

# 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 MOTAII

- 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

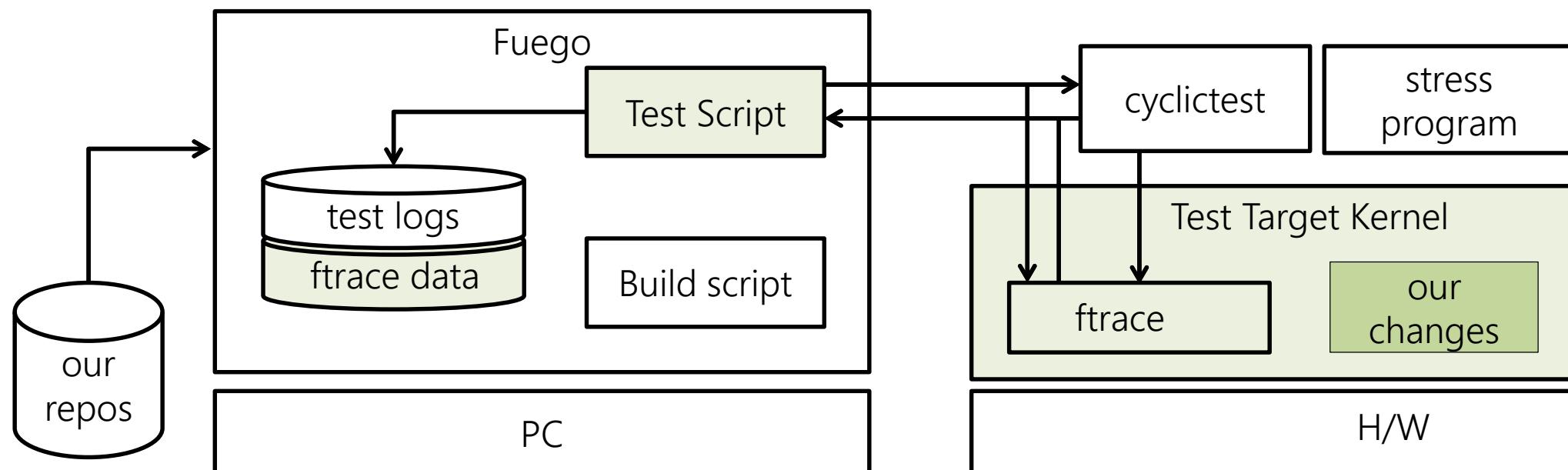


# Overview

---

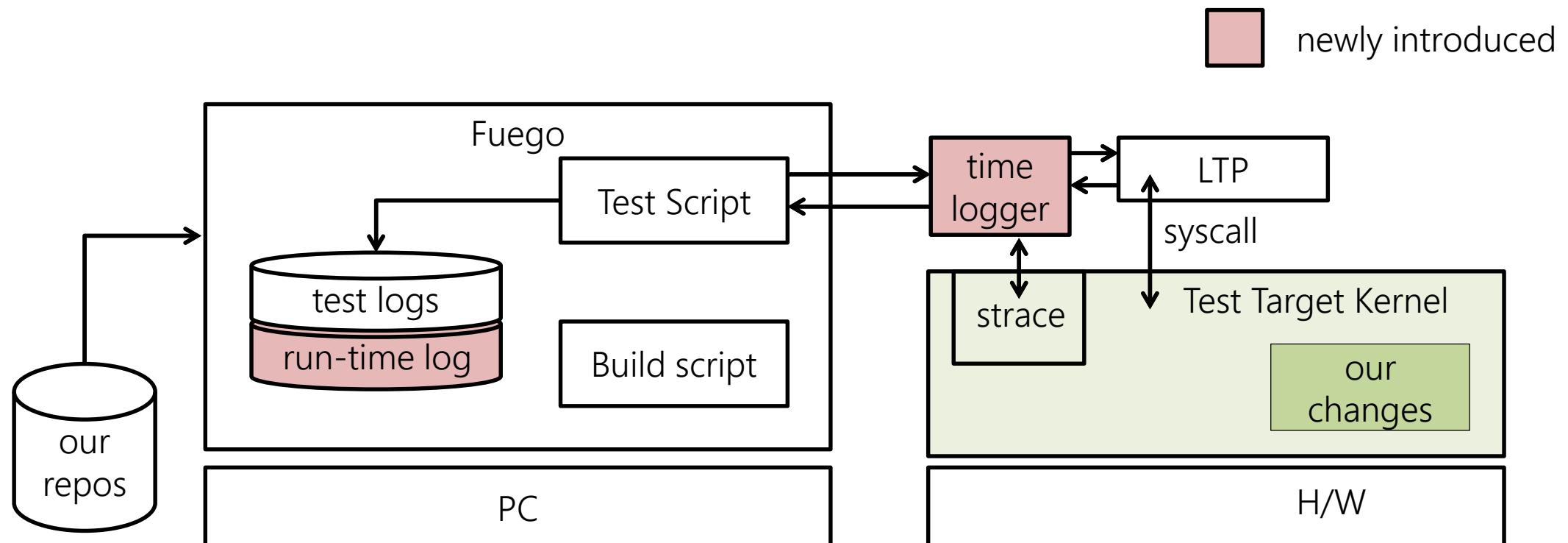
- Linux can be adapted to various embedded devices, even though they need a hard real-time response.
  
- We need tons of time to ensure adequate real-time performance.
  - Real-time applications need to satisfy timing constraints.
  - We have to avoid kernel changes which might cause long delays.

