kvm_stat and Beyond
Past, Present and Future of Performance Monitoring with KVM
–

Christian Borntträger
Maintainer KVM on IBM Z
borntraeger@de.ibm.com

IBM

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.
If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries.

The following are trademarks or registered trademarks of other companies.

- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Motivation

If something in KVM does not work, or is too slow it often lands on my desk
Very unspecific
Reporter has no knowledge if the problem is in the hypervisor or in the guest

Problems like
“Is slower than z/VM”
“high cpu usage”
“not fast enough”

No quick logon possible
Hard to re-recreate setups
Private data on the system
Performance data needs to be collected at the system and transferred somewhere else

Getting Data Local and Remote

Sysstat
Collects cpu (e.g. user, system, nice, interrupt),disk,memory data
Available on almost all distributions
Provides history data
Pretty easy to setup
Data can be transferred and analyzed somewhere else

Other tools
Nagios, ELK, prometheus/grafana.....

Sometimes the standard Linux data gives an insight, sometime it does not

So what else do we have?

Guest Time Accounting

Guest Time Accounting / Steal time accounting

KVM added the notion of guest time

Vs user, idle, hi, si, steal, and system time

```
host # sar -u ALL  1 100
[...]
09:58:28 AM     CPU      %usr     %nice      %sys   %iowait    %steal      %irq     %soft    %guest    %gnice     %idle
09:58:29 AM     all      0.00      0.00      0.00      0.00      0.00      0.00      0.00      7.14      0.00     92.86
09:58:30 AM     all      0.00      0.00      0.07      0.00      0.00      0.00      0.00      7.14      0.00     92.79
09:58:31 AM     all      0.00      0.00      0.00      0.00      0.00      0.00      0.00      7.14      0.00     92.8
[...]
```

Guests have the notion of steal time

I was runnable but the hypervisor scheduled me away


```
guest # sar -u 1 100
[...]
10:19:54        CPU     %user     %nice   %system   %iowait    %steal     %idle
10:19:55        all    100.00      0.00      0.00      0.00      0.00      0.00
10:19:56        all    100.00      0.00      0.00      0.00      0.00      0.00
10:19:57        all    100.00      0.00      0.00      0.00      0.00      0.00
10:19:58        all     99.01      0.00      0.00      0.00      0.99      0.00
10:19:59        all    100.00      0.00      0.00      0.00      0.00      0.00
[...]
```

Sysstat understands both

Top/procps

Top does not understand guest time
Neither in summary nor in per process

Same for vmstat

In 2007 Laurent Vivier provided a patch for top to understand guest time
Procps was badly maintained at that point in time
Maintainer was reluctant and did not followup in discussions
Is it time to retry today with procps-ng?

```
%Cpu(s):  4,0 us,  1,9 sy,  0,0 ni, 94,0 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
KiB Mem : 32451168 total,  4341112 free,  4257496 used, 23852560 buff/cache
KiB Swap: 32448508 total, 32318732 free,   129776 used. 27298876 avail Mem
[...]
```

No guest time

```
$ vmstat -w 1
procs -----------------------memory---------------------- ---swap-- -----io---- -system-- --------cpu--------
 r  b         swpd         free         buff        cache   si   so    bi    bo   in   cs  us  sy  id  wa  st
56  0            0      5374312       198724      1611768    0    0     0     0 58127  363  61  17  22   0   0
56  0            0      5374060       198724      1611768    0    0     0     0 58166  500  60  17  22   0   0
56  0            0      5373556       198724      1611768    0    0     0     0 58224  428  61  17  22   0   0
56  0            0      5373304       198732      1611768    0    0     0  7864 64461 8044  62  16  22   0   0
56  0            0      5372548       198732      1611768    0    0     0     0 58869 1351  60  18  22   0   0
56  0            0      5371524       198732      1611768    0    0     0    16 58082  441  60  17  22   0   0
56  0            0      5371020       198732      1611768    0    0     0     0 58120  393  60  18  22   0   0
56  0            0      5370516       198732      1611768    0    0     0     0 58055  448  60  17  22   0   0
56  1            0      5370516       198740      1611760    0    0     0  1876 59387 2016  61  17  22   0   0
56  0            0      5370264       198740      1611768    0    0     0  4584 61922 5148  61  17  22   0   0
```

