第二届中国eBPF开发者大会

WWW.ebpftravel.com

kvm_watcher-基于eBPF的KVM性能洞察工具

南帅波（西安邮电大学）

中 国 · 西 安

个人简介：

南帅波，西安邮电大学陈莉君老师研二学生，内核之旅社区成员，研究方向为Linux内核，内核虚拟化技术等。

github主页：
https://github.com/nanshuaibo

CSDN主页：
https://blog.csdn.net/weixin_46324627?spm=1000.2115.3001.5343

kvm_watcher项目地址：
https://github.com/nanshuaibo/lmp/tree/develop/eBPF_Supermarket/kvm_watcher

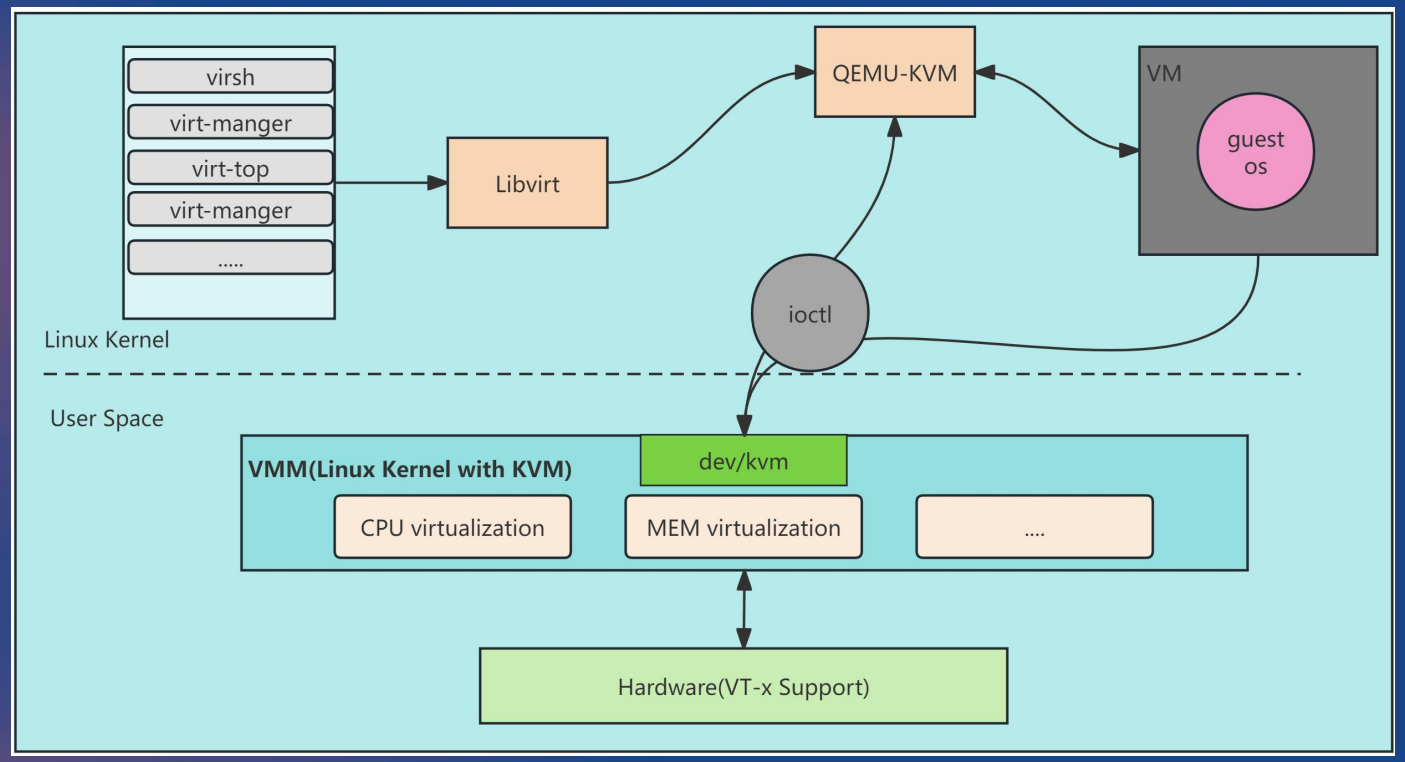
bgb
陕西 咸阳

扫一扫上面的二维码图案，加我为朋友。

1. KVM内核虚拟化
2. kvm_watcher
3. 性能测试
4. 未来展望

1 KVM内核虚拟化

1.1KVM介绍



优势：

- 资源整合
- 灵活性高
- 降低成本
- 隔离性强

地位：

- 基于Linux内核　性能出色 生态丰富

1.2 传统观测工具

工具	特征	挑战
virsh	基于 libvirt 库，提供简单的虚拟机管理和状态监控。支持命令行操作，易于使用。	有限的监控功能，无法深入分析和调整性能。不够灵活，无法满足高级需求，可扩展性低。
kvm_stat	基于 Python 编写，主要使用 debugfs 读取数据，统计 KVM 相关事件信息。 提供了针对性的 KVM 事件统计功能。	依赖 debugfs，对操作系统版本和配置有一定要求。可能需要较高的技术水平进行使用和定制化，提取信息有限。

2 kvm_watcher

2.1kvm_watcher项目简介

`kvm_watcher`是一款基于eBPF的kvm虚拟机检测工具，其旨在使用户方便快捷在宿主机侧获取kvm虚拟机中的各种信息，报告所有正在运行的guest行为。

