# MySQL性能诊断实践

# 之 系统观测工具

# 第一印象

慢

- 再试一次
- 优化 SQL
- 调大 buffer pool
- 换 SSD
- "MySQL慢怎么办"

# 第一印象

慢

- 再试一次 (不可复现的外界因素影响)
- 优化 SQL (执行复杂度 >> 需求复杂度)
- 调大 buffer pool (MySQL资源限制)
- 换 SSD (服务器资源限制)
- “MySQL 慢怎么办"

# 目录

- MySQL 慢的诊断思路
- 系统观测工具介绍
- bcc (eBPF脚本集) 使用举例
- eBPF 使用方法/限制

# MySQL 慢的诊断思路

- MySQL内部 观测
- 外部资源 观测
- 外部需求 改造

# MySQL 慢的诊断思路

- MySQL 内部观测
  - show processlist
  - explain
  - profiling
  - performance_schema / sys
  - innodb_metrics
  - ???

# MySQL 慢的诊断思路

– 外部资源观测

– 60s 快速巡检

TECHNO.OGY ACTION 爱可生

# MySQL 慢的诊断思路

1. uptime
2. dmesg -T | tail
3. vmstat 1
4. mpstat -P ALL
5. pidstat 1

# MySQL 慢的诊断思路

6. iostat -xz 1
7. free -m
8. sar -n DEV 1
9. sar -n TCP,ETCP 1
10. top

# MySQL 慢的诊断思路

- 外部需求 改造
  - Examples of Common Queries

# MySQL 慢的诊断思路

```
SELECT article, dealer, price
FROM   shop s1
WHERE  price=(
           SELECT MAX(s2.price)
           FROM shop s2
           WHERE s1.article = s2.article);
```

# MySQL 慢的诊断思路

```
SELECT s1.article, dealer, s1.price
FROM shop s1
JOIN (
    SELECT article, MAX(price) AS price
    FROM shop GROUP BY article) AS s2
    ON s1.article = s2.article AND s1.price =
s2.price;
```

# MySQL 慢的诊断思路

```
SELECT s1.article, s1.dealer, s1.price
FROM shop s1
LEFT JOIN shop s2 ON s1.article =
s2.article AND s1.price < s2.price
WHERE s2.article IS NULL;
```