- 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



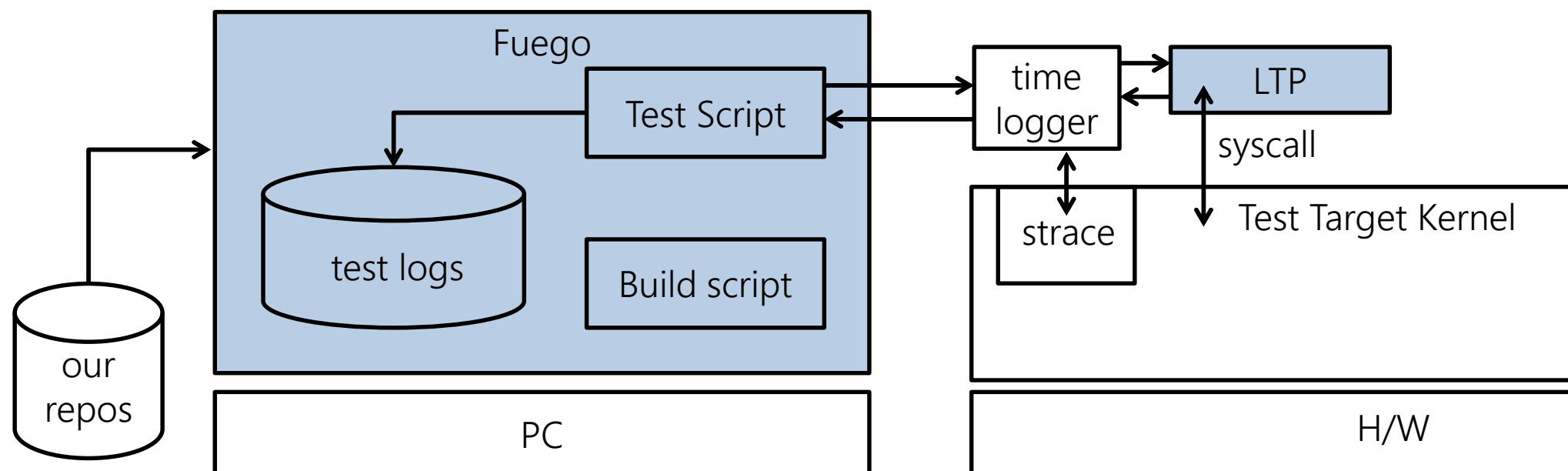
LTP

## ● Fuego:

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

## ● LTP: Linux Test Project

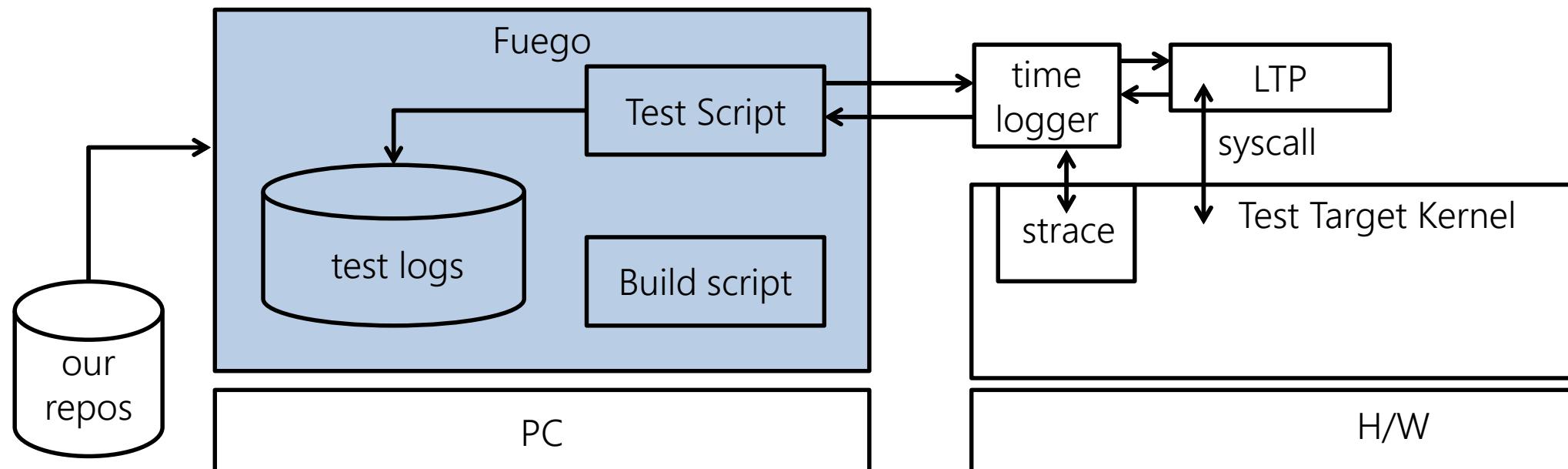
- regression and conformance tests designed to confirm the behavior of the Linux kernel and glibc
- <http://linux-test-project.github.io/>



# Fuego

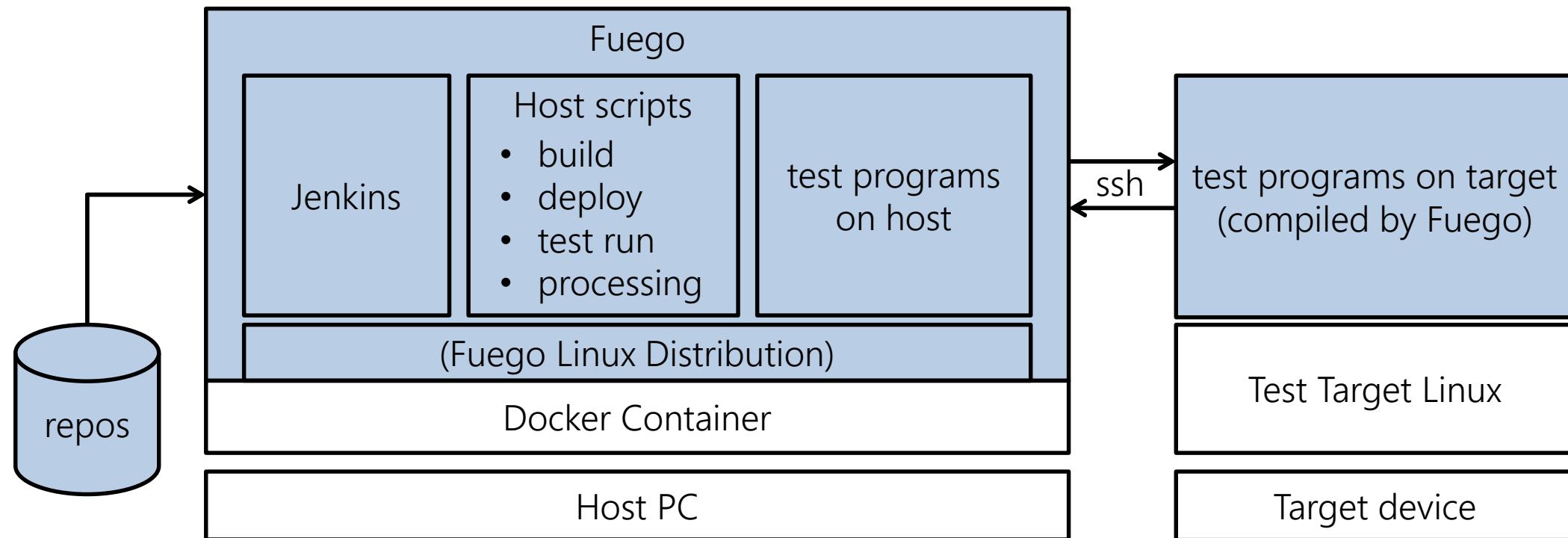
- Fuego is an automated test system
  - created by LTSI project, based on Jenkins.
  - OSS: anyone can use and contribute!
  - AGL-JTA: AGL chose Fuego as standard test environment.

