
yocto
PROJECT
THE
LINUX
FOUNDATION

resulttool
oe-selftest
ptest

Tim Orling

A Brief History of `resulttool`

- **October of 2018, in an effort to move away from Testopia (a plugin for Bugzilla):**
  - Manual test cases were migrated to `meta/lib/oeqa/manual/*.json`
  - Test results from `oeselftest`, etc. output as `testresults.json` (standardized format) by `meta/lib/oeqa/core/runner.py`

```
4ef72d55 (Yeoh Ee Peng  2018-10-23 …) class OETestResultJSONHelper(object):
4ef72d55 (Yeoh Ee Peng  2018-10-23 …)
4ef72d55 (Yeoh Ee Peng  2018-10-23 …)     testresult_filename = 'testresults.json'
```

- **Managing those result files needed a new command line tool, enter `resulttool`**

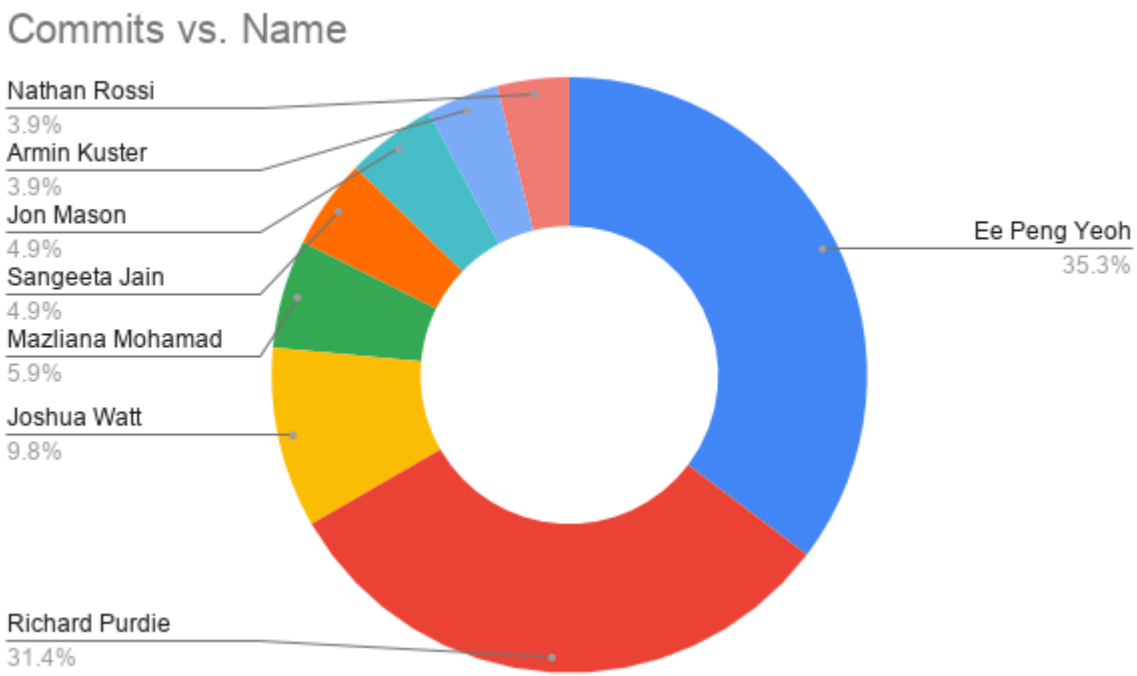
A Brief History of `resulttool` (cont'd)

