Introduction of AGL's approach to Fuego

Feb 21, 2018
AGL ALL Member Meeting, Tokyo

Kyohei Oki
Denso TEN
Liu Wenlong
Nanjing Fujitsu Nanda Software Tech. Co., Ltd(FNST)

DENSO TEN Limited
Self Introduction

- Kyohei Oki(kyohei.oki@jp.fujitsu.com)

- Automotive Software Engineer (2015 ~ )

- AGL CIAT Member (2016 ~ )
  - engaging on CIAT for AGL
  - especially Fuego
Agenda

- **What** is Fuego
- **How** to use Fuego
- **Introduction** to Fuego test results report
- **Introduction** to Fuego LTP network test
- Fuego Roadmap
- **Our Future Work**
What is Fuego

- A test system specifically designed for embedded Linux testing. (Refer to: http://fuegotest.org/)

- Automated test framework for Kernel’s LTSI

- Features
  - source code management
  - test code build & deploy
  - tests executing & results report

- Advantages
  - highly customizable & unified test outputs
  - flexible test configuration & running tests in batches
  - board setup is simple & flexible
  - 50 pre-packaged tests & do tests with command lines
What is Fuego

Have a look at the interior of Fuego

**Host Machine:**
- Container build system
- 2 repositories:
  - fuego (e.g. docker build scripts)
  - fuego-core

**Docker container:**
- Jenkins
- Test programs
- Scripts (Shell/Python)

**Volume Mount**
- pre-packaged tests
- builds & logs
- configs

**Schedule your jobs**

**Start**

**Web control interface**

**Check the result**

**Run**

**SSH Connect**

**Get results**
What is Fuego

Have a look at Fuego work flow

- frontend (jenkins) call backend (script engine) to run test cases
- backend do the works
  - Build test cases for the target board
  - Run test cases on target board
  - Parse test logs and show the summary to users
- frontend get the result/log from the backend

![Diagram of Fuego workflow]

Test code
Build
Deploy

jenkins (frontend)
script engine (backend)

execute test script
run test on board

return result
return logs

R-Car M3
How to use Fuego

1. Build the Fuego container

2. Add board file to Fuego

3. Add a yocto project toolchain

4. Add board to Jenkins Interface

5. Add some jobs to Fuego

6. Run tests with Fuego

Each step will be explained in detail
How to use Fuego

1. Build the Fuego container

1.1 download the latest code from Fuego repository

$ git clone https://bitbucket.org/tbird20d/fuego.git
$ git clone https://bitbucket.org/tbird20d/fuego-core.git

1.2 create the docker image

$ cd fuego ; ./install.sh
$ fuego-host-scripts/docker-create-container.sh
$ fuego-host-scripts/docker-start-container.sh

1.3 Now, the container build step is done

# We can access to the web control page with the URL below.
$ firefox http://localhost:8080/fuego

No job and node were created. Setup Fuego next step~
How to use Fuego

2. Add board file to Fuego

Customize your own board file (example with R-Car M3)

# cd fuego-ro/boards
$ cp template-dev.board m3ulcb.board
# change the value according to your own env.
$ cat m3ulcb.board

......

IPADDR="192.168.10.6" IP address or hostname of the target machine.
LOGIN="root"
BOARD_TESTDIR="/home/a"
PASSWORD=""
#SSH_KEY="path/to/id_rsa"
PLATFORM="m3ulcb"
TRANSPORT="ssh"
ARCHITECTURE="arm64"
SATA_DEV="/dev/sdb1"
SATA_MP="/mnt/sata"
USB_DEV="/dev/sda1"
USB_MP="/mnt/usb"
......

Next, we need to add a yocto toolchain to Fuego.
3. **Add a yocto project toolchain** (example with R-Car M3)

1. **generate toolchain**

   ```bash
   # build yocto sdk
   $ bitbake agl-demo-platform -c do_populate_sdk
   # toolchain will be generated at "./tmp/deploy/sdk/
   ```

2. **install the toolchain inside the container**

   ```bash
   $ ./poky-agl-glibc-x86_64-meta-toolchain-aarch64-toolchain-*.sh
   # install the toolchain to "/opt/poky-agl/m3ulcb" in container.
   ```

