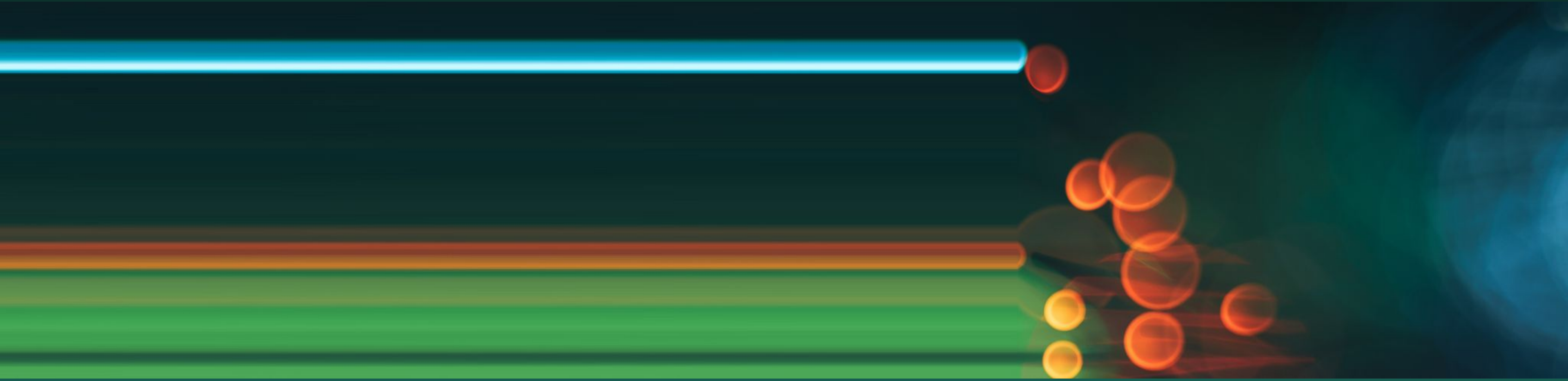
SUSE

SUSE Labs Conference 2021

QEMU & KVM Automated Performance Benchmarking @ SUSE

Dario Faggioli, SUSE, Italy



Self Introductions

Who I am, what I do...

dfaggioli@suse.com
@DarioFaggioli
@dfaggioli:matrix.org
dariof (IRC)

Dario Faggioli

Virtualization Software Engineer

Ph.D in real-time systems, then Citrix (2011), now SUSE (2018)

Worked on Linux scheduling (`SCHED_DEADLINE`), Xen hypervisor (not only) scheduling (`Credit2`), QEMU & KVM upstream and downstream

Tend to focus on "anything performance"

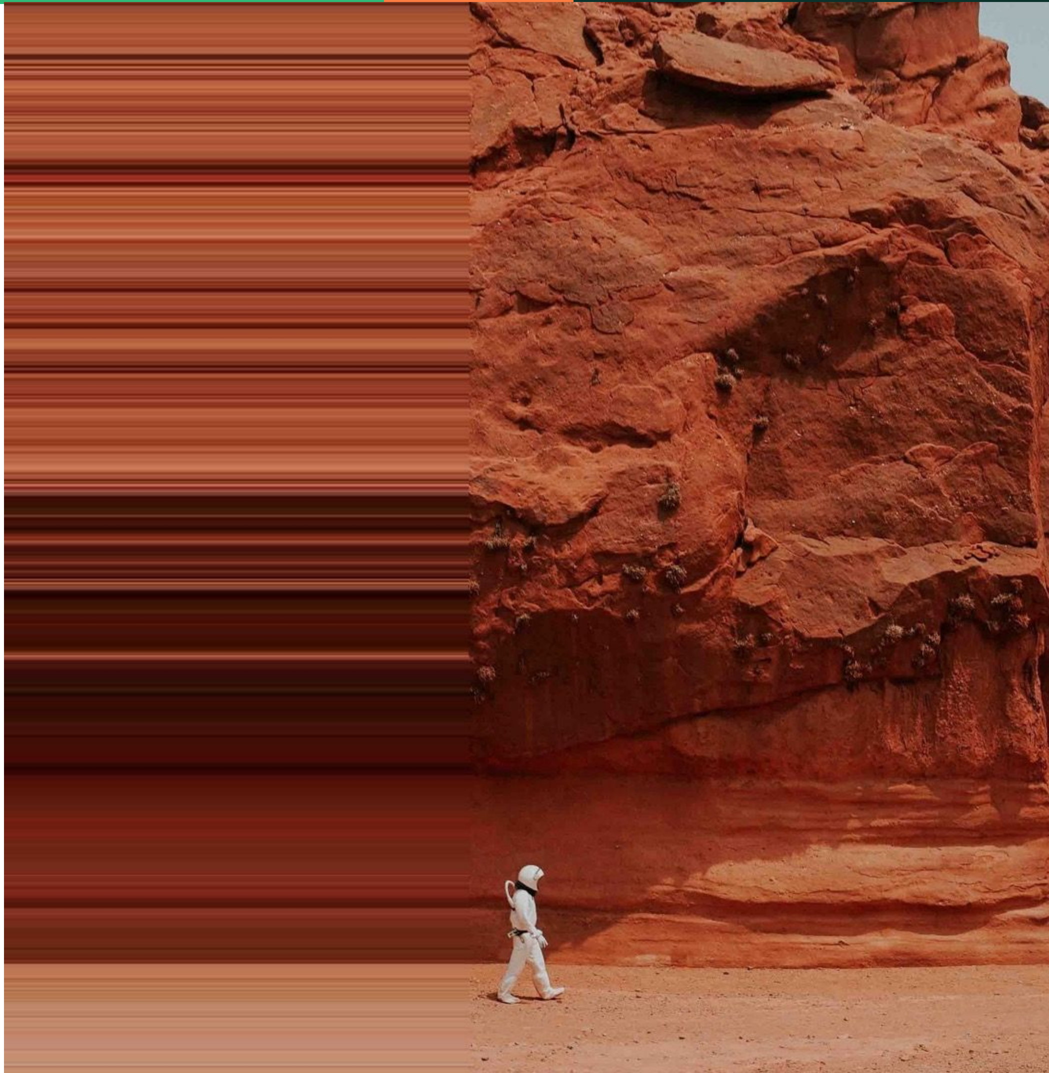
In openSUSE, since 2018 too

User:
- Tumbleweed
- MicroOS Desktop

Contributor:
- Virtualization packages
- Some tracing packages
- MicroOS Deskop

Agenda

- Why benchmarking Virtualization
- Tools for benchmarking Virtualization
- What to do for benchmarking Virtualization



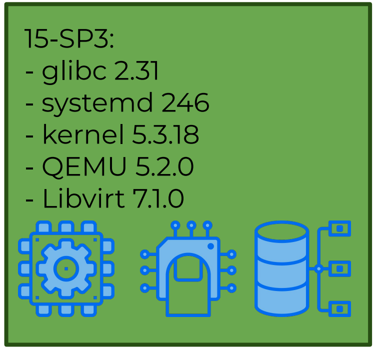
Icons by Smashicons and Payungkead @ www.flaticon.com

Performance Testing

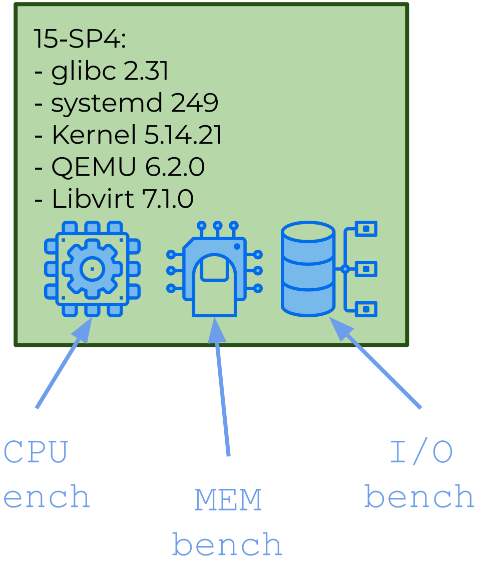
A (real) story: releasing a new version of the OS

Let’s avoid performance regressions, between **SLE[1] 15-SP3** and **SLE 15-SP4**:

- Run benchmarks CPU bench, I/O bench and MEM bench on **15-SP3**
- Run benchmarks CPU bench, I/O bench and MEM bench on **15-SP4**



15-SP3:
- glibc 2.31
- systemd 246
- kernel 5.3.18
- QEMU 5.2.0
- Libvirt 7.1.0



15-SP4:
- glibc 2.31
- systemd 249
- Kernel 5.14.21
- QEMU 6.2.0
- Libvirt 7.1.0

- Compare!

[1] SUSE Linux Enterprise

Performance Testing

A (real) story: releasing a new version of the OS

Let’s avoid performance regressions, between **SLE 15-SP3** and **SLE 15-SP4**:

- ALERT: CPU bench is 12% slower on SP4 !!
  - It can be systemd ⇒ different version
  - It can be the kernel ⇒ different version

15-SP3:
- glibc 2.31
- systemd 246
- kernel 5.3.18
- QEMU 5.2.0
- Libvirt 7.1.0

15-SP4:
- glibc 2.31
- systemd 249
- Kernel 5.14.21
- QEMU 6.2.0
- Libvirt 7.1.0

- Note that:
  - It can’t be glibc ⇒ same version [1]
  - It can’t be QEMU ⇒ different version, but not involved
  - It can’t be Libvirt ⇒ same version and not involved

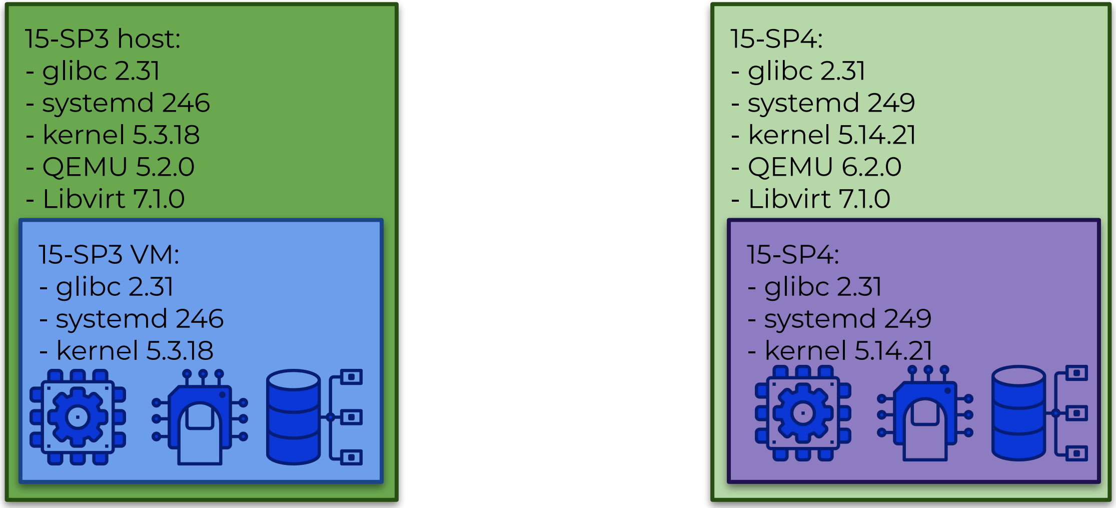
[1] Well, it still can be, due to different patches, etc. Just bear with me, it’s an example

Performance Testing: Enters Virtualization

A (real) story: releasing a new version of the OS

Let’s avoid performance regressions, between **SLE 15-SP3** and **SLE 15-SP4**:

- Run benchmarks CPU bench, I/O benchand MEM bench in a **15-SP3 VM**, on a **15-SP3 host**
- Run benchmarks CPU bench, I/O bench and MEM bench in a **15-SP4 VM**, on a **15-SP4 host**



15-SP3 host:
- glibc 2.31
- systemd 246
- kernel 5.3.18
- QEMU 5.2.0
- Libvirt 7.1.0

15-SP3 VM:
- glibc 2.31
- systemd 246
- kernel 5.3.18

15-SP4:
- glibc 2.31
- systemd 249
- kernel 5.14.21
- QEMU 6.2.0
- Libvirt 7.1.0

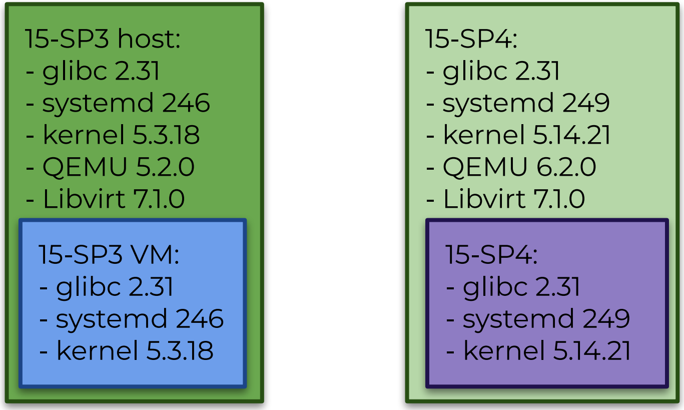
15-SP4:
- glibc 2.31
- systemd 249
- kernel 5.14.21

Performance Testing: Enters Virtualization

The plot thickens…

Let’s avoid performance regressions, between **SLE 15-SP3** and **SLE 15-SP4**:

- ALERT: CPU bench is 12% slower on SP4 !!
  - It can be host systemd
  - It can be guest systemd
  - It can be host kernel
  - It can be guest kernel
  - It can be (host!) QEMU



15-SP3 host:
- glibc 2.31
- systemd 246
- kernel 5.3.18
- QEMU 5.2.0
- Libvirt 7.1.0

15-SP3 VM:
- glibc 2.31
- systemd 246
- kernel 5.3.18

15-SP4:
- glibc 2.31
- systemd 249
- kernel 5.14.21
- QEMU 6.2.0
- Libvirt 7.1.0

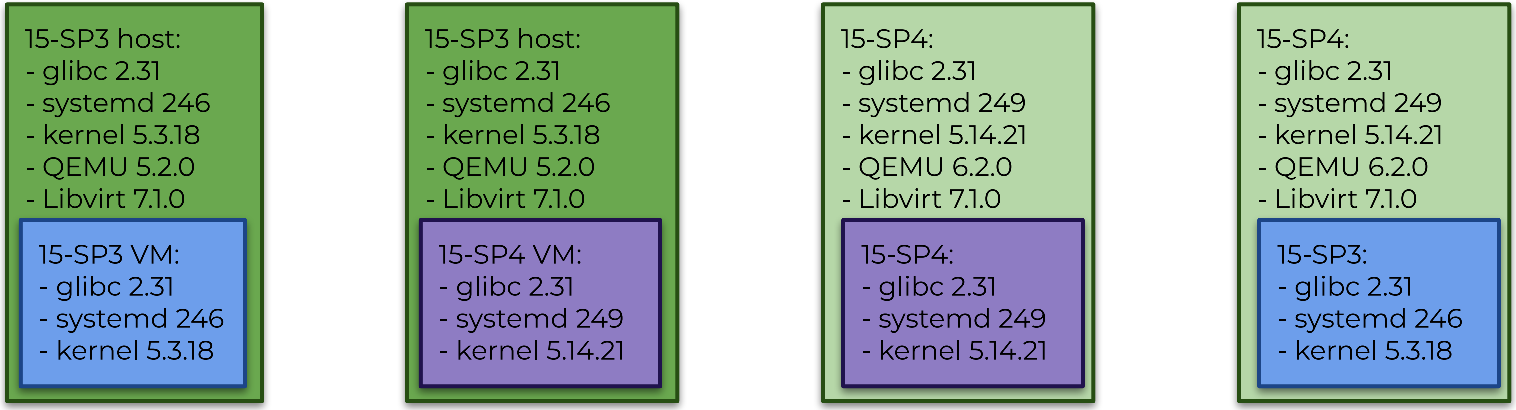
15-SP4:
- glibc 2.31
- systemd 249
- kernel 5.14.21

