

# 如何使用Gcov和Linux Perf 工具抓热点代码

PCLT Lab

[xiaou@iscas.ac.cn](mailto:xiaou@iscas.ac.cn)

2021.03.31

# 目录

- Gcov抓热点
- Linux Perf抓热点

# Gcov简介

- Gcov
  - Linux下GCC自带的C/C++代码覆盖率分析工具
  - 可以用于查找热点，有针对性地优化代码
  - 也可以查看代码覆盖情况，有针对性的为软件添加测试用例
  - 配合lcov使用，图形化显示

# Gcov使用

1. **gcc -fprofile-arcs -ftest-coverage test.c -o test**

————→ test test.gcno

2. **./test**

————→ test.gcda

3. **gcov test.c**

————→ test.c.gcov

4. **vim test.c.gcov**

# Gcov使用

```
// test.c
#include <stdio.h>
#include <stdlib.h>

void test(int count)
{
    int i;
    for (i = 1; i < count; i++)
    {
        if (i % 3 == 0)
            printf ("%d is divisible by 3\n", i);
        if (i % 11 == 0)
            printf ("%d is divisible by 11\n", i);
        if (i % 13 == 0)
            printf ("%d is divisible by 13\n", i);
    }
}

int main(int argc, char *argv[])
{
    int i = 0;
    if(argc == 2)
        i = atoi(argv[1]);
    else
        i = 10;

    printf("arg is %d\n", i);

    test(i);

    return EXIT_SUCCESS;
}
```

# Gcov使用

```
molly@molly-Huawei:~/test$ gcc -fprofile-arcs -ftest-coverage test.c -o test
molly@molly-Huawei:~/test$ ls
test  test.c  test.gcno
```

# Gcov使用

```
molly@molly-Huawei:~/test$ ./test
arg is 10
3 is divisible by 3
6 is divisible by 3
9 is divisible by 3
molly@molly-Huawei:~/test$ ls
test  test.c  test.gcda  test.gcno
```

```
molly@molly-Huawei:~/test$ gcov test.c
File 'test.c'
Lines executed:82.35% of 17
Creating 'test.c.gcov'

molly@molly-Huawei:~/test$ ls
test  test.c  test.c.gcov  test.gcda  test.gcno
```

# Gcov使用

```
-: 0:Source:test.c
-: 0:Graph:test.gcno
-: 0:Data:test.gcda
-: 0:Runs:1
-: 1:// test.c
-: 2:#include <stdio.h>
-: 3:#include <stdlib.h>
-: 4:
1: 5:void test(int count)
-: 6:{
-: 7:     int i;
10: 8:     for (i = 1; i < count; i++)
-: 9:     {
9: 10:         if (i % 3 == 0)
3: 11:             printf ("%d is divisible by 3\n", i);
9: 12:         if (i % 11 == 0)
#####: 13:             printf ("%d is divisible by 11\n", i);
9: 14:         if (i % 13 == 0)
#####: 15:             printf ("%d is divisible by 13\n", i);
-: 16:     }
1: 17:}
-: 18:
1: 19:int main(int argc, char *argv[])
-: 20:{
1: 21:     int i = 0;
1: 22:     if(argc == 2)
#####: 23:         i = atoi(argv[1]);
-: 24:     else
1: 25:         i = 10;
-: 26:
1: 27:     printf("arg is %d\n", i);
-: 28:
1: 29:     test(i);
-: 30:
1: 31:     return EXIT_SUCCESS;
-: 32:}
```



# LCOV图形化显示

1. **lcov -d . -t 'Test Coverage' -o 'test.info' -b . -c**

————→ test.info

2. **genhtml -o result test.info**

————→ 生成result文件夹，其中包含index.html

# LCOV图形化显示



## *LCOV - code coverage report*

Current view: [top level](#)

Test: [test.info](#)

Date: 2021-03-30 21:36:43

	Hit	Total	Coverage
Lines:	14	17	82.4 %
Functions:	2	2	100.0 %

Directory	Line Coverage	Functions
<a href="#">test</a>	<div><div></div></div> 82.4 % 14 / 17	100.0 % 2 / 2

Generated by: [LCOV version 1.14](#)

# LCOV图形化显示



## LCOV - code coverage report

Current view: [top level](#) - [test](#) - [test.c](#) ([source](#) / [functions](#))