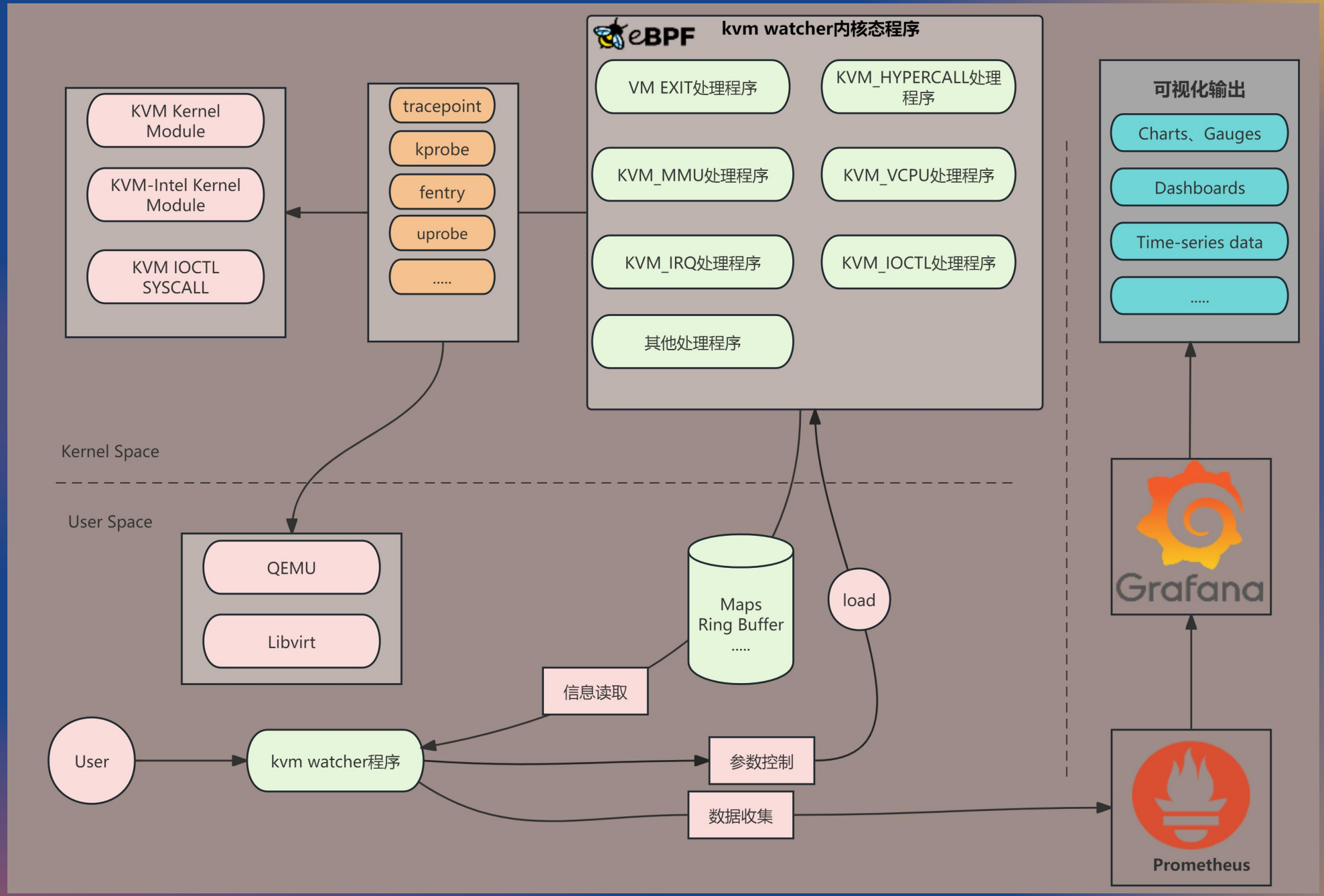
目前，其实现的功能主要包括：

- VM Exit 事件分析
- KVM mmu事件分析
- vCPU相关指标分析
- kvm中中断注入记录
- hypercall信息统计

2.2kvm_watcher项目特性

- 基于eBPF技术
- 数据全面
- 粒度更细
- 可定制化
- 易于使用
- 开源

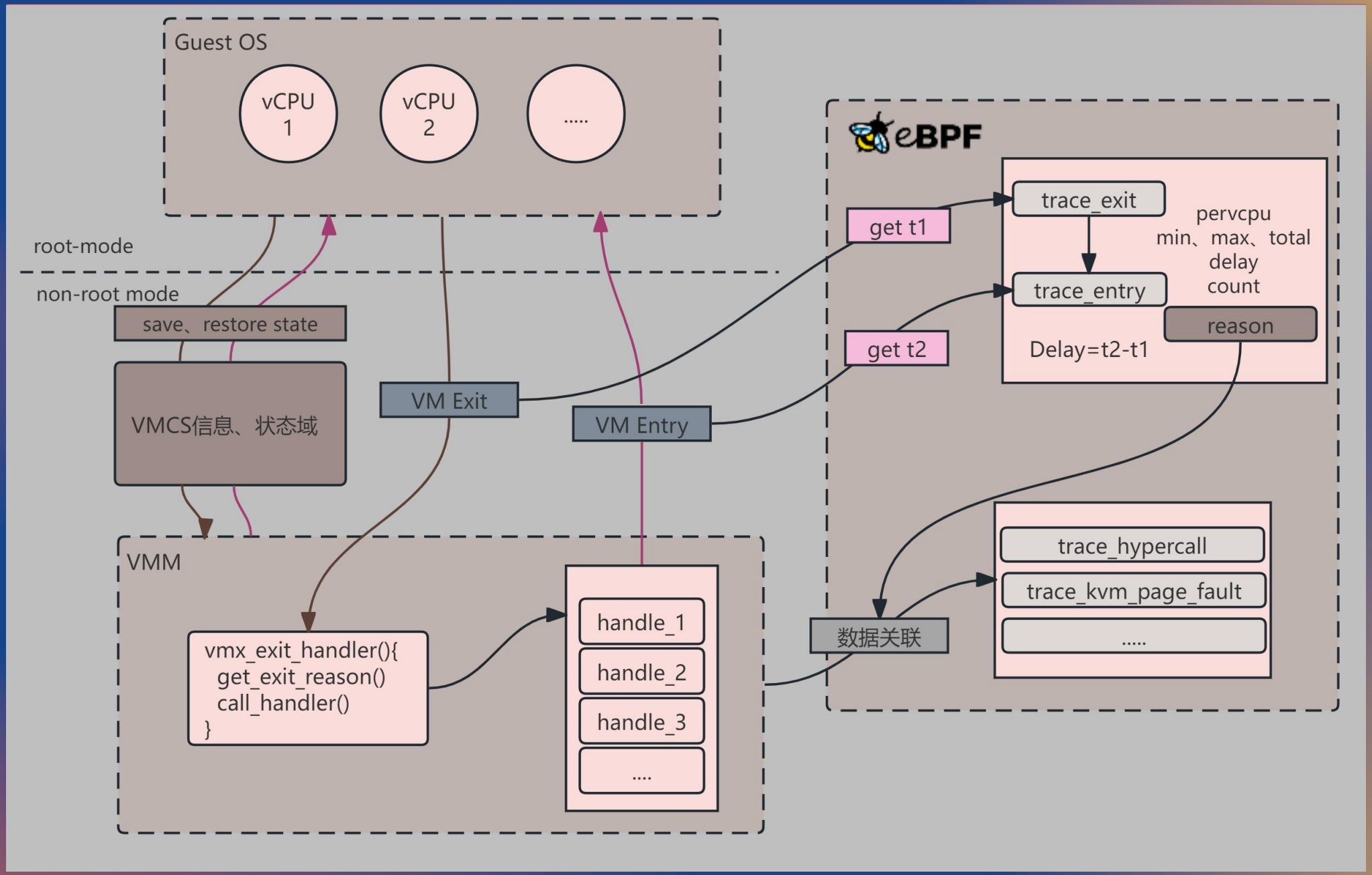
2.3 项目框图


eBPF
kvm watcher内核态程序
VM EXIT处理程序
KVM_HYPERCALL处理程序
KVM_MMU处理程序
KVM_VCPU处理程序
KVM_IRQ处理程序
KVM_IOCTL处理程序
其他处理程序
KVM Kernel Module
KVM-Intel Kernel Module
KVM IOCTL SYSCALL
tracepoint
kprobe
fentry
uprobe
.....
可视化输出
Charts、Gauges
Dashboards
Time-series data
.....
Kernel Space
User Space
QEMU
Libvirt
Maps
Ring Buffer
.....
load
Grafana
信息读取
User
kvm watcher程序
参数控制
数据收集
Prometheus

