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

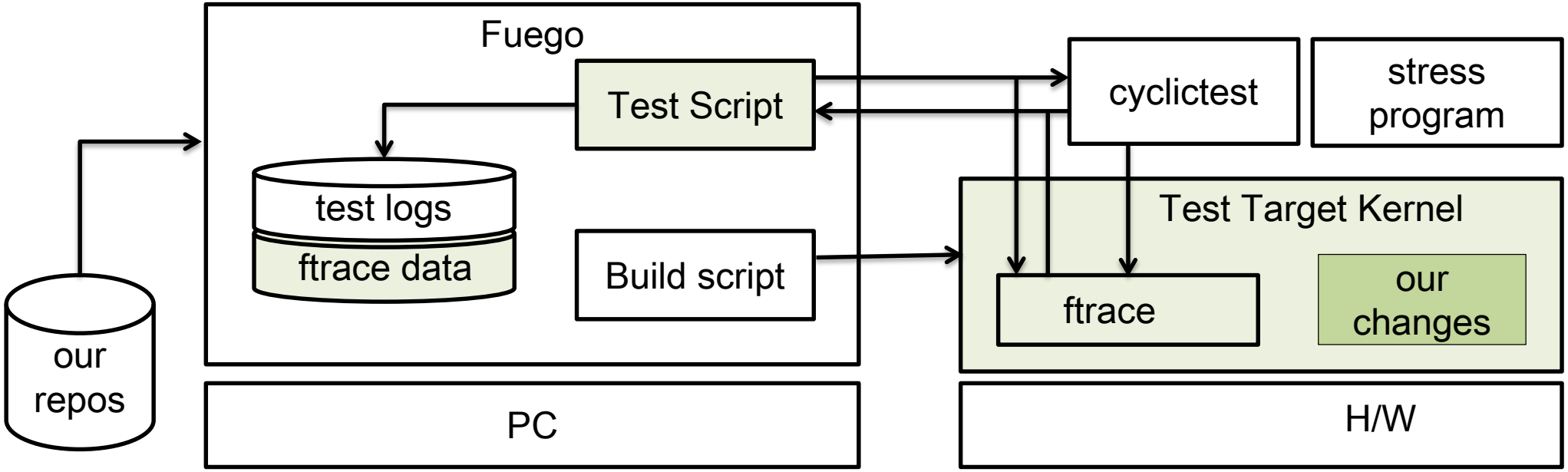
Overview

- 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.

