基于OpenResty的百万级长连接推送

酷狗音乐　朱德江

doujiang24@gmail.com

大纲

- 实时推送系统设计
- 实施过程与优化经历

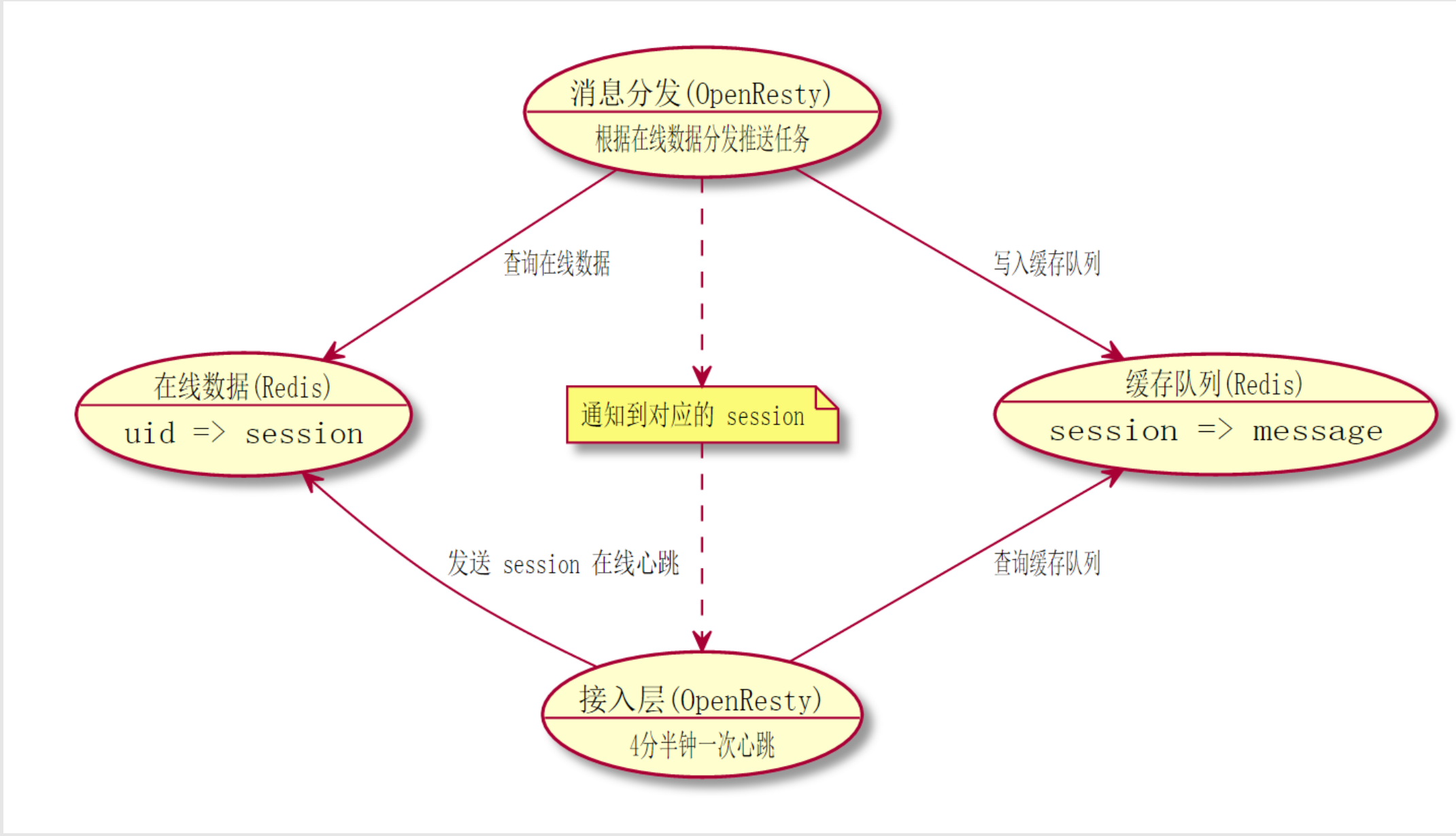
特点与挑战

- 并发量大（单连接消耗内存）
- 一般不活跃（维持心跳，4.5min）
- 偶尔很活跃（需要流控）

为什么选择 OpenResty

- 其他的我也不会

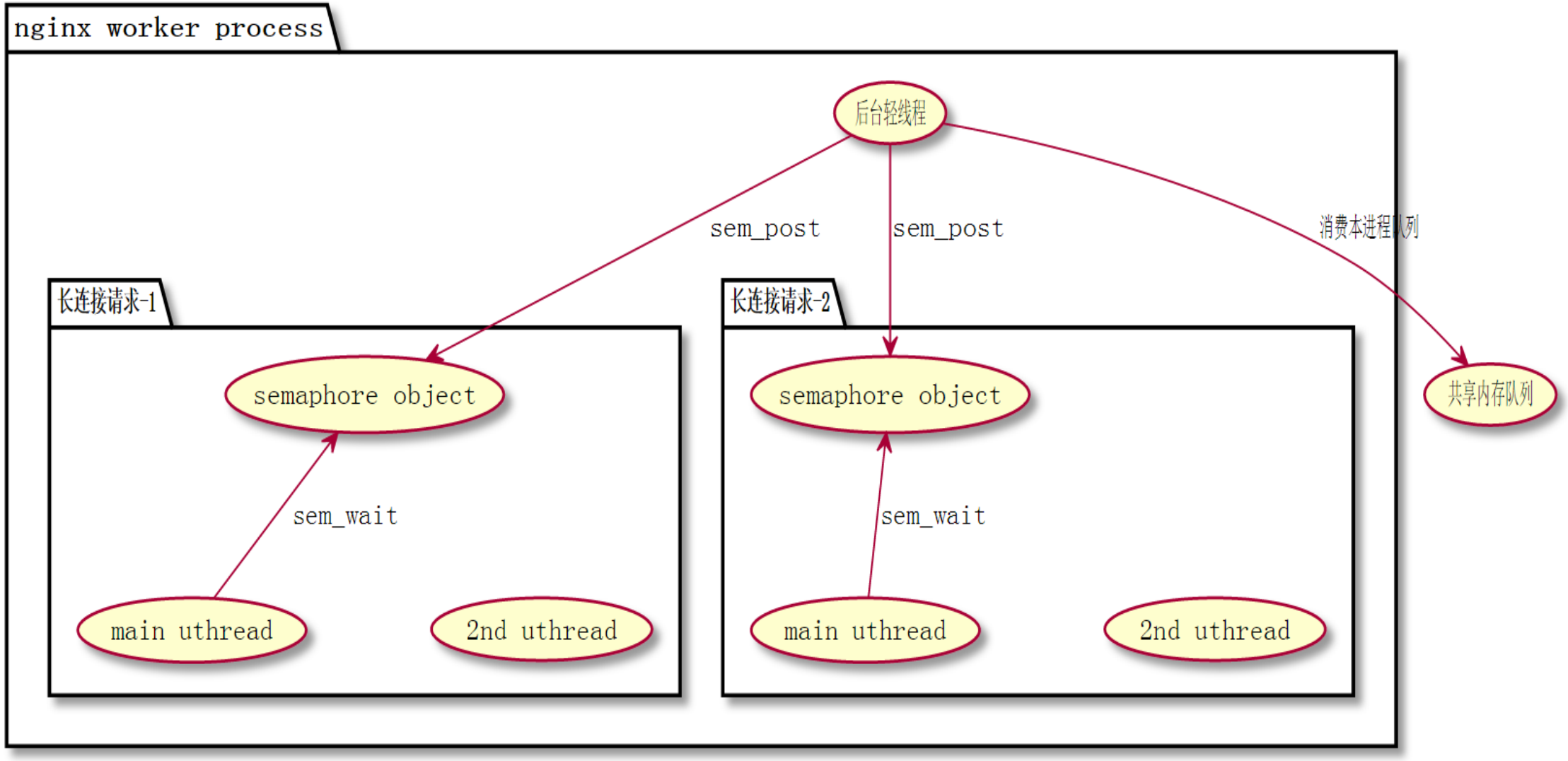
- 开发效率快，有多快呢
- 运行效率高，有多高呢


消息分发(OpenResty)
根据在线数据分发推送任务
查询在线数据
写入缓存队列
在线数据(Redis)
uid => session
通知到对应的 session
缓存队列(Redis)
session => message
发送 session 在线心跳
查询缓存队列
接入层(OpenResty)
4分半钟一次心跳

- 接入层尽量简单，无状态化

- 长连接维持： 全双工 cosocket + 轻线程

- main uthread： wait on notification, send message
- 2nd uthread: ping-pong, keepalive session in Redis


nginx worker process
后台轻线程
sem_post
sem_post
消费本进程队列
长连接请求-1
长连接请求-2
semaphore object
semaphore object
共享内存队列
sem_wait
sem_wait
main uthread
2nd uthread
main uthread
2nd uthread

