LKP-tests

Linux kernel performance test tool

What is LKP-tests

- Framework to run benchmarks
  - Fetch/install the benchmarks
  - Prepare test environment
    - For example, format disk
  - Run benchmark with various parameters
    - Collect performance statistics
  - Benchmark/performance statistics parsing
- Framework for performance analysis
  - Result compare
- Used for benchmark running and reproducing regression captured by 0-Day

Highlights

- Rich benchmarks support
- Capture performance statistics (monitors) in addition to benchmark score
  - For example, perf stat, perf profile, vmstat, etc.
- Compare/analyze all performance statistics data (benchmark and monitors)
  - Unified data model

Benchmarks

- Integrated ~50 benchmarks
  - Fetch/install benchmarks
  - Run benchmarks with various parameters
  - Parse result of benchmarks
- Mostly micro-benchmarks, several macro-benchmarks
- Mostly server workload

Benchmark List - 1

- Scheduler
  - Hackbench, perf-bench-sched-pipe, etc.
- File system/IO
  - Fio, fsmark, iozone, dbench, dd, fileio, postmark, etc.
- Network
  - Netperf, iperf, apachebench, ku-latency, nepim, netpipe, nuttcp, qperf, siege, sockperf, stutter, tbench, thrulay, etc.
- Scalability
  - Reaim, aim7, will-it-scale, etc.
- Memory management
  - vm-scalability, chromeswap, exit_free, pft, perf-bench-numa-mem, pmbench, swapin, tlbflush, etc.

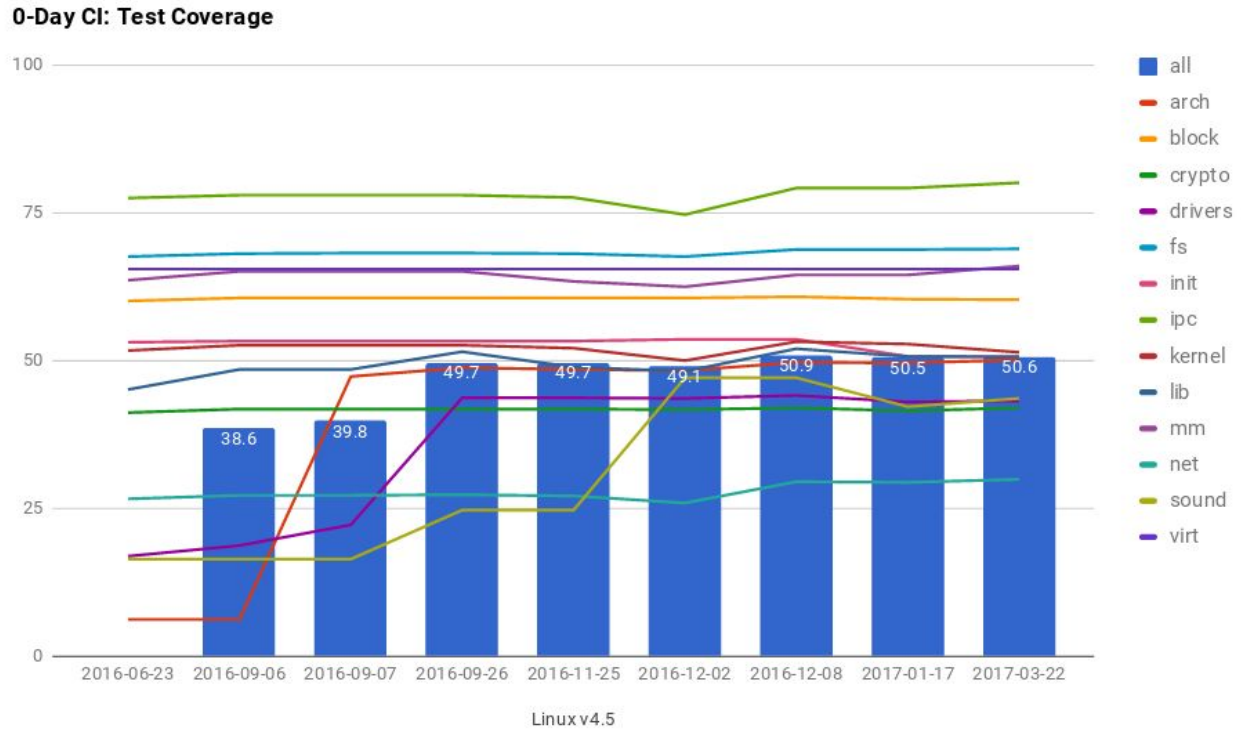
Benchmark List - 2

- Database
  - Pgbench, oltp, etc.
- Noise
  - Ftq, fwq, etc.
- HPC
  - Hpcc, linpack
- Workload emulation
  - Blogbench, ebizzy (web application server), kbuild, pbzip2, pigz, pixz, plzip, pxz, etc.
- Others
  - Tcrypt, unixbench

Functionality Test

- Memory Management
  - Bust_shm_exit, libhugetlbfs, ndctl, nvml
- File system
  - Xfstests, ext4-frags, ocfs2test
- Virtualization
  - Kvm-unit-tests
- Network
  - packetdrill
- Others
  - Cpu-hotplug, ftrace_onoff, hwsim(wifi), kernel_selftests, locktorture, ltp, mce-log, mce-test, perf_event_tests, perf-sanity-tests, pm-qa, rcutorture, stress-ng, suspend, test_bpf, trinity, piglit, etc.