Test: [test.info](#)

Date: 2021-03-30 21:36:43

	Hit	Total	Coverage
Lines:	14	17	82.4 %
Functions:	2	2	100.0 %

Line data	Source code
1	: // test.c
2	: #include <stdio.h>
3	: #include <stdlib.h>
4	:
5	1 : void test(int count)
6	: {
7	:     int i;
8	10 :     for (i = 1; i < count; i++)
9	:     {
10	9 :         if (i % 3 == 0)
11	3 :             printf ("%d is divisible by 3\n", i);
12	9 :             if (i % 11 == 0)
13	0 :                 printf ("%d is divisible by 11\n", i);
14	9 :             if (i % 13 == 0)
15	0 :                 printf ("%d is divisible by 13\n", i);
16	:         }
17	1 :     }
18	:
19	1 : int main(int argc, char *argv[])
20	: {
21	1 :     int i = 0;
22	1 :     if(argc == 2)
23	0 :         i = atoi(argv[1]);
24	:     else
25	1 :         i = 10;
26	:
27	1 :     printf("arg is %d\n", i);
28	:
29	1 :     test(i);
30	:
31	1 :     return EXIT_SUCCESS;
32	: }

# Linux Perf 简介

- Perf
  - Linux内核自带， profiling工具， **支持CPU PMU事件和软件事件计数**
  - 用于找到程序热点， 找到程序的性能瓶颈（cpu计算能力， cache利用率， 内存访问， 硬盘访问）

`perf list [hw|sw|cache|tracepoint|pmu|...]`

- 硬件： instructions, cycles, cache, branch
- 软件： page faults, context swith

# Perf 使用

- **perf top**

- 实时动态查看系统热点，函数热点，汇编热点

- **perf stat**

- 统计单个应用程序整个生命周期的硬软件事件数目。

- **perf record & perf report**

- 统计程序各模块、函数在整个生命周期中的性能数据

# perf top

```
Samples: 33K of event 'cycles', 4000 Hz, Event count (approx.): 3667127735 lost: 0/0 drop
Overhead Shared Object Symbol
21.93% [kernel] [k] 0xffffffff8557f7b3
21.54% [kernel] [k] 0xffffffff852bb25f
5.26% [kernel] [k] 0xffffffffc00681eb
5.24% [kernel] [k] 0xffffffffc0068204
3.75% [kernel] [k] 0xffffffffc01acabc
1.11% [kernel] [k] 0xffffffffc12f337f
1.05% [kernel] [k] 0xffffffffc0068449
1.01% [unknown] [k] 0xffffffffc0000000
0.57% [kernel] [k] 0xffffffff8530c40a
0.40% [kernel] [k] 0xffffffff8557f792
0.37% perf [.] rb_next
0.31% [kernel] [k] 0xffffffff8557fd53
0.25% libglib-2.0.so.0.6400.6 [.] g_hash_table_lookup
0.25% perf [.] hists__findnew_entry
0.25% [kernel] [k] 0xffffffff85600fe7
0.25% perf [.] hpp__sort_overhead
0.23% perf [.] perf_hpp__is_dynamic_entry
0.22% perf [.] hist_entry__sort
0.18% libc-2.31.so [.] _int_malloc
0.18% libc-2.31.so [.] _int_free
0.18% perf [.] output_resort
0.18% [kernel] [k] 0xffffffff85173fac
0.17% [kernel] [k] 0xffffffff8557fdbb
0.15% [kernel] [k] 0xffffffff84a6c6c9
0.14% [kernel] [k] 0xffffffff84afd152
```

