May 15 2020, *Would have been Pisa...*

MMTests for Benchmarking the Scheduler. And Power Management. And Virtualization. And...

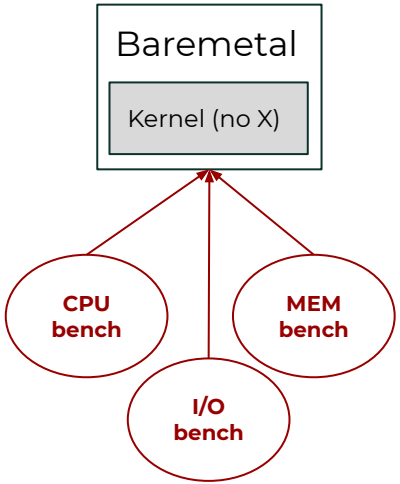
Dario Faggioli <dfaggioli@suse.com>
Software Engineer - Virtualization Specialist, SUSE
GPG: 4B9B 2C3A 3DD5 86BD 163E 738B 1642 7889 A5B8 73EE
https://about.me/dario.faggioli
https://twitter.com/DarioFaggioli (@DarioFaggioli)

SUSE

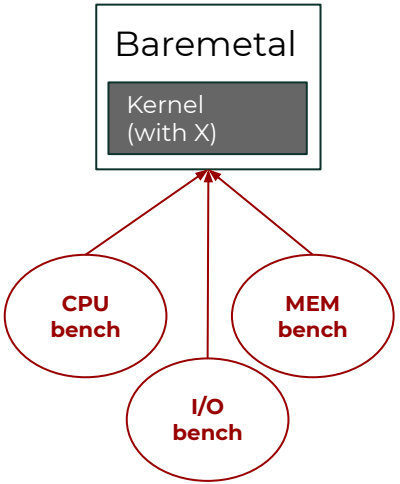
There’s Benchmarking & Benchmarking …

Benchmarking on Baremetal

What's the performance impact of kernel code change "X" ?



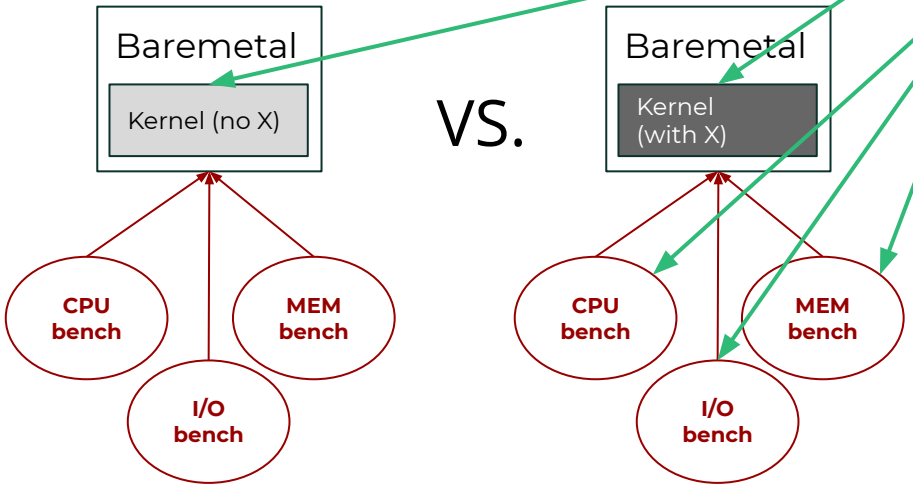
VS.



Benchmarking on Baremetal

What's the performance impact of kernel code change "X"?

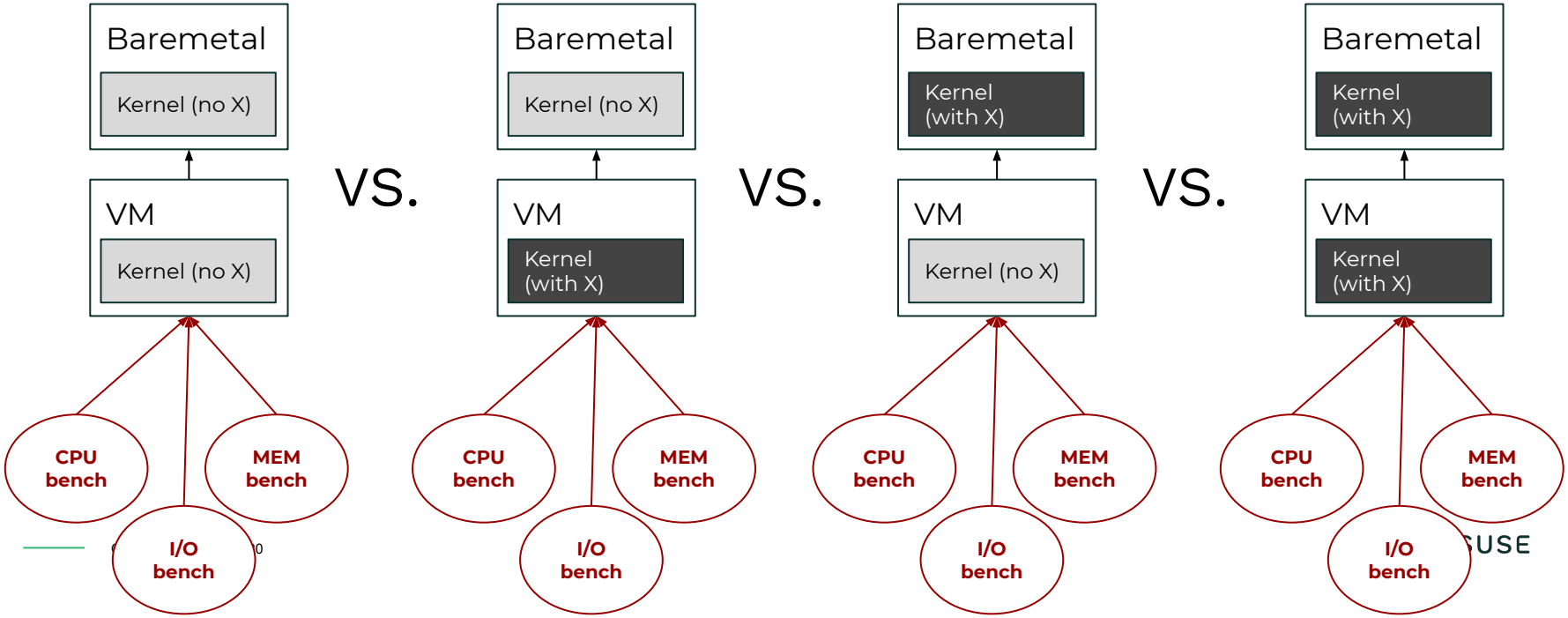
- Without X, with X
- Different benchmarks



SUSE

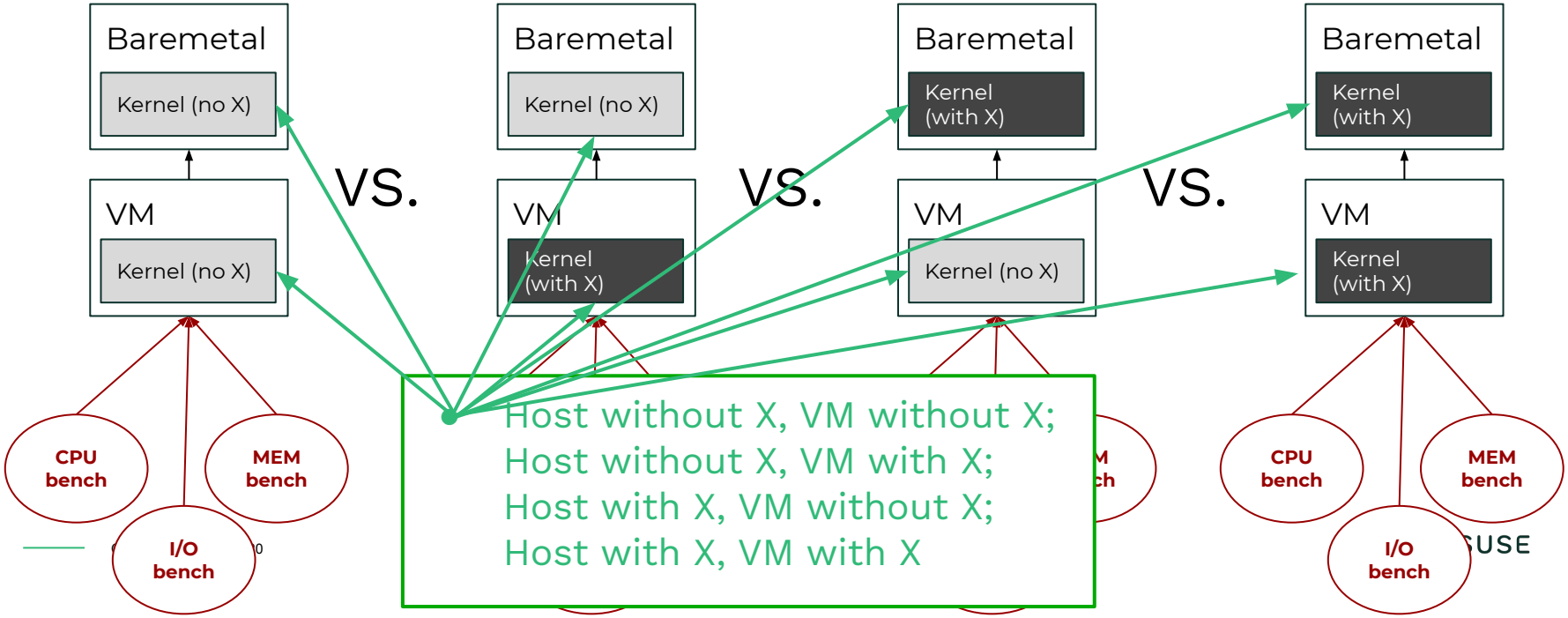
Benchmarking in Virtualization

What's the performance impact of kernel code change "X" ?



Benchmarking in Virtualization

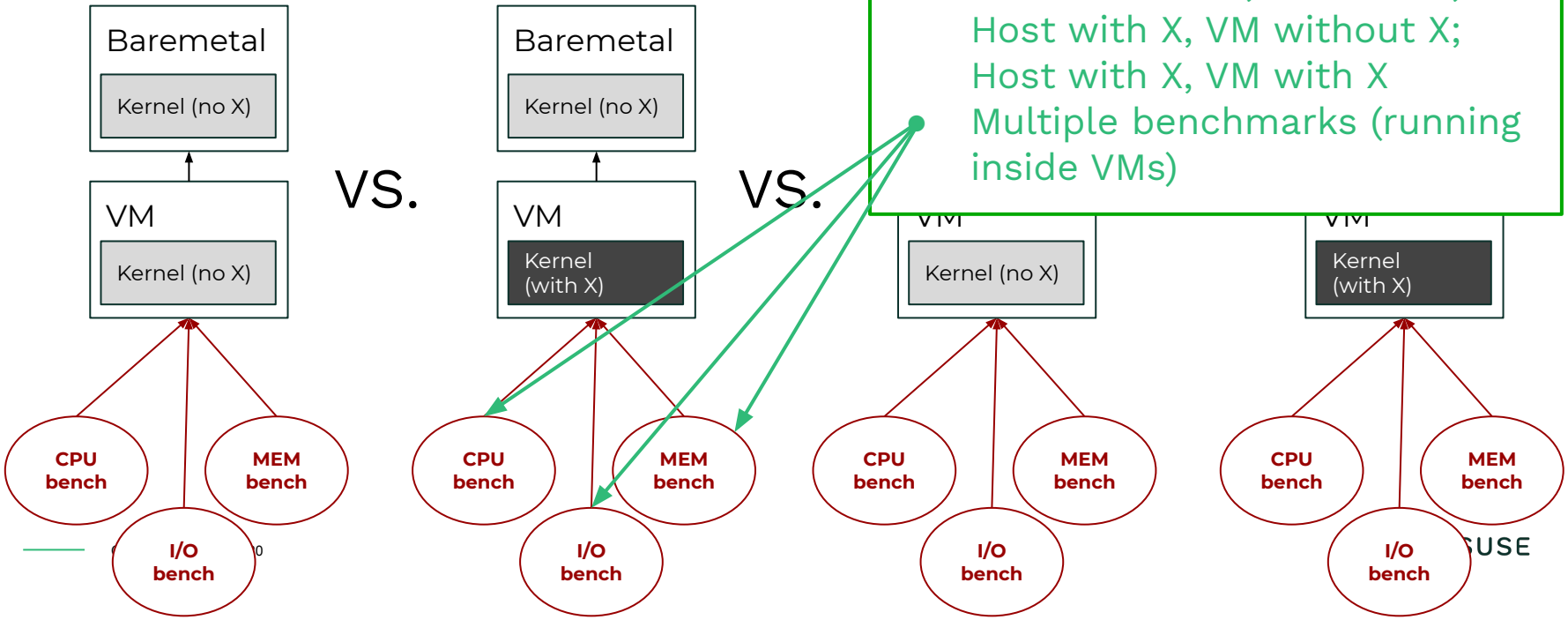
What's the performance impact of kernel code change "X" ?



