Linux 的系统级性能剖析工具-perf

（二）

承刚

TAOBAO Kernel Team
chenggang.qin@gmail.com

第三章 Perf top 工具

3.1 perf top 的基本使用方法

top 工具主要用于实时剖析各个函数在某个性能事件上的热度。利用 perf top，能够直观地观察到当前的热点函数，并利用工具中内置的 annotate 功能，进一步查找热点指令。

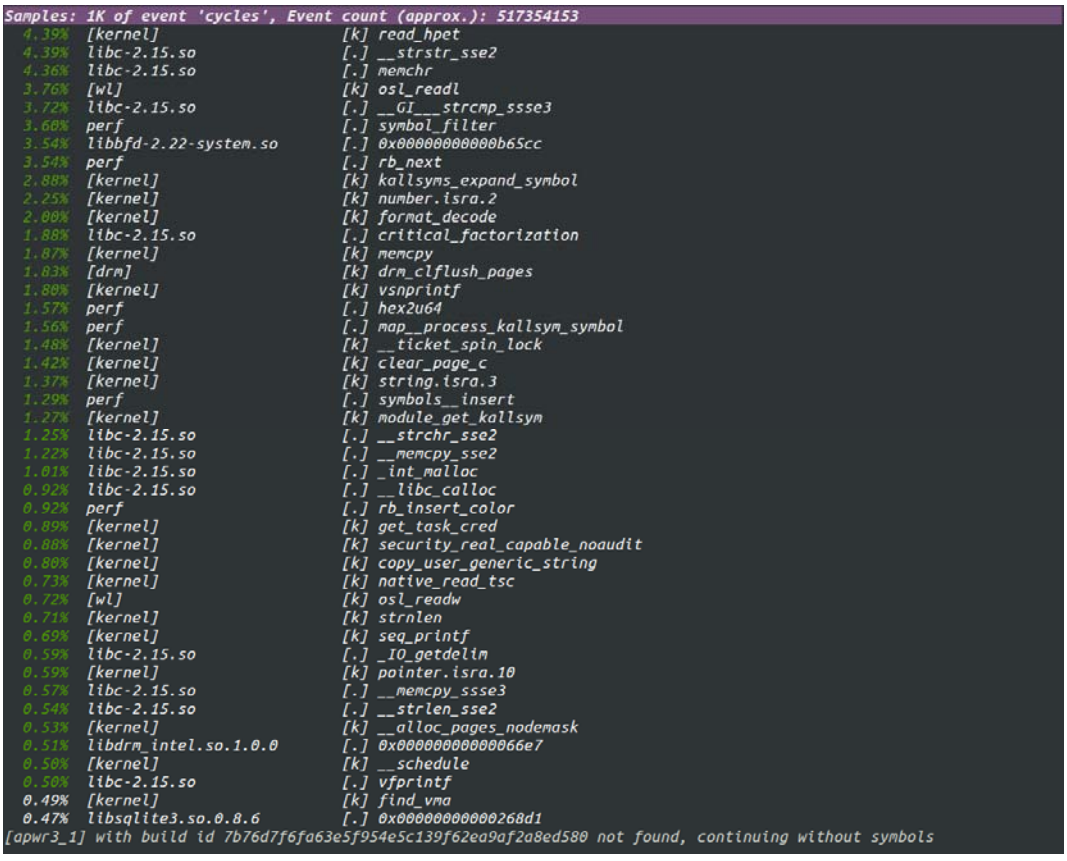

图 3. perf top 的界面

Perf top 的基本使用方法为：

$perf top

该命令以默认性能事件“cycles（CPU 周期数）”进行全系统的性能剖析，检测系统中的所有应用程序函数与内核函数的热度。图 3 为上述命令的执行结果。

perf 提供了 3 种用户界面，分别是 tui，gtk 以及 tty。其中可操作性最强，功能最丰富的界面是 tui，本文主要基于此界面讲解 perf。top 工具仅支持 tui 与 tty，默认界面为 tui。图 3 中展示的就是 tui 界面。

top 工具的界面具有 4 列信息。右面第一列为符号名，也就是函数名。左面第一列为该符号引发的性能事件在整个监测域中占的比例，我们将其称为该符号的热度。监测域是指 perf 监控的所有符号。默认情况下包括系统中所有进程、内核以及内核模块的函数。左面第二列为该符号所在的 DSO。DSO 即动态共享对象（Dynamic Shared Object）的缩写。第 3 列为 DSO 的类型。perf 中 DSO 共有 5 种类型，分别是：ELF 可执行文件，动态链接库，内核，内核模块，VDSO 等。当第 3 列为 [.]时表示此符号属于用户态的 ELF 文件（包括可执行文件与动态链接库）。为[k]表示此符号属于内核或内核模块。

需要提一下的是，必须在系统安装 newt 软件包，perf 才能使能 tui 界面。

在 tui 界面下按'h'，'?'或'F1'键时，会弹出一个帮助窗口，如图 4 所示。

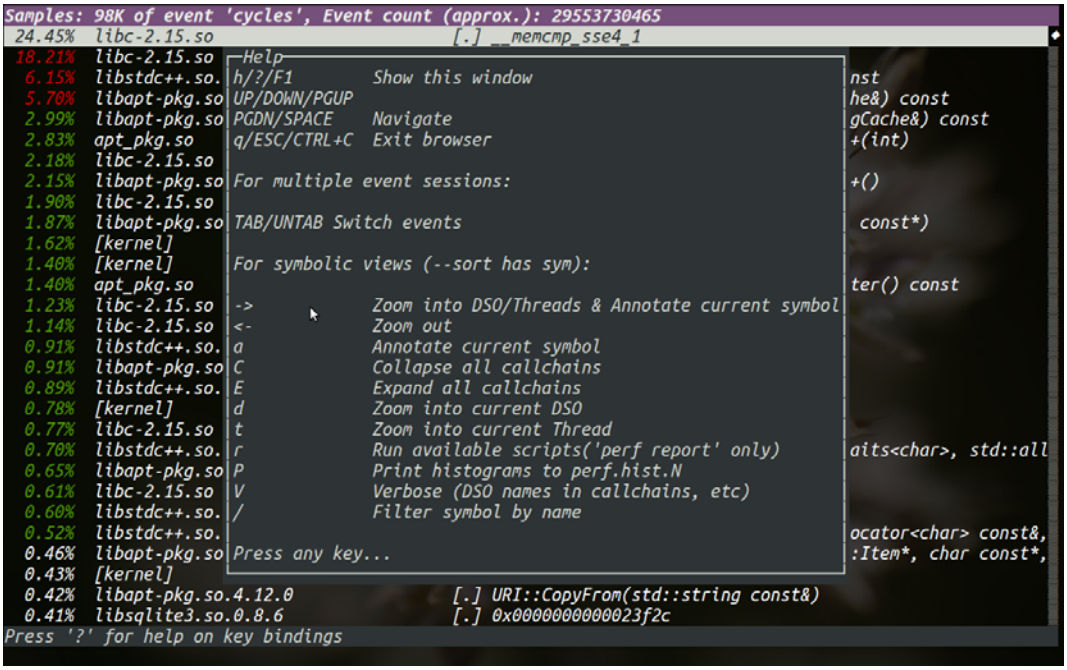

图 4. tui 界面的帮助窗口

帮助窗口列出了 perf top 的所有功能。我们首先来看注解（Annotate）功能。注解功能可以进一步深入分析某个符号。给出对应线程的代码并且为热点代码标记出它们触发的性能事件在整个测试域中的百分比。下面让我们来看一下如何使用注解功能。在界面上按上下键，将光标在各个 symbol 间移动。选定某个符号后，按下 a 键，得到如图 5 所示的界面。