- **Initial commit of `scripts/resulttool` in February, 2019.**

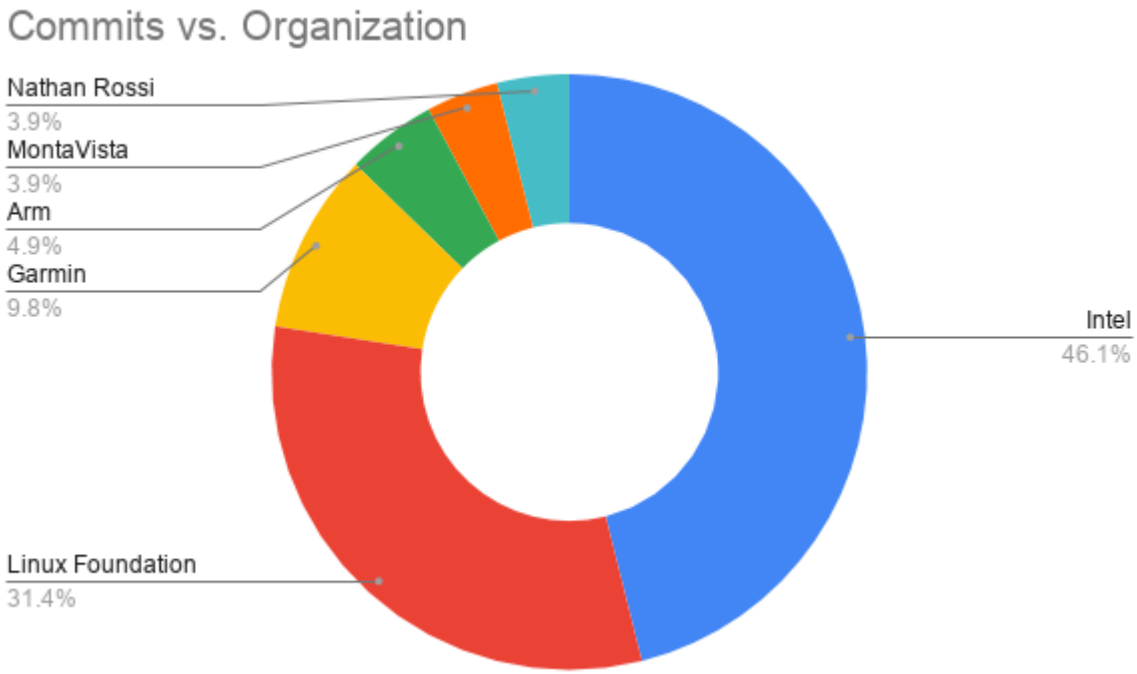
```
commit 1fd5ebdb06224489ad056e261962e23ece36fc87
Author: Yeoh Ee Peng <ee.peng.yeoh@intel.com>
Date:   Thu Feb 14 13:50:37 2019 +0800

    resulttool: enable merge, store, report and regression analysis

    OEQA outputs test results into json files and these files were
    archived by Autobuilder during QA releases. Example: each oe-selftest
    run by Autobuilder for different host distro generate a
    testresults.json file.

...
```

See also: https://wiki.yoctoproject.org/wiki/Resulttool

Contributions by Contributor Name



Contributions by Organization



`resulttool` – documentation

```
$ resulttool --help
usage: resulttool [-h] [-d] [-q] <subcommand> ...

OEQA test result manipulation tool.

options:
  -h, --help        show this help message and exit
  -d, --debug       enable debug output
  -q, --quiet       print only errors

subcommands:
  manual testcases:
    manualexecution  helper script for results populating during manual test
                     execution.
  setup:
    merge            merge test result files/directories/URLs
    store            store test results into a git repository
```

...

resulttool – documentation

```
$ resultool --help
resulttool --help
usage: resulttool [-h] [-d] [-q] <subcommand> ...

OEQA test result manipulation tool.

options:
  -h, --help        show this help message and exit
  -d, --debug       enable debug output
  -q, --quiet       print only errors

subcommands:
analysis:
    regression        regression file/directory analysis
    regression-git    regression git analysis
    report            summarise test results
    log               show logs
Use resulttool <subcommand> --help to get help on a specific command
```

...

`resulttool merge` – documentation

```
$ resulttool merge --help
usage: resulttool merge [-h] [-t] [-x EXECUTED_BY] base_results
target_results

merge the results from multiple files/directories/URLs into the target file
or directory

arguments:
  base_results          the results file/directory/URL to import
  target_results        the target file or directory to merge the
                        base_results with

options:
  -h, --help            show this help message and exit
  -t, --not-add-testseries
                        do not add testseries configuration to results
  -x EXECUTED_BY, --executed-by EXECUTED_BY
                        add executed-by configuration to each result file
```

Useful for:

- merging `manualexecution` test results with automated test results for the same platform
- merging multiple automated test results from multiple platforms for the same commit/release/build

resulttool store – documentation

```
$ resulttool store --help
usage: resulttool store [-h] [-a] [-e] [-x EXECUTED_BY] source git_dir

takes a results file or directory of results files and stores them into the
destination git repository, splitting out the results files as configured

arguments:
  source                source file/directory/URL that contain the test
                        result files to be stored
  git_dir               the location of the git repository to store the
                        results in

options:
  -h, --help            show this help message and exit
  -a, --all             include all files, not just testresults.json files
  -e, --allow-empty     don't error if no results to store are found
  -x EXECUTED_BY, --executed-by EXECUTED_BY
                        add executed-by configuration to each result file
```