Benchmarking in Virtualization

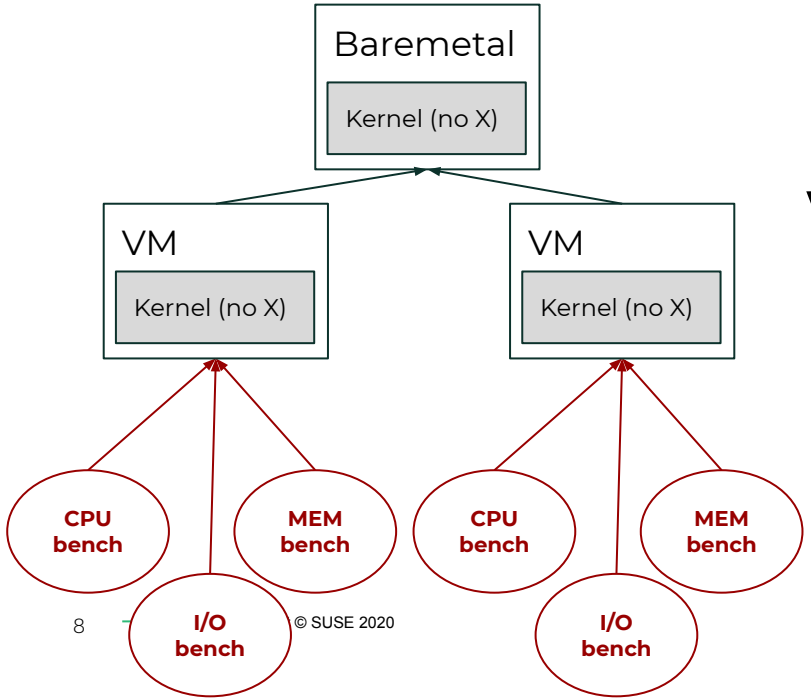
What's the performance impact of kernel code change

- Host without X, VM without X;
  Host without X, VM with X;
  Host with X, VM without X;
  Host with X, VM with X
- Multiple benchmarks (running inside VMs)

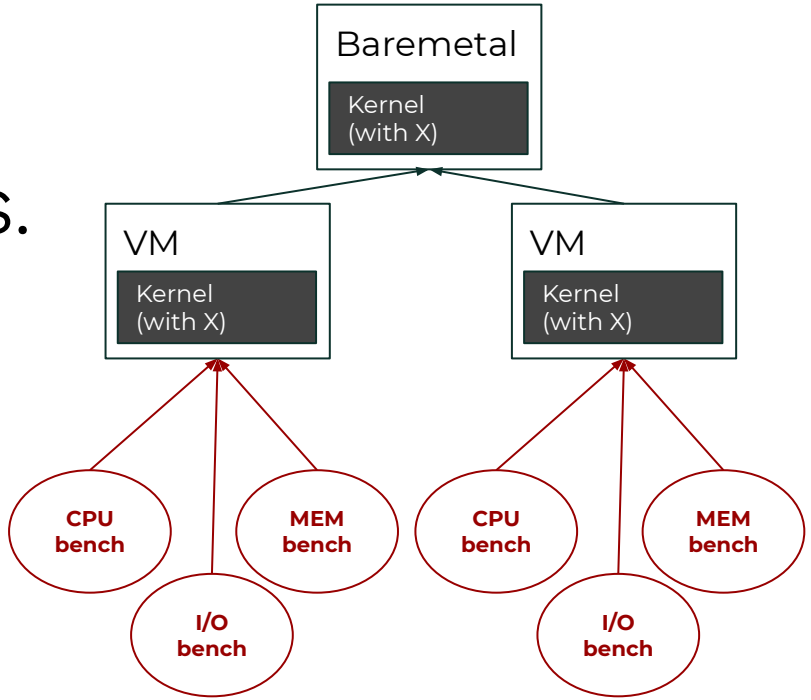


Benchmarking in Virtualization

What's the performance impact of kernel code change "X" ?



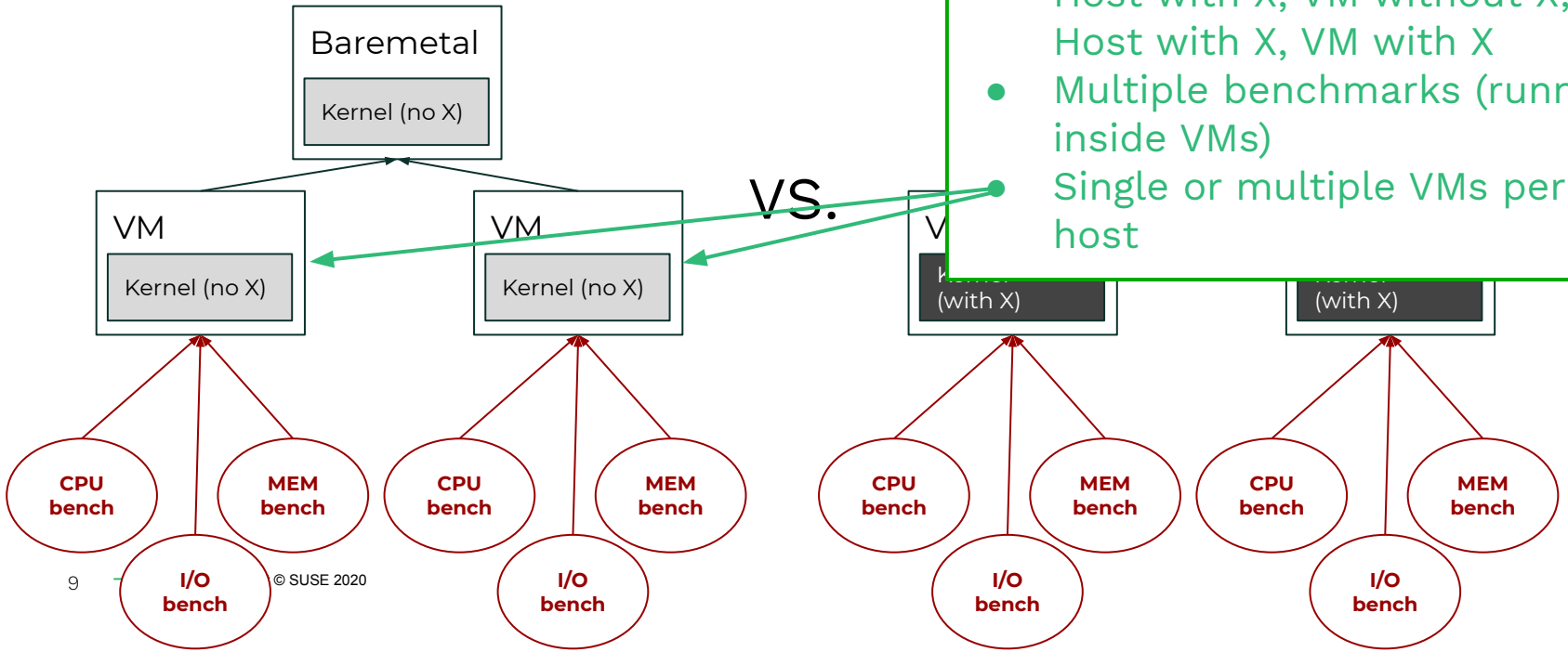
VS.



Benchmarking in Virtualization

What's the performance impact of kernel code cha

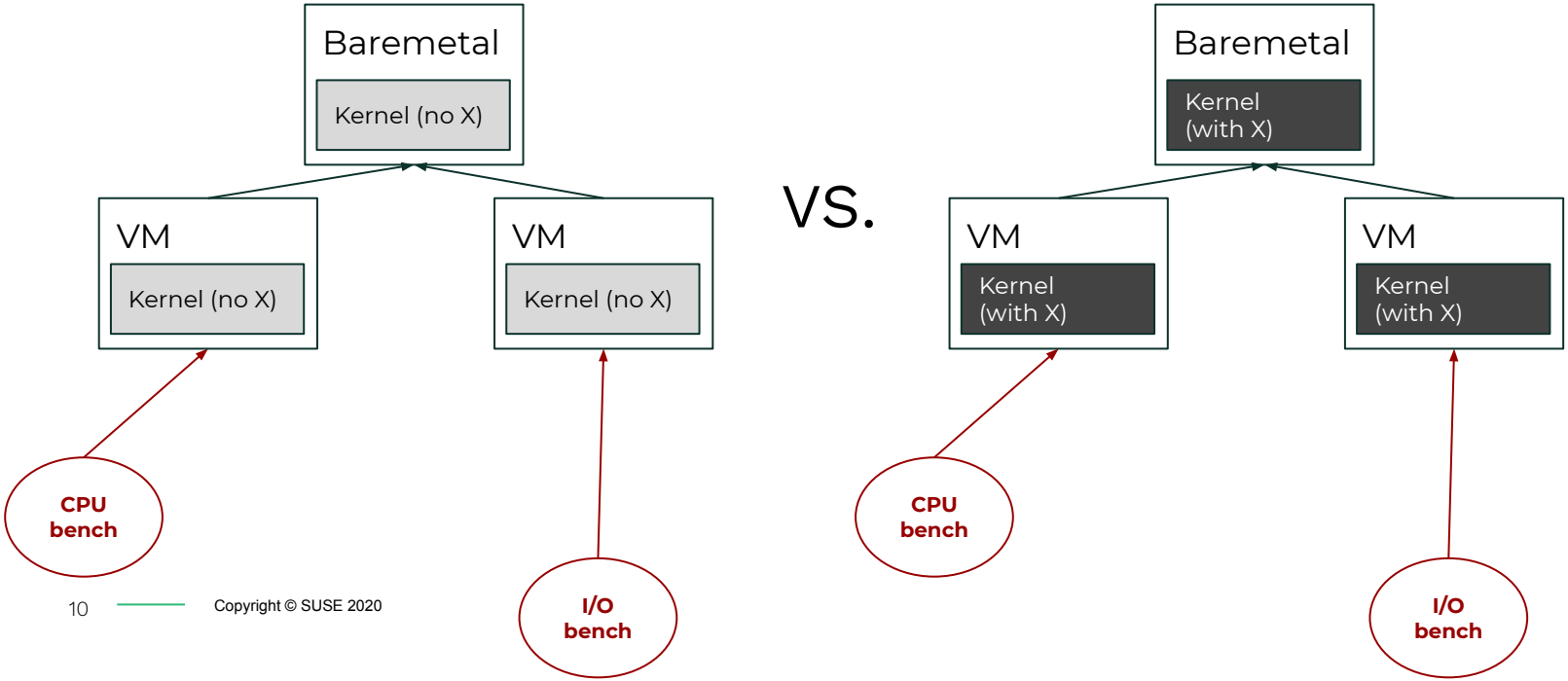
- Host without X, VM without X;
  Host without X, VM with X;
  Host with X, VM without X;
  Host with X, VM with X
- Multiple benchmarks (running inside VMs)
- Single or multiple VMs per host