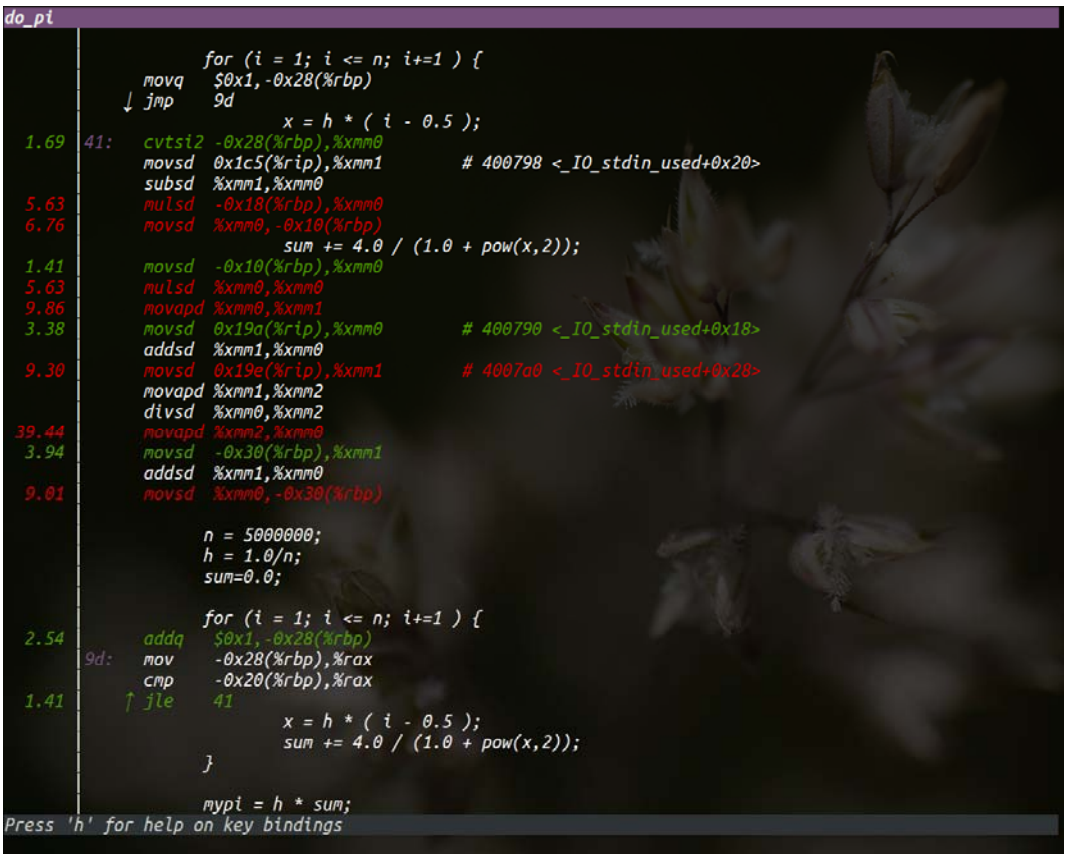

图 5. perf top 的注解功能

图 5 显示的是[code1]中 do_pi()函数的注解。从图上可以看到 perf 对 do_pi()中的 C 代码给出了汇编语言的注解。并且给出了各条指令的采样率。从图上可以看到耗时较多的指令（如访存指令）比较容易被 perf 采到。

选定某个符号后按下热键'd'，perf 会过滤掉所有不属于此 DSO 的文件。以图 1 中的实验为例，符号 do_pi 归属的 DSO 为 thread。当在符号 do_pi 上按下'd'键后，perf 过滤掉了所有不属于 thread 的符号。如图 6 所示。