Performance Testing: Enters Virtualization… Take II

The plot thickens…

More combinations:

- Run benchmarks CPU bench, I/O bench and MEM bench in a **15-SP3 VM**, on a **15-SP3 host**
- Run benchmarks CPU bench, I/O bench and MEM bench in a **15-SP3 VM**, on a **15-SP4 host**
- Run benchmarks CPU bench, I/O bench and MEM bench in a **15-SP4 VM**, on a **15-SP4 host**
- Run benchmarks CPU bench, I/O bench and MEM bench in a **15-SP4 VM**, on a **15-SP3 host**



15-SP3 host:
- glibc 2.31
- systemd 246
- kernel 5.3.18
- QEMU 5.2.0
- Libvirt 7.1.0

15-SP3 VM:
- glibc 2.31
- systemd 246
- kernel 5.3.18

15-SP3 host:
- glibc 2.31
- systemd 246
- kernel 5.3.18
- QEMU 5.2.0
- Libvirt 7.1.0

15-SP4 VM:
- glibc 2.31
- systemd 249
- kernel 5.14.21

15-SP4:
- glibc 2.31
- systemd 249
- kernel 5.14.21
- QEMU 6.2.0
- Libvirt 7.1.0

15-SP4:
- glibc 2.31
- systemd 249
- kernel 5.14.21

15-SP4:
- glibc 2.31
- systemd 249
- kernel 5.14.21
- QEMU 6.2.0
- Libvirt 7.1.0

15-SP3:
- glibc 2.31
- systemd 246
- kernel 5.3.18

Performance Testing: Enters Virtualization… Take III

The plot thickens…

Different VM “sizes”:

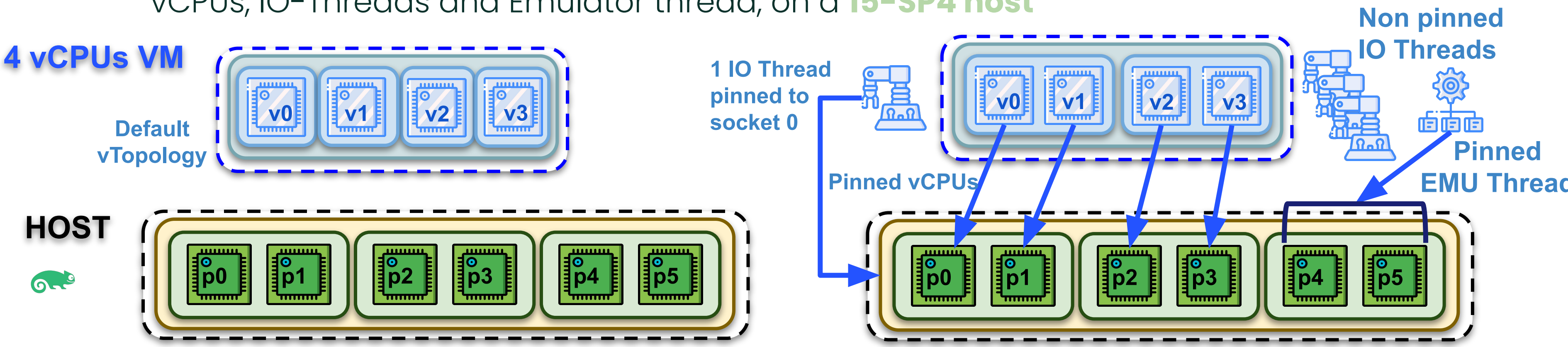
- Run benchmarks in a 1 vCPU 4GB RAM **15-SP3 VM**, on a **15-SP3 host**
- Run benchmarks in a 16 vCPUs 64GB RAM **15-SP3 VM**, on a **15-SP3 host**
- Run benchmarks in a 128 vCPUs 2TB RAM **15-SP3 VM**, on a **15-SP3 host**
- … … …
- Run benchmarks in a 1 vCPU 4GB RAM **15-SP3 VM**, on a **15-SP4 host**
- Run benchmarks in a 16 vCPUs 64GB RAM **15-SP3 VM**, on a **15-SP4 host**
- Run benchmarks in a 128 vCPUs 2TB RAM **15-SP3 VM**, on a **15-SP4 host**
- … … …
- Run benchmarks in a 1 vCPU 4GB RAM **15-SP4 VM**, on a **15-SP4 host**
- Run benchmarks in a 16 vCPUs 64GB RAM **15-SP4 VM**, on a **15-SP4 host**
- Run benchmarks in a 128 vCPUs 2TB RAM **15-SP4 VM**, on a **15-SP4 host**
- … … …

Performance Testing: Enters Virtualization… Take IV

The plot thickens…

Different VM & host configuration (e.g., VM's virtual topology, host-level tuning):

- Run benchmarks in a 4 vCPU 8GB RAM **15-SP3 VM**, default config, on a **15-SP3 host**
- Run benchmarks in a 4 vCPUs 8GB RAM **15-SP3 VM**, with virtual topology, pinned vCPUs, IO-Threads and Emulator threads, on a **15-SP3 host**
- … … …
- Run benchmarks in a 4 vCPU 8GB RAM **15-SP3 VM**, default config, on a **15-SP4 host**
- Run benchmarks in a 4 vCPUs 8GB RAM **15-SP3 VM**, with virtual topology, pinned vCPUs, IO-Threads and Emulator thread, on a **15-SP4 host**

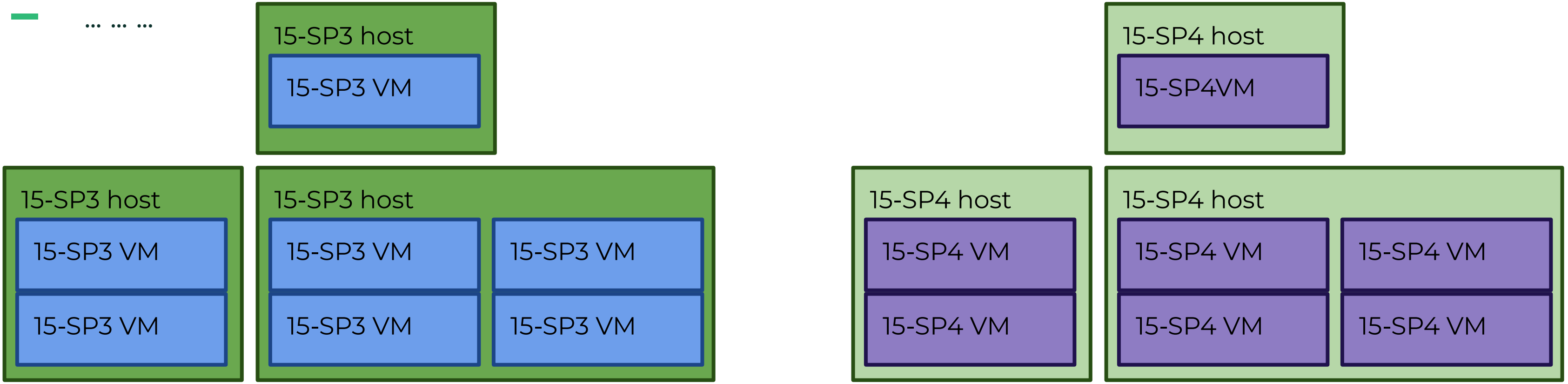


Performance Testing: Enters Virtualization… Take V

The plot thickens…

Multiple VMs:

- Run benchmarks CPU bench, I/O bench and MEM bench in a **15-SP3 VM**, on a **15-SP3 host**
- Run benchmarks CPU bench, I/O bench and MEM bench in 2 **15-SP3 VMs**, on a **15-SP3 host**
- Run benchmarks CPU bench, I/O bench and MEM bench in 4 **15-SP3 VMs**, on a **15-SP3 host**
- … … …
- Run benchmarks CPU bench, I/O bench and MEM bench in a **15-SP4 VM**, on a **15-SP4 host**
- Run benchmarks CPU bench, I/O bench and MEM bench in 2 **15-SP4 VMs**, on a **15-SP4 host**
- Run benchmarks CPU bench, I/O bench and MEM bench in 4 **15-SP4 VMs**, on a **15-SP4 host**
- … … …



15-SP3 host
15-SP3 VM

15-SP4 host
15-SP4VM

15-SP3 host
15-SP3 VM
15-SP3 VM

15-SP3 host
15-SP3 VM
15-SP3 VM
15-SP3 VM
15-SP3 VM

15-SP4 host
15-SP4 VM
15-SP4 VM

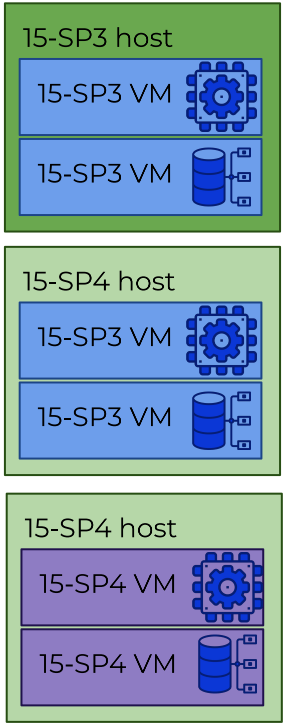
15-SP4 host
15-SP4 VM
15-SP4 VM
15-SP4 VM
15-SP4 VM

Performance Testing: Enters Virtualization… Take VI

The plot thickens…

Heterogeneous workloads, inside the various VMs:

- Run:
  - benchmark CPU bench in a **15-SP3 VM**, on a **15-SP3 host**
  - benchmark I/O bench in a **15-SP3 VM**, on the same **15-SP3 host**
- … … …
- Run:
  - benchmark CPU bench in a **15-SP3 VM**, on a **15-SP4 host**
  - benchmark I/O bench in a **15-SP3 VM**, on the same **15-SP4 host**
- … … …
- Run:
  - benchmark CPU bench in a **15-SP4 VM**, on a **15-SP4 host**
  - benchmark I/O bench in a **15-SP4 VM**, on the same **15-SP4 host**



Performance Testing: Enters Virtualization… Take WTH !!!

<<Infinite Diversity in Infinite Combinations>> (cit.)

Putting everything together:

- For each host OS:
  - Guest OS == host OS …
    - … but also different!
  - In one VM …
    - … but also in > 1 VMs!
  - When all of them have the same size …
    - … but also when they have different sizes!
  - In one (e.g., the default one) host & VMs configuration
    - … but also in different host & VMs configurations!
  - Running the same workloads in all the VMs …
    - … but also running different workloads in each one!

Examples I

VM Size

- Default number of memory pre-allocation threads in QEMU went from #vCPUS to 1
  - (Large) VM startup time was not happy!
  - Learned the hard way with customer bug 1197084

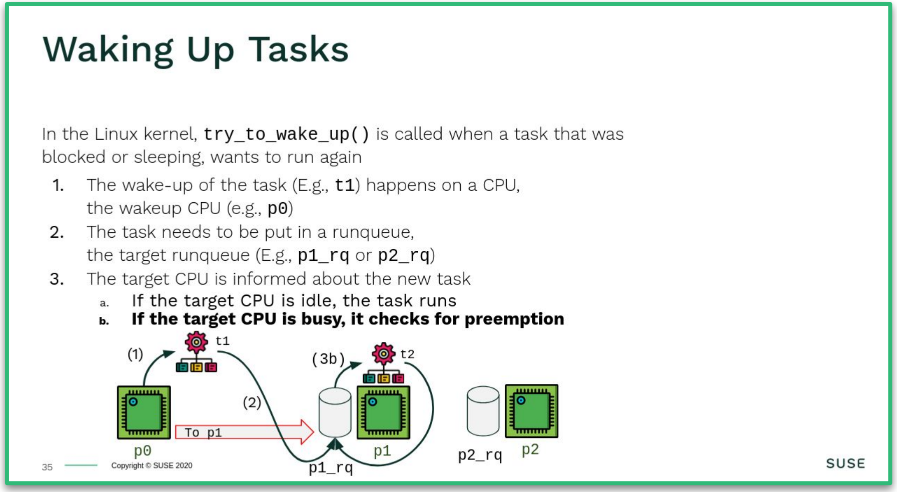
```
|--------|----------------------------------|--------------------------|
| VM RAM | SLES15SP2 = QEMU 5.2.0           | SLES15SP3 == QEMU 6.2.0  |
|--------|----------------------------------|--------------------------|
| 2G     | real    0m2.642s                 | real    0m4.508s         |
|--------|----------------------------------|--------------------------|
| 224G   | real    0m14.867s                | real    0m57.992s        |
|--------|----------------------------------|--------------------------|
| 1024G  | real    0m34.106s                | real    4m10.741s        |
|--------|----------------------------------|--------------------------|
```

Examples II

VM Configuration

- Virtual topology + vCPU pinning resulted in different in-guest kernel behavior when waking up tasks
  - See: Virtual Topology for Virtual Machines: Friend or Foe?

- Presence or absence of an L3 in the virtual topology resulted in glibc to behave differently, inside the guest, in turn resulting in performance anomalies:
  - See: Virtual Topology for Virtual Machines: Friend or Foe?



