Test metadata extraction

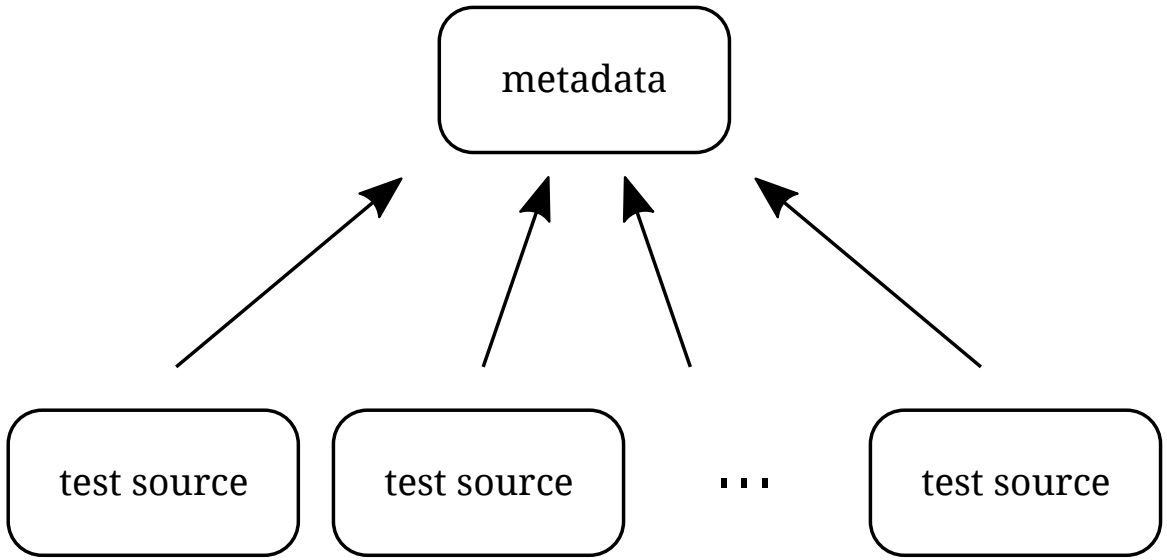
ATS 2019

Cyril Hrubis

SUSE Linux

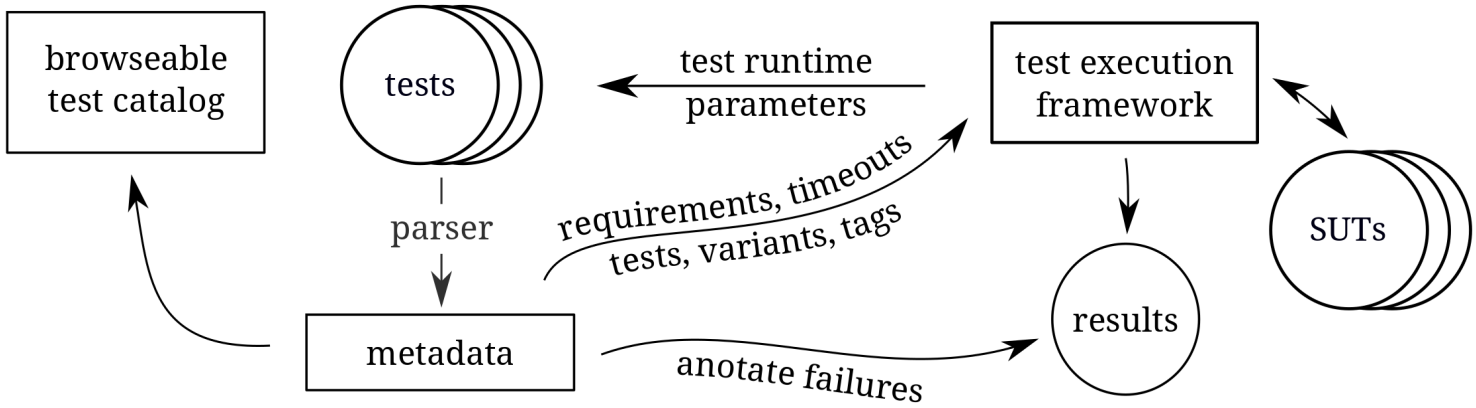
31. October 2019

The end goal



- Make each test self-describing
- The test suite is described by data extracted from tests
- This information is extracted at a build
- The main consumer for the metadata is the test execution framework at run-time

Why we need the metadata?