KVM Counters (kvm_stat)

Kvm_stat Counters

2 kinds of counters: per cpu and per guest

Very cheap

Debugfs shows accumulated values across all VMs+all CPUs

No measurable overhead with kvm-unit-tests

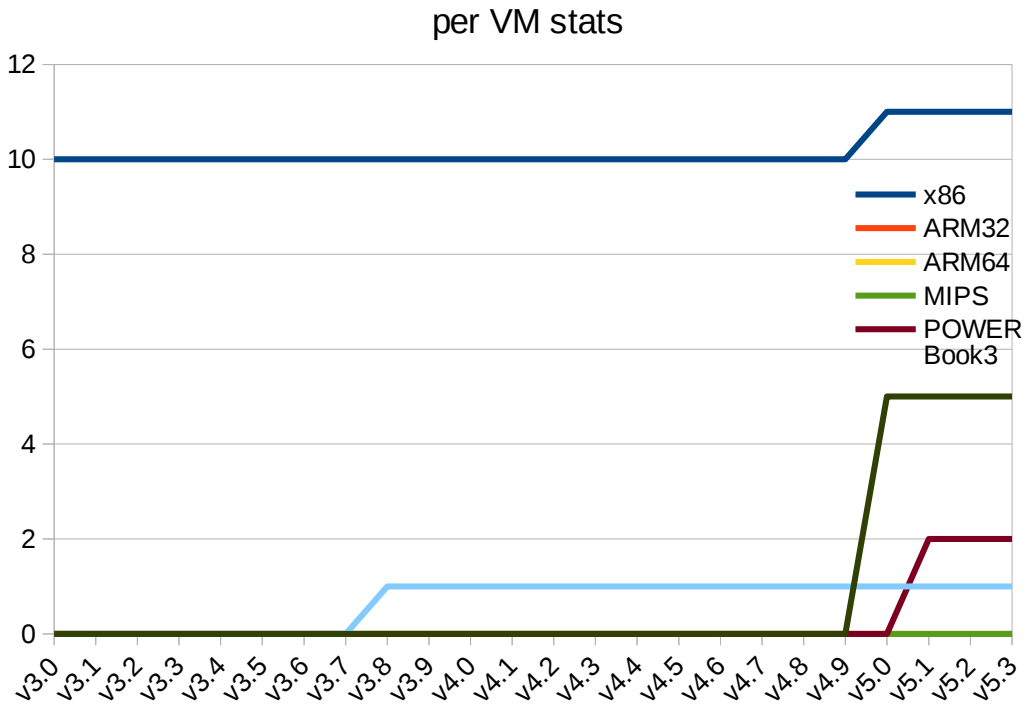
```
[root@kvm]# pwd
/sys/kernel/debug/kvm
[root@kvm# ls -R
.:
7848-11               halt_wakeup          irq_exits       mmu_flooded        nmi_injections    signal_exits
efer_reload           host_state_reload    irq_injections  mmu_pde_zapped     nmi_window        tlb_flush
exits                 hypercalls           irq_window      mmu_pte_updated    pf_fixed
fpu_reload            insn_emulation       l1d_flush       mmu_pte_write      pf_guest
halt_attempted_poll   insn_emulation_fail  largepages      mmu_recycled       remote_tlb_flush
halt_exits            invlpg               mmio_exits      mmu_shadow_zapped  req_event
halt_successful_poll  io_exits             mmu_cache_miss  mmu_unsync         request_irq

./7848-11:
efer_reload           host_state_reload    irq_injections  mmu_pde_zapped     nmi_window        tlb_flush
exits                 hypercalls           irq_window      mmu_pte_updated    pf_fixed          vcpu0
fpu_reload            insn_emulation       l1d_flush       mmu_pte_write      pf_guest
halt_attempted_poll   insn_emulation_fail  largepages      mmu_recycled       remote_tlb_flush
halt_exits            invlpg               mmio_exits      mmu_shadow_zapped  req_event
halt_successful_poll  io_exits             mmu_cache_miss  mmu_unsync         request_irq
halt_wakeup           irq_exits            mmu_flooded     nmi_injections     signal_exits

./7848-11/vcpu0:
tsc-offset
```