JTA: Jenkins Test Automation

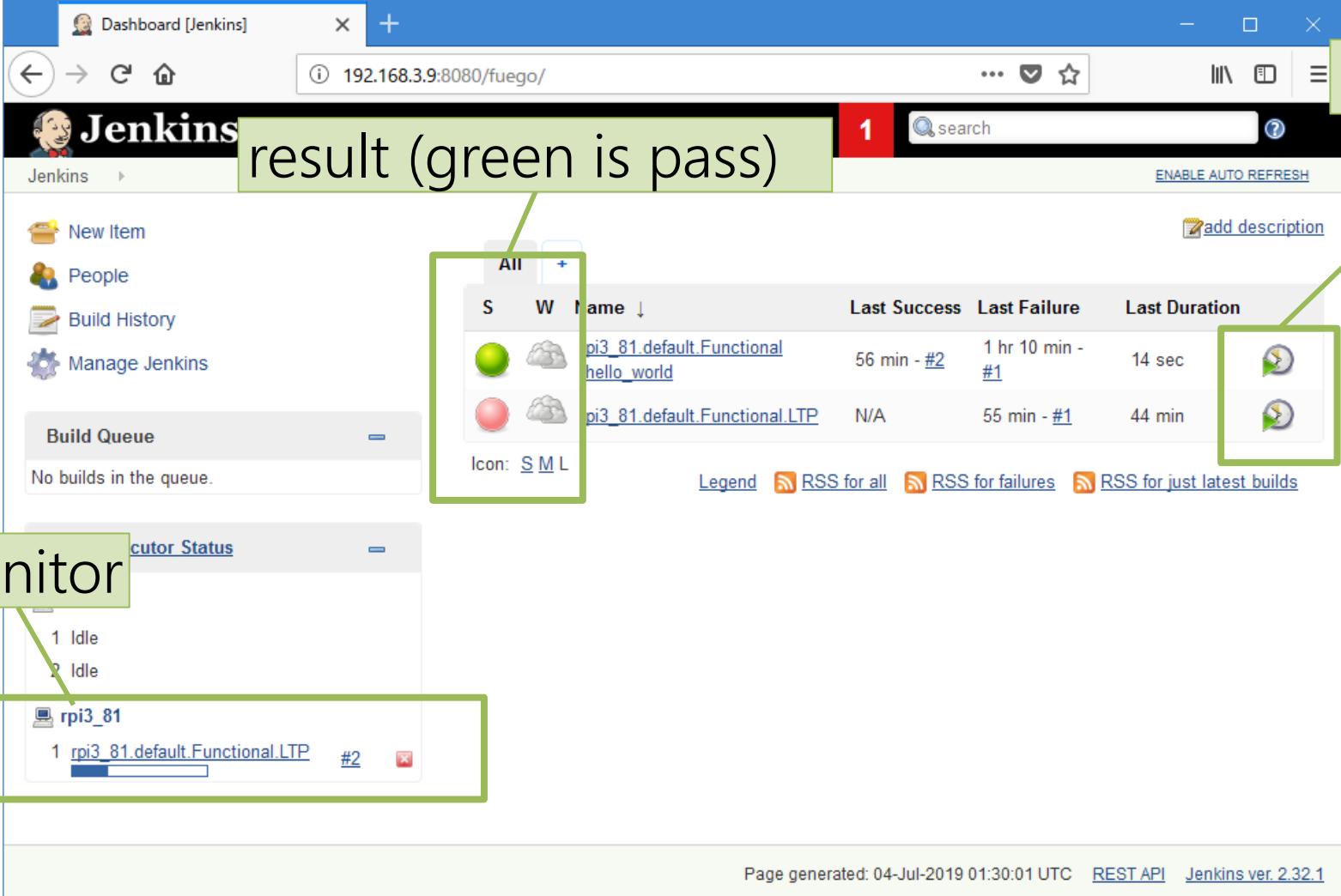


# Fuego

- 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 ...



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



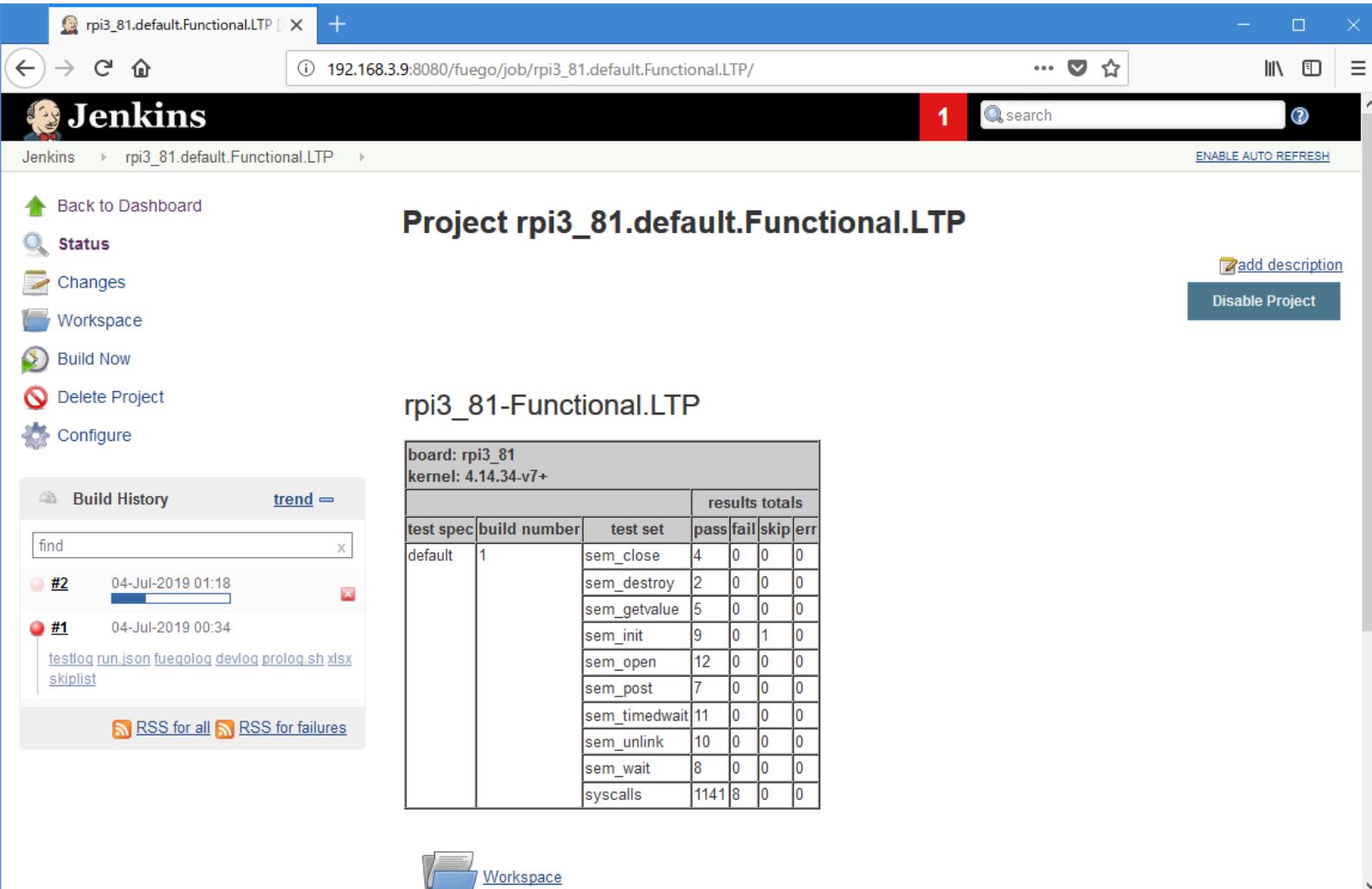
The screenshot shows the Jenkins dashboard at [192.168.3.9:8080/fuego/](http://192.168.3.9:8080/fuego/). The main area displays a table of build results for two jobs: 'pi3\_81.default.Functional\_hello\_world' and 'pi3\_81.default.Functional.LTP'. A green box labeled 'result (green is pass)' highlights the status column, which shows green icons for success. A red box labeled 'click to start' points to the 'Start' button for the 'Functional.LTP' job. Below the table, a 'Monitor Status' section shows two idle agents: 'rpi3\_81' and 'rpi3\_82'. A green box labeled 'monitor' encloses this section. At the bottom, a footer bar indicates the page was generated on 04-Jul-2019 01:30:01 UTC, provides links to the REST API and Jenkins version 2.32.1, and includes the Fuego logo.