SUSE

Benchmarking in Virtualization

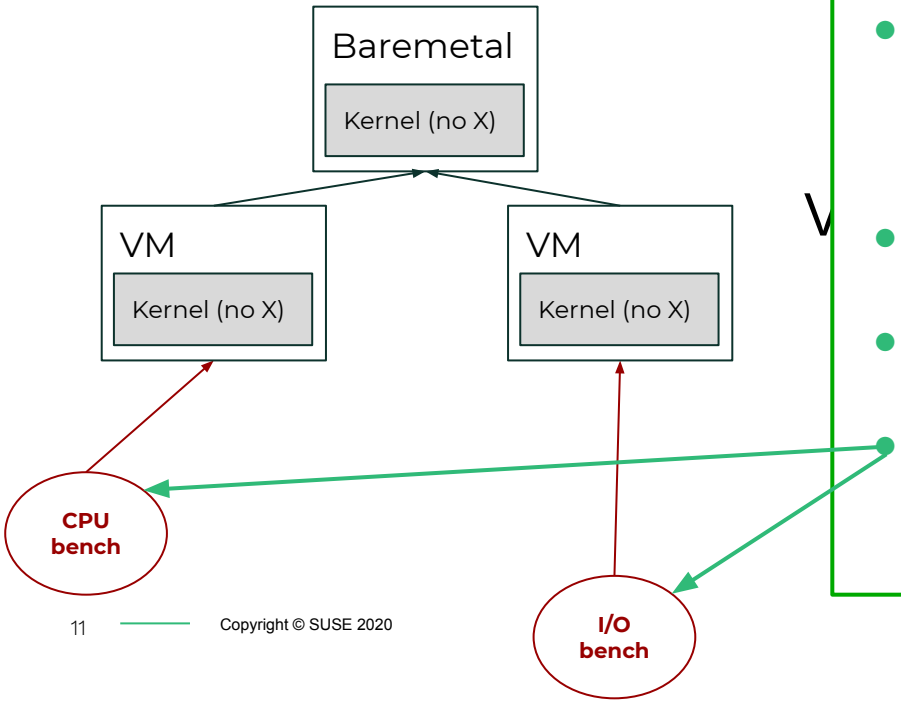
What's the performance impact of kernel code change "X" ?



SUSE

Benchmarking in Virtualization

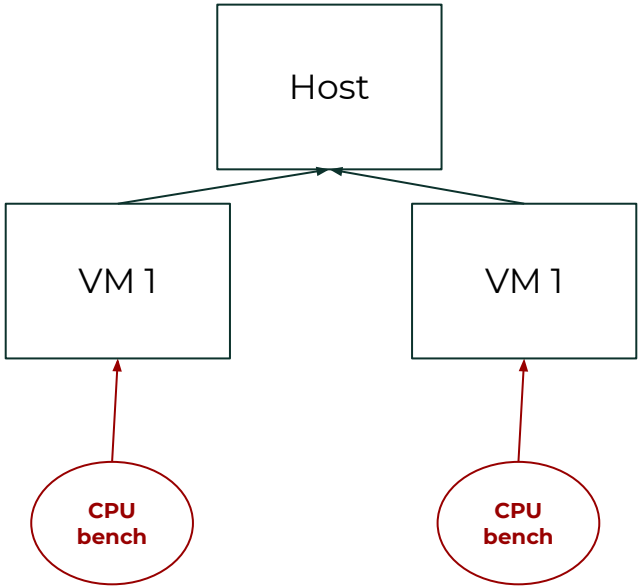
What's the performance impact of kernel code change "X" ?



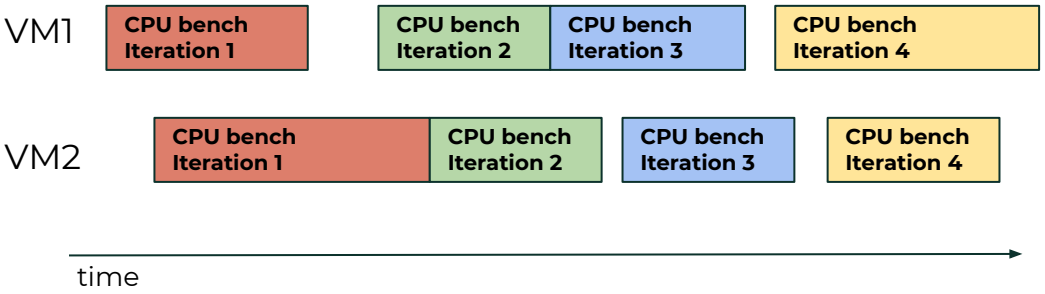
- Host without X, VM without X;
  Host without X, VM with X;
  Host with X, VM without X;
  Host with X, VM with X
- Multiple benchmarks (running inside VMs)
- Single or multiple VMs per host
- Homogeneous or heterogeneous workload inside VMs

Synchronized Runs & Iterations

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



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



SUSE

Synchronized Runs & Iterations

What's the p

Ms,

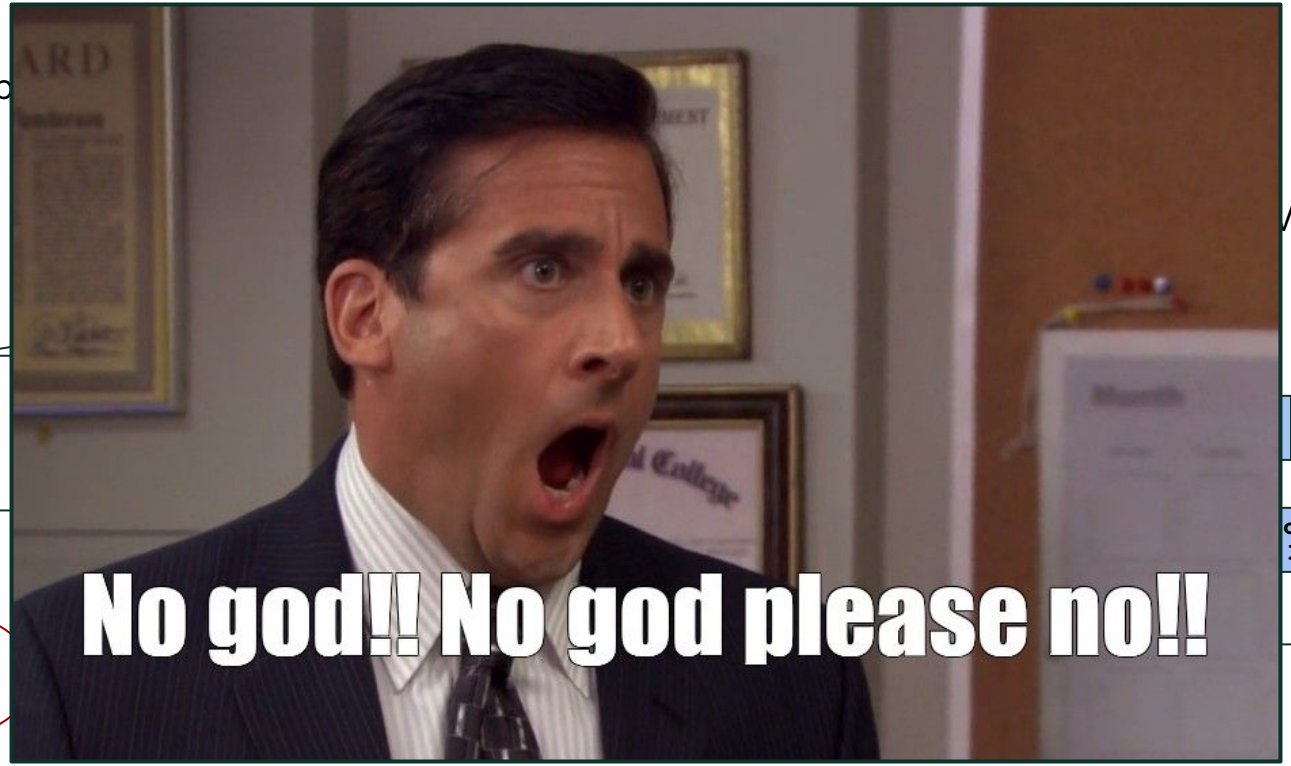
VM 1

CPU bench Iteration 4

ch 3

CPU bench Iteration 4

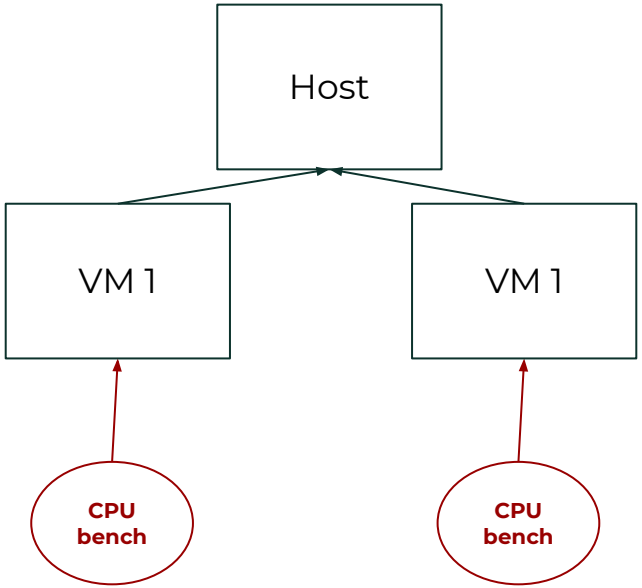
CPU bench



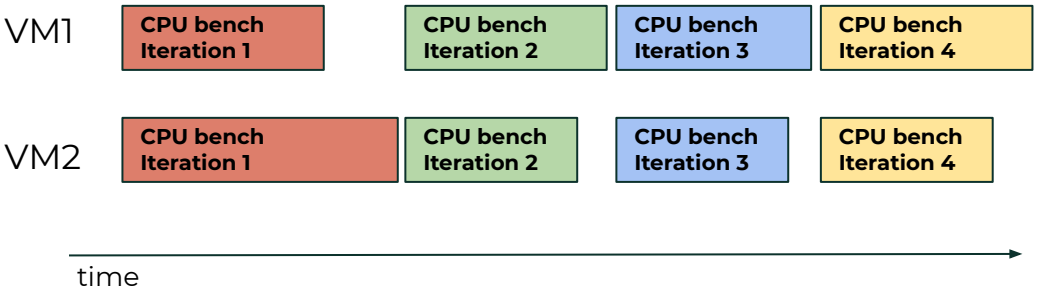
SUSE

Synchronized Runs & Iterations

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



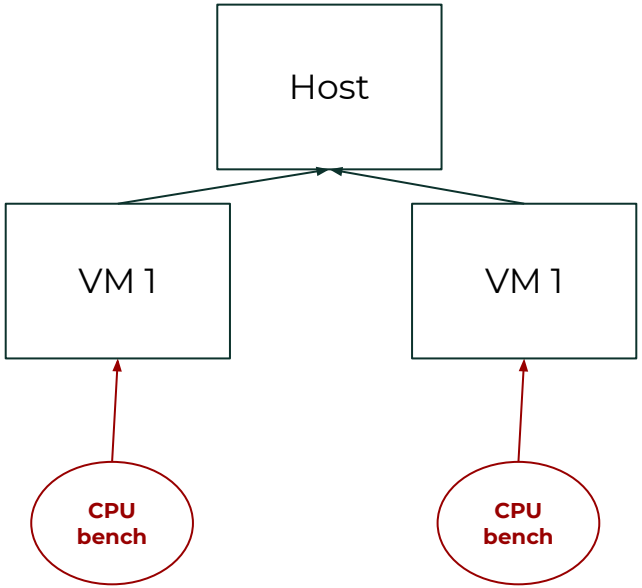
Synchronization of each (iteration of each) benchmark inside the various VMs:



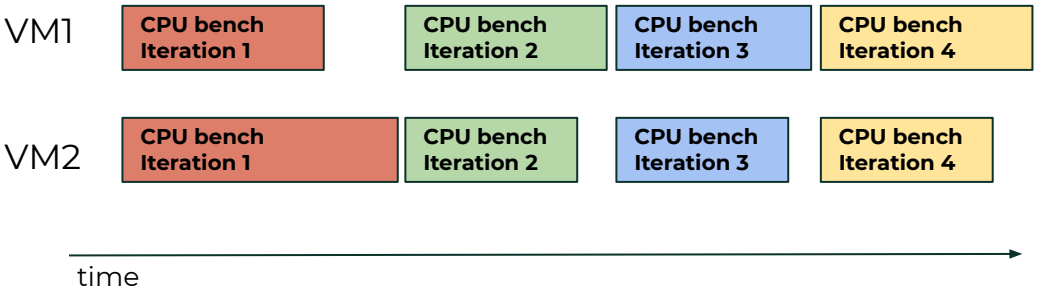
SUSE

Synchronized Runs & Iterations

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



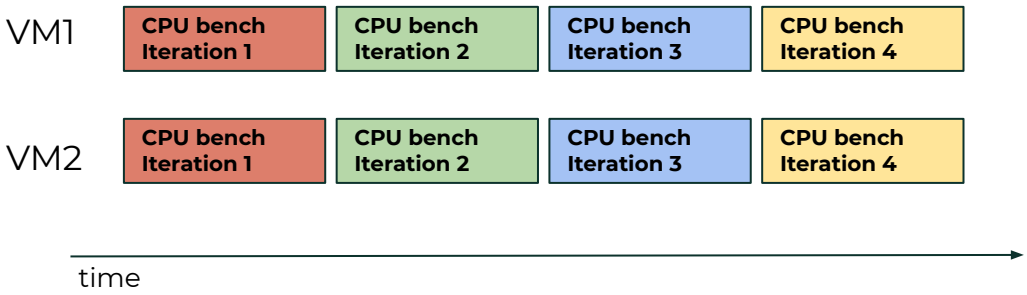
Synchronization of each (iteration of each) benchmark inside the various VMs:



Example: Hypervisor Scheduler Fairness

Scenario: 2 VMs on a shared host; identical; same priority/shares; CPU benchmark

Ideally:

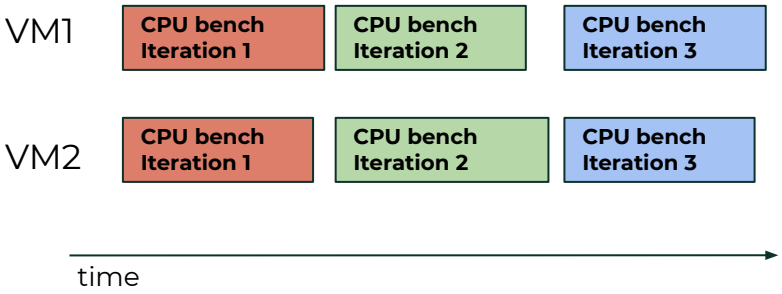


- For each iteration, VMs receive equal amount of CPU time
- Each iteration of the benchmarks in the VMs lasts exactly the same
- Perfect fairness!

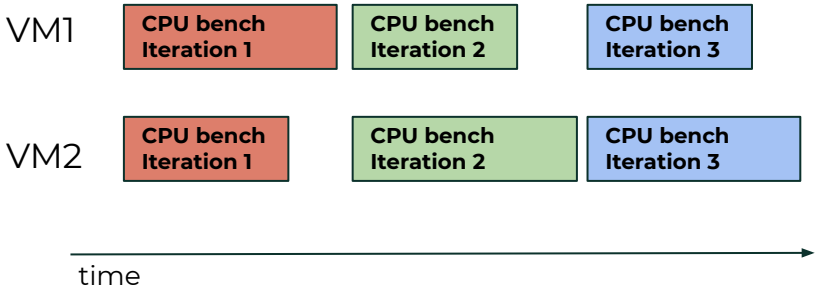
Example: Hypervisor Scheduler Fairness

Scenario: 2 VMs on a shared host; identical; same priority/shares; CPU benchmark

Reality:



Good fairness

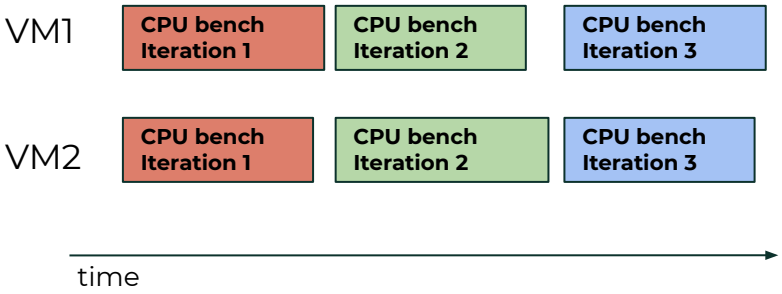


Bad fairness

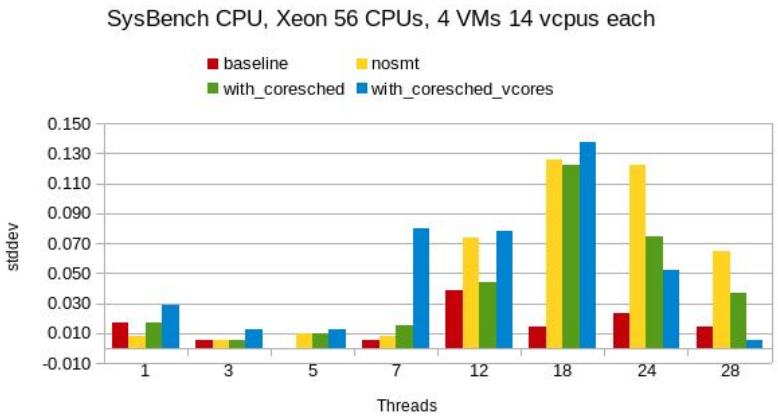
Example: Hypervisor Scheduler Fairness

Scenario: 2 VMs on a shared host; identical; same priority/shares; CPU benchmark

Reality:



Good fairness



Std-dev of benchmark runtime across 4 VMs. The lower the bar, the “fairer”

SUSE

Testing & Benchmarking Suites, CI Tools, ...

- OpenQA
- Jenkins
- Kernel CI
- Autotest / Avocado-framework / Avocado-vt
- Phoronix Test Suite
- Fuego
- Linux Test Project
- Xen-Project's OSSTests
- ...
- ...

Enters MMTests

Some History of MMtests

“MMTests is a configurable test suite that runs a number of common workloads of interest to MM developers.”

E.g., MMTests 0.05, in Sept. 2012 (on LKML)

Evolved a lot. Not MM-only any longer.

Now on https://github.com/gormanm/mmtests

- Emails to: Mel Gorman <mgorman@suse.com>
- To me is also fine: Dario Faggioli <dfaggioli@suse.com>
- GitHub issues / pull requests **not** preferred

MMTests

- Bash & Perl
- Fetches, builds, configures & runs a (set of) benchmark(s)
  - Config files: collection of bash exported variables
  - Benchmarks are run through wrappers (“shellpacks”)
  - Each bench 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, ...
- Can plot
- “Monitors”: while the benchmark is running, it can:
  - Sample top, mpstat, vmstat, iostat, ...
  - Collect data from: perf, ftrace, ...

```
Solace:/home/dario/Local/src/mmtests # ls work/log/NET_TEST_NOM/iter-0/
cgroup-tree.txt.gz            cstate-latencies-NET_TEST_NOM.txt  lscpu.txt.gz    netperf-tcp      tests-sysstate.gz
cpu-topology-mmtests.txt.gz   dmesg.gz                           lsscsi.txt.gz   numactl.txt.gz   tests-timestamp
cpu-vulnerabilities.txt       kconfig-5.6.11-1-default.txt.gz    lstopo.pdf.gz   tests-activity
cpupower.txt.gz               kernel.version                     lstopo.txt.gz   tests-sysstate
```

MMTests: Available Benchmarks

Among the others, already preconfigured:

- pgbench, sysbench-oltp (mariadb and postgres), pgioperf, ...
- bonnie, fio, filebench, iozone, tbench, dbench4, ...
- redis, memcached, john-the-ripper, ebizzy, nas-pb, ...
- hackbench, schbench, cyclictest, ...
- netperf, iperf, sockperf, ...
- Custom ones:
  - Linux kernel load balancer, program startup time, ...
- Workload like:
  - git workload, kernel dev. workload, ...
- Check in configs/ directory
  - More combination auto-generated (bin/generate-* scripts)

A Benchmark Config File

```
# 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"
```

A Benchmark Config File

```
# 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
export MONITORS_AL
export MONITORS_GZ
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.

SUSE

A Benchmark Config File

```
# 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
export MONITORS_AL
export MONITORS_GZ
export MONITORS_WI
export MONITOR_UPD
```

```
# 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

SUSE

A Benchmark Config File

```
# 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"
```

```
# 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 of each benchmark; kind of intuitive, but only if you know the benchmark already. To be sure, check the shellpack

SUSE

MMTests Workload

```
# ./run-mmtests.sh --config configs/config-netperf BASELINE
  [ change kernel / configuration / etc , e.g., disable KPTI ]
# ./run-mmtests.sh --config configs/config-netperf PTI-OFF

$ ./bin/compare-mmtests.pl --directory work/log --benchmark netperf-tcp \
    --names BASELINE,PTI-OFF
                                BASELINE                PTI-OFF
    Hmean      64          1205.33 (   0.00%)      2451.01 ( 103.35%)
    Hmean      128         2275.90 (   0.00%)      4406.26 (  93.61%)
    … … …
    Hmean      8192       36768.43 (   0.00%)     43695.93 (  18.84%)
    Hmean      16384      42795.57 (   0.00%)     48929.16 (  14.33%)
```

There's a `./compare-kernels.sh` script, but I personally prefer `compare-mmtests.pl`

SUSE

MMTests Workload

```
# ./run-mmtests.
  [ change kerne
# ./run-mmtests.

$ ./bin/compare-
  --names BASE

  Hmean      64
  Hmean      12
  … … …
  Hmean      81
  Hmean      16
```

```
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%)
```

There's a ./compare-kernels.sh sc

SUSE

MMTests: Recap Comparisons

```
$ ./bin/compare-mmtests.pl --directory work/log --benchmark netperf-tcp \
    --names BASELINE,PTI-OFF --print-ratio

                    BASELINE PTI-OFF
    Gmean Higher       1.00    0.28
```

- Useful as an overview
  - E.g., multiple runs of netperf, different packet sizes
  - … But how are things looking overall (i.e., taking account all the pkt. sizes) ?
- Ratios between baseline and compares + geometric mean of ratios
- Geometric mean, because it’s ratio friendly (nice explanation here)
- (First column, always 1.00… it’s the baseline)

MMTests: Recap Comparisons

```
$ ./bin/compare-mmtests.pl --directory work/log --benchmark netperf-tcp \
    --names BASELINE,PTI-OFF --print-ratio
```

```
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
```

- (First column, always 1.00… it’s the baseline)

MMTests: Monitors

```
$ ./bin.compare-mmtests.pl -d work/log -b stream -n SINGLE,OMP \
    --print-monitor duration

                        SINGLE         OMP
Duration User            45.04       50.75
Duration System           6.15       20.36
Duration Elapsed         51.16       20.26
```

Monitors:

- top, iotop, vmstat, mpstat, iostat, df, ...
- perf-event-stat, perf-time-stat, pert-top, ...
- monitors/

MMTests: Monitors