perf top查看整个系统

# perf top

```
Samples: 6K of event 'cycles', 4000 Hz, Event count (approx.): 235309873 lost: 0/0 drop: 0/5423
Overhead Shared Object Symbol
2.97% [unknown] [.] 0000000000000000
1.85% firefox [.] 0x0000000000000e863
1.78% libpthread-2.31.so [.] __pthread_mutex_unlock
1.40% libpthread-2.31.so [.] __pthread_mutex_lock
1.29% [kernel] [k] 0xfffffffffa3abb25f
0.98% [kernel] [k] 0xfffffffffa32fd152
0.95% ld-2.31.so [.] __tls_get_addr
0.90% libpthread-2.31.so [.] pthread_cond_wait@@GLIBC_2.3.2
0.80% [vdso] [.] __vdso_clock_gettime
0.72% firefox [.] 0x00000000000013266
0.69% libpthread-2.31.so [.] pthread_cond_timedwait@@GLIBC_2.3.2
0.68% [kernel] [k] 0xfffffffffa3d7fdb
0.63% firefox [.] mozilla::BaseTimeDurationPlatformUtils::ToSeconds
0.57% ld-2.31.so [.] _dl_update_slotinfo
0.56% firefox [.] 0x000000000000dafd
0.51% firefox [.] 0x0000000000001323b
0.50% libc-2.31.so [.] clock_gettime@GLIBC_2.2.5
0.49% [kernel] [k] 0xfffffffffa323be75
0.49% [kernel] [k] 0xfffffffffa3411db9
0.49% firefox [.] 0x00000000000013a70
0.48% [kernel] [k] 0xfffffffffc024c1eb
0.48% libxul.so [.] 0x00000000000f8b520
```

perf top -pid xxxx  
查看单个应用程序

# perf top

Samples: 513K of event 'cycles', 2250 Hz, Event count (approx.): 17952307867 lost: 0/0 drop: 0/20694

Children	Self	Shared Object	Symbol
+ 42.42%	0.00%	[unknown]	[k] 0xfffffffffa32db8a0
+ 39.84%	0.00%	[unknown]	[k] 0xfffffffffa32db677
+ 39.83%	0.00%	[unknown]	[k] 0xfffffffffa32db3b3
+ 39.82%	0.00%	[unknown]	[k] 0xfffffffffa3b0c76e
+ 22.70%	0.02%	[unknown]	[k] 0xfffffffffa3b0c3c2
+ 22.30%	0.06%	[unknown]	[k] 0xfffffffffa3d7f7bc
+ 21.93%	21.93%	[unknown]	[k] 0xfffffffffa3d7f7b3
16.90%	0.30%		a
16.12%	0.00%		f
15.03%	0.00%		c
14.65%	0.00%		1
13.76%	0.00%		3
+ 9.29%	0.00%	[unknown]	[k] 0xfffffffffa397258f
+ 9.29%	0.02%	[unknown]	[k] 0xfffffffffa3abb261
9.24%	9.24%	[unknown]	[k] 0xfffffffffa3abb25f
+ 9.15%	0.41%	perf	[.] output_resort
+ 8.74%	0.78%	[unknown]	[k] 0000000000000000
6.96%	0.00%	perf	[.] process_thread
+ 6.96%	0.02%	perf	[.] __ordered_events__flush.part.0
+ 6.92%	0.00%	[unknown]	[k] 0xfffffffffa32000e6
+ 6.92%	0.05%	perf	[.] deliver_event
+ 6.79%	0.00%	[unknown]	[k] 0xfffffffffa3263699
+ 6.65%	0.04%	perf	[.] hist_entry_iter__add
- 5.07%	0.34%	perf	[.] iter_add_next_cumulative_entry
4.74%	4.71%	perf	[.] rb_next
4.55%	0.01%	[unknown]	[.] 0xfffffffffa3e0008c
4.42%	0.77%	perf	[.] hist_entry__sort
4.34%	0.03%	perf	[.] hists__decay_entries
4.23%	0.02%	[unknown]	[.] 0xfffffffffa3d6dbb9
3.36%	0.26%	perf	[.] hists__decay_entry



# perf top

Annotate iter\_add\_next\_cumulative\_entry

Zoom into perf DSO (use the 'k' hotkey to zoom directly into the kernel)

Expand [iter\_add\_next\_cumulative\_entry] callchain (one level, same as '+' hotkey, use 'e'/'c' for the whole main level entry)

Browse map details

Exit

1. Annotate ...查看函数的反汇编，及在汇编级的热点
2. Zoom into perf DSO 查看当前DSO的热点情况
3. Expand callchain 展开一级调用关系
4. 函数符号的内存映射情况

# perf top

Annotate iter\_add\_next\_cumulative  
Zoom into perf DSO (use the 'k'  
Expand [iter\_add\_next\_cumulative  
Browse map details  
Exit