S	W	Name	Last Success	Last Failure	Last Duration
		<a href="#">pi3_81.default.Functional_hello_world</a>	56 min - #2	1 hr 10 min - #1	14 sec
		<a href="#">pi3_81.default.Functional.LTP</a>	N/A	55 min - #1	44 min

Legend: RSS for all RSS for failures RSS for just latest builds

Page generated: 04-Jul-2019 01:30:01 UTC [REST API](#) [Jenkins ver. 2.32.1](#)

● You can also check test results on Jenkins.

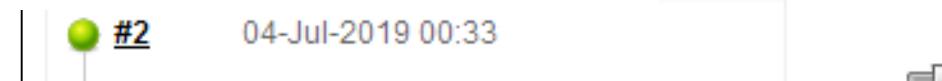


The screenshot shows a Jenkins project page for 'rpi3\_81.default.Functional.LTP'. The left sidebar includes links for Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, and Configure. The main content area displays the project name 'Project rpi3\_81.default.Functional.LTP' and a summary table for 'rpi3\_81-Functional.LTP'. The table details test results for various system calls and syscalls across two builds (#1 and #2). The table has columns for test spec, build number, test set, pass, fail, skip, and err counts.

test spec	build number	test set	results totals			
			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

## ● Functional test

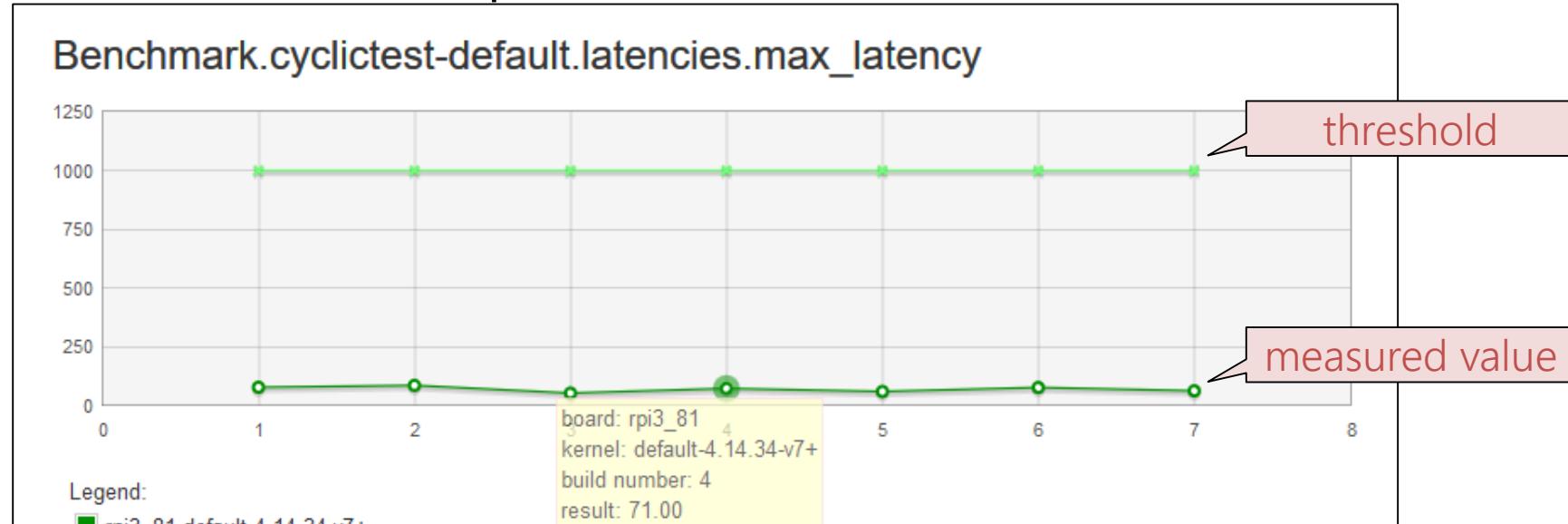
- Test result: judged by return value
  - Historical results: "PASS" or "FAIL"
- 102 testsuits as functional tests:
  - LTP, LTP\_one\_test, OpenSSL, aiostress, busybox, bzip2, glibc, hello\_world, iptables, kernel\_build, kselftest, linus\_stress, netperf, ptest, stress, tar, year2038, ...



rpi3_81-Functional.hello_world					
board: rpi3_81					
test set: default					
kernel: 4.14.34-v7+					
results		build_number			
test case		1	2	3	4
hello_world		FAIL	PASS	PASS	PASS
Totals					
pass		0	1	1	1
fail		1	0	0	0
skip		0	0	0	0
error		0	0	0	0

## ● Benchmark test

- Test result: "PASS" if measured value < threshold
  - Historical results: measured value
- 42 testsuits as benchmark tests:
  - Dhrystone, IOzone, Interbench, Whetstone, bonnie, cyclictest, dbench4, deadlinetest, hackbench, iperf, Imbench2, nbench\_byte, netperf, svsematest, x11perf ...



# LTP: Linux Test Project

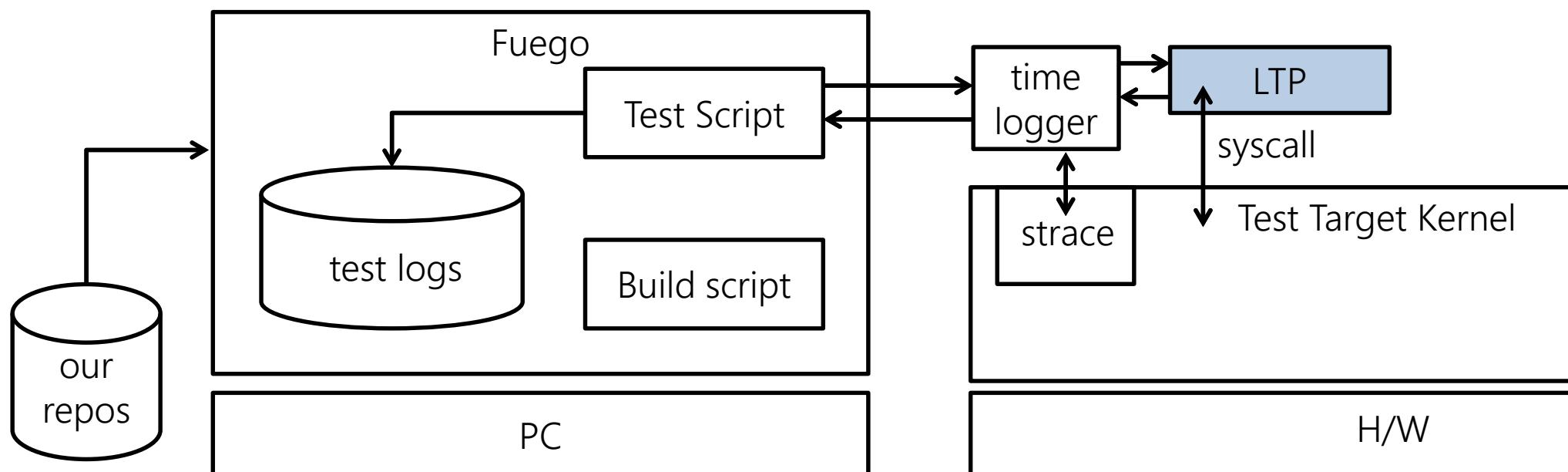


- A huge collection of tests for Linux
  - systemcalls, semaphore, POSIX, ...
- Difficult to understand test results
  - Tester has to know what to ignore, and why
    - depend on system or kernel configurations.
  - In a regression test, tester check the gaps between previous and current results.