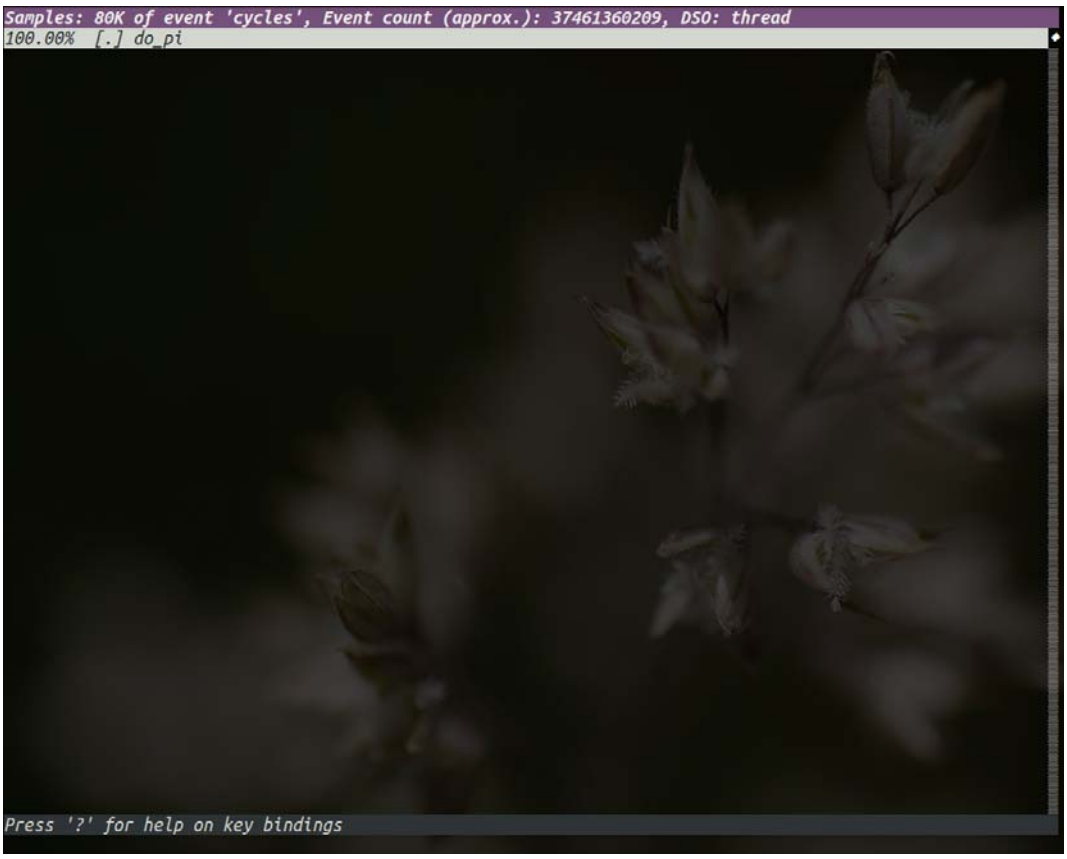

图 6. 符号的 DSO 级过滤功能

类似于热键'd'，热键't'能够过滤所有不属于当前符号所属线程的符号。

热键'P'可以将 perf top 的当前显示的信息输出到文件'perf.hist.XXX'中。

右方向键也是热键，它可以打开 perf top 的功能菜单。菜单上列出的各项功能分别对应上述各个热键的功能。

3.2 perf top 的参数介绍

Perf top 的参数较多，本文只介绍几个常用参数的使用方法。

'-e' or '--event' <event>:

该参数用于指定待分析的性能事件。具体可以参考第 3 节 perf list。如果我们希望利用 top 工具剖析系统中的 Cache 丢失次数，可以采用以下命令:

```
$perf top -e cache-misses
```

'-c' or '--count' <n>

该参数用于指定性能计数器的采样周期。默认情况下，每秒钟采样 4000 次。

‘-p’ or ‘--pid’ <pid>

该参数用于指定待分析进程的 pid。指定 pid 后，perf top 仅仅分析目标进程以及目标进程创建的线程。

‘-t’ or ‘--tid’ <tid>

该参数用于指定带分析线程的 tid。指定 tid 后，perf top 仅分析目标线程，不包括此线程创建的其它线程。

‘-a’ or ‘--all-cpus’

采集系统中所有 CPU 产生的性能事件。这也是 perf top 的默认情况。

‘-C’ or ‘--cpu’ <cpu>

指定待分析的 CPU 列表。如系统中有 4 个 CPU，如果仅需采集 CPU0 与 CPU3 的数据，可通过如下方法调用 perf top：

```
$perf top -C 0,3
```

‘-k’ or ‘--vmlinux’ <file>

指定带符号表的内核映像所在的路径。与 GDB 类似，perf 只有在 DSO 存在符号表的情况下才能解析出 IP 对应的具体符号。Perf 通常采用以下顺序加载内核符号：

1. /proc/kallsyms
2. 用户通过’-k’参数指定的路径。
3. 当前路径下的“vmlinux”文件
4. /boot/vmlinux
5. /boot/vmlinux-$(uts.release)
6. /lib/modules/$(uts.release)/build/vmlinux
7. /usr/lib/debug/lib/modules/$(uts.release)/build/vmlinux

‘-K’ or ‘--hide_kernel_symbols’

不显示属于内核的符号。对于只想分析应用程序的用户而言，使用此参数后，perf top 的界面会清爽很多。

‘-m’ or ‘--mmap-pages’ <n>

指定 perf 开辟的 mmap 页面的数量。mmap 缓存主要用于用户空间与内核空间的数据通信。Perf 在内核中的驱动将采集到的性能数据存入 ring buffer，用户

空间的分析程序则通过 mmap 机制从 ring buffer 中读取数据。

默认情况下,mmap 的页面数量为 128,此时 mmap 内存区域的大小为 512KB。perf top 默认每隔 2s 读取一次性能数据，当内核生成性能数据的速度过快时，就可能因为缓冲区满而导致数据丢失，从而影响到分析结果。在此情况下，用户可以通过'-m'参数扩大缓冲区，以避免性能数据的大量丢失。

'-r', '--realtime' <n>

指定分析程序的实时优先级。如上文所述，如果 perf 分析程序读取数据的速度长期小于内核生成数据的速度时，就可能导致采样数据的大量丢失，影响分析精度。在系统负载过高时，分析程序可能会因为调度延迟过高而不能及时读取数据。因此，在高负载系统中可以通过参数'-r'将分析程序设置为实时进程，并为其设定较高的优先级。

```
$perf top -r 0
```

上述命令中，0 即为指定给分析程序的实时优先级。顺便说一句，Linux 中实时优先级的范围是[0,99]，其中优先级 0 最高。不要与范围是[-20,19]的 nice 值搞混。

'-d' or '--delay' <n>

指定 perf top 界面的刷新周期，<n>的单位为秒。默认值为 2s。

'-D' or '--dump-symtab'

打印 DSO 的符号表，此功能主要用于 perf 自身的调试。该参数仅与'--stdio'（即 TTY 界面）配合使用时才起作用。启用此参数后，perf top 会在退出时打印所有 DSO 的符号表，如图 7 所示。

```
$perf top --stdio –D
```

