Testing your AGL, yocto ptest, lava and more

ALS 2018

Jan-Simon Möller
Release Manager, AGL, The Linux Foundation

jsmoeller@linuxfoundation.org,
DL9PF @IRC and elsewhere
Dipl.-Ing. Jan-Simon Möller
j smoeller@linuxfoundation.org
'DL9PF' on #freenode

AGL Release Manager, EG CIAT Lead
Introduction
## Platform and Applications in AGL

### Platform
- Base system incl. libraries
- Built with the Yocto Project
- Application framework
- Other middleware

→ Part of filesystem image

### Applications & Services
- Services provide APIs
- Applications consume APIs
- Built with SDK
- Packaged as .wgt

→ Installed at runtime.
What to do where?

<table>
<thead>
<tr>
<th>You work on the Platform if you deal with a:</th>
<th>You work on the Applications/Services if you deal with a:</th>
</tr>
</thead>
<tbody>
<tr>
<td>● system library</td>
<td>● Service (agl-service-*)</td>
</tr>
<tr>
<td>● kernel driver</td>
<td>● Application</td>
</tr>
<tr>
<td>● BSP</td>
<td></td>
</tr>
<tr>
<td>● framework (itself)</td>
<td></td>
</tr>
</tbody>
</table>

→ low level

→ high level
The outcome here is usually a **filesystem image** but it can also be a **package feed**.

We have two options to inject tests in the process:

- **'Early'** as compile-time tests
  - Actually a great option as we get feedback very early – at compile-time
  - But this usually does not work well as we're cross-compiling and cannot execute the generated binaries

- **'Late'** once the image is created and booted
  - This works well but requires the target to be deployed and booted
  - For CI this needs to be automated
"Applications & Services"

- The outcome of the compilation is a *.wgt file
- Code is compiled for the target arch
- wgt files need to be **installed at runtime** (dynamic IDs / smack labels for security)
- Thus tests need to be **executed at runtime**
Let's explore

- How to add tests to AGL 'Platform'
- How to add tests to AGL 'Apps / Services'
- How to run the tests on the target
- What automation framework can be used
- and how reporting is done
How to add tests to the AGL 'platform'
The Platform is built using the YP

As discussed – compile-time tests would allow us to fail early, but we cannot execute the code if cross-compiled.

But what can we do:

- system libraries and programs usually come with a testsuite (aka 'make test')
- you have your own testsuite?
- let's use it!
• The YP has a feature for this called **ptest**
• In principle a **ptest** is the 'make test' packaged
• It can then be deployed on the target and executed using **ptest-runner**
from zlib_1.2.11.bb:
SRC_URI += "file://run-pptest"
inherit ptest
do_compile_pptest() {
  oe_runmake test
}
do_install_pptest() {
  install ${B}/Makefile   ${D}${PTEST_PATH}
  install ${B}/example    ${D}${PTEST_PATH}
  install ${B}/minigzip   ${D}${PTEST_PATH}
  install ${B}/examplesh  ${D}${PTEST_PATH}
  install ${B}/minigzipsh ${D}${PTEST_PATH}

  # Remove buildhost references...
  sed -i -e "s,--sysroot=${STAGING_DIR_TARGET},,g" \
     -e 's|${DEBUG_PREFIX_MAP}||g' \
     ${D}${PTEST_PATH}/Makefile
}
RDEPENDS_${PN}-ptest += "make"
• How is it added to the filesystem?
  • To add package testing to your build, set the `DISTRO_FEATURES` and `EXTRA_IMAGE_FEATURES`
    
    ```
    DISTROFEATURES_append = "ptest"
    EXTRA_IMAGE_FEATURES += "ptest-pkgs"
    ```
  
  • Shorthand is the `agl-ptest` feature for `aglsetup.sh`
  
  • All ptest files are installed in
    /usr/lib/<package>/ptest
• How is it executed?
• The "ptest-runner" package installs a "ptest-runner" which loops through all installed ptest test suites and runs them in sequence.
How to add tests to AGL 'Apps / Services'
For the applications and services, we actually face multiple areas

