0:11.62 test
   794 root        20   0 926268 114124  16128 R  23.8   6.1   2:12.58 node
0
```

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

```
root@VM-8-6-ubuntu:~/webserver/test/tcptest# cat /proc/sys/fs/file-nr
98016 0 183445
root@VM-8-6-ubuntu:~/webserver/test/tcptest#
165088 2021/07/12 21:27:38 [info] 收到了1个事件
165089 2021/07/12 21:27:38 [info] 监听端口: 8080 client:127.0.0.1:49552
165090 2021/07/12 21:27:38 [info] accept_fd:56249 加入了epoll中
165091 2021/07/12 21:27:38 [info] 收到了1个事件
165092 2021/07/12 21:27:38 [info] 监听端口: 8080 client:127.0.0.1:50064
165093 2021/07/12 21:27:38 [info] accept_fd:56250 加入了epoll中
165094 2021/07/12 21:27:38 [info] 收到了1个事件
165095 2021/07/12 21:27:38 [info] 监听端口: 8080 client:127.0.0.1:50576
165096 2021/07/12 21:27:38 [info] accept_fd:56251 加入了epoll中
165097 2021/07/12 21:27:38 [info] 收到了1个事件
165098
```

### 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）

```
root@VM-8-6-ubuntu:~/bench# ./bin/webbench -t 20 -c 100 -2 --get http://127.0.0.1:8080/
Webbench - Simple Web Benchmark 1.5
Copyright (c) Radim Kolar 1997-2004, GPL Open Source Software.

Request:
GET / HTTP/1.1
User-Agent: WebBench 1.5
Host: 127.0.0.1
Connection: close

Running info: 100 clients, running 20 sec.

Speed=260106 pages/min, 13863656 bytes/sec.
Requests: 86702 succeed, 0 failed.
root@VM-8-6-ubuntu:~/bench# ./bin/webbench -t 20 -c 100 -2 --get http://127.0.0.1/
Webbench - Simple Web Benchmark 1.5
Copyright (c) Radim Kolar 1997-2004, GPL Open Source Software.

Request:
GET / HTTP/1.1
User-Agent: WebBench 1.5
Host: 127.0.0.1
Connection: close

Running info: 100 clients, running 20 sec.

Speed=824232 pages/min, 99857712 bytes/sec.
Requests: 274744 succeed, 0 failed.
root@VM-8-6-ubuntu:~/bench#
```

使用单线程，未使用连接池、线程池还加了日志输出，大概是nginx的1/3

nginx的效果

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

```
root@VM-8-6-ubuntu:~/bench# ./bin/webbench -t 20 -c 100 -2 --get http://127.0.0.1:8080/
Webbench - Simple Web Benchmark 1.5
Copyright (c) Radim Kolar 1997-2004, GPL Open Source Software.

Request:
GET / HTTP/1.1
User-Agent: WebBench 1.5
Host: 127.0.0.1
Connection: close

Running info: 100 clients, running 20 sec.

Speed=707772 pages/min, 37724248 bytes/sec.
Requests: 235924 succeed, 0 failed.
root@VM-8-6-ubuntu:~/bench# ./bin/webbench -t 20 -c 100 -2 --get http://127.0.0.1/
Webbench - Simple Web Benchmark 1.5
Copyright (c) Radim Kolar 1997-2004, GPL Open Source Software.

Request:
GET / HTTP/1.1
User-Agent: WebBench 1.5
Host: 127.0.0.1
Connection: close

Running info: 100 clients, running 20 sec.

Speed=814827 pages/min, 98723184 bytes/sec.
Requests: 271609 succeed, 0 failed.
root@VM-8-6-ubuntu:~/bench#
```

关闭日志输出，效果提升很大

nginx

2. 加了连接池的效果:

```

root@VM-8-6-ubuntu:~/bench# ./bin/webbench -t 20 -c 100 -2 --get http://127.0.0.1:8080/
Webbench - Simple Web Benchmark 1.5
Copyright (c) Radim Kolar 1997-2004, GPL Open Source Software.

Request:
GET / HTTP/1.1
User-Agent: WebBench 1.5
Host: 127.0.0.1
Connection: close

Runing info: 100 clients, running 20 sec.

Speed=712494 pages/min, 37976088 bytes/sec.
Requests: 237498 succeed, 0 failed.
root@VM-8-6-ubuntu:~/bench# ./bin/webbench -t 20 -c 100 -2 --get http://127.0.0.1/
Webbench - Simple Web Benchmark 1.5
Copyright (c) Radim Kolar 1997-2004, GPL Open Source Software.

Request:
GET / HTTP/1.1
User-Agent: WebBench 1.5
Host: 127.0.0.1
Connection: close

Runing info: 100 clients, running 20 sec.

Speed=825783 pages/min, 100045456 bytes/sec.
Requests: 275261 succeed, 0 failed.
root@VM-8-6-ubuntu:~/bench# █

```

加了连接池，效果跟没加一样

nginx

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

```

root@VM-8-6-ubuntu:~/bench# ./bin/webbench -t 20 -c 100 -2 --get http://127.0.0.1:8080/
Webbench - Simple Web Benchmark 1.5
Copyright (c) Radim Kolar 1997-2004, GPL Open Source Software.

Request:
GET / HTTP/1.1
User-Agent: WebBench 1.5
Host: 127.0.0.1
Connection: close

Runing info: 100 clients, running 20 sec.

Speed=570837 pages/min, 30425612 bytes/sec.
Requests: 190279 succeed, 0 failed.
root@VM-8-6-ubuntu:~/bench# ./bin/webbench -t 20 -c 100 -2 --get http://127.0.0.1:8080/
Webbench - Simple Web Benchmark 1.5
Copyright (c) Radim Kolar 1997-2004, GPL Open Source Software.

Request:
GET / HTTP/1.1
User-Agent: WebBench 1.5
Host: 127.0.0.1
Connection: close

Runing info: 100 clients, running 20 sec.

Speed=659760 pages/min, 35165208 bytes/sec.
Requests: 219920 succeed, 0 failed.
root@VM-8-6-ubuntu:~/bench#

```

使用了线程池

未使用线程池，调用的一个全局函数进行处理