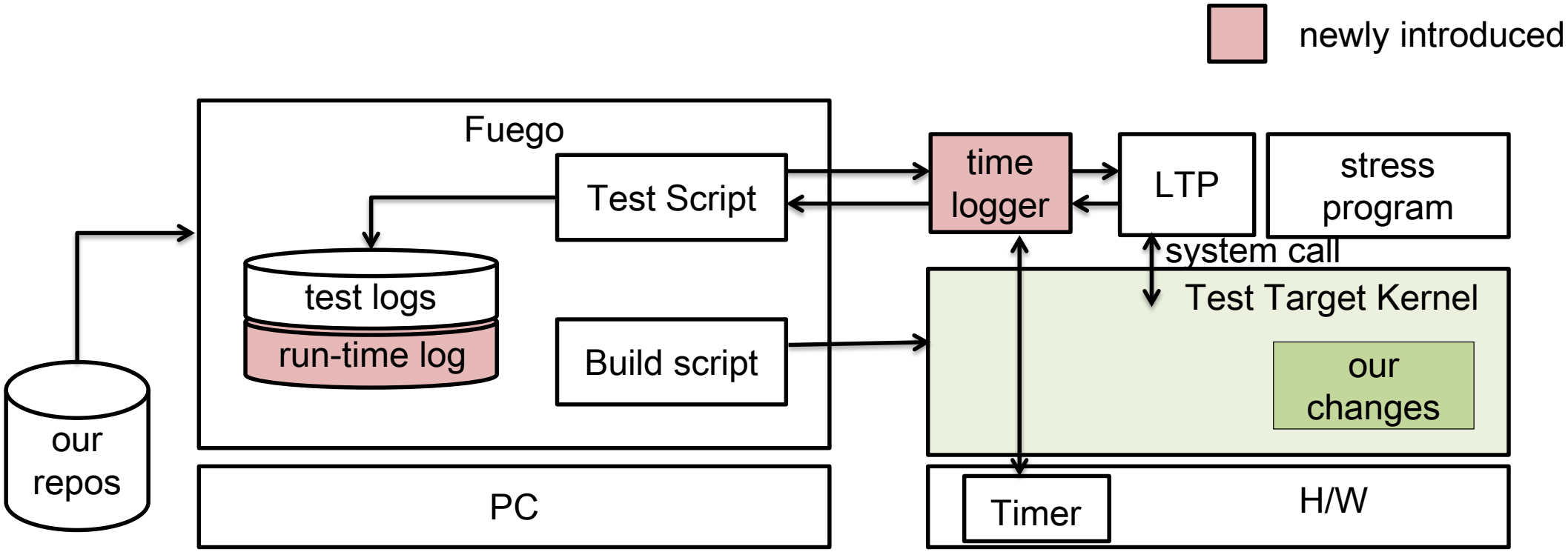
Overview

- 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

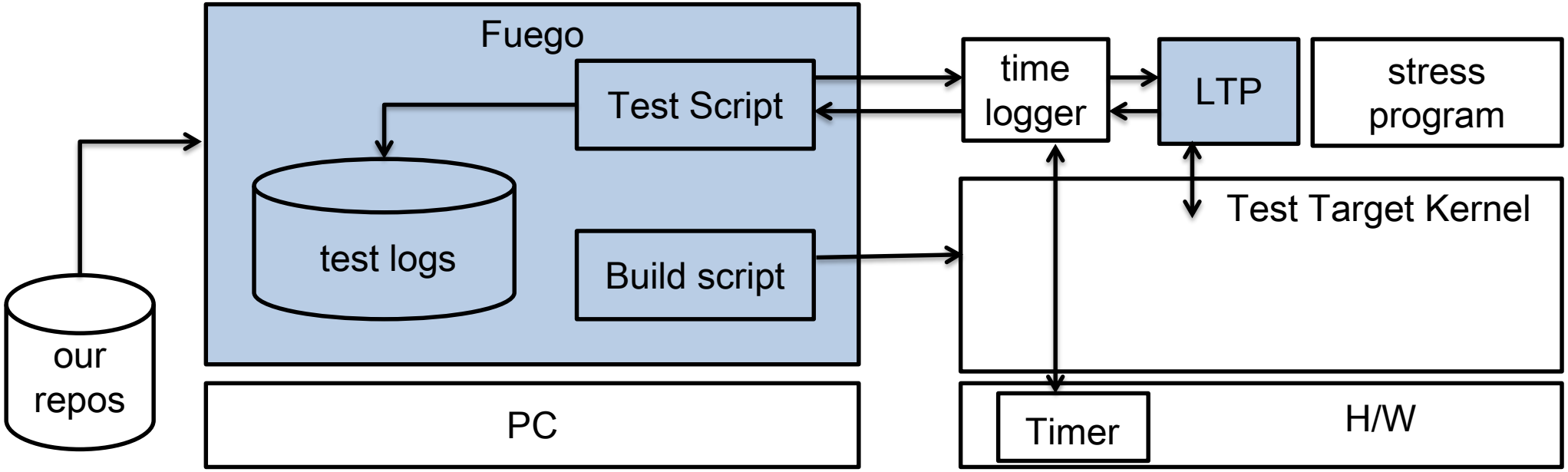
- We have developed a part of Functional-test run-time logger to get clues to detect internal performance problems even if all of the function test are successful.



Related Tools

- Fuego:
  - an automated test system specifically designed for embedded Linux testing
  - http://fuegotest.org/

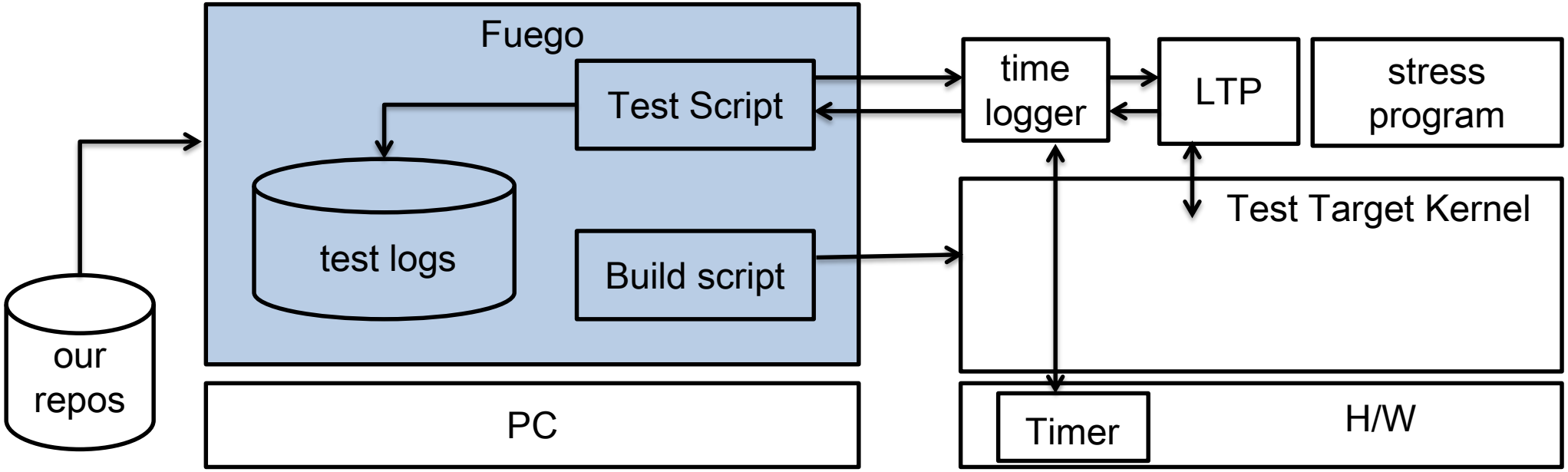
- LTP: Linux Test Project
  - regression and conformance tests designed to confirm the behavior of the Linux kernel and glibc
  - http://linux-test-project.github.io/