```
$ ./bi
    --

Durati
Durati
Durati

Monito
  ● to
  ● pe
  ● m
```

```
Solace:/home/dario/Local/src/mmtests # ls monitors/
latency-output                watch-kcache-slabs.pl         watch-proc-net-dev.sh
watch-app-launch.sh           watch-kcache.pl               watch-proc-pagetypeinfo.sh
watch-blktrace.sh             watch-kswapd-stack.sh         watch-proc-sched_debug.sh
watch-compaction.sh           watch-mmap-access-latency.sh  watch-proc-schedstat.sh
watch-cpuoffline-stress.sh    watch-mpstat.sh               watch-proc-stack.sh
watch-cpuoffline.sh           watch-numa-convergence.pl     watch-proc-stat.sh
watch-df.sh                   watch-numa-meminfo.sh         watch-proc-vmstat.sh
watch-dstate.pl               watch-numa-numastat.sh        watch-proc-zoneinfo.sh
watch-extfrag.sh              watch-numa-scheduling.pl      watch-read-latency.sh
watch-file-frag.sh            watch-numad.sh                watch-slabinfo.sh
watch-ftrace.sh               watch-pcpu-usage.sh           watch-storage-cache-status.sh
watch-function-frequency.pl   watch-perf-event-stat.sh      watch-stress-highorder-atomic.pl
watch-highorder-latency.pl    watch-perf-event.sh           watch-sync-latency.sh
watch-highorder.pl            watch-perf-sched.sh           watch-syscalls.pl
watch-inbox-open.sh           watch-perf-time-stat.sh       watch-tlb.pl
watch-interactive-apps.sh     watch-perf-top.sh             watch-top.sh
watch-iostat.sh               watch-proc-buddyinfo.sh       watch-turbostat.sh
watch-iotop.sh                watch-proc-interrupts.sh      watch-vmstat.sh
watch-irqsoff.pl              watch-proc-latency_stats.sh   watch-write-latency.sh
watch-kcache-detailed.pl      watch-proc-meminfo.sh
```

MMTests: Monitors

```
$ egrep "MONITORS|EVENTS" configs/config-workload-stockfish
export MONITORS_GZIP="proc-vmstat mpstat perf-time-stat ftrace"
export MONITORS_WITH_LATENCY="vmstat"
export MONITOR_PERF_EVENTS="cpu-migrations context-switches"

$ ./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
```

Monitors: ftrace

```
     netserver-30917 [002] ..s1 66388.419755: netif_rx: dev=lo skbaddr=000000006386d2b5 len=12468
     netserver-30917 [002] .... 66388.419793: netif_rx: dev=lo skbaddr=0000000031448e8f len=52
     netserver-30917 [002] ..s1 66388.419800: netif_rx: dev=lo skbaddr=00000000e7f1eba4 len=13364
  ksoftirqd/2-24    [002] d.s. 66388.419813: sched_migrate_task: comm=rcu_sched pid=10 prio=120 orig_cpu=2 dest_cpu=6
        <idle>-0     [004] d.s. 66388.419814: sched_migrate_task: comm=netserver pid=30917 prio=120 orig_cpu=2 dest_cpu=6
           cat-30856 [003] d... 66388.419828: sched_migrate_task: comm=kworker/u64:4 pid=29846 prio=120 orig_cpu=6 dest_cpu=4
kworker/u64:4-29846 [004] d... 66388.419833: sched_migrate_task: comm=tclsh pid=30861 prio=120 orig_cpu=4 dest_cpu=1
     netserver-30917 [006] .... 66388.419859: netif_rx: dev=lo skbaddr=00000000ae72432a len=52
     netserver-30917 [006] ..s1 66388.419869: netif_rx: dev=lo skbaddr=00000000d27f134b len=17204
         tclsh-30785 [007] d... 66388.419875: sched_migrate_task: comm=tclsh pid=30861 prio=120 orig_cpu=1 dest_cpu=4
     netserver-30917 [006] .... 66388.419915: netif_rx: dev=lo skbaddr=00000000ae72432a len=52
       netperf-30916 [000] .... 66388.419926: netif_rx: dev=lo skbaddr=000000002402037d len=14132
kworker/u64:4-29846 [004] d... 66388.419934: sched_migrate_task: comm=tclsh pid=30861 prio=120 orig_cpu=4 dest_cpu=1
     netserver-30917 [006] .... 66388.419969: netif_rx: dev=lo skbaddr=00000000fa6506a8 len=52
         tclsh-30785 [007] d... 66388.419969: sched_migrate_task: comm=tclsh pid=30861 prio=120 orig_cpu=1 dest_cpu=4
     netserver-30917 [006] ..s1 66388.419977: netif_rx: dev=lo skbaddr=00000000d27f134b len=12852
kworker/u64:4-29846 [004] d... 66388.419994: sched_migrate_task: comm=tclsh pid=30861 prio=120 orig_cpu=4 dest_cpu=1
     netserver-30917 [006] .... 66388.420012: netif_rx: dev=lo skbaddr=00000000fa6506a8 len=52
       netperf-30916 [000] .... 66388.420022: netif_rx: dev=lo skbaddr=0000000024a7a8ee len=12084
         tclsh-30785 [007] d... 66388.420027: sched_migrate_task: comm=tclsh pid=30861 prio=120 orig_cpu=1 dest_cpu=4
```

Monitors: vmstat, mpstat

```
1589506775.8309   9.9399     9.9399 --  2  0  25856 2424716     12 852788    0    0     7    19 1335 6448  5 20 75  0  0
1589506785.8311  19.9401    10.0002 --  2  0  25856 2424444     12 852828    0    0     0     0 1280 8756  3 24 73  0  0
1589506795.8314  29.9404    10.0003 --  2  0  25856 2424972     12 852828    0    0     0     0 1245 9435  3 24 73  0  0
1589506805.8316  39.9406    10.0003 --  2  0  25856 2425212     12 852848    0    0     0   686 1281 9043  3 24 73  0  0
1589506815.8319  49.9409    10.0003 --  2  0  25856 2425964     12 852880    0    0     0     0 1287 8461  3 24 73  0  0
1589506825.8322  59.9412    10.0003 --  2  0  25856 2425476     12 852932    0    0     0     0 1329 12019  3 24 73  0  0
1589506835.8325  69.9415    10.0003 --  2  0  25856 2425908     12 853080    0    0     0   772 1456 31842  3 23 74  0  0
1589506845.8326  79.9416    10.0002 --  2  0  25856 2426248     12 853116    0    0     0     0 1518 31757  2 23 74  0  0
1589506855.8330  89.9420    10.0004 --  2  0  25856 2416588     12 853120    0    0     0     0 1406 23022  3 24 73  0  0
1589506865.8333  99.9423    10.0003 --  2  0  25856 2425996     12 853128    0    0     0   616 1561 29851  3 24 74  0  0
1589506875.8336 109.9425    10.0003 --  2  0  25856 2424528     12 853128    0    0     0     0 1501 38455  3 24 74  0  0
1589506885.8339 119.9429    10.0003 --  2  0  25856 2415364     12 853176    0    0     0     0 1518 37279  3 24 74  0  0
1589506895.8342 129.9432    10.0003 --  2  0  25856 2424268     12 853180    0    0     0   742 1360 38296  2 23 74  0  0
1589506905.8345 139.9435    10.0003 --  2  0  25856 2414524     12 853180    0    0     0     0 1393 36677  2 23 75  0  0
1589506915.8347 149.9436    10.0002 --  4  0  25856 2414576     12 853184    0    0     0     0 1393 37022  2 23 75  0  0
```

```
1589506935.8961 170.0053     9.9986 --
1589506935.8962 170.0053     0.0001 -- time: 1589506935
1589506935.8963 170.0055     0.0001 -- 03:42:05     CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
1589506935.8964 170.0056     0.0001 -- 03:42:15     all    1.62    0.00   19.68    0.01    0.00    4.73    0.00    0.00    0.00   73.96
1589506935.8965 170.0057     0.0001 -- 03:42:15       0    2.40    0.00   85.40    0.00    0.00   12.20    0.00    0.00    0.00    0.00
1589506935.8967 170.0058     0.0001 -- 03:42:15       1    4.94    0.00    4.85    0.00    0.00   11.05    0.00    0.00    0.00   79.16
1589506935.8968 170.0059     0.0001 -- 03:42:15       2    2.36    0.00   51.33    0.00    0.00    9.73    0.00    0.00    0.00   36.58
1589506935.8969 170.0061     0.0001 -- 03:42:15       3    0.00    0.00    0.10    0.00    0.00    1.29    0.00    0.00    0.00   98.61
1589506935.8970 170.0062     0.0001 -- 03:42:15       4    0.50    0.00    0.30    0.00    0.00    0.20    0.00    0.00    0.00   99.00
1589506935.8972 170.0063     0.0002 -- 03:42:15       5    0.90    0.00    1.00    0.00    0.00    0.10    0.00    0.00    0.00   98.00
1589506935.8973 170.0065     0.0001 -- 03:42:15       6    0.91    0.00   16.80    0.10    0.00    2.63    0.00    0.00    0.00   79.55
1589506935.8974 170.0066     0.0001 -- 03:42:15       7    0.60    0.00    0.20    0.00    0.00    0.00    0.00    0.00    0.00   99.20
1589506945.8964 180.0055     9.9990 --
1589506945.8964 180.0056     0.0001 -- time: 1589506945
1589506945.8965 180.0057     0.0001 -- 03:42:15     CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
1589506945.8966 180.0057     0.0001 -- 03:42:25     all    1.74    0.00   19.68    0.02    0.00    4.14    0.00    0.00    0.00   74.41
1589506945.8966 180.0058     0.0001 -- 03:42:25       0    1.06    0.00   26.54    0.00    0.00    9.62    0.00    0.00    0.00   62.79
1589506945.8967 180.0059     0.0001 -- 03:42:25       1    5.78    0.00    1.03    0.00    0.00    7.00    0.00    0.00    0.00   86.19
1589506945.8967 180.0059     0.0001 -- 03:42:25       2    0.71    0.00   21.07    0.00    0.00    3.04    0.00    0.00    0.00   75.18
1589506945.8968 180.0060     0.0001 -- 03:42:25       3    3.05    0.00   68.76    0.00    0.00    8.45    0.00    0.00    0.00   19.74
1589506945.8969 180.0061     0.0001 -- 03:42:25       4    0.30    0.00    0.40    0.00    0.00    0.10    0.00    0.00    0.00   99.20
```


SUSE

Monitors: turbostat