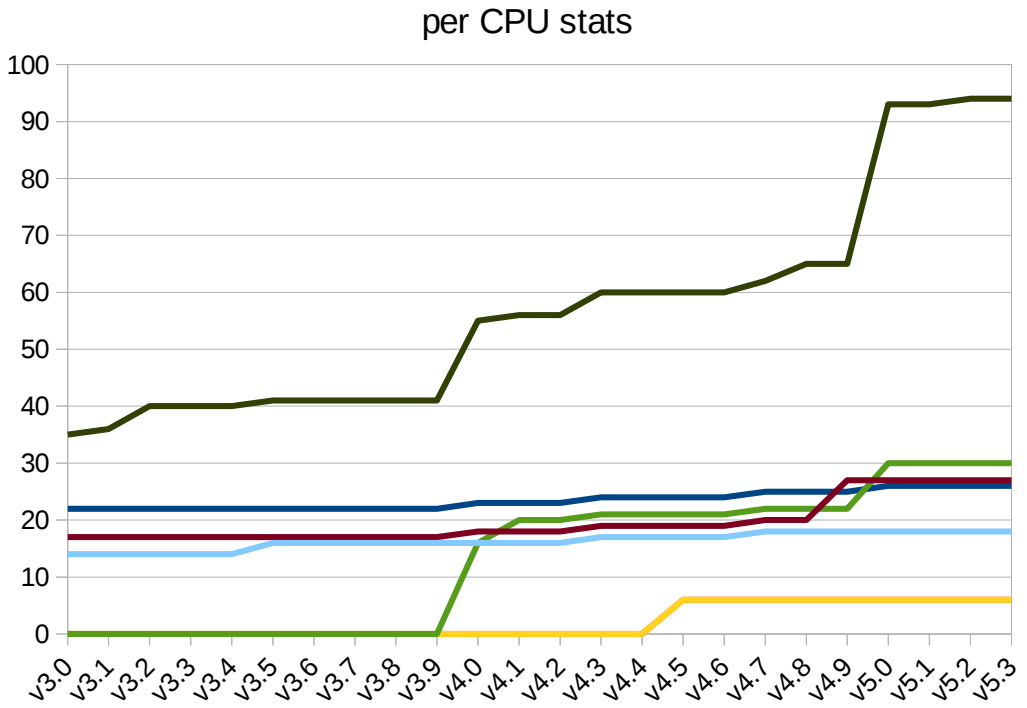
What Counters

Common code use
Only 4 common code counters
halt_attempted_poll
halt_successful_poll
halt_poll_invalid
halt_wakeup

Architectures differ a lot
Almost no use by ARM
S390 has highest use and growth

No presentation without a patch
ARM defined halt_* counters but did not expose those in debugfs → will increase the number of ARM couters to 10





kvm_stat Tool

kvm_stat Tool

Tool first shipped with qemu-kvm
Later moved into kernel

Initial tool kvm_stat
56 lines of code

Lots of additional features over time

```
kvm statistics

 efer_reload            0
 exits            2435313   62176
 fpu_reload        563348   24934
 halt_attemp        24490       6
 halt_exits         31591      10
 halt_succes        15606
 halt_wakeup        16732      10
 host_state_       568124   24721
 hypercalls             0
 insn_emulat       298971   11699
 insn_emulat            0
 invlpg                 0
 io_exits          548885   24465
 irq_exits         174556   13328
 irq_injecti       197133   10160
 irq_window         12064     626
 l1d_flush        2059091   50214
 largepages            23
 mmio_exits          6441       7
 mmu_cache_m         3351      49
 mmu_flooded            0
 mmu_pde_zap            0
 mmu_pte_upd            0
 mmu_pte_wri            0
 mmu_recycle            0
 mmu_shadow_         2314
 mmu_unsync             0
 nmi_injecti            0
 nmi_window             0
 pf_fixed         1333525
[….]
```

