第三届 eBPF开发者大会

www.ebpftravel.com

eBPF-交流研讨

中国·西安

bperf：利用eBPF优化perf子系统的效率

刘松

2025年4月19日

perf子系统简介
perf子系统在数据中心场景遇到的问题
bperf: bpf + perf
bperf在容器监控场景的应用
bperf在线程监控场景的应用

- perf子系统简介
- perf子系统在数据中心场景遇到的问题
- bperf: bpf + perf
- bperf在容器监控场景的应用
- bperf在线程监控场景的应用

perf子系统简介

利用硬件计数器实现性能检测的功能
计数功能：记录指定时间内的事件数量：指令数，缓存访问数
采样功能：产生中断（NMI）采样程序栈

bperf：用eBPF优化perf子系统的计数功能

perf子系统简介

性能监控工具
perf, bpftrace

性能监控守护进程

用户进程自监控

sys_perf_event_open => perf_event

内核perf子系统

硬件计数器

硬件计数器

硬件计数器

perf_event的上下文

- Per CPU perf_event
  - 监控指定CPU的perf event
  - 需要更高权限
- Per task perf_event
  - 跟随用户线程的perf event
  - 在进程切换时加入当前CPU
  - 无需高级权限
  - 适用于用户进程自监控

perf子系统简介

 perf子系统在数据中心场景遇到的问题

bperf: bpf + perf

bperf在容器监控场景的应用

bperf在线程监控场景的应用

perf子系统在数据中心场景

性能监控守护进程，24小时不间断监控
用户进程进行自我监控
各种维度的监控：全系统，容器，进程，线程，单个请求

perf子系统在数据中心场景遇到的问题

1. 不同维度监控同一指标，占用额外硬件计数器
2. 在上下文切换重新配置硬件计数器，增加上下文切换延迟
3. 需要很多perf_event， 占用内存

perf子系统在数据中心场景遇到的问题

硬件计数器有限：每个CPU只有6到11个可用计数器
需要监控的指标很多：指令数，时钟周期，缓存访问，内存访问，...
对常用指标，如指令数，需要监控各种维度。每个维度都需要独立的perf_event和硬件计数器。

当perf_event多于硬件计数器。内核会使用基于中断的时分复用。精度降低，系统开销增大。

```
# perf stat -e cycles,branch-misses,iTLB-loads -C 3 -I 1000
#           time             counts unit events
     1.000154552         52,768,266      cycles             (80.02%)
     1.000154552            104,864      branch-misses      (80.02%)
     1.000154552              4,731      iTLB-loads         (79.97%)
```

perf子系统在数据中心场景遇到的问题

针对容器（cgroup）的监控，使用per CPU perf_event，效率低
每个被监控容器，每个CPU，每个指标，都需要单独的perf_event

128个CPU，16个容器，2个指标（周期数，指令数）-> 4096个perf_event
512个CPU，64个容器，4个指标 -> 131072个perf_event

