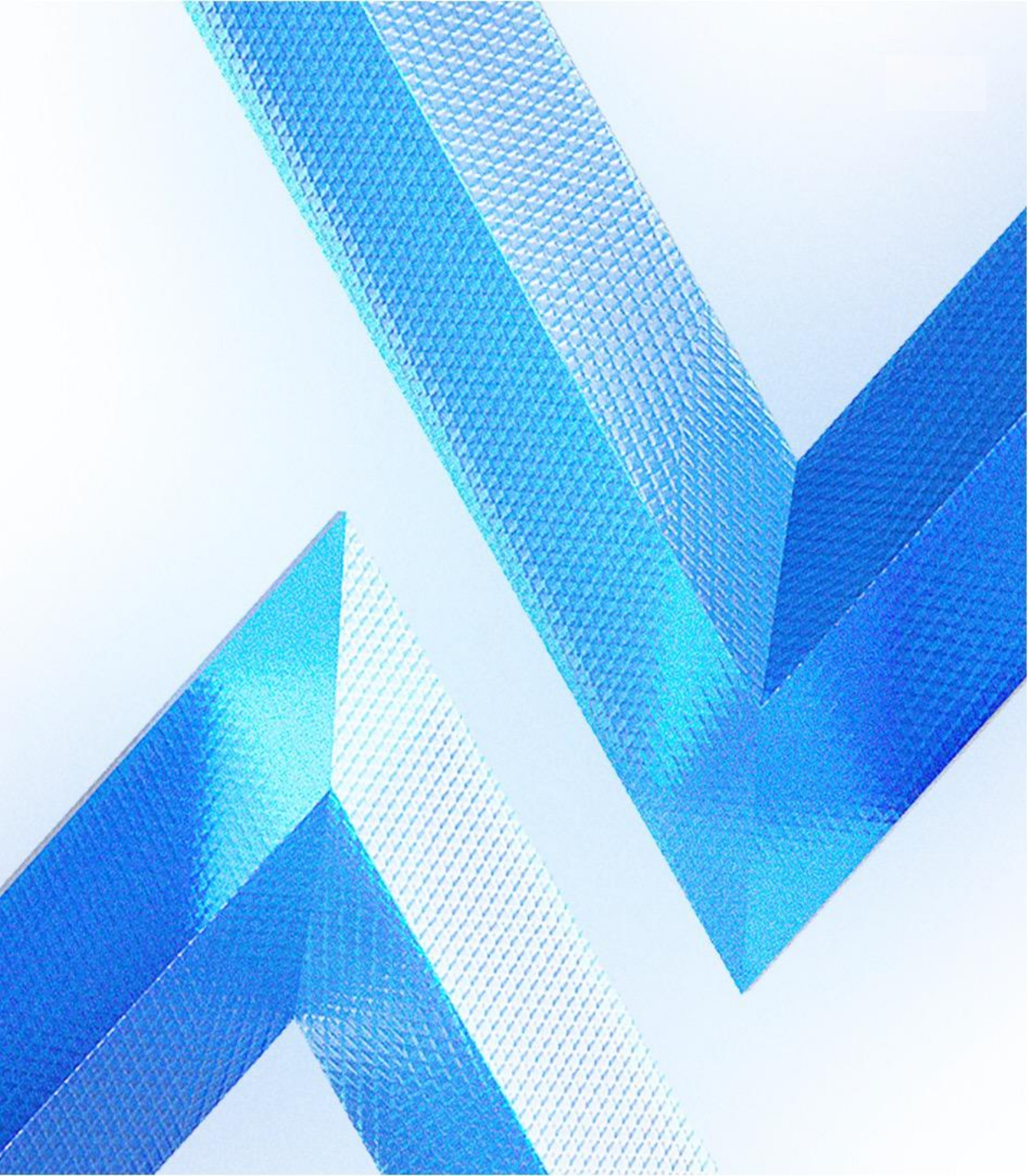
火山引擎

bcc应用最佳实践

皮振伟

2022/10 首届中国eBPF大会

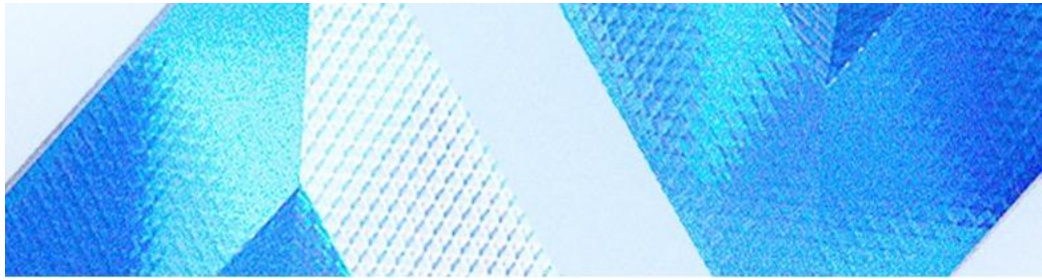


CONTENTS

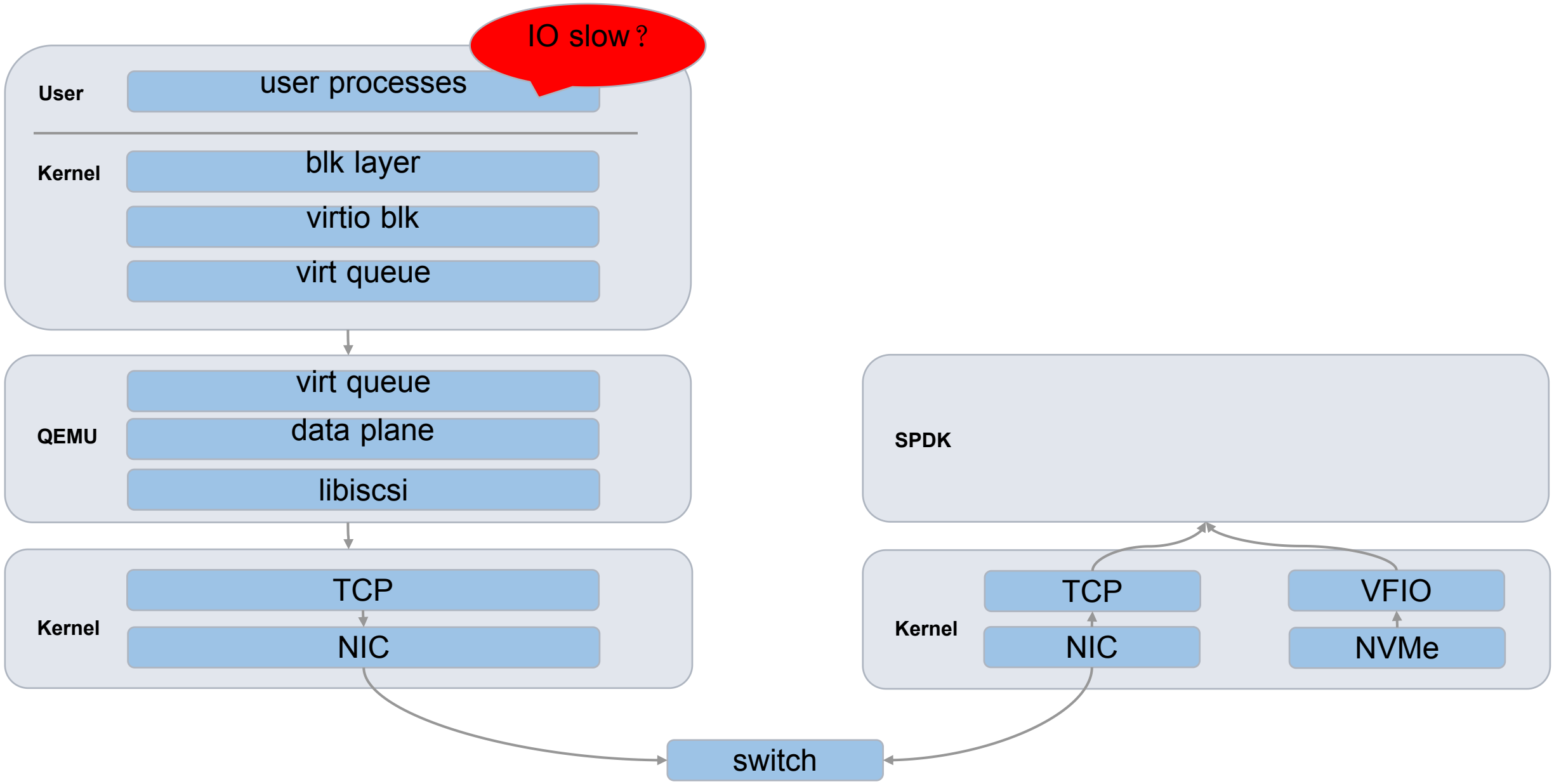
目录

01

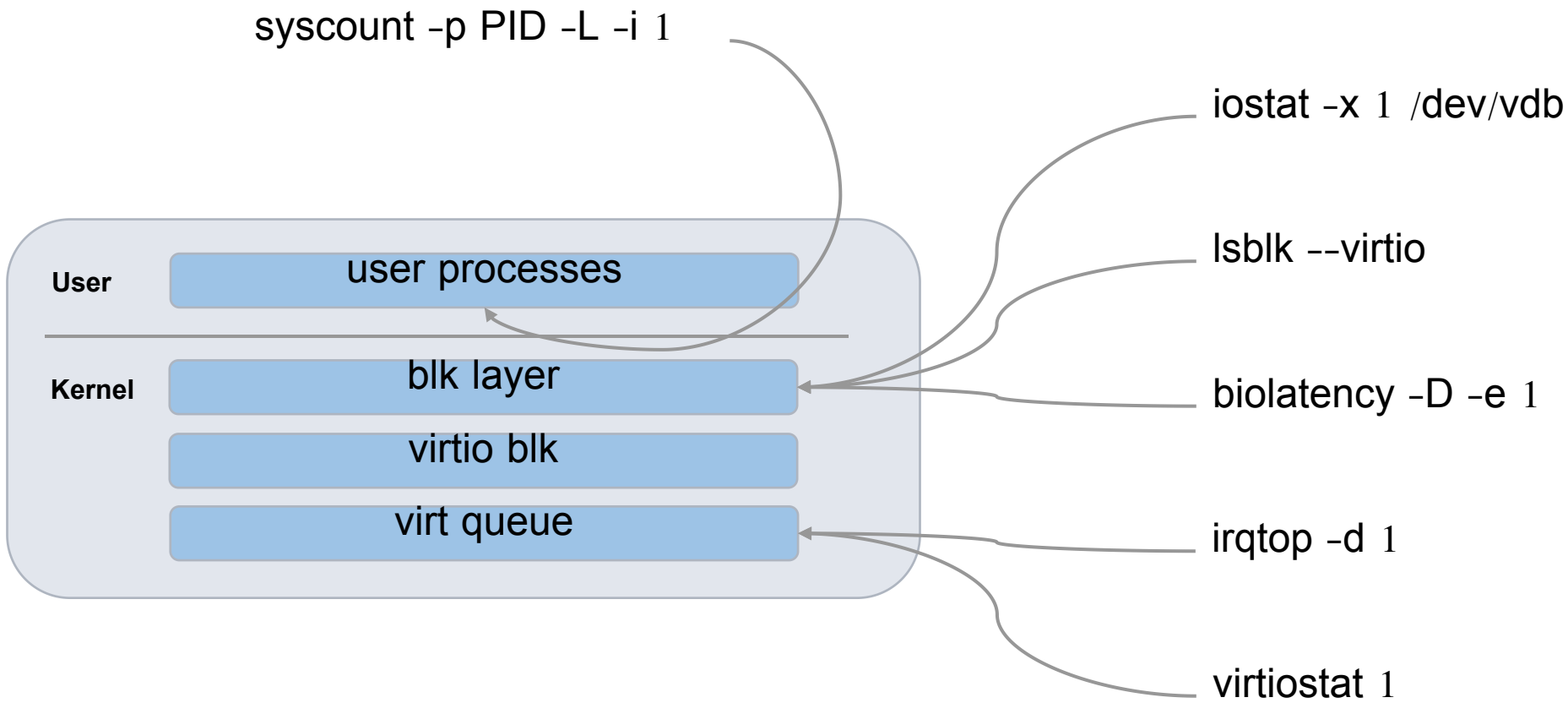
虚拟化IO案例



典型的虚拟化IO路径


IO slow ?
User
user processes
Kernel
blk layer
virtio blk
virt queue
QEMU
virt queue
data plane
libiscsi
Kernel
TCP
NIC
SPDK
Kernel
TCP
NIC
VFIO
NVMe
switch

diagnosis in guest



syscount

火山引擎

```
root@bytedance:/usr/share/bcc/tools# ./syscount -p 24421 -L -i 1
Tracing syscalls, printing top 10... Ctrl+C to quit.
[14:51:51]
SYSCALL                   COUNT        TIME (us)
pread                      1228      663052.493
pwrite                     1251      324440.808

[14:51:52]
SYSCALL                   COUNT        TIME (us)
pread                      1219      645998.345
pwrite                     1239      340066.081

[14:51:53]
SYSCALL                   COUNT        TIME (us)
pread                      1193      648945.069
pwrite                     1162      337197.216
```

syscount作用于TRACEPOINT_PROBE(raw_syscalls, sys_enter)，用于分析目标进程的系统调用的频率和延迟，界定问题来自于用户态、内核态。对于Direct IO，pread/pwrite的延迟则直接反映IO延迟。