1. Annotate ... 查看函数
2. Zoom into perf DSO
3. Expand callchain 展开
4. 函数符号的内存映射

0.18

0.18

0.16

0.53

0.28

28.21

0.47

0.27

0.08

```
mov    %r12, -0x2a0(%rbp)
movdqa %xmm1, %xmm0
shufpd $0x1, %xmm2, %xmm0
mov    %fs:0x28, %rax
mov    %rax, -0x38(%rbp)
xor    %eax, %eax
mov    0x30(%rdi), %rax
mov    %r15, %rdi
mov    %r12, -0x298(%rbp)
movaps %xmm1, -0x280(%rbp)
mov    %rax, -0x268(%rbp)
xor    %eax, %eax
rep    stos %rax, %es:(%rdi)
mov    (%rsi), %rdi
movaps %xmm0, -0x1b0(%rbp)
movdqa %xmm2, %xmm0
shufpd $0x1, %xmm1, %xmm0
movaps %xmm2, -0x290(%rbp)
movaps %xmm0, -0x1a0(%rbp)
→callq thread__comm
mov    0x28(%r13), %r10
```

# perf top

Samples: 75K of event 'cycles', 2250 Hz, Event count (approx.): 8752932			
	Children	Self	Symbol
Annotate iter_add_next			
Zoom into perf DSO (use	+ 9.94%	0.03%	[.] ordered_events_flush.part.0
Expand [iter_add_next_c	+ 9.69%	0.05%	[.] deliver_event
Browse map details	+ 9.39%	0.03%	[.] hist_entry_iter__add
Exit	+ 7.14%	0.50%	[.] iter_add_next_cumulative_entry
	+ 4.12%	3.57%	[.] append_chain_children
	+ 3.74%	0.28%	[.] __hists__add_entry.constprop.0
	+ 3.32%	0.04%	[.] callchain_append
	+ 2.99%	0.13%	[.] output_resort
	+ 2.91%	1.35%	[.] hists__findnew_entry
	+ 1.80%	0.25%	[.] hists__collapse_resort
	+ 1.76%	0.00%	[.] process_thread
	1.68%	0.01%	[.] hists__decay_entries
	1.63%	1.63%	[.] rb_next
	1.32%	0.18%	[.] __sort_chain_graph_abs
	1.30%	0.01%	[.] sort_chain_graph_abs
	+ 1.28%	0.00%	[.] display_thread_tui
	+ 1.28%	0.00%	[.] perf_evlist__tui_browse_hists
	+ 1.28%	0.00%	[.] perf_evsel__hists_browse

1. Annotate ... 查找
2. Zoom into per
3. Expand callcha
4. 函数符号的内

# perf top

Annotate iter\_add\_next\_cumulative  
Zoom into perf DSO (use the 'k'  
Expand [iter\_add\_next\_cumulative  
Browse map details  
Exit

```
/usr/lib/linux-hwe-5.8-tools-5.8.0-48/perf
1c6000 1c7bd0 l init
1c6030 1c6040 g log10@plt
1c6040 1c6050 g gelf_getsym@plt
1c6050 1c6060 g ctime@plt
1c6060 1c6070 g gelf_getrela@plt
1c6070 1c6080 g dwarf_decl_file@plt
1c6080 1c6090 g mount@plt
1c6090 1c60a0 g symlink@plt
1c60a0 1c60b0 g tcsetattr@plt
1c60b0 1c60c0 g _Ux86_64_set_caching_policy@plt
1c60c0 1c60d0 g sem_wait@plt
1c60d0 1c60e0 g chdir@plt
1c60e0 1c60f0 g fileno@plt
1c60f0 1c6100 g dirname@plt
1c6100 1c6110 g dwfl_report_begin@plt
1c6110 1c6120 g dup2@plt
1c6120 1c6130 g pthread_cond_destroy@plt
1c6130 1c6140 g printf@plt
1c6140 1c6150 g gelf_fsize@plt
1c6150 1c6160 g __getdelim@plt
1c6160 1c6170 g dwarf_diename@plt
1c6170 1c6180 g strcasestr@plt
1c6180 1c6190 g memset@plt
1c6190 1c61a0 g elf_strptr@plt
1c61a0 1c61b0 g dwarf_lineendsequence@plt
1c61b0 1c61c0 g snprintf@plt
1c61c0 1c61d0 g mmap64@plt
1c61d0 1c61e0 g fmemopen@plt
```

main level entry)