Per-guest Counters

Initially kvm stats were global
Accumulated over all guests over all vcpus
Data was available in code

Just needed to be exposed via debugfs

kvm_stat tool
--guest or g key allows to filter

```
Show statistics for specific guest or pid.
This might limit the shown data to the trace statistics.

Guest or pid [ENTER exits]:



       Pid  Guest Name (fuzzy list, might be inaccurate!)
     72597  test0
     72621  test4
     72623  test2
     72629  test3
     72663  test7
     72672  test1
     72676  test9
     72696  test6
     72698  test8
     72700  test5
```

Logging

kvm_stat -l
Every second

```
kvm_ack_i kvm_age_p  kvm_apic kvm_apic_ kvm_apic_ kvm_async kvm_async kvm_async kvm_async kvm_avic_ kvm_avic_ kvm_cpuid     kvm_cr kvm_emula kvm_enter kvm_entry   kvm_eoi  kvm_exit kvm_exit( kvm_exit( kvm_exit( kvm_exit( kvm_exit( kvm_exit( kvm_exit( kvm_exit( kvm_exit( kvm_fast_   kvm_fpu kvm_halt_
kvm_exit( kvm_exit( kvm_exit( kvm_exit( kvm_exit( kvm_exit( kvm_exit( kvm_exit( kvm_exit( kvm_exit( kvm_exit(  kvm_exit( kvm_exit( kvm_exit( kvm_exit(  kvm_exit( kvm_exit( kvm_exit( kvm_exit( kvm_exit( kvm_exit( kvm_exit( kvm_exit( kvm_exit( kvm_exit(
```

Trace Events

Trace Events

Used to be the new cool kid in the house
Trace events everywhere

Plus sides
Trace events allow to bind values to guest /cpus
Almost no overhead when off
Allows latency measurements

Trace events for kvm have all the nice filters as normal trace events
Very fine grained tracing if needed
Great for live debugging

kvm_stat defaults to trace events
I suggest to always use -d -t to get trace events and counters