Fuego

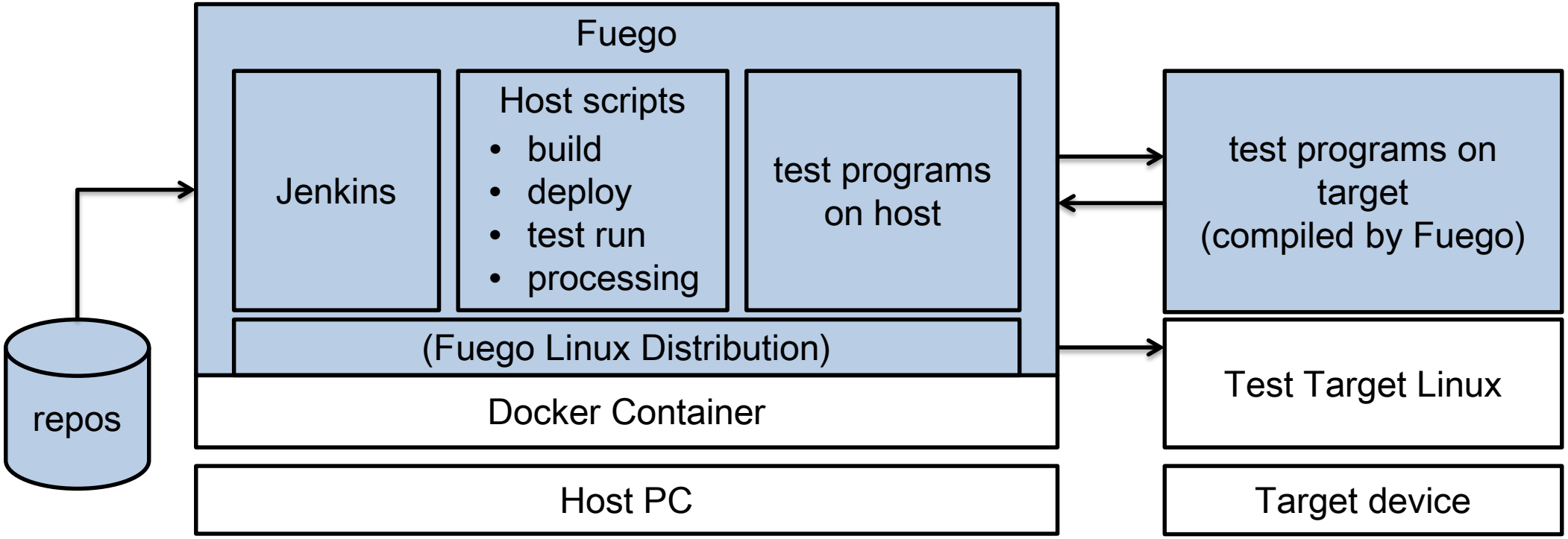
JTA: Jenkins Test Automation

- 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.