每次容器切换，需要重新设置硬件计数器，增加上下文切换延迟。

perf子系统简介

perf子系统在数据中心场景遇到的问题

 bperf: bpf + perf

bperf在容器监控场景的应用

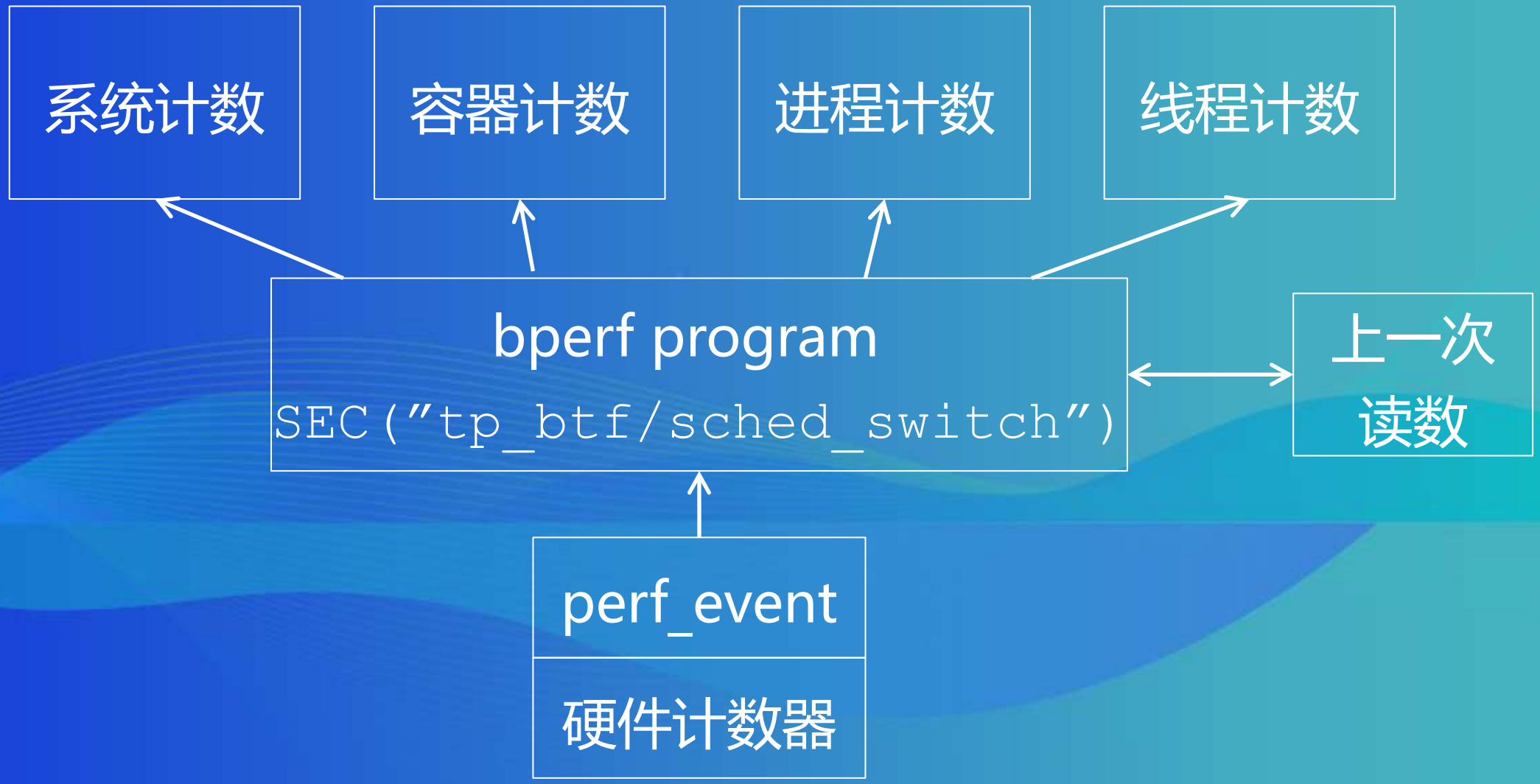
bperf在线程监控场景的应用

bperf：bpf + perf

让同一指标在不同维度的监控共享perf_event和硬件计数器

在每个CPU开启perf_event和硬件计数器
不同维度的监控共享perf_event和硬件计数器
在上下文切换时，用eBPF程序读取计数器数值，并把结果统计在不同维度的输出空间（eBPF map）：全系统，容器，线程，等。
用户通过eBPF map读取对应维度的监控数值