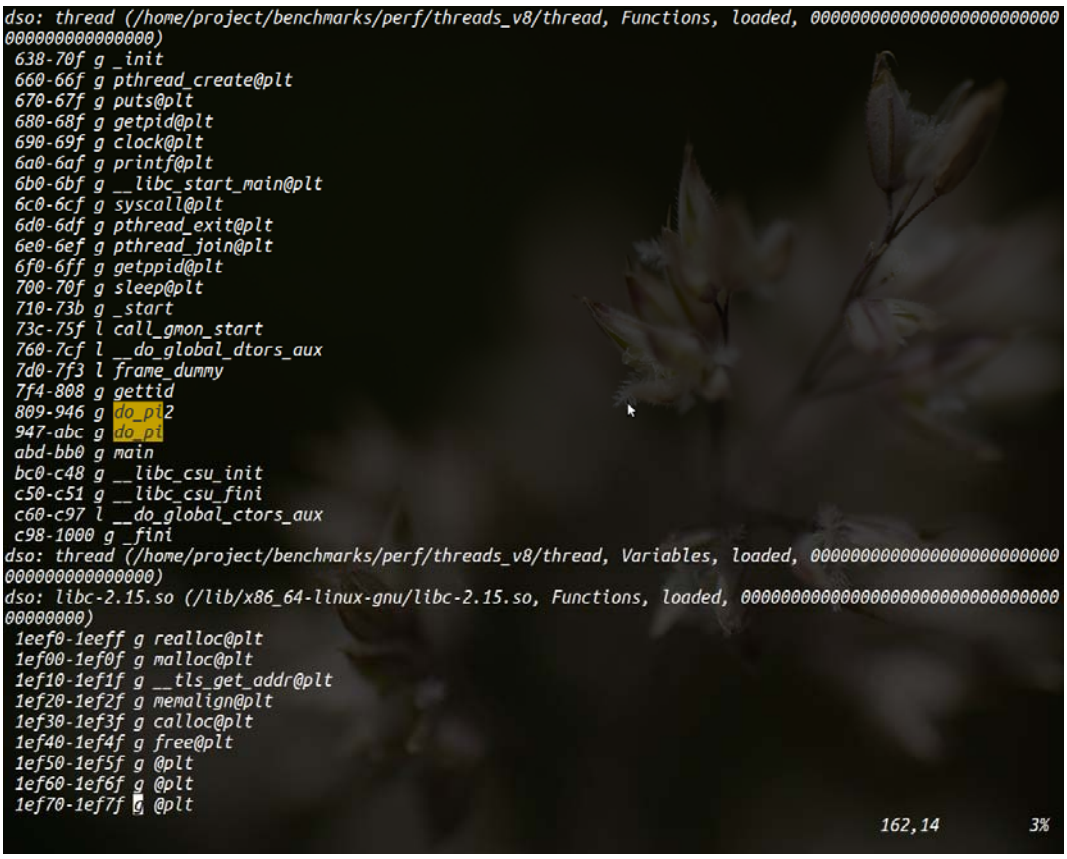

图 7. perf top 打印出的符号表

图 7 给出了 perf top 输出的符号表的片段，这些符号来自于 ELF 文件'thread'的 symtable section。符号表的第一列为该符号的地址范围。第二列表示该符号的属性，如'g'表示全局符号，'l'表示局部符号等。

'-f' or '--count-filter' <n>

此参数主要用于符号过滤。指定此参数后，界面上将仅显示采样数大于<n>的符号。

'-g' or '--group'

将计数器组合成计数器组。Perf 在默认情况下会为每个计数器创建一个独立的 ring buffer。如果将计数器组合成计数器组，perf 则会将所有 counter 的数据输出到 group leader 的 ring buffer 中。当内核生成采样数据不多并且内存较为紧张时，可采用此参数以节省内存。

'-i' or '--inherit'

采用此参数后，子进程将自动继承父进程的性能事件。从而使得 perf 能够采集到动态创建的进程的性能数据。但是，当采用'-p'参数仅分析特定进程的性能数据时，继承机制会被禁用。这主要是出于性能的考虑。

'--sym-annotate' <symbol name>

指定待解析的符号名。

'-z' or '--zero'

更新界面的数据后，清除历史信息。

'-F' or '--freq' <n>

指定采样频率，此参数与'-c'参数指定一个即可。当两个参数同时指定时，perf 仅使用'-c'指定的值作为采样周期。

'-E' or '--entries' <n>

指定界面上显示的符号数。如果用户仅希望查看 top <n>个符号，可以通过此参数实现。

'-U' or '--hide_user_symbols'

仅显示属于内核的符号，隐藏属于用户程序的符号，即类型为[.]的符号。

'--tui'

使用 tui 界面。tui 为 perf top 的默认界面。如前所述，tui 界面需要 newt 软件包的支持。如果用户打开 perf top 后，出现的不是 tui 界面，请检查系统中是否已安装 newt 包。

'--stdio'

使用 TTY 界面。当系统中未装 newt 软件包时，此界面为默认界面。界面风格如图 8 所示。

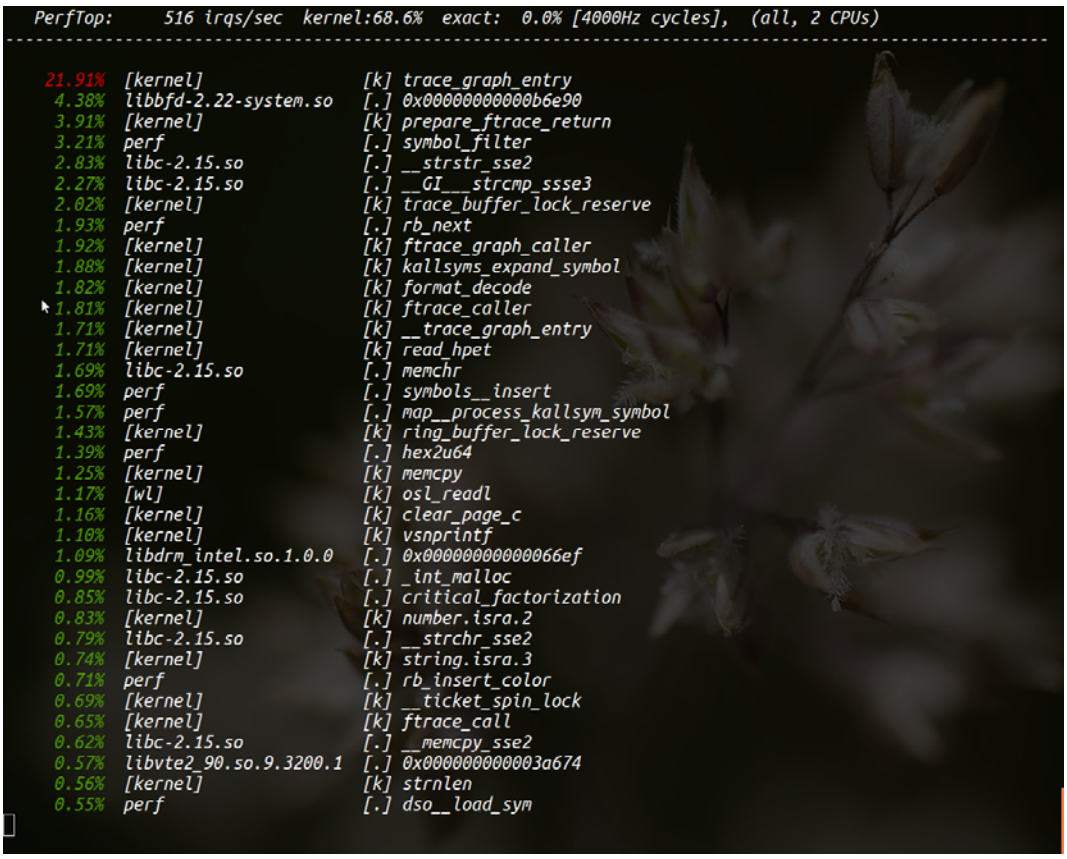

图 8. Perf top 的 stdio 界面

‘-v’ or ‘--verbose’