# 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

- 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....?

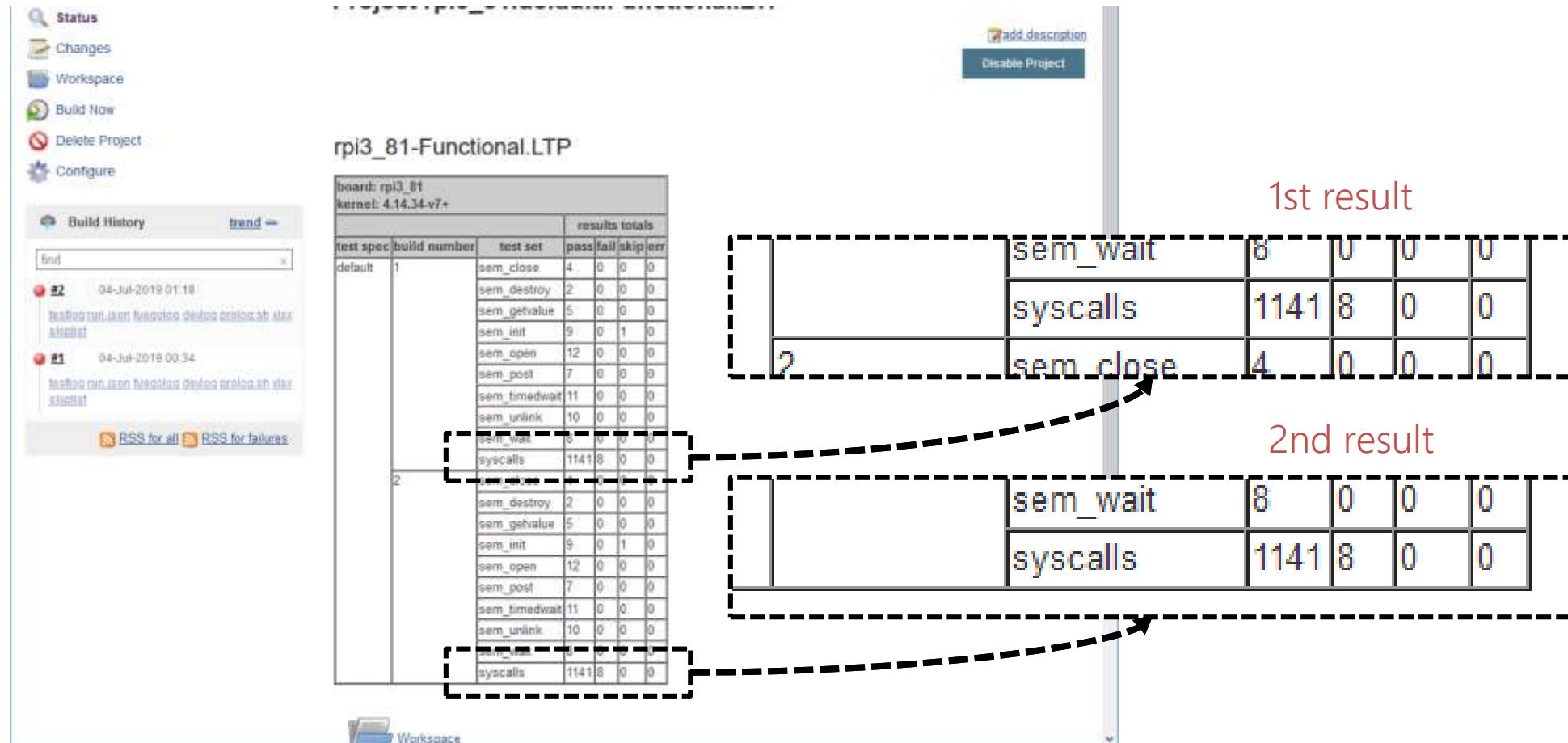
rp3\_81-Functional.LTP

board: rp3\_81  
kernel: 4.14.34-v7+

test spec	build number	test set	pass	fail	skip	err	results totals
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	
2	2	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	