ACTION 爱可生 TECHNOLOGY

# MySQL 慢的诊断思路

## Relational algebra

ACTION 爱可生

# MySQL 慢的诊断思路

- MySQL内部 观测
- 外部资源 观测
- 外部需求 改造

# 目录

- MySQL 慢的诊断思路
- 系统观测工具介绍
- bcc (eBPF脚本集) 使用举例
- eBPF 使用方法/限制

# 系统观测工具介绍

https://jvns.ca/blog/2017/07/05/linux-tracing-systems/

ACTION 爱可生

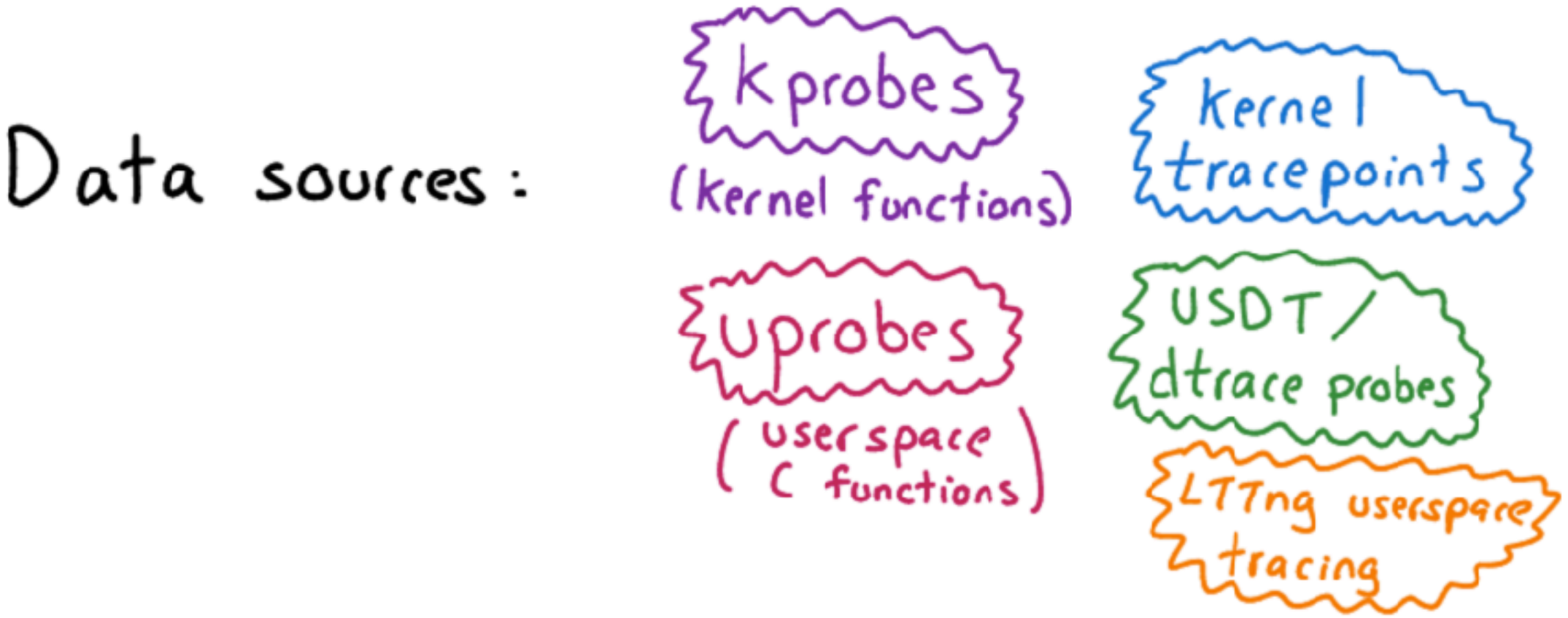
# 系统观测工具介绍

- 数据源（看什么）
- 数据采集过程（怎么拿）
- 数据处理前端（怎么看）

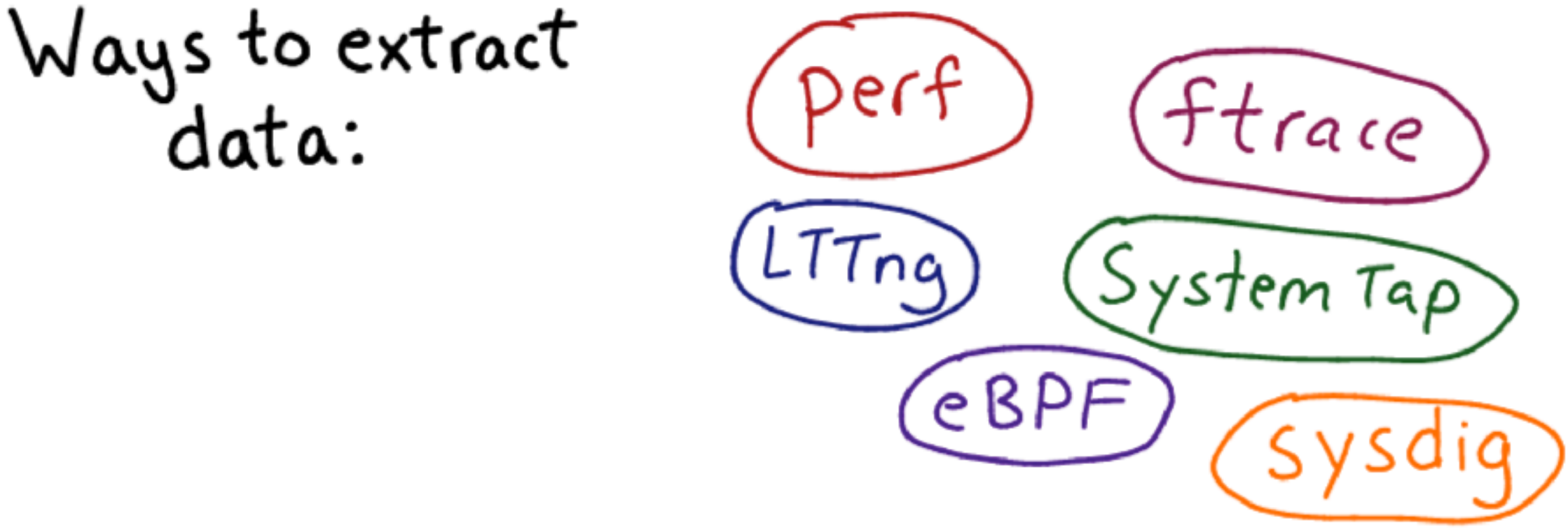
# 系统观测工具介绍

- 数据源（看什么）
- 数据采集过程（怎么拿）
- 数据处理前端（怎么看）

# 系统观测工具介绍


Data sources:
kprobes
(kernel functions)
kernel tracepoints
uprobes
(userspace C functions)
USDT / dtrace probes
LTTng userspace tracing

# 系统观测工具介绍



# 系统观测工具介绍

frontends:

perf

ftrace

trace-cmd

catapult

kernelshark

trace compass

bcc

sysdig

LTTng

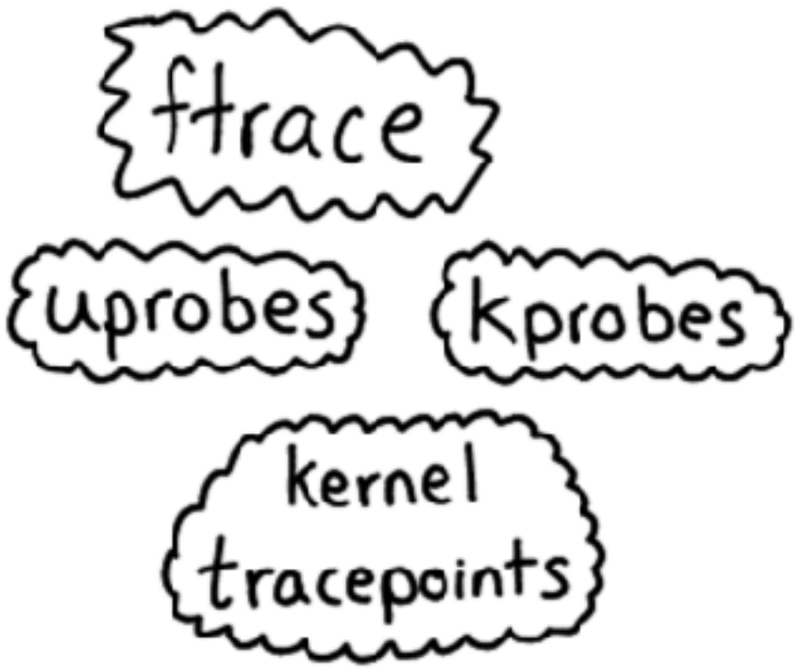
System Tap

# 系统观测工具介绍

## 对比

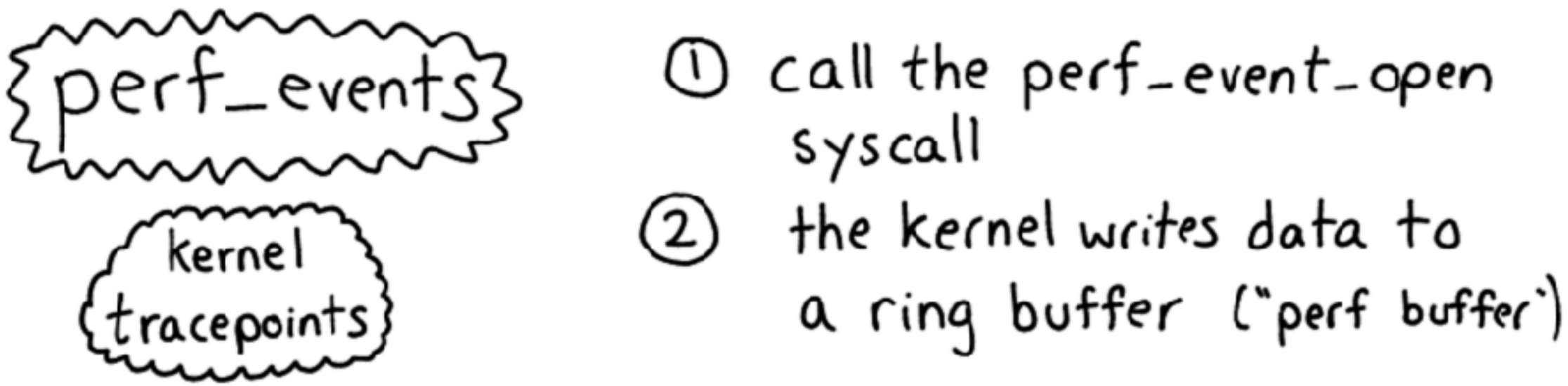
- ftrace
- perf_events
- eBPF
- Systemtap

# 系统观测工具介绍




magical filesystem at /sys/kernel/debug/tracing. Super powerful, you interact with it by reading from / writing to files.