bperf：bpf + perf



perf子系统简介

perf子系统在数据中心场景遇到的问题

bperf: bpf + perf

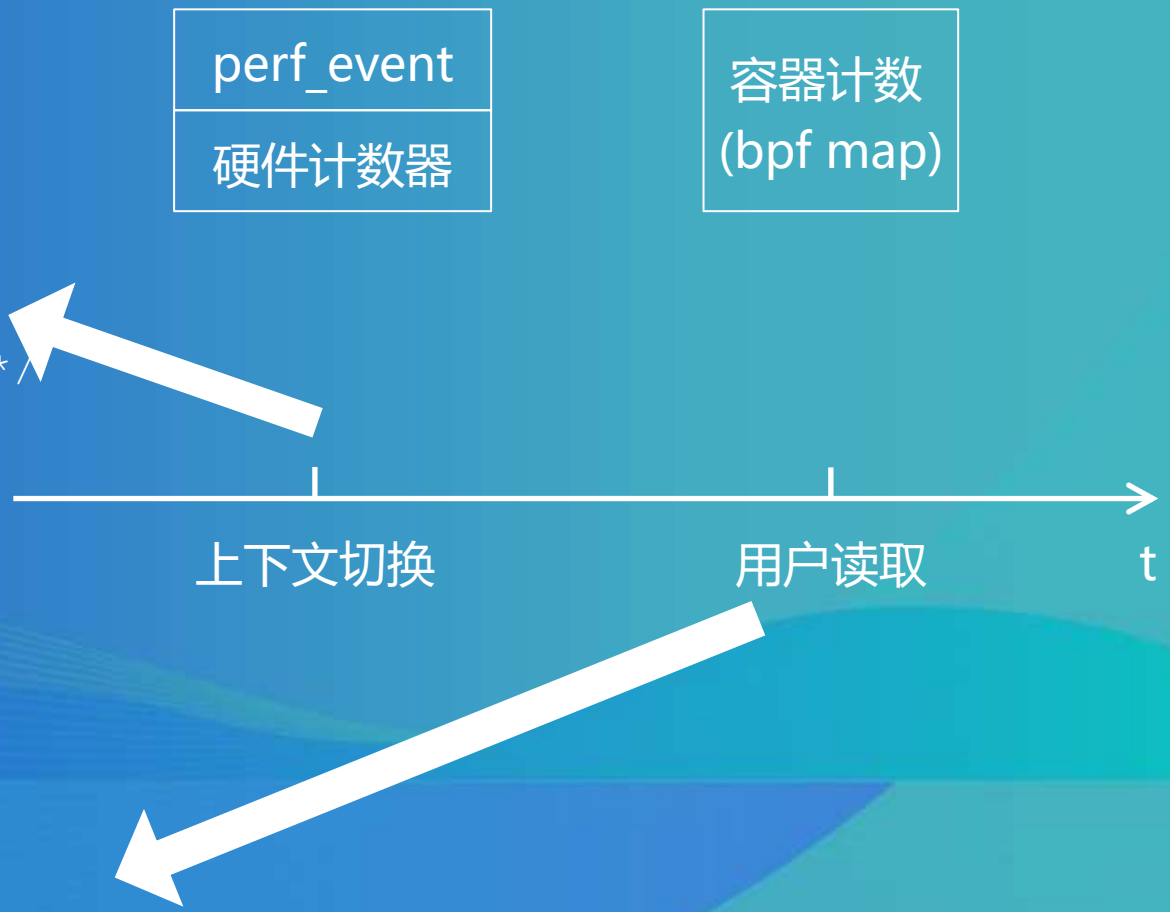
 bperf在容器监控场景的应用

bperf在线程监控场景的应用

bperf在容器监控场景的应用

```
SEC("tp_btf/sched_switch")  {
        /* 读取当前perf_event读数 */
        /* 增加读数 = 当前读数 - 上一次读数 */
        /* 把增加的读数计入前一个线程的容器的输出 */
}
```

```
/* 在用户态读取输出之前，对每一个CPU调用 */
/* sys_bpf(BPF_PROG_TEST_RUN, CPU_ID) */
SEC("raw_tp/") {
        /* 读取当前perf_event读数 */
        /* 增加读数 = 当前读数 - 上一次读数 */
        /* 把增加的读数计入当前线程的容器的输出 */
}
```