As used on the autobuilder:

- `$ git clone git://git.yoctoproject.org/yocto-testresults.git`
- `$ resulttool store <workdir>/<path>/<to>/testresults.json yocto-testresults`

Other files (for instance, logs) can also be included with the `-a` option

Contributions can added to the yocto-testresults-contrib repository

resulttool regression – documentation

```
$ resulttool regression --help
usage: resulttool regression [-h] [-b BASE_RESULT_ID] [-t TARGET_RESULT_ID]
                             base_result target_result

regression analysis comparing the base set of results to the target results

arguments:
  base_result           base result file/directory/URL for the comparison
  target_result         target result file/directory/URL to compare with

options:
  -h, --help            show this help message and exit
  -b BASE_RESULT_ID, --base-result-id BASE_RESULT_ID
                        (optional) filter the base results to this result ID
  -t TARGET_RESULT_ID, --target-result-id TARGET_RESULT_ID
                        (optional) filter the target results to this result
                        ID
```

resulttool regression-git – documentation

```
$ resulttool regression-git --help
resulttool regression-git --help
usage: resulttool regression-git [-h] [-b BASE_RESULT_ID]
                                 [-t TARGET_RESULT_ID] [--branch BRANCH]
                                 [--branch2 BRANCH2] [--commit COMMIT]
                                 [--commit-number COMMIT_NUMBER]
                                 [--commit2 COMMIT2]
                                 [--commit-number2 COMMIT_NUMBER2]
                                 repo

regression analysis comparing base result set to target result set

arguments:
  repo                  the git repository containing the data

options:
  -h, --help            show this help message and exit
  -b BASE_RESULT_ID, --base-result-id BASE_RESULT_ID
                        (optional) default select regression based on
                        configurations unless base result id was provided
  -t TARGET_RESULT_ID, --target-result-id TARGET_RESULT_ID
                        (optional) default select regression based on
                        configurations unless target result id was provided
  ...
```

resulttool regression-git – documentation (cont'd)

```
$ resulttool regression-git --help
resulttool regression-git --help

options:
  -h, --help            show this help message and exit
  -b BASE_RESULT_ID, --base-result-id BASE_RESULT_ID
                        (optional) default select regression based on
                        configurations unless base result id was provided
  -t TARGET_RESULT_ID, --target-result-id TARGET_RESULT_ID
                        (optional) default select regression based on
                        configurations unless target result id was provided
  --branch BRANCH, -B BRANCH
                        Branch to find commit in
  --branch2 BRANCH2     Branch to find comparision revisions in
  --commit COMMIT       Revision to search for
  --commit-number COMMIT_NUMBER
                        Revision number to search for, redundant if --commit
                        is specified
  --commit2 COMMIT2     Revision to compare with
  --commit-number2 COMMIT_NUMBER2
                        Revision number to compare with, redundant if
                        --commit2 is specified
```

For example: https://autobuilder.yocto.io/pub/releases/yocto-3.0.rc2/testresults/testresult-regressions-report.txt

resulttool report – documentation

```
$ resulttool report --help
usage: resulttool report [-h] [--branch BRANCH] [--commit COMMIT] [-t TAG]
                         source_dir

print a text-based summary of the test results

arguments:
  source_dir            source file/directory/URL that contain the test
result
                        files to summarise

options:
  -h, --help            show this help message and exit
  --branch BRANCH, -B BRANCH
                        Branch to find commit in
  --commit COMMIT       Revision to report
  -t TAG, --tag TAG     source_dir is a git repository, report on the tag
                        specified from that repository
```

resulttool report – example of output

```
$ cat testresult-report.txt

...
================================================================================
qemux86 PTest Result Summary
================================================================================
--------------------------------------------------------------------------------
Recipe                          | Passed       | Failed    | Skipped     | Time(s)
--------------------------------------------------------------------------------
binutils                        | 197          | 0         | 5           | -
binutils-gas                    | 1184         | 0         | 0           | -
binutils-ld                     | 1401         | 0         | 313         | -
gcc                             | 106250       | 66        | 24394       | -
gcc-g++                         | 116784       | 38        | 12469       | -
gcc-libatomic                   | 22           | 1         | 27          | -
gcc-libgomp                     | 1137         | 1         | 1329        | -
gcc-libitm                      | 23           | 1         | 23          | -
gcc-libstdc++-v3                | 7732         | 32        | 4722        | -
glibc                           | 5947         | 47        | 7           | -
--------------------------------------------------------------------------------
...
```