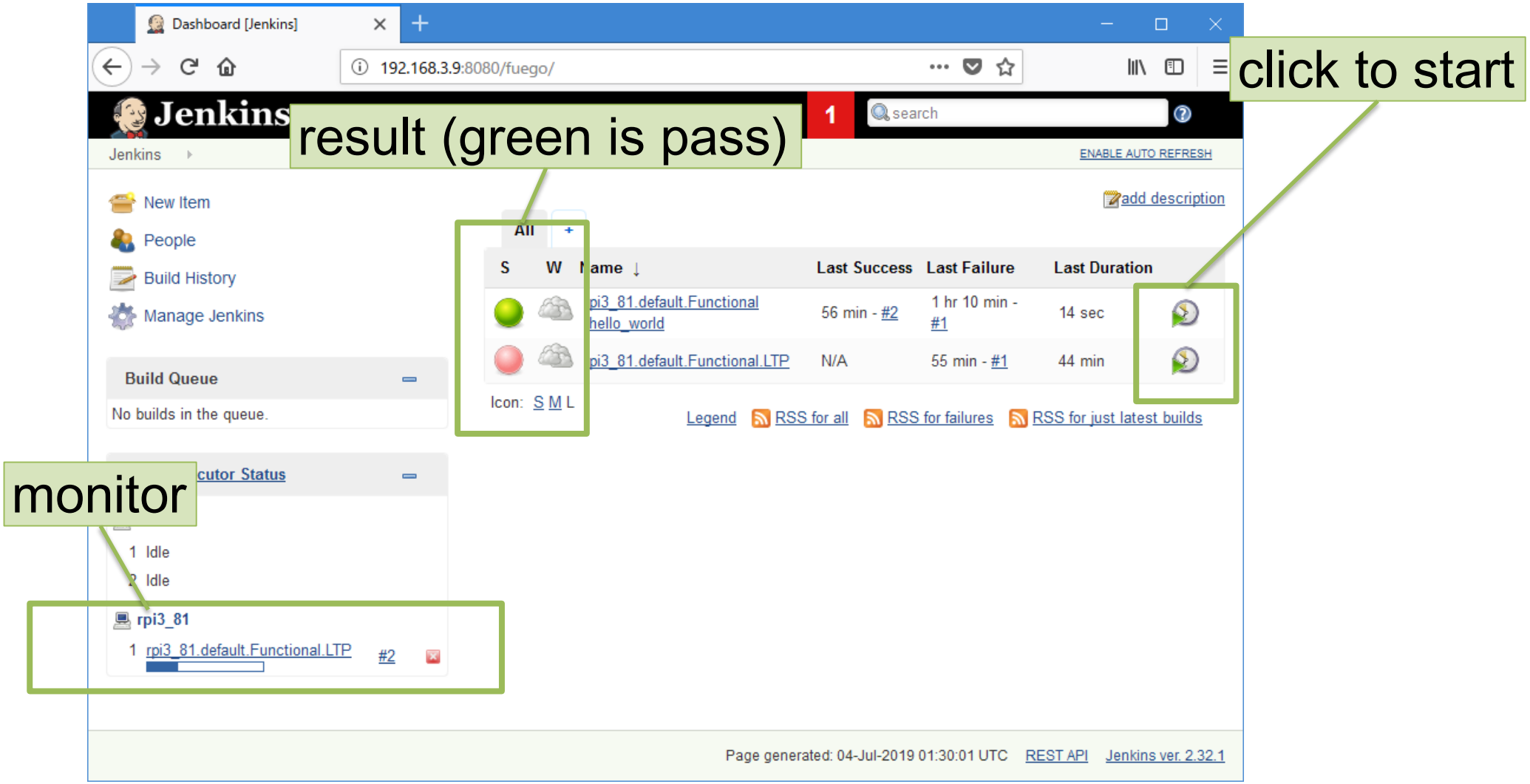
Fuego

- Fuego = "test distribution + Jenkins + host scripts + pre-packaged tests" on container
- Fuego can do specific tests automatically that is triggered by software update.



Fuego

- You can click to start manually and monitor tests on Jenkins.



Fuego

- You can also check test results on Jenkins.

rpi3_81.default.Functional.LTP

192.168.3.9:8080/fuego/job/rpi3_81.default.Functional.LTP/

Jenkins

search

Jenkins › rpi3_81.default.Functional.LTP ›

ENABLE AUTO REFRESH

Back to Dashboard
Status
Changes
Workspace
Build Now
Delete Project
Configure

Build History trend

find

#2 04-Jul-2019 01:18

#1 04-Jul-2019 00:34

testlog run.json fuegolog devlog prolog.sh xlsx skiplist

RSS for all RSS for failures

Project rpi3_81.default.Functional.LTP

add description

Disable Project

rpi3_81-Functional.LTP

board: rpi3_81
kernel: 4.14.34-v7+

			results totals			
test spec	build number	test set	pass	fail	skip	err
default	1	sem_close	4	0	0	0
		sem_destroy	2	0	0	0
		sem_getvalue	5	0	0	0
		sem_init	9	0	1	0
		sem_open	12	0	0	0
		sem_post	7	0	0	0
		sem_timedwait	11	0	0	0
		sem_unlink	10	0	0	0
		sem_wait	8	0	0	0
		syscalls	1141	8	0	0

Workspace

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..)