iostat

火山引擎

```
root@bytedance:/usr/share/bcc/tools# iostat -x 1 /dev/vdb
Linux 6.0.0-rc2.bm.1-amd64 (bytedance)  09/28/2022      _x86_64_        (8 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          1.55    0.01    4.83   38.83    0.00   54.77

Device            r/s     w/s     rkB/s     wkB/s   rrqm/s   wrqm/s  %rrqm  %wrqm r_await w_await aqu-sz rareq-sz
 wareq-sz  svctm  %util
vdb          7429.19 7438.05 950881.08 952070.97     0.00     0.00   0.00   0.00    0.51    0.15   4.87   127.99
   128.00   0.04  64.26

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          1.81    0.00    7.12   65.33    0.00   25.74

Device            r/s     w/s     rkB/s     wkB/s   rrqm/s   wrqm/s  %rrqm  %wrqm r_await w_await aqu-sz rareq-sz
 wareq-sz  svctm  %util
vdb          12687.00 12827.00 1623936.00 1641984.00     0.00     0.00   0.00   0.00    0.50    0.09   7.55   12
8.00   128.01   0.04 100.00
```

iostat命令用于分析磁盘设备的IO概况，其中util不再准确，但是能反映一部分问题。

biolatency

火山引擎

```
/usr/share/bcc/tools# ./biolatency -D -e 1
Tracing block device I/O... Hit Ctrl-C to end.

disk = 'vdb'
     usecs               : count     distribution
         0 -> 1          : 0        |                                        |
         2 -> 3          : 0        |                                        |
         4 -> 7          : 0        |                                        |
         8 -> 15         : 0        |                                        |
        16 -> 31         : 12       |                                        |
        32 -> 63         : 7610     |****************************************|
        64 -> 127        : 5162     |**************************              |
       128 -> 255        : 583      |***                                     |
       256 -> 511        : 7106     |*************************************   |
       512 -> 1023       : 5616     |*****************************           |
      1024 -> 2047       : 238      |*                                       |
      2048 -> 4095       : 10       |                                        |
```

biolatency基于Linux block layer，作用于内核函数blk_account_io_start&blk_account_io_done，用于统计每一个IO的发起、完成时间，精确统计每个IO请求的延迟，并查看延迟直方图。

lsblk & virtiostat & irqtop

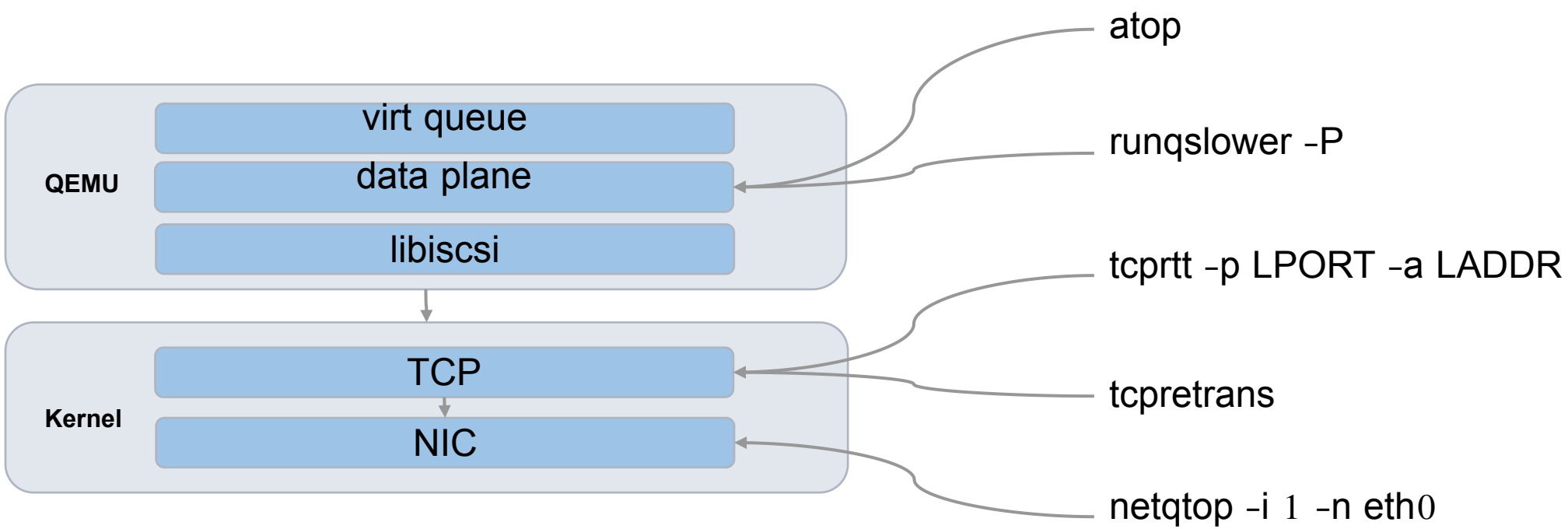
火山引擎

```
root@bytedance:/usr/share/bcc/tools# lsblk --virtio
NAME TYPE TRAN    SIZE RQ-SIZE  MQ
vda  disk virtio   50G     256   1
vdb  disk virtio  256G     128   8
```

```
root@bytedance:/usr/share/bcc/tools# ./virtiostat 1
Tracing virtio devices statistics ... Hit Ctrl-C to end.
--------
        Driver   Device    VQ Name  In SGs Out SGs          In BW         Out BW
   virtio_net  virtio0   output.8       0       2             0            380
   virtio_net  virtio0   output.7       0       1             0            210
   virtio_blk  virtio3      req.0       7      12             7          98416
   virtio_blk  virtio4      req.1    6556    6593     284823839      289739248
   virtio_blk  virtio4      req.6    1381    1400      59507615       62011888
   virtio_blk  virtio4      req.4    2990    3043     128321499      135298480
   virtio_blk  virtio4      req.7    3645    3657     158730626      160340000
   virtio_blk  virtio4      req.2   10793   10873     468065334      478659424
   virtio_blk  virtio4      req.0    4412    4429     192023427      194295856
   virtio_blk  virtio4      req.5    7582    7487     335418271      323041776
   virtio_blk  virtio4      req.3    6848    6970     293868030      309927904
```

