Performance Monitoring in the Linux Kernel

Davidlohr Bueso
SUSE Labs
Setting Expectations

• This is not a kernel topic per-se.

• Most of this can also be applied to userland.

• Lots to cover, not enough time to go into depth.
  - Ping me after the talk.
SUSE Kernel Performance Team

• Actively monitor and fix performance regressions.
  - Including developing new features that address scalability limitations. – R&D

• Compare performance on a wide range of workloads, configurations and systems.
  - SLE products, including openSUSE. (priority)
  - Upstream.

• We rely on MMtests almost entirely nowadays. New benchmarks and monitors are constantly being added.
Agenda

- Introduction
- Performance Methodologies.
- Performance in the Kernel.
- Marvin/MMTests.
- Real Example.
- Q&A.
Introduction

• Systems performance is a primary concern for the server industry.
  - We just bought a super expensive server, why don't I get lineal scalability?
  - Why does my system actually performance worse than on my 2-socket box?

• Unfortunately performance and scalability go way beyond buying expensive toys.

• With increasingly more powerful machines and more complex software, performance becomes more and more ambiguous.
Introduction

• As with bugs, performance issues can appear in all levels of the software stack. This talk focuses on the OS kernel.

• Performance is an ever-changing, never ending story.
  – Even simple workloads can trigger unexpected results.
Performance Methodologies
Performance Methodologies

• There are dozens of Linux performance tools available for free.
  - Observability
  - Benchmarking
  - Tuning

• Methodologies are useful in that they are guidelines for choosing tools for appropriate for performance analysis.
Methodologies (B. Gregg)

• Anti-Methodologies
  – Blame-Someone-Else, Streetlight, Drunk Man, Random Change Anti-Method, etc.

• Methodologies
  – Scientific method, workload characterization, by layer, TSA, problem statement, RTFM, drill-down.

• Whatever methodology is chosen, it will at least require thinking about the process.
Methodologies: Problem Statement

• Primary starting point.
• Is this a performance problem?
• Did this ever perform well?
  - Importance of historical data.
• What, if anything, changed recently?
• What are the relevant metrics?
  - I.e., regression report stating that IO ops/sec are low may not be a performance regression if it can be shown that the throughput of the application is high and the low IOps/sec.
• QA or customers report a bug.
  - The more data, the better.
Methodologies: Bisection

- Select two points in time (good and bad) and do binary search.
- Only works when dealing with one component and a single root cause.
- Can identify problems from unknown sources.
- Does not require familiarity with the software component.
- Results are easy to verify and the process can be entirely automated.
Methodologies: Workload Characterization

• Identifies the source, reason and type of the load, even between phases of behavior
  - What is the load?
  - Who is causing the load?
  - Why is the load called?
  - How is the load changing over time?

• Can be combined with a drill-down approach
  - Iteratively drills down until a small subset of components are identified and the problem area is well bounded.
  - May also be combined with live monitoring.
Interpreting Results

• There are generally three outcomes that may occur:
  - Legitimate problem. Fix it and carry on.
  - Not a real issue.
  - Trade-off: Certain parts can be negatively affected, but the overall benefits exceed those issues.
Performance Monitoring in the Kernel
Performance in the Kernel

- The kernel can be tightly coupled with hardware or workloads.
  - As such, performance issues might be very different from one machine to another.

- The kernel is a complex system, millions of LoC, users, bugs etc. It is too much to cope with manually.

- Efforts to automate and simplify testing in the kernel are not new.
  - Should get out of the way and let users focus on the problem and not the configs and setting up the environment.
  - Less prone to errors.
Performance in the Kernel

• Developers are more focused on performance of their own patches. Which is certainly a good thing.
• But we also need a picture of the forest.
  ‒ Enables Linux to be scalable, not just a part of it.
  ‒ Requires infrastructure both at hardware and software levels.
• … and how kernels compare between each other.
• Approaches to performance can be considered either reactive or proactive.
Reactive Monitoring: 0-day tests

- Well known in the kernel community.
- Developed by Fengguang Wu.
- Kernel test infrastructure.
- Per-patch granularity.
- Short lived workloads.
- Incredible valuable
  - The kernel is undoubtedly better because of kbuild robot.
Reactive Monitoring: 0-day tests