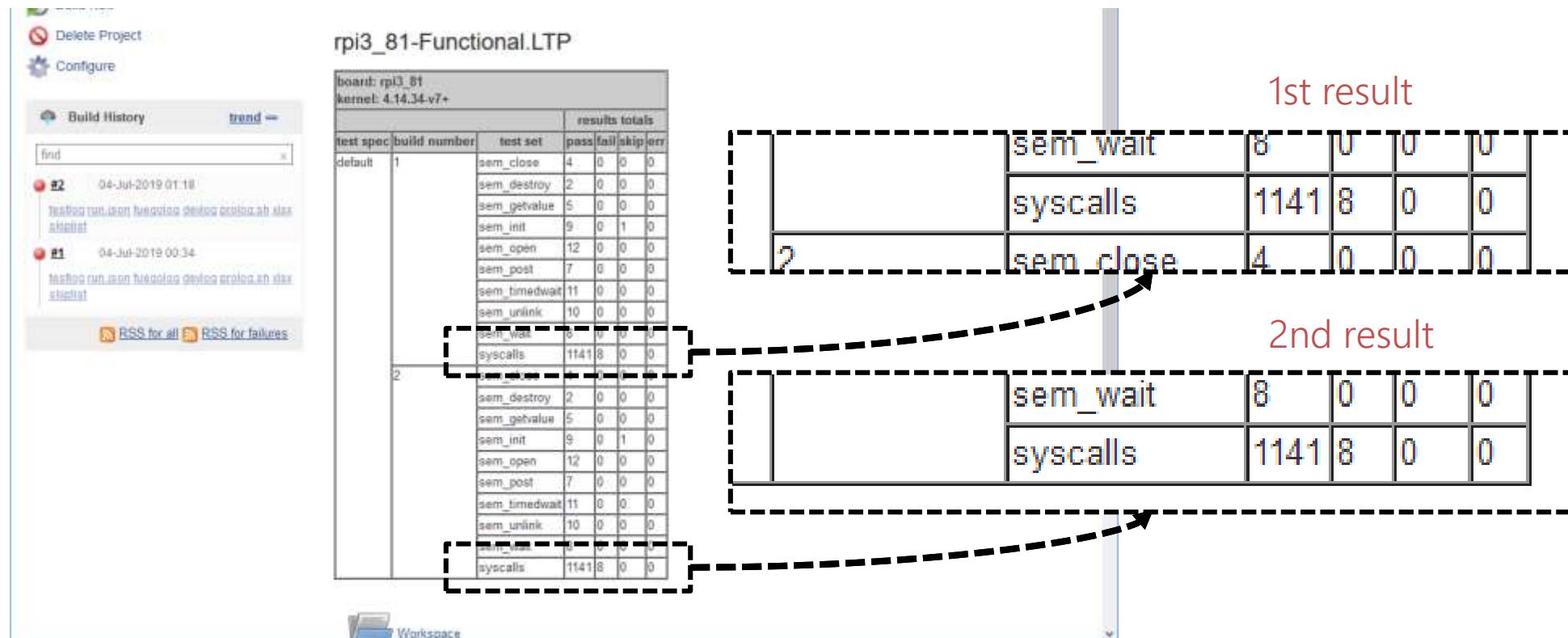
1st result

2nd result



# 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

```
# 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
```

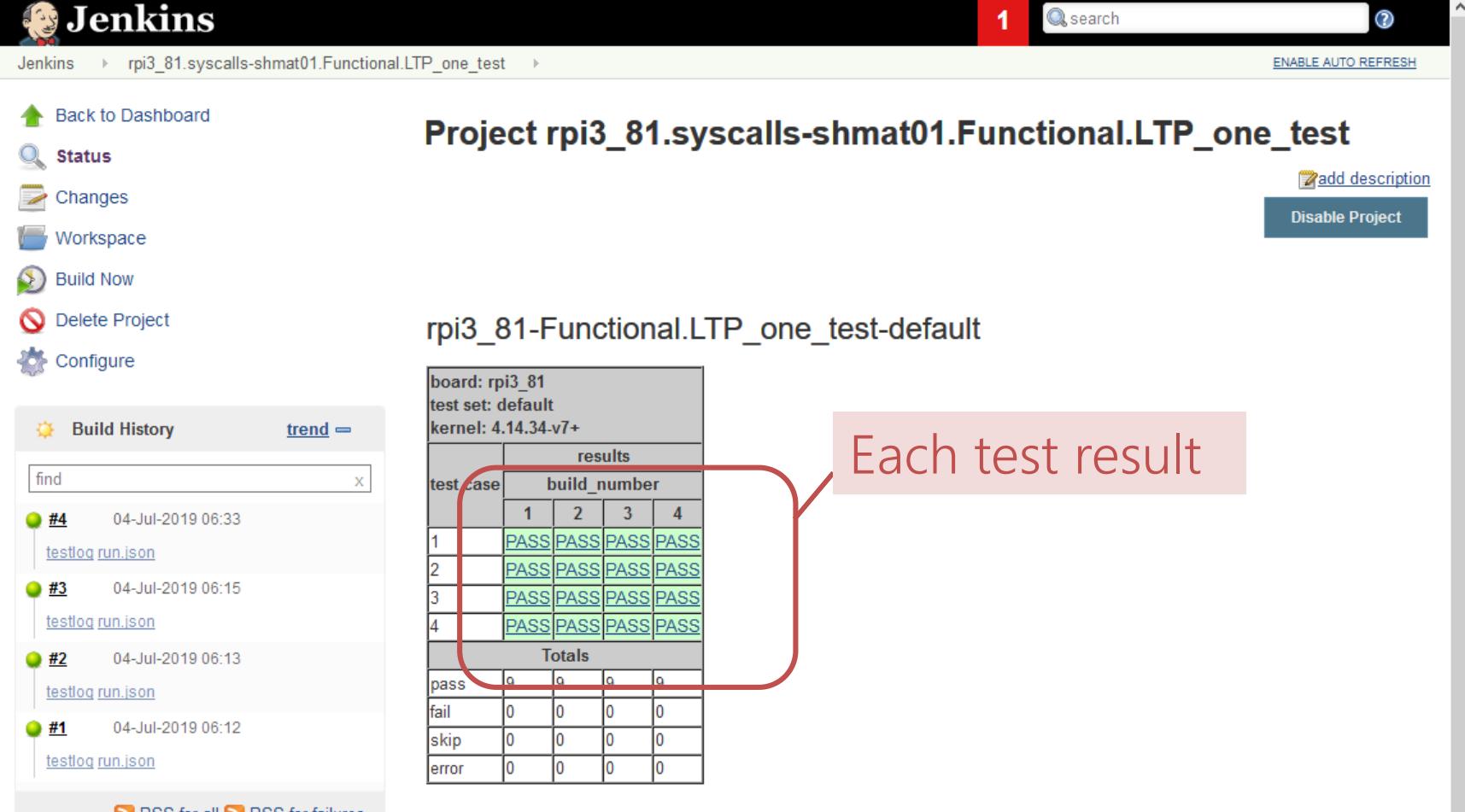
- Sample: shmat(), shmdt()

```
{
  "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 syscall become clear.



Jenkins

Project rpi3\_81.syscalls-shmat01.Functional.LTP\_one\_test

1 search ENABLE AUTO REFRESH

Back to Dashboard Status Changes Workspace Build Now Delete Project Configure

Build History trend → find

#4 04-Jul-2019 06:33 testlog run.json

#3 04-Jul-2019 06:15 testlog run.json

#2 04-Jul-2019 06:13 testlog run.json

#1 04-Jul-2019 06:12 testlog run.json

rss RSS for all rss RSS for failures

board: rpi3\_81  
test set: default  
kernel: 4.14.34-v7+

results

test_case	build_number	1	2	3	4
1	PASS	PASS	PASS	PASS	
2	PASS	PASS	PASS	PASS	
3	PASS	PASS	PASS	PASS	
4	PASS	PASS	PASS	PASS	

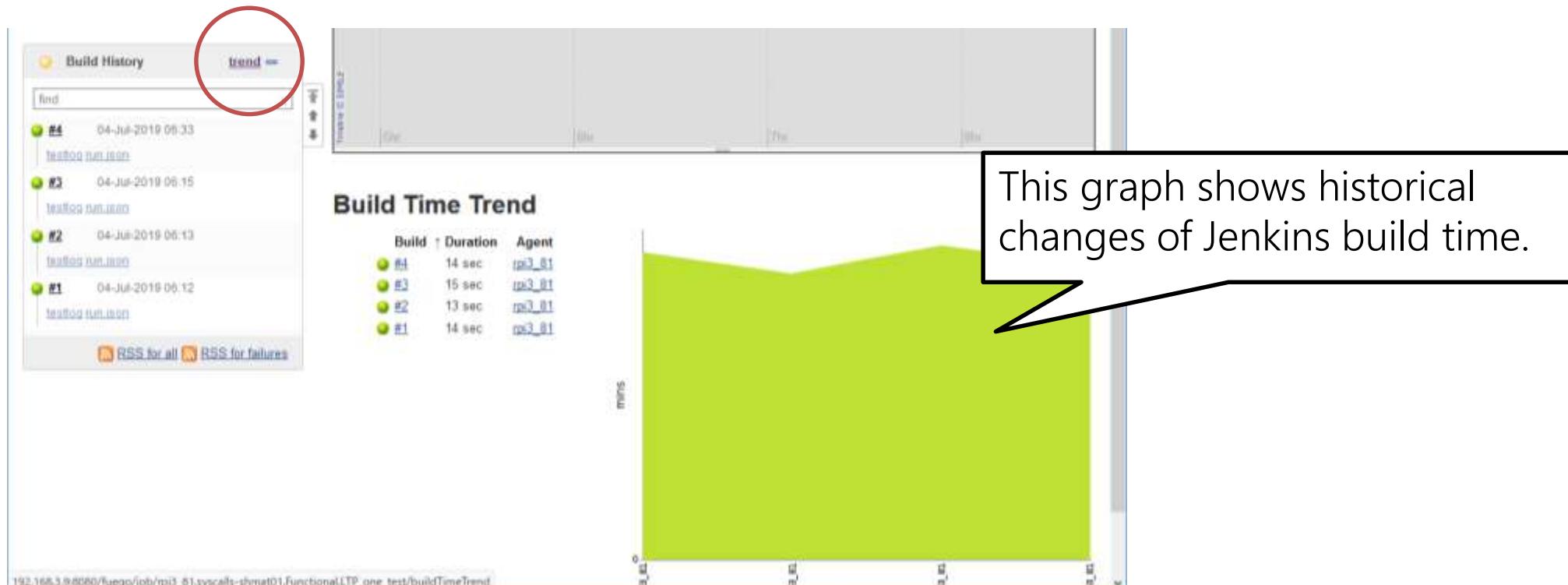
Totals

	pass	fail	skip	error
pass	0	0	0	0
fail	0	0	0	0
skip	0	0	0	0
error	0	0	0	0

Each test result

# 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.



# 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.