3. **Create a \${PLATFORM}-tools.sh file for the toolchain**

   ```bash
   $ cat fuego/fuego-ro/toolchains/m3ulcb-tools.sh
   # this script should be sourced by \${FUEGO_RO}/toolchains/tools.sh
   SDKROOT=/opt/poky-agl/m3ulcb/sysroots/aarch64-agl-linux
   # the Yocto project environment setup script changes PATH so that python uses
   # libs from sysroot, which is not what we want, so save the original path
   # and use it later
   ORIG_PATH=$PATH
   PREFIX=aarch64-agl-linux
   #export_tools
   
   source /opt/poky-agl/m3ulcb/environment-setup-aarch64-agl-linux
   HOST=aarch64-agl-linux
   ```

   Note: we can also install debian-based toolchain with “install_cross_toolchain.sh”.
How to use Fuego

4. Add board to Jenkins Interface

Before, how to add boards to AGL-JTA? (example with R-Car M3)

Steps:
1. Click “New Node”;
2. Select node name “m3ulcb”;
3. Copy Existing Node input: “porter”;
4. Click “OK” button;
5. Change “value” to “board/m3ulcb.board”;
6. Click “Save” button.

Now, we use ftc tool to add board in Fuego.
How to use Fuego

We can use ftc tool to add board in Fuego.

What is ftc tool:
- FTC is the "fuego test control" tool.
- A command line tool used to perform various functions in the fuego system.

What we can do with ftc tool:(example with R-Car M3)

Use ftc tool to add board(called “node” in Jenkins interface)

$ ftc add-nodes m3ulcb

Add specified jobs for specified target

$ ftc add-jobs -b m3ulcb -t Functional.bc -s bc-add

Or

$ ftc add-jobs -b m3ulcb -p testplan_agl

Run tests

$ ftc run-test -b m3ulcb -t Functional.bc -p dr

Those ftc commands above will be used in next steps.
How to use Fuego

Now, add node to Jenkins interface as below

```bash
# Now, we use ftc tool to add a node to Jenkins interface
$ ftc add-nodes m3ulcb
```

Then, we will find that a node named “m3ulcb” will be created.

Run “ftc add-nodes m3ulcb”

Node “m3ulcb” was generated.
4. Add board to Jenkins Interface

Before, how to add boards to AGL-JTA? (example with R-Car M3)

Steps:
1. Click “New Node”;
2. Select node name “m3ulcb”;
3. Copy Existing Node input: “porter”;
4. Click “OK” button;
5. Change “value” to “board/m3ulcb.board”;
6. Click “Save” button.

Now, we use ftc tool to add board in Fuego.

No complicated steps!!
How to use Fuego

5. Add some jobs to Fuego (example with R-Car M3)
Add some tests to the Jenkins interface.

```
# add a job for demo
$ ftc add-jobs -b m3ulcb -t Functional.bc -s bc-add

# add a batch agl jobs
$ ftc add-jobs -b m3ulcb -p testplan_agl
```

Check that all jobs have been generated as below.

“Functional.bc” and tests in “testplan_agl.json” were added.
6. Run tests with Fuego (example with Functional.bc on R-Car M3)

6.1 Run tests with ftc tool,

```bash
# run job "m3ulcb.bc-add.Functional.bc"
$ ftc run-test -b m3ulcb -t Functional.bc -s bc-add

Notice: non-Jenkins test request detected
Running test 'Functional.bc' on board 'm3ulcb' using spec 'bc-mult'
!!! >>> Ready to run test! <<< !!!
DEBUG: python var build_number=1
Running remotely on 'm3ulcb' in workspace /fuego-rw/buildzone
DEBUG: python var command=timeout --signal=9 30m /bin/bash

chart_config.json not available or is wrong format, using default values
Writing chart data to /fuego-rw/logs/Functional.bc/flot_chart_data.json
+ rc=0
+ '[' 0 -eq 1 ']
+ echo 'Fuego: requested test phases complete!'
Fuego: requested test phases complete!
+ exit 0
```

About “ftc run-test”:
- the same as running tests with Jenkins interface.
- useful to debug test during tests development.
How to use Fuego

6.2 Run tests with Jenkins interface

Advantages:
- Easy to check the output of tests and Jenkins.
- Clear description for Jenkins related errors.