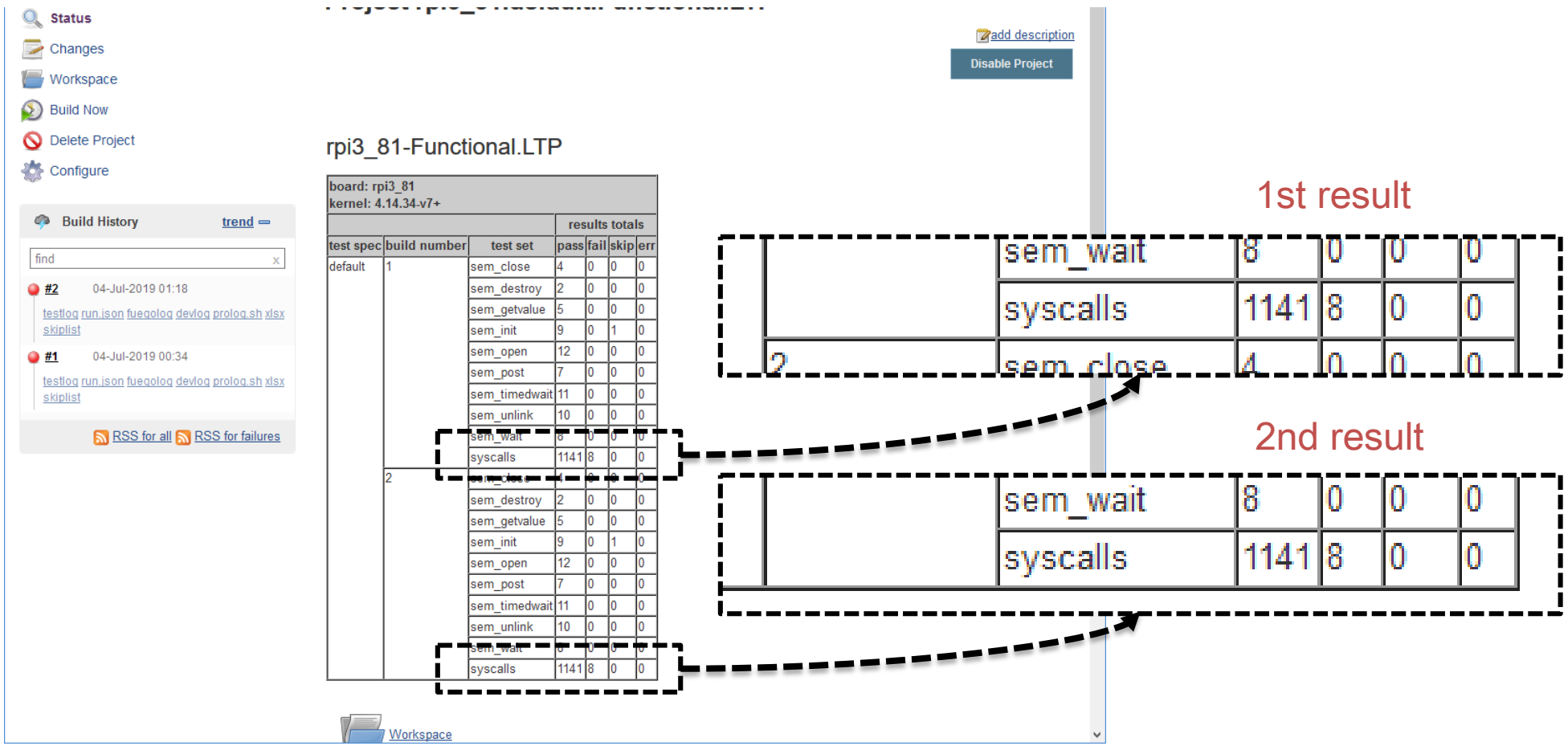
Issue

- 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

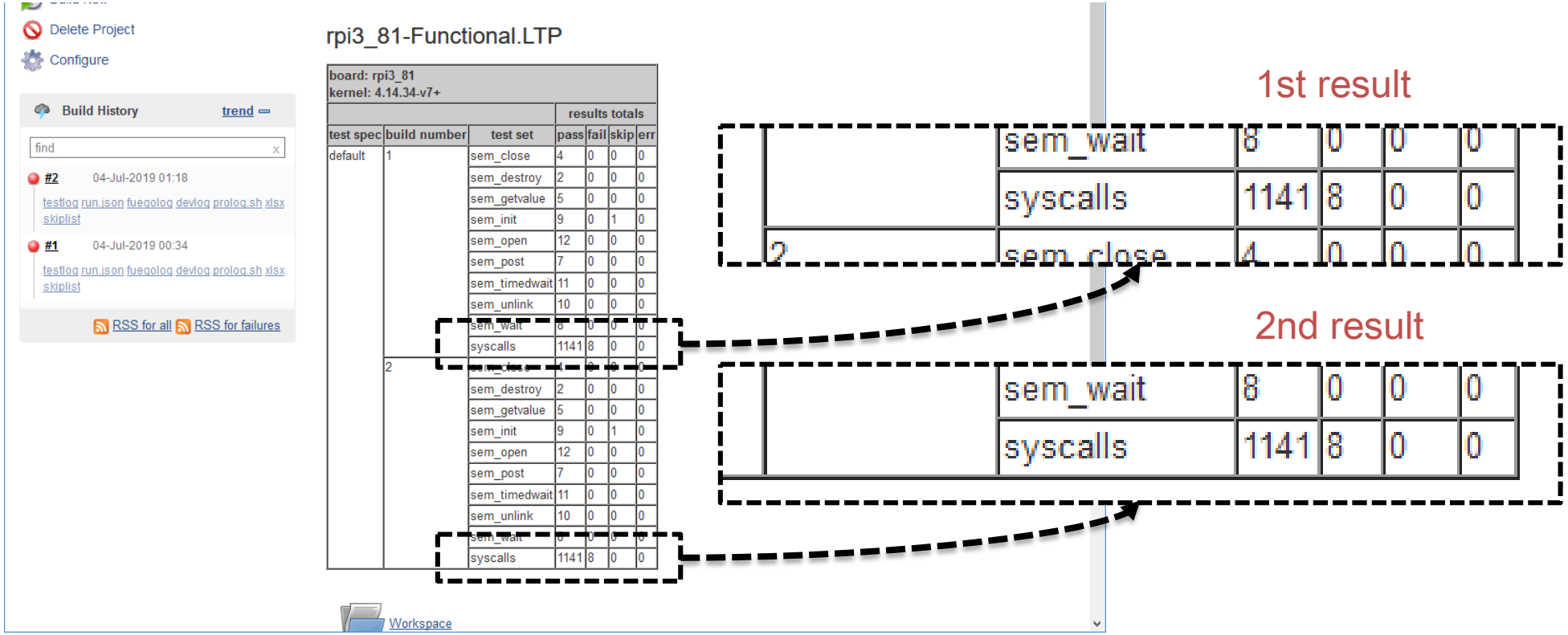
Issue

- Results for system call tests look same…