```
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.

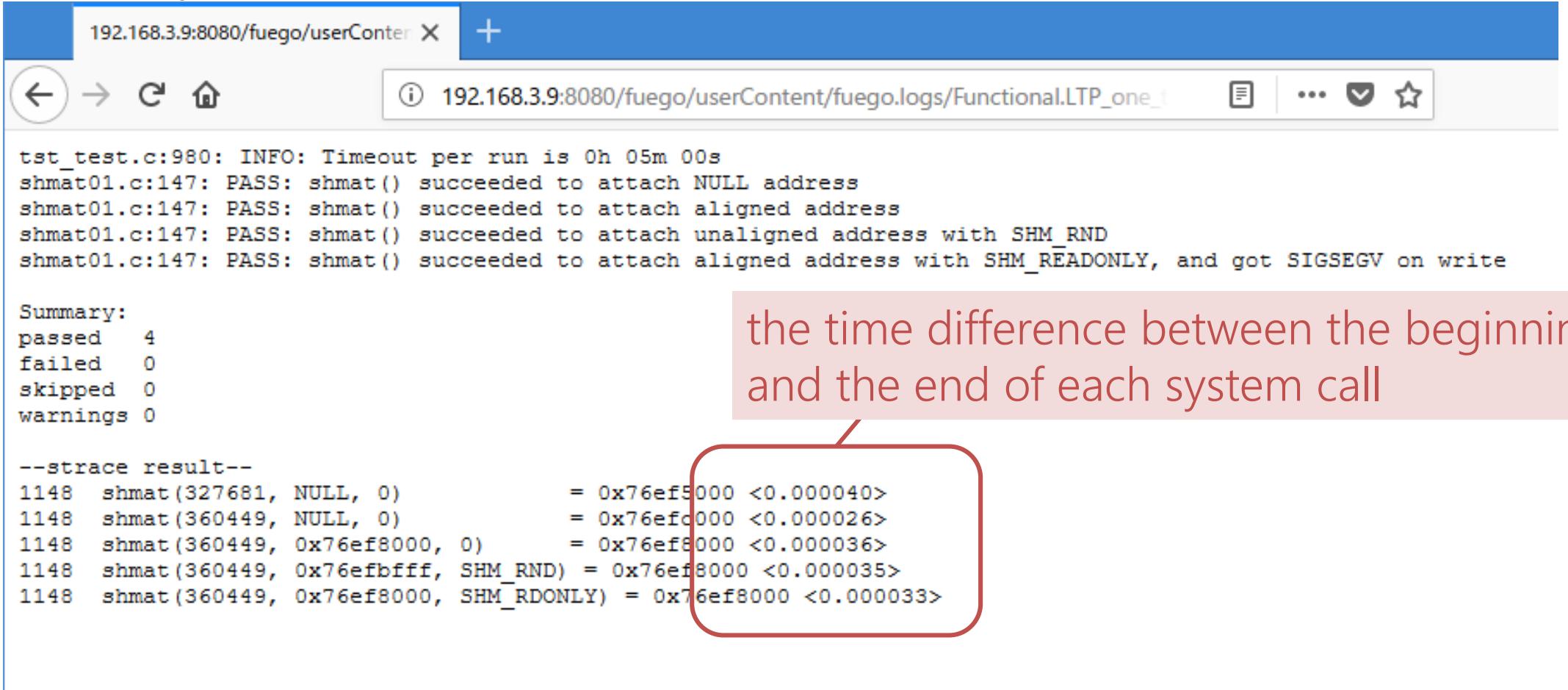
- Fuego provides a script running on the target, in `fuego_test.sh`.
- measure the execution time of the test process as below.

```
## 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--";
grep $SYSCALL $OUTPUT
exit $RETVAL

function test_run {
    local bdir="$BOARD_TES
    local scenario=$FUNCTIONAL_LTP_ONE_TEST_ARGS
    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

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



192.168.3.9:8080/fuego/userContent X +  
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\_RDONLY, 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, 0x76efbff, 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

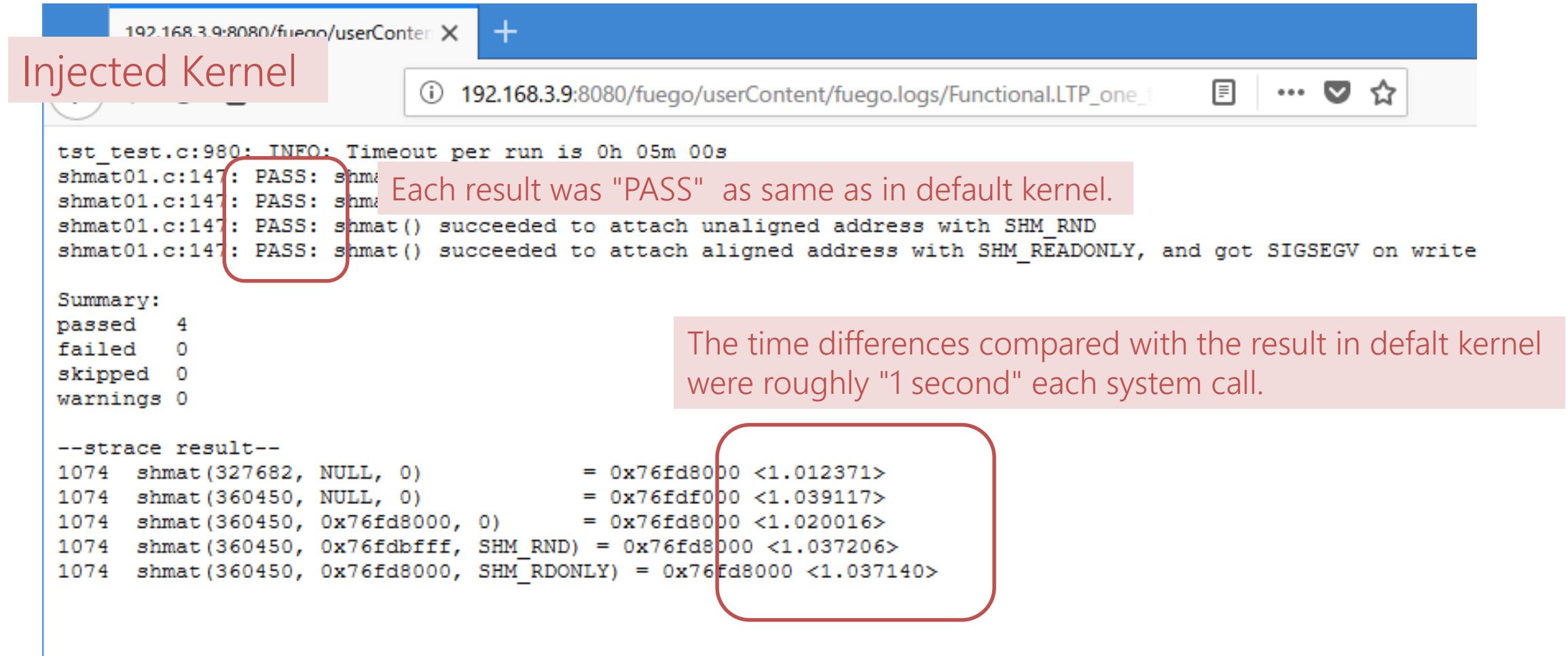
- Inject 1sec waiting patch to "shmat()" interface in kernel.
- Test and check whether the result include >1sec delay.

