Welcome!

Thank you for coming and spending some time on Fuego
Welcome!

- Thank you for coming and spending some time on Fuego
- Note: We haven’t done this before
  - You are all guinea pigs
Agenda

• Hackathon Timeline
  • Welcome 10:30
  • Presentations/Discussion 10:45-11:30
  • Project selection/Hacking setup 11:30-12:00
  • Lunch 12:00-13:00
  • Hacking 13:00-16:30

• Presentations/Discussion
  • Unknown features
  • Features Brainstorming
  • Roadmap priorities
Hackathon setup

• Network Details
  • SSID: XXX-XXXXXXX
  • WEP Key: xxxxxxxxxxxxxxxxxxx

• Wiki account:
  • Create an account, right now, at:
    • http://fuegotest.org/wiki/FrontPage
  • Bookmark:
    http://fuegotest.org/wiki/Japan_Fuego_Hackathon_2017

• Hardware setup:

• Software setup:
  • Local repository or SSH to “theFalcon”
Outline

Agenda
Unknown Features
Features Brainstorming
Roadmap Priorities
Hacking
Unknown Features

- Can run ftc outside the container
- per_job_build test variable
- Test variables in the board file
- Dynamic board variables
- ftc add-view
- FUEGO_DEBUG bitmasks
- Add jobs to multiple boards
- Jenkins search for job
- Jenkins history
- Jenkins console log links and menus
Can run ftc outside the container

- Useful to be able to run commands straight from host
- Can use environment variable to specify the container name
  - FUEGO_CONTAINER
  - Otherwise, uses first running container with the word 'fuego' in its name
- Requires extra packages installed on host:
  - python-lxml, python-jenkins, python-requests, python-yaml
per_job_build test variable

- Default build behavior:
  - Tests with same PLATFORM (toolchain) share the same build directory
  - This is done to save time and space
- per_job_build forces Fuego to use a different build directory for each job
  - Important if a spec has variables that affect the build
    - e.g. board1.default.Functional.sometest vs. board1.build_with_debug.Functional.sometest
  - Used with Functional.kernel_build, where spec can indicate a whole different source base
Test variables in the board file

- Test variables from test specs are generated with the test prefix:
  - Overlay system converts “my_var” in the spec to FUNCTIONAL_TEST_MY_VAR
- You can declare these in the board file, and they take precedence over ones declared in the spec
- Examples:
  - FUNCTIONAL_LTP_SKIPLIST
  - FUNCTIONAL_HELLO_WORLD_ARGS
- Useful because:
  - Can override variables on a per-board basis
Dynamic board variables

- You can use dynamic board variables to store test variables temporarily for a board.

How to use:
- `ftc set-var` - to set a variable
- `ftc query-board` – to see variables
- `ftc delete-var` – to remove a variable

- Dynamic vars are stored in `/fuego-rw/boards/<board>.vars`

- Can use at command line, or in a test
ftc add-view

- Handy command to quickly create Jenkins views
- Can create with job regex or job list
- How to use:
  - `ftc add-view name <regex>`
  - `ftc add-view name =<joblist>`
- "ftc add-view name" (with no job specification) creates a regex using name:
  - e.g. `ftc add-view myboard`
  - e.g. `ftc add-view LTP`
FUEGO_DEBUG bitmask

- Select Fuego subsystem to see debug messages for
- FUEGO_DEBUG=non-zero
  - Debug the main script (all bash activity)
- Additional bit values:
  - 2 = debug the parser
  - 4 = debug the results saver
  - 8 = debug the chart generator code
- FUEGO_DEBUG=1 -> shell messages
- FUEGO_DEBUG=15 -> everything
Add jobs to multiple boards

- Can add jobs for multiple boards with a single `ftc` command

How to use:
- Just use multiple boards, separated by commas, with `ftc add-job`:

Examples:
- `ftc add-job -b board1,board2,board3 -t Functional.sometest`
- `ftc add-jobs -b board1,board2 -p testplan_ltsi`
Jenkins search

- Can use Jenkins search to get a quick list of jobs matching a name
- Very handy for quickly finding jobs
  - Fuego job lists can get long, especially for multi-board farms
- Don’t have to create a view
Jenkins history

- Jenkins history is shown per-view (!!)
- You can examine history of runs per-node, per-job, or for arbitrary set of tests
  - May have to create a view with the jobs you are interested in
Jenkins console log links and menus