Test Coverage



Monitors

- Capture performance statistics when benchmark running
  - Important for performance analysis, regression root causing
- About 38 monitors now
- Examples:
  - Perf profile: find the hot spot
  - Vmstat: CPU usage, IO bandwidth, memory usage
  - Iostat: IO bandwidth, requests
  - Latencytop: measure latency
  - Turbostat: idle, frequency, and power of the CPU
  - /proc files: meminfo, sched-debug, interrupts, etc.

Setups

- Scripts to prepare the test environment
- About 20 setup scripts
- Example setup scripts
  - disk: wait/check test disk to be ready
  - fs: format disk with file system, mount it
  - cpufreq_governor: CPU frequency governor
  - iosched: choose IO scheduler
  - dirty_thresh: writeback dirty threshold

Jobs

- Define the performance tests
- Benchmark + monitors + setups
  - Each with parameters
- In yaml format
  - Key -> value with array
- Combination of various test parameters
  - One jobfile can define a set of tests
  - For example: ext4+performance, ext4+powersave, xfs+performance, xfs+powersave

```
fs:
  - ext4
  - xfs
cpufreq_governor
  - performance
  - powersave
```

Job file example

```
# swap-test.yaml
testcase: vm-scalability
swap:
vmstat:
perf-profile:
  delay: 20
# ...
vm-scalability
  test: swap-w-seq
  nr_task:
    - 1
    - 2
    - 4
    - 8
```

Supported distribution

- Debian sid
- Ubuntu 14.04
- Centos 7
- Archlinux

Getting started

```
$ git clone https://github.com/01org/lkp-tests
$ cd lkp-tests
$ make install
$ lkp help
```

Prepare host description

- lkp-tests/hosts/<hostname>
  - Yaml format
  - Setup disks for test
- Example

```
memory: 128G
ssd_partitions: /dev/sda1
```

Install packages for a job

```
# browse and select a job you want to run, for example, job/hackbench.yaml
$ ls lkp-tests/jobs
$ lkp install $LKP_SRC/jobs/hackbench.yaml
```

Split job

```
# Generate atomic job from one job file
$ lkp split-job lkp-tests/jobs/hackbench.yaml
jobs/hackbench.yaml => ./hackbench-1600%-process-pipe.yaml
jobs/hackbench.yaml => ./hackbench-1600%-process-socket.yaml
jobs/hackbench.yaml => ./hackbench-1600%-threads-pipe.yaml
jobs/hackbench.yaml => ./hackbench-1600%-threads-socket.yaml
jobs/hackbench.yaml => ./hackbench-50%-process-pipe.yaml
jobs/hackbench.yaml => ./hackbench-50%-process-socket.yaml
jobs/hackbench.yaml => ./hackbench-50%-threads-pipe.yaml
jobs/hackbench.yaml => ./hackbench-50%-threads-socket.yaml
```

Run atomic job

```
$ lkp run hackbench-50%-process-pipe.yaml
...
2016-11-18 07:55:12 /usr/bin/hackbench -g 1 --process --pipe -l 60000
Running in process mode with 1 groups using 40 file descriptors each (== 40
tasks)
Each sender will pass 60000 messages of 100 bytes
Time: 7.388
...
```

Check result and analyze

```
# link to result directory of latest run is created automatically
$ ls result
...
$ lkp result hackbench
/result/hackbench/50%-process-pipe-performance/lkp-kvm/debian/x86_64-rhel-7.
2/gcc-6/4.5.0-2-amd64/1/
/result/hackbench/50%-process-pipe-performance/lkp-kvm/debian/x86_64-rhel-7.
2/gcc-6/4.8.0-1-amd64/0/
$ lkp ncompare -s commit=<kernel1> -o commit=<kernel2>
```

Linux kernel source

- Optional
- Put it in /c/repo/linux or $GIT_ROOT_DIR/linux

Run user supplied workload

- Use “mytest” test

Gaps

- Bug fixes
- Feedback from more users

References

- Git repo: https://github.com/01org/lkp-tests
- Documents in source code: find lkp-tests -name README.md
- Intel internal mailing list: lkp@eclists.intel.com
- External mailing list: lkp@01.org

Thanks!

Backup

Cluster: Multiple machine test

- One server and multiple clients
- Steps
  - Queue job for server
  - When schedule server job, queue job for client
  - Sync among server and clients
    - Server and client will all run the cluster job
  - Server run server side
  - Clients run client side

Cluster: Example job file

```
suite: netperf
testcase: netperf
category: benchmark

ip: ipv4

runtime: 300s
nr_threads: 200%

cluster: cs-localhost

if role server:
  netserver:

if role client:
  netperf:
    test: TCP_RR
```