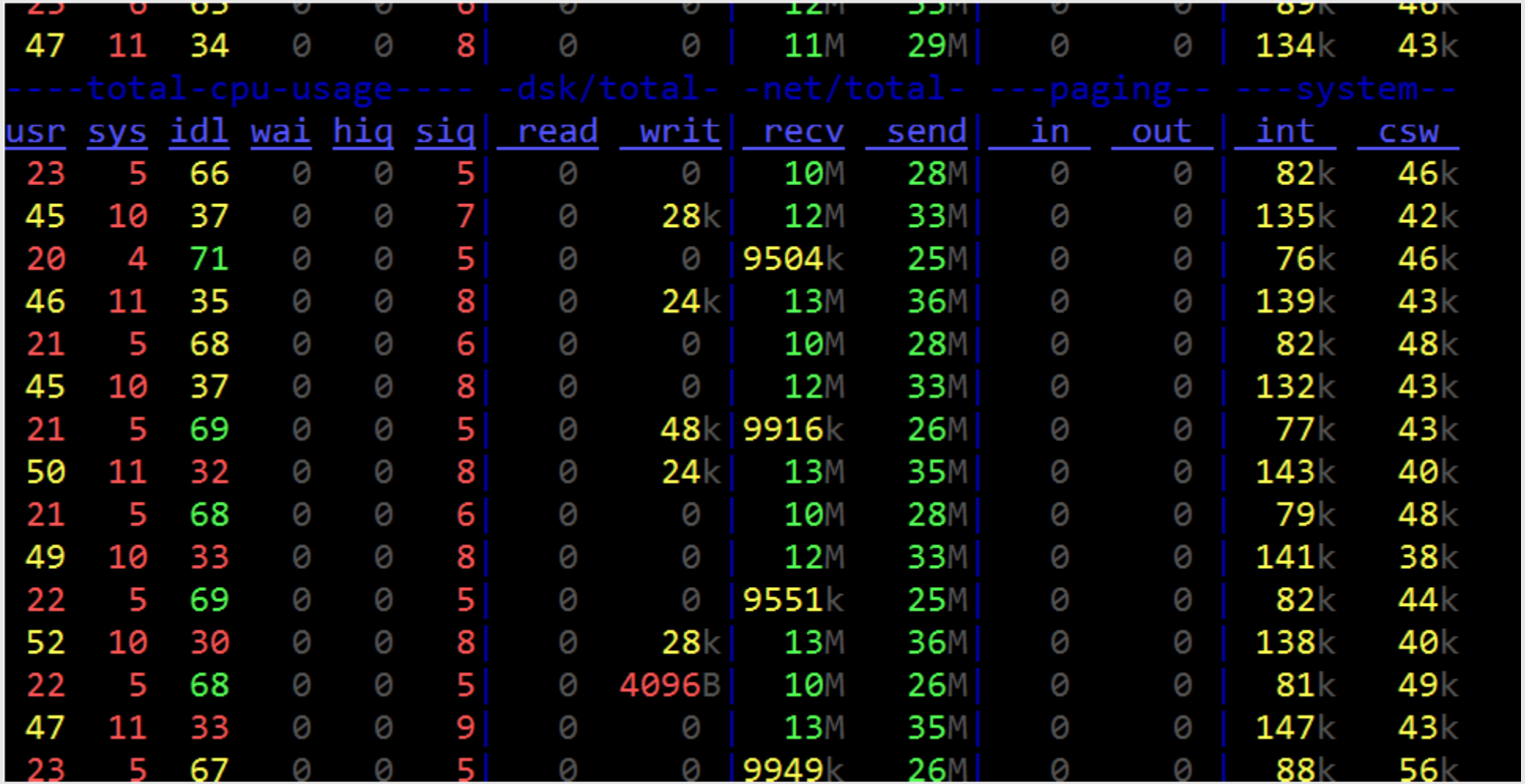
- Sessionid（serverid + worker_id + uid + incr_num）
- （Lua number size， 52bit）

- 共享内存队列（进程间通讯）
- worker 级别的流控

- ngx.semaphore（进程内 “轻线程”间通讯）

```
Total: 2004020 (kernel 2004083)
TCP:   2003871 (estab 2003858, closed 0, orphaned 0, synrecv 0, timewait 0/0), po

Transport Total     IP        IPv6
*         2004083   -         -
RAW       0         0         0
UDP       10        6         4
TCP       2003871   2003870   1
INET      2003881   2003876   5
FRAG      0         0         0

             total       used       free     shared    buffers     cached
Mem:         64531      54933       9598          0        870      15653
-/+ buffers/cache:      38410      26121
Swap:         8191        337       7854
```

```
top - 14:19:04 up 116 days, 23:23, 18 users,  load average: 1.95, 2.45, 1.59
Tasks: 322 total,   2 running, 305 sleeping,  14 stopped,   1 zombie
Cpu(s):  3.8%us,  1.4%sy,  0.0%ni, 93.7%id,  0.0%wa,  0.0%hi,  1.1%si,  0.0%st
Mem:  66080532k total, 57981128k used,  8099404k free,   890952k buffers
Swap:  8388604k total,   330760k used,  8057844k free, 16028888k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
31727 root      20   0  722m 443m 1196 S 26.6  0.7 115:37.92 redis-server
55083 root      20   0 12.9g 1.6g 385m S  6.3  2.6   0:59.07 nginx
55088 root      20   0 12.9g 1.6g 384m S  5.6  2.6   0:58.85 nginx
55073 root      20   0 12.9g 1.7g 386m S  5.3  2.7   0:59.05 nginx
```