```
1589506767.2661   1.3751     0.0002 -- cpu0: MSR_IA32_TEMPERATURE_TARGET: 0x00641400 (100 C)
1589506777.2786  11.3876    10.0125 -- Core    CPU     Avg_MHz Busy%   Bzy_MHz TSC_MHz IRQ     SMI     POLL    C1      C1E     C3      C6      POLL%   C1%     C1E%    C3%     C6%     CPU%c1  CPU%c3  CPU%c6  CoreTmp Pkg%pc3 Pkg%pc6
1589506777.2787  11.3877     0.0001 -- -       -       727     24.81   2931    2800    13478   0       19392   11211   1042    3290    14673   0.10    0.53    0.57    5.28    68.93   26.35   3.19    45.65   53      0.12    4.52
1589506777.2789  11.3878     0.0001 -- 0       0       73      2.53    2867    2800    5133    0       23      518     188     882     3608    0.00    0.73    0.81    8.92    87.20   6.09    8.36    83.02   46      0.12    4.52
1589506777.2790  11.3879     0.0001 -- 0       4       39      1.36    2872    2800    887     0       84      309     119     677     3800    0.00    1.51    0.86    7.32    89.25   7.26
1589506777.2791  11.3880     0.0001 -- 1       1       216     7.36    2930    2800    676     0       1006    397     12      410     2785    0.04    0.12    0.06    4.05    88.53   3.00    4.16    85.48   49
1589506777.2792  11.3881     0.0001 -- 1       5       23      0.78    2873    2800    800     0       1       326     268     387     2819    0.00    0.53    1.45    7.10    90.27   9.58
1589506777.2793  11.3882     0.0001 -- 2       2       2541    86.62   2933    2800    2459    0       18265   9044    119     389     494     0.75    1.04    0.29    5.19    6.95    6.61    0.06    6.71    53
1589506777.2794  11.3883     0.0001 -- 2       6       204     6.97    2925    2800    579     0       12      283     141     41      584     0.00    0.10    0.06    1.99    90.93   86.27
1589506777.2795  11.3884     0.0001 -- 3       3       2706    92.27   2933    2800    2379    0       0       0       0       11      102     0.00    0.00    0.00    0.11    7.63    0.15    0.17    7.41    50
1589506777.2796  11.3885     0.0001 -- 3       7       16      0.56    2830    2800    565     0       1       334     195     493     481     0.00    0.24    1.01    7.58    90.66   91.86
1589506787.3018  21.4108    10.0222 -- Core    CPU     Avg_MHz Busy%   Bzy_MHz TSC_MHz IRQ     SMI     POLL    C1      C1E     C3      C6      POLL%   C1%     C1E%    C3%     C6%     CPU%c1  CPU%c3  CPU%c6  CoreTmp Pkg%pc3 Pkg%pc6
1589506787.3019  21.4109     0.0001 -- -       -       749     25.54   2933    2798    12264   0       27900   12558   393     9388    0       0.14    0.30    0.13    74.07   0.00    26.23   48.23   0.00    52      0.00    0.00
1589506787.3021  21.4110     0.0001 -- 0       0       49      1.67    2933    2800    4525    0       1       18      15      3878    0       0.00    0.23    0.09    98.34   0.00    22.49   75.84   0.00    52      0.00    0.00
1589506787.3022  21.4111     0.0001 -- 0       4       614     20.93   2933    2800    714     0       6904    3173    48      1129    0       0.29    0.62    0.11    78.40   0.00    3.23
1589506787.3023  21.4112     0.0001 -- 1       1       15      0.52    2933    2800    418     0       35      629     14      865     0       0.00    0.51    0.03    99.04   0.00    70.11   29.37   0.00    43
1589506787.3024  21.4113     0.0001 -- 1       5       2027    69.09   2933    2800    2384    0       18481   7854    240     872     0       0.75    0.67    0.31    30.01   0.00    1.54
1589506787.3025  21.4114     0.0001 -- 2       2       31      1.04    2933    2800    425     0       0       23      33      787     0       0.00    0.22    0.33    98.55   0.00    11.15   87.81   0.00    46
1589506787.3026  21.4115     0.0001 -- 2       6       313     10.67   2933    2800    867     0       2478    826     13      1317    0       0.11    0.12    0.00    89.24   0.00    1.62
1589506787.3026  21.4116     0.0001 -- 3       3       2933    99.99   2933    2800    2542    0       0       0       0       4       0       0.00    0.00    0.00    0.01    0.00    0.00    0.00    0.00    50
1589506787.3027  21.4117     0.0001 -- 3       7       10      0.33    2933    2800    389     0       1       35      30      536     0       0.00    0.02    0.21    99.46   0.00    99.67
1589506797.3108  31.4197    10.0080 -- Core    CPU     Avg_MHz Busy%   Bzy_MHz TSC_MHz IRQ     SMI     POLL    C1      C1E     C3      C6      POLL%   C1%     C1E%    C3%     C6%     CPU%c1  CPU%c3  CPU%c6  CoreTmp Pkg%pc3 Pkg%pc6
1589506797.3109  31.4198     0.0001 -- -       -       774     26.36   2933    2802    12375   0       27779   13282   1302    10305   0       0.14    0.64    0.49    72.66   0.00    27.17   46.47   0.00    54      0.00    0.00
```

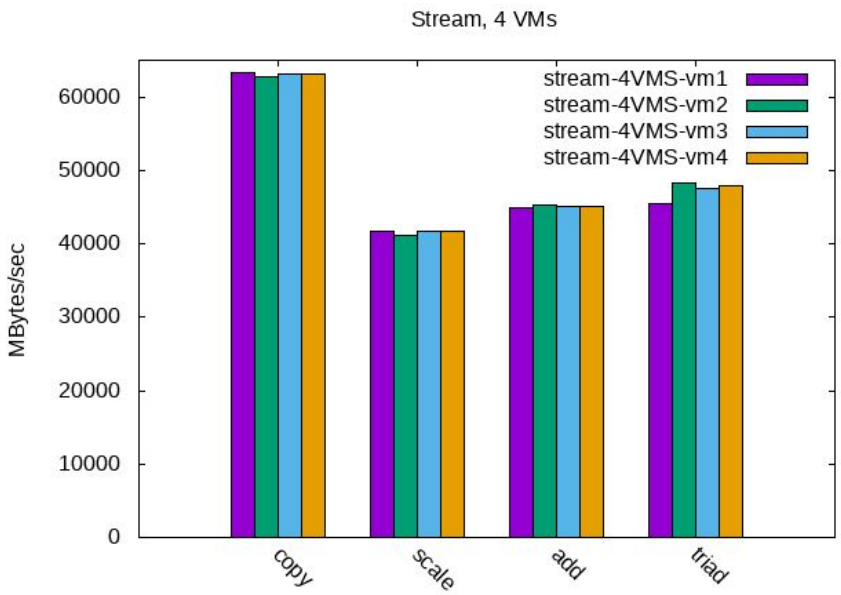

SUSE

Monitors: perf

```
time: 1589511639

 Performance counter stats for 'system wide':

              124      cpu-migrations
           351220      context-switches
      49747783485      cycles                                          (42.93%)
          2279710      LLC-load-misses                                 (47.80%)
          1867998      LLC-store-misses                                (9.46%)

     10.001426558 seconds time elapsed

time: 1589511649

 Performance counter stats for 'system wide':

              117      cpu-migrations
           248001      context-switches
      51933896254      cycles                                          (46.05%)
         36618856      LLC-load-misses                                 (46.26%)
          1671520      LLC-store-misses                                (7.69%)

     10.001763859 seconds time elapsed
```

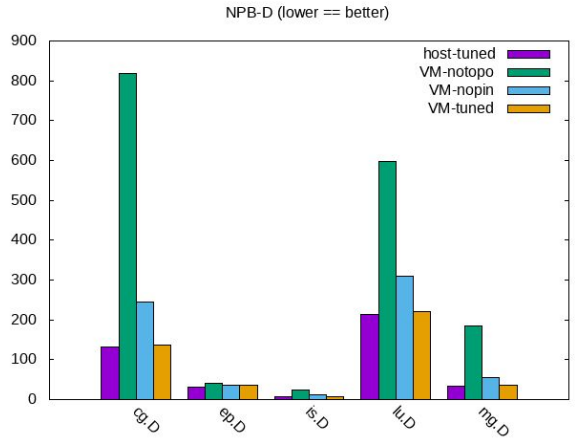
MMTests: Plots

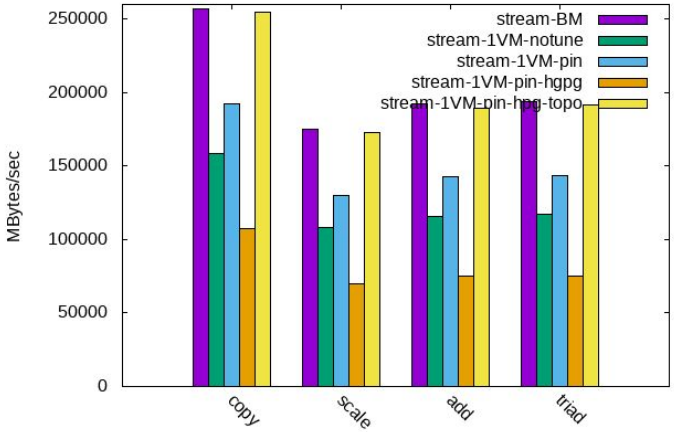
```
$ graph-mmtests.sh -d . -b stream -n stream-4VMS-vm1,stream-4VMS-vm2, \
    stream-4VMS-vm3,stream-4VMS-vm4 --format png --yrange 0:65000     \
    --title "Stream, 4 VMs"
```

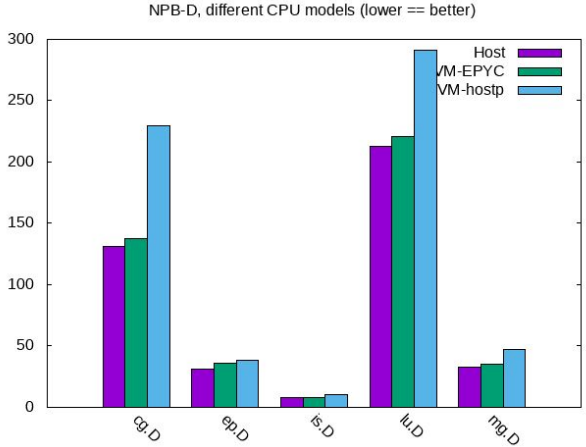


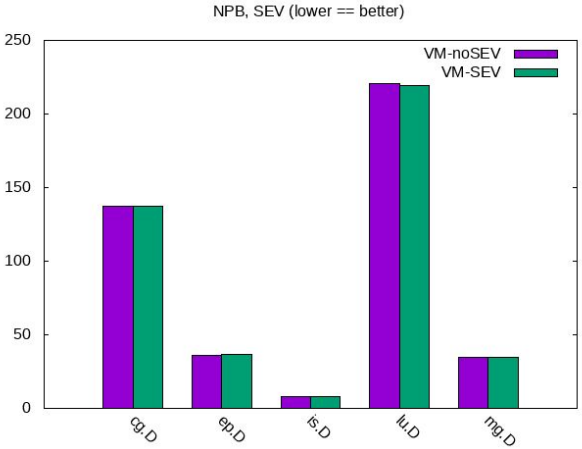
SUSE

MMTests: Plots









SUSE

Couple of “Beware of …”

- (Kind of) requires `root`
  - May need to change system properties (e.g., CPU frequency governor)
  - Tries to undo all it has done
  - Still, better used on “cattle” test machines than on “pet” workstations

- It downloads the benchmarks from Internet
  - Slow ? Can be trusted ?
  - Easy enough to configure a mirror (how it’s used internally)

MMTests for Virtualization

Doesn't have to be KVM, can be anything that Libvirt can manage

MMTests & Virtualization

```
# ./run-kvm.sh -k -L --vm VM1 --config netperf-vm BASELINE
# ./run-kvm.sh -k -L --vm VM1 --config netperf-vm PTI-ON

$ ./bin/compare-mmtests.pl -d work/log -b netperf-tcp \
    --names BASELINE-VM1,PTI-ON-VM1

$ ls work/log/BASELINE-host
...

$ ls work/log/PTI-host
...
```

MMTests & Virtualization

```
# ./run-kvm.sh -k -L --vm VM1 --config netperf-vm BASELINE
# ./run-kvm.sh -k -L --vm VM1 --config netperf-vm PTI-ON

$ ./bin/compare-mmtests.pl -d work/log -b netperf-tcp \
    --names BASELINE-VM1,PTI-ON-VM1

$ ls work/log/BASELINE-host
...

$ ls work/log/PTI-host
...
```

- Results, logs and monitors of the VM runs go here
- Host logs and monitors go here

Running MMTests in a VM

The `run-kvm.sh` script takes care of:

- Defining (`virsh define`), if necessary and starting (`virsh start`) the VM
  - VM has to exist or an XML file must be provided
  - VM provisioning, installing, etc: work in progress
- Establishing SSH keys for passwordless connections to the VM
  - The host and guest must be able to talk via network
- Copying the whole MMTests directory inside the VM
- Run the benchmark in the VM with `run-mmtests.sh`
- Store the host logs and info
- Fetch the logs and the results from the VM and store them as well

Host Config File

```
# 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
```

Host Config File

```
# 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
```

- IP of the host that VMs can reach (needed for syncrhonization)
- The list of VM to use can be specified here as well (instead than on the command line)
- Tuned and numad (if these lines are present) as they will run on the host
- Monitors that will run on the host

MMTests & Multiple VMs

```
# ./run-kvm.sh -k -L --vm VM1,VM2 --config netperf-vms BASELINE
# ./run-kvm.sh -k -L --vm VM1,VM2 --config netperf-vms PTI-ON

$ ./bin/compare-mmtests.pl --directory work/log --benchmark netperf-tcp \
 --names BASELINE-VM1,BASELINE-VM2,PTI-ON-VM1,PTI-ON-VM1

$ ls work/log/BASELINE-host
...