lsblk --virtio展示队列数；virtiostat命令查看virtio设备的多队列是否数据均匀；irqtop展示中断是否均匀。

diagnosis in host initiator



atop

火山引擎

PID	TID	SYSCPU	USRCPU	RDELAY	BDELAY	VGROW	RGROW	RDDSK	WRDSK	ST	EXC	THR	S	CPUNR	CPU	CMD 1/66
1121483	-	0.94s	8.06s	0.00s	0.00s	0B	0B	0B	4.0K	--	-	17	S	45	909%	qemu-system-x8
1121483	1121483	0.02s	0.01s	0.00s	0.00s	0B	0B	0B	0B	--	-	1	S	45	3%	qemu-system-x8
1121483	1121606	0.00s	0.00s	0.00s	0.00s	0B	0B	0B	4.0K	--	-	1	S	154	0%	IO iothread1
1121483	1121607	0.92s	0.09s	0.00s	0.00s	0B	0B	0B	0B	--	-	1	R	60	100%	IO iothread2
1121483	1121685	0.01s	0.99s	0.00s	0.00s	0B	0B	0B	0B	--	-	1	R	127	100%	CPU 0/KVM
1121483	1121687	0.00s	1.00s	0.00s	0.00s	0B	0B	0B	0B	--	-	1	R	27	100%	CPU 1/KVM
1121483	1121689	0.01s	1.00s	0.00s	0.00s	0B	0B	0B	0B	--	-	1	R	143	100%	CPU 2/KVM
1121483	1121691	0.00s	1.00s	0.00s	0.00s	0B	0B	0B	0B	--	-	1	R	9	100%	CPU 3/KVM
1121483	1121693	0.00s	1.00s	0.00s	0.00s	0B	0B	0B	0B	--	-	1	R	12	100%	CPU 4/KVM
1121483	1121695	0.00s	1.00s	0.00s	0.00s	0B	0B	0B	0B	--	-	1	R	153	100%	CPU 5/KVM
1121483	1121696	0.01s	0.99s	0.00s	0.00s	0B	0B	0B	0B	--	-	1	R	161	100%	CPU 6/KVM
1121483	1121698	0.00s	0.99s	0.00s	0.00s	0B	0B	0B	0B	--	-	1	R	21	100%	CPU 7/KVM

在atop主面板上按y键，观察iothread的RDELAY，用于判断IO线程是否出现了CPU抢占。在有些场景下，per thread的RDELAY非常必要，例如：一个进程有40个threads，平均RDELAY是1%似乎问题不明显。但是如果所有的RDELAY时间来自于一个thread，就会出现较严重的问题。

runqslower

火山引擎

```
/usr/share/bcc/tools# ./runqslower -P
Tracing run queue latency higher than 10000 us
TIME     COMM             TID           LAT(us) PREV COMM        PREV TID
15:48:09 qemu-system-x86  1121483         12262 lspci            1787366
```