Excerpt from: https://autobuilder.yocto.io/pub/releases/yocto-3.0.rc2/testresults/testresult-report.txt

resulttool log – documentation

```
$ resulttool log --help
usage: resulttool log [-h] [--ptest PTEST] [--dump-ptest DIR]
                      [--reproducible REPRODUCIBLE] [--prepend-run] [--raw]
                      [--raw-ptest] [--raw-reproducible]
                      source

show the logs from test results

arguments:
  source                the results file/directory/URL to import

options:
  -h, --help            show this help message and exit
  --ptest PTEST         show logs for a ptest
  --dump-ptest DIR      Dump all ptest log files to the specified directory.
  --reproducible REPRODUCIBLE
                        show logs for a reproducible test
  --prepend-run         Dump ptest results to a subdirectory named after the
                        test run when using --dump-ptest. Required if more
                        than one test run is present in the result file
  --raw                 show raw (ptest) logs. Deprecated. Alias for "--raw-
                        ptest"
  --raw-ptest           show raw ptest log
  --raw-reproducible    show raw reproducible build logs
```

For example, --dump-ptest:
http://git.yoctoproject.org/cgit/cgit.cgi/yocto-testresults/tree/runtime/poky/qemux86-64/core-image-sato-ptest-fast

How to get test results into `resulttool`

- **Two easy routes:**
  - lib/oeqa – python unittest based
  - ptest – package tests

https://wiki.yoctoproject.org/wiki/Oe-selftest
https://wiki.yoctoproject.org/wiki/Image_tests
https://wiki.yoctoproject.org/wiki/Ptest

lib/oeqa tests (aka oe-selftest)

- **`scripts/oe-selftest` is the test runner**
  - commonly used to run internal “self tests”
  - can also run runtime tests on target through a ssh tunnel
  - commonly qemu target, but any target with a known IP address and ssh or serial console can be used

https://git.yoctoproject.org/cgit/cgit.cgi/poky/tree/scripts/oe-selftest

oe-selftest – documentation

```
$ oe-selftest --help
oe-selftest --help
usage: oe-selftest [-h]
                   (-a | -R SKIPS [SKIPS ...] | -r RUN_TESTS [RUN_TESTS ...]
| -m | --list-classes | -l)
                   [-j PROCESSES] [--machine {random,all}] [-t SELECT_TAGS]
                   [-T EXCLUDE_TAGS]

Script that runs unit tests against bitbake and other Yocto related tools.
The
goal is to validate tools functionality and metadata integrity. Refer to
https://wiki.yoctoproject.org/wiki/Oe-selftest for more information.

options:
  -h, --help            show this help message and exit
  -a, --run-all-tests   Run all (unhidden) tests
  -R SKIPS [SKIPS ...], --skip-tests SKIPS [SKIPS ...]
                        Run all (unhidden) tests except the ones specified.
                        Format should be <module>[.<class>[.<test_method>]]
```

. . .

oe-selftest – documentation (cont'd)

```
$ oe-selftest --help
oe-selftest --help

options:
  -r RUN_TESTS [RUN_TESTS ...], --run-tests RUN_TESTS [RUN_TESTS ...]
                        Select what tests to run (modules, classes or test
                        methods). Format should be:
                        <module>.<class>.<test_method>
  -m, --list-modules    List all available test modules.
  --list-classes        List all available test classes.
  -l, --list-tests      List all available tests.
  -j PROCESSES, --num-processes PROCESSES
                        number of processes to execute in parallel with
  --machine {random,all}
                        Run tests on different machines (random/all).
  -t SELECT_TAGS, --select-tag SELECT_TAGS
                        Filter all (unhidden) tests to any that match any of
                        the specified tag(s).
  -T EXCLUDE_TAGS, --exclude-tag EXCLUDE_TAGS
                        Exclude all (unhidden) tests that match any of the
                        specified tag(s). (exclude applies before select)
```