But for tests that have heavy logs, it will be hard for us to locate the error output easily.
So, let’s make some changes.
Introduction to Fuego test results report

About the test results, what we want is as below. How should we?

Advantage:
- Compare the results of different build.

Disadvantage:
- Check the error log difficulty
We need to search error points.
Introduction to Fuego test results report

About the test results, what we want is as below.

How should we?

Advantages:
- Compare the results of different build.
- Check the error log easily.
Introduction to Fuego test results report

1. Current Fuego test results report flow

- Get the test outputs from the target
- Generate “run.json”
  ① test results for each build number
- Generate “results.json”
  ② test results for each job
- Generate “flat_plot_data.txt”
  ③ test results for each case
- Generate “flot_chart_data.json”
- Final test results report, - Test status for each case
- ④ HTML test results report
Introduction to Fuego test results report

**run.json:** ① test results for each build

```
"duration_ms": 7084,
"fuego_version": "v1.2.1",
"name": "Functional.bc",
"schema_version": "1.0",
"status": "PASS",
"test_sets": [
{
"name": "default",
"status": "PASS",
"test_cases": [
{
"name": "bc",
"status": "PASS"
}
]
},
```

**results.json:** ② test results for each job

```
"m3ulcb-bc-add-4.9.0-yocto-standard-default-bc": {
  "board": "m3ulcb",
  "build_number": [
    "1",
    "2"
  ],
  "duration_ms": [
    20198,
    14977
  ],
  "status": [
    "PASS",
    "PASS"
  ],
```

**flat_plot_data.txt:** ③ test results for each case

```
m3ulcb Functional.bc default 1 2018-01-15T10:00:32+0000 4.9.0-yocto-standard default.bc PASS PASS
m3ulcb Functional.bc default 2 2018-01-16T07:56:51+0000 4.9.0-yocto-standard default.bc PASS PASS
```

**flot_chart_data.json:** ④ HTML test results report

```
"chart_type": "testcase_table",
"data": "<table border="1" cellspacing="0"><tr style="background-color:#cccccc">"```
Introduction to Fuego test results report

2. Split the outputs
We have added the following processing in parser.py of some tests in Fuego.

```python
$ cat /fuego-core/engine/tests/Functional.fuego_tguid_check/parser.py
......
import common as plib

results = {}

regex_string = '^(ok|not ok) (¥d+)$'
matches = plib.parse_log(regex_string)
results = collections.OrderedDict()

for m in matches:
    print("DEBUG: in parser.py: m=%s" % str(m))
    status = m[0]
    test_num = m[1]
    test_id = m[2].replace(" ", ".")
    results[test_id] = 'PASS' if status == 'ok' else 'FAIL'

# split the output for each testcase
plib.split_output_per_testcase(regex_string, results)

sys.exit(plib.process(results))
......
```

Use ordered dictionary

Added generic function: “split_output_per_testcase”, that help us to split the test results.
3. Add log links

We have added the following processing in Fuego.

```python
$ cat /fuego-core/engine/scripts/parser/prepare_chart_data.py

......

def make_testcase_table(test_name, chart_config, entries):
......

tguid_testset = ".".join(tguid_parts[:-1])
tguid_testcase = tguid_parts[-1]

# get the name that contains board, spec, build number. E.g.
log_bts_name = '%s.%s.%s.%s' %
    (entry.board, entry.spec, str(entry.build_number), str(entry.build_number))

# separated log files, e.g. /Functional.LTP/ubuntu.math.7.7/result/math/outputs/abs01.log
log_file = '/userContent/fuego.logs/%s/%s/result/%s/outputs/%s.log' %
    (entry.testname, log_bts_name, tguid_testset, tguid_testcase)

# testlog files, e.g. /Functional.croco/porter.default.${BUILD_NUMBER}.${BUILD_ID}/testlog.txt
testlog_file = '/userContent/fuego.logs/%s/%s/testlog.txt' % (entry.testname, log_bts_name)

......

