AGL-Fuego display change flow

March 17, 2017
Japan Technical Jamboree 60

Song Cai
Fujitsu TEN
Self Introduction

- Song Cai ([cais.fnst@cn.fujitsu.com](mailto:cais.fnst@cn.fujitsu.com))
- Linux Software Engineer (2014 ~ )
- AGL (2016 ~ )
  - engaging on CIAT for AGL
  - especially AGL-Fuego(also named AGL-JTA)
Agenda

- What is AGL-Fuego
- How to use AGL-Fuego
- Introduce about CIAT.display
- Change html display flow
- Future work
What is AGL-Fuego

- automated test framework for AGL’s LTSI

- based on Fuego
  - formerly called JTA
  - a Jenkins front-end and a host/target script engine
  - official automated test framework for LTSI project

- features
  - job triggers
  - source code management & image build & image deploy
  - tests executing & results publication

- advantages
  - highly customizable
  - flexible test configuration & running tests in batches
  - easy & flexible board setup
How to use AGL-Fuego

- add test case view
- add test case
- execute test on target board
- display result
How to use AGL-Fuego

- add test case view
- dashboard view plugin required
How to use AGL–Fuego

- add test case
  - test script (backend)
  - job configuration (frontend)
  - configuration of target board [optional]
  - test spec & test plan [optional]

reference
- docs/jta-docs.pdf
- docs/How-to-Add-Test-Cases-on-JTA.pdf
- docs/How-to-Configure-CIAT-on-AGL-JTA.pdf
How to use AGL-Fuego

execute test on target board

Jenkins (frontend) → script engine (backend)

- Jenkins calls script engine first, then gathers test result/log from script engine
  a. cross-compile test suite for target board
  b. load test suite to target board then execute
  c. gather test logs
  d. analyze result

Jenkins (frontend) ← return result

script engine (backend) ← return logs
How to use AGL-Fuego

## display result

**Functional bzip2 Test Result**

- Latest Total: 11
- Latest Pass: 11
- Latest Fail: 0
- Latest Unittest: 0

<table>
<thead>
<tr>
<th>No.</th>
<th>Functional bzip2</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TEST-1</td>
<td>PASS</td>
</tr>
<tr>
<td>2</td>
<td>TEST-2</td>
<td>PASS</td>
</tr>
<tr>
<td>3</td>
<td>TEST-3</td>
<td>PASS</td>
</tr>
<tr>
<td>4</td>
<td>TEST-4</td>
<td>PASS</td>
</tr>
<tr>
<td>5</td>
<td>TEST-5</td>
<td>PASS</td>
</tr>
<tr>
<td>6</td>
<td>TEST-6</td>
<td>PASS</td>
</tr>
<tr>
<td>7</td>
<td>TEST-7</td>
<td>PASS</td>
</tr>
<tr>
<td>8</td>
<td>TEST-8</td>
<td>PASS</td>
</tr>
<tr>
<td>9</td>
<td>TEST-9</td>
<td>PASS</td>
</tr>
<tr>
<td>10</td>
<td>TEST-10</td>
<td>PASS</td>
</tr>
<tr>
<td>11</td>
<td>TEST-11</td>
<td>PASS</td>
</tr>
</tbody>
</table>

Start time: 2017-03-13 07:08:51
End time: 2017-03-13 07:08:59
Board version: koelsch
Test dir: /var/backups/aaa/jta.Functional.bzip2
Test Device: /dev/mmcblk0p2
Filesystem for Test Device: ext2
Command line: sh -v run-tests.sh 2>&1
Introduce about CIAT.display

- CIAT.display

A company’s AGL-Fuego

B company’s AGL-Fuego

C company’s AGL-Fuego

CIAT.upload

public git repo

CIAT.display

my AGL-Fuego (compare & display)

- public git repo:
  
  [https://gerrit.automotivelinux.org/gerrit/gitweb?p=staging/agl-jta-results.git;a=summary](https://gerrit.automotivelinux.org/gerrit/gitweb?p=staging/agl-jta-results.git;a=summary)
## Introduce about CIAT.display

- **glance** – display & compare the result

### Project CIAT.display

**Note:** After CIAT.display finishes, please **refresh** and test results will be shown!

**Latest Test:** 2017-01-24 @ fnst.dragonboard  
**Latest Total:** 1146  
**Latest Pass:** 961  
**Latest Fail:** 5  
**Latest Unittest:** 180

<table>
<thead>
<tr>
<th>No.</th>
<th>Test Name</th>
<th>2017-01-24 @ fnst.dragonboard</th>
<th>2017-01-24 @ ften.m3skit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Functional.LTP.Filesystem (detail)</td>
<td>total 24 pass 24 fail 0 untest 0</td>
<td>total 24 pass 24 fail 0 untest 0</td>
</tr>
<tr>
<td>2</td>
<td>Functional.LTP.Syscalls (detail)</td>
<td>total 1112 pass 927 fail 5 untest 180</td>
<td>total 1112 pass 924 fail 6 untest 182</td>
</tr>
<tr>
<td>3</td>
<td>Functional.LTP.Tests (detail)</td>
<td>total 10 pass 10 fail 0 untest 180</td>
<td>total 10 pass 10 fail 0 untest 182</td>
</tr>
</tbody>
</table>

**SUM**  
| total 1146 pass 961 fail 5 untest 180 | total 1146 pass 958 fail 6 untest 182 |
Change html display flow

Html display flow

Present:
1. xml may be not the best choice to generate html
2. Jenkins log is always huge
3. some process are redundant

Solution:
1. replace xml with json
2. simplify the process

Target:
1. parse device log
2. create json
3. generate html
4. display html

Present:
1. parse Jenkins log
2. create xml
3. generate html
4. display html
Why JSON (http://www.json.org/xml.html)