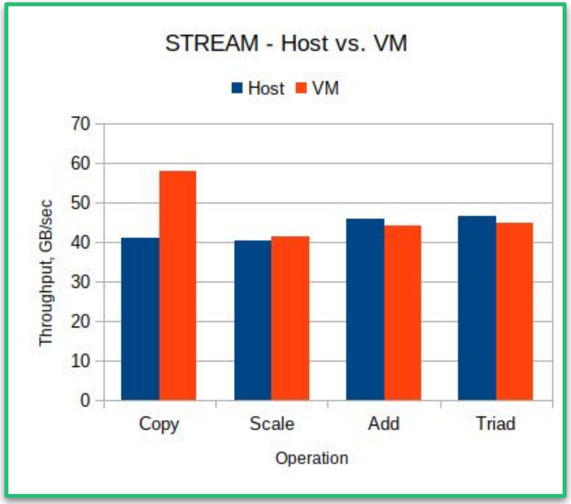
Waking Up Tasks

In the Linux kernel, `try_to_wake_up()` is called when a task that was blocked or sleeping, wants to run again

1. The wake-up of the task (E.g., `t1`) happens on a CPU, the wakeup CPU (e.g., `p0`)
2. The task needs to be put in a runqueue, the target runqueue (E.g., `p1_rq` or `p2_rq`)
3. The target CPU is informed about the new task
   a. If the target CPU is idle, the task runs
   b. **If the target CPU is busy, it checks for preemption**

35 Copyright © SUSE 2020

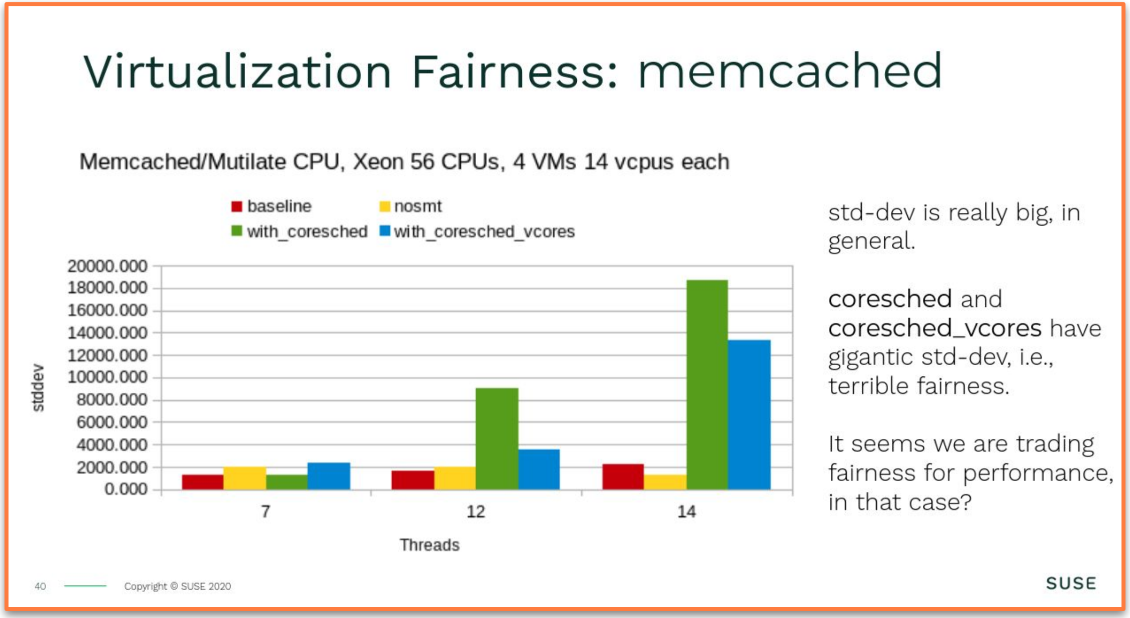
SUSE



Examples III

Multiple VMs

- Fairness issues with the Linux Core-Scheduling patch, identified by benchmarking with multiple VMs
  - See: Core Scheduling, Some USe Cases, OSPM 2020



Virtualization Fairness: memcached

std-dev is really big, in general.

coresched and coresched_vcores have gigantic std-dev, i.e., terrible fairness.

It seems we are trading fairness for performance, in that case?

40 Copyright © SUSE 2020 SUSE

Virtualization Performance Test Suites, Anyone?

MMTests

The tool for the job

"MMTests is a configurable test suite that runs a number of common workloads of interest to MM developers." (Sept. 2012, LKML)

⇒ Now it is a lot more !

⇒ Including Virtualization !!!

- Fetches, builds, configures & runs a (set of) benchmark(s)
- https://github.com/gormanm/mmtests
- GH PRs, email to Mel Gorman or myself
- Bash & Perl

There has been previous talks.

- Me, FOSDEM 2020, Automated Performance Testing for Virtualization with MMTests
- Me, OSPM 2020, Scheduler benchmarking with MMTests
- Me, SUSE Labs Conference 2021, (Not Just) VM BEnchmarking with MMTests: Some Updates
- Mel Gorman, SUSE Labs Conference 2018, Marvin: Automated assistant for development and CI
- Jan Kara, Open Source Summit 2017, Detecting Performance Regressions in the Linux Kernel
- Jan Kara, SUSE Labs Conference 2017, The Performance Team's Grid
- Davidlohr Bueso, LinuxCon NA 2015, Performance Monitoring in the Linux Kernel

MMTests

Supported Benchmarks

Already preconfigured (only a subset, in no particular order):

- `pgbench`, `sysbench-oltp` (`mariadb` and `postgres`), `pgioperf`, `...`
- `bonnie`, `fio`, `filebench`, `iozone`, `dbench4`, `...`
- `redis`, `memcached`, `john-the-ripper`, `ebizzy`, `phpbench`, `apachebench`, `siege`, `. . .`
- `nas-pb`, `parsec`, `openfoam`, `kernbench`, `stream`, `. . .`
- `hackbench`, `schbench`, `cyclictest`, `unixbench`, `. . .`
- `netperf`, `iperf`, `sockperf`, `tbench`, `. . .`
- *And new ones are actively being added*

Custom ones:

- Linux kernel load balancer, program startup time, THP scaling

Workloads like:

- git workload, kernel dev. Workload, shell-scripts workload

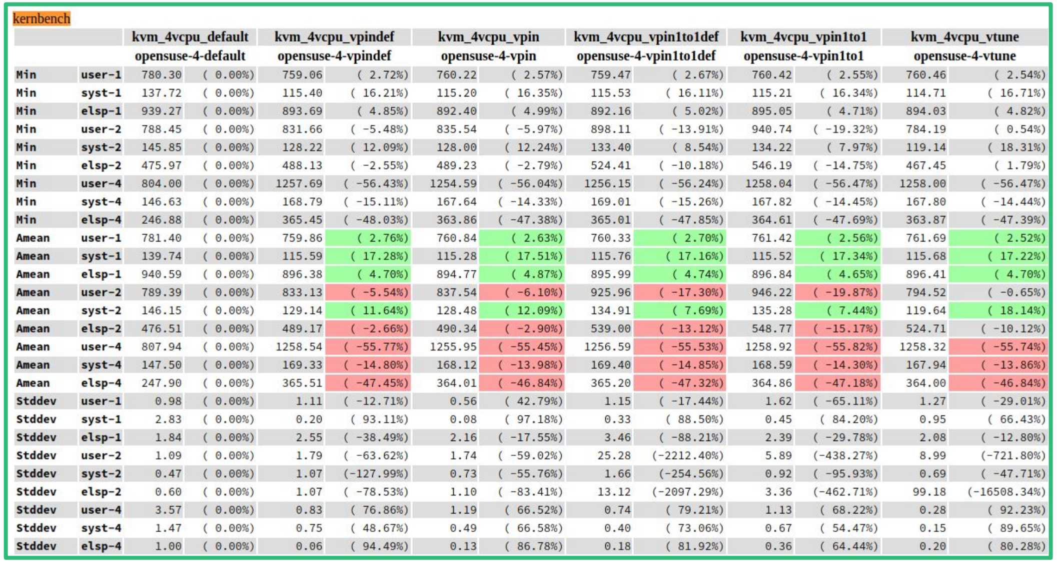
Can run multiple of them:

- Sequentially or in parallel

MMTests

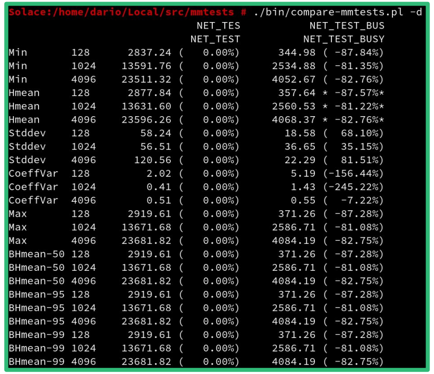
Results and Stats

- Each benchmark is run multiple times (configurable) for statistical significance
  - Collects and store configuration info and results
- Can do comparisons and statistic analysis:
  - A-mean, H-mean, Geo-mean, significance, percentiles, …

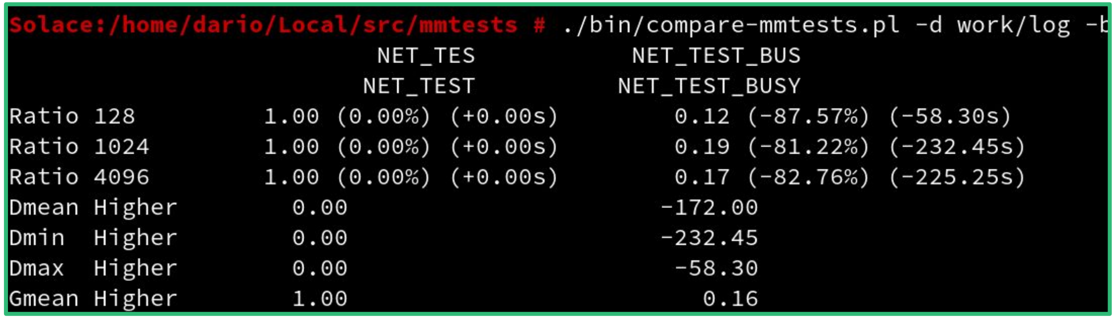


kernbench

		kvm_4vcpu_default		kvm_4vcpu_vpindef		kvm_4vcpu_vpin		kvm_4vcpu_vpin1to1def		kvm_4vcpu_vpin1to1		kvm_4vcpu_vtune	
		opensuse-4-default		opensuse-4-vpindef		opensuse-4-vpin		opensuse-4-vpin1to1def		opensuse-4-vpin1to1		opensuse-4-vtune	
Min	user-1	780.30	(0.00%)	759.06	(2.72%)	760.22	(2.57%)	759.47	(2.67%)	760.42	(2.55%)	760.46	(2.54%)
Min	syst-1	137.72	(0.00%)	115.40	(16.21%)	115.20	(16.35%)	115.53	(16.11%)	115.21	(16.34%)	114.71	(16.71%)
Min	elsp-1	939.27	(0.00%)	893.69	(4.85%)	892.40	(4.99%)	892.16	(5.02%)	895.05	(4.71%)	894.03	(4.82%)
Min	user-2	788.45	(0.00%)	831.66	(-5.48%)	835.54	(-5.97%)	898.11	(-13.91%)	940.74	(-19.32%)	784.19	(0.54%)
Min	syst-2	145.85	(0.00%)	128.22	(12.09%)	128.00	(12.24%)	133.40	(8.54%)	134.22	(7.97%)	119.14	(18.31%)
Min	elsp-2	475.97	(0.00%)	488.13	(-2.55%)	489.23	(-2.79%)	524.41	(-10.18%)	546.19	(-14.75%)	467.45	(1.79%)
Min	user-4	804.00	(0.00%)	1257.69	(-56.43%)	1254.59	(-56.04%)	1256.15	(-56.24%)	1258.04	(-56.47%)	1258.00	(-56.47%)
Min	syst-4	146.63	(0.00%)	168.79	(-15.11%)	167.64	(-14.33%)	169.01	(-15.26%)	167.82	(-14.45%)	167.80	(-14.44%)
Min	elsp-4	246.88	(0.00%)	365.45	(-48.03%)	363.86	(-47.38%)	365.01	(-47.85%)	364.61	(-47.69%)	363.87	(-47.39%)
Amean	user-1	781.40	(0.00%)	759.86	(2.76%)	760.84	(2.63%)	760.33	(2.70%)	761.42	(2.56%)	761.69	(2.52%)
Amean	syst-1	139.74	(0.00%)	115.59	(17.28%)	115.28	(17.51%)	115.76	(17.16%)	115.52	(17.34%)	115.68	(17.22%)
Amean	elsp-1	940.59	(0.00%)	896.38	(4.70%)	894.77	(4.87%)	895.99	(4.74%)	896.84	(4.65%)	896.41	(4.70%)
Amean	user-2	789.39	(0.00%)	833.13	(-5.54%)	837.54	(-6.10%)	925.96	(-17.30%)	946.22	(-19.87%)	794.52	(-0.65%)
Amean	syst-2	146.15	(0.00%)	129.14	(11.64%)	128.48	(12.09%)	134.91	(7.69%)	135.28	(7.44%)	119.64	(18.14%)
Amean	elsp-2	476.51	(0.00%)	489.17	(-2.66%)	490.34	(-2.90%)	539.00	(-13.12%)	548.77	(-15.17%)	524.71	(-10.12%)
Amean	user-4	807.94	(0.00%)	1258.54	(-55.77%)	1255.95	(-55.45%)	1256.59	(-55.53%)	1258.92	(-55.82%)	1258.32	(-55.74%)
Amean	syst-4	147.50	(0.00%)	169.33	(-14.80%)	168.12	(-13.98%)	169.40	(-14.85%)	168.59	(-14.30%)	167.94	(-13.86%)
Amean	elsp-4	247.90	(0.00%)	365.51	(-47.45%)	364.01	(-46.84%)	365.20	(-47.32%)	364.86	(-47.18%)	364.00	(-46.84%)
Stddev	user-1	0.98	(0.00%)	1.11	(-12.71%)	0.56	(42.79%)	1.15	(-17.44%)	1.62	(-65.11%)	1.27	(-29.01%)
Stddev	syst-1	2.83	(0.00%)	0.20	(93.11%)	0.08	(97.18%)	0.33	(88.50%)	0.45	(84.20%)	0.95	(66.43%)
Stddev	elsp-1	1.84	(0.00%)	2.55	(-38.49%)	2.16	(-17.55%)	3.46	(-88.21%)	2.39	(-29.78%)	2.08	(-12.80%)
Stddev	user-2	1.09	(0.00%)	1.79	(-63.62%)	1.74	(-59.02%)	25.28	(-2212.40%)	5.89	(-438.27%)	8.99	(-721.80%)
Stddev	syst-2	0.47	(0.00%)	1.07	(-127.99%)	0.73	(-55.76%)	1.66	(-254.56%)	0.92	(-95.93%)	0.69	(-47.71%)
Stddev	elsp-2	0.60	(0.00%)	1.07	(-78.53%)	1.10	(-83.41%)	13.12	(-2097.29%)	3.36	(-462.71%)	99.18	(-16508.34%)
Stddev	user-4	3.57	(0.00%)	0.83	(76.86%)	1.19	(66.52%)	0.74	(79.21%)	1.13	(68.22%)	0.28	(92.23%)
Stddev	syst-4	1.47	(0.00%)	0.75	(48.67%)	0.49	(66.58%)	0.40	(73.06%)	0.67	(54.47%)	0.15	(89.65%)
Stddev	elsp-4	1.00	(0.00%)	0.06	(94.49%)	0.13	(86.78%)	0.18	(81.92%)	0.36	(64.44%)	0.20	(80.28%)