# 系统观测工具介绍



# 系统观测工具介绍

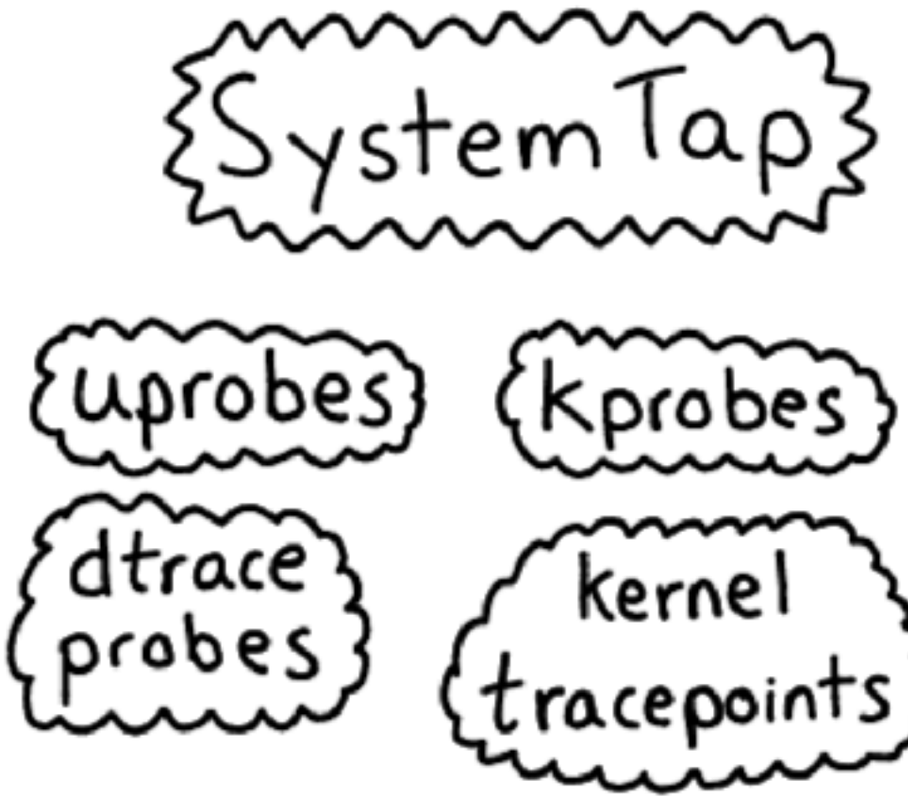

The newest and most powerful

① Write a small eBPF program

② Ask Linux to attach it to a kprobe/uprobe/tracepoint

③ The eBPF program sends data to userspace with ftrace/perf/BPF maps

# 系统观测工具介绍

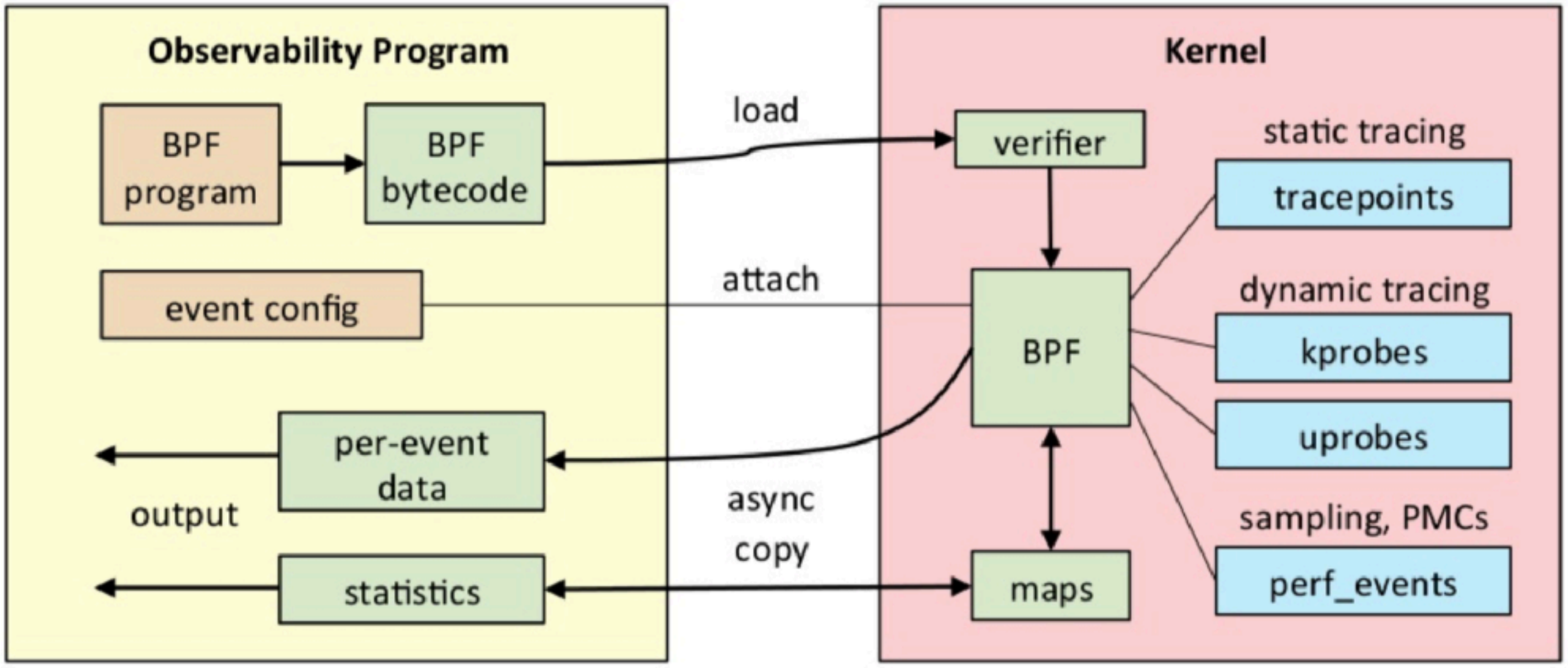



① Write some C code

② Compile it into a custom kernel module

③ Insert that module into the kernel

# 系统观测工具介绍


Observability Program
BPF program
BPF bytecode
event config
per-event data
output
statistics
load
attach
async copy
Kernel
verifier
BPF
maps
static tracing
tracepoints
dynamic tracing
kprobes
uprobes
sampling, PMCs
perf_events

# 目录

- MySQL 慢的诊断思路
- 系统观测工具介绍
- bcc (eBPF脚本集) 使用举例
- eBPF 使用方法/限制

# bcc 使用举例 – 1

## MySQL请求延迟

```
./dbstat -p `pidof mysqld` -u -- mysql
```

# bcc 使用举例 – 1

select

```
query latency (us)    count     distribution
    0 -> 1            0                                          |
    2 -> 3            0                                          |
    4 -> 7            0                                          |
    8 -> 15           0                                          |
   16 -> 31           0                                          |
   32 -> 63           0                                          |
   64 -> 127          400308    ************************************|
  128 -> 255          148021    **************                      |
  256 -> 511          261                                        |
  512 -> 1023         3                                          |
 1024 -> 2047         0                                          |
 2048 -> 4095         1                                          |
```

# bcc 使用举例 – 1

select + insert

```
query latency (us)    | count     | distribution
    0 -> 1            | 0         |                                        |

                      |           |  ...

   32 -> 63           | 0         |                                        |
   64 -> 127          | 9198      | **************                         |
  128 -> 255          | 25826     | ****************************************|
  256 -> 511          | 8283      | ************                           |
  512 -> 1023         | 12568     | ********************                   |
 1024 -> 2047         | 14533     | **********************                 |
 2048 -> 4095         | 9840      | ***************                        |
 4096 -> 8191         | 4031      | ******                                 |
 8192 -> 16383        | 463       |                                        |
```

# bcc 使用举例 – 2

## MySQL 慢查询

```
./dbslower -p `pidof mysqld` -m 5 -- mysql
```

# bcc 使用举例 – 2

## select + update

```
Tracing database queries for pids 4754 slower than 5 ms...
TIME(s)          PID          MS QUERY
0.956044         4754      5.358 UPDATE sbtest1 SET k=k+1 WHERE id=514
0.956199         4754      5.837 UPDATE sbtest1 SET k=k+1 WHERE id=505
0.956876         4754      5.257 UPDATE sbtest1 SET k=k+1 WHERE id=503
0.955977         4754      6.656 UPDATE sbtest1 SET k=k+1 WHERE id=503
0.956287         4754      6.801 UPDATE sbtest1 SET k=k+1 WHERE id=503
0.955870         4754      7.554 UPDATE sbtest1 SET k=k+1 WHERE id=498
0.956329         4754      7.121 UPDATE sbtest1 SET k=k+1 WHERE id=497
```