# check if the separated log path exist
if os.path.exists(jenkins_root_path + log_file):
    entry.result = '<a href="%s">%s</a>' % (jenkins_web_prefix, log_file, entry.result)
elif os.path.exists(jenkins_root_path + testlog_file):
    entry.result = '<a href="%s">%s</a>' % (jenkins_web_prefix, testlog_file, entry.result)
......
```

Find the separated log files for each testcase.

Add link to those separated log files.
Introduction to Fuego LTP test

1. Check the current LTP tests

About LTP test in Fuego, we can use 4 different usage scenarios

- 1) fuego builds, deploys and runs LTP on target
- 2) fuego runs an existing installation of LTP on target
- 3) fuego builds and deploys LTP, to a special location of the target
- 4) fuego builds LTP, but deploy is left to user

We have used the scenarios 2) to do all LTP tests on different AGL distro

However, we find the following tests cannot be tested properly

- LTP network related tests. (E.g. net_stress.*)

Now, we can do network related tests and we will inform you of the test result of AGL
2. Setup the test server for LTP network tests

**Physical Topology:**
- These tests require two machines.
- Each machine needs to have 2 or more interfaces.

**Software setup on Remote Host:**
- ssh/rsh between Remote Host and Local Host
- http/ftp service
- Install LTP and add those test commands to $PATH

Details
- Refer to https://github.com/linux-test-project/ltp/blob/master/testcases/network/README.md
Introduction to Fuego LTP test

3. Customize the spec for LTP network tests

```bash
# Add those required parameters to spec.json for Functional.LTP
$ cat /fuego-core/engine/tests/Functional.LTP/spec.json

    "rpc": {
        "tests": "net.rpc_tests net.tirpc_tests",
        "RHOST": "192.168.10.51",
        "LHOST_HWADDRS": "00:0e:c6:c5:b2:fe",
        "RHOST_HWADDRS": "00:0e:c6:c5:c0:b4",
        "homedir": "/opt/ltp",
        ...
    },

...
```

Some critical parameters in LTP network test:

- **RHOST**: the hostname of remote host;
- **LHOST_HWADDRS**: local hw addr of test Link;
- **RHOST_HWADDRS**: remote hw addr of test Link;
- **HTTP_DOWNLOAD_DIR/FTP_DOWNLOAD_DIR/FTP_UPLOAD_DIR/FTP_UPLOAD_URLDIR**;
- **NS_DURATION/NS_TIMES/CONNECTION_TOTAL**;
- **DOWNLOAD_BIGFILESIZE/DOWNLOAD_REGFILESIZE/UPLOAD_BIGFILESIZE/UPLOAD_REGFILESIZE**.

Those env variables will be exported before testing on the target.

Those optional environment variables can help us to change the test pressure.
## Introduction to Fuego LTP test

4. Check the test results

**Test env**: eel_5.0.0, agl-demo-platform-qa, R-Car M3.

<table>
<thead>
<tr>
<th>Tests</th>
<th>Result</th>
<th>Testcases</th>
<th>Execute Time</th>
<th>Logs &amp; Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tests</strong></td>
<td><strong>Result</strong></td>
<td><strong>Testcases</strong></td>
<td><strong>Execute Time</strong></td>
<td><strong>Logs &amp; Comments</strong></td>
</tr>
<tr>
<td>41</td>
<td>0</td>
<td>41</td>
<td>00:01:13</td>
<td></td>
</tr>
<tr>
<td><strong>net.sctp</strong></td>
<td>PASS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>36</td>
<td>70</td>
<td>00:13:00</td>
<td>All PASS with cc_3.0.0. - ipv4 PASS, but ipv6 related cases failed.</td>
</tr>
<tr>
<td><strong>net.rpc_tests</strong></td>
<td>PASS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>9</td>
<td>00:09:00</td>
<td>sendfile601: cannot startup sendfile on server. dhcpd6: port 53 already in use.</td>
</tr>
<tr>
<td><strong>stress.part3</strong></td>
<td>PASS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>2</td>
<td>43</td>
<td>00:18:28</td>
<td>2 failures - Known issues for LTP community.</td>
</tr>
<tr>
<td><strong>net.ipv6</strong></td>
<td>PASS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>87</td>
<td>0</td>
<td>87</td>
<td>00:17:09</td>
<td></td>
</tr>
<tr>
<td><strong>net_stress.ipsec_tcp</strong></td>
<td>FAIL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>0</td>
<td>24</td>
<td>00:04:11</td>
<td></td>
</tr>
<tr>
<td><strong>stress.part1</strong></td>
<td>FAIL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>12</td>
<td>00:09:49</td>
<td>Not support, - Ethernet0 and Ethernet1 has the same drivers which cannot be removed.</td>
</tr>
</tbody>
</table>
## Introduction to Fuego LTP test

<table>
<thead>
<tr>
<th>Tests</th>
<th>Result</th>
<th>Testcases</th>
<th>Execute Time</th>
<th>Logs &amp; Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>net.multicast</td>
<td>PASS</td>
<td>4</td>
<td>0</td>
<td>All PASS with cc_3.0.0. The causes is , - dnsmasq: port 53: Address already in use. - command rdist/finger not exist on the target. - ipneigh01: unknown yet. - xinetd: telnet on server cannot give expected output.</td>
</tr>
<tr>
<td>net.tcp_cmds</td>
<td>PASS</td>
<td>20</td>
<td>5</td>
<td>00:18:00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>net_stress.broken_ip</td>
<td>PASS</td>
<td>11</td>
<td>0</td>
<td>11:00:00</td>
</tr>
<tr>
<td>net_stress.interface</td>
<td>PASS</td>
<td>13</td>
<td>0</td>
<td>01:12:00</td>
</tr>
<tr>
<td>net.nfs</td>
<td>FAIL</td>
<td>36</td>
<td>38</td>
<td>04:19:16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>All PASS with cc_3.0.0. Test results: - ipv4 related tests FAIL.. - ipv6 related tests PASS. Now, still under investigation.</td>
</tr>
<tr>
<td>net.rpc</td>
<td>FAIL</td>
<td>2</td>
<td>2</td>
<td>00:06:14</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rpcinfo: unknown yet. rup: rup: 192.168.10.51: RPC: Timed out</td>
</tr>
<tr>
<td>net_stress.appl</td>
<td>PASS</td>
<td>10</td>
<td>0</td>
<td>00:05:56</td>
</tr>
<tr>
<td>net.features</td>
<td>FAIL</td>
<td>11</td>
<td>16</td>
<td>01:40:00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>All PASS with cc_3.0.0. Still under investigation.</td>
</tr>
<tr>
<td>net.ipv6_lib</td>
<td>FAIL</td>
<td>6</td>
<td>0</td>
<td>00:01:01</td>
</tr>
<tr>
<td>net_stress.ipsec_udp</td>
<td>FAIL</td>
<td>0</td>
<td>70</td>
<td>03:46:49</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>All PASS with cc_3.0.0. Still under investigation.</td>
</tr>
</tbody>
</table>
5. Customize the criteria file

The criteria.json file is used to specify the criteria used to determine whether a test has passed or failed.