47	11	34	0	0	8	0	0	11M	29M	0	0	134k	43k
----total-cpu-usage----						-dsk/total-		-net/total-		---paging--		---system--	
usr	sys	idl	wai	hiq	siq	read	writ	recv	send	in	out	int	csw
23	5	66	0	0	5	0	0	10M	28M	0	0	82k	46k
45	10	37	0	0	7	0	28k	12M	33M	0	0	135k	42k
20	4	71	0	0	5	0	0	9504k	25M	0	0	76k	46k
46	11	35	0	0	8	0	24k	13M	36M	0	0	139k	43k
21	5	68	0	0	6	0	0	10M	28M	0	0	82k	48k
45	10	37	0	0	8	0	0	12M	33M	0	0	132k	43k
21	5	69	0	0	5	0	48k	9916k	26M	0	0	77k	43k
50	11	32	0	0	8	0	24k	13M	35M	0	0	143k	40k
21	5	68	0	0	6	0	0	10M	28M	0	0	79k	48k
49	10	33	0	0	8	0	0	12M	33M	0	0	141k	38k
22	5	69	0	0	5	0	0	9551k	25M	0	0	82k	44k
52	10	30	0	0	8	0	28k	13M	36M	0	0	138k	40k
22	5	68	0	0	5	0	4096B	10M	26M	0	0	81k	49k
47	11	33	0	0	9	0	0	13M	35M	0	0	147k	43k
23	5	67	0	0	5	0	0	9949k	26M	0	0	88k	56k

- 200w 连接消耗约 40G 内存（约20k 每连接）
- 维持心跳消耗 7% CPU

- send 100byte message 20w/2s
- CPU: 50%
- Mem: 基本不变
- 得益于各级 buffer，实际上需要动态申请的内存很少

磨刀不误砍柴工

- nginx-systemstap-tools
- https://github.com/openresty/nginx-systemtap-toolkit

- stap++
- https://github.com/openresty/stapxx

- nginx-gdb-utils
- https://github.com/openresty/nginx-gdb-utils

- 前提是：看懂宿主程序代码
- https://github.com/agentzh/code2ebook

内存都去哪了呢

- kernel
- TCP/IP 协议栈

- nginx
- request pool

- LuaJIT
- uthread, cosocket

```
probe @pfunc(ngx_palloc)
{
    if (pid() == target()) {
        printf("\n\nrequest pool size: %d\n", ngx_pool_size($pool))
        printf("alloc size: %d\n\n", $size)
        print_ubacktrace()
        println(luajit_print_backtrace(1))
    }
}
```

https://github.com/doujiang24/stapxx/blob/palloc/samples/ngx-palloc.sxx

```
request pool size: 8192
alloc size: 4096

 0x41b220 : ngx_palloc+0x0/0x50 [/usr/local/openresty-debug/nginx/sbin/nginx]
 0x41ced3 : ngx_create_temp_buf+0x53/0x80 [/usr/local/openresty-debug/nginx/sbin/nginx]
 0x4ba01e : ngx_http_lua_chain_get_free_buf+0x1ee/0x250 [/usr/local/openresty-debug/nginx/sbin/nginx]
 0x4cbc53 : ngx_http_lua_socket_tcp_receive+0x4e3/0x550 [/usr/local/openresty-debug/nginx/sbin/nginx]
 0x7f020358d99b : lj_BC_FUNCC+0x34/0x59 [/usr/local/openresty-debug/luajit/lib/libluajit-5.1.so.2.1.0]
 0x4bc429 : ngx_http_lua_run_thread+0xd9/0x13d0 [/usr/local/openresty-debug/nginx/sbin/nginx]
 0x4c63e2 : ngx_http_lua_socket_tcp_resume_helper+0xf2/0x210 [/usr/local/openresty-debug/nginx/sbin/nginx]
 0x4cbe17 : ngx_http_lua_socket_connected_handler+0x157/0x230 [/usr/local/openresty-debug/nginx/sbin/nginx]
 0x4c5d8b : ngx_http_lua_socket_tcp_handler+0x8b/0xd0 [/usr/local/openresty-debug/nginx/sbin/nginx]
 0x43b0c3 : ngx_epoll_process_events+0x453/0x480 [/usr/local/openresty-debug/nginx/sbin/nginx]
 0x431ca5 : ngx_process_events_and_timers+0x65/0x1b0 [/usr/local/openresty-debug/nginx/sbin/nginx]
 0x439695 : ngx_worker_process_cycle+0xd5/0x200 [/usr/local/openresty-debug/nginx/sbin/nginx]
 0x4379c4 : ngx_spawn_process+0x194/0x590 [/usr/local/openresty-debug/nginx/sbin/nginx]
 0x43895c : ngx_start_worker_processes+0x6c/0xd0 [/usr/local/openresty-debug/nginx/sbin/nginx]
 0x439e33 : ngx_master_process_cycle+0x1c3/0xab0 [/usr/local/openresty-debug/nginx/sbin/nginx]
 0x419fa6 : main+0x936/0x9d0 [/usr/local/openresty-debug/nginx/sbin/nginx]
 0x382c41ed5d : __libc_start_main+0xfd/0x1d0 [/lib64/libc-2.12.so]
 0x4186d9 : _start+0x29/0x2c [/usr/local/openresty-debug/nginx/sbin/nginx]

C:ngx_http_lua_socket_tcp_receive
@/usr/local/openresty-debug/lualib/resty/redis.lua:165
```

```
request pool size: 16416
alloc size: 24

 0x41b220 : ngx_palloc+0x0/0x50 [/usr/local/openresty-debug/nginx/sbin/nginx]
 0x449d5b : ngx_http_cleanup_add+0x2b/0xc0 [/usr/local/openresty-debug/nginx/sbin/nginx]
 0x4ca321 : ngx_http_lua_socket_tcp_connect+0x881/0xd20 [/usr/local/openresty-debug/nginx/sbin/nginx]
 0x7f020358d99b : lj_BC_FUNCC+0x34/0x59 [/usr/local/openresty-debug/luajit/lib/libluajit-5.1.so.2.1.0]
 0x4bc429 : ngx_http_lua_run_thread+0xd9/0x13d0 [/usr/local/openresty-debug/nginx/sbin/nginx]
 0x4c63e2 : ngx_http_lua_socket_tcp_resume_helper+0xf2/0x210 [/usr/local/openresty-debug/nginx/sbin/nginx]
 0x4cb172 : ngx_http_lua_socket_tcp_read+0x5e2/0x7b0 [/usr/local/openresty-debug/nginx/sbin/nginx]
 0x44d39f : ngx_http_request_handler+0x3f/0xa0 [/usr/local/openresty-debug/nginx/sbin/nginx]
 0x43b091 : ngx_epoll_process_events+0x421/0x480 [/usr/local/openresty-debug/nginx/sbin/nginx]
 0x431ca5 : ngx_process_events_and_timers+0x65/0x1b0 [/usr/local/openresty-debug/nginx/sbin/nginx]
 0x439695 : ngx_worker_process_cycle+0xd5/0x200 [/usr/local/openresty-debug/nginx/sbin/nginx]
 0x4379c4 : ngx_spawn_process+0x194/0x590 [/usr/local/openresty-debug/nginx/sbin/nginx]
 0x43895c : ngx_start_worker_processes+0x6c/0xd0 [/usr/local/openresty-debug/nginx/sbin/nginx]
 0x439e33 : ngx_master_process_cycle+0x1c3/0xab0 [/usr/local/openresty-debug/nginx/sbin/nginx]
 0x419fa6 : main+0x936/0x9d0 [/usr/local/openresty-debug/nginx/sbin/nginx]
 0x382c41ed5d : __libc_start_main+0xfd/0x1d0 [/lib64/libc-2.12.so]
 0x4186d9 : _start+0x29/0x2c [/usr/local/openresty-debug/nginx/sbin/nginx]
```