- simplicity
  JSON has a much smaller grammar and maps more directly onto the data
- human readable
  JSON is much easier for human to read than XML
- better data exchange format
  JSON is a better data exchange format. XML is a better document exchange format.
- suited for python
  JSON is better suited for python data structures, such as dict. Jenkins owns many python scripts in backend.
- compatible for Jenkins
  JSON has been applied to Jenkins for benchmark suit, which is easy for reconstruction.
Change html display flow

- **Functional test case**

  **Before**
  - Jenkins log
  - get_device_log
  - build.xml
  - device log
  - create xml
  - generate html
  - html

  **After**
  - Jenkins log
  - get_device_log
  - build.xml
  - device log
  - create json
  - generate html
  - html

**Item** | **comment**
--- | ---
build.xml | information about case
Jenkins log | all output in process
device log | output of case
html | final result web page

**Modification:**
1. get device log **directly**
2. generate html from **json**

**Advantage:**
1. avoid parsing huge Jenkins log
2. simplify the process
3. expect to shorten execution time
Change html display flow

Functional test case

- **before**
  - Jenkins log
  - Functional.cmt Test Result
    - Latest Total: 3
    - Latest Pass: 3
    - Latest Fail: 0
    - Latest Unittest: 0

- **after**
  - Functional.cmt Test Result

<table>
<thead>
<tr>
<th>Item</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>build.xml</td>
<td>information about case</td>
</tr>
<tr>
<td></td>
<td>all output in process</td>
</tr>
<tr>
<td></td>
<td>output of case</td>
</tr>
<tr>
<td></td>
<td>final result web page</td>
</tr>
</tbody>
</table>

**Modification:**
1. get device log directly
2. generate html from json
3. expect to shorten execution time

**Advantage:**
1. avoid parsing huge Jenkins log
2. simplify the process
3. expect to shorten execution time

**Item**

<table>
<thead>
<tr>
<th>No.</th>
<th>Functional.cmt</th>
<th>2017-03-04 11:52:22</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CMT interrupt</td>
<td>PASS</td>
</tr>
<tr>
<td>2</td>
<td>sh_cmt.0 dmesg feature</td>
<td>PASS</td>
</tr>
<tr>
<td>3</td>
<td>sh_cmt.0 /proc/interrupts presence</td>
<td>PASS</td>
</tr>
</tbody>
</table>
Change html display flow

Whole process of benchmark test

- device
  - run case
  - deploy case
  - get log
  - analyze
  - create html

- Jenkins
  - deploy case
  - get log
  - analyze
  - create html

- reference log
  - parse
  - save
  - res.json

<table>
<thead>
<tr>
<th>item</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>device log</td>
<td>output of case</td>
</tr>
<tr>
<td>reference log</td>
<td>threshold item</td>
</tr>
<tr>
<td>res.json</td>
<td>result data</td>
</tr>
</tbody>
</table>

valuable but unused
Change html display flow

Benchmark test case

<table>
<thead>
<tr>
<th>Item</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>build.xml</td>
<td>information about case</td>
</tr>
<tr>
<td>res.json</td>
<td>result data</td>
</tr>
<tr>
<td>device.board</td>
<td>board configuration file</td>
</tr>
<tr>
<td>config_device</td>
<td>range of threshold</td>
</tr>
<tr>
<td>html</td>
<td>final result web page</td>
</tr>
</tbody>
</table>

Modification:
1. avoid extra analysis of Jenkins log file
2. generate html from json

Advantage:
1. avoid parsing huge Jenkins log
2. simplify the process
3. expect to shorten execution time
Change html display flow

Benchmark test case

Benchmark iperf Test Result

- Latest Total: 3
- Latest Pass: 3
- Latest Fail: 0
- Latest Unittest: 0

<table>
<thead>
<tr>
<th>No.</th>
<th>Benchmark.iperf</th>
<th>Average</th>
<th>Unit</th>
<th>Criterion</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>tcp.bl rx</td>
<td>8.21 Mbits/sec</td>
<td>0.92 ~ 1.06</td>
<td>PASS</td>
</tr>
<tr>
<td>2</td>
<td>tcp.tx</td>
<td>11.60 Mbits/sec</td>
<td>1.00 ~ 1.00</td>
<td>PASS</td>
</tr>
<tr>
<td>3</td>
<td>tcp.bl tx</td>
<td>7.34 Mbits/sec</td>
<td>0.96 ~ 1.05</td>
<td>PASS</td>
</tr>
</tbody>
</table>

- Start time: 2017-03-10 13:19:09
- End time: 2017-03-10 13:22:21
- Board version: 010B.OCT100.07CR
- Test dir: /var/backups/aaa/jta/Benchmark.iperf
- Test Device: /dev/mmcblk0p5
- Filesystem for Test Device: rofs
- Command line: /iperf -c 192.168.10.1 -t 20; /iperf -c 192.168.10.1 -d -t 20

Modification:
1. Avoid extra analysis of Jenkins log file
2. Generate html from json
3. Expect to shorten execution time

Advantage:
1. Avoid parsing huge Jenkins log
2. Simplify the process
3. Expect to shorten execution time

Jenkins log

Reference log

Before

Create json

Create html

Jenkins log file

2. Simplify the process
3. Expect to shorten execution time

After
Future work

- LTP test suit
  - Current
    - solve the FAIL items in LTP cases
    - pass LTP cases on R-car M3 and dragon board
    - share results to AGL community
  - future work
    - feed back the change to the AGL community
    - solve other items on R-car M3 and dragon board

- cooperate with fuego

- participate in improving AGL-Fuego and AGL’s CIAT
Thank you!

cais.fnst@cn.fujitsu.com