$ ls work/log/PTI-host
...
```

MMTests & Multiple VMs

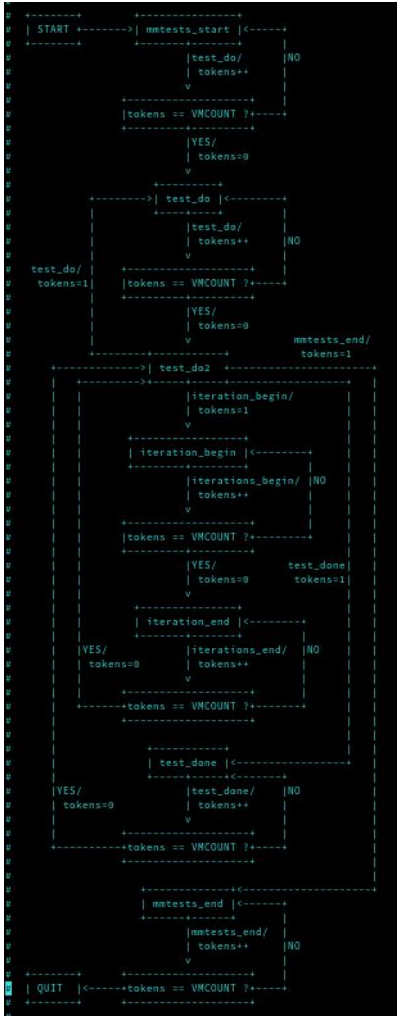
```
Solace:/home/dario/Local/src/mmtests # ./run-kvm.sh -k -L -C host_config --vm leap15.1-1,leap15.1-2 -c configs/config-scheduler-sysbench-cpu TEST_2VMS
Solace:/home/dario/Local/src/mmtests #
Solace:/home/dario/Local/src/mmtests #
Solace:/home/dario/Local/src/mmtests # ls work/log/
TEST_2VMS-host  TEST_2VMS-leap15.1-1  TEST_2VMS-leap15.1-2
Solace:/home/dario/Local/src/mmtests #
Solace:/home/dario/Local/src/mmtests #
Solace:/home/dario/Local/src/mmtests # ls work/log/TEST_2VMS-host/
cgroup-tree.txt.gz              dmesg.gz                        lscpu.txt.gz    mpstat-host.start       tests-sysstate.gz  vmstat-host.start
cpu-topology-mmtests.txt.gz     kconfig-5.6.11-1-default.txt.gz lsscsi.txt.gz   numactl.txt.gz          tests-timestamp
cpu-vulnerabilities.txt         kernel.version                  lstopo.pdf.gz   proc-vmstat-host.gz     tuned-log
cpupower.txt.gz                 leap15.1-1.xml                  lstopo.txt.gz   proc-vmstat-host.start  tuned-stdout
cstate-latencies-TEST_2VMS.txt  leap15.1-2.xml                  mpstat-host.gz  tests-activity          vmstat-host.gz
Solace:/home/dario/Local/src/mmtests #
Solace:/home/dario/Local/src/mmtests #
Solace:/home/dario/Local/src/mmtests # ls work/log/TEST_2VMS-leap15.1-1/iter-0/
cpu-topology-mmtests.txt.gz                 lscpu.txt.gz              perf-time-stat-sysbenchcpu.gz     tests-sysstate.gz
cpu-vulnerabilities.txt                     lsscsi.txt.gz             perf-time-stat-sysbenchcpu.start  tests-timestamp
cpupower.txt.gz                             lstopo.pdf.gz             proc-vmstat-sysbenchcpu.gz        vmstat-sysbenchcpu.gz
cstate-latencies-TEST_2VMS.txt              lstopo.txt.gz             proc-vmstat-sysbenchcpu.start     vmstat-sysbenchcpu.start
dmesg.gz                                    mpstat-sysbenchcpu.gz     sysbenchcpu
kconfig-4.12.14-lp151.28.10-default.txt.gz  mpstat-sysbenchcpu.start  tests-activity
kernel.version                              numactl.txt.gz            tests-sysstate
Solace:/home/dario/Local/src/mmtests #
Solace:/home/dario/Local/src/mmtests #
Solace:/home/dario/Local/src/mmtests # ls work/log/TEST_2VMS-leap15.1-2/iter-0/
cpu-topology-mmtests.txt.gz                 lscpu.txt.gz              perf-time-stat-sysbenchcpu.gz     tests-sysstate.gz
cpu-vulnerabilities.txt                     lsscsi.txt.gz             perf-time-stat-sysbenchcpu.start  tests-timestamp
cpupower.txt.gz                             lstopo.pdf.gz             proc-vmstat-sysbenchcpu.gz        vmstat-sysbenchcpu.gz
cstate-latencies-TEST_2VMS.txt              lstopo.txt.gz             proc-vmstat-sysbenchcpu.start     vmstat-sysbenchcpu.start
dmesg.gz                                    mpstat-sysbenchcpu.gz     sysbenchcpu
kconfig-4.12.14-lp151.28.10-default.txt.gz  mpstat-sysbenchcpu.start  tests-activity
kernel.version                              numactl.txt.gz            tests-sysstate
```

Synchronized Runs & Iterations

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
(in run-kvm.sh:L233, see here).



SUSE

Synchronized Runs & Iterations

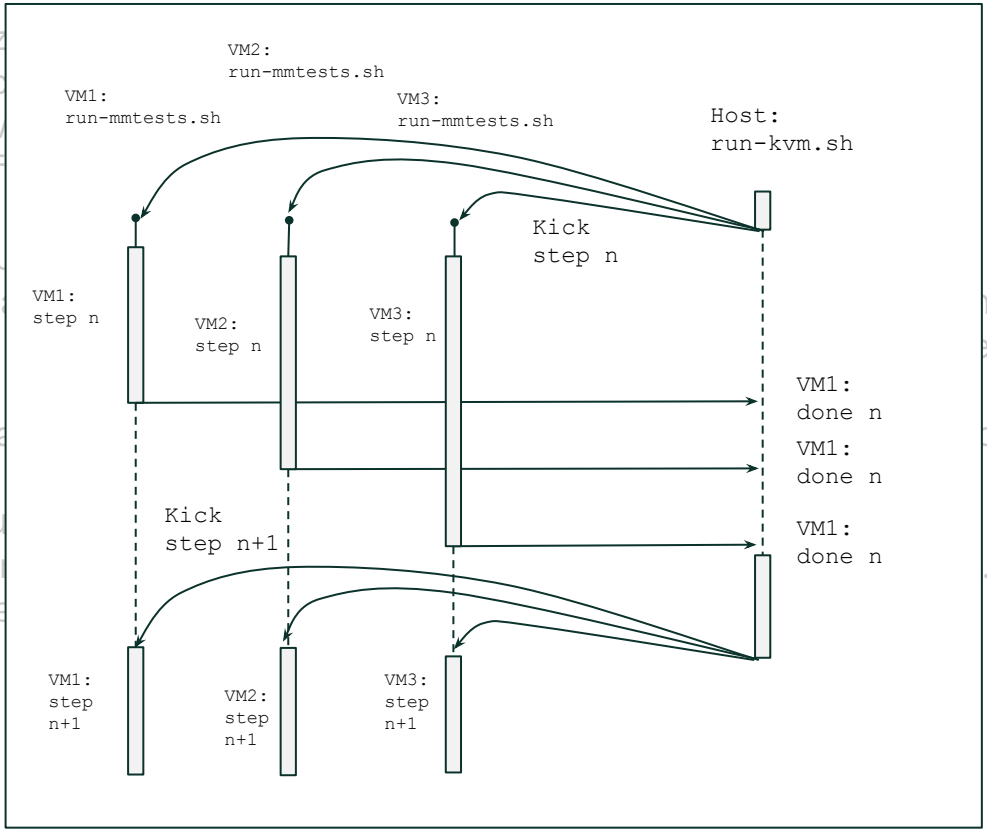
Achieving synchronization:

- VMs and host communicate:
  - Over network, for now (future: virtio-vsock / Xen's pvcalls ?)
  - With nc (future: gRPC ?)
- Tokens:
  - Host (in run-kvm.sh):
    - In state n (e.g., test_do, or iteration_begin, or iteration_end)
