Monitoring with RHEL 8 and KVM

Getting Started, Hints and Tips

—

Solution Assurance

Bodo Brand, Daniel Kaiser, David Stark

Document version: 03-2022

IBM Z

IBM

Notices and disclaimers

U.S. Government Users Restricted Rights – use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. **This document is distributed "as is" without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.** IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.

IBM products are manufactured from new parts or new and used parts. In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply.

Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to ensure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

Notices and disclaimers, *continued*

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.**

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, ibm.com and IBM Z, IBM z15, and z/VM are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml

Monitoring - **Agenda**

- **Terminology**
- **Introduction**
- **Prerequisites** and **Scope**

1.x - Tool Comparison

2.x - kvm_stat – Getting Started & Logging
2.x - kvm_stat – Live Stats / Interactive Use

`Hypervisor`

3.x - Sysstat – Getting Started
3.x - Sysstat – Logging
3.x - Sysstat – Change Logging Interval

`Hypervisor` `Guest`

4.x - PCP – Compared with sysstat

`Hypervisor` `Guest`

5.x - PERF – perf kvm

`Hypervisor` `Guest`

Terminology

Logging
Log - usually events - to a log file with a certain level of detail. Some logs are meant to be read by humans, others require post-processing before the result is readable. There are also hybrids.

Monitoring
Observe the state - usually a set of metrics - of a system at a particular point in time (optionally over a period of time).

This Metrics can be post processed in various ways. You might want to “deflate”, “keep certain amount/time of history”, “Set a reasonable Interval”, “Store the Logs on a separate Server”, ...

Monitoring viewer/analyser – Produce reduced/filtered statistics and graphs that are useful to resolve a given problem.

Debugger – Find and Fix Bugs in Applications by stepping through a running application. Pause and Investigate.

Profiler
Identify *where* performance is lost. Results in a profile which is a statistical summary of observed events. “Which function took the most time to execute?”

Profiling
Collect samples at specific period/frequency. Samples contain information about CPU State, Instruction Address, etc.

Methods (Introduction, Details):
- Overflow Interrupt-based Sampling
- Processor Event-based Sampling

Tracer
Write history of execution, with **only** the recorded values of variables and context fields **you choose**. “Execution Flow and Data Progression”.

Trace viewers/analysers – Produce reduced/filtered statistics and graphs that are useful to resolve a given problem