Perf kvm stat live

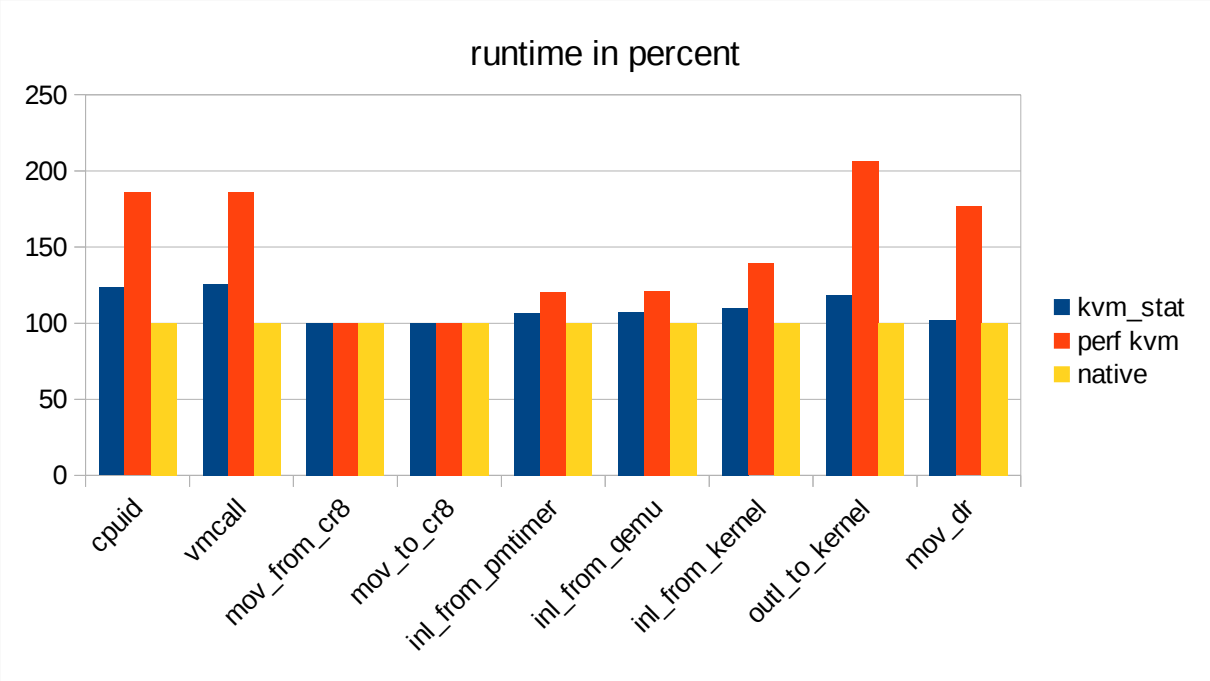
Using trace events to analyze latencies
From exit to re-entry

```
Analyze events for all VMs, all VCPUs:

             VM-EXIT    Samples  Samples%     Time%    Min Time    Max Time         Avg time

  EXTERNAL_INTERRUPT       2845    41.89%    27.40%      0.56us  22085.57us     67.79us ( +-  21.82% )
           MSR_WRITE       1246    18.35%    16.14%      0.71us  14989.71us     91.17us ( +-  28.48% )
      IO_INSTRUCTION       1186    17.46%    37.96%      2.93us  19441.21us    225.30us ( +-  15.43% )
        EPT_VIOLATION        616     9.07%    10.81%      1.05us  21252.50us    123.58us ( +-  37.88% )
 TPR_BELOW_THRESHOLD        265     3.90%     4.08%      0.72us  10350.12us    108.47us ( +-  49.82% )
   PAUSE_INSTRUCTION        203     2.99%     0.09%      0.43us    158.07us      3.08us ( +-  28.50% )
               CPUID        182     2.68%     0.77%      0.46us   4290.89us     29.71us ( +-  81.08% )
   PENDING_INTERRUPT        122     1.80%     1.04%      0.71us   7084.68us     59.78us ( +-  97.11% )
              VMCALL         93     1.37%     0.43%      0.66us   2796.13us     32.85us ( +-  91.42% )
       EPT_MISCONFIG         13     0.19%     1.03%      6.22us   5349.92us    555.56us ( +-  75.00% )
    PREEMPTION_TIMER         10     0.15%     0.00%      0.91us      1.36us      1.09us ( +-   3.83% )
                 HLT          8     0.12%     0.24%      3.45us   1143.19us    214.47us ( +-  62.21% )
          DR_ACCESS          3     0.04%     0.00%      2.16us      2.65us      2.44us ( +-   5.98% )

Total Samples:6792, Total events handled time:703886.16us.
```

Trace Events Everywhere?

Change everything to trace events?
Not quite

Downsides
Measurable overhead for microbenchmarks
Must be activated to “measure” → no history
Kind of ABI (or not?)



libvirt

Libvirt Interfaces

Several interfaces to query domain...

```
    cpu-stats                   show domain cpu statistics
    domblkstat                  get device block stats for a domain
    domifstat                   get network interface stats for a domain
    dommemstat                  get memory statistics for a domain
    domstats                    get statistics about one or multiple domains
```

...Or node information

```
    nodecpustats                Prints cpu stats of the node.
    nodememstats                Prints memory stats of the node.
```