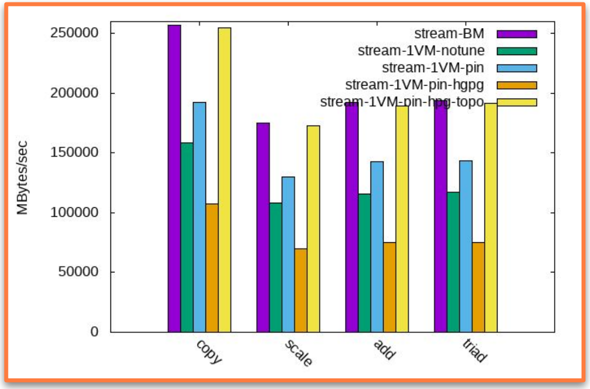
```
Solace:/home/dario/Local/src/mmtests # ./bin/compare-mmtests.pl -d
                               NET_TES                NET_TEST_BUS
                              NET_TEST               NET_TEST_BUSY
Min         128       2837.24 (   0.00%)      344.98 ( -87.84%)
Min         1024     13591.76 (   0.00%)     2534.88 ( -81.35%)
Min         4096     23511.32 (   0.00%)     4052.67 ( -82.76%)
Hmean       128       2877.84 (   0.00%)      357.64 * -87.57%*
Hmean       1024     13631.60 (   0.00%)     2560.53 * -81.22%*
Hmean       4096     23596.26 (   0.00%)     4068.37 * -82.76%*
Stddev      128         58.24 (   0.00%)       18.58 (  68.10%)
Stddev      1024        56.51 (   0.00%)       36.65 (  35.15%)
Stddev      4096       120.56 (   0.00%)       22.29 (  81.51%)
CoeffVar    128          2.02 (   0.00%)        5.19 (-156.44%)
CoeffVar    1024         0.41 (   0.00%)        1.43 (-245.22%)
CoeffVar    4096         0.51 (   0.00%)        0.55 (  -7.22%)
Max         128       2919.61 (   0.00%)      371.26 ( -87.28%)
Max         1024     13671.68 (   0.00%)     2586.71 ( -81.08%)
Max         4096     23681.82 (   0.00%)     4084.19 ( -82.75%)
BHmean-50 128         2919.61 (   0.00%)      371.26 ( -87.28%)
BHmean-50 1024       13671.68 (   0.00%)     2586.71 ( -81.08%)
BHmean-50 4096       23681.82 (   0.00%)     4084.19 ( -82.75%)
BHmean-95 128         2919.61 (   0.00%)      371.26 ( -87.28%)
BHmean-95 1024       13671.68 (   0.00%)     2586.71 ( -81.08%)
BHmean-95 4096       23681.82 (   0.00%)     4084.19 ( -82.75%)
BHmean-99 128         2919.61 (   0.00%)      371.26 ( -87.28%)
BHmean-99 1024       13671.68 (   0.00%)     2586.71 ( -81.08%)
BHmean-99 4096       23681.82 (   0.00%)     4084.19 ( -82.75%)
```

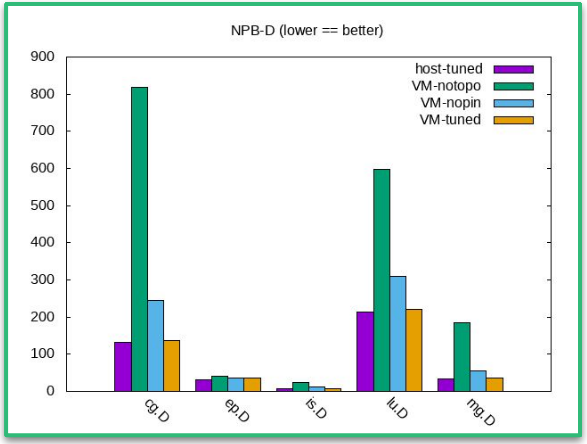


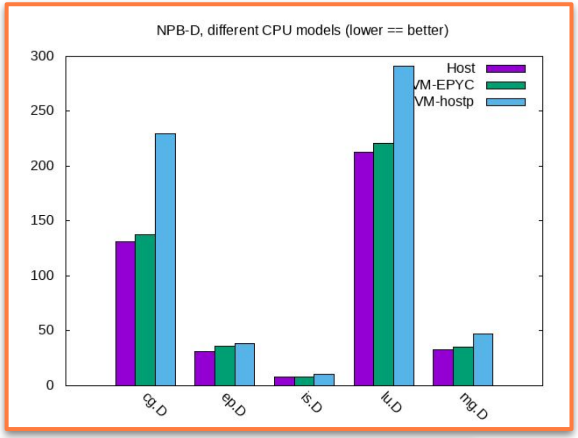
```
Solace:/home/dario/Local/src/mmtests # ./bin/compare-mmtests.pl -d work/log -b
                           NET_TES              NET_TEST_BUS
                          NET_TEST             NET_TEST_BUSY
Ratio 128         1.00 (0.00%) (+0.00s)       0.12 (-87.57%) (-58.30s)
Ratio 1024        1.00 (0.00%) (+0.00s)       0.19 (-81.22%) (-232.45s)
Ratio 4096        1.00 (0.00%) (+0.00s)       0.17 (-82.76%) (-225.25s)
Dmean Higher         0.00                     -172.00
Dmin  Higher         0.00                     -232.45
Dmax  Higher         0.00                      -58.30
Gmean Higher         1.00                        0.16
```

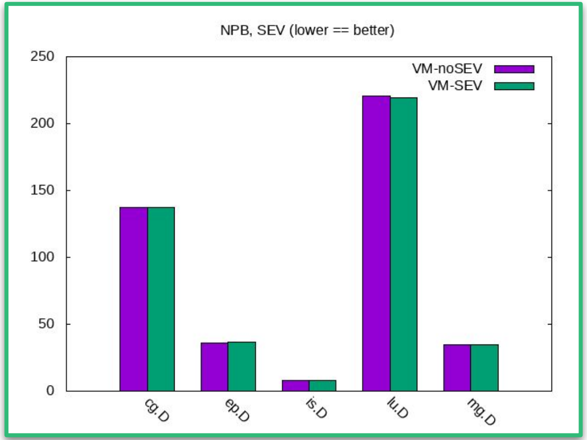
MMTests

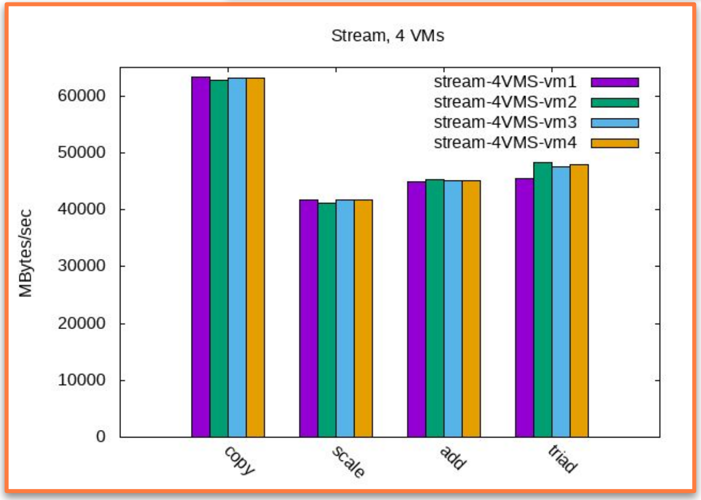
Plots







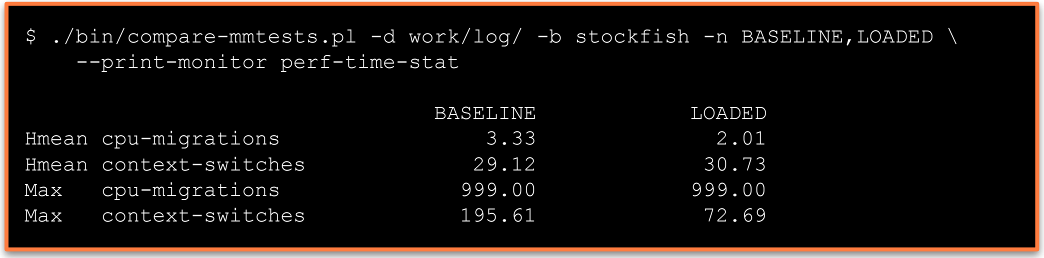




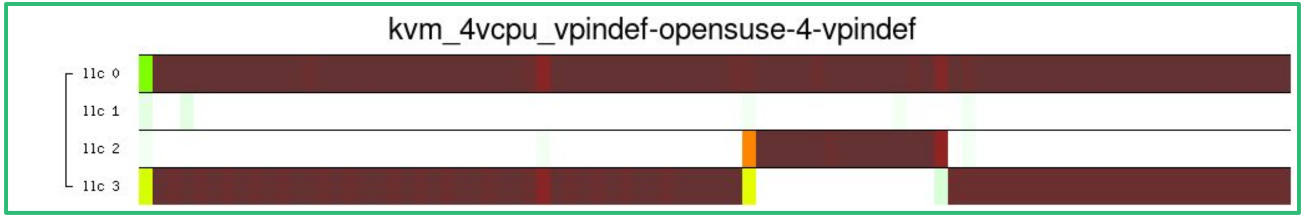
MMTests

Monitors

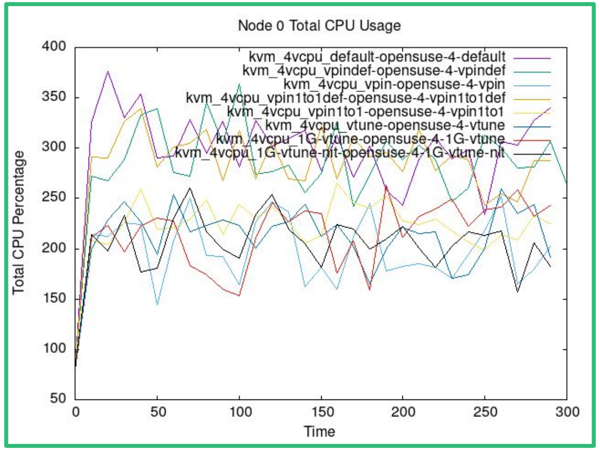
- While the benchmark is running:
  - Samples:
    - top, mpstat, vmstat, iostat, df, …
  - Collect data from:
    - perf, ftrace, …
- Plots monitors too



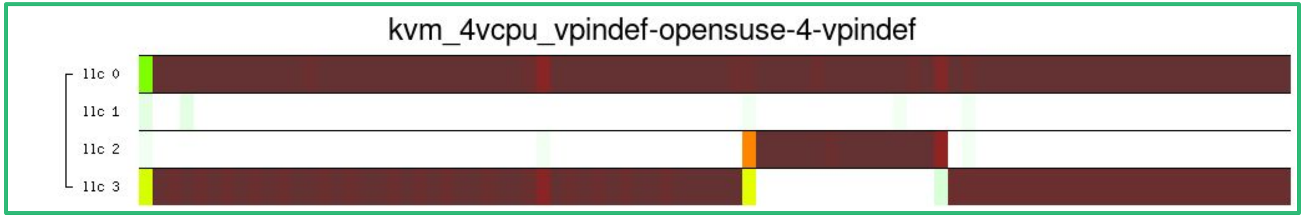
```
$ ./bin/compare-mmtests.pl -d work/log/ -b stockfish -n BASELINE,LOADED \
    --print-monitor perf-time-stat

                                     BASELINE            LOADED
Hmean  cpu-migrations                    3.33              2.01
Hmean  context-switches                 29.12             30.73
Max    cpu-migrations                  999.00            999.00
Max    context-switches                195.61             72.69
```



kvm_4vcpu_vpindef-opensuse-4-vpindef



Node 0 Total CPU Usage



kvm_4vcpu_vpindef-opensuse-4-vpindef

MMTests

Benchmark Config Files

- Collection of bash `export`-ed variables
  - They're script themselves (can contain commands!)

```
# MM Test Parameters
export MMTESTS="stream"

. $SHELLPACK_INCLUDE/include-sizes.sh
get_numa_details

# Test disk to setup (optional)
#export TESTDISK_PARTITION=/dev/sda6
#export TESTDISK_FILESYSTEM=xfs
#export TESTDISK_MKFS_PARAM="-f -d agcount=8"

# List of monitors
export RUN_MONITOR=yes
export MONITORS_ALWAYS=
export MONITORS_GZIP="proc-vmstat top"
export MONITORS_WITH_LATENCY="vmstat"
export MONITOR_UPDATE_FREQUENCY=10
```

```
# stream
export STREAM_SIZE=$((1048576*3*2048))
export STREAM_THREADS=$((NUMNODES*2))
export STREAM_METHOD=omp
export STREAM_ITERATIONS=5
export OMP_PROC_BIND=SPREAD
export MMTESTS_BUILD_CFLAGS="-m64 -lm -Ofast
   -march=znver1 -mcmodel=medium -DOFFSET=512"
```

MMTests

Benchmark Config Files

- Collection of bash `export`-ed variable
  - They're script themselves (can contain

- In Virtualization, these config files are shipped to and used inside of the guest(s)
- Monitoring defined here happens inside the guest(s) too ⇒ How the benchmark threads run on the vCPU

```
# MM Test Parameters
export MMTESTS="stream"

. $SHELLPACK_INCLUDE/include-sizes.sh
get_numa_details

# Test disk to setup (optional)
#export TESTDISK_PARTITION=/dev/sda6
#export TESTDISK_FILESYSTEM=xfs
#export TESTDISK_MKFS_PARAM="-f -d agcount=8"

# List of monitors
export RUN_MONITOR=yes
export MONITORS_ALWAYS=
export MONITORS_GZIP="proc-vmstat top"
export MONITORS_WITH_LATENCY="vmstat"
export MONITOR_UPDATE_FREQUENCY=10
```

```
# stream
export STREAM_SIZE=$((1048576*3*2048))
export STREAM_THREADS=$((NUMNODES*2))
export STREAM_METHOD=omp
export STREAM_ITERATIONS=5
export OMP_PROC_BIND=SPREAD
export MMTESTS_BUILD_CFLAGS="-m64 -lm -Ofast
   -march=znver1 -mcmodel=medium -DOFFSET=512"
```

- Can query the system characteristics.
- Benchmark parameters can depend on that
- Specific configurations, for each benchmark; Check the shellpaks for details (TODO: improve the docs about this)

MMTests

Host Config Files

- Collection of bash `export`-ed variables
  - They’re script themselves (can contain commands!)

```
# Example MM Test host config file, for run-kvm.sh
export MMTESTS_HOST_IP="192.168.122.1"
export MMTESTS_AUTO_PACKAGE_INSTALL="yes"

export MMTESTS_VM=vm1,vm2

export MMTESTS_NUMA_POLICY="numad"
export MMTESTS_TUNED_PROFILE="latency-performance"

# List of monitors
export RUN_MONITOR=yes
export MONITORS_ALWAYS=
export MONITORS_GZIP="proc-vmstat mpstat"
export MONITORS_WITH_LATENCY="vmstat"
export MONITOR_PERF_EVENTS=cpu-migrations
export MONITOR_UPDATE_FREQUENCY=30
```