# bcc 使用举例 – 2

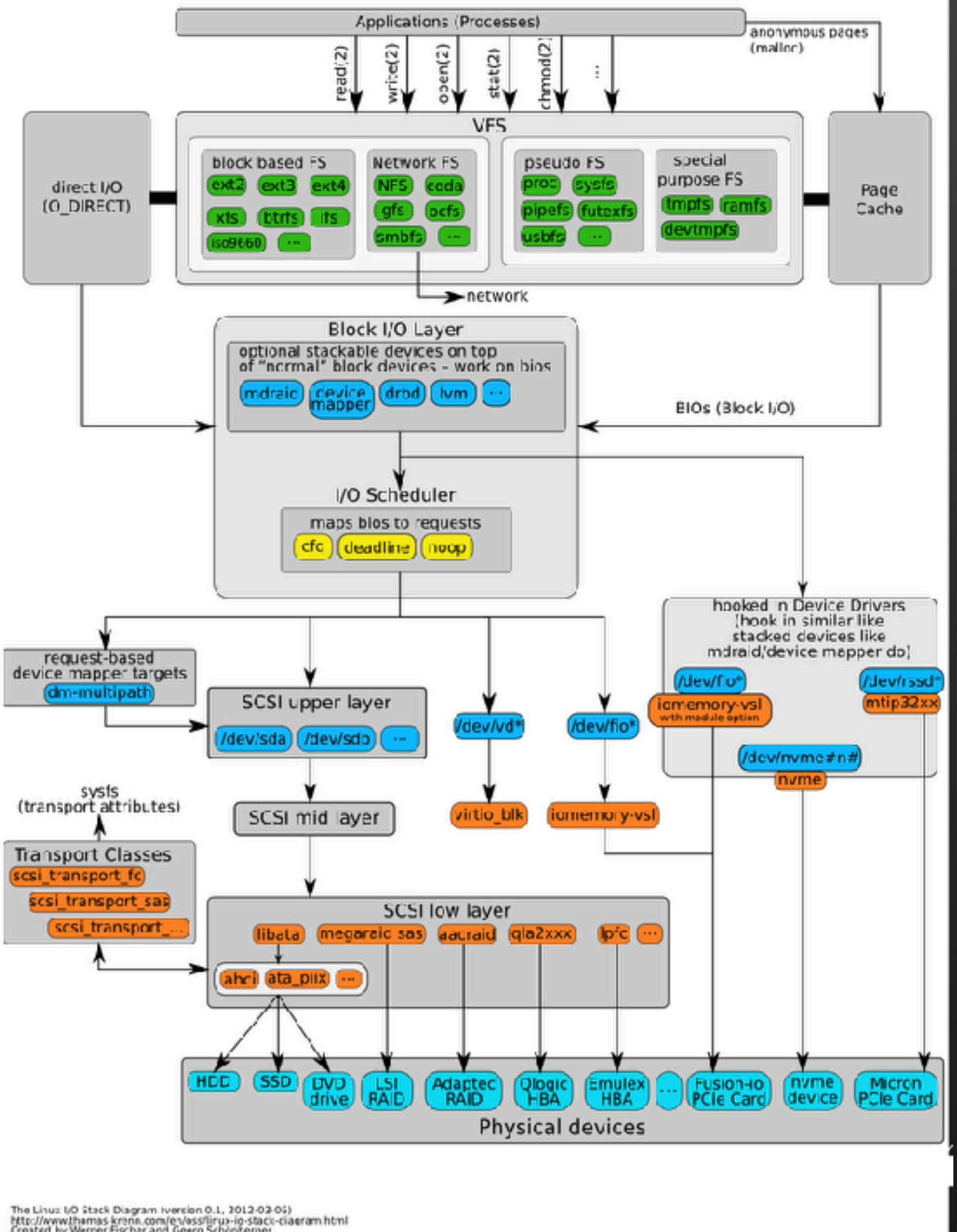
与MySQL的慢日志相比, 可以低成本地完成:

1. 获取少量慢查询
2. 获取某种模式的慢查询
3. 获取某个用户的慢查询

...

# The Linux I/O Stack Diagram

version 0.1, 2012-03-06
outlines the Linux I/O stack as of Kernel version 3.3




The Linux I/O Stack Diagram (version 0.1, 2012-03-06)
http://www.thomas-krenn.com/en/oss/linux-io-stack-diagram.html
Created by Werner Fischer and Georg Schönberger

# Linux IO stack

- MySQL
- VFS (read/write/fdatasync/…)
- Filesystem (ext4/…)
- Block I/O (device mapper/…)
- I/O scheduler (cfq/deadline/…)
- SCSI
- Physical Device

# bcc 使用举例 – 3

## VFS 延迟分析

./ext4dist 2 1

# bcc 使用举例 – 3

```
...
operation = write
                usecs               count     distribution
                    0 -> 1          0                                       |
                    2 -> 3          2                                       |
                    4 -> 7          507                                     |
                    8 -> 15         22123     ************************      |
                   16 -> 31         10444     *********                     |
                   32 -> 63         2073      *                             |
                   64 -> 127        3598      **                            |
                  128 -> 255        11234     **********                    |
                  256 -> 511        2409      *                             |
...
```

# bcc 使用举例 – 4

## Ext4 文件IO延迟分析

./ext4slower 1

# bcc 使用举例 – 4

```
Tracing ext4 operations slower than 1 ms
TIME     COMM           PID    T BYTES   OFF_KB   LAT(ms) FILENAME
21:59:40 mysqld         4754   S 0       0          3.56 ib_logfile1
21:59:40 mysqld         4754   S 0       0          8.42 sbtest1.ibd
21:59:41 mysqld         4754   S 0       0          3.83 ib_logfile1
21:59:41 mysqld         4754   S 0       0          8.35 sbtest1.ibd
21:59:42 mysqld         4754   S 0       0          8.50 sbtest1.ibd
21:59:42 mysqld         4754   S 0       0          3.53 ib_logfile1
21:59:42 mysqld         4754   S 0       0          8.34 sbtest1.ibd
21:59:43 mysqld         4754   S 0       0          2.69 ib_logfile1
21:59:43 mysqld         4754   S 0       0          8.41 sbtest1.ibd
```

# bcc 使用举例 – 4

排查其他程序对IO的影响，抓住证据

```
Tracing ext4 operations slower than 10 ms
TIME     COMM           PID    T BYTES      OFF_KB   LAT(ms) FILENAME
22:03:14 dd             42639  W 1073741824 0         873.20 test1.img
22:03:15 mysqld         4754   W 1048576    1024        16.48 ibdata1
22:03:15 mysqld         4754   W 507904     2048        13.98 ibdata1
22:03:15 mysqld         4754   W 1048576    1302528     15.10 sbtest1.ibd
22:03:15 mysqld         4754   S 0          0         110.94 ibdata1
22:03:16 mysqld         4754   W 1048576    1306624     22.35 sbtest1.ibd
```