2.4子模块 kvm_exit

- VM Exit 原因统计
- VM Exit 延时分析
- VM Exit 次数计数
- 细粒度vcpu定位
- 子模块数据关联

2.4 子模块 kvm exit



2.4子模块 kvm_exit

TIME:11:37:42

KVM_EXIT

PID	TID	TOTAL_TIME	MAX_TIME	MIN_TIME	COUNT	REASON
4449	4534	0.1246	0.0170	0.0009	45	MSR_WRITE
		0.0166	0.0166	0.0166	1	LDTR_TR
		0.0102	0.0102	0.0102	1	VMWRITE
		0.0116	0.0086	0.0030	2	MSR_READ
		0.0651	0.0382	0.0061	5	VMRESUME
		0.0789	0.0126	0.0022	18	EPT_MISCONFIG
		0.0513	0.0026	0.0009	31	PREEMPTION_TIMER
		0.0115	0.0115	0.0115	1	GDTR_IDTR
		0.0084	0.0055	0.0029	2	PAUSE_INSTRUCTION
		0.9636	0.0279	0.0019	68	EXTERNAL_INTERRUPT
	4535	1.0142	0.0406	0.0075	50	EXTERNAL_INTERRUPT
		0.1063	0.0042	0.0019	35	PREEMPTION_TIMER
		0.0358	0.0313	0.0045	2	EPT_VIOLATION
		0.1429	0.0063	0.0011	49	MSR_WRITE
	4536	0.0474	0.0200	0.0135	3	EPT_MISCONFIG
		0.1084	0.0044	0.0019	35	PREEMPTION_TIMER
		0.2149	0.0086	0.0019	57	MSR_WRITE
		1.1041	0.0950	0.0098	47	EXTERNAL_INTERRUPT
		0.0210	0.0112	0.0026	3	PAUSE_INSTRUCTION
	4537	0.1281	0.0077	0.0022	37	MSR_WRITE
		0.9794	0.0336	0.0132	47	EXTERNAL_INTERRUPT
		0.0587	0.0150	0.0018	11	EPT_MISCONFIG
		0.1005	0.0046	0.0014	33	PREEMPTION_TIMER
		0.0214	0.0111	0.0103	2	PAUSE_INSTRUCTION
		0.1845	0.0823	0.0028	12	EPT_VIOLATION

vm exit时间处理详细时延信息，可定位到具体的pid及vcpu的tid

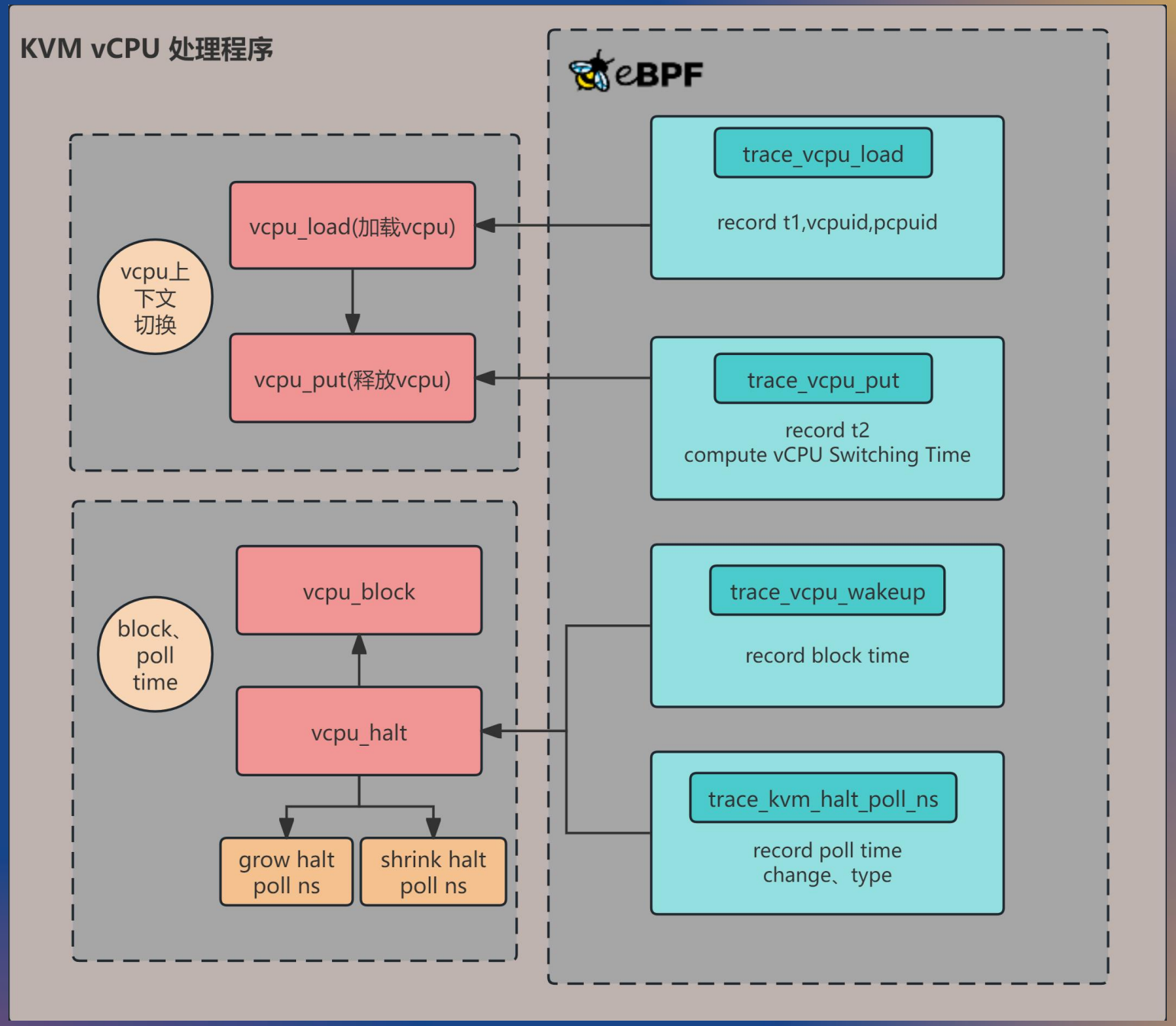
可以在host侧定位到定位到guest具体vcpu线程号

2.5 子模块 kvm_vcpu

- 精确记录 VCPU 的唤醒/挂起事件
- 统计halt poll 时间的变化
- 记录vcpu的调度情况，记录vcpu的上下文切换时间

可以通过分析 vCPU 调度信息，优化调度策略，调整虚拟机 CPU 拓扑结构或 CPU 亲和性设置,提高 vCPU 利用率.

