Automated run-time regression testing with Fuego

18 July 2019
Hirotaka MOTAI
Outline

● Who I am
● Overview
● Related Tools
  ○ Automated Test System / Fuego
  ○ Linux Test Project / LTP
● Issue
● Approach
● Conclusion and Future work
Who I am

Hirotaka MOTAI
- Software researcher for embedded systems of MITSUBISHI ELECTRIC Corp.

We have collaborated with LF projects.
- LTSI: Long Term Support Initiative
- AGL: Automotive Grade Linux
- Fuego: Automated Test System
  - specifically designed for testing Embedded Linux
Overview

- Linux can be adapted to various embedded devices, even though they need a hard real-time response.

- We need tons of time to ensure adequate real-time performance.
  - Real-time applications need to satisfy timing constraints.
  - We have to avoid kernel changes which might cause long delays.
Detect and Ready for analysis performance issue in Automated Testing Framework.
- In our use case with “Fuego” (presented in ELCE2018)
- measure the real-time performance, plus get tracing.
- get clues to distinguish the problem whether it was caused by our changes or not.
Overview

We have developed a part of Functional-test run-time logger to get clues to detect internal performance problems even if all of the function test are successful.
Related Tools

- **Fuego:**
  - an automated test system specifically designed for embedded Linux testing
  - [http://fuegotest.org/](http://fuegotest.org/)

- **LTP: Linux Test Project**
  - regression and conformance tests designed to confirm the behavior of the Linux kernel and glibc
  - [http://linux-test-project.github.io/](http://linux-test-project.github.io/)
Fuego is an automated test system
- created by LTSI project, based on Jenkins.
- OSS: anyone can use and contribute!
- AGL-JTA: AGL chose Fuego as standard test environment.
Fuego

- **Fuego** = "test distribution + Jenkins + host scripts + pre-packaged tests" on container
- **Fuego** can do specific tests automatically that is triggered by software update.
You can click to start manually and monitor tests on Jenkins.
You can also check test results on Jenkins.
LTP: Linux Test Project

- A huge collection of tests for Linux
  - systemcalls, semaphore, POSIX, …

- Difficult to understand test results
  - Tester has to know what to ignore, and why
    - depend on system or kernel configurations.
  - In a regression test, tester check the gaps between previous and current results.
LTP on Fuego

- Fuego has 2 categories related to LTP
  - Functional.LTP
    - 14 test scenarios with using LTP test suit
  - Functional.LTP_one_test
    - Only one LTP test that you can define with using LTP test suit

  (Show the detail later..)
Issue

- Focus on system call interface for checking regression
  - Influence performance of real-time process directly

- LTP can test system call interfaces.
  - LTP on Fuego is helpful for checking compatibility
**Issue**

● Results for system call tests look same…
● In term of regression check, looks good…..?
It is important to make the difference clear.

- What system calls were "pass"ed? Is the results same?
- Were new results "execution time of each system call" as same as previous one?
Alternative way

- Using LTP_one_test in Fuego with some modifications
  - list our important system call in spec.json

- add jobs
  ```
  # ftc add-jobs -b rpi3_81
  -t Functional.LTP_one_test
  -s syscalls-shmat01
  ```

- build jobs
  ```
  # ftc build-jobs
  rpi3_81.syscalls-* Functional.LTP_one_test
  ```

Sample: shmat(), shmdt()

```javascript
{
  "testName": "Functional.LTP_one_test",
  "specs": {
    "default": {
      "TEST":"brk01"
    },
    "syscalls-shmat01": { "TEST":"shmat01" },
    "syscalls-shmat02": { "TEST":"shmat02" },
    "syscalls-shmdt01": { "TEST":"shmdt01" },
    "syscalls-shmdt02": { "TEST":"shmdt02" },
    "syscalls-mlock03": {
      "TEST":"mlock03",
      "scenario": "syscalls"
    }
  }
}
```
Alternative way

Gap of test result of each system call become clear.
Alternative way

- Gap of test result of each system call become clear.
- However each execution time has not been clear yet.
  - The figure below shows Build Time Trend, not the execution time of system call.