# bcc 使用举例 – 5

## 块设备延迟分析

./biolatency -D 2

# bcc 使用举例 – 5

## select & update

```
disk = 'sdb'
                 usecs               count     distribution
                     0 -> 1          0         |                                        |
…
                    16 -> 31         0         |                                        |
                    32 -> 63         4694      |**********************                  |
                    64 -> 127        3399      |***************                         |
                   128 -> 255        2211      |**********                              |
                   256 -> 511        2250      |**********                              |
                   512 -> 1023       642       |**                                      |
                  1024 -> 2047       0         |                                        |
                  2048 -> 4095       0         |                                        |
```

# bcc 使用举例 – 5

## select

```
disk = 'sdb'
     usecs               count     distribution
         0 -> 1          0                                      |
…
        64 -> 127        0                                      |
       128 -> 255        0                                      |
       256 -> 511        2        ****************************  |
       512 -> 1023       0                                      |
      1024 -> 2047       0                                      |
      2048 -> 4095       3        ****************************************
```

# bcc 使用举例 – 6

## MySQL线程对文件的IO压力汇总

```
./filetop -p `pidof mysqld` -C 5
```

# bcc 使用举例 – 6

```
22:26:30 loadavg: 7.50 5.28 4.87 18/1925 44235
TID    | COMM             READS   WRITES R_Kb     W_Kb     T | FILE
39956  | mysqld           0       115    0        462      R | ib_logfile1
40075  | mysqld           0       107    0        424      R | ib_logfile1
39900  | mysqld           0       1220   0        137      R | R820-08.log
38046  | mysqld           0       1263   0        142      R | R820-08.log
39085  | mysqld           0       101    0        332      R | ib_logfile1
38957  | mysqld           0       114    0        425      R | ib_logfile1
39959  | mysqld           0       1      0        2        R | ibmPAQIO
4780   | mysqld           0       4      0        28       R | ib_logfile1
40266  | mysqld           0       107    0        361      R | ib_logfile1
39984  | mysqld           0       111    0        414      R | ib_logfile1
```

# bcc 使用举例 – 7

## 短生命周期的临时文件检测

```
./filetop -p `pidof mysqld` -C 5
```

# bcc 使用举例 – 7

```
root@R820-08:/usr/share/bcc/tools# ./filelife
TIME     | PID    | COMM            | AGE(s) | FILE
22:17:01 | 43687  | cron            |   0.00 | tmpfgHF5vY
22:22:21 | 39170  | mysqld          |   5.30 | #sql1292_59a1f_0.frm
```

# bcc 使用举例 – 8

## 短连接分析

## ./tcplife

# bcc 使用举例 – 8

| PID | COMM | LADDR | LPORT | RADDR | RPORT | TX_KB | RX_KB | MS |
|---|---|---|---|---|---|---|---|---|
| 44245 | sysbench | 127.0.0.1 | 35038 | 127.0.0.1 | 3306 | 16 | 699 | 312.05 |
| 44245 | sysbench | 127.0.0.1 | 35036 | 127.0.0.1 | 3306 | 17 | 736 | 312.20 |
| 44245 | sysbench | 127.0.0.1 | 35034 | 127.0.0.1 | 3306 | 15 | 662 | 312.41 |
| 44245 | sysbench | 127.0.0.1 | 35032 | 127.0.0.1 | 3306 | 14 | 638 | 312.45 |
| 44245 | sysbench | 127.0.0.1 | 35026 | 127.0.0.1 | 3306 | 14 | 626 | 313.17 |
| 44245 | sysbench | 127.0.0.1 | 35028 | 127.0.0.1 | 3306 | 12 | 552 | 313.18 |

# bcc 使用举例 – 9

## 长连接分析

./tcptop -C 5

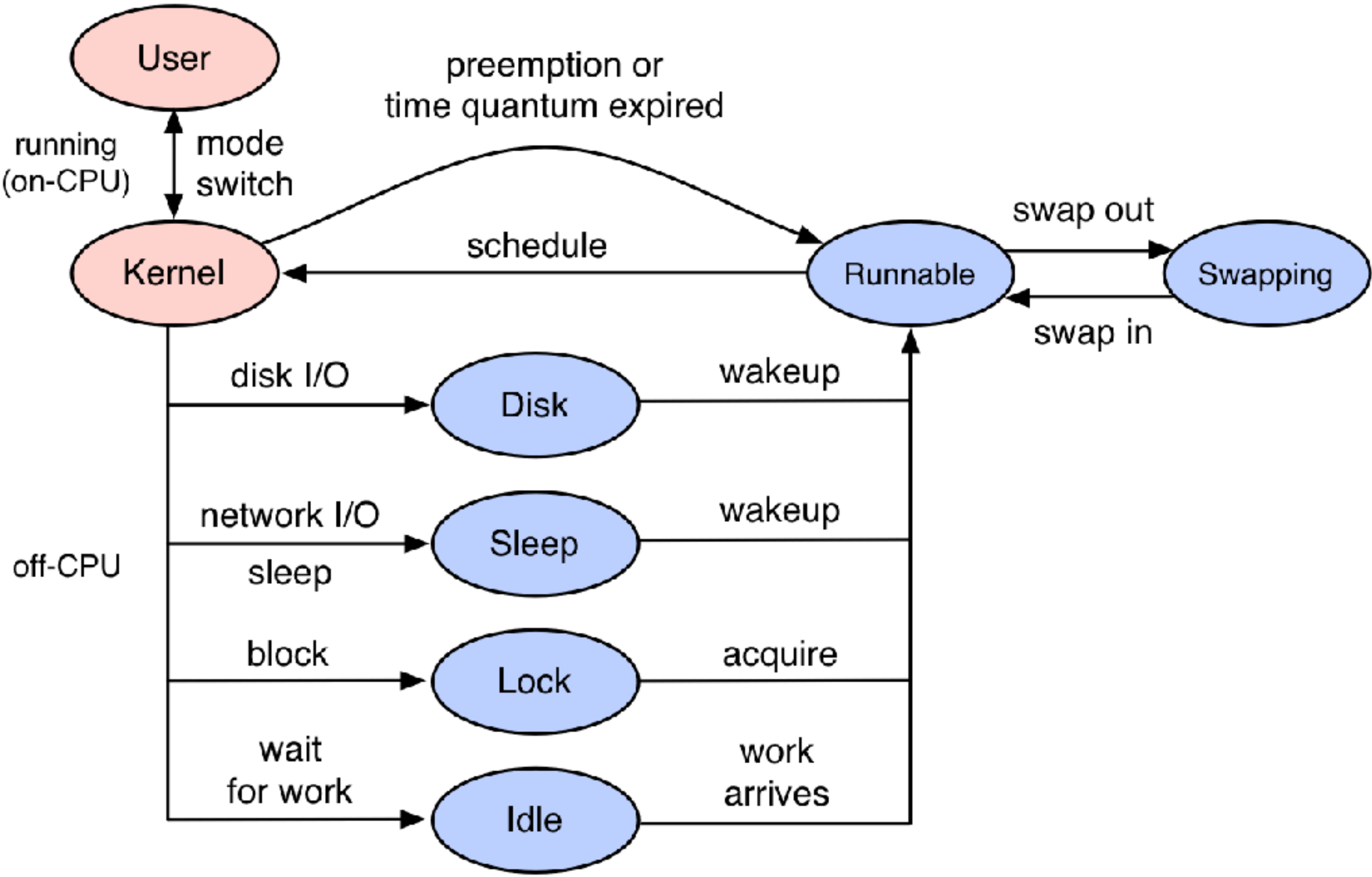
# bcc 使用举例 – 9

```
22:33:41 loadavg: 17.28 6.81 5.01 126/1933 44788
PID    COMM         LADDR                RADDR                RX_KB  TX_KB
44668  sysbench     127.0.0.1:35654      127.0.0.1:3306       16116    369
44669  sysbench     127.0.0.1:35650      127.0.0.1:3306       15957    365
44702  sysbench     127.0.0.1:35728      127.0.0.1:3306       15871    363
44758  sysbench     127.0.0.1:35838      127.0.0.1:3306       15834    362
44698  sysbench     127.0.0.1:35718      127.0.0.1:3306       15797    362
```