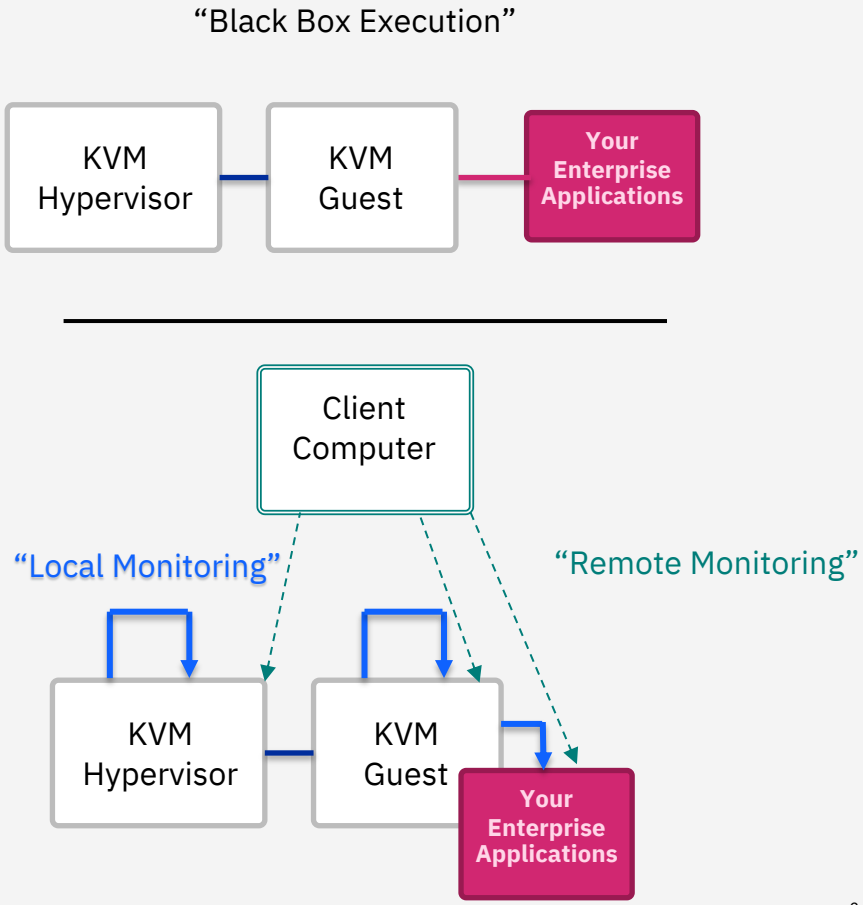
Introduction

Monitoring:

Target / Granularity:
- Application / System / Hardware / (mix)

Purpose:
- health / availability / performance / (mix)
- Usually **only determine** that a problem exists
- **advanced** Troubleshooting together with Tracing/Debugging/Logging

Decisive Factors:
- Intended Viewer / Target / Purpose
-> Metric Selection, Interval, History



Prerequisites and **Scope**

Setup:

z15™

KVM Hypervisor

2 KVM Guests

KVM Guests:

Distro: RHEL **8.4**

System Monitoring:

sysstat

pcp

KVM Monitoring:

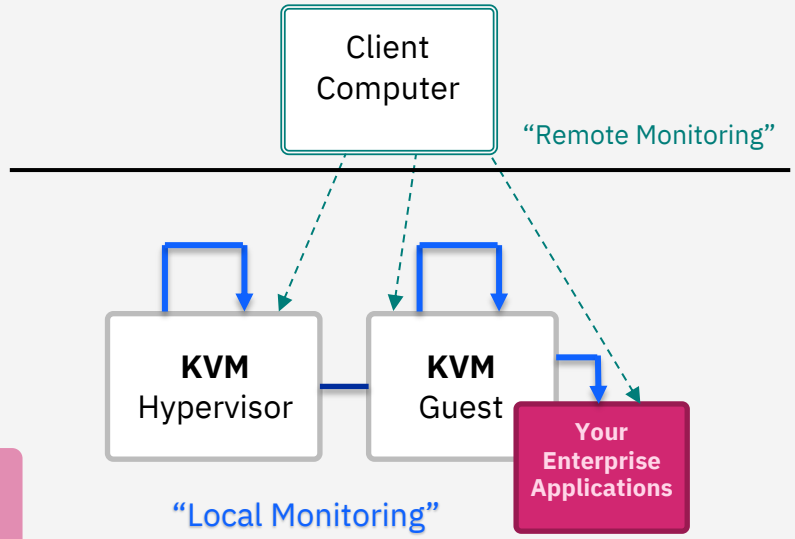
kvm_stat

perf [kvm] [stat/record]

-> cpumf

Focus Areas:

- Getting Started
- Configuration Examples
- Hints

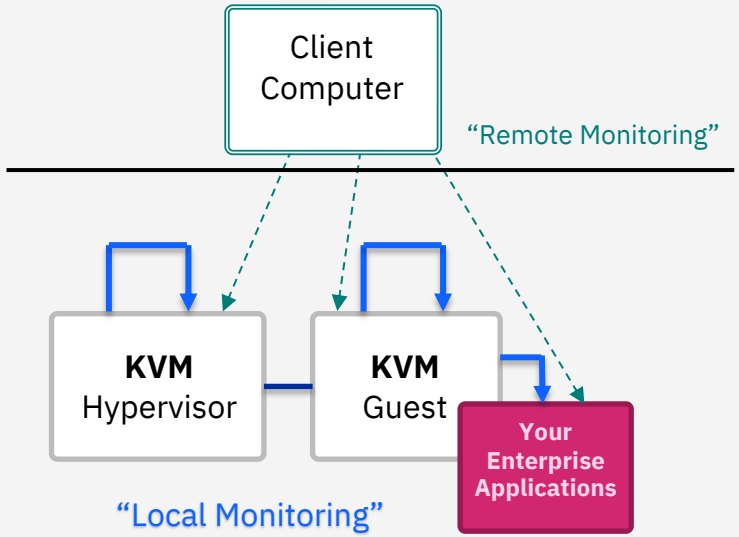


1.0 - Linux - Tool Comparison

Linux Monitoring Tools Comparison:

	has KVM capabilities	Systemd service	Remote Monitoring	Tags
kvm_stat	YES	YES (csv)	NO	Advanced
sysstat	NO	YES	NO	Simple; low system overhead
pcp	YES (collects some KVM metrics)	YES	YES	Advanced; Client/Server; Modular
perf	YES (perf kvm)	NO	NO	Expert