FYI, we noticed the below changes on

git://git.kernel.org/pub/scm/linux/kernel/git/next/linux-next.git master
commit b3fd4f03ca0b9952221f39ae6790e698bf4b39e7 ("locking/rwsem: Avoid deceiving lock spinners")

testbox/testcase/testparams: brickland3/vm-scalability/performance-300s-small-allocs

<table>
<thead>
<tr>
<th></th>
<th>%stddev</th>
<th>%change</th>
<th>%stddev</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7a215f89a0335582</td>
<td>b3fd4f03ca0b9952221f39ae67</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.38 ± 18%</td>
<td>+247.4%</td>
<td>8.27 ± 16%</td>
<td>vm-scalability.stddev</td>
</tr>
<tr>
<td>24226456 ± 0%</td>
<td>-23.0%</td>
<td>18650874 ± 1%</td>
<td>vm-scalability.time.voluntary_context_switches</td>
</tr>
<tr>
<td>1622 ± 0%</td>
<td>-15.6%</td>
<td>1369 ± 1%</td>
<td>vm-scalability.time.system_time</td>
</tr>
<tr>
<td>765 ± 0%</td>
<td>-13.1%</td>
<td>665 ± 1%</td>
<td>vm-scalability.time.percent_of_cpu_this_job_got</td>
</tr>
<tr>
<td>1036 ± 0%</td>
<td>-11.9%</td>
<td>914 ± 1%</td>
<td>vm-scalability.time.user_time</td>
</tr>
<tr>
<td>7.244e+08 ± 0%</td>
<td>-11.1%</td>
<td>6.439e+08 ± 1%</td>
<td>vm-scalability.time.minor_page_faults</td>
</tr>
<tr>
<td>1.0841276 ± 0%</td>
<td>-11.1%</td>
<td>9639981 ± 1%</td>
<td>vm-scalability.throughput</td>
</tr>
<tr>
<td>8922883 ± 3%</td>
<td>-70.2%</td>
<td>2658543 ± 4%</td>
<td>cpuidle.C1-IVT-4S.usage</td>
</tr>
<tr>
<td>1.19e+10 ± 2%</td>
<td>-66.2%</td>
<td>4.016e+09 ± 4%</td>
<td>cpuidle.C1-IVT-4S.time</td>
</tr>
<tr>
<td>1.97 ± 4%</td>
<td>+106.7%</td>
<td>4.08 ± 6%</td>
<td>perf-profile.cpu-</td>
</tr>
</tbody>
</table>
Reactive Monitoring: 0-day tests

[*] bisect-good sample
[0] bisect-bad sample
Phronix Test Suite (PTS)

- Industry-backed benchmark suite.
- Fully automated and extremely easy to use.
- Traditionally used to compare performance of different operating systems.
- Mostly multimedia (CPU-bound) workloads (Doom, ffmpeg) etc.
  - There is certainly things we can learn from this.
  - Goes way beyond microbenchmarks (Doom3).
- But the data needed to debug a problem is rarely included in the reports.
- Very compiler-dependent.
Performance in the Kernel

- There is no unique solution, approaches ought to be complimentary.
- Manual inspection is always required.
Marvin

• Created by Mel Gorman.
• System that continually runs performance-related tests.
• Mostly written in perl and bash.
• Composed of:
  - Bob: Monitors git trees looking for new kernels to test. Builds and queues these new kernels.
  - MMTests: Performance framework.
  - Inventory Manager: Configuration manager that is responsible for machine reservation, managing power, serial consoles and deploying distributions automatically
Active Monitoring: MMTests

- Design-once.
- Nowadays actively developed (mostly) by the SUSE Performance team.
- Focus more on long lived workloads.
- Complimentary to 0-day tests.
- Guided by run configurations.
- Provides multiple monitors which gives the necessary data to do performance analysis.
  - /sys/kernel/debug/*, /proc/*, perf events, ftrace, stap.
- Reports can be either standard text (ideal for changelogs) or html.
Workload Coverage

• MMTests includes most of the traditional microbenchmarks
  - Unixbench, WIS, Imbench, ebizzy, etc etc.

• Industry standard loads
  - Aim, SPEC*, etc.

• Specific subsystems
  - futex, NUMA, ipc, real-time, etc.