2.5 子模块 kvm_vcpu



2.5 子模块 kvm_vcpu

vcpu上下文切换时间

高负载 VM

空闲VM

TIME:11:40:45

pid	tid	total_time	max_time	min_time	counts	vcpuid	pcpuid
4449	4539	1631.3972	732.0363	0.3243	9	5	2
4449	4541	1651.0964	415.9920	0.0174	17	7	9
4449	4536	1567.7145	479.9427	159.9706	5	2	36
4449	4544	1023.7236	511.9326	31.9687	6	10	11
4449	4538	1919.6978	831.9644	63.9123	5	4	6
4449	4537	1151.8665	543.9935	127.9301	3	3	37
4449	4543	1283.9614	1027.9930	255.9684	2	9	5
4449	4534	1967.6530	508.0021	0.0161	29	0	8
4449	4545	1503.6555	621.5516	0.0090	10	11	15
4449	4540	1311.7978	507.9800	31.9612	4	6	14
4449	4542	1535.9337	1023.9744	511.9593	2	8	27
4449	4535	2015.8935	795.9840	0.0304	18	1	38

TIME:11:41:42

pid	tid	total_time	max_time	min_time	counts	vcpuid	pcpuid
4724	5106	0.5089	0.1916	0.0764	4	7	9
4724	5091	1.9783	0.3036	0.0770	14	0	28
4724	5095	7.1847	0.6348	0.0497	28	1	1
4724	5102	1.3621	0.2903	0.0676	8	4	18
4724	5097	22.2310	0.7861	0.0746	56	2	13
4724	5098	3.3465	0.3363	0.0694	15	3	0
4724	5104	12.4416	0.4312	0.0763	55	6	39
4724	5103	1.7487	0.4068	0.1031	8	5	0

2.5 子模块 kvm_vcpu

halt poll time变化情况

halt-polling的机制保证虚拟机的vCPU线程的及时响应，但在虚拟机空载的时候，主机侧也会polling，导致主机看到vCPU所在CPU占用率比较高，而实际虚拟机内部CPU占用率并不高。

```
~/virtual/lmp/eBPF_Supermarket/kvm_watcher (develop X) ▷ sudo ./kvm_watcher -n
[sudo] nans 的密码:
TIME(ms)          COMM        PID/TID          TYPE    VCPU_ID OLD(ns)     NEW(ns)
659020018.419750  CPU 4/KVM   262306/262359    grow    4       0       --> 10000
659020018.716444  CPU 4/KVM   262306/262359    grow    4       10000   --> 20000
659020018.942041  CPU 4/KVM   262306/262359    grow    4       20000   --> 40000
659020510.195010  CPU 4/KVM   262306/262359    shrink  4       40000   --> 0
659022731.605966  CPU 1/KVM   262306/262356    grow    1       0       --> 10000
659022938.129875  CPU 1/KVM   262306/262356    shrink  1       10000   --> 0
659024021.819201  CPU 5/KVM   262306/262360    grow    5       0       --> 10000
659024126.213244  CPU 5/KVM   262306/262360    shrink  5       10000   --> 0
659027659.503923  CPU 4/KVM   262306/262359    grow    4       0       --> 10000
659027666.518084  CPU 4/KVM   262306/262359    shrink  4       10000   --> 0
659027711.258226  CPU 5/KVM   262306/262360    grow    5       0       --> 10000
659028020.691242  CPU 5/KVM   262306/262360    shrink  5       10000   --> 0
659028856.157311  CPU 1/KVM   262306/262356    grow    1       0       --> 10000
659028878.103347  CPU 1/KVM   262306/262356    shrink  1       10000   --> 0
659029060.096272  CPU 1/KVM   262306/262356    grow    1       0       --> 10000
659029262.543066  CPU 1/KVM   262306/262356    shrink  1       10000   --> 0
659030282.770579  CPU 2/KVM   262306/262357    grow    2       0       --> 10000
659030283.784202  CPU 2/KVM   262306/262357    shrink  2       10000   --> 0
```

2.6 子模块 kvm_mmu

kvm mmu子功能模块特别关注于捕捉和分析两类关键的虚拟化环境中的内存管理事件：

- EPT page fault(VM exit -> EPT VIOLATION ）
- 热迁移中的dirty page

2.6 子模块 kvm_mmu

EPT_VIOLATION

0.0144	0.0055	0.0041	3	MSR_READ
0.0253	0.0158	0.0095	2	VMCALL
0.0060	0.0030	0.0030	2	PREEMPTION_TIMER
0.4512	0.0219	0.0025	103	EPT_VIOLATION

page fault的详细信息

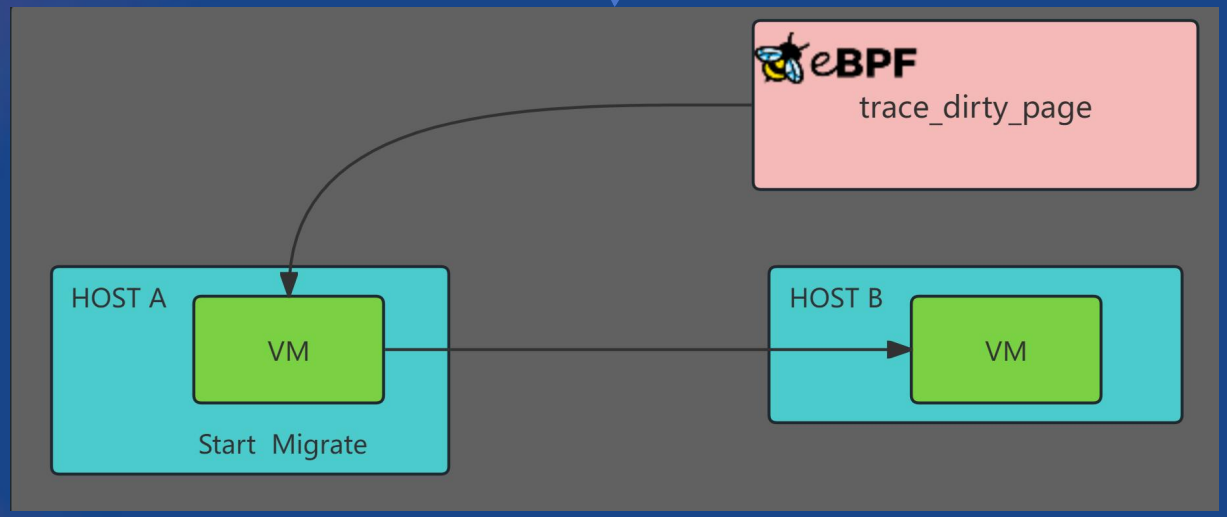
178771529.297616	CPU 1/KVM	4449	183877b18	1	3.3680	7f4577677000	2a6c00	1	User
178771529.310035	CPU 1/KVM	4449	18201bdc8	1	1.8970	7f4575e1b000	20a01b	1	User
178771529.317765	CPU 1/KVM	4449	182556e20	1	1.9450	7f4576356000	245d56	1	Write
178771529.328614	CPU 1/KVM	4449	1854d9b68	1	2.0480	7f45792d9000	25f6d9	1	User
178771529.334821	CPU 1/KVM	4449	1854da098	1	2.1720	7f45792da000	25f6da	1	User
178771529.342025	CPU 1/KVM	4449	18584c2e0	1	3.5510	7f457964c000	2a6e00	1	Write User
178771529.350638	CPU 1/KVM	4449	183f8fad8	1	3.4080	7f4577d8f000	27b800	1	User
178771529.363818	CPU 1/KVM	4449	1824252b8	1	2.1450	7f4576225000	245c25	1	User
178771529.374119	CPU 1/KVM	4449	18241edc8	1	1.9770	7f457621e000	245c1e	1	User
178771529.387159	CPU 1/KVM	4449	18241bfd0	1	1.9770	7f457621b000	245c1b	1	User

