Automated run-time regression testing with Fuego

21 Aug 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 lots 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

LTP
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/)

![Diagram](image_url)
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.

Related Tools »

Fuego

JTA: Jenkins Test Automation
Fuego

- **Fuego** = "test distribution + Jenkins + host scripts + pre-packaged tests" on container
- **Features:** test code build, deploy, run, results report.
  - simple board setup, running tests in batches...

```
<table>
<thead>
<tr>
<th>repos</th>
<th>Docker Container</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(Fuego Linux Distribution)</td>
</tr>
<tr>
<td>Jenkins</td>
<td>Host scripts</td>
</tr>
<tr>
<td></td>
<td>• build</td>
</tr>
<tr>
<td></td>
<td>• deploy</td>
</tr>
<tr>
<td></td>
<td>• test run</td>
</tr>
<tr>
<td></td>
<td>• processing</td>
</tr>
<tr>
<td>Fuego</td>
<td>test programs on host</td>
</tr>
<tr>
<td></td>
<td>ssh</td>
</tr>
<tr>
<td></td>
<td>test programs on target (compiled by Fuego)</td>
</tr>
<tr>
<td>Test Target Linux</td>
<td></td>
</tr>
<tr>
<td>Target device</td>
<td></td>
</tr>
</tbody>
</table>
```
You can click to start manually and monitor tests on Jenkins.

Related Tools »

Fuego

- Embedded Linux Conference 2019 - Automated run-time regression testing with Fuego
You can also check test results on Jenkins.
Functional test

- Test result: judged by return value
  - Historical results: "PASS" or "FAIL"
- 102 testsuits as functional tests:
  - LTP, LTP_one_test, OpenSSL, aiostress, busybox, bzip2, glibc, hello_world, iptables, kernel_build, kselftest, linus_stress, netperf, ptest, stress, tar, year2038, ...
**Fuego**

- **Benchmark test**
  - Test result: "PASS" if measured value < threshold
  - Historical results: measured value
  - 42 testsuits as benchmark tests:
    - Dhrystone, IOzone, Interbench, Whetstone, bonnie, cyclic test, dbench4, deadline test, hackbench, iperf, Imbench2, nbench_byte, netperf, svsetest, x11perf ...

![Graph showing benchmark test results with threshold and measured value.]
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
ISSUE & APPROACH
Issue

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

- LTP can test syscall interfaces.
  - LTP on Fuego is helpful for checking compatibility
Issue

● Results for syscall tests look same...
● In term of regression check, looks good.....?
Issue

● It is important to make the difference clear.
○ What syscalls were "pass"ed? Is the results same?
○ Were new results "execution time of each syscall" as same as previous one?
Alternative way

● Using LTP_one_test in Fuego with some modifications
  ○ list our important syscall in spec.json

● add jobs

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

● build jobs

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

Sample: shmat(), shmdt()

```json
{
  "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"
    }
  }
}
```

ftc: "fuego test control" tool. a command line tool used to perform various functions in Fuego.
Alternative way

- Gap of test result of each syscall become clear.
Alternative way

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

![Build Time Trend](image)

This graph shows historical changes of Jenkins build time.
How to check the syscall 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
}
```
How to check the syscall 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 $FUNCTIONAL_LTP_ONE_TEST_ARGS"
      strace -f -T -e "$SYSCALL" -o $OUTPUT $* ; RETVAL=$?
      echo -e "$OUTPUT
      grep "$SYSCALL" $OUTPUT
      exit $RETVAL
    else
      report "cd $bdir; ./runtime-logger.sh ./one_test $FUNCTIONAL_LTP_ONE_TEST_ARGS"
      strace -f -T -e "$SYSCALL" -o $OUTPUT $* ; RETVAL=$?
      echo -e "$OUTPUT
      grep "$SYSCALL" $OUTPUT
      exit $RETVAL
    fi
}
```

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

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

```
tst_test.c:980: INFO: Timeout per run is 0h 05m 00s
shmat01.c:147: PASS: shmat() succeeded to attach NULL address
shmat01.c:147: PASS: shmat() succeeded to attach aligned address
shmat01.c:147: PASS: shmat() succeeded to attach unaligned address with SHM_RD
shmat01.c:147: PASS: shmat() succeeded to attach aligned address with SHM_READONLY, and got SIGSEGV on write

Summary:
  passed 4
  failed 0
  skipped 0
  warnings 0

--strace result--
1148  shmat(327681, NULL, 0)  = 0x76ef8000 <0.000040>
1148  shmat(360449, NULL, 0)  = 0x76efc000 <0.000026>
1148  shmat(360449, 0x76ef8000, 0)  = 0x76ef8000 <0.000036>
1148  shmat(360449, 0x76efbff, SHM_RD) = 0x76ef8000 <0.000035>
1148  shmat(360449, 0x76ef8000, SHM_RDONLY) = 0x76ef8000 <0.000033>
```