    - Wait for all the VMs to send state n token, when they have all reached that point
    - Signal all the VMs (at same time, with *GNU parallel*) and go to state n+1
  - VMs (in run-mmtests.sh):
    - When reaching stage n, send the relevant token to host (e.g., test_do, or iteration_begin, or iteration_end)
    - Wait for the host signal. When signal received, continue

Synchronized Runs & Iterations

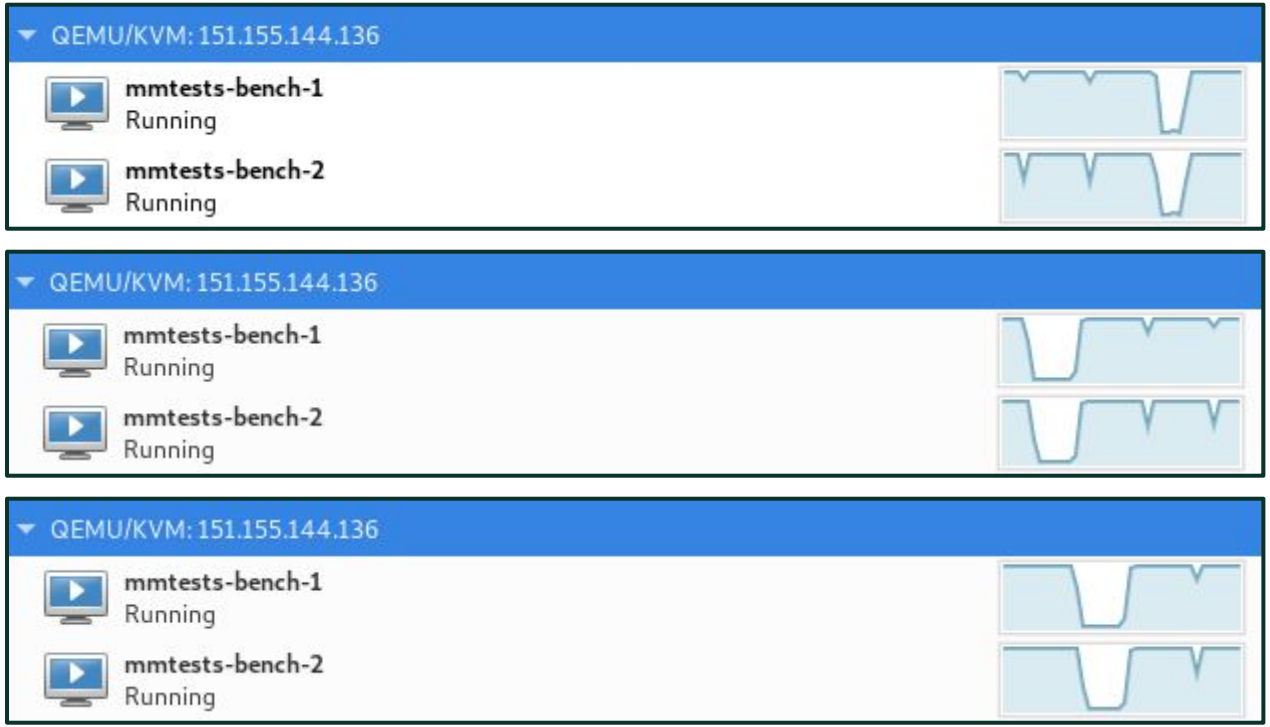
Achieving synchroniz
- VMs and host c
  - Over netw
  - With nc (f
- Tokens:
  - Host (in ru
    - In st
    - Wait
      that
    - Signa
      n+1
  - VMs (in ru
    - Wher
      or ite
    - Wait

nd)
e all reached
o to state
.g., test_do,

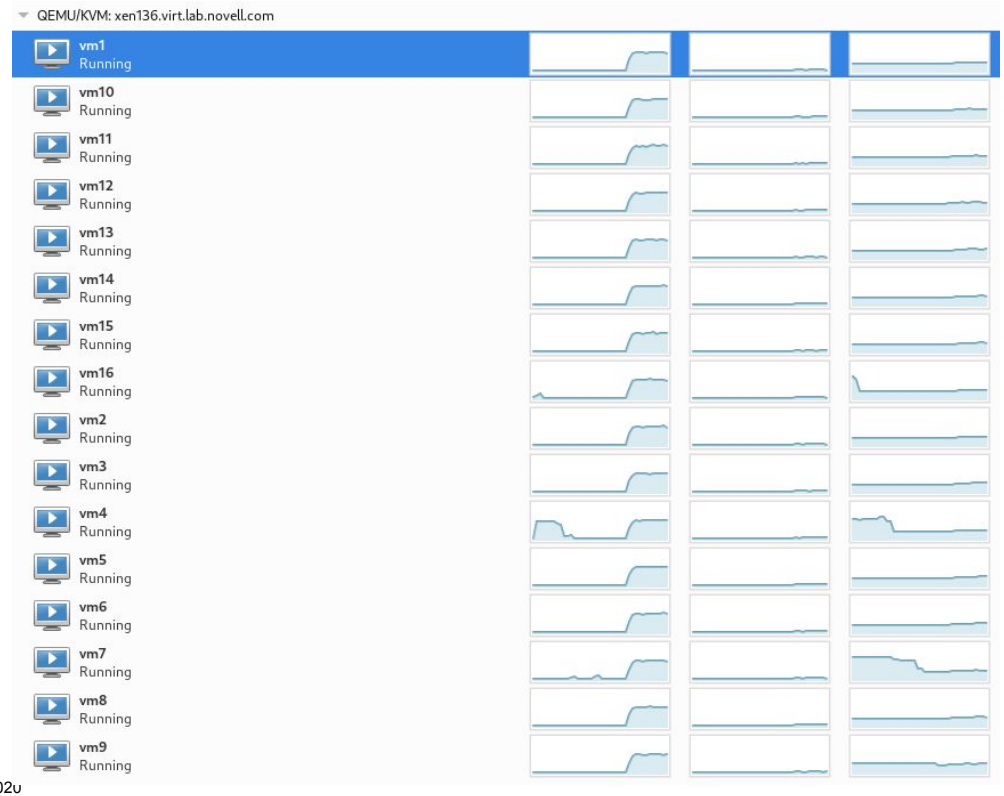


SUSE

Synchronized Runs & Iterations Examples

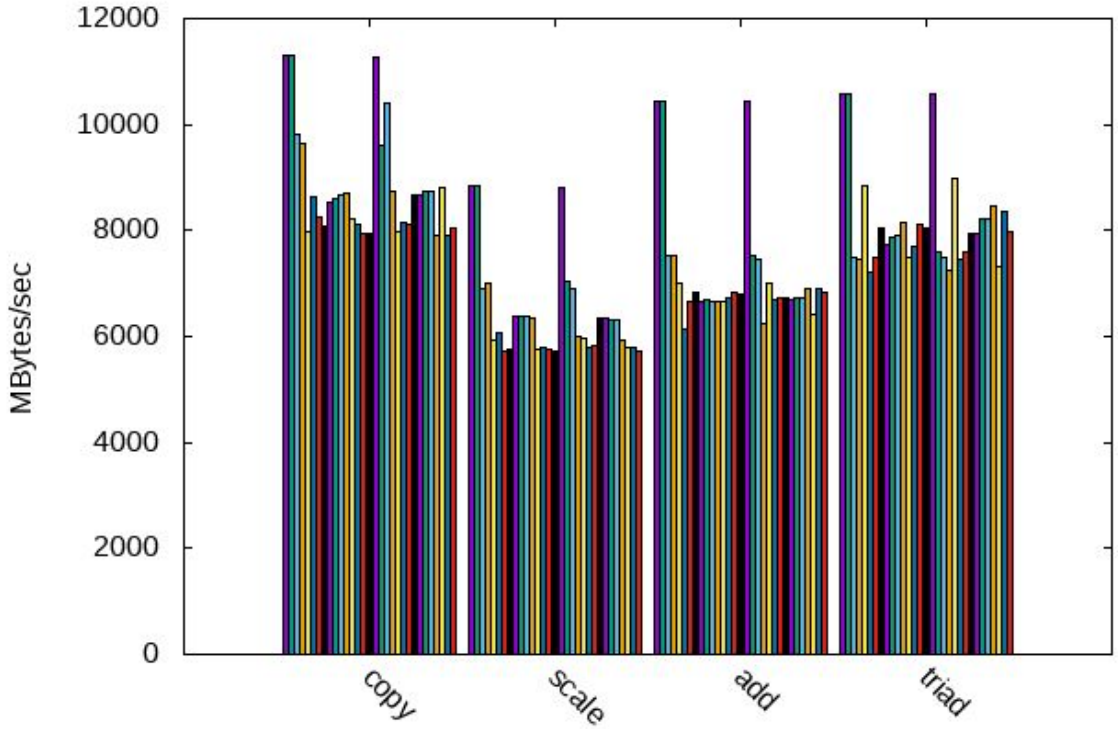


Synchronized Runs & Iterations Examples



SUSE

Synchronized Runs & Iterations Examples

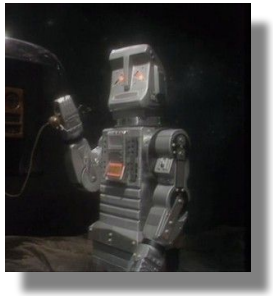


Results of 30 VMs running STREAM in sync!

SUSE

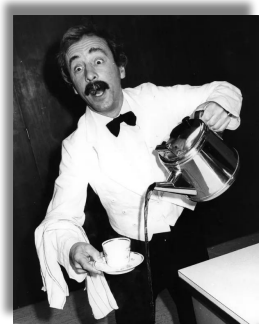
MMTests: Present & Future

MMTests @ Performance @ SUSE



Used within Marvin, our Performance Team CI (see here)

- *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* : new! For distro comparisons



MMTests @ Virtualization @ SUSE

SUSE Virtualization Team

- Jenkins: builds packages (QEMU, libvirt, kernel, Xen, ...) for all our products / distros
- Install the packages on a Jenkins “slave”
- Start (predefined) VMs and do functional testing

TODO:

- Deploy MMTests on the slave and do performance testing
- Store results
- Check for performance regressions

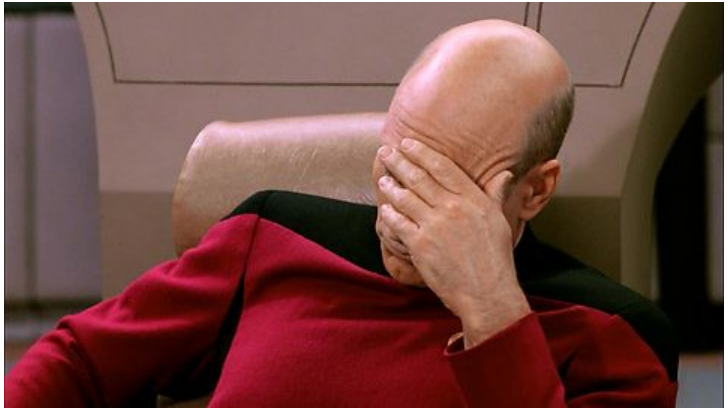
Ongoing Activities & TODOs

- ~~VM management: define or tweak XML files~~ **patches ready**
- Remote management: trigger and control the tests from outside the host
- Actual VMs provisioning: install, disk images, *patches in the works*
- Improved usability: more feedback while benchmarks are running in guests
- VMs-host communications: add more means
- Monitors: perf-kvm stats/events and kvm_stat, *patches in the works*
- ~~Integrate with tuned~~ **patches ready**
- ~~Monitors on the host, not only inside VMs~~ **patches ready**
- Run benchmarks inside Containers, Pods, Kata Containers, etc
- ~~More parallelism: VM starting / stopping~~ **patches ready**
- Packaging: make sure all dependencies available on major distros, issues with pssh (python2 related)
- Maybe, rewrite everything in Go ? [*]

[*] Note to self: only mention this if Mel **is not** in the audience!

Documentation

<<This slide has been intentionally left blank>>

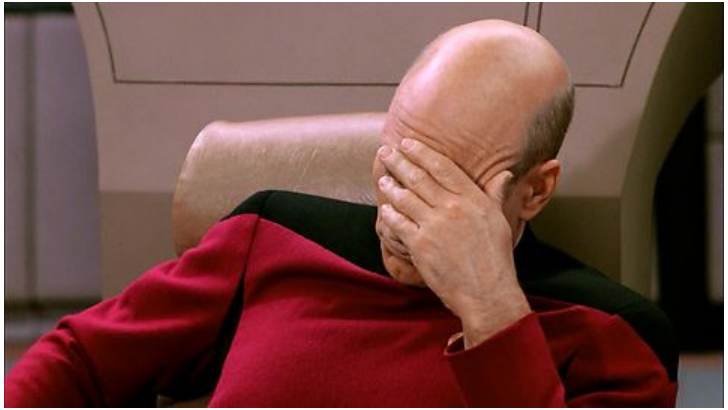


Documentation

<<Hey, didn’t I say, on February, at FOSDEM, that I’d work on this…
What’s the status now?>>

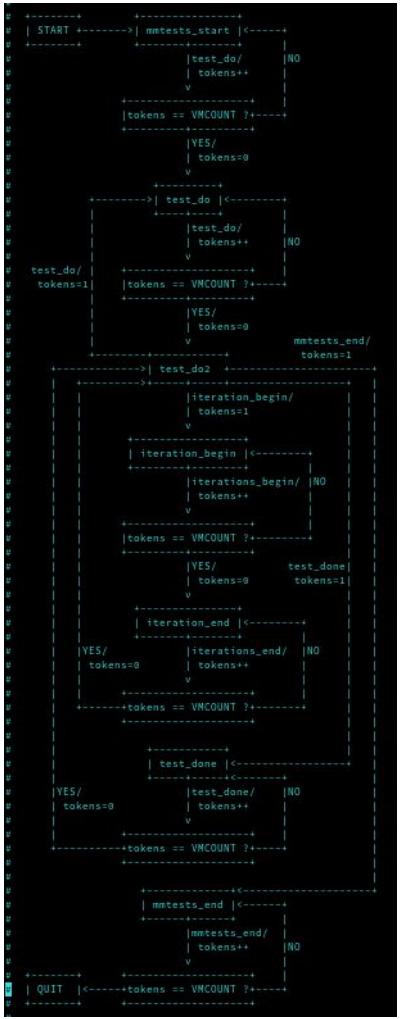
Documentation

<<This slide has been intentionally left blank>>



Documentation

<<WAIT!!! Does this count as documentation?>>



SUSE

Some Info about Myself

- Ph.D on Real-Time Scheduling, ReTiS Lab, Scuola Sant'Anna, Pisa; SCHED_DEADLINE
- 2011, Sr. Software Engineer @ Citrix
  The Xen-Project, hypervisor internals,
  Credit2 scheduler, Xen scheduler maintainer
- 2018, Virtualization Software Engineer @ SUSE; Xen, KVM, QEMU, Libvirt; Scheduling, VM's virtual topology, performance evaluation & tuning
- Spoke at XenSummit, Linux Plumbers, FOSDEM, LinuxLab, OSPM, KVM Forum, ...

Thank you!

For more information, contact SUSE at:
+1 800 796 3700 (U.S./Canada)
+49 (0)911-740 53-0 (Worldwide)

SUSE
Maxfeldstrasse
90409 Nuremberg

www.suse.com

SUSE