- In term of regression check, looks good…..?



Issue

- 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
```

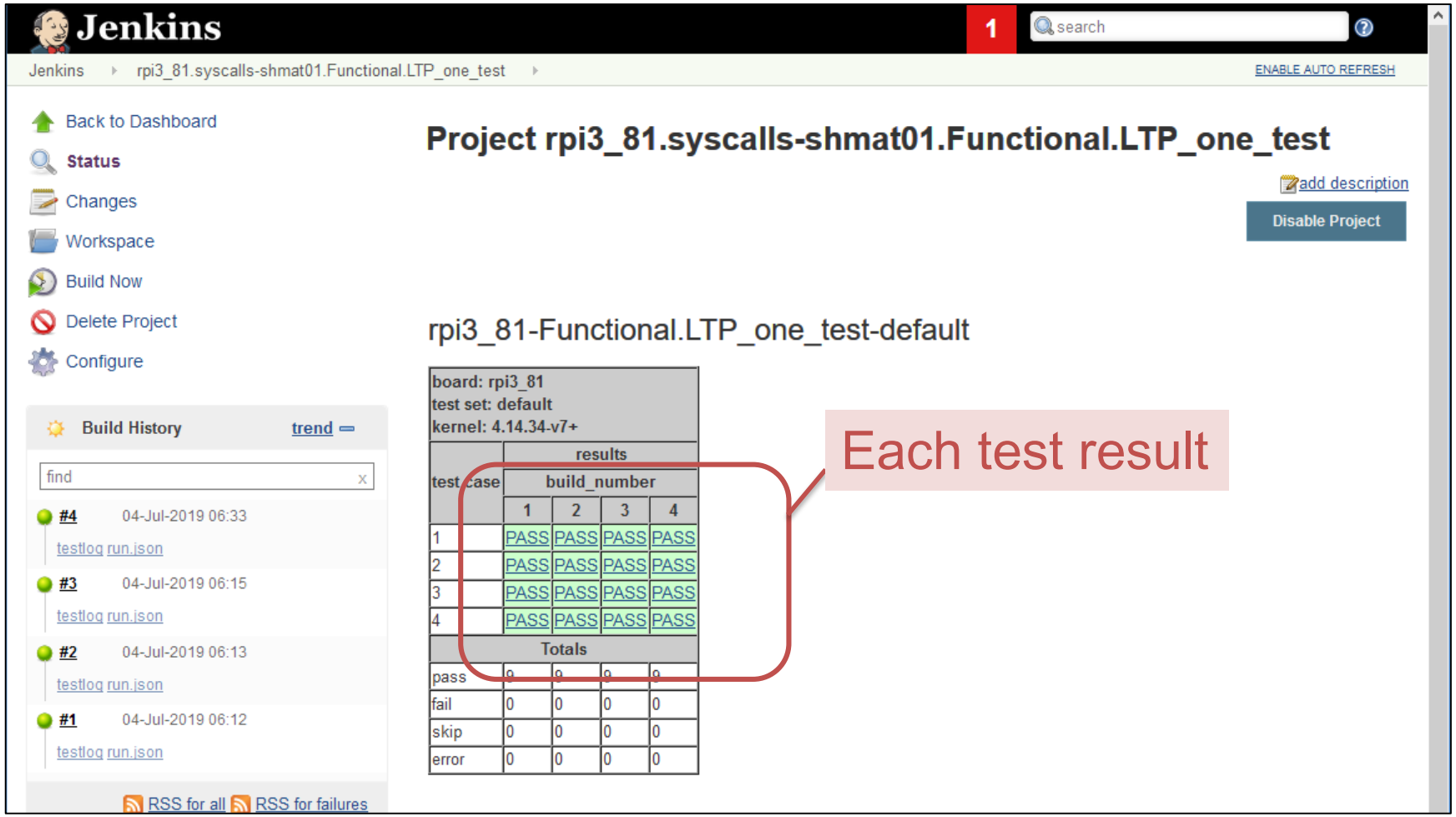
- Sample: shmat(), shmdt()

