LAVA External Device Monitoring

Contents

Test Cases **2**

LAVA Features **2**
LXC 2
MultiNode 2
Secondary Connections 3

Approach Overview **3**

LAVA Job Connection Layout **3**
Test Job 3
Job File Example 4

QA Report **6**

This document describes how to execute automated LAVA tests controlling resources external to the DUT across a network implementing a LAVA parallel pipeline job.

Test Cases

The approach proposed in this document will help to address test cases like:

- Executing a test in the DUT where certain power states are simulated (for example a power loss) during specific test actions using a programmable PSU external to the DUT.
- Executing a test in the DUT simulating SD card insertion and removal using an external device.

The only assumption, in both scenario, proposed in this document is that the external device (either a programmable PSU or SD-card simulator) can be accessed through the network using SSH.

LAVA Features

LAVA offers the following features that can be combined to implement a solution for the test cases mentioned in this document:

- LXC to deploy required software and tools to access the external device.
- MultiNode to communicate data between jobs actions.
- Secondary connections for executing tests through SSH.

LXC

LAVA supports LXC containers both as a standalone device type and as dynamic transparent environments in order to interact with external devices. In either case the LXC Protocol[1] is used.

MultiNode

The MultiNode Protocol[2] allows data to be shared between actions, including data generated in one test shell definition being made available over the protocol to a deploy or boot action of jobs with a different role.

Synchronisation is done using the MultiNode API, specifically the `lava-send` and `lava-wait` calls.

Secondary Connections

LAVA allows Secondary Connections[3] to open network connections to external devices using MultiNode submissions.

Approach Overview

The main idea is to create a LXC container device associated to the DUT responsible to execute the automated test, then opens a SSH connection to an external device, and use the MultiNode API in order to synchronize both devices and pass data between them with the LXC container serving like a coordinator of the different LAVA tests actions.

In this way, a server-client layout is setup that will help to execute tests in a board attached to LAVA (server side) with intervention of external devices (client side).

LAVA Job Connection Layout

The LXC container is deployed directly from the LAVA dispatcher and coordinate the execution of the parallel pipeline between the DUT and the external device (secondary connection) from there.

The layout model would be something like:

```
            ------------- DUT
          / MultiNode
LAVA (LXC)
```

[1] https://lava.collabora.co.uk/static/docs/v2/actions-protocols.html#lxc-protocol-reference

[2] https://lava.collabora.co.uk/static/docs/v2/actions-protocols.html#multinode-protocol

[3] https://lava.collabora.co.uk/static/docs/v2/pipeline-writer-secondary.html

```
            \
             ------------- Secondary Connection (PSU, SD-Card HW)
              MultiNode
```

Test Job

This section shows the basics proposed in this document using a LAVA job file example.

The following steps describe the main flow of the job:

1 - Create two types of roles `host` and `guest`. The `host` role will contain the LXC container and the DUT, the `guest` role will label the SSH connection for the external device. This creates two groups (`host` and `guest`) that can communicate using the MultiNode API, so messages can be sent between the LXC and Device as the server and the secondary connection as the client.

2 - Label both types of roles in the `protocols` section of the job.

3 - Deploy and boot the `LXC` container (`host`).

4 - Execute a test in the LXC container using the MultiNode API to send the `lava_start` message, so the `deploy` action for the external device can start, and waits for remaining clients to start using the `lava-sync` call.

5 - Deploy the DUT (`host`).

6 - Deploy the external device (`guest`) , which is waiting for the LXC `lava_start` message to start deployment. Once this message is recevied, the guest device is deployed.

7 - Boot DUT.

8 - Boot external device.

9 - Execute a test in the DUT sending the `lava-sync` call.

10 - Execute a test in the external device sending the `lava-sync` call.

11 - Once all clients are synchronized (the LXC, DUT and external device), start executing tests.

12 - Tests executed in the DUT and external device needs to use the MultiNodeAPI[4] in order to pass data between them.

As the LXC is deployed and booted first, the LXC can run a test shell before deploying the device, before booting the device, before the test shell action on the device which starts the secondary connection guests or at any later point (AddingTestsActions[5]).

[4] https://lava.collabora.co.uk/static/docs/v2/multinodeapi.html#multinode-api

[5] https://lava.collabora.co.uk/static/docs/v2/writing-multinode.html#adding-test-actions

Job File Example

```
job_name: LXC and Secondary connection with a Device

timeouts:
  job:
    minutes: 30
  action:
    minutes: 3
  connection:
    minutes: 5
priority: medium
visibility: public

protocols:
  lava-lxc:
    host:
      name: lxc-ssh-test
      template: debian
      distribution: debian
      release: stretch
  lava-multinode:
    # expect_role is used by the dispatcher and is part of delay_start
    # host_role is used by the scheduler, unrelated to delay_start.
    roles:
      host:
        device_type: beaglebone-black
    # This makes this role essential in order to execute the test.
    essential: True
        count: 1
        timeout:
          minutes: 10
      guest:
        # protocol API call to make during protocol setup
        request: lava-start
        # set the role for which this role will wait
        expect_role: host
        timeout:
          minutes: 15
        # no device_type, just a connection
        connection: ssh
        count: 3
     # each ssh connection will attempt to connect to the device of role 'host'
        host_role: host

actions:
- deploy:
    role:
    - host
    namespace: probe
    timeout:
      minutes: 5
    to: lxc
    # authorize for ssh adds the ssh public key to authorized_keys
    authorize: ssh
    packages:
    - usbutils
```

QA Report

Once tests results are available at LAVA , and the test cases are enabled for the specific images from the test case repository, the results will be available from the QA Report App automatically.