2.6 子模块 kvm_mmu

dirty_page

```
root@nans:~# virsh migrate --live --unsafe --persistent ubuntu18.04 qemu+ssh://nans@192.168.40.129/system tcp://192.168.40.129
```

使用virsh执行迁移



在虚拟机热迁移过程中，源虚拟机上的内存页在复制到目标虚拟机的同时仍然处于活动状态，任何在此过程中对这些页的修改都会导致脏页的产生。

2.6 子模块 kvm_mmu

dirty_page日志统计

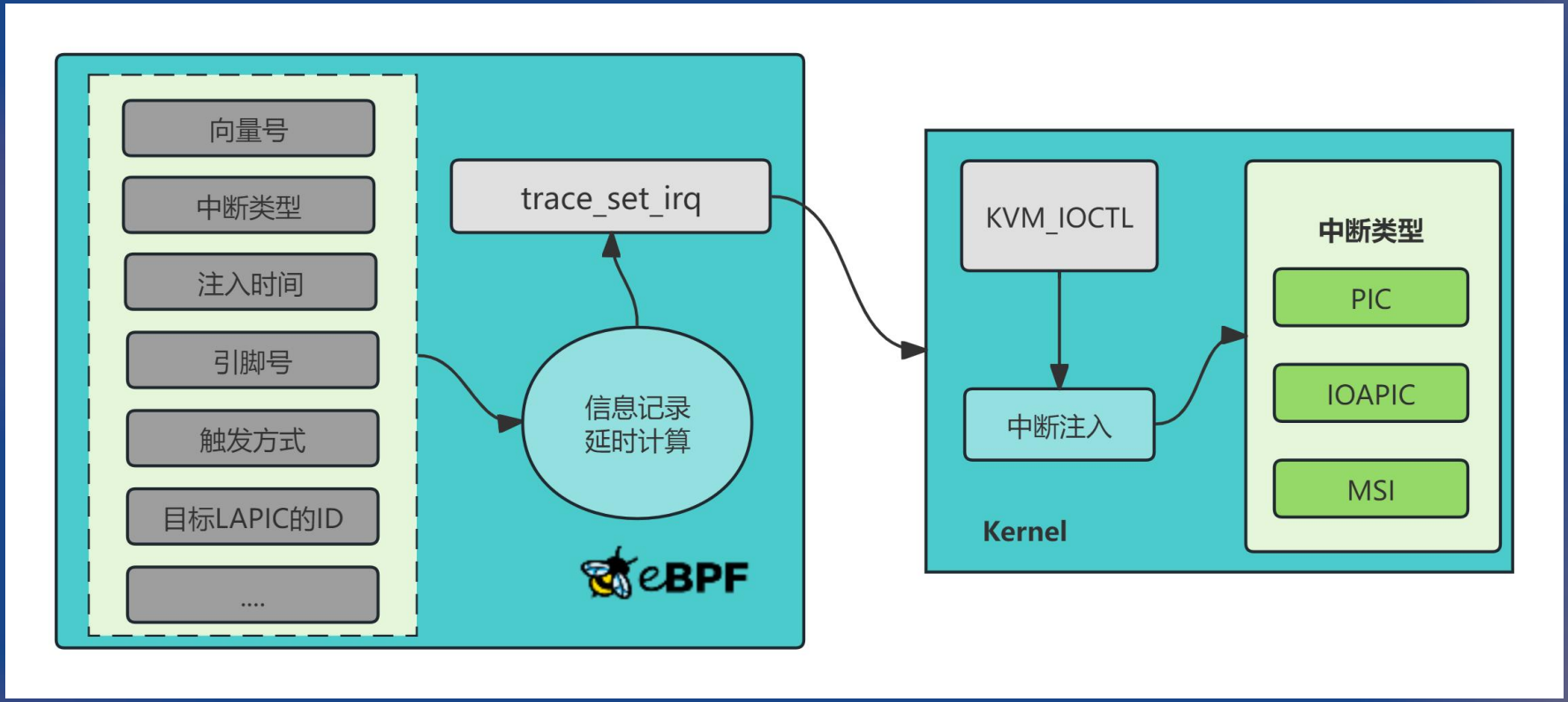
PID	GFN	REL_GFN	SLOT_ID	COUNTS
262306	7dc73	7db73	10	1933
262306	7dc33	7db33	10	1558
262306	7dcb3	7dbb3	10	1323
262306	7dd73	7dc73	10	291
262306	7dcf3	7dbf3	10	194
262306	7dd33	7dc33	10	149
262306	3568	3468	10	1
262306	31241	31141	10	1
262306	619a6	618a6	10	1
262306	1263e	1253e	10	1
262306	307c0	306c0	10	1
262306	139ed	138ed	10	1
262306	2fda7	2fca7	10	1

677311760.149175	CPU 1/KVM	262306/262356	7dc73	7db73	524032	7f16bff00000	10
677311765.940676	CPU 0/KVM	262306/262355	7dc33	7db33	524032	7f16bff00000	10
677311766.806388	CPU 1/KVM	262306/262356	7dc73	7db73	524032	7f16bff00000	10
677311767.565870	CPU 1/KVM	262306/262356	7dc73	7db73	524032	7f16bff00000	10
677311771.691104	CPU 1/KVM	262306/262356	7dc73	7db73	524032	7f16bff00000	10
677311772.407242	CPU 4/KVM	262306/262359	7dd33	7dc33	524032	7f16bff00000	10
677311775.952822	CPU 1/KVM	262306/262356	7dc73	7db73	524032	7f16bff00000	10
677311776.297101	CPU 4/KVM	262306/262359	7dd33	7dc33	524032	7f16bff00000	10
677311776.761210	CPU 3/KVM	262306/262358	7dcf3	7dbf3	524032	7f16bff00000	10
677311781.232490	CPU 1/KVM	262306/262356	7dc73	7db73	524032	7f16bff00000	10
677311783.416623	CPU 3/KVM	262306/262358	7dcf3	7dbf3	524032	7f16bff00000	10
677311784.371075	CPU 1/KVM	262306/262356	7dc73	7db73	524032	7f16bff00000	10
677311785.012778	CPU 4/KVM	262306/262359	7dd33	7dc33	524032	7f16bff00000	10
677311785.192102	CPU 3/KVM	262306/262358	7dcf3	7dbf3	524032	7f16bff00000	10
677311787.407302	CPU 4/KVM	262306/262359	7dd33	7dc33	524032	7f16bff00000	10