bperf在容器监控场景的应用

每个perf_event内存开销：struct perf_event + struct file, ~2kB

cgroup event

128个CPU，16个容器，2个指标 -> 4096个perf_event, ~8MB

512个CPU，64个容器，4个指标 -> 131072个perf_event, ~256MB

bperf event

128个CPU，16个容器，2个指标 -> 256个perf_event, ~512kB

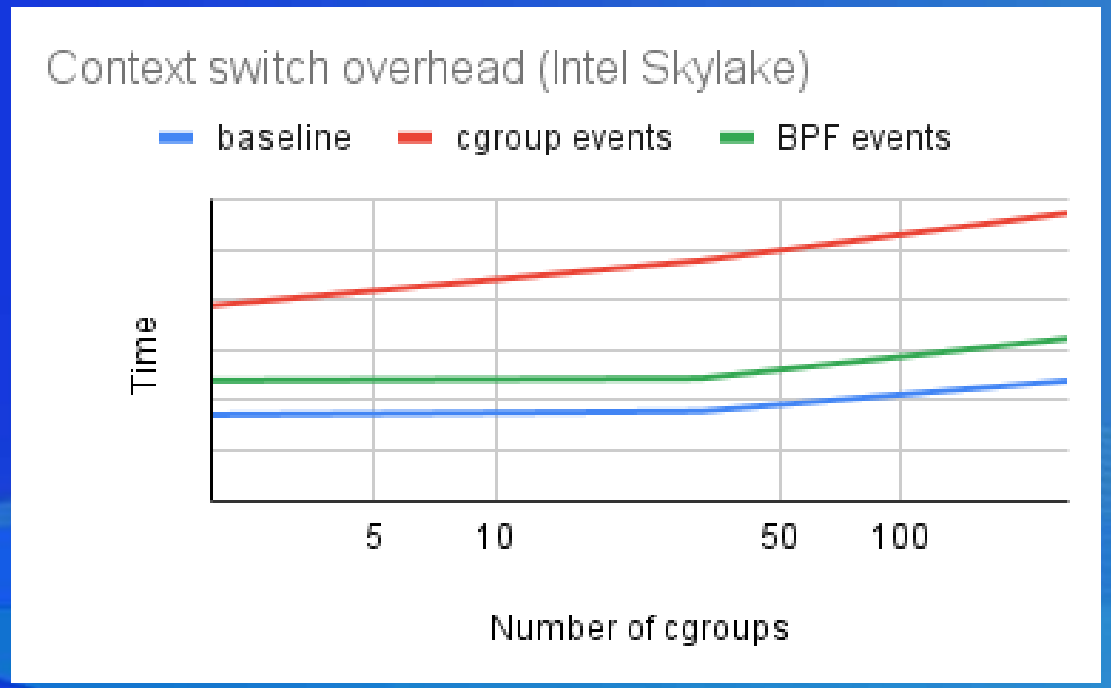
512个CPU，64个容器，4个指标 -> 2048个perf_event, ~4MB