• Customer-like and higher level workloads
  - Compression, RDBMS, Big Data/Hadoop.
MMTests Config Files (1/3)

• Currently ~120 default files to choose from.
• Run one or more workloads, and define each workload's input parameters.
  – Default values of these parameters are quite good in many cases, but can be modified to suit specific needs.
• Grouped by workload component and characteristics

```bash
dave@linux-q0g1:~/code/mmtests/configs> ls *__workload*
config-global-dhp__workload_dedup config-global-dhp__workload_poundtime
config-global-dhp__workload_freqmine config-global-dhp__workload_pyarray
config-global-dhp__workload_futex config-global-dhp__workload_syscalls
config-global-dhp__workload_ipc config-global-dhp__workload_thpscale
config-global-dhp__workload_kerndevel config-global-dhp__workload_will-it-scale-io
config-global-dhp__workload_mailserver config-global-dhp__workload_will-it-scale-sys
```
<table>
<thead>
<tr>
<th>config-global-dhp__io-bonnie-async</th>
<th>config-global-dhp__io-filebench-webserver-small</th>
</tr>
</thead>
<tbody>
<tr>
<td>config-global-dhp__io-bonnie-async-fixed</td>
<td>config-global-dhp__io-iozone</td>
</tr>
<tr>
<td>config-global-dhp__io-bonnie-fsync</td>
<td>config-global-dhp__io-iozone-dio</td>
</tr>
<tr>
<td>config-global-dhp__io-bonnie-fsync-fixed</td>
<td>config-global-dhp__io-iozone-ram</td>
</tr>
<tr>
<td>config-global-dhp__io-compress-tmpfs</td>
<td>config-global-dhp__io-iozone-ram-dio</td>
</tr>
<tr>
<td>config-global-dhp__io-dbench4-async</td>
<td>config-global-dhp__io-iozone-small</td>
</tr>
<tr>
<td>config-global-dhp__io-dbench4-fsync</td>
<td>config-global-dhp__io-largeread-starvation</td>
</tr>
<tr>
<td>config-global-dhp__io-ddread</td>
<td>config-global-dhp__io-metadata</td>
</tr>
<tr>
<td>config-global-dhp__io-ddwrite</td>
<td>config-global-dhp__io-randread-starvation</td>
</tr>
<tr>
<td>config-global-dhp__io-filebench-oltp</td>
<td>config-global-dhp__io-seeker-block-read</td>
</tr>
<tr>
<td>config-global-dhp__io-filebench-oltp-directio</td>
<td>config-global-dhp__io-seeker-block-write</td>
</tr>
<tr>
<td>config-global-dhp__io-filebench-varmail-large</td>
<td>config-global-dhp__io-seeker-file-read</td>
</tr>
<tr>
<td>config-global-dhp__io-filebench-varmail-medium</td>
<td>config-global-dhp__io-seeker-file-write</td>
</tr>
<tr>
<td>config-global-dhp__io-filebench-varmail-small</td>
<td>config-global-dhp__io-sync-starvation</td>
</tr>
<tr>
<td>config-global-dhp__io-filebench-webproxy-large</td>
<td>config-global-dhp__io-threaded</td>
</tr>
<tr>
<td>config-global-dhp__io-filebench-webproxy-medium</td>
<td>config-global-dhp__io-xfsrepair</td>
</tr>
<tr>
<td>config-global-dhp__io-filebench-webproxy-small</td>
<td></td>
</tr>
</tbody>
</table>
export MMTESTS="futexbench-hash futexbench-requeue futexbench-wake"

# Machine configuration
# Swap configuration can be one of default, partitions, swapfile, NFS
export SWAP_CONFIGURATION=default
export SWAP_PARTITION=
export SWAP_SWAPFILE_SIZEMB=$((MEMTOTAL_BYTES/1048576))
export SWAP_NFS_MOUNT=192.168.10.7:/exports/`hostname`-swapfile
...
export VM_TRANSPARENT_HUGEPAGES_DEFAULT=default

# Optionally use a memory control group
# export MEMCG_SIZE=$((MEMTOTAL_BYTES/2))

# List of monitors
export RUN_MONITOR=yes
export MONITORS_GZIP="proc-vmstat top numa-numastat numa-meminfo numa-convergence numa-scheduling"
export MONITORS_WITH_LATENCY="vmstat iostat"
export MONITOR_UPDATE_FREQUENCY=10