Nginx 内存优化项

- request_pool_size 1k
- lua_socket_buffer_size 1k
- cosocket patch
- https://github.com/openresty/lua-nginx-module/pull/590
- 8k / request

```
(gdb) lgcstat
724222 str        objects: max=1104, avg = 32, min=18, sum=23693826
 510 upval       objects: max=24, avg = 24, min=24, sum=12240
320807 thread     objects: max=976, avg = 777, min=424, sum=249476104
 187 proto       objects: max=2579, avg = 361, min=74, sum=67656
160709 func       objects: max=144, avg = 20, min=20, sum=3221976
 130 trace       objects: max=1828, avg = 1124, min=160, sum=146132
160052 cdata      objects: max=4112, avg = 16, min=12, sum=2564920
2119880 tab        objects: max=6291488, avg = 117, min=32, sum=248670968
449141 udata      objects: max=17694, avg = 440, min=32, sum=197711090

 sizeof strhash 4194304
 sizeof g->tmpbuf 1024
 sizeof ctype_state 4568
 sizeof jit_state 6032

total sz 729776072
g->strnum 724222, g->gc.total 729776072
elapsed: 447.080000 sec
```

LuaJIT 内存优化项

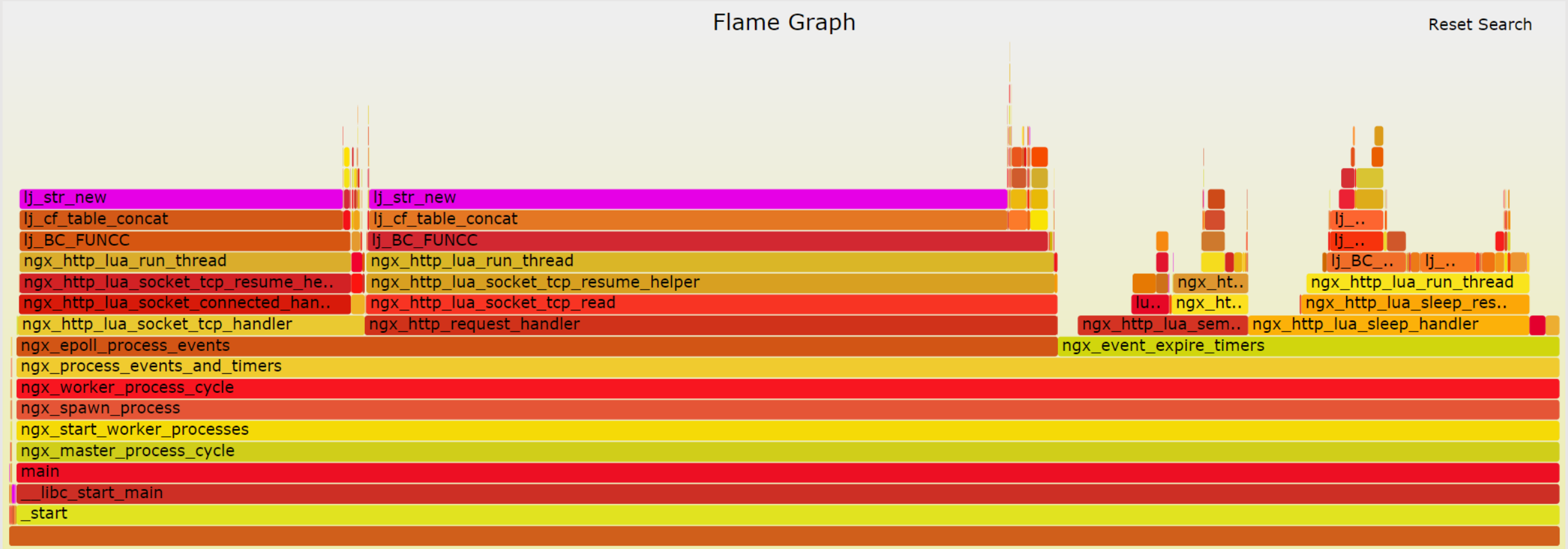
- 避免动态创建对象:
- closure -> shared table + static func
- 尽量共享 table