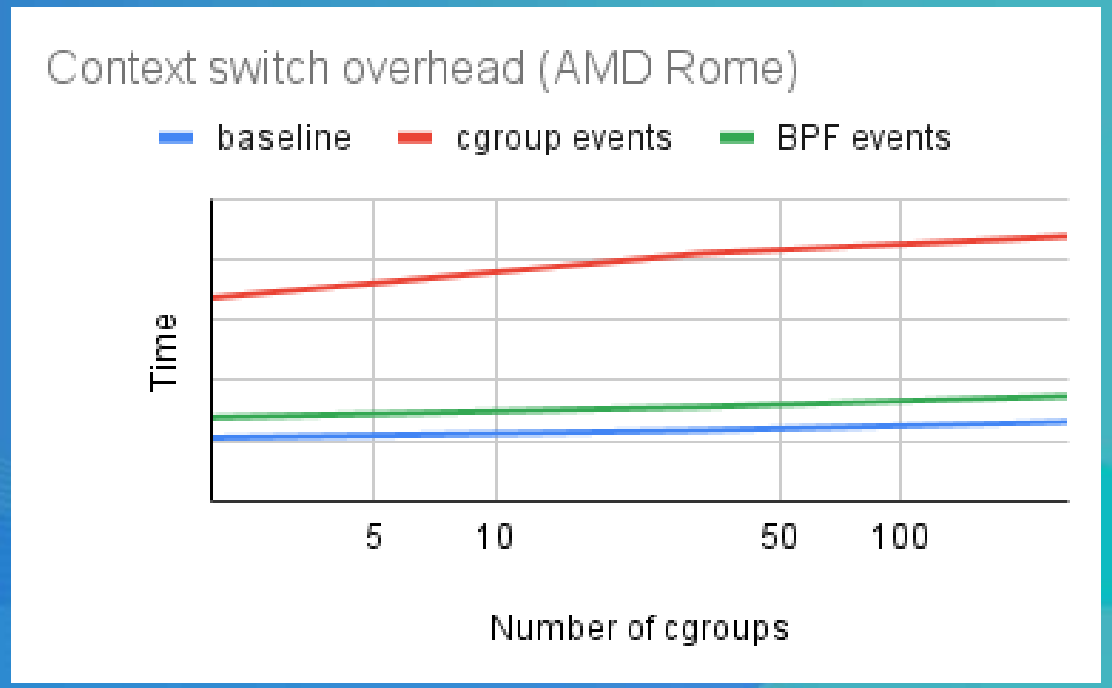
上下文切换延迟

cgroup perf_event：每次cgroup切换都需要重新设置硬件计数器

bperf：在cgroup切换（或者上下文切换）只需读取硬件计数器，从而降低上下文切换延迟

bperf在容器监控场景的应用





上下文切换延迟

http://vger.kernel.org/~acme/bpf/plumbers-2021-Adding-features-to-perf-using-BPF/