Running a single “self” test

```
$ oe-selftest -r devtool
2019-10-28 09:15:06,691 - oe-selftest - WARNING - meta-selftest layer not found in
BBLAYERS, adding it
2019-10-28 09:15:11,413 - oe-selftest - INFO - Adding layer libraries:
2019-10-28 09:15:11,413 - oe-selftest - INFO -  <path>/<to>/poky/meta/lib
2019-10-28 09:15:11,414 - oe-selftest - INFO -  <path>/<to>/poky/meta-yocto-bsp/lib
2019-10-28 09:15:11,414 - oe-selftest - INFO -  <path>/<to>/poky/meta-selftest/lib
2019-10-28 09:15:11,415 - oe-selftest - INFO - Running bitbake -e to test the
configuration is valid/parsable
2019-10-28 09:15:14,122 - oe-selftest - INFO - Adding: "include selftest.inc" in
/build/ttorling/build-resulttool/conf/local.conf
2019-10-28 09:15:14,123 - oe-selftest - INFO - Adding: "include bblayers.inc" in
bblayers.conf
2019-10-28 09:15:15,887 - oe-selftest - INFO - test_devtool_add (devtool.DevtoolAddTests)
2019-10-28 09:16:07,847 - oe-selftest - INFO -  ... ok
2019-10-28 09:16:07,848 - oe-selftest - INFO - test_devtool_add_fetch
(devtool.DevtoolAddTests)
2019-10-28 09:17:40,931 - oe-selftest - INFO -  ... Ok
...
2019-10-28 10:01:19,076 - oe-selftest - INFO - SUMMARY:
2019-10-28 10:01:19,077 - oe-selftest - INFO - oe-selftest () - Ran 38 tests in 2760.686s
2019-10-28 10:01:19,077 - oe-selftest - INFO - oe-selftest - OK - All required tests
passed (successes=38, skipped=0, failures=0, errors=0)
```

lib/oeqa directory structure

```
$ tree poky/meta/lib/oeqa -L 1
poky/meta/lib/oeqa
├── buildperf
├── controllers
├── core
├── files
├── manual
├── oetest.py
├── runexported.py
├── runtime
│   ├── case.py
│   ├── cases
    …
    │   ├── gcc.py
    …
    │   ├── ping.py
    …
├── sdk
├── sdkext
├── selftest
├── targetcontrol.py
└── utils
```

The testresult.json file – in your builddir!

```
$ cat ./tmp/lib/oeqa/testresults.json
{
    "oeselftest_fedora-29_qemux86-64_20191023074504": {
        "configuration": {
            "HOST_DISTRO": "fedora-29",
            "HOST_NAME": "<hostname>",
            "LAYERS": {
                "meta": {
                    "branch": "master",
                    "commit": "f484aa6090c5fa3f69c3aed0cab154ed1f46b7ac",
                    "commit_count": 56002
                },
...
            },
            "MACHINE": "qemux86-64",
            "STARTTIME": "20191023074504",
            "TEST_TYPE": "oeselftest"
        },
        "result": {
            "devtool.DevtoolAddTests.test_devtool_add": {
                "status": "PASSED"
            },
            "devtool.DevtoolAddTests.test_devtool_add_fetch": {
                "status": "PASSED"
            },
...
}
```

lib/oeqa directory structure

```
$ tree poky/meta/lib/oeqa -L 1
poky/meta/lib/oeqa
├── buildperf
├── controllers
├── core
├── files
├── manual
├── oetest.py
├── runexported.py
├── runtime
│   ├── case.py
│   ├── cases
    …
    │   ├── gcc.py
    …
    │   ├── ping.py
    …
├── sdk
├── sdkext
├── selftest
├── targetcontrol.py
└── utils
```

A simple test: ping.py

```
$ cat lib/oeqa/runtime/cases/ping.py
#
# SPDX-License-Identifier: MIT
#

from subprocess import Popen, PIPE

from oeqa.runtime.case import OERuntimeTestCase
from oeqa.core.decorator.oetimeout import OETimeout

class PingTest(OERuntimeTestCase):

    @OETimeout(30)
    def test_ping(self):
        output = ''
        count = 0
        while count < 5:
            cmd = 'ping -c 1 %s' % self.target.ip
            proc = Popen(cmd, shell=True, stdout=PIPE)
            output += proc.communicate()[0].decode('utf-8')
            if proc.poll() == 0:
                count += 1
            else:
                count = 0
        msg = ('Expected 5 consecutive, got %d.\n'
               'ping output is:\n%s' % (count,output))
        self.assertEqual(count, 5, msg = msg)
```

Package tests (ptest)