- lgcstat　查看所有没释放的对象（死的也有）
- lgcpath 只查看活的路径
- GC setpause 90

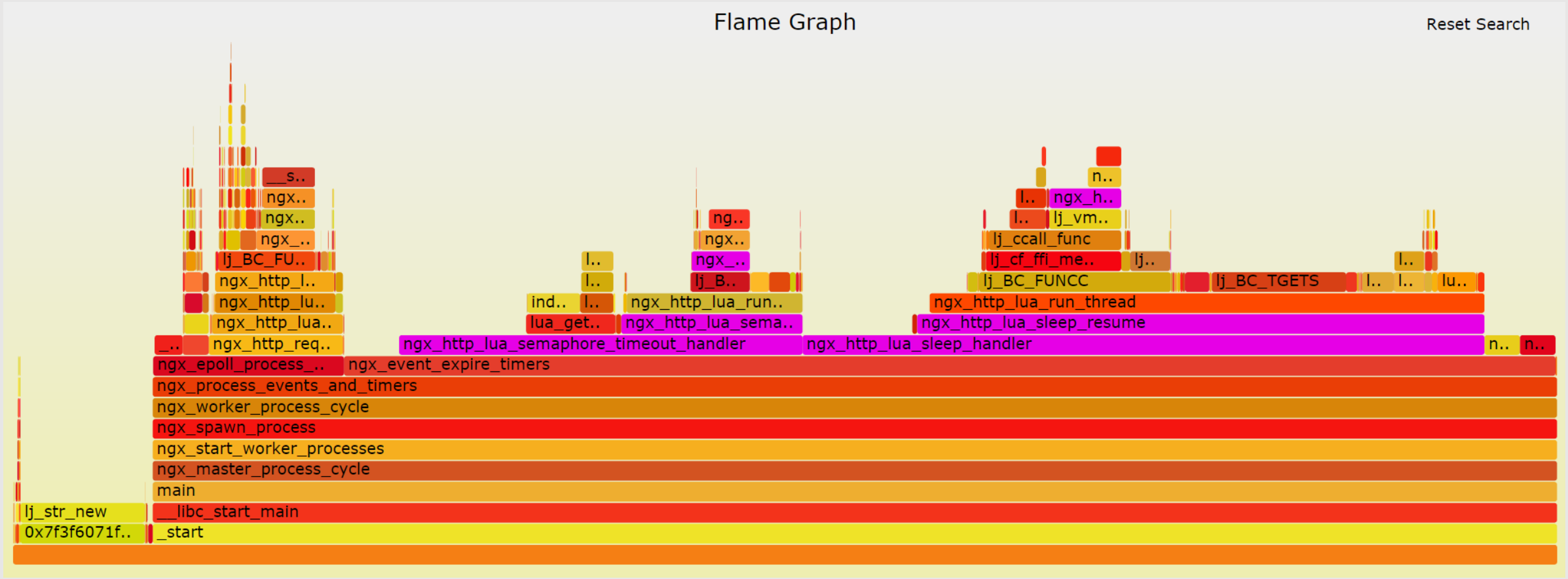
- gcore -o file pid

- 3k / request

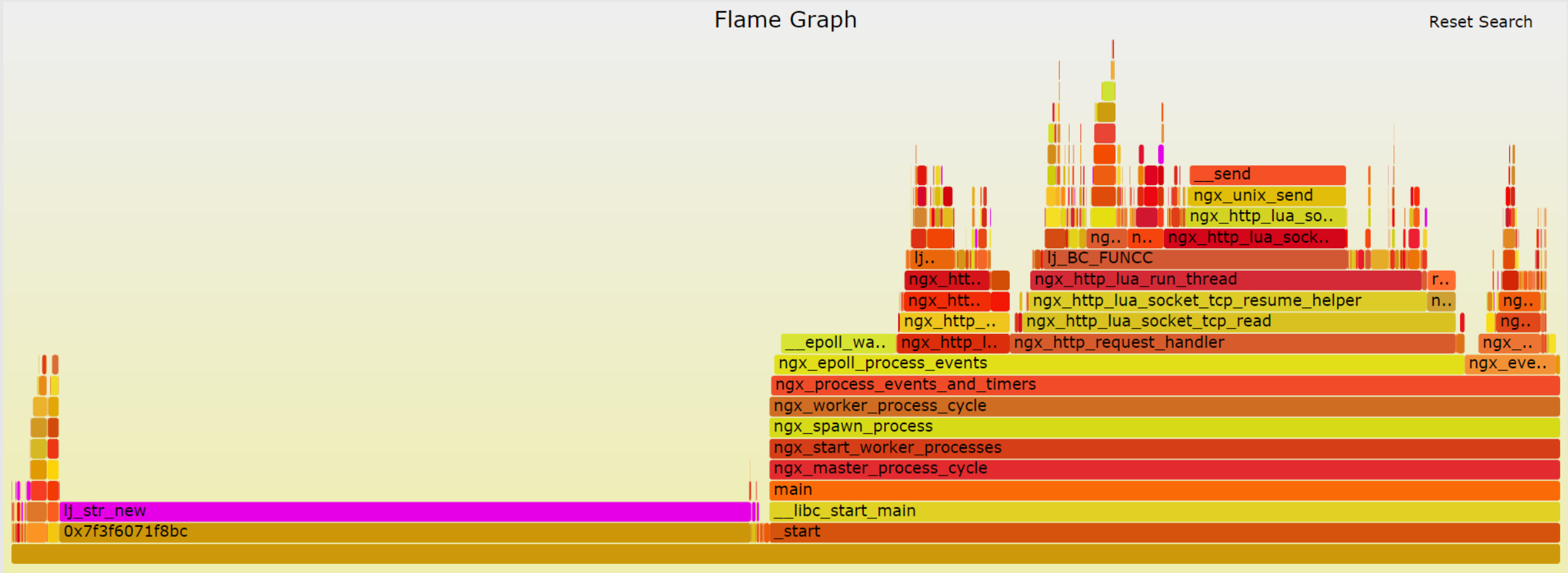
```
(gdb) lgcstat
301463 str        objects: max=2012, avg = 31, min=18, sum=9597298
 586 upval      objects: max=24, avg = 24, min=24, sum=14064
320345 thread     objects: max=1120, avg = 664, min=424, sum=212712496
 213 proto      objects: max=2579, avg = 356, min=74, sum=75945
160947 func       objects: max=144, avg = 20, min=20, sum=3231576