Libvirt Interfaces - cpu-stats

Provides per-cpu times
vcpu understands guest time
Queried from cgroup cpuacct

```
virsh cpu-stats myguest
CPU0:
      cpu_time             38.250348806 seconds
      vcpu_time            36.329019159 seconds
CPU1:
      cpu_time             34.048214224 seconds
      vcpu_time            32.213941613 seconds
CPU2:
      cpu_time             39.577205065 seconds
      vcpu_time            37.958059656 seconds
CPU3:
      cpu_time             36.528732041 seconds
      vcpu_time            34.997068067 seconds
CPU4:
      cpu_time             27.706520847 seconds
      vcpu_time            27.414863236 seconds
CPU5:
      cpu_time             22.345352968 seconds
      vcpu_time            21.986850492 seconds
CPU6:
      cpu_time             28.786509907 seconds
      vcpu_time            27.919375487 seconds
CPU7:
      cpu_time             25.532488444 seconds
      vcpu_time            25.103798154 seconds
Total:
      cpu_time            252.775372302 seconds
      user_time             0.890000000 seconds
      system_time          90.320000000 seconds
```

Libvirt Interfaces - nodecpustats

Not all interfaces understand guest time
The internal interface virNodeGetCPUStats and virs nodecpustats do not know about guest time

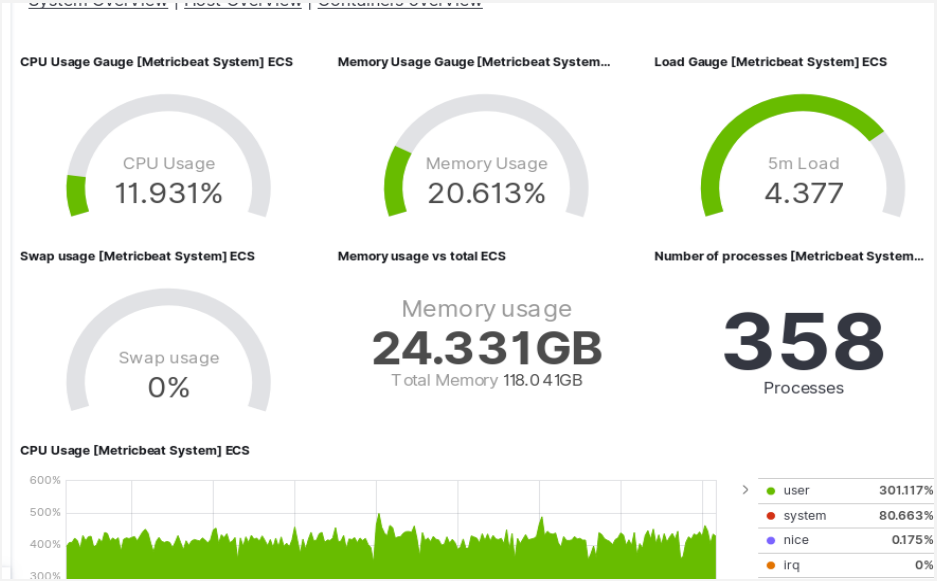
```
$ virsh nodecpustats
user:                32885220000000
system:              26527760000000
idle:              3035468670000000
iowait:                869040000000
```

Large scale tools

ELK Stack

Elastic search logstash kibana
History data
Can be analyzed on a different system
Metricbeat understands system, user, nice, irq steal and other times
No understanding of guest and guest nice time
KVM beat understands dommemstat from libvirt

No understanding of kvm counter and trace points
No understanding of other libvirt interfaces



Prometheus + Grafana

Prometheus/graphana
History data
Can be analyzed on a different system

Several dashboards for libvirt
Via prometheus or via collectd from libvirt
System-wide data (cpu, memory, swap)
Again, no kvm stat
Some dashboards understand guest time (hooray)

Moving Forward and Summary

Use the interfaces

Extend the interfaces if you miss something
Most tools only support a tiny subset of the data
We need to add things where appropriate

Talk about whats there
KVM provides more data than most people know
We need to get this message out

Next steps

More common kvm_stat counters	todo
Revalidate if we have the right counters	todo
Enable guest time in vmstat	**submitted**
Enable guest time in top	**patch written**
Find a new place for kvm stat(no debugfs)	todo
Integrate kvm_stat in ELK stack	todo
Integrate kvm_stat in Prometheus/Grafana	todo
Integrate kvm_stat in sysstat	todo

Thanks

- **Christian Borntraeger borntraeger@de.ibm.com**

BACKUP

32 bit counters

Initially all counters 32 bit
Power introduced a new counter that added time: halt_poll_success_ns et al
64 bit needed

64bit counters on 32 bit CPUs
Per cpu-counters are easy
Per VM counters would require locking
Idea: 64bit per cpu, unsigned long per vm.