Also check out: pcp compared with sysstat



2.0 - kvm_stat - Getting Started & Logging

Getting Started

- kvm_stat is a **python script** which prints counts of **KVM kernel module trace events**

- Install kvm_stat:

```
# dnf install kernel-tools
```

Enable Logging:

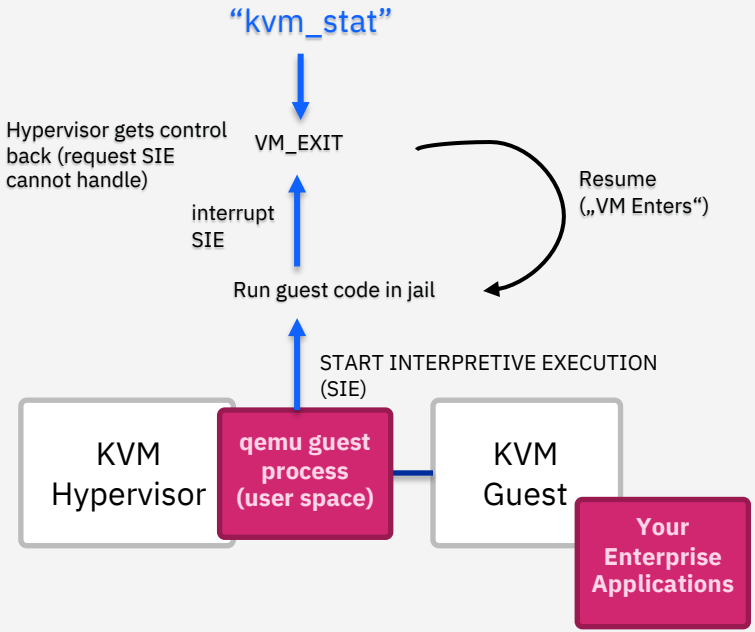
- Run (Link to service file):

```
# systemctl enable kvm_stat --now
```

Note:
- writes to /var/log/kvm_stat.csv
- refreshes every 10 seconds by default
  -> can be changed with option: `-s` (via systemctl edit kvm_stat)

Lookup VM_Exit Meaning:

- Lookup Diag Codes (PDF page. 723)



2.1 - kvm_stat – Live Stats / Interactive Use

- Live Stats / Interactive Use:

- kvm_stat using debugfs (/sys/kernel/debug/kvm)

```
# kvm_stat -d
# kvm_stat -di  # (use all past data as well)
```

- kvm_stat using tracepoints

```
# kvm_stat -t
```

- kvm_stat using both (full output)

```
# kvm_stat -dt
```

- kvm_stat full output in logging mode

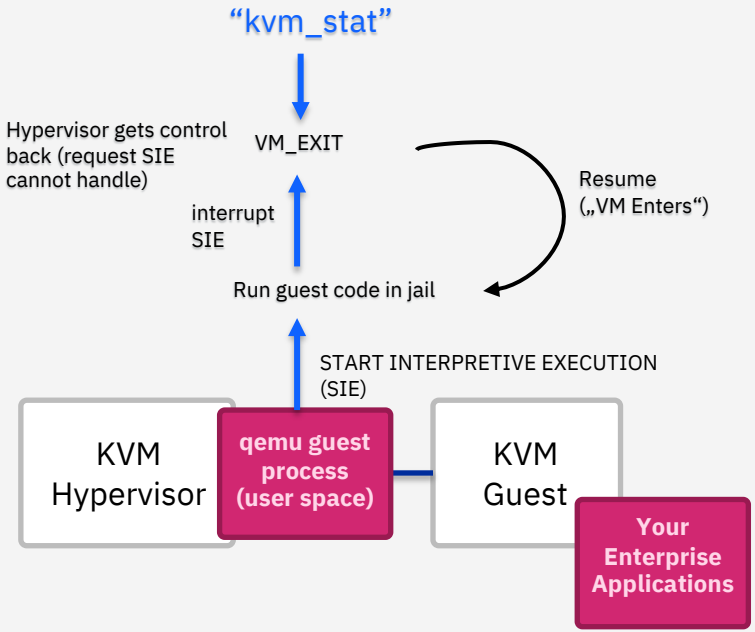
```
# kvm_stat -dtl
```

x – more fine-grained output
f – regex filter
s – set update interval

Notes:
- debugfs is more complete then tracepoints on s390 (Source)
- use "-l" to run in logging mode