显示冗余信息，如符号地址、符号类型（全局、局部、用户、内核等）。

‘-s’ or ‘--sort’ <key[,key2…>

指定界面显示的信息，以及这些信息的排序。Perf 提供的备选信息有：

表 4. Perf top 界面支持的信息

Comm	触发事件的进程名
PID	触发事件的进程号
DSO	符号所属的 DSO 的名称
Symbol	符号名
Parent	调用路径的入口

当采用如下命令时，perf top 会给出更加丰富的信息。

```
$perf top -s comm,pid,dso,parent,symbol
```

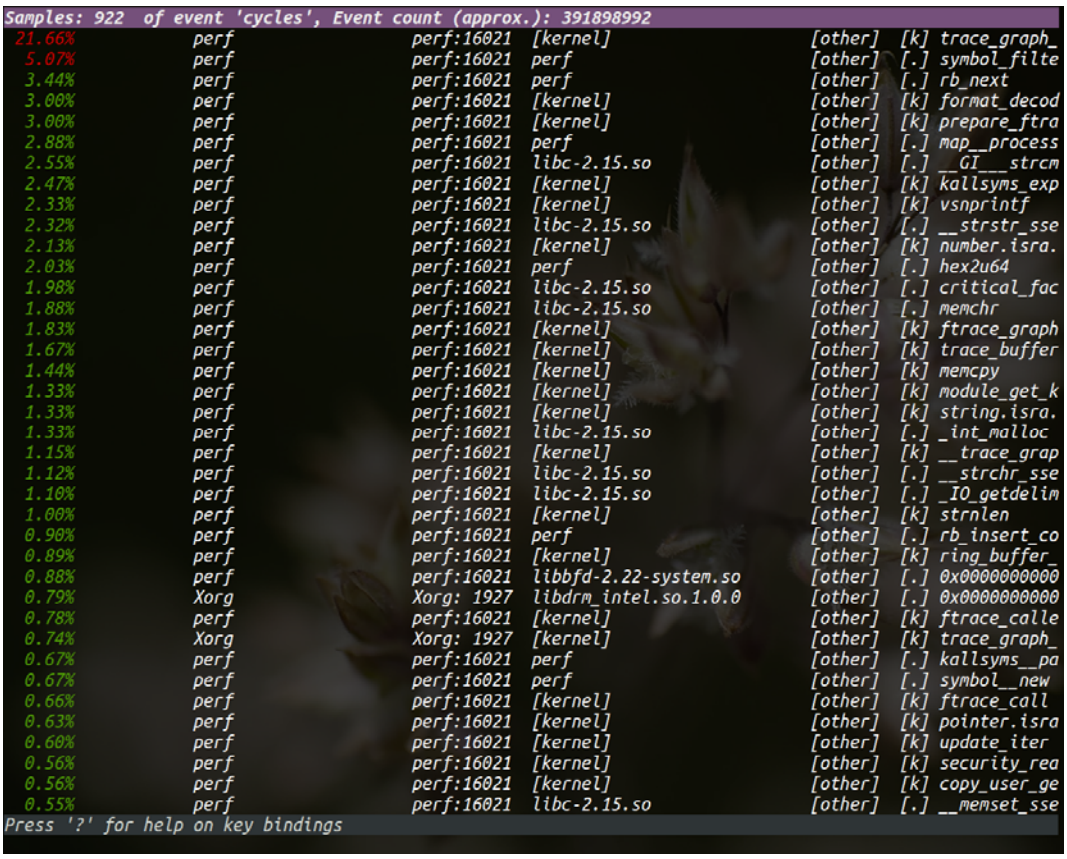

图 9. Perf top --sort 的界面

从图 9 上可以查看更丰富的信息，各行信息的意义与上述命令中个字段的排序相对应。

‘-n’ or ‘--show-nr-samples’

显示每个符号对应的采样数量。

如下命令对应的界面如图 10 所示。

```
$perf top –n
```

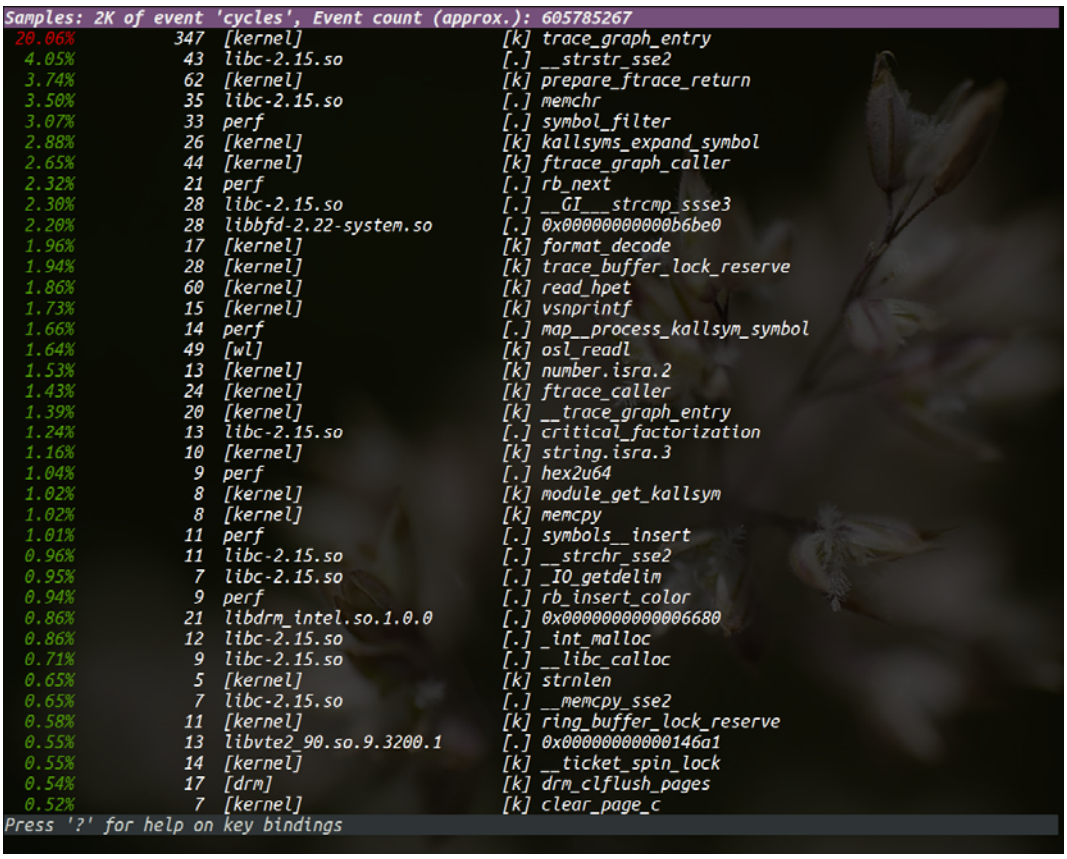

图 10. perf top -n 的界面

图 10 上的第二列为本行的符号在本轮分析中对应的采样数量。

‘--show-total-period’