  91 trace      objects: max=1860, avg = 606, min=160, sum=55208
160032 cdata      objects: max=4112, avg = 16, min=12, sum=2564600
1213829 tab        objects: max=6291488, avg = 159, min=32, sum=194150416
272538 udata      objects: max=17694, avg = 440, min=32, sum=119927236

 sizeof strhash 2097152
 sizeof g->tmpbuf 2048
 sizeof ctype_state 4568
 sizeof jit_state 3472

total sz 544441311
g->strnum 301463, g->gc.total 544441311
elapsed: 284.000000 sec
```



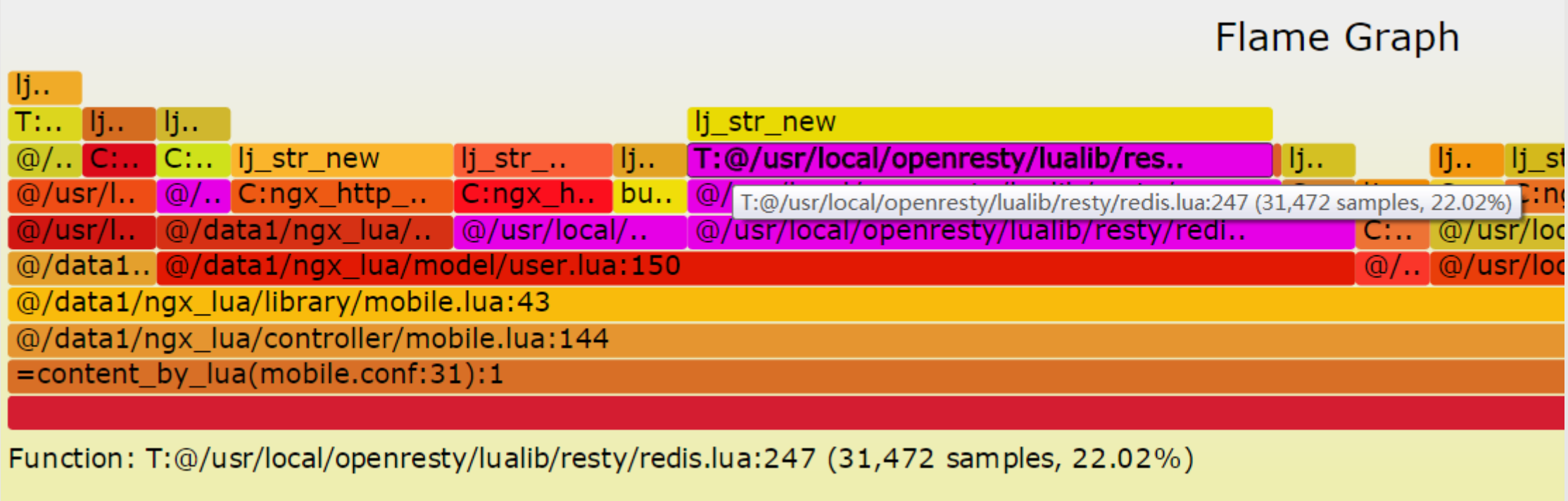
CPU: 90%(去掉 resty.redis 里的 concat)