Use Case (Example):

- Blogpost about "KVM Guest Analysis on LoZ"
- Micro Optimizing KVM VM_Exits



3.0 - Sysstat - Getting Started

Getting Started

- Sysstat is a **low system overhead** option for **System Monitoring**
  - Simple **C Program** Interface to /proc, /sys, /dev
  - Redhat documentation
  - Official Website

- Install sysstat:

```
# dnf install sysstat
```

- Current Stats / Live Stats

- **Utilities**: iostat, mpstat, pidstat, tapestat, cifsiostat
  -> the first report is a statistic since the system was booted
  -> successive reports are statistics from the specified time interval

Notes:
- Designed for humans to view the output
- Not possible to set intervals below 1 second
-> For such use case use Tracing methods e.g. “perf”
- **sar** can also be used to live-view the logged metrics

Examples:

iostat (extended, human readable, 2 second interval):

```
# iostat -xht -d 2
```

mpstat (all CPUs, all informations 2 second interval):

```
# mpstat -A 2
```

pidstat (all informations, all processes 2 second interval):

```
# pidstat -dRrlstuvw -p ALL -T ALL 2
```

3.1 - Sysstat - Logging

Enable Logging:

```
# systemctl start sysstat
```

Note:
- Under RHEL8.4 the **sysstat service** and systemd-timers sysstat-collect.timer and sysstat-summary.timer are already enabled after installation causing the services to start on **reboot**
- The sysstat service also starts both timer services

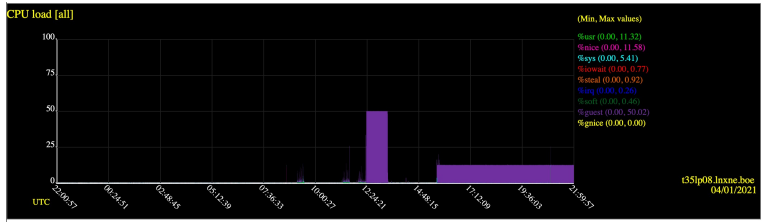
Default Logging Configuration

- sysstat.service
  - sysstat-collect.timer (**every 10 minutes current Snapshot, keeps 1 month history**)
  -> sysstat-collect.service: **`/usr/lib64/sa/sa1 1 1`**
    `/var/log/sa/[saDD|saYYYYMMDD]`
    -> binary file to generate reports from
  - sysstat-summary.timer (**every 24 hours summary from current saDD-file, keeps 1 month history**)
  -> sysstat-summary.service: **`/usr/lib64/sa/sa2 -A`**
    `/var/log/sa/[sarDD|sarYYYYMMDD]`
    -> parsable file to collect and post process further

Generate reports from saDD (Examples)

```
# sadf -g /var/log/sa/saDD -O showinfo -- -A > result.svg
```

`result.svg` (one of many generated graphics)



3.2 - Sysstat - Change Logging Interval

The default Logging interval is 10 minutes

-> You might benefit from a **lower value** to get a **more fine-grained view**

-> Our example below shows how to change the interval to 10 seconds. If you want to go below 10 seconds see the disclaimer on the right.

Change Interval to 10 seconds

```
# systemctl edit sysstat-collect.timer
```

```
[Unit]
Description=Run system activity accounting tool every 10 seconds

[Timer]
OnCalendar=
OnCalendar=*:*:0/10
AccuracySec=500ms
```

```
# systemctl daemon-reload
# systemctl status sysstat-collect.timer
```

Note:
- "OnCalendar=" resets previously set OnCalender options and is required. (OnCalender is an additive option) If you leave this option out the Event will be called twice every 10 minutes which may lead to problems.
- The default AccuracySec of systemd-timers is 1 Minute to optimize power consumption by suppressing unnecessary CPU wake-ups. "1us" is the maximum accuracy possible but the value should be as high as possible and as low as necessary. (source)

If you choose an interval **below 10 seconds** you have to change the StartLimitIntervalSec of the systemd sysstat-collect.service (to prevent errors):

```
# systemctl edit sysstat-collect.service
```

```
[Unit]
StartLimitIntervalSec=1
```

```
# systemd-analyze verify sysstat-collect.service
```

Note:
- The default is 10 seconds which leads to a "start-limit-hit" failure which indicates that the StartLimit was not reached multiple times. The DefaultStartLimitBurst is 5.