MMTests

Virtual Machines - fully managed mode

MMTests on the host manages the VMs' lifecycle via Libvirt [*]

- Can create VMs from Libvirt xml file
- Can automatically provision/install VMs
  - VM configuration is customizable in the host config file
- Start, restart, shutdown the VMs as it sees fit

```
# From the host config file:
export MMTESTS_VMS="vm1,vm2"

# Generic values, will be used for all VMs,
# if not otherwise specified
export MMTESTS_VMS_CPUS=4
export MMTESTS_VMS_MEMORY=8129

vm1_CPUS=2
vm1_MEMORY=4096

vm2_CPUS=6
vm2_MEMORY=6144
```

[*] Some of this is not merged. Available here: https://github.com/dfaggioli/mmtests/tree/wip/bench-virt

MMTests

Containers

MMTests on the “host” manages the containers' lifecycle via `podman` or `docker`

- Can build containers from Dockerfile
- Can create/pull down containers from registries
  - Containers configuration is customizable in the host config file
- Start, restart, shutdown the containers as it sees fit

```
# From the host config file:
export MMTESTS_VMS=”cnt1,cnt2”

# Generic values, will be used for all containers,
# if not otherwise specified
export MMTESTS_CONTS_REGISTRY="registry.opensuse.org"
export MMTESTS_CONTS_IMAGE="opensuse/tumbleweed:latest"
export MMTESTS_CONTS_DOCKERFILE=./bin-cont/Dockerfile/opensuse-tumbleweed.Dockerfile

export cnt1_IMAGE="opensuse/leap:latest"
export cnt2_DOCKERFILE=/var/foo/bar/Dockerfiles/my_cnt.Dockerfile
```

MMTests

Generic “remote entity” mode

MMTests **does not** manage the entities where the benchmarks run

- It only knows their IP addresses
- It is not even aware of what they are!
  - VMs ?
  - Containers ?
  - KubeVirt VMIs ?
  - Physical Hosts ? [*]
- **Can’t** start, restart, deploy, etc
  - Still useful if wanting to run benchmarks, but MMTets support is not there yet

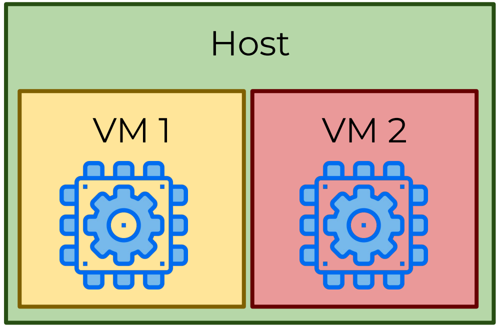
```
# From the host config file:
export MMTESTS_VMS_IP="192.168.122.24 192.168.122.38"
export MMTESTS_VMS="vmi_a vmi_b"
```

[*] Should work just fine. But never tested!

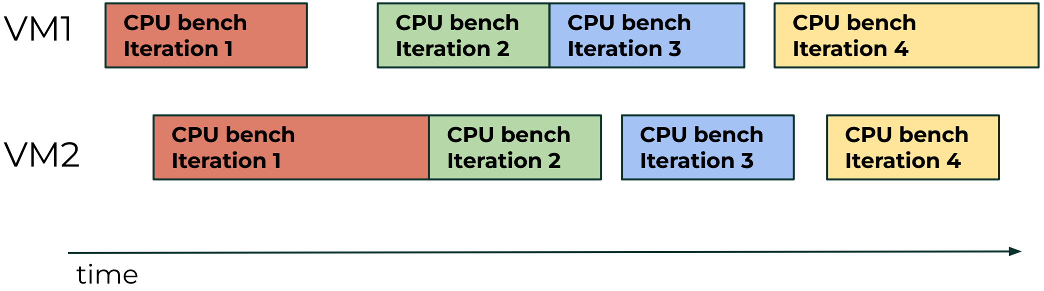
Running Benchmarks in Multiple VMs

We need a bit of a special benchmarking suite

- What's the performance of CPU bench, running concurrently in 2 VMs?



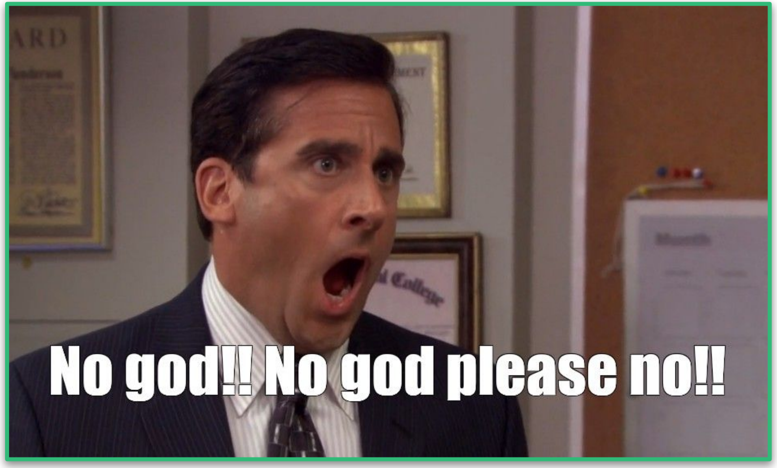
Just start the benchmarks inside the VMs, and let them run?



Running Benchmarks in Multiple VMs

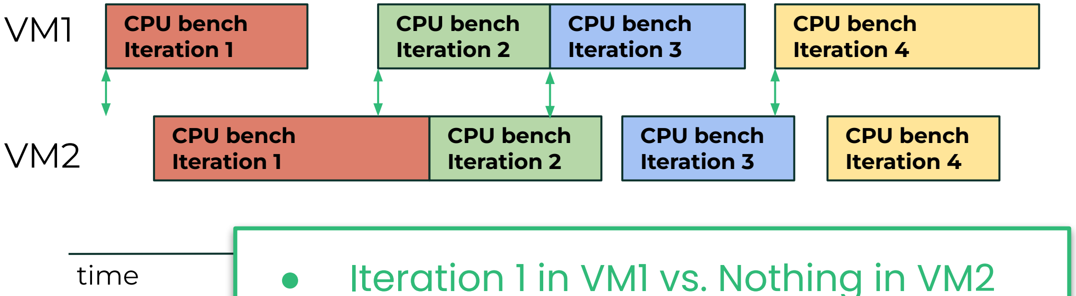
We need a bit of a special benchmarking suite

- What's the performance of CPU bench running concurrently in 2 VMs?



Just start the benchmarks inside the VMs,

and let them run?



VM1: CPU bench Iteration 1 | CPU bench Iteration 2 | CPU bench Iteration 3 | CPU bench Iteration 4

VM2: CPU bench Iteration 1 | CPU bench Iteration 2 | CPU bench Iteration 3 | CPU bench Iteration 4

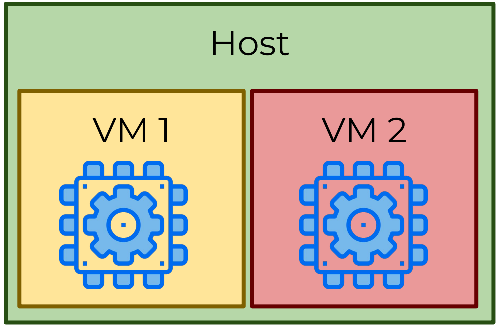
time

- Iteration 1 in VM1 vs. Nothing in VM2
- Iteration 2 in VM2 vs. Iteration 1 in VM2
- Iteration 3 in VM1 vs. Iteration 2 in VM1
- Iteration 4 in VM1 vs. Iteration 3 in VM2

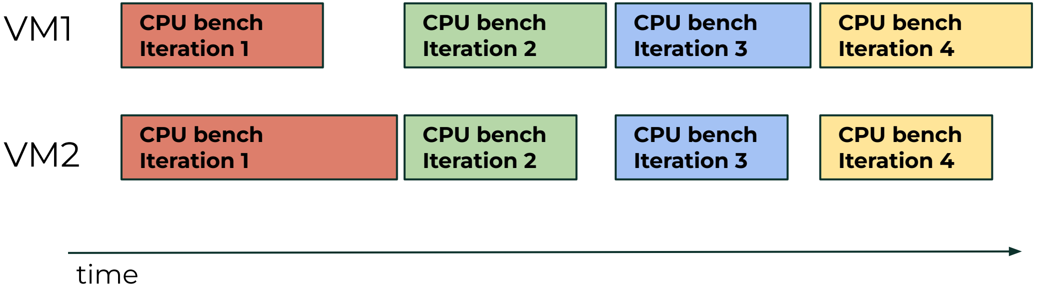
Running Benchmarks in Multiple VMs

We need a bit of a special benchmarking suite

- What's the performance of CPU bench running concurrently in 2 VMs?



Synchronization of each (iteration of each) benchmark inside the various VMs:



Running Benchmarks in Multiple VMs

We need a bit of a special benchmarking suite

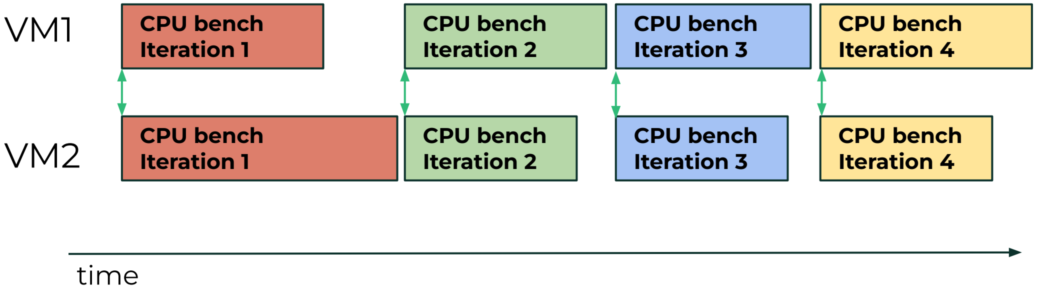
- What's the performance of CPU bench running concur

Synchronizat

benchmark i

Iterations **start** in sync!

- In all VMs
- No matter when the previous one finished



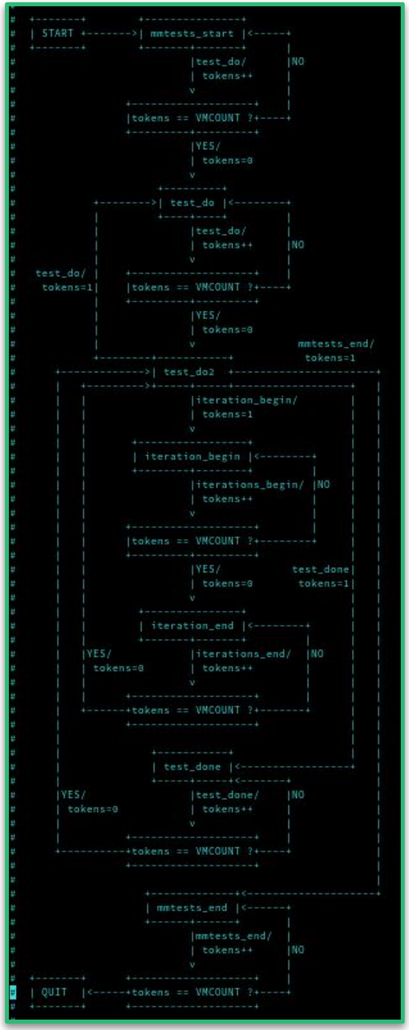
MMTests :Synchronized Runs & Iterations

It’s worth checking out the code, even just for this ASCII diagram! :-D

Achieving synchronization:

- Host and the VMs communicate
- Token passing protocol
  - VMs do not talk to each other
  - All VMs talk to the host
- The host implements the “barriers”
  - Before the start of a new benchmark
  - Before each iteration of the same benchmark