- **On target run time tests**
  - commonly used to run unit tests
  - NOT limited to unit tests, but should still be self-contained
- **ptest-runner**
  - is a compiled C program
  - looks for ptest enabled packages in a known directory (by default /usr/lib/{foo}/ptest)
  - requires a “run-ptest” executable script to actually run the tests (python, shell, perl, anything that is executable)
  - any collateral (test code, source, special development artifacts) is also stored in /usr/lib/{foo}/ptest

https://git.yoctoproject.org/cgit/cgit.cgi/ptest-runner2/

Enabling ptest in the recipe

```
$ cat poky/meta/recipes-support/attr/acl_2.2.52.bb
SUMMARY = "Utilities for managing POSIX Access Control Lists"
...
SRC_URI = "${SAVANNAH_GNU_MIRROR}/acl/${BP}.src.tar.gz \
           file://configure.ac;subdir=${BP} \
           file://run-ptest \
...
"


inherit ptest

PTEST_BUILD_HOST_FILES = "builddefs"
PTEST_BUILD_HOST_PATTERN = "^RPM"
do_install_ptest() {
        tar -c --exclude=nfs test/ | ( cd ${D}${PTEST_PATH} && tar -xf - )
        install -d ${D}${PTEST_PATH}/include
        install -m 644 ${S}/include/builddefs ${S}/include/buildmacros
${S}/include/buildrules ${D}${PTEST_PATH}/include/
}

RDEPENDS_${PN}-ptest = "acl bash coreutils perl perl-module-filehandle perl-module-
getopt-std perl-module-posix shadow"

BBCLASSEXTEND = "native nativesdk"
```

run-ptest script

```
$ cat poky/meta/recipes-support/attr/acl/run-ptest
#!/bin/sh
#
#This script is used to run acl test suites

#umask 077

EXT3_IMAGE=ext3.img
EXT3_MOUNT_POINT=/mnt/ext3

trap 'rm -f ${EXT3_IMAGE}' EXIT

dd if=/dev/zero of=${EXT3_IMAGE} bs=1M count=1
if [ "$?" -eq 0 ]; then
        echo "PASS: dump ext3.img"
else
        echo "FAIL: dump ext3.img"
        exit 1
fi

mkfs.ext3 -F ${EXT3_IMAGE}
if [ "$?" -eq 0 ]; then
        echo "PASS: mkfs.ext3 -F ext3.img"
else
        echo "FAIL: mkfs.ext3 -F ext3.img"
        exit 1
fi
...
```

Configure your build for automated ptest

- **In local.conf (or site.conf or auto.conf):**

```
#
# Runtime testing of images
#
# The build system can test booting virtual machine images under qemu (an emulator)
# after any root filesystems are created and run tests against those images. It can also
# run tests against any SDK that are built. To enable this uncomment these lines.
# See classes/test{image,sdk}.bbclass for further details.
IMAGE_CLASSES += "testimage testsdk"
TESTIMAGE_AUTO_qemuall = "1"

#TEST_TARGET = "simpleremote" # if you are not using QEMU but running on a HW target
TEST_TARGET_IP = "192.168.7.1" # or another TAP or HW IP address
TEST_SERVER_IP = "192.168.7.1"
DISTRO_FEATURES_append = " ptest"
TEST_SUITES = "ping ssh ptest"
IMAGE_INSTALL_append = " ptest-runner acl-ptest"
```