Interval Options:

```
OnCalendar=*:0/1          # every 1 minutes
OnCalendar=*:*:0/10       # every 10 seconds
OnCalendar=*:*:0/1        # every 1 second
```

Note:
- The event triggers when:
  - the number on the * position changes the event triggers.
  - divisible by the given number the event triggers.
- Change AccuracySec accordingly

3.3 - Sysstat - File Size Reference

For sysstat-collect interval considerations:

File Size Comparison for Reference (8 cpus, 2 disks, 1 network iface):

Disclaimer:
Be aware that this values can be significantly higher depending on how much Hardware you configure to your system.

Interval (AccuracySec=1us)	sa file size (per day)	sar file size (per day)
*:0/1	~4 mb	~ 6 mb
::0/10	~22 mb	~34 mb
***:*:0/1**		

4.0 - PCP - Getting Started

Getting Started

- PCP (Performance Co-Pilot) is a system monitoring suite which provides a lot of functionality by default
- Python Program – Modular –
  Remote Monitoring Capabilities from multiple hosts
- Redhat documentation
- Official PCP documentation

- Install preconfigured pcp:

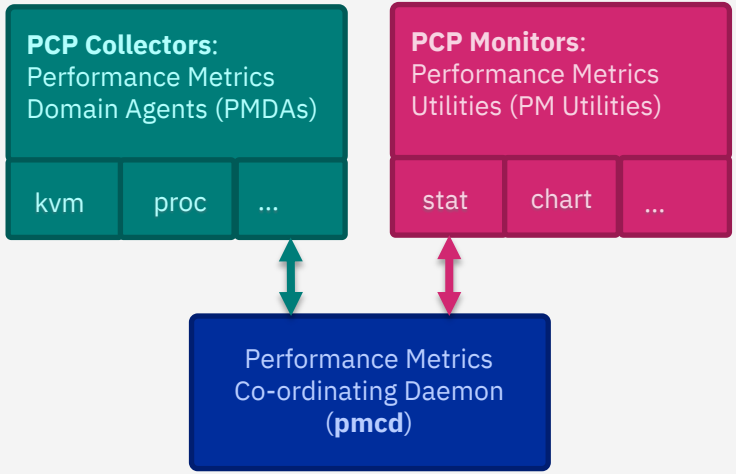
```
# dnf install pcp-zeroconf
```

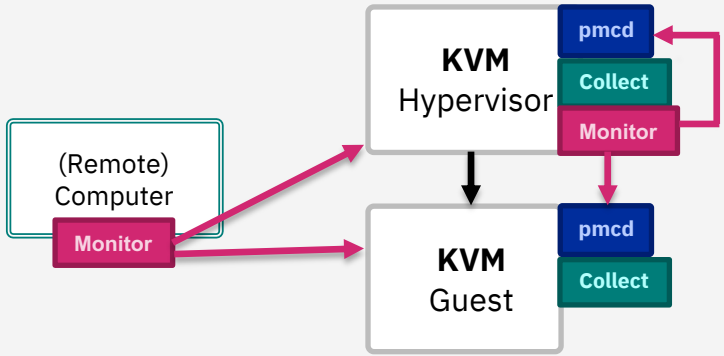
Note:
- Make sure to reserve at least 10 GB storage for Monitoring data collection

- Default Configuration

- PMDAs running:
  root, proc, xfs, linux, nfsclient, **kvm,** adm, openmetrics

- remote monitoring **disabled** (look here to **enable** it)
- pmlogger.service + multiple systemd timers
  -> handles logging, archiving and more
- pmie.service
  -> create monitoring reaction rules (e.g. show popup)