```
 {
    "testName": "Functional.LTP_one_test",
    "specs": {
      "default": {
        "TEST":"brk01"
      },
<SNIP>
      },
+     "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.

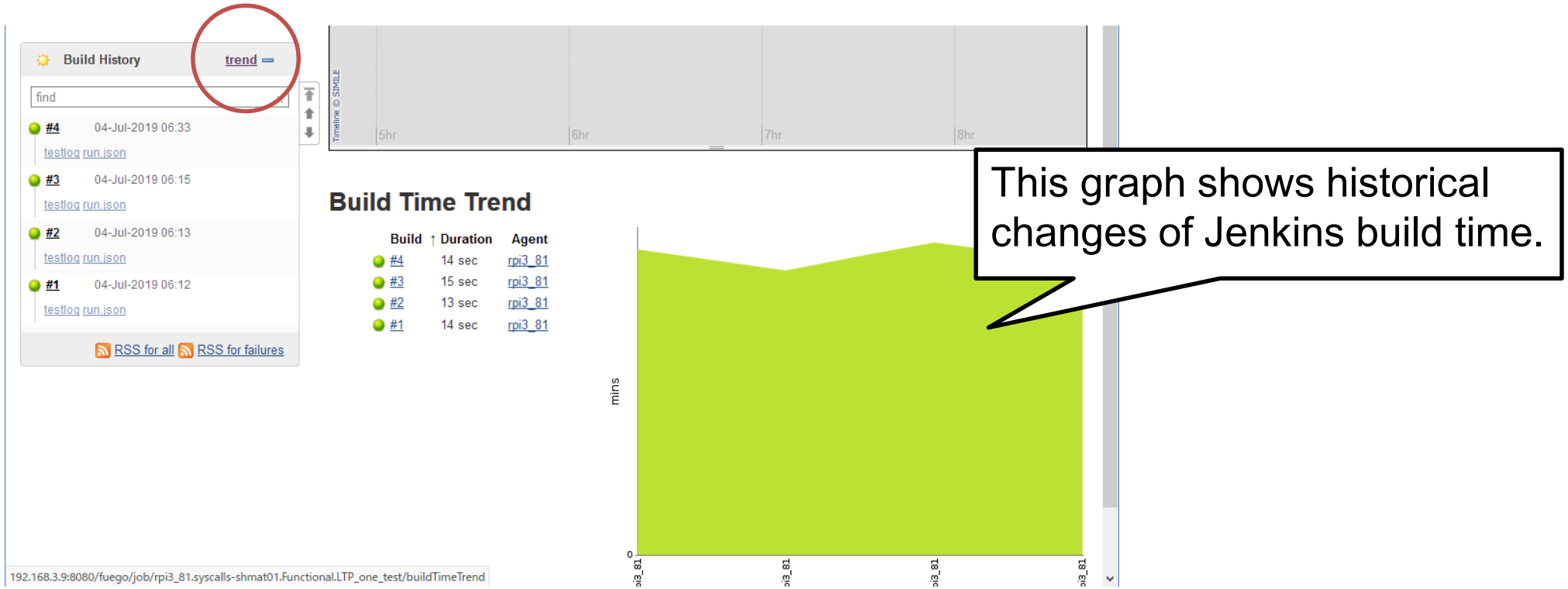
Alternative way

- Gap of test result of each 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.



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.

```
 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 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.

