Automated Performance Testing
For Virtualization with MMTests

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://www.linkedin.com/in/dfaggioli/
https://twitter.com/DarioFaggioli (@DarioFaggioli)
Testing / Benchmarking / CI Tools & Suites

- OpenQA
- Jenkins
- Kernel CI
- Autotest / Avocado-framework / Avocado-vt
- Phoronix Test Suite
- Fuego
- Linux Test Project
- Xen-Project’s OSSTests
- ...
- ...
SRSLY THINKING I’LL TALK ABOUT & SUGGEST USING ANOTHER ONE?

REALLY?
HELL YEAH

I AM!
Benchmarking on Baremetal

What’s the performance impact of kernel code change “X”?
Benchmarking in Virtualization

What’s the performance impact of kernel code change “X”?

Baremetal
  Kernel (no X)
  VS.
  VM
  Kernel (no X)
  VS.
  VM
  Kernel (with X)
  VS.
  VM
  Kernel (with X)
Benchmarks in Virtualization

What’s the performance impact of kernel code change “X”?

We want to run the benchmarks inside VMs
Benchmarking in Virtualization

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

- Baremetal (Kernel (no X)) 
  - VM (Kernel (no X)) 
    - CPU bench 
    - MEM bench 
    - I/O bench

- Baremetal (Kernel (with X)) 
  - VM (Kernel (with X)) 
    - CPU bench 
    - MEM bench 
    - I/O bench

- Baremetal (Kernel (no X)) 
  - VM (Kernel (with X)) 
    - CPU bench 
    - MEM bench 
    - I/O bench

- Baremetal (Kernel (with X)) 
  - VM (Kernel (no X)) 
    - CPU bench 
    - MEM bench 
    - I/O bench
Benchmarking in Virtualization (II)

What’s the performance impact of kernel code change “X”?
Benchmarking in Virtualization (II)

We need to be able to run the benchmarks:
- Inside multiple VMs
- At the same time
  - Synchronize, among VMs, when a benchmark starts
  - Synchronize, among VMs, within each benchmark, when an iteration starts
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>
• Or me, or GH issues
MMTests

- Bash & Perl
- Fetch, build, configure & run a (set of) benchmark(s)
  - Configuration: through bash exported variables (put in config files)
  - Run the bench through wrappers (“shellpacks”)
  - Tests are run multiple times (configurable) for statistical significance
- Collects and store results
- Let you compare results
  - We have statistics: A-mean, H-mean, Geo-mean, significance, etc.
  - Can plot
- Monitors
  - While the benchmark is running:
    - Sampling `top, mpstat, vmstat, iostat, ...`
    - Collecting data from: `perf, ftrace, ...`
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, cyclicittest, …
• 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 autogenerated (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"
# ./run-mmtests.sh --config configs/config-netperf BASELINE

<change kernel / configuration / etc >

# ./run-mmtests.sh --config configs/config-netperf PTI-OFF

$ ./compare-kernels.sh  ... Or

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

<table>
<thead>
<tr>
<th></th>
<th>BASELINE</th>
<th>PTI-OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hmean</td>
<td>64</td>
<td>1205.33 ( 0.00%)</td>
</tr>
<tr>
<td>Hmean</td>
<td>128</td>
<td>2275.90 ( 0.00%)</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Hmean</td>
<td>8192</td>
<td>36768.43 ( 0.00%)</td>
</tr>
<tr>
<td>Hmean</td>
<td>16384</td>
<td>42795.57 ( 0.00%)</td>
</tr>
</tbody>
</table>
MMTests: Recap Comparisons

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

<table>
<thead>
<tr>
<th>BASELINE</th>
<th>PTI-OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gmean</td>
<td>Higher</td>
</tr>
</tbody>
</table>

• Useful as an overview
  - E.g., multiple runs of `netperf`, different packet sizes
  - ... But how are things looking overall (taking account all the sized) ?
• 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: Monitors

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

<table>
<thead>
<tr>
<th></th>
<th>SINGLE</th>
<th>OMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration User</td>
<td>45.04</td>
<td>50.75</td>
</tr>
<tr>
<td>Duration System</td>
<td>6.15</td>
<td>20.36</td>
</tr>
<tr>
<td>Duration Elapsed</td>
<td>51.16</td>
<td>20.26</td>
</tr>
</tbody>
</table>

Monitors:
- Top, iotop, vmstat, mpstat, iostat, df, ...
- Perf-event-stat, perf-time-stat, pert-top, ...
- monitors/
$ egrep "MONITORS|EVENTS" configs/config-workload-stockfish
export MONITORS_GZIP="proc-vmstat mpstat perf-time-stat"
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

<table>
<thead>
<tr>
<th></th>
<th>BASELINE</th>
<th>LOADED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hmean cpu-migrations</td>
<td>3.33</td>
<td>2.01</td>
</tr>
<tr>
<td>Hmean context-switches</td>
<td>29.12</td>
<td>30.73</td>
</tr>
<tr>
<td>Max cpu-migrations</td>
<td>999.00</td>
<td>999.00</td>
</tr>
<tr>
<td>Max context-switches</td>
<td>195.61</td>
<td>72.69</td>
</tr>
</tbody>
</table>
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"
Beware of

• (Kinf of) requires `root`
  - May need to change system properties (e.g., cpufreq governor)
  - Tries to undo all it has done...
  - ... IAC, 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 & Virtualization

Start the VM with `virsh start`
- The VM needs to exist already on the host
- The host and guest must be able to talk via network
- The host must be able to SSH in the VM without password (keys)

Copy the whole MMTests directory in 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 back in the host

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

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

The config file must have the following variables:

```bash
export MMTESTS_HOST_IP=192.168.122.1
export AUTO_PACKAGE_INSTALL=yes
```
MMTests & Multiple VMs

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

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

- Start all the VMs
- Copy MMTests dir in all of them (with `pscp`)
- Invoke `run-mmtests.sh` in all of them (with `pssh`)
- Benchmarks iterations run in sync in all VMs
- Store the host logs and info
- Fetch logs and results from the VMs and store them
MMTests & Synchronized Iterations

How to make sure tests / iterations execution is synchronized?