4.1 - PCP - Compared with sysstat

Comparison to sysstat

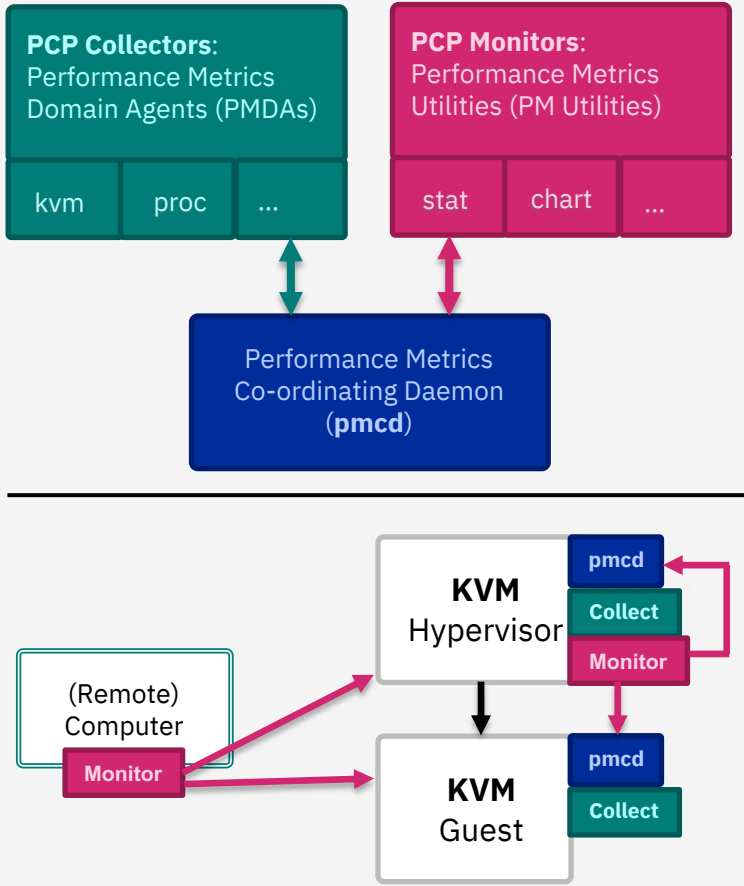
- **Advantage over sysstat**
  - remote monitoring capabilities
    (discovery and collection from multiple hosts included)
  - **Limited KVM metrics included by default**
  - **Extensible and Flexible**
    - **write your own PMDAs**
    - **a lot of post processing and import capabilities**
  - APIs allowing the use of Prometheus, Grafana and more

- **Disadvantage over sysstat**
  - **performance hungry** and **disk space hungry**
    (Each additional metric collector costs performance)
  - much more complex

- **Simple Performance Comparision**

```
perf stat -e cycles -- sar -u 2 5
```
-> 15,362,948 cycles

```
perf stat -e cycles -- pcp -t 2 -s 5 mpstat
```
-> 370,699,803 cycles



5.0 - PERF - Getting Started

PERF Introduction -

- Install:

```
# dnf install perf
```

- Example Use Cases:

- Capacity Planning, Validate Performance, Insights into new features and functions, Deep inspection

- Links to get started:

- Introduction to perf with a lot of examples
- perf kernel.org wiki
- RHEL8 introduction to perf