脏页次数统计

脏页详细信息

2.7 子模块 kvm_irq

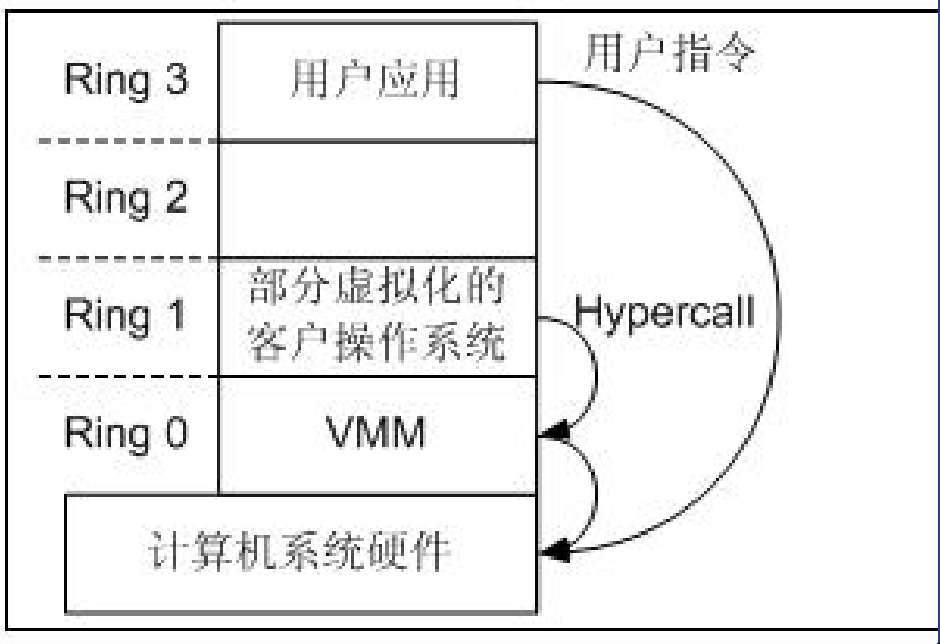


2.7 子模块 kvm_irq

```
[192.168.40.1] nans@nans:~/virtual/lmp/eBPF_Supermarket/kvm_watcher (develop *%) $ sudo ./kvm_watcher -c
[sudo] nans 的密码:
TIME(ms)          COMM           PID      DELAY    TYPE/PIN      DST/VEC   OTHERS
679611205.401270  CPU 3/KVM      262306   49182    MSI       /-  0x2/39    Fixed  |physical|edge |-     |-
679611205.693259  CPU 3/KVM      262306   3124     MSI       /-  0x2/39    Fixed  |physical|edge |-     |-
679613220.564853  CPU 3/KVM      262306   35091    MSI       /-  0x2/39    Fixed  |physical|edge |-     |-
679613220.661612  CPU 3/KVM      262306   2526     MSI       /-  0x2/39    Fixed  |physical|edge |-     |-
679615236.619273  CPU 3/KVM      262306   35637    MSI       /-  0x2/39    Fixed  |physical|edge |-     |-
679615236.693233  CPU 3/KVM      262306   2271     MSI       /-  0x2/39    Fixed  |physical|edge |-     |-
679616353.653697  qemu-system-x86 262306   28696    IOAPIC    /21 0x4/42    Fixed  |physical|level|-     |-
679616353.709370  qemu-system-x86 262306   4651     PIC slave /2  -  /-     -      |-       |level|masked|-
679616353.715180  qemu-system-x86 262306   704      IOAPIC    /10 0  /0     Fixed  |physical|edge |masked|-
679616353.718696  qemu-system-x86 262306   1106     PIC slave /2  -  /-     -      |-       |level|masked|-
679616353.720484  qemu-system-x86 262306   406      IOAPIC    /10 0  /0     Fixed  |physical|edge |masked|-
679616354.032263  CPU 4/KVM      262306   1506     IOAPIC    /21 0x4/42    Fixed  |physical|level|-     |-
679616354.038335  CPU 4/KVM      262306   2874     PIC slave /2  -  /-     -      |-       |level|masked|-
```

注入延时　中断类型　中断向量号

2.8 kvm hypercall



vm exit

0.0253	0.0158	0.0095	2	VMCALL
0.0060	0.0030	0.0030	2	PREEMPTION_TIMER
0.4512	0.0219	0.0025	103	EPT_VIOLATION

hypercall

```
Waiting hypercall ...
------------------------------------------------------------------------
TIME:13:06:54
PID          VCPU_ID      NAME         COUNTS       HYPERCALLS
964609       3            KICK_CPU     2            102
964609       2            KICK_CPU     1            132
------------------------------------------------------------------------
```

日志文件记录详细信息

PID	VCPU_ID	NAME	HYPERCALLS	ARGS
964609	3	KICK_CPU	101	apic_id:4
964609	2	KICK_CPU	132	apic_id:0
964609	3	KICK_CPU	102	apic_id:2
964609	2	SEND_IPI	133	ipi_bitmap_low:0x19,ipi_bitmap_high:0,min(apic_id):1,icr:0xfc
964609	0	KICK_CPU	193	apic_id:3
964609	4	KICK_CPU	240	apic_id:3
964609	3	KICK_CPU	103	apic_id:1
964609	4	SEND_IPI	241	ipi_bitmap_low:0x27,ipi_bitmap_high:0,min(apic_id):0,icr:0xfc
964609	4	KICK_CPU	242	apic_id:2
964609	4	KICK_CPU	243	apic_id:2
964609	4	SEND_IPI	244	ipi_bitmap_low:0x2d,ipi_bitmap_high:0,min(apic_id):0,icr:0xfc
964609	4	SEND_IPI	245	ipi_bitmap_low:0x11,ipi_bitmap_high:0,min(apic_id):1,icr:0xfc
964609	2	SEND_IPI	134	ipi_bitmap_low:0x33,ipi_bitmap_high:0,min(apic_id):0,icr:0xfc

2.9 kvm ioctl

- 记录vm的创建情况
- 记录vcpu创建情况
- vcpu运行情况
- 统计vm的mem’slot的内存区域信息
- 获取 vCPU 内存映射区域的大小
- .....

```
KVM_SET_USER_MEMORY_REGION: guest_phys_addr=f4000000, memory_size=65536K,userspace_addr=7f1697c00000
KVM_SET_USER_MEMORY_REGION: fd=16, slot=5,flags=2
KVM_SET_USER_MEMORY_REGION: guest_phys_addr=f9c14000, memory_size=8K,userspace_addr=7f16bda00000
KVM_SET_USER_MEMORY_REGION: fd=16, slot=65541,flags=2
KVM_SET_USER_MEMORY_REGION: guest_phys_addr=f9c14000, memory_size=8K,userspace_addr=7f16bda00000
KVM_CREATE_VM: fd=14
KVM_CREATE_VCPU: fd=16, vcpu_id=0
KVM_CREATE_VCPU: fd=16, vcpu_id=0
KVM_CREATE_VCPU: fd=16, vcpu_id=0
KVM_CREATE_VCPU: fd=16, vcpu_id=0
KVM_CREATE_VCPU: fd=16, vcpu_id=0
KVM_CREATE_VCPU: fd=16, vcpu_id=0
KVM_SET_USER_MEMORY_REGION: fd=16, slot=0,flags=0
```