在界面上显示符号对应的性能事件总数。如第二节所述，性能事件计数器在溢出时才会触发一次采样。两次采样之间的计数值即为这段时间内发生的事件总数，perf 将其称之为周期（period）。

使用方法为：

```
$perf top --show-total-period
```

执行上述命令后，将得到如图 11 所示的结果。

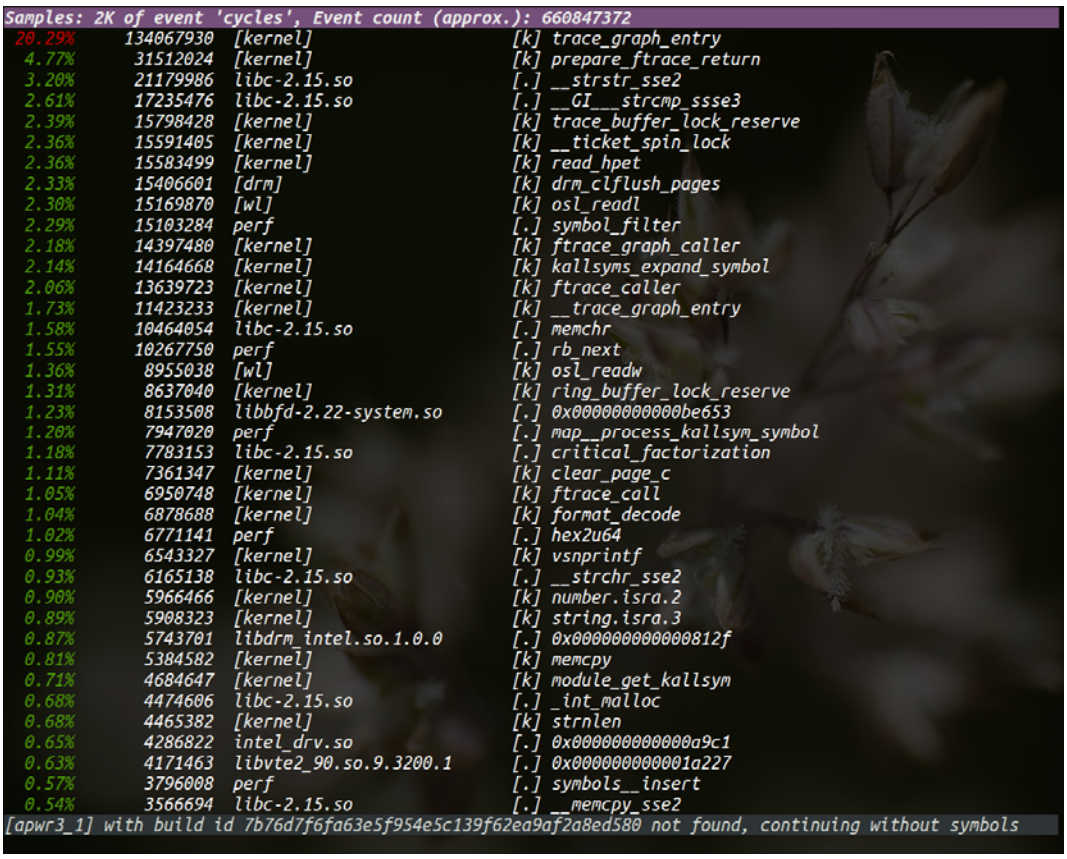

图 11. perf top --show-total-period 的界面

图 11 上的第二列即为该行符号在本轮分析中对应的事件总数。

‘-G’ or ‘--call-graph’ <output_type,min_percent,call_order>

在界面中显示函数的调用图谱。用户还可以指定显示类型，采样率的最小阈值，以及调用顺序。下面通过一个简单的程序观察一下 perf top 如何显示调用图谱。

Code2

```
#include <stdio.h>
#include <math.h>
#include <sys/types.h>
#include <linux/unistd.h>

int do_pi(int p){
        double mypi,h,sum,x;
        long long    n,i;

        n = 500000000000;
        h = 1.0/n;
        sum=0.0;

        for (i = 1; i <= n; i+=1 ) {
          x = h * ( i - 0.5 );
          sum += 4.0 / (1.0 + pow(x,2));
        }

        mypi = h * sum;

        return 0;
}

int bk()
{
        do_pi(0);
}

int ak()
{
          double mypi,h,sum,x;
          long long    n,i;

          n = 500000;
          h = 1.0/n;
          sum=0.0;

          for (i = 1; i <= n; i+=1 ) {
                    x = h * ( i - 0.5 );
                    sum += 4.0 / (1.0 + pow(x,2));
          }

          mypi = h * sum;
```

```
        bk();
}

int b()
{
        do_pi(1);
}

int a()
{
        b();
}

int main()
{
        pid_t pid;
        printf("pid: %d\n", getpid());

        pid = fork();

        if (pid == 0) {
          ak();
        }

        a();

        return 0;
}
```

Code2 是一个多进程程序，主进程通过 main()->a()->b()->do_pi()这条路径最终调用了 do_pi()，子进程则通过 main()->ak()->bk()->do_pi()这条路径最终调用了 do_pi()。在系统中运行 code2 后，执行下述命令：

```
$perf top –G
```