```
## 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
```

```
 function test_run {
    local bdir="$BOARD_TE
    local scenario=$FUNCT

    if [ -z "$scenario" ]
-       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 system call time

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

192.168.3.9:8080/fuego/userConte

192.168.3.9:8080/fuego/userContent/fuego.logs/Functional.LTP_one_t

```
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_RND
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)            = 0x76ef5000 <0.000040>
1148  shmat(360449, NULL, 0)            = 0x76efc000 <0.000026>
1148  shmat(360449, 0x76ef8000, 0)      = 0x76ef8000 <0.000036>
1148  shmat(360449, 0x76efbfff, SHM_RND) = 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.

```
 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

Injected Kernel

192.168.3.9:8080/fuego/userContent/fuego.logs/Functional.LTP_one_

```
tst_test.c:980: INFO: Timeout per run is 0h 05m 00s
shmat01.c:147: PASS: shma
shmat01.c:147: PASS: shma
shmat01.c:147: PASS: shmat() succeeded to attach unaligned address with SHM_RND
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--
1074  shmat(327682, NULL, 0)            = 0x76fd8000 <1.012371>
1074  shmat(360450, NULL, 0)            = 0x76fdf000 <1.039117>
1074  shmat(360450, 0x76fd8000, 0)      = 0x76fd8000 <1.020016>
1074  shmat(360450, 0x76fdbfff, SHM_RND) = 0x76fd8000 <1.037206>
1074  shmat(360450, 0x76fd8000, SHM_RDONLY) = 0x76fd8000 <1.037140>
```

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

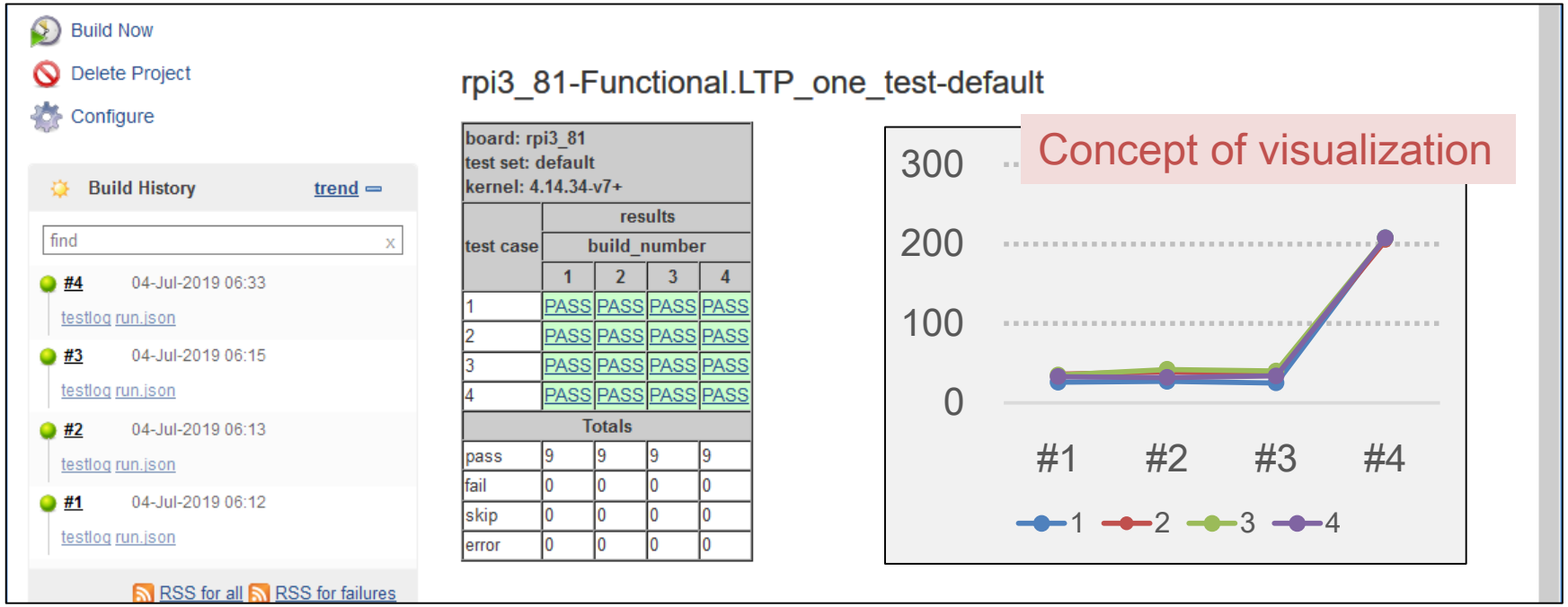
The time difference compared with the result in defalt kernel is roughly "1 second" each system call.

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!



MITSUBISHI ELECTRIC
Changes for the Better

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+