kvm_watcher 可以利用 eBPF 技术监控 KVM ioctl 系统调用，通过统计调用频率和调用延时，分析虚拟机管理程序和用户空间程序之间的交互情况。

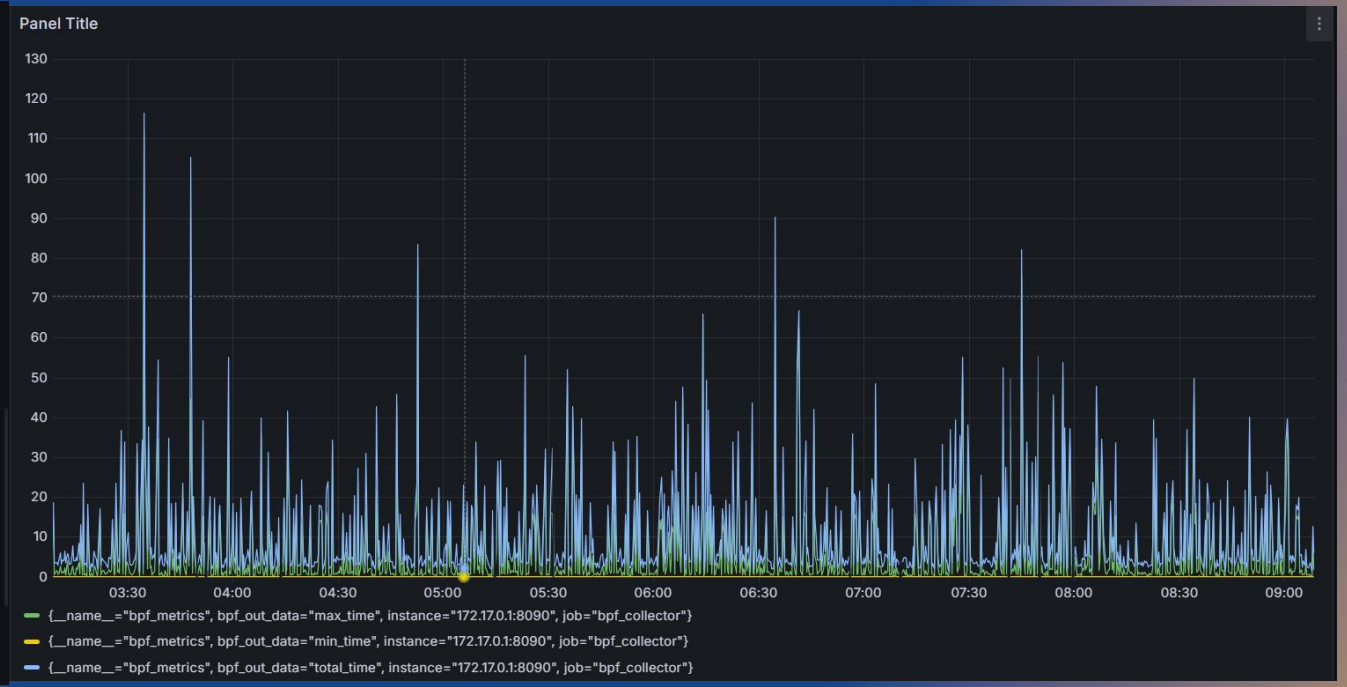
2.10 与传统工具对比

特性	传统工具（virsh、kvm_stat）	eBPF(kvm_watcher)
指标范围	有限，主要为基本资源指标	广泛，包括各种事件和指标
实时性	较差，通过查询 libvirt 守护进程，或者debugfs	优秀，直接从内核获取数据
灵活性	较差，功能相对固定	优秀，可以根据需求进行定制开发
性能开销	较低	极低，对虚拟机性能影响微乎其微

传统工具缺乏对 KVM 内部机制的可见性，难以获取细粒度性能数据。

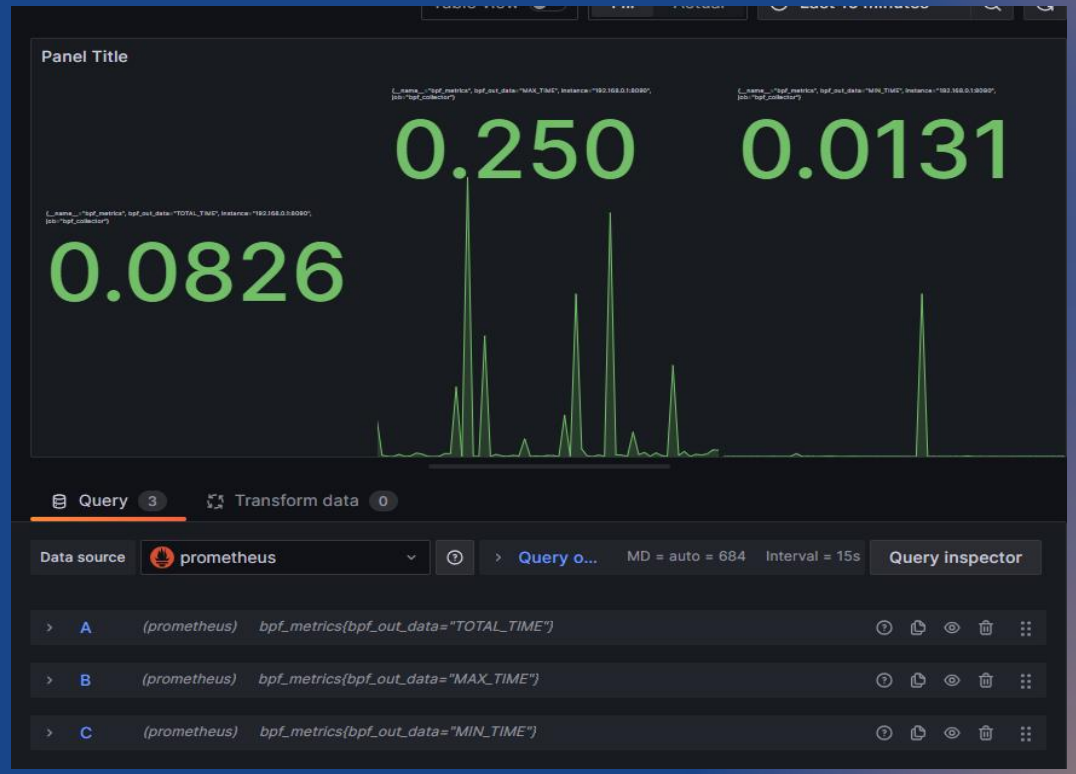
2.11 可视化结果输出


Panel Title
262360
Query 1
Transform data 0
Data source
prometheus
Query options
MD = auto = 1079
Interval = 15s
Query inspector
A
(prometheus)
bpf_metrics{bpf_out_data="TID"}


Panel Title
130
120
110
100
90
80
70
60
50
40
30
20
10
0
03:30
04:00
04:30
05:00
05:30
06:00
06:30
07:00
07:30
08:00
08:30
09:00
{__name__="bpf_metrics", bpf_out_data="max_time", instance="172.17.0.1:8090", job="bpf_collector"}
{__name__="bpf_metrics", bpf_out_data="min_time", instance="172.17.0.1:8090", job="bpf_collector"}
{__name__="bpf_metrics", bpf_out_data="total_time", instance="172.17.0.1:8090", job="bpf_collector"}