将得到如图*所示的界面。

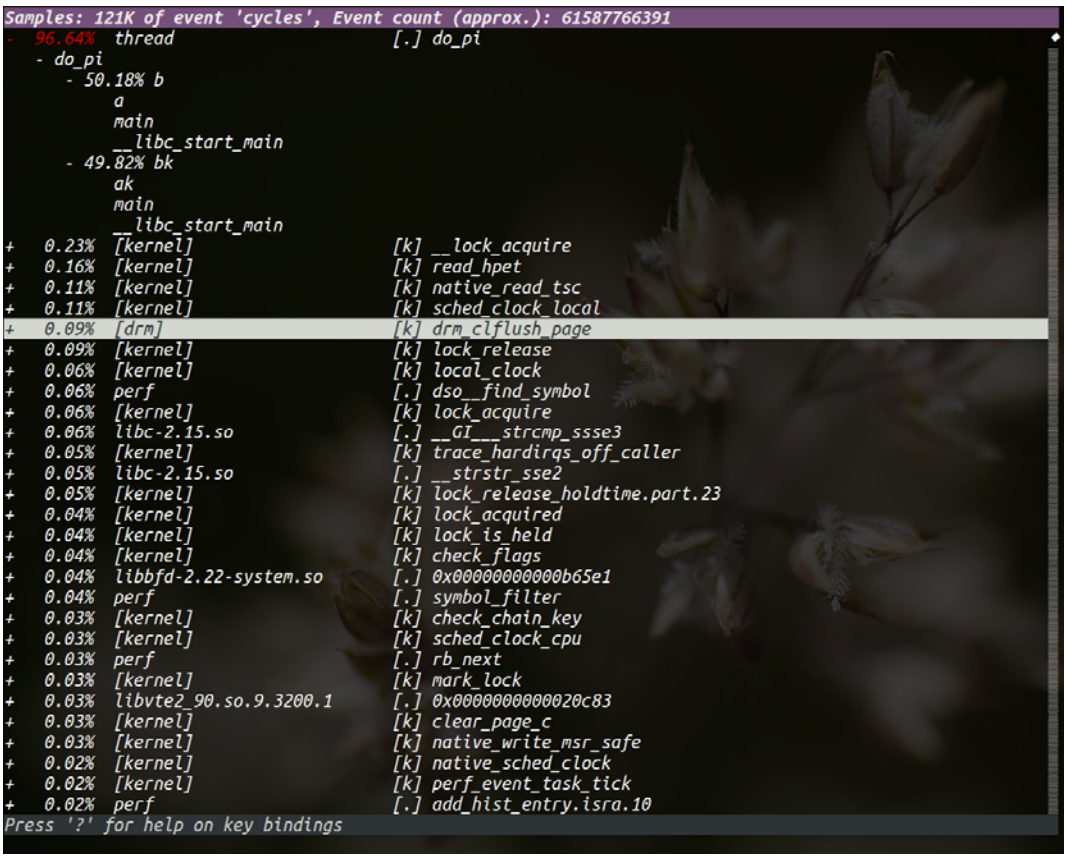

图 12 perf top -G 显示的函数调用图谱

从图 12 上可以看到，与普通的 perf top 界面相比，每行的最左面多了一个’+’号，表示此符号可以展开函数调用图谱。将焦点定位在符号 do_pi 后，通过回车键，即可展开 do_pi()的调用图谱。展开以后，perf top 会在每条调用路径前显示相应的采样率，表示有多少采样是由该条路径触发的。在图 12 中，共有 2 条路径调用了 do_pi()，分别是 main()->a()->b()->do_pi()与 main()->ak()->bk()->do_pi()。这两条路径的采样率分别是 50.18%与 49.82%。对于复杂的软件而言，函数调用图谱为更加深入的性能分析提供了途径。我们不仅能够直观地查找热点函数，还能够进一步直观地查找到使得该函数成为热点的代码路径。

参数-G 具有 3 个配置选项 output_type，min_percent 与 call_order。选项 output_type 允许用户指定调用图谱的显示方式。Perf 提供了 3 种显示方式:

1) graph: 使用调用树，将每条调用路径进一步折叠。这种显示方式更加直观。

每条调用路径的采样率为绝对值。也就是该条路径占整个采样域的比率。使用方法为:

```
$perf top -G graph
```

输出结果如图 13 所示。

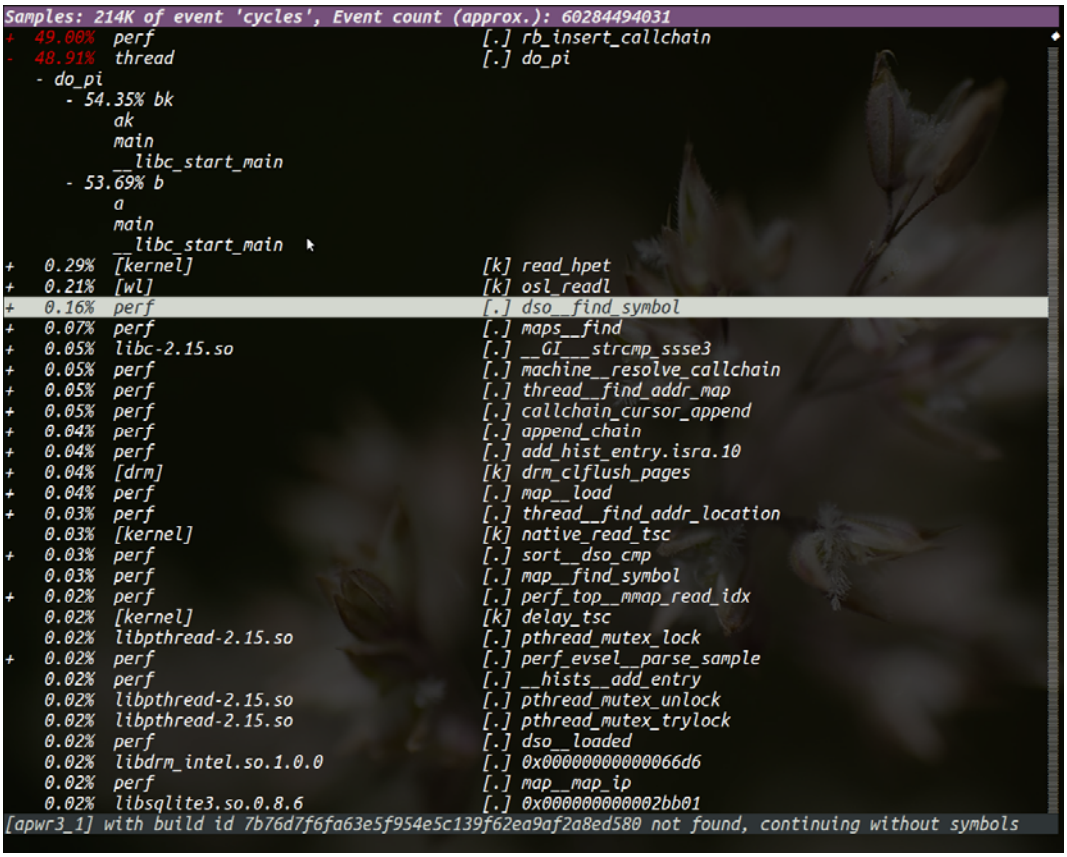

图 13 函数调用图谱的 graph 风格

2) fractal

默认选项。类似与 graph，但是每条路径前的采样率为相对值。图 12 中采用的就是 fractal 风格。

3) flat

不折叠各条调用路径，也不显示每条调用路径的采样率。如图 14 所示。

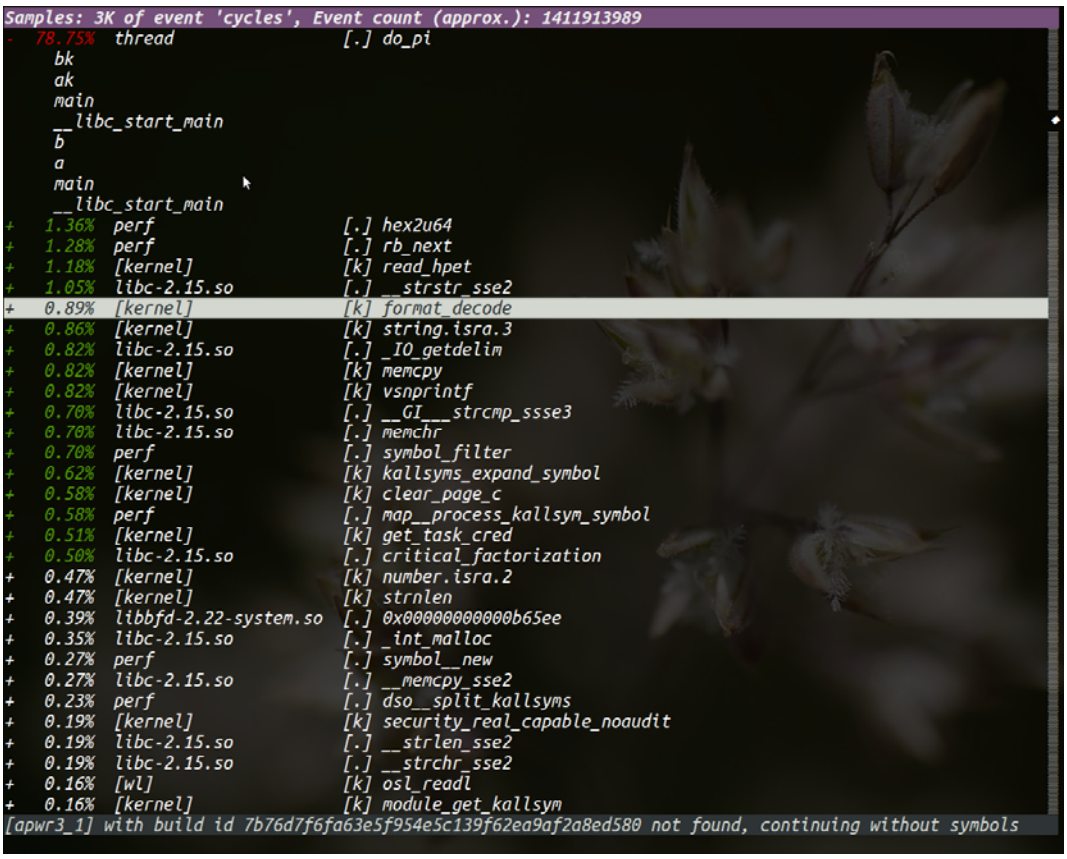

图 14. flat 风格的函数调用图谱

我们还可以通过 min_percent 选项过滤采样率低于 min_percent 的调用路径。min_percent 的默认值为 0.5。

选项 call_order 用以设定调用图谱的显示顺序，该选项有 2 个取值，分别是 callee 与 caller。将该选项设为 callee 时，perf 按照被调用的顺序显示调用图谱，上层函数被下层函数所调用，如图*所示。该选项被设为 caller 时，按照调用顺序显示调用图谱，即上层函数调用了下层函数。执行如下命令后，可以得到 caller 风格的调用图谱，如图 15 所示。

```
$perf top -G graph,0.5,caller
```

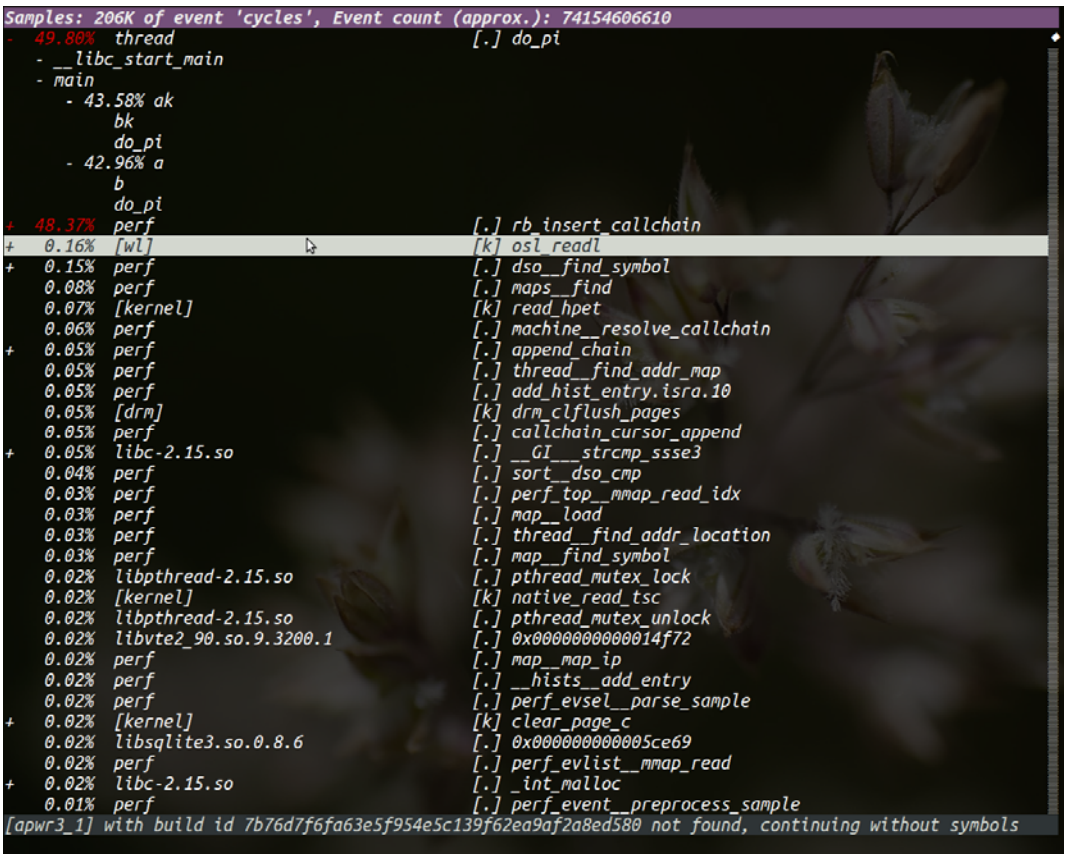

图 15. caller 风格的函数调用图谱

‘--dsos’ <dso_name[,dso_name…]>

仅显示 dso 名为 dso_name 的符号。可以同时指定多个 dso，各个 dso 名字之间通过逗号隔开。

‘--comms’ <comm[,comm…]>

仅显示属于进程“comm”的符号。

‘--symbols <symbol[,symbol…>

仅显示指定的符号。

‘-M’ or ‘--disassembler-style’

显示符号注解时，可以通过此参数指定汇编语言的风格。