CPU: 40% (去掉多余的 ngx.sleep, sem:wait from 2 to 240)



CPU: 10-18% (如何优化呢)



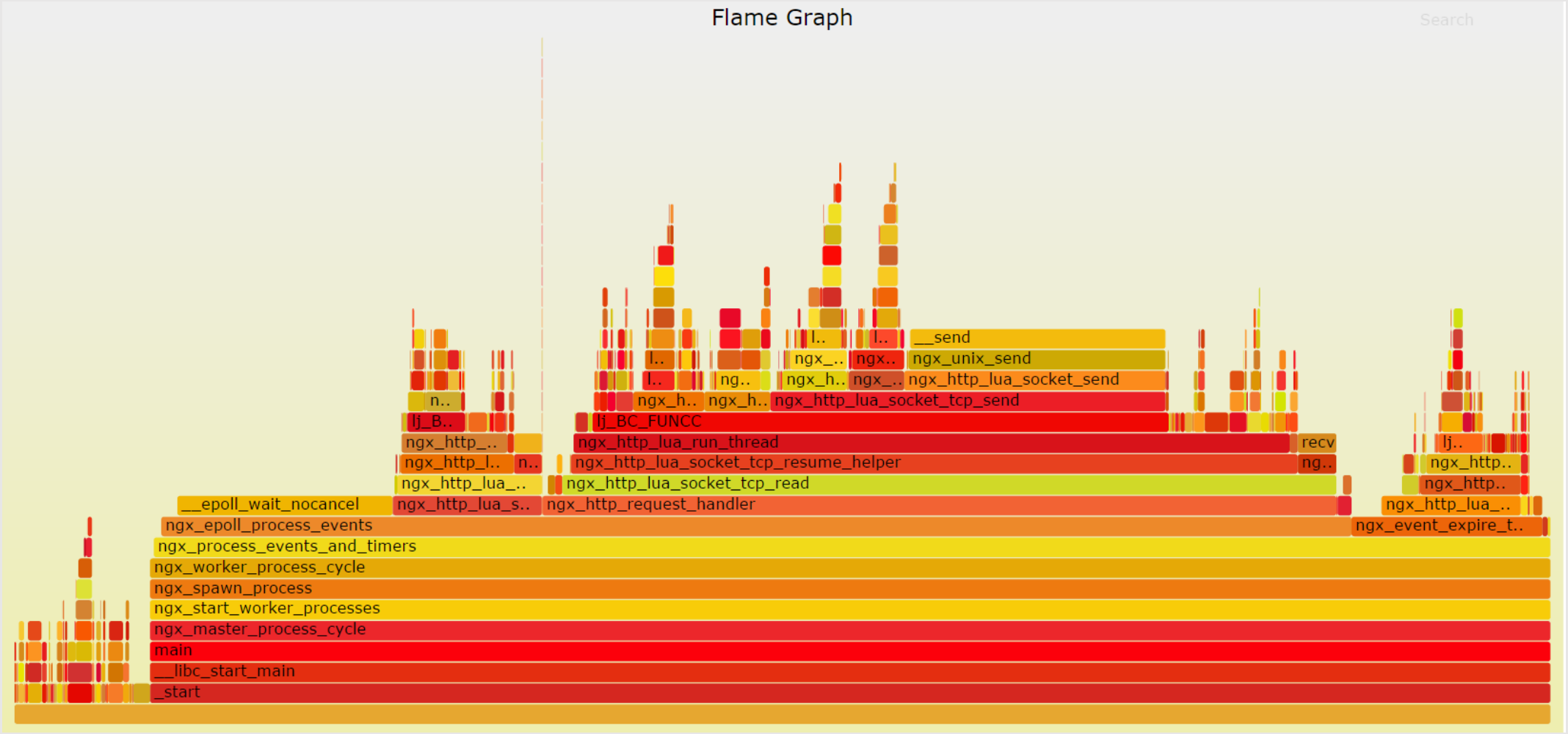
./samples/lj-lua-stacks.sxx -x $pid

--skip-badvars

--arg time=5

--arg probe='process("$^liblua_path").function("lj_str_new")‘

-D STP_NO_OVERLOAD



CPU: 7% (gc faster & use table instead .. in _gen_req)

CPU 优化项

- ngx.semaphore + shdict list
- lua-resty-redis

未完待续...

- 压榨内存？
- kernel 态 tcp 栈 内存消耗?

- lua-resty-redis-cluster

- ngx-stream-lua

个人小体会

- 测试驱动，嗯，test-nginx 很赞
- 动态追踪真的很赞，激励我去看代码，insight
- curl http://localhost/internal --data-urlencode script@script.lua
- github 开发方式

OVER ☺

doujiang24@gmail.com