2.11 可视化结果输出


Panel Title
50
40
30
20
10
0
12:00
12:30
13:00
13:30
14:00
14:30
15:00
15:30
16:00
16:30
17:00
17:30
Query 1
Transform data 0
Alert 0
Data source
prometheus
Query o...
MD = auto = 857
Interval = 30s
Query inspector
A
(prometheus)
bpf_metrics{bpf_out_data="REASON"}


Panel Title
0.250
0.0131
0.0826
Query 3
Transform data 0
Data source
prometheus
Query o...
MD = auto = 684
Interval = 15s
Query inspector
A
(prometheus)
bpf_metrics{bpf_out_data="TOTAL_TIME"}
B
(prometheus)
bpf_metrics{bpf_out_data="MAX_TIME"}
C
(prometheus)
bpf_metrics{bpf_out_data="MIN_TIME"}

3 性能测试

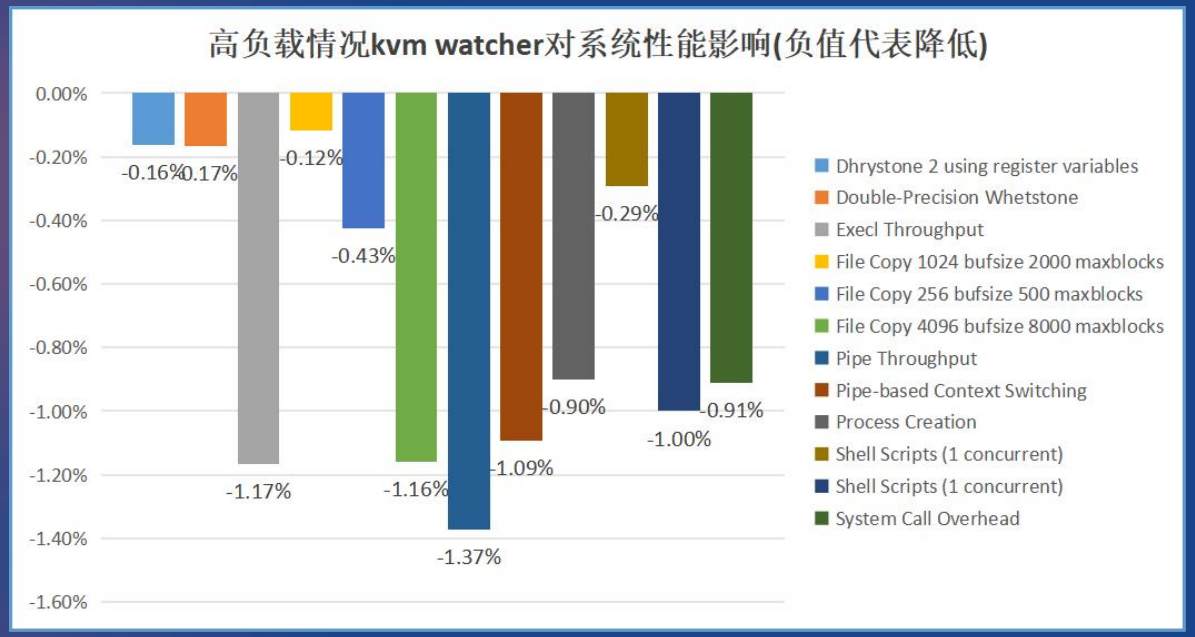
3.1 性能测试

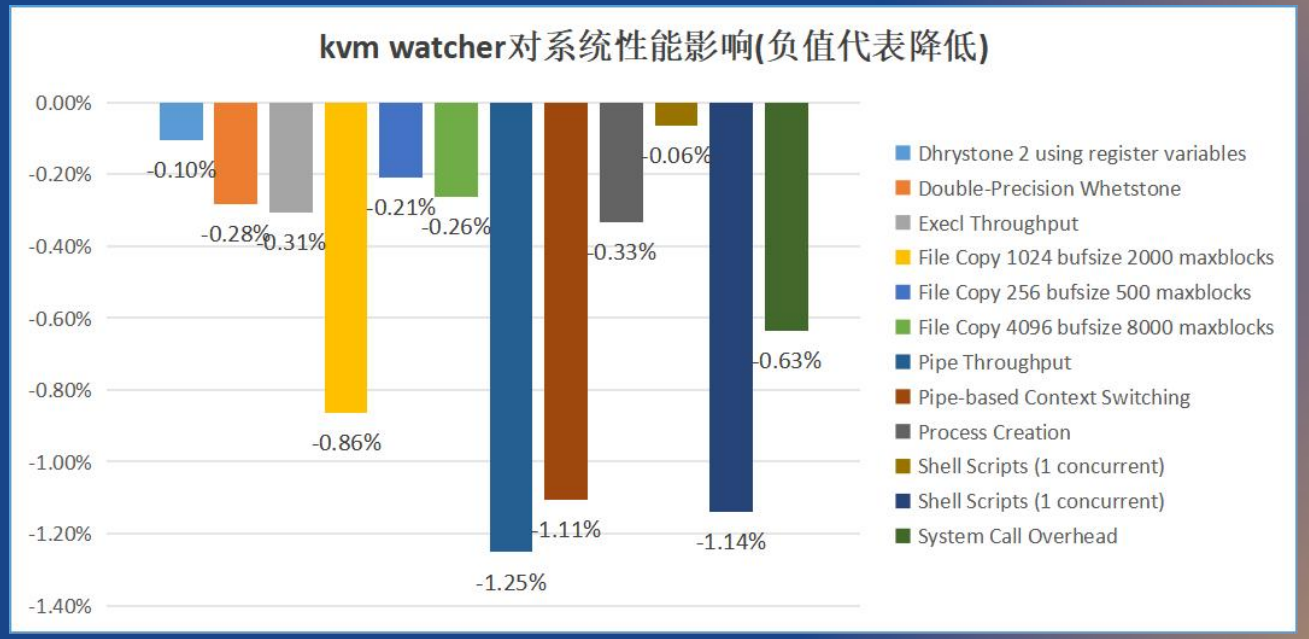
Item	Description
CPU	Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz
MiB Mem	95%
MiB Swap	98%
Load Ave	43.46, 39.38, 24.66（19 cores）
benchmark tool	unixbench
high-load simulation tool	stress-ng

UnixBench是一个通用的基准测试工具，旨在评估Unix和类Unix系统的性能。它包含了一系列的测试项目，涵盖了CPU、内存、磁盘、文件系统等方面的性能测试。UnixBench的测试结果可以帮助用户了解系统的整体性能表现，评估硬件升级或系统调优的效果，以及与其他系统的性能比较。

为了测试工具在高负载情况下对系统性能的影响，使用stress-ng模拟服务器高负载环境，使用基准测试工具unixbench在高负载环境下对系统进行测试。

3.2 性能测试





结果可以看出对系统性能的影响在1%左右，由此可以说明高负载情况下扩充的功能对系统性能几乎没有影响。

4 未来展望

4.1 未来展望

- 扩展功能，覆盖更多 KVM 性能分析领域
- 增强易用性和可视化
- 探索 eBPF 在 KVM 性能分析中的更多可能性
- 集成机器学习和人工智能算法

Thanks