```
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



The screenshot shows a web browser window with two tabs. The active tab is titled "Injected Kernel" and displays a terminal-like output of Fuego test results. The output includes logs from "tst\_test.c" and "shmat01.c", a summary of test results, and a strace output. A red box highlights the "shmat01.c" log entries, and another red box highlights the strace output. Red annotations provide context for these highlighted sections.

192.168.3.9:8080/fuego/userContent X +

Injected Kernel

i 192.168.3.9:8080/fuego/userContent/fuego.logs/Functional.LTP\_one\_t

tst\_test.c:980: TINFO: Timeout per run is 0h 05m 00s  
shmat01.c:147: PASS: shma... Each result was "PASS" as same as in default kernel.  
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>

The time differences compared with the result in defalt 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.

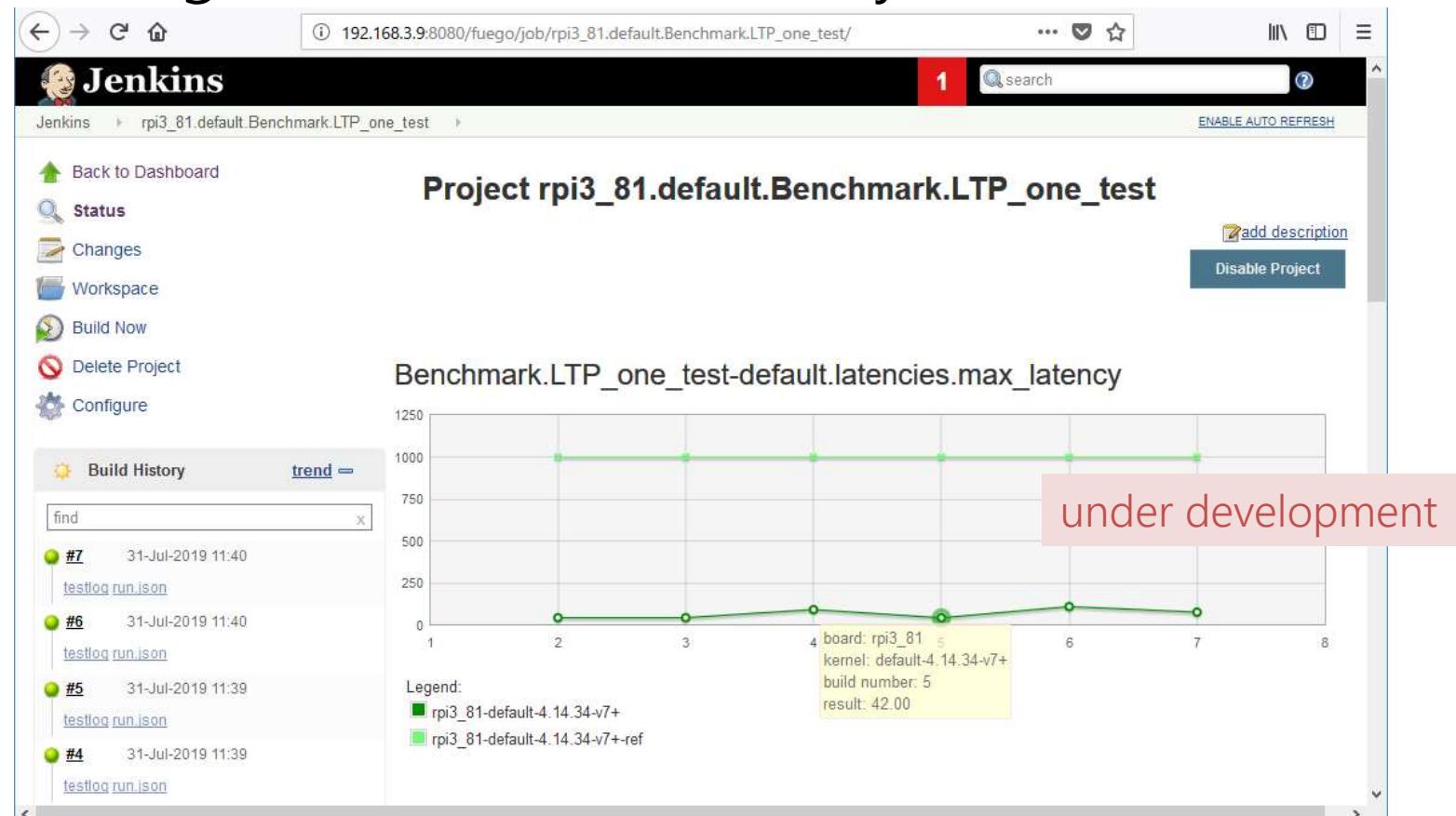
# Conclusion and Future work

## ● 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

- FUEGO

- <http://fuegotest.org/>

- LTP: Linux Test Project

- <http://linux-test-project.github.io/>

- strace

- <https://strace.io/>

- LTSI Project

- <https://ltsi.linuxfoundation.org/>

- AGL Test framework: AGL-JTA

- <https://wiki.automotivelinux.org/agl-jta>

- 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+

## ● Benchmark: 42

- Dhrystone, GLMark, IOzone, Interbench, Java, OpenSSL, Stream, Whetstone, aim7, backfire, blobsallad, bonnie, cyclictest, 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\_Dbus, 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, iutils, ipv6connect, jpeg, kernel\_build, kmod, ksselftest, 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)

- 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 slectionwithrt
- # ftc add-jobs -b yourboard -t Functional.LTP -s ltplite
- # 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