1. Annotate ... 查看函数
2. Zoom into perf DSO
3. Expand callchain 展开
4. 函数符号的内存映射

# perf stat

- `sudo perf stat dd if=/dev/zero of=/dev/null count=1000000`

```
molly@molly-Huawei:~$ sudo perf stat dd if=/dev/zero of=/dev/null count=1000000
```

记录了1000000+0 的读入

记录了1000000+0 的写出

512000000字节 (512 MB, 488 MiB) 已复制, 0.333274 s, 1.5 GB/s

Performance counter stats for 'dd if=/dev/zero of=/dev/null count=1000000':

333.68 msec	task-clock	#	0.999 CPUs utilized
2	context-switches	#	0.006 K/sec
0	cpu-migrations	#	0.000 K/sec
82	page-faults	#	0.246 K/sec
1,176,133,823	cycles	#	3.525 GHz
2,137,561,864	instructions	#	1.82 insn per cycle
411,308,952	branches	#	1232.653 M/sec
4,740,534	branch-misses	#	1.15% of all branches

0.333987355 seconds time elapsed

0.120717000 seconds user

0.213267000 seconds sys

## -e 指定事件

```
perf stat -e cycles dd if=/dev/zero of=/dev/null count=100000
```

```
perf stat -e instructions dd if=/dev/zero of=/dev/null count=100000
```

# perf record & report

## 1. 编译程序 (加-g)

```
gcc -g -O0 test.c -o test
```

## 2. perf record

```
perf record -a -g ./test
```

生成perf.data

## 3. perf report

# perf record & report (例子)

```
#include <stdio.h>

void test_little(void)
{
    int i,j;
    for(i = 0; i < 30000000; i++)
        j=i;
}

void test_mdedium(void)
{
    int i,j;
    for(i = 0; i < 60000000; i++)
        j=i;
}

void test_high(void)
{
    int i,j;
    for(i = 0; i < 90000000; i++)
        j=i;
}

void test_hi(void)
{
    int i,j;
    for(i = 0; i < 120000000; i++)
        j=i;
}
```

test.c 文件

```
int main(void)
{
    int i, pid, result;

    for(i = 0; i<2; i++) {
        result = fork();
        if(result>0)
            printf("i=%d parent parent=%d current=%d child=%d\n", i, getppid(), getpid(), result);
        else
            printf("i=%d child parent=%d current=%d\n", i, getppid(), getpid());

        if(i==0)
        {
            test_little();
            sleep(1);
        } else {
            test_mdedium();
            sleep(1);
        }
    }

    pid = wait(NULL);
    test_high();
    printf("pid=%d wait=%d\n", getpid(), pid);
    sleep(1);
    pid = wait(NULL);
    test_hi();
    printf("pid=%d wait=%d\n", getpid(), pid);
    return 0;
}
```

# perf record & report (例子)

Samples: 31K of event 'cycles', Event count (approx.): 10662567324					
	Children	Self	Command	Shared Object	Symbol
+	73.35%	0.00%	test	libc-2.31.so	[.] libc_start_main
+	73.34%	0.00%	test	test	[.] main
+	30.22%	30.21%	test	test	[.] test_hi
+	22.72%	22.65%	test	test	[.] test_high
+	16.60%	16.56%	test	test	[.] test_mdedium
+	10.96%	0.00%	swapper	[unknown]	[k] 0xffffffffb9c000e6
+	10.96%	0.00%	swapper	[unknown]	[k] 0xffffffffb9cdb8a0
+	9.66%	0.00%	swapper	[unknown]	[k] 0xffffffffb9cdb677
+	9.65%	0.00%	swapper	[unknown]	[k] 0xffffffffb9cdb3b3
+	9.65%	0.00%	swapper	[unknown]	[k] 0xffffffffba50c76e
+	9.24%	0.00%	swapper	[unknown]	[k] 0xffffffffb9c63699
+	8.39%	0.01%	swapper	[unknown]	[k] 0xffffffffba50c3c2
+	8.24%	0.02%	swapper	[unknown]	[k] 0xffffffffba77f7bc
+	8.01%	8.01%	swapper	[unknown]	[k] 0xffffffffba77f7b3
+	3.79%	3.79%	test	test	[.] test_little
+	3.12%	0.00%	chrome	libc-2.31.so	[.] __libc_start_main
+	3.12%	0.01%	chrome	chrome	[.] free
+	3.11%	0.00%	chrome	chrome	[.] 0x00005599abd5a4ca
+	3.11%	0.00%	chrome	chrome	[.] 0x00005599abd5c0f0
+	3.11%	0.00%	chrome	chrome	[.] 0x00005599b0f669f7
+	3.11%	0.00%	chrome	chrome	[.] 0x00005599abdefe00
+	3.11%	0.00%	chrome	chrome	[.] 0x00005599abe1b09c
+	3.06%	0.00%	chrome	chrome	[.] 0x00005599abdc8bb8
+	2.87%	0.00%	chrome	chrome	[.] 0x00005599abe1a504
+	2.86%	0.00%	chrome	chrome	[.] 0x00005599abe074a8
+	2.50%	0.00%	chrome	chrome	[.] 0x00005599ad310292
+	1.72%	0.00%	swapper	[unknown]	[k] 0xffffffffbb4c2597