# bcc 使用举例 – 10

## CPU offcpu 消耗分析

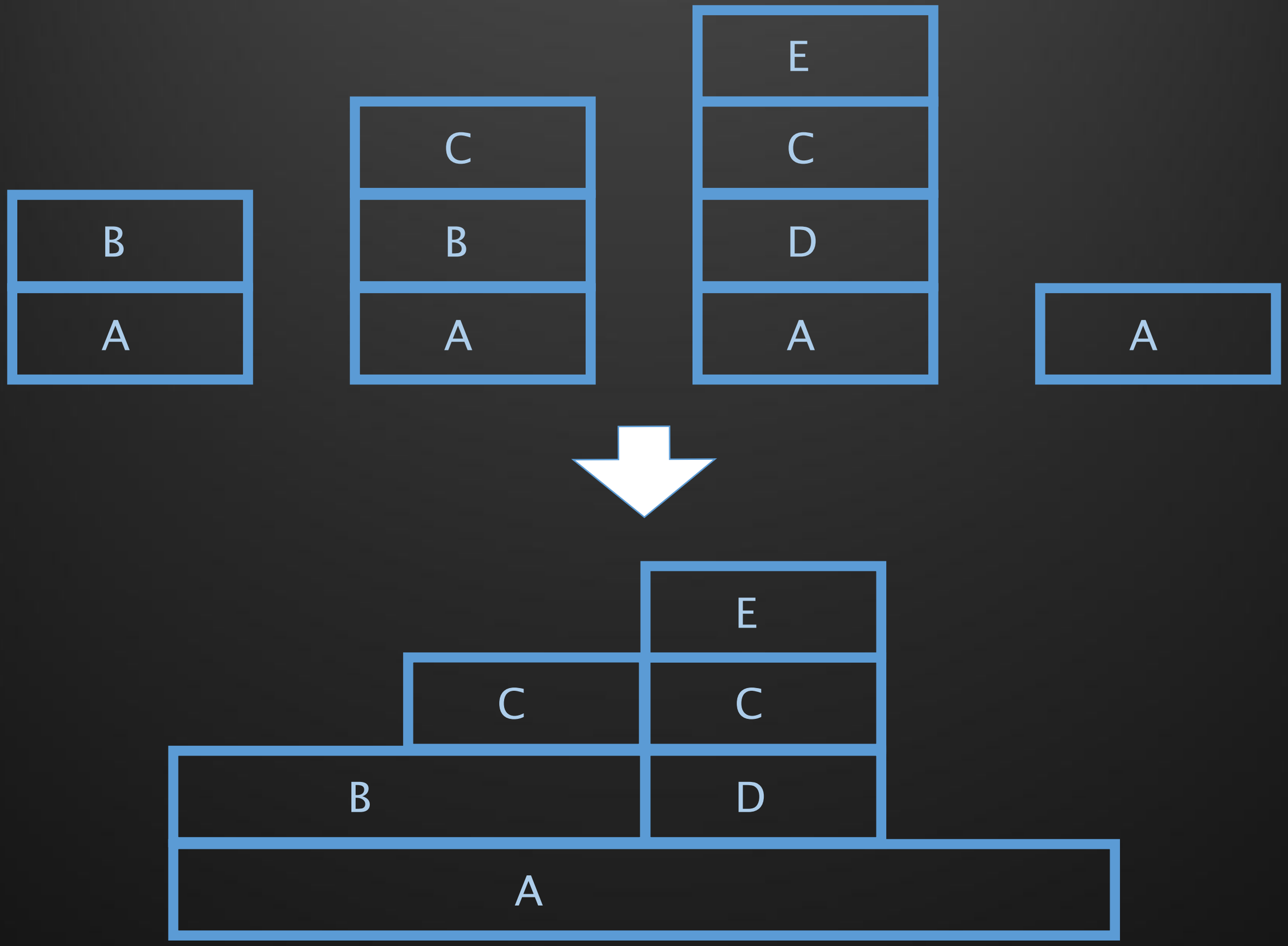
offcputime -df -p `pgrep -nx mysqld` 30


User
running
(on-CPU)
mode
switch
preemption or
time quantum expired
Kernel
schedule
Runnable
swap out
Swapping
swap in
disk I/O
Disk
wakeup
off-CPU
network I/O
sleep
Sleep
wakeup
block
Lock
acquire
wait
for work
Idle
work
arrives

Off-CPU Time Flame Graph

Search

finish_task_swi..
finish_task_switch | __schedule | finish_task_switch
__schedule | schedule | __schedule
schedule | io_schedule | schedule
io_schedule | generic_file_re.. | io_schedule
generic_file_read_iter | __vfs_read | generic_file_read_iter
__vfs_read | vfs_read | __vfs_read
vfs_read | SyS_pread64 | vfs_read
SyS_pread64 | do_syscall_64 | SyS_pread64
entry_SYSCALL_64_fastpath | return_from_SYS.. | do_syscall_64
- | - | return_from_SYSCALL_64
__GI___libc_pread | - | -
[unknown] | __GI___libc_pread | __GI___libc_pread
os_file_read_func(IORequest&, int, void*, unsigned lo.. | [unknown] | [unknown]
fil_io(IORequest const&, bool, page_id_t const&, page.. | os_file_read_func(IORequest&, int, v.. | os_file_read_func(IORequest&, int, v..
buf_read_page(page_id_t const&, page_size_t const&) | fil_io(IORequest const&, bool, page_.. | fil_io(IORequest const&, bool, page_..
buf_page_get_gen(page_id_t const&, page_size_t const&.. | buf_read_page(page_id_t const&, page_.. | buf_read_page(page_id_t const&, page_..
btr_cur_search_to_nth_level(dict_index_t*, unsigned l.. | buf_page_get_gen(page_id_t const&, pa.. | buf_page_get_gen(page_id_t const&, pa..
[unknown] | btr_cur_search_to_nth_level(dict_inde.. | btr_cur_search_to_nth_level(dict_inde..
ha_innobase::records_in_range(unsigned int, st_key_ran.. | [unknown] | [unknown]
handler::multi_range_read_info_const(unsigned int, st_.. | row_search_mvcc(unsigned char*, page_cur_mode.. | row_search_mvcc(unsigned char*, page_cur_mode..
DsMrr_impl::dsmrr_info_const(unsigned int, st_range_sc.. | ha_innobase::index_read(unsigned char*, unsig.. | ha_innobase::index_read(unsigned char*, unsig..
[unknown] | handler::ha_index_read_idx_map(unsigned char*, un.. | handler::ha_index_read_idx_map(unsigned char*, un..
test_quick_select(THD*, Bitmap<64u>, unsigned long lon.. | join_read_const_table(JOIN_TAB*, st_position*) | join_read_const_table(JOIN_TAB*, st_position*)
JOIN::estimate_rowcount() | JOIN::extract_func_dependent_tables() | JOIN::extract_func_dependent_tables()
JOIN::make_join_plan()
JOIN::optimize()
st_select_lex::optimize(THD*)
handle_query(THD*, LEX*, Query_result*, unsigned long long, unsigned long long)
[unknown]
mysql_execute_command(THD*, bool)
Prepared_statement::execute(String*, bool)
Prepared_statement::execute_loop(String*, bool, unsigned char*, unsigned char*)
mysqld_stmt_execute(THD*, unsigned long, unsigned long, unsigned char*, unsigned long)
dispatch_command(THD*, COM_DATA const*, enum_server_command)
do_command(THD*)
handle_connection
pfs_spawn_thread
start_thread
mysqld