通过runqslower -P，可以分析出来IO线程的精确的调度延迟，以及同一个CPU上运行的上一个任务，大概率可以抓取到是哪个进程影响的。

tcprtt & tcpretrans

火山引擎

```
/usr/share/bcc/tools# tcprtt -i 1 -p 55998 -a 127.0.0.1 -P 3260
Tracing TCP RTT... Hit Ctrl-C to end.

All Addresses = *******
     usecs               : count     distribution
         0 -> 1          : 0        |                                        |
         2 -> 3          : 0        |                                        |
         4 -> 7          : 0        |                                        |
         8 -> 15         : 104      |                                        |
        16 -> 31         : 46977    |****************************************|
        32 -> 63         : 4782     |****                                    |
        64 -> 127        : 10       |                                        |
       128 -> 255        : 8        |                                        |
       256 -> 511        : 2        |                                        |
```

使用tcprtt判断网络延迟，如果tcprtt稳定且较低，那么可以排除网络因素；

如果网络延迟较高，或者存在延迟突刺，那么IO延迟也会受到响应的影响。接下来则可以使用tcpretrans查看TCP重传情况。

特别需要注意的是，tcprtt不等同于ping延迟，因为有ECMP的缘故。

netqtop & irqtop

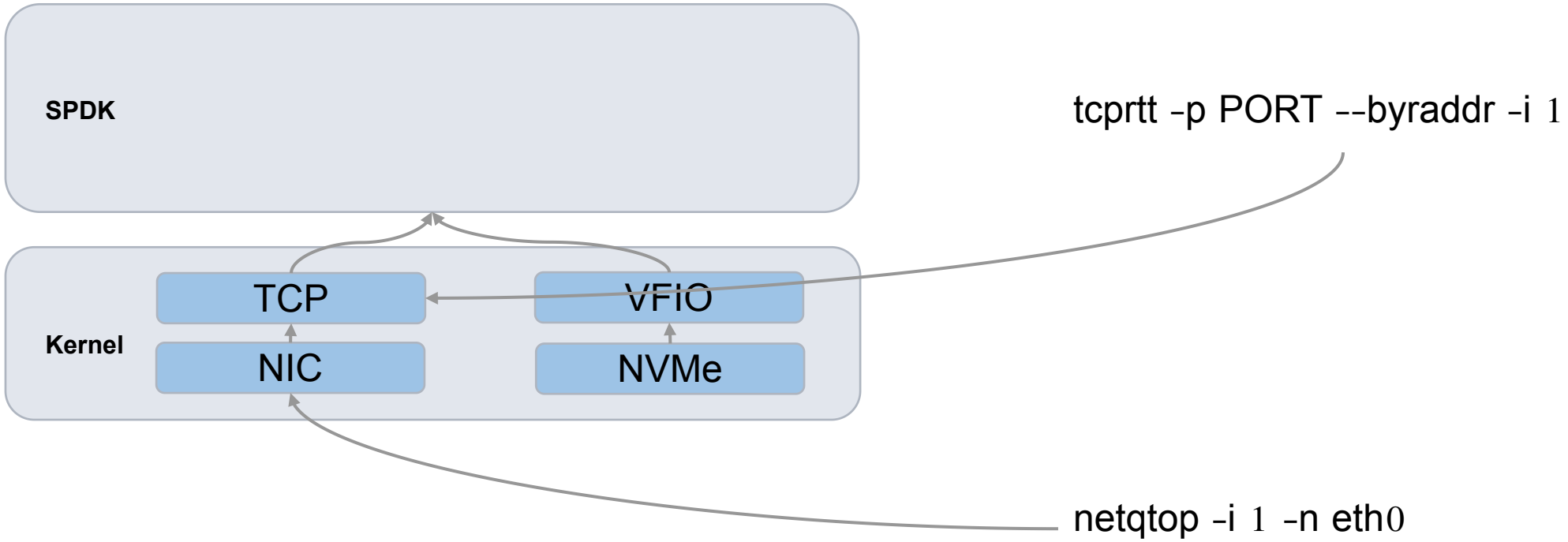
火山引擎

```
irqtop | total: 11429961582 delta: 99264 | n2-016-101 | 2022-09-29 11:04:27+08:00

       cpu0 cpu1 cpu2 cpu3 cpu4 cpu5 cpu6 cpu7 cpu8 cpu9 cpu10 cpu11 cpu12 cpu13 cpu14 cpu15 cpu16 cpu17 cpu18 c
 %irq:  2.1  0.6  0.6  0.8  0.6  0.6  0.6  0.5  0.5  0.6   0.7   0.6   0.6   0.9   0.5   0.6   0.5   0.5   0.8 0
%delta:  3.4  0.9  0.2  0.7  0.3  0.1  0.9  0.2  0.1  0.1   0.0   0.1   0.1   0.1   0.0   0.5   0.6   0.6   0.3 0

        IRQ      TOTAL      DELTA NAME
      TIMER 5450011430      56271 timer softirq
        RCU 2963063530      25094 RCU softirq
      SCHED 1957651103      15461 schedule softirq
     NET_RX 1000166910       1401 network receive softirq
    TASKLET   59044416       1037 normal priority tasklet softirq
     NET_TX      18229          0 network transmit softirq
      BLOCK       5958          0 block device softirq
         HI          6          0 high priority tasklet softirq
   IRQ_POLL          0          0 IO poll softirq
    HRTIMER          0          0 high resolution timer softirq
```