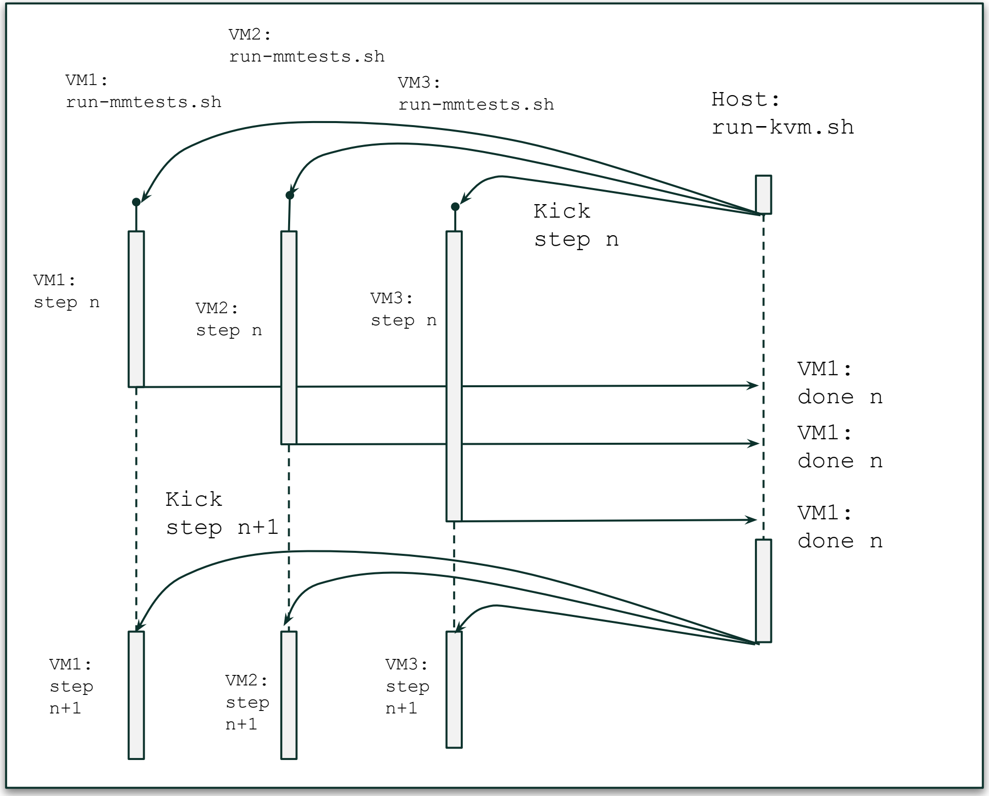
ASCII block diagram of the protocol

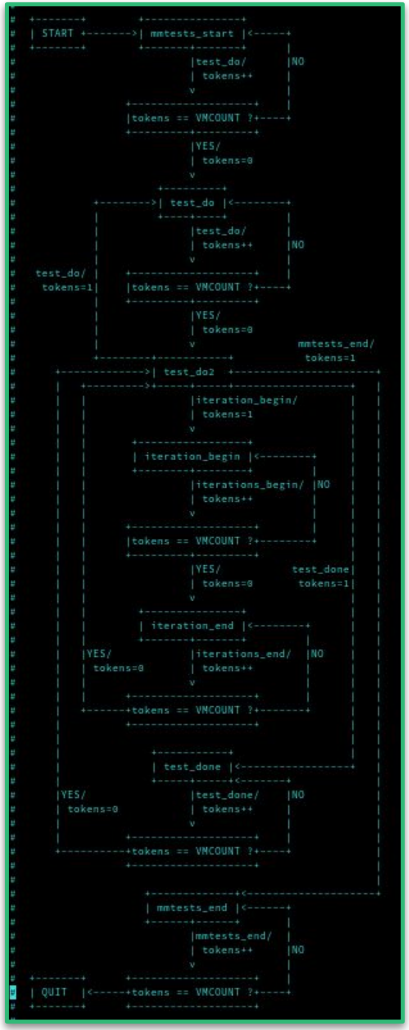


```
#  +-------+        +---------------+
#  | START +------->| mmtests_start |<-----+
#  +-------+        +-------+-------+      |
#                           |test_do/      |NO
#                           | tokens++     |
#                           v              |
#                 +-------------------+    |
#                 |tokens == VMCOUNT ?+----+
#                 +---------+---------+
#                           |YES/
#                           | tokens=0
#                           v
#                      +---------+
#            +-------->| test_do |<--------+
#            |         +----+----+         |
#            |              |test_do/      |
#            |              | tokens++     |NO
#            |              v              |
#   test_do/ |    +-------------------+    |
#    tokens=1|    |tokens == VMCOUNT ?+----+
#            |    +---------+---------+
#            |              |YES/
#            |              | tokens=0
#            |              v               mmtests_end/
#            +--------+-----------+          tokens=1
#      +------------->| test_do2  +----------------------+
#      |   +--------->+-----+-----+------------------+   |
#      |   |                |iteration_begin/        |   |
#      |   |                | tokens=1               |   |
#      |   |                v                        |   |
#      |   |       +-----------------+               |   |
#      |   |       | iteration_begin |<--------+     |   |
#      |   |       +--------+--------+         |     |   |
#      |   |                |iterations_begin/ |NO   |   |
#      |   |                | tokens++         |     |   |
#      |   |                v                  |     |   |
#      |   |      +-------------------+        |     |   |
#      |   |      |tokens == VMCOUNT ?+--------+     |   |
#      |   |      +---------+---------+              |   |
#      |   |                |YES/           test_done|   |
#      |   |                | tokens=0       tokens=1|   |
#      |   |                v                        |   |
#      |   |        +---------------+                |   |
#      |   |        | iteration_end |<--------+      |   |
#      |   |        +-------+-------+         |      |   |
#      |   |YES/            |iterations_end/  |NO    |   |
#      |   | tokens=0       | tokens++        |      |   |
#      |   |                v                 |      |   |
#      |   |      +-------------------+       |      |   |
#      |   +------+tokens == VMCOUNT ?+-------+      |   |
#      |          +-------------------+              |   |
#      |                                             |   |
#      |              +-----------+                  |   |
#      |              | test_done |<-----------------+   |
#      |              +-----+-----+<-------+             |
#      |YES/                |test_done/    |NO           |
#      | tokens=0           | tokens++     |             |
#      |                    v              |             |
#      |          +-------------------+    |             |
#      +----------+tokens == VMCOUNT ?+----+             |
#                 +-------------------+                  |
#                                                        |
#                    +-------------+<--------------------+
#                    | mmtests_end |<-----+
#                    +------+------+      |
#                           |mmtests_end/ |
#                           | tokens++    |NO
#                           v             |
#  +-------+      +-------------------+   |
#  | QUIT  |<-----+tokens == VMCOUNT ?+---+
#  +-------+      +-------------------+
```

MMTests: Synchronized Runs & Iterations

And this is even prettier... :-P

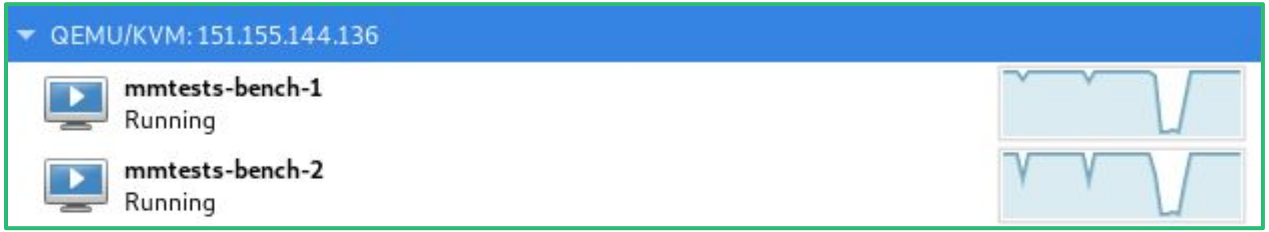




```
#  +-------+        +---------------+
#  | START +------->| mmtests_start |<-----+
#  +-------+        +-------+-------+      |
#                           |test_do/      |NO
#                           | tokens++     |
#                           v              |
#                 +-------------------+    |
#                 |tokens == VMCOUNT ?+----+
#                 +---------+---------+
#                           |YES/
#                           | tokens=0
#                           v
#                      +---------+
#            +-------->| test_do |<--------+
#            |         +----+----+         |
#            |              |test_do/      |
#            |              | tokens++     |NO
#            |              v              |
#   test_do/ |    +-------------------+    |
#    tokens=1|    |tokens == VMCOUNT ?+----+
#            |    +---------+---------+
#            |              |YES/
#            |              | tokens=0
#            |              v                mmtests_end/
#            +--------+-----------+           tokens=1
#      +------------->| test_do2  +----------------------+
#      |   +--------->+-----+-----+------------------+   |
#      |   |                |iteration_begin/        |   |
#      |   |                | tokens=1               |   |
#      |   |                v                        |   |
#      |   |       +-----------------+               |   |
#      |   |       | iteration_begin |<--------+     |   |
#      |   |       +--------+--------+         |     |   |
#      |   |                |iterations_begin/ |NO   |   |
#      |   |                | tokens++         |     |   |
#      |   |                v                  |     |   |
#      |   |      +-------------------+        |     |   |
#      |   |      |tokens == VMCOUNT ?+--------+     |   |
#      |   |      +---------+---------+              |   |
#      |   |                |YES/           test_done|   |
#      |   |                | tokens=0       tokens=1|   |
#      |   |                v                        |   |
#      |   |        +---------------+                |   |
#      |   |        | iteration_end |<--------+      |   |
#      |   |        +-------+-------+         |      |   |
#      |   |YES/            |iterations_end/  |NO    |   |
#      |   | tokens=0       | tokens++        |      |   |
#      |   |                v                 |      |   |
#      |   |      +-------------------+       |      |   |
#      |   +------+tokens == VMCOUNT ?+-------+      |   |
#      |          +-------------------+              |   |
#      |                                             |   |
#      |              +-----------+                  |   |
#      |              | test_done |<-----------------+   |
#      |              +-----+-----+<------+              |
#      |YES/                |test_done/    |NO           |
#      | tokens=0           | tokens++     |             |
#      |                    v              |             |
#      |          +-------------------+    |             |
#      +----------+tokens == VMCOUNT ?+----+             |
#                 +-------------------+                  |
#                                                        |
#                    +-------------+<--------------------+
#                    | mmtests_end |<------+
#                    +------+------+       |
#                           |mmtests_end/  |
#                           | tokens++     |NO
#                           v              |
#  +-------+      +-------------------+    |
#  | QUIT  |<-----+tokens == VMCOUNT ?+----+
#  +-------+      +-------------------+
```

MMTests: Synchronized Runs & Iterations

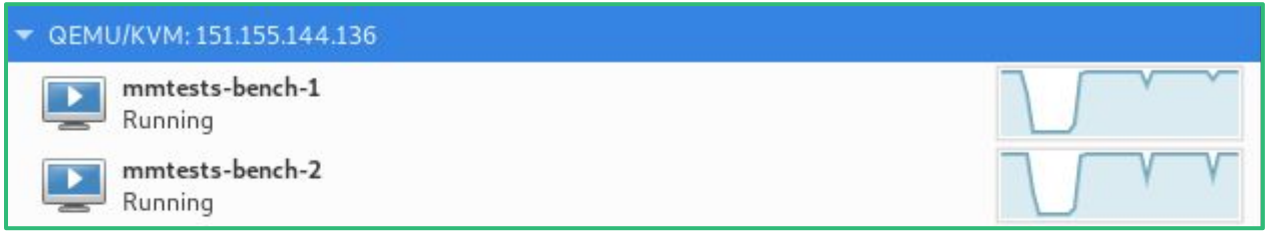
Some 2 VMs Examples



QEMU/KVM: 151.155.144.136

mmtests-bench-1
Running

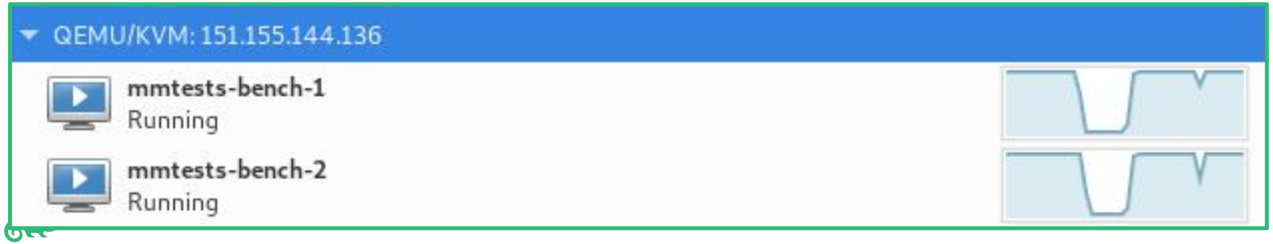
mmtests-bench-2
Running



QEMU/KVM: 151.155.144.136

mmtests-bench-1
Running

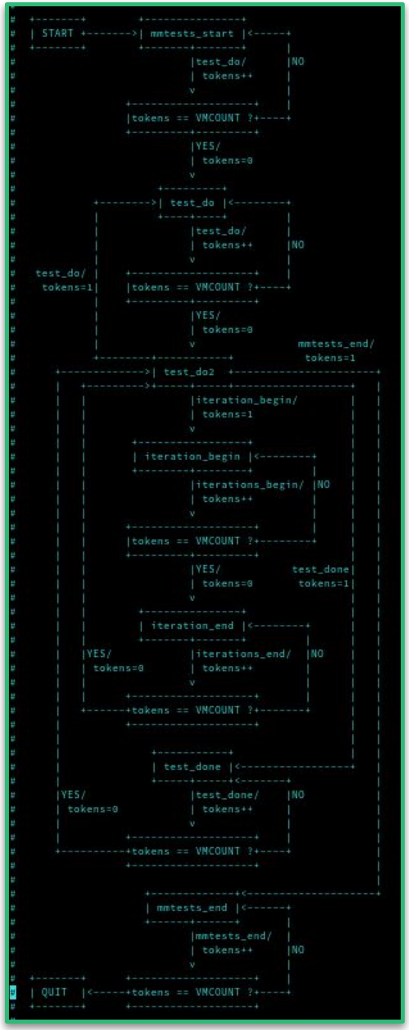
mmtests-bench-2
Running



QEMU/KVM: 151.155.144.136

mmtests-bench-1
Running

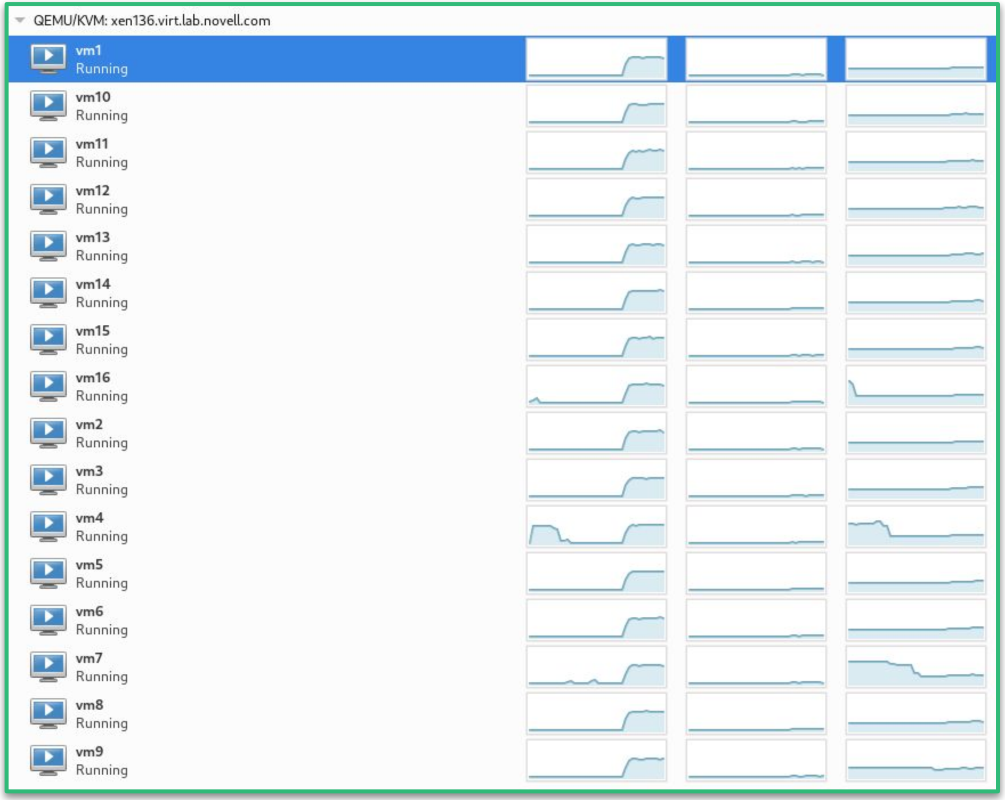
mmtests-bench-2
Running



```
#  +-------+       +---------------+
#  | START +------>| mmtests_start |<-----+
#  +-------+       +-------+-------+      |
#                          |test_do/      |NO
#                          | tokens++     |
#                          v              |
#                +-------------------+    |
#                |tokens == VMCOUNT ?+----+
#                +---------+---------+
#                          |YES/
#                          | tokens=0
#                          v
#                     +---------+
#            +------->| test_do |<--------+
#            |        +----+----+         |
#            |             |test_do/      |
#            |             | tokens++     |NO
#            |             v              |
#   test_do/ |   +-------------------+    |
#    tokens=1|   |tokens == VMCOUNT ?+----+
#            |   +---------+---------+
#            |             |YES/
#            |             | tokens=0
#            |             v               mmtests_end/
#            +-------+-----------+          tokens=1
#      +------------>| test_do2  +----------------------+
#      |   +-------->+-----+-----+------------------+   |
#      |   |               |iteration_begin/        |   |
#      |   |               | tokens=1               |   |
#      |   |               v                        |   |
#      |   |     +-----------------+                |   |
#      |   |     | iteration_begin |<--------+      |   |
#      |   |     +--------+--------+         |      |   |
#      |   |               |iterations_begin/ |NO   |   |
#      |   |               | tokens++         |     |   |
#      |   |               v                  |     |   |
#      |   |    +-------------------+         |     |   |
#      |   |    |tokens == VMCOUNT ?+---------+     |   |
#      |   |    +---------+---------+               |   |
#      |   |               |YES/          test_done|   |
#      |   |               | tokens=0       tokens=1|   |
#      |   |               v                        |   |
#      |   |       +---------------+                |   |
#      |   |       | iteration_end |<--------+      |   |
#      |   |       +-------+-------+         |      |   |
#      |   |YES/           |iterations_end/  |NO    |   |
#      |   | tokens=0      | tokens++        |      |   |
#      |   |               v                 |      |   |
#      |   |    +-------------------+        |      |   |
#      |   +-----+tokens == VMCOUNT ?+-------+      |   |
#      |         +-------------------+              |   |
#      |                                            |   |
#      |             +-----------+                  |   |
#      |             | test_done |<-----------------+   |
#      |             +-----+-----+<-------+             |
#      |YES/               |test_done/    |NO           |
#      | tokens=0          | tokens++     |             |
#      |                   v              |             |
#      |         +-------------------+    |             |
#      +---------+tokens == VMCOUNT ?+----+             |
#                +-------------------+                  |
#                                                       |
#                   +-------------+<--------------------+
#                   | mmtests_end |<------+
#                   +------+------+       |
#                          |mmtests_end/  |
#                          | tokens++     |NO
#                          v              |
#  +-------+     +-------------------+    |
#  | QUIT  |<----+tokens == VMCOUNT ?+----+
#  +-------+     +-------------------+
#
```