B
A
C
B
A
E
C
D
A
A
E
C
C
B
D
A

...

vfs_read

do_syscall_64

...

fil_io

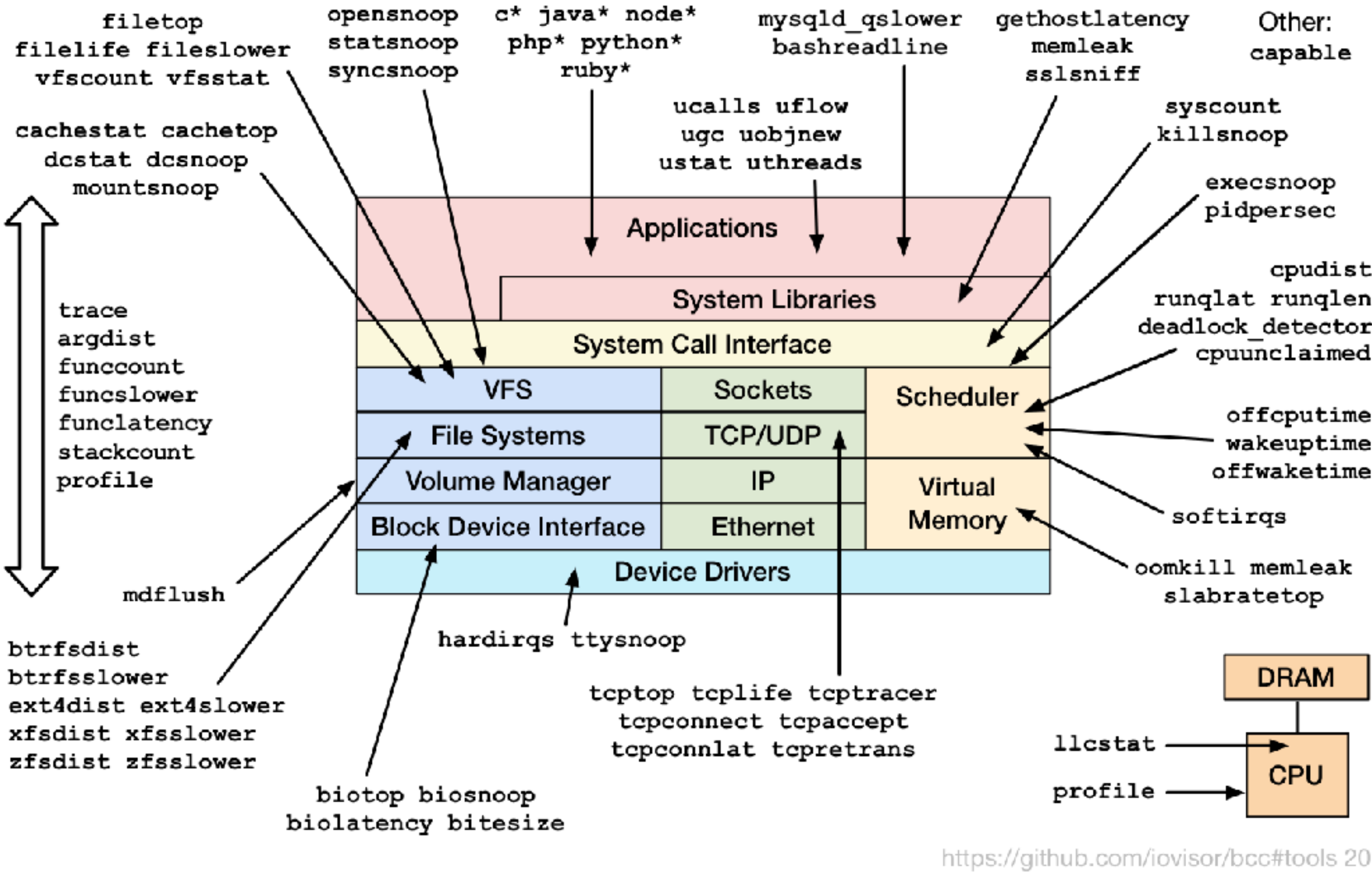
buf_read_page

btr_cur_search_to_nth_level

row_search_mvcc

ha_innobase::index_read

# Linux bcc/BPF Tracing Tools



# 目录

- MySQL 慢的诊断思路
- 系统观测工具介绍
- bcc (eBPF脚本集) 使用举例
- eBPF 使用方法/限制

# eBPF 使用方法/限制

MySQL 延迟统计 - eBPF程序

- 请求开始
  - 记录 时间戳A
- 请求结束
  - 找到对应的 时间戳A
  - 延迟 = 当前时间 - 时间戳A
  - 将 延迟 加入 结果集

# eBPF 使用方法/限制

MySQL 延迟统计 – 主程序

- 将 eBPF程序 钩到 MySQL的uprobe上
- 获取结果集
- 打印结果集

```
1. //一段C++代码，嵌入kprobe/uprobe
2. program = """
3. #include <uapi/linux/ptrace.h>
4.
5. BPF_HASH(temp, u64, u64); //临时容器
6. BPF_HISTOGRAM(latency); //存放结果的容器
7.
8. int probe_start(struct pt_regs *ctx) {
9.     u64 timestamp = bpf_ktime_get_ns(); //快速获取时间戳
10.    u64 pid = bpf_get_current_pid_tgid();
11.    temp.update(&pid, &timestamp);
12.    return 0;
13. }
14.
15. int probe_end(struct pt_regs *ctx) {
16.    u64 *timestampp;
17.    u64 pid = bpf_get_current_pid_tgid();
18.    timestampp = temp.lookup(&pid);
19.    if (!timestampp)
20.        return 0;
21.
22.    u64 delta = bpf_ktime_get_ns() - *timestampp; //获取时间差
23.    FILTER
24.    delta /= SCALE; //规范化时间差
25.    latency.increment(bpf_log2l(delta)); //存放结果
26.    temp.delete(&pid);
27.    return 0;
28. }
29. """
30.
31. ...
32. // 将代码嵌入uprobe
33. usdts = map(lambda pid: USDT(pid=pid), args.pids)
34. for usdt in usdts:
35.    usdt.enable_probe("query__start", "probe_start")
36.    usdt.enable_probe("query__done", "probe_end")
37. bpf = BPF(text=program, usdt_contexts=usdts)
38. ...
39. // 获取结果集
40. latencies = bpf["latency"]
41. ...
42. // 打印结果集
43. latencies.print_log2_hist("query latency (%s)" %
44.                           ("us" if args.microseconds else "ms"))
45. ...
```