perf子系统简介

perf子系统在数据中心场景遇到的问题

bperf: bpf + perf

bperf在容器监控场景的应用

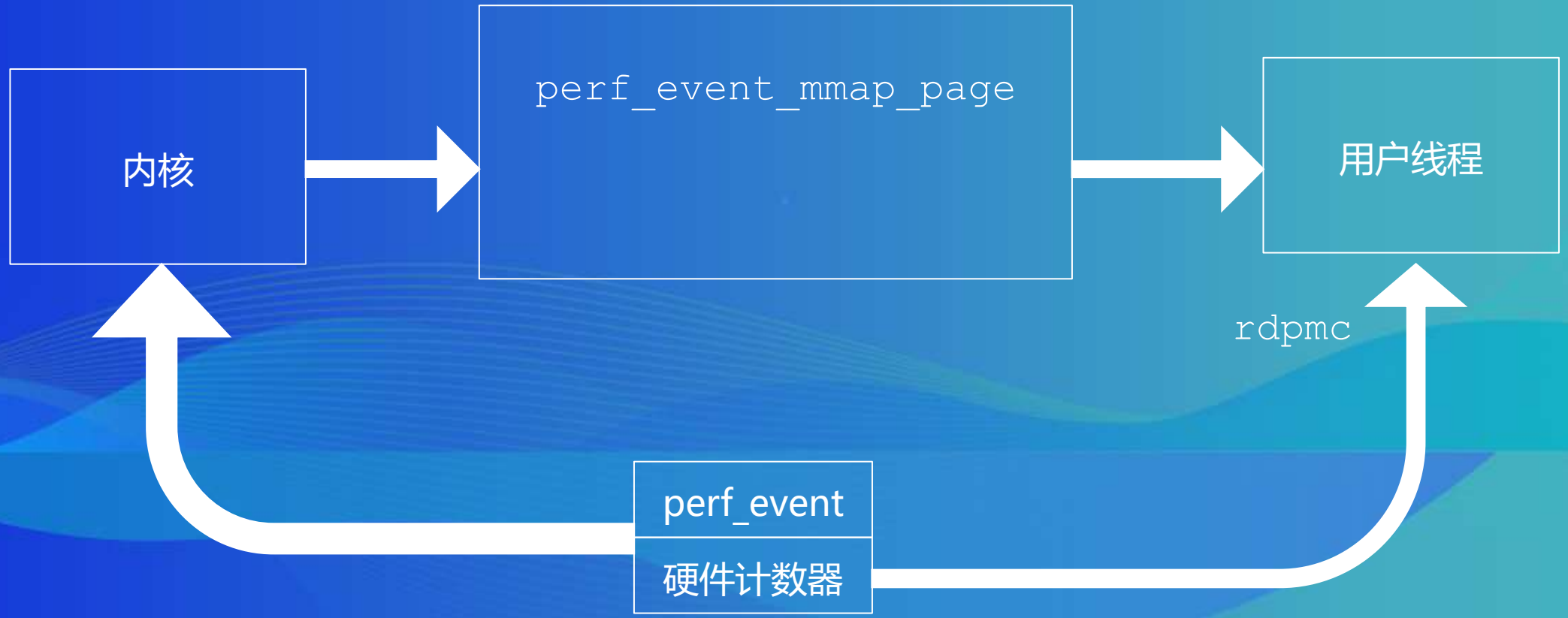
bperf在线程监控场景的应用

用户线程自我监控场景

针对单一用户请求（request）的监控
嵌入用户线程，对延迟敏感
通过系统调用读取bpf map，增加数百纳秒延迟，不能满足要求

为满足这个需求，perf子系统提供了用mmap+rdpmc代替系统调用的使用方式

perf_event提供的mmap+rdpmc方案



perf_event提供的mmap+rdpmc方案

```
struct perf_event_mmap_page {
        /* ... */
        __u32   lock;                   /* seqlock */
        __u32   index;                  /* 硬件计数器编号 */
        __s64   offset;                 /* 对rdpmc读数的修正值 */
        __u64   time_enabled;           /* perf_event启用时间 */
        __u64   time_running;           /* perf_event计数时间 */
        /* rdpmc的相关信息，用于修正rdpmc结果 */
        /* rdtsc的相关信息，用于修正time_enabled和time_running */
};
```

最终读数=[offset + rdpmc(index)] * adjusted_time_enabled / adjusted_time_running

perf_event提供的mmap+rdpmc方案

解决了系统调用的延迟问题，单次读取计数通常只需要<100ns

没有解决perf子系统的其他问题

1. 不同维度监控同一指标，占用额外硬件计数器
2. 在上下文切换重新配置硬件计数器，增加上下文切换延迟
3. 需要很多perf_event， 占用内存

bperf提供mmap+rdpmc方案

具有root权限的守护进程加载perf_event和bperf，并把必要的bpf map通过bpffs分享给用户进程

用户进程mmap bpf map，然后利用这些数据进行自我监控

用户进程不需要额外的perf_event，从而克服了perf解决方案的相应问题

bperf提供mmap+rdpmc方案

用bpf array map实现mmap功能
用iter/task_file实现用户态fd到内核perf_event的查找
监控x86_pmu_enable（或armpmu_enable），从而正确处理perf_event的时分复用

附录：开源的bperf程序

内核源码：tools/perf/util/bpf_skel/
GitHub：facebookincubator/dynolog，
hbt/src/perf_event/bpf/

第三届 eBPF开发者大会

www.ebpftravel.com

eBPF-交流研讨

中国·西安