Power-specific?
Is the idea of halt_poll_success_ns good for everyone?

debugfs

Debugfs considered harmful by Greg
Several ideas, no one is working on this?

```
Subject     Re: [PATCH] kvm: no need to check return value of debugfs_create
functions
From  Paolo Bonzini <>
Date  Wed, 23 Jan 2019 00:11:18 +0100


On 22/01/19 21:48, Greg Kroah-Hartman wrote:
>> This also brings the question: shall we move these counters out of debugfs
into something else?
> If you have code that relies on debugfs, yes, you need to move that out
> of debugfs because more and more systems are trying to disable it due to
> the obvious problems with it (i.e. leaking tons of debugging
> information).
>
> debugfs is for DEBUG information, not for "statistics about how my VM is
> working".  That sounds like something you need to rely on, so debugfs is
> not the place for it.

Yes, we know that and tracepoints are already one replacement.  However,
they are slower that just a lock-free "vcpu->stats.foo_happened++".
Another idea that Steven Rostedt and I discussed a while ago is some
kind of "statfs" which would already provide some code, similar to the
one that KVM uses to accumulate statistics from multiple VMs or multiple
VCPUs into a single counter.

Paolo
```

Potential micro optimization

Almost no measurable overhead with unit test
For testing I removed all vpcu→stat.<counter>++ lines

There is one thing that can be optimized
Group frequent counter with other frequent counters to properly use the cache lines containing the counter
Usually 8 counters per cacheline, for a random workload I marked the most common ones

```
struct kvm_vcpu_stat {
        u64 pf_fixed;
        u64 pf_guest;
        u64 tlb_flush;
        u64 invlpg;

        u64 exits;
        u64 io_exits;
        u64 mmio_exits;
        u64 signal_exits;
        u64 irq_window_exits;
        u64 nmi_window_exits;
        u64 l1d_flush;
        u64 halt_exits;
        u64 halt_successful_poll;
        u64 halt_attempted_poll;
        u64 halt_poll_invalid;
        u64 halt_wakeup;
        u64 request_irq_exits;
        u64 irq_exits;
        u64 host_state_reload;
        u64 fpu_reload;
        u64 insn_emulation;
        u64 insn_emulation_fail;
        u64 hypercalls;
        u64 irq_injections;
        u64 nmi_injections;
        u64 req_event;
};
```

Grouping/Drill Down

Overview with summary counters
“x” will expand the details

Skvm statistics - summary

Event	Total	%Total	CurAvg/s
kvm_exit	3943	17.1	236
kvm_exit(MSR_WRITE)	**2647**	**67.1**	**162**
kvm_exit(HLT)	**936**	**23.7**	**56**
kvm_exit(EXTERNAL_INTERRUPT)	**209**	**5.3**	**12**
kvm_exit(PENDING_INTERRUPT)	**23**	**0.6**	**2**
kvm_exit(IO_INSTRUCTION)	**7**	**0.2**	**0**
kvm_entry	3943	17.1	236
kvm_hv_timer_state	3587	15.5	217
kvm_msr	2647	11.5	162
kvm_fpu	1872	8.1	112
kvm_inj_virq	1061	4.6	61
kvm_eoi	1061	4.6	61
kvm_apic_accept_irq	1061	4.6	61
kvm_pv_eoi	1048	4.5	61
kvm_ple_window	937	4.1	56
kvm_vcpu_wakeup	936	4.1	56
kvm_wait_lapic_expire	913	4.0	56
kvm_halt_poll_ns	39	0.2	0
kvm_msi_set_irq	21	0.1	0
kvm_apic	13	0.1	0
kvm_pio	7	0.0	0
Total	23089		1373