MMTests: Synchronized Runs & Iterations

A 16 VMs Example



QEMU/KVM: xen136.virt.lab.novell.com

vm1
Running

vm10
Running

vm11
Running

vm12
Running

vm13
Running

vm14
Running

vm15
Running

vm16
Running

vm2
Running

vm3
Running

vm4
Running

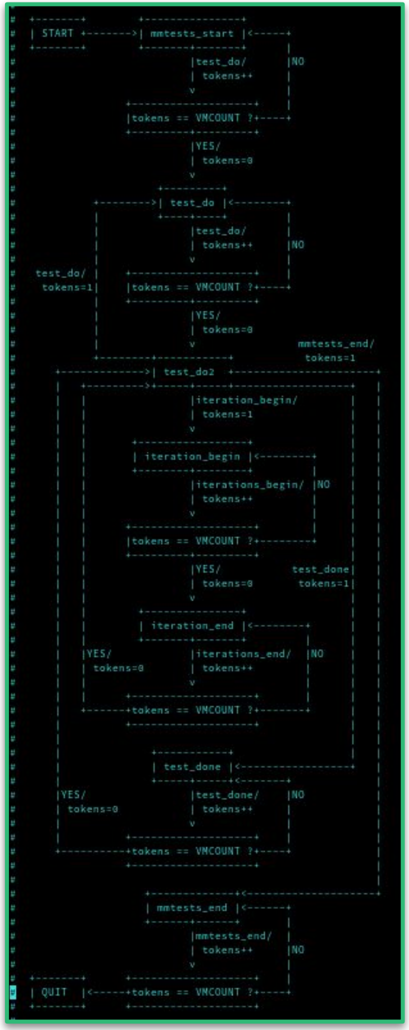
vm5
Running

vm6
Running

vm7
Running

vm8
Running

vm9
Running



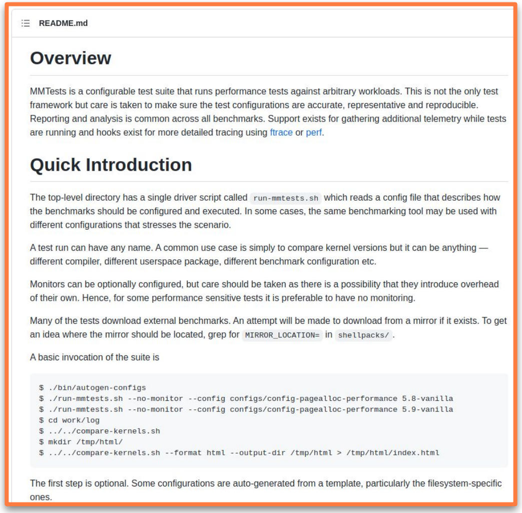
```
#  +-------+        +---------------+
#  | START +------->| mmtests_start |<-----+
#  +-------+        +-------+-------+      |
#                           |test_do/      |NO
#                           | tokens++     |
#                           v              |
#                 +-------------------+    |
#                 |tokens == VMCOUNT ?+----+
#                 +---------+---------+
#                           |YES/
#                           | tokens=0
#                           v
#                      +---------+
#            +-------->| test_do |<--------+
#            |         +----+----+         |
#            |              |test_do/      |
#            |              | tokens++     |NO
#            |              v              |
#   test_do/ |    +-------------------+    |
#    tokens=1|    |tokens == VMCOUNT ?+----+
#            |    +---------+---------+
#            |              |YES/
#            |              | tokens=0
#            |              v               mmtests_end/
#            +--------+-----------+           tokens=1
#      +------------->| test_do2  +----------------------+
#      |   +--------->+-----+-----+------------------+   |
#      |   |                |iteration_begin/        |   |
#      |   |                | tokens=1               |   |
#      |   |                v                        |   |
#      |   |       +-----------------+               |   |
#      |   |       | iteration_begin |<--------+     |   |
#      |   |       +--------+--------+         |     |   |
#      |   |                |iterations_begin/ |NO   |   |
#      |   |                | tokens++         |     |   |
#      |   |                v                  |     |   |
#      |   |      +-------------------+        |     |   |
#      |   |      |tokens == VMCOUNT ?+--------+     |   |
#      |   |      +---------+---------+              |   |
#      |   |                |YES/           test_done|   |
#      |   |                | tokens=0       tokens=1|   |
#      |   |                v                        |   |
#      |   |        +---------------+                |   |
#      |   |        | iteration_end |<--------+      |   |
#      |   |        +-------+-------+         |      |   |
#      |   |YES/            |iterations_end/  |NO    |   |
#      |   | tokens=0       | tokens++        |      |   |
#      |   |                v                 |      |   |
#      |   |      +-------------------+       |      |   |
#      |   +------+tokens == VMCOUNT ?+-------+      |   |
#      |          +-------------------+              |   |
#      |                                             |   |
#      |              +-----------+                  |   |
#      |              | test_done |<-----------------+   |
#      |              +-----+-----+<-------+             |
#      |YES/                |test_done/    |NO           |
#      | tokens=0           | tokens++     |             |
#      |                    v              |             |
#      |          +-------------------+    |             |
#      +----------+tokens == VMCOUNT ?+----+             |
#                 +-------------------+                  |
#                                                        |
#                    +-------------+<--------------------+
#                    | mmtests_end |<------+
#                    +------+------+       |
#                           |mmtests_end/   |
#                           | tokens++     |NO
#                           v              |
#  +-------+      +-------------------+    |
#  | QUIT  |<-----+tokens == VMCOUNT ?+----+
#  +-------+      +-------------------+
```

MMTests

Documentation

There’s something:

- On GitHub
- In-line help & manpages
- It’s incomplete :-(
  - We’re working on it…



Overview

MMTests is a configurable test suite that runs performance tests against arbitrary workloads. This is not the only test framework but care is taken to make sure the test configurations are accurate, representative and reproducible. Reporting and analysis is common across all benchmarks. Support exists for gathering additional telemetry while tests are running and hooks exist for more detailed tracing using ftrace or perf.

Quick Introduction

The top-level directory has a single driver script called `run-mmtests.sh` which reads a config file that describes how the benchmarks should be configured and executed. In some cases, the same benchmarking tool may be used with different configurations that stresses the scenario.

A test run can have any name. A common use case is simply to compare kernel versions but it can be anything — different compiler, different userspace package, different benchmark configuration etc.

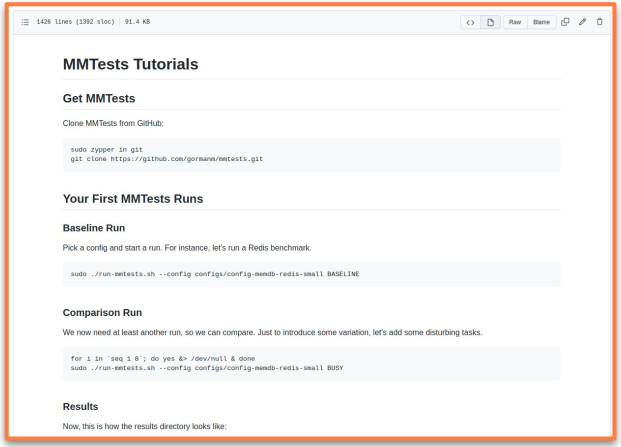
Monitors can be optionally configured, but care should be taken as there is a possibility that they introduce overhead of their own. Hence, for some performance sensitive tests it is preferable to have no monitoring.

Many of the tests download external benchmarks. An attempt will be made to download from a mirror if it exists. To get an idea where the mirror should be located, grep for `MIRROR_LOCATION=` in `shellpacks/`.

A basic invocation of the suite is

```
$ ./bin/autogen-configs
$ ./run-mmtests.sh --no-monitor --config configs/config-pagealloc-performance 5.8-vanilla
$ ./run-mmtests.sh --no-monitor --config configs/config-pagealloc-performance 5.9-vanilla
$ cd work/log
$ ../../compare-kernels.sh
$ mkdir /tmp/html/
$ ../../compare-kernels.sh --format html --output-dir /tmp/html > /tmp/html/index.html
```

The first step is optional. Some configurations are auto-generated from a template, particularly the filesystem-specific ones.



MMTests Tutorials

Get MMTests

Clone MMTests from GitHub:

```
sudo zypper in git
git clone https://github.com/gormanm/mmtests.git
```

Your First MMTests Runs

Baseline Run

Pick a config and start a run. For instance, let's run a Redis benchmark.

```
sudo ./run-mmtests.sh --config configs/config-memdb-redis-small BASELINE
```

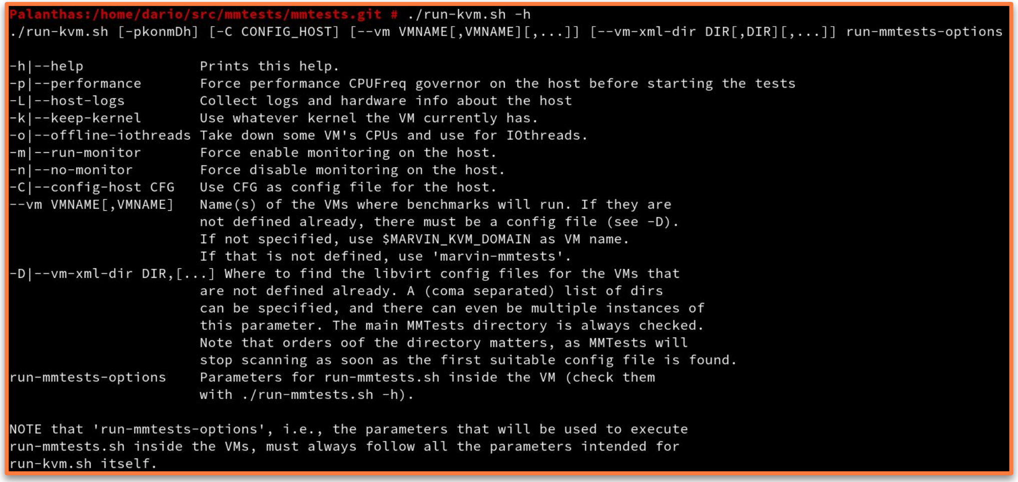
Comparison Run

We now need at least another run, so we can compare. Just to introduce some variation, let's add some disturbing tasks.

```
for i in `seq 1 8`; do yes &> /dev/null & done
sudo ./run-mmtests.sh --config configs/config-memdb-redis-small BUSY
```

Results

Now, this is how the results directory looks like:



```
Palanthas:/home/dario/src/mmtests/mmtests.git # ./run-kvm.sh -h
./run-kvm.sh [-pkonmDh] [-C CONFIG_HOST] [--vm VMNAME[,VMNAME][,...]] [--vm-xml-dir DIR[,DIR][,...]] run-mmtests-options

-h|--help              Prints this help.
-p|--performance       Force performance CPUFreq governor on the host before starting the tests
-L|--host-logs         Collect logs and hardware info about the host
-k|--keep-kernel       Use whatever kernel the VM currently has.
-o|--offline-iothreads Take down some VM's CPUs and use for IOthreads.
-m|--run-monitor       Force enable monitoring on the host.
-n|--no-monitor        Force disable monitoring on the host.
-C|--config-host CFG   Use CFG as config file for the host.
--vm VMNAME[,VMNAME]   Name(s) of the VMs where benchmarks will run. If they are
                       not defined already, there must be a config file (see -D).
                       If not specified, use $MARVIN_KVM_DOMAIN as VM name.
                       If that is not defined, use 'marvin-mmtests'.
-D|--vm-xml-dir DIR,[...] Where to find the libvirt config files for the VMs that
                       are not defined already. A (coma separated) list of dirs
                       can be specified, and there can even be multiple instances of
                       this parameter. The main MMTests directory is always checked.
                       Note that orders oof the directory matters, as MMTests will
                       stop scanning as soon as the first suitable config file is found.
run-mmtests-options    Parameters for run-mmtests.sh inside the VM (check them
                       with ./run-mmtests.sh -h).

NOTE that 'run-mmtests-options', i.e., the parameters that will be used to execute
run-mmtests.sh inside the VMs, must always follow all the parameters intended for
run-kvm.sh itself.
```

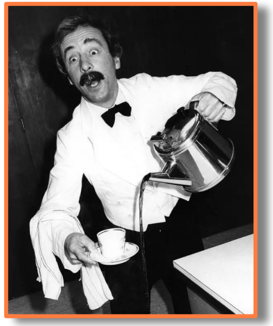
Building a Virtualization CI Around MMTests

MMTests @ SUSE Kernel Performance Team

Meet Marvin (see Marvin: Automated assistant for development and CI)

Used internally, Linux kernel performance testing. Reports sent to LKML.

- *Marvin* : reserves machines, manages deployments (with autoyast), copies MMTests across, executes tests and copies results back
- *Bob The Builder* : monitors kernel trees, trigger (re)builds
- *Johnny Bravo* : generating reports
- *Manual* : developer tool (manual queueing)
- *Sentinel* : “guards” against regressions
- *Impera* : bisection
- *Janus* : For distro comparisons



MMTests-CI

Another Marvin? Well, different needs, different environment/lab

Code available at: https://github.com/dfaggioli/mmtests-ci.