# eBPF 使用方法/限制

## 限制

- Linux kernel
  - 4.4+ (存在统计Bug)
  - 推荐 4.9+
  - 部分功能需要 4.13+
- MySQL 编译参数开启
  - -DENABLE_DTRACE=1

# eBPF 使用方法/限制

systemtap

ftrace

perf_events

eBPF

SystemTap

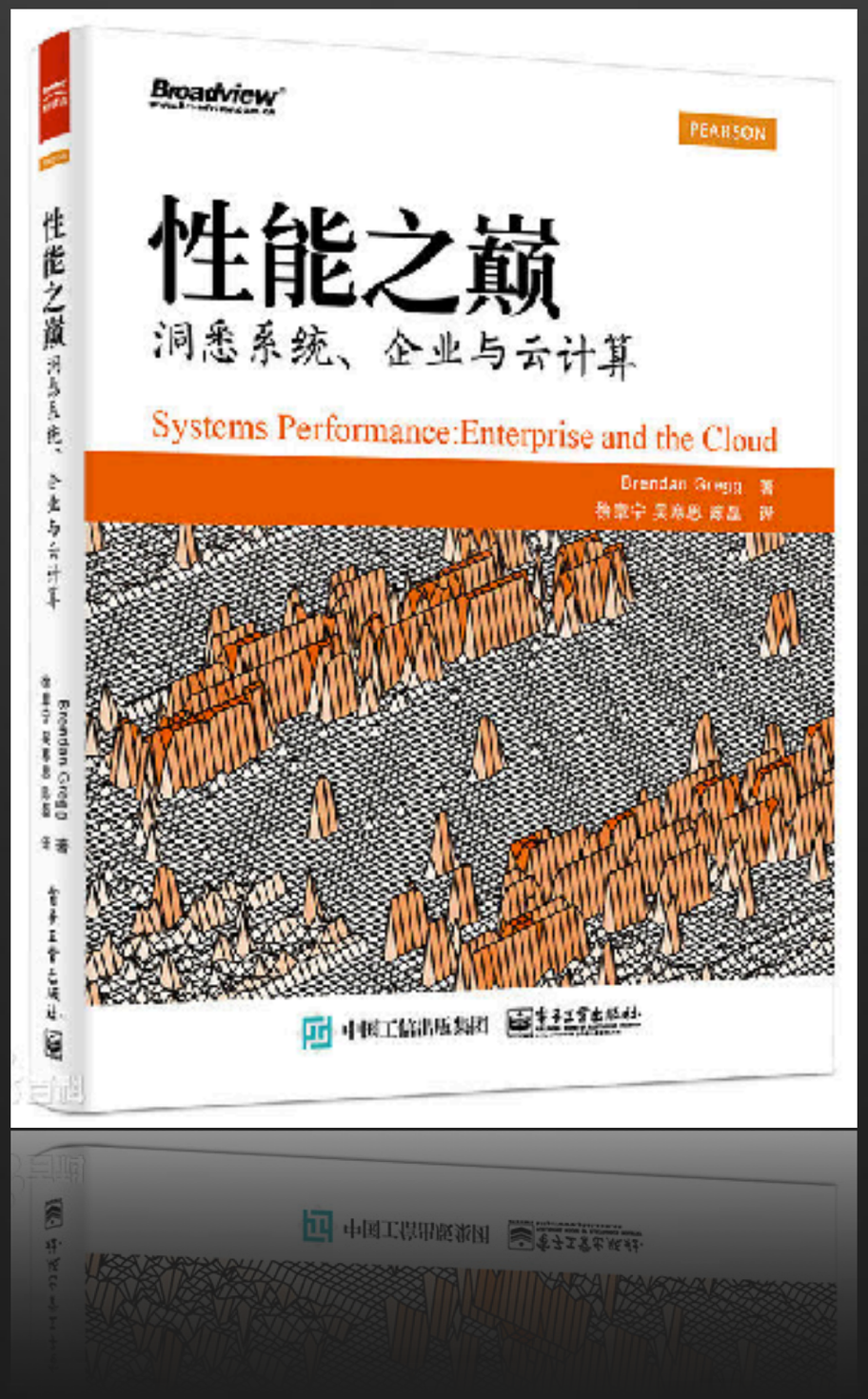
LTTng

ktap

dtrace4linux

OEL DTrace

sysdig



# PERFORMANCE TUNING GUIDE

RED HAT
ENTERPRISE LINUX
7

## Optimizing subsystem throughput in Red Hat Enterprise Linux 7

Edited by

**Marek Suchánek**

Red Hat Customer Content Services
msuchane@redhat.com

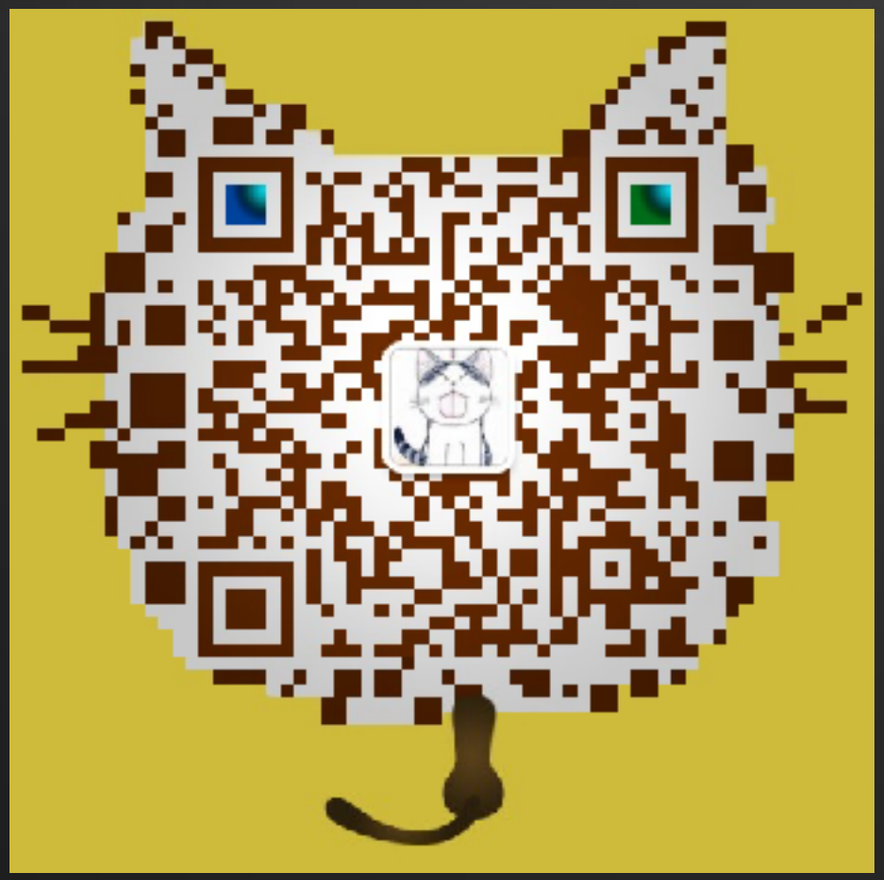
**Milan Navrátil**

Red Hat Customer Content Services

**Laura Bailey**

Red Hat Customer Content Services

**Charlie Boyle**



ACTION
爱可生