# futexbench
export FUTEXBENCH_MIN_THREADS=2
export FUTEXBENCH_MAX_THREADS=$((NUMCPUS*64))
The Dashboard (1/3)

- Graphical (html) layout of all tests across all systems.
- Provides a high-level view of multiple kernel subsystems.
- Provide single value analysis.
  - Performance delta. Ie: 1.02 would mean there is a 2% difference and the color indicates whether it is a performance regression or gain.
  - Automatically guess if the result is significant. Ie: 0.98 on one test might be a 2% regression or in the noise range for different one.
  - Red = bad, green = good, gray/white = neutral.
- It's fun!
  - But it can be wrong, or misleading.
The Dashboard (2/3)

• Designed for over-simplification.

• Might appear to be a *Traffic Light* anti-methodology.
  - Intuitive, quick, attention grabbing.
  - But we don't set thresholds by guessing.
  - Comparisons are against how *other* kernels perform.
## Dashboard

<table>
<thead>
<tr>
<th>Config</th>
<th>alchiba</th>
<th>balada</th>
<th>bartholody cherubini</th>
<th>decker</th>
<th>jacobi</th>
<th>joplin</th>
<th>marvin2</th>
<th>mas</th>
<th>uv300</th>
</tr>
</thead>
<tbody>
<tr>
<td>global-dhp__cl-dbt5-smallest-ext3</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>global-dhp__cl-dbt5-smallest-ext4</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>global-dhp__cl-dbt5-smallest-xfs</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>global-dhp__cl-pgbench-ro-large-ext3</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>global-dhp__cl-pgbench-ro-large-ext4</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>global-dhp__cl-pgbench-ro-large-xfs</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>global-dhp__cl-pgbench-ro-medium-ext3</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>global-dhp__cl-pgbench-ro-medium-ext4</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>global-dhp__cl-pgbench-ro-medium-xfs</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>global-dhp__cl-pgbench-ro-small-btrfs</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>global-dhp__cl-pgbench-ro-small-ext3</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>global-dhp__cl-pgbench-ro-small-ext4</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>global-dhp__cl-pgbench-ro-small-xfs</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>global-dhp__cl-pgbench-ro-xlarge-ext3</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>global-dhp__cl-pgbench-ro-xlarge-ext4</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>global-dhp__cl-pgbench-ro-xlarge-xfs</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>global-dhp__cl-pgbench-rw-large-ext3</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>global-dhp__cl-pgbench-rw-large-ext4</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>global-dhp__cl-pgbench-rw-large-xfs</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>global-dhp__cl-pgbench-rw-medium-ext3</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>global-dhp__cl-pgbench-rw-medium-ext4</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>global-dhp__cl-pgbench-rw-medium-xfs</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
</tr>
</tbody>
</table>
Visual Aids (1/2)

• Analyzing a performance regression quickly becomes less tedious when not looking at numbers after numbers after numbers.
  - Of course, as analysis gets more involved, there is no substitution for manual inspection. Period.

• While it is expected to at least graph the benchmark results, much of the data to debug issues are in the metrics and system-wide stats.
Visual Aids (2/2)
Real Example
Thanks!
Questions?
Unpublished Work of SUSE LLC. All Rights Reserved.
This work is an unpublished work and contains confidential, proprietary and trade secret information of SUSE LLC. Access to this work is restricted to SUSE employees who have a need to know to perform tasks within the scope of their assignments. No part of this work may be practiced, performed, copied, distributed, revised, modified, translated, abridged, condensed, expanded, collected, or adapted without the prior written consent of SUSE. Any use or exploitation of this work without authorization could subject the perpetrator to criminal and civil liability.

General Disclaimer
This document is not to be construed as a promise by any participating company to develop, deliver, or market a product. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. SUSE makes no representations or warranties with respect to the contents of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. The development, release, and timing of features or functionality described for SUSE products remains at the sole discretion of SUSE. Further, SUSE reserves the right to revise this document and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes. All SUSE marks referenced in this presentation are trademarks or registered trademarks of Novell, Inc. in the United States and other countries. All third-party trademarks are the property of their respective owners.