⇒ Still WiP, not sure it’s worth checking it out for now…

- Runs entirely on the test-host. No ext. Controller machine
  - Try to avoid reprovisioning the host at each run
  - If we want/need that, the host need to “self-reprovision” itself
    - Doable in our lab, with internal scripts
- Multiple OSes allowed on the same host, in different partitions
  - Cycle through all of them
- Runs periodically
  - No change/commit/update triggered. Not yet, at least
    - Too many test cases, too few servers :-(
- Code & configuration: all in that git repo
  - Checkout the repo, change things like the followings and commit:
    - Adding/changing scripts for running tests
    - Altering an OS’ test-plan
  - At the next run (for that OS) changes will be picked up and be effective

Running on 2 servers (not both 100% dedicated to this yet)

```
AMD EPYC 7713
Online CPU(s):          256
Thread(s) per core:     2
Core(s) per socket:     64
Socket(s):              2
NUMA node(s):           2
RAM:                    1.2Ti
```

```
AMD EPYC 7452
Online CPU(s):          64
Thread(s) per core:     2
Core(s) per socket:     32
Socket(s):              1
NUMA node(s):           1
RAM:                    62GB
```

MMTests-CI

Another Marvin? Kind of. We have different needs + different environment/lab

Code available at: https://github.com/dfaggioli/mmtests-ci.

⇒ Still WiP, not sure it's worth checking it out for now…

- Each OS has a test-plan
  - Example of tests: distro RPMs, upstream QEMU, upstream kernel, etc
  - Each test can consist of:
    - Multiple VM configurations:
      - single VM, multi-VMs, different VM sizes, host & guest tuning
    - Multiple benchmarks:
      - run one after the other, in all the configurations
- Tests (can) have setup phases. Done before starting running the benchmarks
- It's possible to reboot the server (even multiple times)
  - During setup. E.g., for installing a specific kernel
  - Between & during tests the tests. E.g., between benchmarks or group of benchmarks

Running on 2 servers
(not both 100% dedicated
to this yet)

```
AMD EPYC 7713
Online CPU(s):          256
Thread(s) per core:     2
Core(s) per socket:     64
Socket(s):              2
NUMA node(s):           2
RAM:                    1.2Ti
```

```
AMD EPYC 7452
Online CPU(s):          64
Thread(s) per core:     2
Core(s) per socket:     32
Socket(s):              1
NUMA node(s):           1
RAM:                    62GB
```

MMTests-CI @ SUSE

Our Own Downstream Testing: on-going (with some parts still WiP)

E.g., testing the virtualization stack of our currently supported OSes

1. Boot the server into one of our supported OSes, say, SLE 15-SP4
2. Try to updates all the OS packages
3. Did anything change since previous run ?
   - E.g., packages receiving maintenance updates & backport
4. If yes, run the benchmarks
   - Baremetal (can be useful, for reference)
   - In 1 VM, multiple sizes, multiple configurations
   - In multiple VMs, multiple sizes, multiple configurations
5. Store results
   - Check for regressions
6. Optional: [Re]Provision another OS (say, SLE 15-SP3) in a different partition on the server
7. Boot into that OS
   - There, go through all these same steps
   - Then reboot into yet another partition/OS
   - At some point, one of those other OSes on the server will reboot it into "us"
8. Go back to 1

MMTests-CI @ QEMU Community

Upstream Testing: Working on It

E.g., testing the last two released versions of QEMU

1. Boot the server into openSUSE Tumbleweed
2. Updates all the OS packages
3. Did anything change since previous run ?
   - E.g., new versions of some packages (Tumbleweed is rolling!)
4. If yes:
   - Download QEMU 7.1.0. Build it. Install it.
   - Run the benchmarks:
     - In 1 VM, multiple sizes, multiple configurations
     - In multiple VMs, multiple sizes, multiple configurations
   - Download QEMU 7.0.0. Build it. Install it
   - Run the benchmarks:
     - In 1 VM, multiple sizes, multiple configurations
     - In multiple VMs, multiple sizes, multiple configurations
5. Store results
   - Check for regressions
6. Go back to 1

MMTests-CI @ QEMU Community

Upstream Testing: Working on It

E.g., testing the “latest” QEMU git commit:

1. Boot the server into … What ?
2. Updates all the OS packages … Or not ?
3. Has the tip of QEMU git been updated since last run ?
4. If yes:
   - Pull QEMU master, with the latest changes. Build it. Install it
   - Run the benchmarks:
     - In 1 VM, multiple sizes, multiple configurations
     - In multiple VMs, multiple sizes, multiple configurations
5. Store results
   - Check for regressions
6. Go back to 1

MMTests-CI @ QEMU Community

Analyzing results. Triaging and reporting

Making the results available:

- Full logs and results preserved
- Generate per-*{host, OS, test, benchmark}* MMTests' dashboard
  - Should be fine to publish them
- Comparison baseline:
  - Milestones, if any
  - If not, move (for now, manually) it forward each week/month/..., if results are consistent

When a regression is identified:

- Triage and "bisect"
  - Manually, for now
  - By our team

Reporting to upstream:

- Similar to what Lukáš is doing here
- Manually, done by us: no automatic reports/email, *for now*

Are There Any Questions ?

~~Are There Any Questions ?~~

I Have Some Questions !

MMTests-CI @ QEMU Community

Feedback Wanted! On what baseline shall we test the latest git ?

"Boot the server into … What ?"

- openSUSE Tumbleweed (or anything that may have received updates, since the last run)
  - Packages versions on host ⇒ changes
  - QEMU codebase ⇒ changes

- **GOOD:** We always test latest git against an OS with the most recent software components (e.g., new kernels!)
- **BAD:** How do we know which of the two (set of) changes caused a regression ?

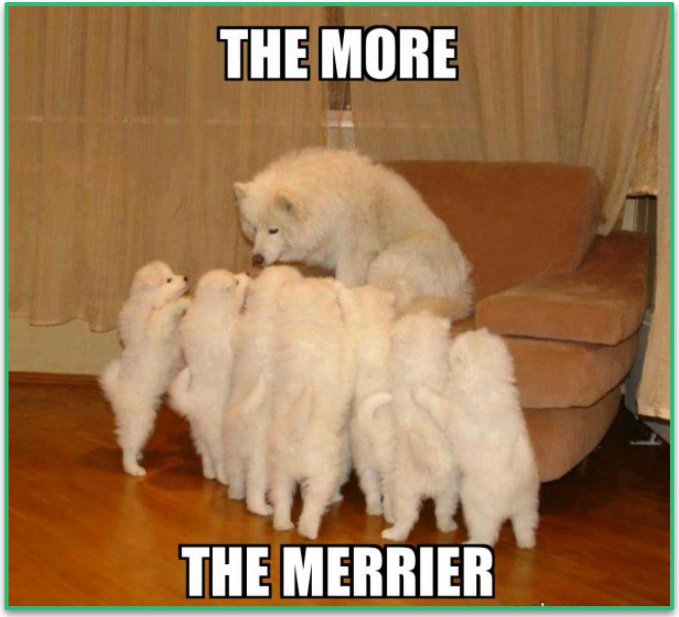
- Frozen (or rarely updated) host OS
  - Packages versions on host ⇒ never change
  - QEMU codebase ⇒ changes

- **GOOD:** If there's a difference between two runs, we know for sure from where it comes
- **BAD:** With time, host OS software will become stale

MMTests-CI @ QEMU Community

Choices! Choices! Choices!

- How many benchmarks ?
  - Which ones ?
- How many different configurations ?
  - Which ones ?
- How many different VM sizes ?
  - Which one
- … … …



Isn’t it just the more, the merrier ? Yes! But:

- The more, the longer it takes:
  - Longer time spans between consecutive iterations of the same benchmark
  - Harder to identify changes responsible of regressions
- The more, the higher the volume of data produced:
  - Time consuming/difficult to analyze

MMTests-CI @ QEMU Community

Feedback Wanted! Do the following proposals/ideas make sense ?

“Run the benchmarks” ⇒ Which ones?

- CPU benchmarks:
  - `nas-pb`, `kernbench`, `sysbench`, `hackbench`
- Memory benchmarks:
  - `Stream`, `memcached`
- I/O benchmarks:
  - `fio`, `iozone`, `sockperf`, `netperf`
- More complex “workloads”:
  - VM Startup time, cyclictest + hackbench, ebizzy

MMTests-CI @ QEMU Community

Feedback Wanted! Do the following proposals/ideas make sense ?

“In 1 VM, multiple sizes, multiple configurations” ⇒ Which ones?

- Sizes:
  - # vCPUs = 1, 2, ½ of the host pCPUs, same as the host pCPUs
  - RAM = 2GB, ½ of the host RAM, 9/10 of the host RAM
- Configuration:
  - Just default
  - With IO-threads
  - With vCPU & memory pinning + virtual topology
  - With PCI-Passthrough / SR-IOV

MMTests-CI @ QEMU Community

Feedback Wanted! Do the following proposals/ideas make sense ?

“In multiple VMs, multiple sizes, multiple configurations” ⇒ Which ones?

- Sizes:
  - # vCPUs: 1, 8, 16
- # VMs:
  - A few: Tot # vCPUS = ½ of the host pCPUs
  - A lot: Tot # of vCPUs = same as the host pCPUs
  - Overload: Tot # vCPUS = 1.5 times the # of host pCPUs

MMTests-CI @ QEMU Community

Feedback Wanted! Not sure how to deal with host OS != guest OS cases …

“Did anything change since previous run ?”

- Host changes:
  - Easy to check (the CI scripts run on the host)
- Guest changes:
  - E.g., I’m on a SLE 15-SP4 host. I want to test a 15-SP3 VM
    - No updates for the host since the last test run I did here
    - An updated kernel have been released for 15-SP3, since the last run
    - But the VM is off, until I actually decide to start the test… So how do I figure that out ?

⇒ Boot all the VMs and check for updates inside of them for deciding whether to rerun ?

Seems the proper solution, but it’s cumbersome and complex

⇒ Somehow setup alerts for (potential) guest updates. Use them to force a test run

Doable… With a lot of test specific alerts/trigger (random OS update available, upstream kernel available, etc)

⇒ Once in a while, run the test anyway, even if it would seem unnecessary ?

Easy, but potentially wasteful

How long is “a while” ?

⇒ Only do “homogenous runs”, i.e., host OS == (all) gues(s) OS(es)

MMTests-CI @ QEMU Community

Feedback Wanted! Some more technical quirks about some of the benchmarks

Measuring a VM's boot time ⇒ How to do properly ?

- From `virsh start` to login: ? But it includes guest kernel boot time
- From git start to GRUB ? Nice but tricky to measure
- From `virsh start` to prompt of a special (small) direct kernel + initrd boot ? ⇐

Measuring QEMU's memory overhead

- What to sample? Which thread's RSS ?
- When to sample? Just once at the beginning ? Does it changes over time ?

Heterogenous configurations:

- Host OS != guest OS(es) ⇒ Not planned right now
- Different workload in each VM ⇒ Not planned right now

MMTests-CI @ QEMU Community

Feedback Wanted! Results aggregation in case of multiple VMs

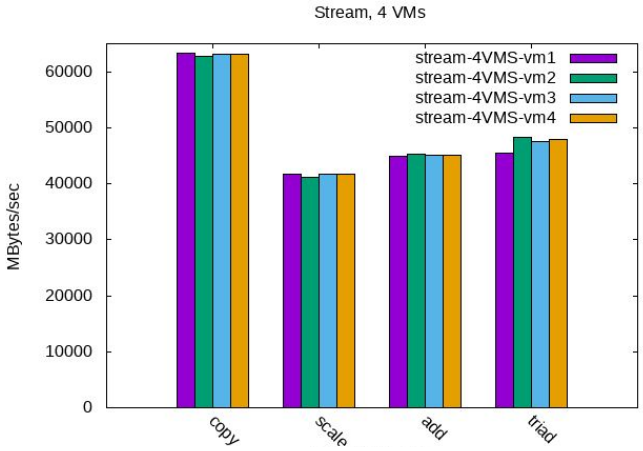
- Run STREAM in 4 VMs
  - Result is 4 results !
- Run STREAM in 16
  - Result is 16 results !

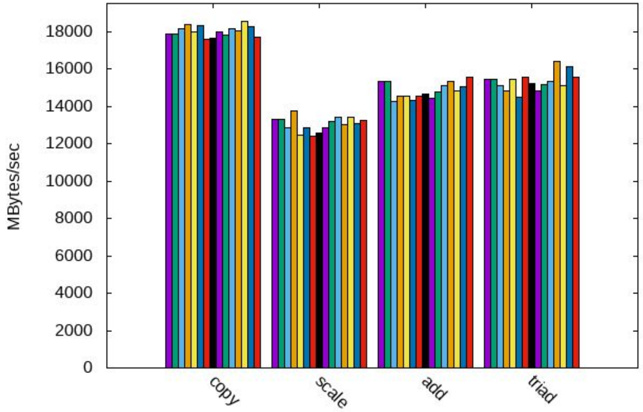
- What's the result of this STREAM run with 4 VMS ?
- How do we compare:
  "Run in 4 VMs today" VS *"Run in 4 VMs last week"* ?
  - We need a single number / set of stats
  - `Max`, `Min`, `Avg`, `Std`: of the 4 per-VM results

    ⇒ `Copy = AVG(copy[vm1], copy[vm2],`
    `copy[vm3], copy[vm4])`

    ⇒ `Std` should give an indication of the fairness, at host scheduling level

    ⇒ `Min` & Max could give indications about latency





SUSE

Thank You!

Any Questions?

For more information, contact SUSE at:

+1 800 796 3700 (U.S./Canada)

Maxfeldstrasse 5

90409 Nuremberg

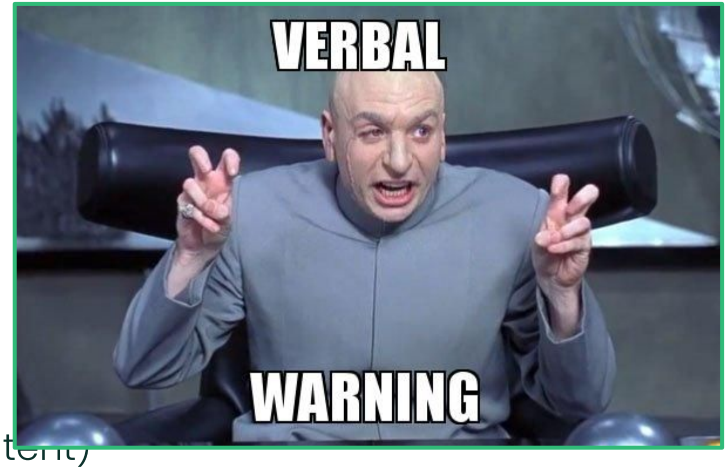
www.suse.com

Disclaimer

When using MMTests for day-to-day development

Beware that:

- must run as `root`
- It *changes* your system
  - Apply policies (e.g., cpufreq governor), install packages
  - Some are rolled back. *Some aren't!*
- It downloads the benchmarks from Internet
  - While running as root
    - Can this be trusted ?
    - (Workaround: setup a mirror and vet content)



Accepting an advice:

- Use it on "cattle" test machines, not on "pet" workstations