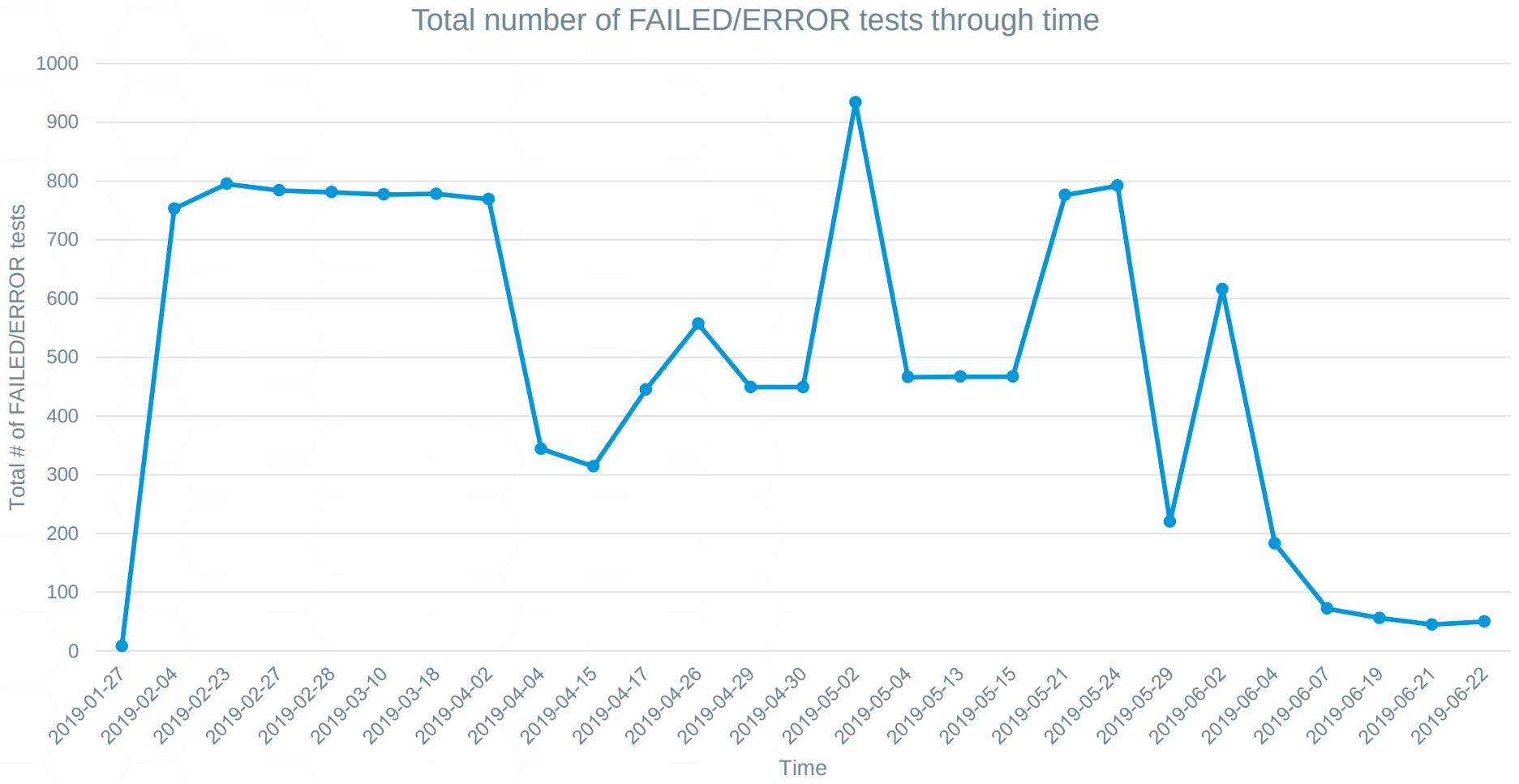
The order is important

If you only want to run a single ptest

The results

```
$ cat tmp/log/oeqa
...
 "runtime_core-image-sato-sdk_qemux86-64_20191029091356": {
...
            "MACHINE": "qemux86-64",
            "STARTTIME": "20191029091356",
            "TEST_TYPE": "runtime"
        },
        "result": {
            "ping.PingTest.test_ping": {
                "status": "PASSED"
            },
            "ptest.PtestRunnerTest.test_ptestrunner": {
                "status": "PASSED"
            },
            "ptestresult.acl.$_:_<>_f": {
                "status": "PASSED"
            },
            "ptestresult.acl.$_:_<_f": {
                "status": "PASSED"
            },
            "ptestresult.acl.$_:_<_fifo": {
                "status": "PASSED"
            },
...
```

What can we do with all this data?



Thanks to Marco Vinicio Leon Sarkis, Kyle Neary, Amanda Brindle and Amber Elliot for contributions

Spike detection

1) Compute moving average (M_k) for each value in our data set ($x_1, x_2, \ldots, x_k$)

$$M_k = (1 - \alpha)M_{k-1} + \alpha x_k$$

where α is a value between 0 and 1 which defines the weight given to the latest value of x_k. A value of zero means we will ignore the latest value.