the time difference between the beginning and the end of each system call
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;
```
The different time can be detected in the result.

Each result was "PASS" as same as in default kernel.

The time differences compared with the result in default kernel were roughly "1 second" each system call.
Conclusion and Future work

Summary

○ Real-time applications need to satisfy timing constraints.
  ● In term of regression, syscall 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 syscalls.
Conclusion and Future work

● Future works
  ○ Visualization: line graph of measurement time
    ● Discussed this idea at Fuego Jamboree #3 (20 July 2019)
    ● Current status: Developing it as Benchmark test, not Functional test.
THANK YOU!

Any Questions?
APPENDIX
Resources

- **FUEGO**
  - [http://fuegotest.org/](http://fuegotest.org/)

- **LTP: Linux Test Project**
  - [http://linux-test-project.github.io/](http://linux-test-project.github.io/)

- **strace**
  - [https://strace.io/](https://strace.io/)

- **LTSI Project**
  - [https://ltsi.linuxfoundation.org/](https://ltsi.linuxfoundation.org/)

- **AGL Test framework: AGL-JTA**
  - [https://wiki.automotivelinux.org/agl-jta](https://wiki.automotivelinux.org/agl-jta)
Information

- **Fuego**
  - **fuego-core:**
    - [https://bitbucket.org/fuegotest/fuego-core.git](https://bitbucket.org/fuegotest/fuego-core.git)
    - e606654b8077 (core: update version numbers in common.sh)
  - **fuego:**
    - [https://bitbucket.org/fuegotest/fuego.git](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+
Fuego testsuit

Benchmark: 42
- Dhrystone, GLMark, IOzone, Interbench, Java, OpenSSL, Stream, Whetstone, aim7, backfire, blobsallad, bonnie, cyclicstest, dbench3, dbench4, dd, deadlinetest, ebizzy, ffsb, fio, fs_mark, gtkperf, hackbench, himeno, iperf, iperf3, linpack, lmbench2, migratetest, nbench_byte, netperf, netpipe, pmqtest, ptsematest, reboot, signaltest, sigwaittest, svsematest, sysbench, tiobench, vuls, x11perf
- (exclude fuego selftests: 2)
Fuego testsuit

- Functional: $102 = 96 + 6$
  - LTP, LTP_one_test, OpenSSL, acpid, aiostress, arch_timer, at, autopkgtest, bc, bgpd, bind, boost, brctl, bsdiff, busybox, bzip2, cmt, commonAPI_C++, commonAPI_Dbush, commonAPI_Somelp, crashme, croco, cryptsetup, curl, dovecot, ethtool, expat, file, fixesproto, fontconfig, fsfuzz, ft2demos, fuse, giflib, glib, glib2, glibc, hciattach, hello_world, imagemagick, iperf3_server, ipmi, iptables, iputils, ipv6connect, jpeg, kernel_build, kmod, kselftest, libogg, libpcap, librsvg, libspeex, libtar, libwebsocket, libxml, linaro, linus_stress, lwip, mcelog, mesa_demos, module_init_tools, multipathd, neon, net-tools, netperf, nscd, nss, openct, openhpid, ospf6d, ospfd, pam, perl-xml-simple, pi_tests, pixman, pppd, protobuf, ptest, rmaptest, rpm, scifab, scrashme, sdhi_0, serial_rx, stress, synctest, tar, thrift, tiff, trousers, vconfig, vsomeip, xorg-macros, year2038, zlib
  - batch, batch_bc, batch_default, batch_hello, batch_nested, batch_smoketest
  - (exclude fuego selftests: 16)
LTP on Fuego

- has 14 specs
  - # ftc add-jobs -b yourboard -t Functional.LTP -s default
  - # ftc add-jobs -b yourboard -t Functional.LTP -s docker
  - # ftc add-jobs -b yourboard -t Functional.LTP -s selection
  - # ftc add-jobs -b yourboard -t Functional.LTP -s install
  - # ftc add-jobs -b yourboard -t Functional.LTP -s make_pkg
  - # ftc add-jobs -b yourboard -t Functional.LTP -s selectionwithrt
  - # ftc add-jobs -b yourboard -t Functional.LTP -s ltle
  - # ftc add-jobs -b yourboard -t Functional.LTP -s ptsonly
  - # ftc add-jobs -b yourboard -t Functional.LTP -s smoketest
  - # ftc add-jobs -b yourboard -t Functional.LTP -s quickhit
  - # ftc add-jobs -b yourboard -t Functional.LTP -s rtonly
  - # ftc add-jobs -b yourboard -t Functional.LTP -s somefail
  - # ftc add-jobs -b yourboard -t Functional.LTP -s quickhitwithskips
  - # ftc add-jobs -b yourboard -t Functional.LTP -s security