- we need to test the highlevel API calls of the services
- we need to test the applications
- we want reports on the code coverage
Applications and Services (2)

- For testing the highlevel calls, there is work in progress to use lua scrips for this task:
  - https://github.com/iotbzh/afb-test
    - https://github.com/iotbzh/afb-test/blob/master/README.md
  
- Final goal:
  add it as part of the application-templates
Applications and Services (3)

• gcov based code-coverage reporting
  • requires a separate build / binary
  • executed on the target, produces *.gcov files for each source file
• Work done to integrate this also into the makefiles of the app templates as well.
Applications and Services (4)

• Common to all:
  • they need to be executed on the target
  • partially with performance penalty (gcov)
  • for automation, this means we add a wrapper script to each service or application to exec the procedure
  • This is being called similar or equal to the ptest-runner
  • Executed in the CIAT infra
How to run the tests on the target
How to run it on the target (1)

• Manual:
  
• Platform:
  – ptest: either by ptest-runner or call run-ptest script directly
  – All ptest files are installed in /usr/lib/<package>/ptest

• Applications/Services
  – wrapper script required as entry point for CI (alike ptest)
  – tbd if this is part of app-templates
• Common issues:
  • needs to run on target
  • we need a common reporting
    – agreement is to use the KernelCI/Fuego json format
    – alternative: tap

• Join the conversation and the upcoming calls
What automation framework(s) can be used
A look back ...

A Vision/Plan (mid/long)

- Gerrit
- Jenkins (Build)
- SPDX Fossology
- Fuego
- Lava (lava)
- openQA?

- SDK
- Test & Debug App on board remotely
- VM

- e.g. AGL XDS
- WIP

- Remote Board Lab
- Remote Board Lab

- UI testing?

- eval tooling

- option / later

- WIP
The AGL CI infra overview

Build Management (jenkins)

Build Artifacts (kernel, rootfs, images)

Gerrit Server

Git Repos

Feedback

AGL Core LAVA Master

LAVA Slave

DUT

LAVA Slave

DUT

AGL Member LAVA Lab

LAVA Slave

DUT

LAVA Slave

DUT

AGL CIAT Webfrontend

Results Database
LAVA

• AGL uses LAVA and hosts an instance on https://lava.automotivelinux.org

• Current remote labs:
  • lab-AGL-core
  • lab-baylibre
  • lab-iotbzh

Welcome to LAVA

LAVA is an automated validation architecture primarily aimed at testing deployments of a current range of boards (device types) supported by this LAVA instance can be seen on the available for tests and currently running jobs.

LAVA components

• Results - viewing results of tests run by you or others.
• Scheduler - jobs are scheduled on available devices and the scheduler pages allow you to
• API - information on how to interact with LAVA and export data from LAVA using
• Help - documentation on using LAVA, worked examples and use cases, development
• Profile - you are logged in as dl9pf. Your profile provides access to jobs you have
**LAVA**

- metadata for the job
- action/deploy section
  - files to be used
- boot section
- test section
One or multiple
- inline
- from git repo
- uses yaml files
- lava-test-* are markers
  - for visualizing in LAVA
  - for visualizing in kernelCI
  - for cross-referencing
- test:
  [[..]]
definitions:
  - repository:
      metadata:
        format: Lava-Test Test Definition 1.0
        name: smoke-tests-basic
        description: "Basic test command for AGL images"
      run:
        steps:
        - agl-basic-test-shell-command
      from: inline
      name: agl-dut-inline-basic
      path: inline/agl-dut-inline-fake-filename.yaml
  - repository: git://git.automotivelinux.org/src/qa-testdefinitions.git
    from: git
    path: test-suites/short-smoke/smoke-tests-basic.yaml
    name: smoke-tests-basic
  - repository: https://git.linaro.org/lava-team/lava-functional-tests.git
    from: git
    path: test-suites/short-smoke/service-check.yaml
    name: service-check
Example: add a 'systemd service up' check


  [...] run:
  
  steps:

  - "cd common/scripts"

  - "./service-check-gfx.sh"
Example: add a 'systemd service up' check

Now its your turn:

- We need you to add your service checks!
  - in above script
- We need you to add your testsuites!
  - in qa-testdefinitions

- More details in my talk from AMM 2017!
and how reporting is done
• We use KernelCI to present the results

• https://kernelci.automotivelinux.org

• e.g.:
  https://kernelci.automotivelinux.org/test/board/r8a7796-m3ulcb/job/AGL-kernel-tree/kernel/AGL-gerrit-14179-1/
## Details for Tree «AGL-kernel-tree» - AGL-gerrit-14179-1

- **Board**: r8a7796-m3ulcb
- **Tree**: AGL-kernel-tree
- **Git branch**: aegl-branch
- **Git describe**: AGL-gerrit-14179-1
- **Git URL**: undefined
- **Git commit**: undefined
- **Date**: 2018-06-05

### «AGL-core-lab-1»

<table>
<thead>
<tr>
<th>Test suite name</th>
<th>Test suite ID</th>
<th>Total test sets</th>
<th>Test Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>busybox</td>
<td>5b161a6d19bd3200370cad8f</td>
<td>1</td>
<td>1 1 0 0</td>
</tr>
<tr>
<td>service-check</td>
<td>5b161a6c19bd3200370cad84</td>
<td>1</td>
<td>9 6 3 0</td>
</tr>
<tr>
<td>smoke-tests-basic</td>
<td>5b161a6b19bd3200370cad7d</td>
<td>1</td>
<td>5 4 1 0</td>
</tr>
<tr>
<td>health-test</td>
<td>5b161a6a19bd3200370cad78</td>
<td>2</td>
<td>2 1 1 0</td>
</tr>
<tr>
<td>yocto-ptest</td>
<td>5b161a6919bd3200370cad75</td>
<td>1</td>
<td>1 0 0 1</td>
</tr>
<tr>
<td>java</td>
<td>5b161a6719bd3200370cad51</td>
<td>1</td>
<td>34 34 0 0</td>
</tr>
</tbody>
</table>
Test details for test suite «service-check» (AGL-core-lab-1)

Lab name: AGL-core-lab-1
Board: r8a7796-m3ulcb
Tree: AGL-kernel-tree
Git branch: agl-branch
Git describe: AGL-gerrit-14179-1
Defconfig: defconfig+CONFIG_AGL=y
Date: 2018-06-05 05:06:52 UTC

Test Reports

Test set: default

<table>
<thead>
<tr>
<th>Test Case Name</th>
<th>Measurements</th>
<th>Date</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>bluetooth.service</td>
<td>⊗</td>
<td>2018-06-05</td>
<td>⊗</td>
</tr>
<tr>
<td>homescreen.service</td>
<td>⊗</td>
<td>2018-06-05</td>
<td>⊗</td>
</tr>
<tr>
<td>Weston.service</td>
<td>⊗</td>
<td>2018-06-05</td>
<td>⊗</td>
</tr>
<tr>
<td>ofono.service</td>
<td>⊗</td>
<td>2018-06-05</td>
<td>⊗</td>
</tr>
<tr>
<td>connman.service</td>
<td>⊗</td>
<td>2018-06-05</td>
<td>⊗</td>
</tr>
<tr>
<td>afm-system-daemon.service</td>
<td>⊗</td>
<td>2018-06-05</td>
<td>⊗</td>
</tr>
<tr>
<td>security-manager.socket</td>
<td>⊗</td>
<td>2018-06-05</td>
<td>⊗</td>
</tr>
</tbody>
</table>
• Next steps:
  • enhance WebUI
  • Cross-references
What's next?
Filling the gaps:

A Vision/Plan (mid/long)

- Gerrit
- Jenkins (Build)
- SPDX Fossology
- Fuego
- Lava (lava)
- openQA?
- Remote Board Lab

e.g. AGL XDS
SDK
VM

Test & Debug App on board remotely
WIP
WIP

UI testing? option / later
QA
Thank you.

Contact:
jsmoeller@linuxfoundation.org
References

- ptest: https://wiki.yoctoproject.org/wiki/Ptest