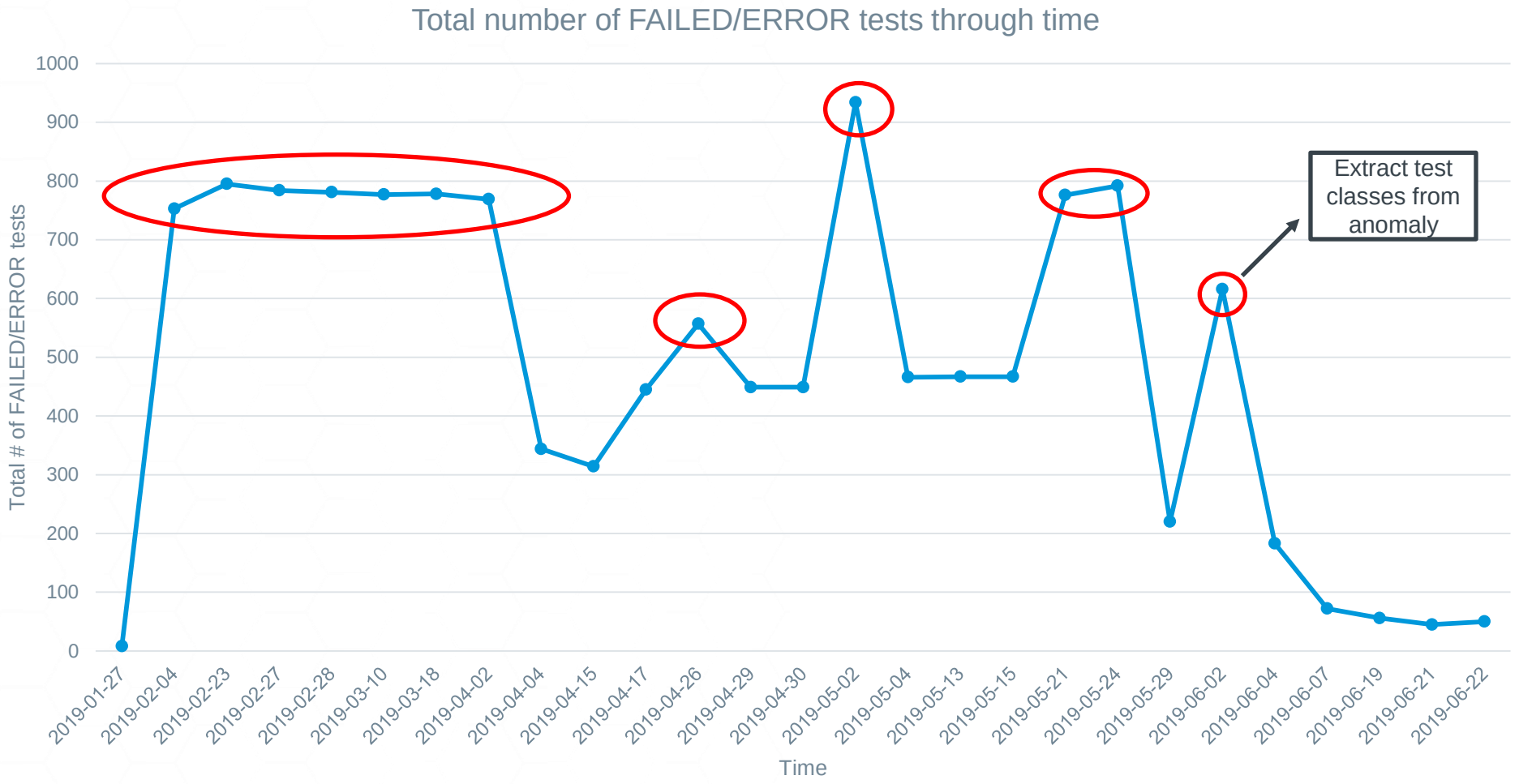
We have defined a value of α = 0.1

2) Compute how much the latest value moved from the moving average

$$\frac{x_k - M_k}{M_k}$$

If the value moved away more than 20% then it will be marked as an anomaly

Thanks to Marco Vinicio Leon Sarkis, Kyle Neary, Amanda Brindle and Amber Elliot for contributions



Thanks to Marco Vinicio Leon Sarkis, Kyle Neary, Amanda Brindle and Amber Elliot for contributions

Extracting test classes from anomaly

From raw data pull class info, sort from largest to smallest and highlight top 3 classes.

Which are the test cases, classes and suites which are generating more failed/error tests?

Suite	Class	Total
ptestresult	valgrind	502
ptestresult	perl	38
ptestresult	strace	36
ptestresult	bash	8
ptestresult	busybox	8
ptestresult	gstreamer1	6
ptestresult	python3	4
ptestresult	util-linux	4
ptestresult	dbus-test	3
ptestresult	glib-2	2
ptestresult	lttng-tools	2
ptestresult	openssl	2
ptestresult	python	1

NOTE: Test case level information is not shown because of the amount of different values won't fit in the ppt

Thanks to Marco Vinicio Leon Sarkis, Kyle Neary, Amanda Brindle and Amber Elliot for contributions

yocto PROJECT

THE LINUX FOUNDATION