使用irqtop --softirq分析网络RX，以及网卡的中断情况。分析是否存在热点、硬件中断、softirq是否均匀等。

使用netqtop可以观察网卡队列的流量情况。

diagnosis in target



tcprtt

tcprtt -p PORT --byraddr -i 1通常使用在server/target端。例如iSCSI场景下，在target端执行：

tcprtt -p 3260 --byraddr

对访问本地3260端口的TCP连接进行区分，每个连接单独输出tcprtt信息。那么则可以清晰地看到每个连接的tcprtt延迟情况：如果部分连接的tcprtt异常，则可以尝试寻找规律，是不是在同一个TOR下（同一个S1交换机下）；如果所有的tcprtt异常，则可以尝试分析一下本地的网络情况。

killsnoop

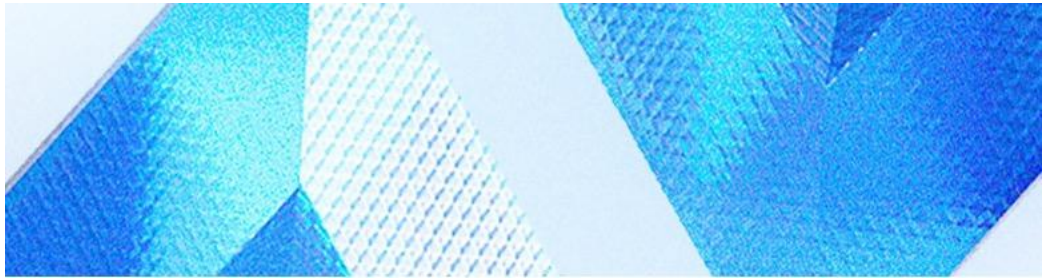
火山引擎

killsnoop作用于kill syscall，用来追踪source PID向target PID发送Sig信号。用于发现进程被异常杀掉的问题。

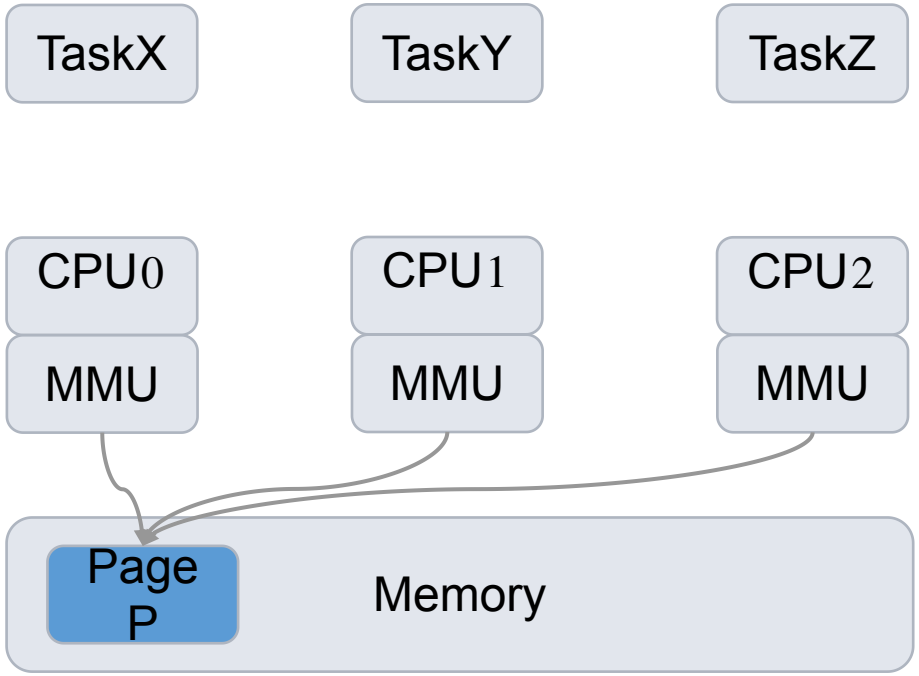
在实际使用中，服务器上通常具有128甚至更多的CPU数量，进程数量也异常多。为了避免log flood，增加killsnoop -T PID，用于过滤特定的target PID。只对我们关心的某进程进行观测。例如killsnoop观察target服务的异常退出。