This graph shows historical changes of Jenkins build time.
How to check the system call time

Do in a simple way.

○ Fuego provides a script running on the target, in fuego_test.sh.
○ Measure the execution time of the test process as below.

```bash
function test_run {
    local bdir="$BOARD_TESTDIR/fuego.$TESTDIR"
    local scenario=$FUNCTIONAL_LTP_ONE_TEST_SCENARIO

    if [ -z "$scenario" ] ; then
        report "cd $bdir; ./$one_test $FUNCTIONAL_LTP_ONE_TEST_ARGS"
    else
        report "cd $bdir; ./runtime-logger.sh ./$one_test $FUNCTIONAL_LTP_ONE_TEST_ARGS"
    fi
}
```
How to check the system call time

Do in a simple way.

- Fuego provides a script running on the target, in `fuego_test.sh`.
- Measure the execution time of the test process as below.

```bash
function test_run {
    local bdir="${BOARD_TESTDIR}/fuego.${TESTDIR}"
    local scenario=${FUNCTIONAL_LTP_ONE_TEST_SCENARIO}
    if [ -z "scenario" ] ; then
        report "cd $bdir; ./$one_test $FUNCTIONAL_LTP_ONE_TEST_ARGS"
        report "cd $bdir; ./runtime-logger.sh ./$one_test $FUNCTIONAL_LTP_ONE_TEST_ARGS"
    else
        report "cd $bdir; ./runltp -f $scenario -s $one_test"
    fi
}
```

```bash
## runtime-logger.sh
SYSCALL=$(echo $1 | sed -e "s:^./::" -e "s:[0-9].*:::")
OUTPUT=strace_${1##*/}.log
strace -f -T -e $SYSCALL -o $OUTPUT * ; RETVAL=?
echo -e "\n--strace result--";
cat $OUTPUT | grep $SYSCALL
exit $RETVAL
```
How to check the system call time

- The execution time of the test process is saved with 1usec accuracy.

The time difference between the beginning and the end of each system call is recorded.
Evaluation

● Confirmation
  ○ Inject 1sec waiting patch to "shmat()" interface in kernel.
  ○ Test and check whether the result include >1sec delay.

```c
long do_shmat(int shmid, char __user *shmaddr, int shmflg, ulong *raddr, unsigned long shmlba)
{
    struct shmid_kernel *shp;
    <<snip>>
    unsigned long populate = 0;
    +       ssleep(1);
    +
    err = -EINVAL;
    if (shmid < 0)
        goto out;

```
Evaluation

The different time can be detected in the result

- Each result is "PASS" as same as in default kernel.
- The time difference compared with the result in default kernel is roughly "1 second" each system call.
Conclusion and Future work

**Summary**
- Real-time applications need to satisfy timing constraints.
  - In term of regression, system call time in new Linux will be shorter or as same as old one.
- Fuego is useful to us for not only functional checking but also measuring to system calls.
Conclusion and Future work

● Future works
  ○ Visualization: line graph of measurement time
  ○ Discussion in Fuego community: Is this idea good, or not?
  ● Fuego Jamboree #3 are going to be held this Saturday!
THANK YOU!

Any Questions?
APPENDIX
Information

● Fuego
  ○ fuego-core:
    ● https://bitbucket.org/fuegotest/fuego-core.git
    ● e606654b8077 (core: update version numbers in common.sh)
  ○ fuego:
    ● https://bitbucket.org/fuegotest/fuego.git
    ● b5b69307f836 (install: fix debian jessie repositories)

● Target device in this slides
  ○ Raspberry Pi 3b
  ○ Rasbian, based on debian 9.4, Linux 4.14.34-v7+