- Propagating test requirements
- Propagating run-time - timeouts
- Exporting documentation
- Getting rid of runtest files (test lists)
- Parallel test execution
- ...

Test requirements propagation

This would allow us to select/prepare right system to run the tests on based on hardware and software requirements.

Example requirements:

- Tests needs root
- Test needs 1GB of RAM
- Test needs block device at least 512MB in size
- Test needs i2c eeprom
- Test runs only on 64bit platforms
- Tests needs kernel version 4.6 or newer
- ...

We may also need to describe how to propagate parameters back to test later on,
e.g. pass i2c address in env variable, etc.

Propagating run-time - timeouts

- Test run-time can be used for resource allocation (prioritize short living tests etc.)
- Tests tend to be both short lived and long running
- It's nearly impossible come up with default timeout
- If timeout is too long we waste time if system crashes
- Unfortunately the run-time may depend on tests variants, we may need to make the run-time depending on variants later on

Exporting documentation

- Tests, at least in LTP, tend to include documentation as a comment in the source code
- We can easily build browse-able test catalog that includes test metadata
  E.g. test purpose, test requirements, tags, etc.
- Tests could be then searched and sorted by various criteria
- Any test suite that will generate metadata in the same format could use the same system

Getting rid of runtest files

- At the moment different subset of tests are maintained manually (e.g. syscalls is super set of syscalls-ipc in LTP)
- This is painful to maintain and prone to errors
- The plan is to be able to select particular subset of test based on the metadata
- Example queries:
  - "Regression tests (tests with Linux commit tag)"
  - "SysV-IPC tests whose run-time is shorter than 10s"
  - ...

Getting rit of runtest files

- Tests variants tend to be duplicated several times in different files as well
- The plan is to maintain test variants in the test metadata as well
- It's not clear how and where to store the data
- This is one of the things that are still in TODO list

Parallel test execution

- These days test are still executed sequentially
- This wastes resources on today's hardware as even embedded hardware has often more than one core
- We need the metadata to avoid running tests that would compete over global resources concurrently (e.g. block device, RAM, i2c eeprom, RTC, wall clock, etc.)
- The plan would be to feed a test dispatcher queue running on the system under test with tests that are safe to run concurrently

The implementation

Currently the docparse tool implements very minimalistic C tokenizer that translates C structures and comments into JSON.

```
struct tst_test test = {
        .needs_root = 1,
        .needs_device = 1,
        .dev_min_size = 1024,
        .dev_fs_type = ext4,
        .restore_wallclock = 1,
        .needs_drivers = (const char *[]) {
                "uinput",
                NULL
        },
        .needs_kconfigs = (const char *[]) {
                "CONFIG_X86_INTEL_UMIP=y",
                NULL
        },
        .tags = (const struct tst_tag[]) {
                {"linux-git", "43a6684519ab"},
                {"CVE", "2017-2671"},
                {}
        }
};
```

The implementation

- Test documentation is stored in special comment
- So far the the plan is to include test description in some kind if markup language
- The documentation format is not defined yet

Test documentation example:

```
/*\
 * Test description
 *
 * This is a test description.
 * Consisting of several lines.
\*/
```

The implementation

Example JSON output for previous examples:

```
"testcaseXY": {
 "needs_root": "1",
 "needs_device": "1",
 "dev_min_size": "1024",
 "dev_fs_type": "ext4",
 "restore_wallclock": "1",
 "needs_drivers": [
   "uinput",
 ],
 "needs_kconfigs": [
   "CONFIG_X86_INTEL_UMIP=y",
 ],
 "tags": [
   [
    "linux-git",
    "43a6684519ab"
   ],
   [
    "CVE",
    "2017-2671"
   ],
  ],
 "doc": [
   " Test description",
   "",
   " This is a test description.",
   " Consisting of several lines."
 ],
 "fname": "testcases/kernel/syscalls/foo/testcaseXY.c"
},
```

The end

Questions?