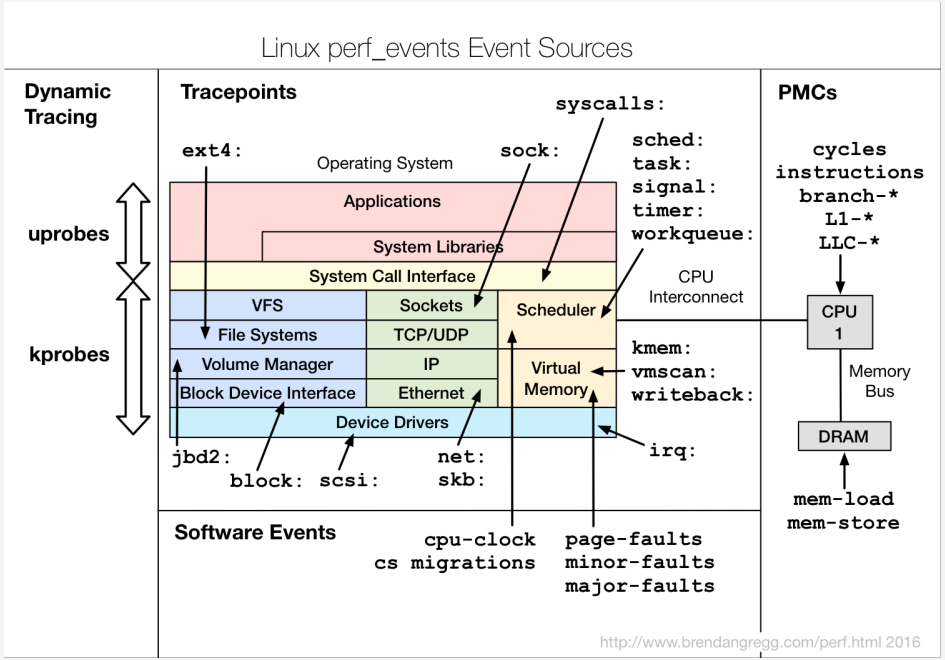


Image license: creative commons Attribution-ShareAlike 4.0.

5.1 - PERF - Introduction

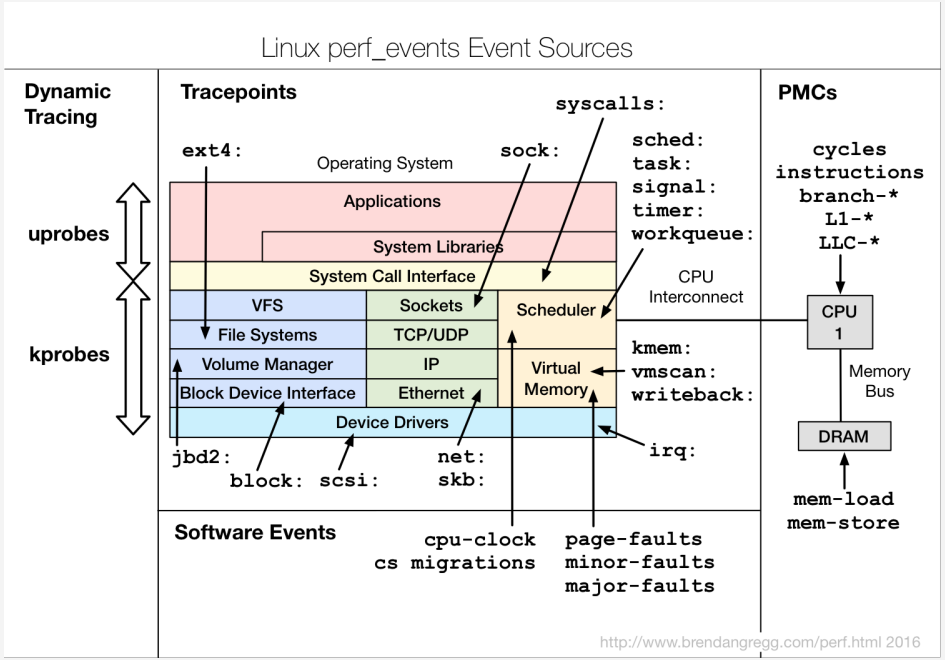
PERF - Available Metrics

- **Hardware Counter** (PMC)
  Software Counter (Kernel Counter)

- **Kernel Tracepoints Events**
  (Hooks to logical places in kernel code)
  -> generate statistics and/or create profiles
  -> Example: "`kvm:kvm_s390_sie_enter`"

- **Dynamic Tracepoints**
  (Hooks to any location in kernel/user code)

- **User-Space Statically Defined Tracing (USDT)**
  (Hooks to predefined Locations dynamically)
  (only supported Applications e.g. `qemu`)



5.2 - PERF kvm - Example

PERF KVM specific example:

Record host and guest informations in parallel:

Mount guest with sshfs

```
dnf install sshfs
PID=$(pgrep qemu-kvm|head -n 1)
mkdir -p /tmp/guestmount/$PID
sshfs -o allow_other,direct_io 192.168.122.39:/ /tmp/guestmount/$PID

perf kvm --host --guest --guestmount=/tmp/guestmount record -a -o perf.data
perf kvm report -i perf.data
```

options

```
Default('')          ->  perf.data.guest
--host               ->  perf.data.kvm
--guest              ->  perf.data.guest
--host --guest       ->  perf.data.kvm
--host --no-guest    ->  perf.data.host
```

Note:
- Use -o and -i respectively to force the perf data filename

More Examples