```bash
$ cat /fuego-rw/boards/m3ulcb-Functional.LTP-criteria.json
{
   "schema_version":"1.0",
   "criteria": [
   {
      "tguid": "syscall",
      "min_pass": 1000,
      "max_fail": 5
   },
   {
      "tguid": "net.tirpc_tests",
      "fail_ok_list": ["tirpc_authdes_seccreate", "tirpc_authdes_create"]
   },
   ...
}
```

The criteria.json can be placed in the following locations:
- Criteria file specified in 'FUEGO_CRITERIA_JSON_PATH';
- Criteria file in /fuego-ro/boards/;
- Criteria file in /fuego-rw/boards/;
- Default criteria file;

- What we should keep in mind is that cases in criteria are not the same with in skiplist

We can use LTP test and some related network test.
About Fuego test framework:

Git repo:
- https://bitbucket.org/tbird20d/fuego.git
- https://bitbucket.org/tbird20d/fuego-core.git

Wiki:
- http://fuegotest.org/wiki/Fuego_Documentation

Fuego new website:
- http://fuegotest.org

Fuego maillist:
- fuego@lists.linuxfoundation.org
Fuego Roadmap

Recent past & Near future
- Priority was stuff affecting test API or test packaging (Needed before big push for new tests)
- Documentation
- New tests for AGL, LTSI, CIP
- Testplan enhancements
- Report generator and more charting control
- System provisioning support

Long-term
- Distributed test network
- Hardware testing
Our Future Work

- Fuego failed tests investigation
- Fuego pre_check improvement
- BSP tests integration
  - Those tests need some improvements
- New tests for AGL(ptest, kselftest, etc)
Thank you!

kyohei.oki@jp.fujitsu.com
liuwli.fnst@cn.fujitsu.com