• 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 (== 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
MMTests & Synchronized Iterations

- 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 (== 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
MMTests as (part of) a CI loop

Already! Marvin: SUSE’s Performance Team CI

- *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
MMTests as (part of) a CI loop

Planned: SUSE’s Virtualization Team
• Jenkins: builds packages (QEMU, libvirt, ...) for all our distros
• Install packages on a “slave”
• Start (predefined) VMs and do functional testing

TODO:
• Deploy MMTests on the slave and do performance testing
• Store results
• Check for performance regressions
TODO / Doing

• VM management: define or tweak XML files
• Remote management: trigger the test from outside the host
• Improved usability: more feedback while benchmarks are running in guests
• VMs-host communications: add more means
• Monitors on the host: not only in guests
• Non VM usecases: run benchmarks in (Kata :-P) containers
• More parallelism: VM starting / stopping (already in the works)
• Packaging: make sure all dependencies available on major distros
• ...

Documentation

<This slide has been intentionally left blank>
Documentation

<This slide has been intentionally left blank>

...

...

...

...

...

But we plan to improve on that!
Conclusions

Give it a try to MMTests... Especially for Virt. benchmarking! :-)

Tell us what you think, what issues you found, etc
Myself and... Questions?

• Ph.D on Real-Time Scheduling, \texttt{SCHED\_DEADLINE}

• 2011, Sr. Software Engineer @ Citrix
  The Xen-Project, hypervisor internals,
  Credit2 scheduler, Xen scheduler maintainer

• 2018, Virtualization Software Engineer @ \texttt{SUSE}
  Still Xen, but also KVM, QEMU, Libvirt;
  Scheduling, VM’s virtual topology,
  performance evaluation & tuning

• Spoke at XenSummit, Linux Plumbers, FOSDEM,
  LinuxLab, OSPM, KVM Forum, ...
Virtualization Benchmarking “War” Stories

Physical CPUs have topology:
• Sockets, cores, threads, L{1,2,3} Caches, ...

Virtual machine can have virtual topology:
• Sockets, cores, threads: important when doing vCPU pinning
• Caches:
  – does it really matter that the VM “thinks” its CPU has caches?
  – (if yes) does the layout of such virtual caches matters?
Virtual Topology: Caches

Cache layout: does it affect guest scheduling (& performance)?

- No
- Yes!!

```c
• ttwu_queue(p, cpu)
  if (cpus_share_cache(spm_processor_id(), cpu)) {
    rq_lock(cpu_rq(cpu))
    ttwu_do_activate(cpu_rq(cpu), p)
    ttwu_do_wakeup(cpu_rq(cpu), p)
    check_preempt_curr(cpu_rq(cpu), p)
    /* If cpu_rq(cpu)->curr higher prio */
    * no IPI to cpu
    rq_unlock()
  } else {
    ttwu_queue_remote()
    llist_add(cpu_rq(cpu)->wake_list)
    smp_send_reschedule(cpu)
    /* IPI to cpu */
  }
```
Virtual Topology: Caches

Cache layout: does it impact guest scheduling (& performance)?

- **No**  Yes!!

- `ttwu_queue(p, cpu)`

  ```c
  if (cpus_share_cache(spm_processor_id(), cpu)) {
    rq_lock(cpu_rq(cpu))
    ttwu_do_activate(cpu_rq(cpu), p)
    ttwu_do_wakeup(cpu_rq(cpu), p)
    check_preempt_curr(cpu_rq(cpu), p)
    /* If cpu_rq(cpu)->curr higher prio */
    /* no IPI to cpu */
    rq_unlock()
  } else {
    ttwu_queue_remote()
    llist_add(cpu_rq(cpu)->wake_list)
    smp_send_reschedule(cpu)
    /* IPI to cpu */
  }
  ```

VM cache layout (before QEMU commit `git:9308401`):
- No L3 cache at all

Always send IPI... TO ANOTHER _virtual_ CPU!

Difference shows!
Virtual Topology: Cache Layout

- STREAM benchmark
- VM (KVM) with pinning and virtual topology tuned to match host performance
Virtual Topology: Cache Layout

Why copy lags behind when in VM?
Virtual Topology: Cache Layout

Why copy lags behind when in VM?

- Perf
  - on host we were seeing `PREFETCH` instructions being used
  - In VM, no `PREFETCH`! How so ?!?!
Virtual Topology: Cache Layout

• "Let’s just expose to the VM whether vCPUs share an L3, no big deal how big such L3 the VM sees”
• Not quite:
  – Glibc heuristics for deciding whether or not memcpy uses non-temporal stores and \texttt{PREFETCH} instrs.
  – \( \texttt{thrs} = (\text{L3 cache size} / \text{nr. threads sharing it}) + \text{L2 cache size} \)
  – Don’t \texttt{PREFETCH} if amount of data mem-copied is smaller than \( \texttt{thrs} \)

We need to expose the correct cache size to the VM