02

TLB Shootdown案例



TLB Shootdown原理



ProcessP中包含3个线程：TaskX、TaskY和TaskZ。
CPU0、CPU1和CPU2使用同一个页表（PGD为PageP）。

TaskX执行madvise(,,MADV_DONTNEED)释放内存时，除释放内存页面外，还需要刷新TLB：

- 使用invlpg指令清理TLB缓存项。
- 使用IPI，清理CPU1和CPU2的TLB缓存项。

频繁的TLB shootdwon严重影响性能！

funccount

火山引擎

- irqtop命令可以观测到整机的TLB shootdown的频率，再进一步使用funccount判断TLB shootdown：

/usr/share/bcc/tools/funccount -i 1 zap_page_range

- 或者对特定的PID追踪：

/usr/share/bcc/tools/funccount -i 1 zap_page_range -p PID

- 再或者对特定的CPU追踪：

/usr/share/bcc/tools/funccount -i 1 zap_page_range -c CPU

trace

火山引擎

- 确认TLB shootdown问题后，进一步抓取madvise的用户态trace：

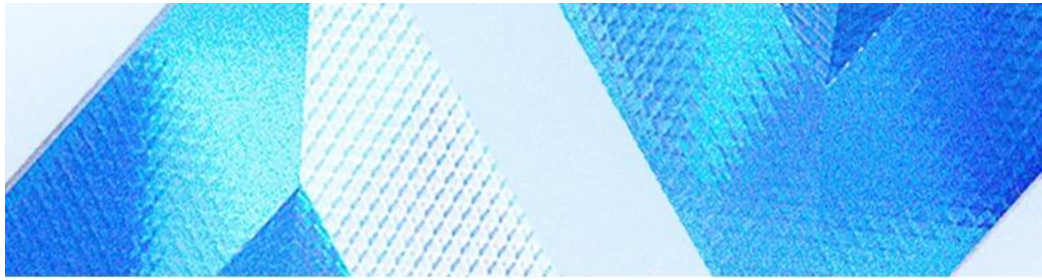
/usr/share/bcc/tools/trace -U 'c:madvise' -p PID -M 10

PID TID COMM FUNC
20119 20123 jemalloc_bg_thd madvise
b'__GI_madvise+0x0 [libc-2.31.so]'

- trace也支持dump内核态stack，同时使用-A参数对堆栈进行聚合，更好的分析热点函数。

03

火山引擎对bcc的社区贡献



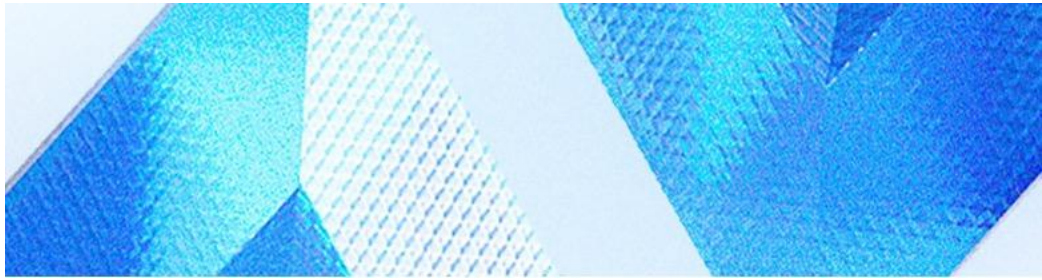
火山引擎对bcc的社区贡献

火山引擎

- 贡献了tcprtt、virtiostat和kvmexit。
- 增强了funccount、trace、tcpretrans、runqlen、runqslower等工具。
- 修复了工具中的若干BUG。

04

bcc工具集成



veLinux对bcc更好的支持

火山引擎

veLinux 是字节跳动在操作系统技术上的长期积累和沉淀，旨在给客户提供稳定、高性能、安全、易用的云上操作系统，同时输出系统安装、部署、升级和补丁修复等全生命周期的完整解决方案。

同时，在基础工具支持上，始终保持了大量的投入和开发。在veLinux上，支持最新的稳定版本bcc，体验bcc的新功能，提升debug效率。

火山引擎

Thanks!