# perf record & report (例子)

Samples: 31K of event 'cycles', Event count (approx.): 10662567324					
	Children	Self	Command	Shared Object	Symbol
+	73.35%	0.00%	test	libc-2.31.so	[.] __libc_start_main
+	73.34%	0.00%	test	test	[.] main
-	30.22%	30.21%	test	test	[.] test_hi
	30.21%		__libc_start_main		
			main		
			test_hi		
+	22.72%	22.65%	test	test	[.] test_high
+	16.60%	16.56%	test	test	[.] test_mdedium
+	10.96%	0.00%	swapper	[unknown]	[k] 0xfffffffffb9c000e6

# perf record & report (例子)

Annotate test\_hi

Zoom into test thread

Zoom into test DSO (use the 'k' hotkey to zoom directly into the kernel)

Collapse [test\_hi] callchain (one level, same as '+' hotkey, use 'e'/'c' for the whole main level entry)

Browse map details

Run scripts for samples of symbol [test\_hi]

Run scripts for all samples

Switch to another data file in PWD

Exit

# perf record & report (例子)

```
Samples: 31K of event 'cycles', 4000 Hz, Event count (approx.): 10662567324
test_hi /home/molly/test/test-perf/test [Percent: local period]
Percent
```

Percent	Disassembly of section .text:
	0000000000001261 <test_hi>: test_hi(): for(i = 0; i < 90000000; i++) j=i; }
	void test_hi(void) { endbr64 push  %rbp mov  %rsp,%rbp int i,j;  for(i = 0; i < 120000000; i++) movl  \$0x0,-0x8(%rbp) ↓ jmp  1b j=i; 11:  mov  -0x8(%rbp),%eax mov  %eax,-0x4(%rbp) for(i = 0; i < 120000000; i++) addl  \$0x1,-0x8(%rbp) 47.67 1b:  cmpl  \$0x7270dff,-0x8(%rbp) 14.31  ↑ jle  11 } nop nop pop  %rbp ←retq

# More perf commands

- perf bench
  - perf中内置的benchmark, 目前包括两套针对调度器和内存管理子系统的benchmark。
- perf mem
  - 内存存取情况
- perf timechart
  - 针对测试期间系统行为进行可视化的工具
- .....

# 参考资料

- gcov—a Test Coverage Program  
<https://gcc.gnu.org/onlinedocs/gcc/Gcov.html>
- Writing Better Function Tests with GCOV  
[https://www.youtube.com/watch?v=U\\_qWLa9KnW8](https://www.youtube.com/watch?v=U_qWLa9KnW8))
- PERF tutorial: Finding execution hot spots  
<http://sandsoftwaresound.net/perf/perf-tutorial-hot-spots/>
- perf Examples  
<http://www.brendangregg.com/perf.html>
- Performance profiling with perf  
<https://fedoramagazine.org/performance-profiling-perf/>
- Rapidly Selecting Good Compiler Optimizations using Performance Counters  
<https://ieeexplore.ieee.org/abstract/document/4145114>
- 系统级性能分析工具perf的介绍与使用  
<https://www.cnblogs.com/arnoldlu/p/6241297.html>

THE END

THANKS FOR WATCHING