- Can click on links in console log
- Links:
  - Upstream project
  - Jenkins user
  - Node
    - Anything starting with ‘http’
- Some items provide menus of Jenkins actions
  - e.g. Build history for node
Agenda
Unknown Features
Features Brainstorming
Roadmap Priorities
Hacking
Brainstorming ideas

- Current features
- Tim’s roadmap
- Daniel’s To-Do list (refactored)
- Feedback from BOFs
- Miscellaneous items
1.1 Feature list

- Jenkins front end
  - Also has a command line interface ("ftc")
- Containerized
- Overlay system, for customization
  - Boards, distros, specs, plans
- Build system
- Test are driven from host
- Multiple Transports
- Collection of Tests
- Results parsing and post-processing
Version 1.2.1 Features

- Unified Output Format
- Test dependency system
- Complex pass criteria handling
- Dynamic board variables
- Charting
- Test source from git repositories
- Transport modifications
- Test improvements
Roadmap

• Recent past:
  • Priority was stuff affecting test API or test packaging
    • Needed before big push for new tests
• Near future:
  • Documentation
    • Conversion to reStructuredText
    • Refactoring
    • Tutorials
  • New tests for AGL, LTSI, CIP
    • What tests to tackle next?
Roadmap (cont.)

• Near future (cont.):
  • Testplan enhancements
    • Controlled test sequences
      • Similar to Jenkins pipelines
      • Processing multiple steps (provisioning, testing, notifications, report generation) in sequence
    • More fields for plan configuration
  • Report generator and more charting control
    • Now that we have unified output, we can do queries, and different output formats
Roadmap (cont.)

• Near future (cont.):
  • **System provisioning** support
    • Install of software under test
      • Has been out-of-scope for Fuego
    • e.g. AGL image deploy, LTSI kernel update, etc.
    • Full automation requires board management API
    • Looking at labgrid as possible solution

• Long-term
  • Distributed test network
  • Hardware testing
Other Priorities

- LAVA integration
  - We have everything needed for transport integration
  - Need test-level integration
    - Separate build phase
    - Deploy to LAVA server
    - Create LAVA test that does:
      - Execute test on board
      - Collect results
To-Do from Daniel

- Provisioning ideas:
  - Provide deploy and boot as in LAVA
    - Deploy: prepare nfs/tftp
    - Boot: poweron board/reboot/ssh
  - Deploy the OS (Provisioning)
    - 1) hawkbeat/ostre... also tests the updates
    - 2) u-boot serial port with pexpect
    - 3) TFPT/NFS or NBDroot
    - 4) Fastboot
  - Update filesystem on the SD card by using update software
    - 2 partitions
To-Do from Daniel (cont.)

- Parallel testing on same device types
  - Use Jenkins labels
- Multi-node tests like in LAVA
- Auto-generate timeouts
- Support matrix of boards/tests
  - Fuzz coverage combinations
- Bisects
- Kernel CI integration
To-Do from Daniel (cont.2)

- LAVA support
  - Just open a hacking shell?
  - Or submitting YAML jobs?
- REST API instead of master-slave model
- Support for read-only filesystems
  - Create a ramfs?
- Support for including strace output or running gdb remotely
To-Do from Daniel (cont.3)

- Ability to deploy standard distributions (for testing the kernel, hardware, or apps!)
  - Yocto based generic filesystem
  - Debian, others
- Allow to enter easily into a developer shell
  - $ fuego shell
- Login
  - support user, root password, ssh key [?]
To-Do from Daniel (cont.4)

- Tests:
  - RT-tests
    - LTP
    - rt-tests
    - rt-eval (disturbance)
  - Software update tests
- Disturbance loads
  - stress, hackbench, ...
  - Power cut tests
    - target_poweroff/poweron
  - Simulate application environment...
Other ideas

• Create an interface to install and list new tests
  • Finish “ftc install-test”?
  • Add remote support to “ftc list-tests”?
• Plugin system like avocado (for ftc)
• Ability to run tests already in the target
  • LTP does this, but need to support general mechanism
• ADB Transport
• Configurable Jenkins port (8090 default?)
• ADB support
  • Run abd outside container (on host), and container doesn’t have to know about usb changes

• Could use transport=local for host as DUT
  • Now currently used for docker container as DUT

• Bypassing build step
  • It’s OK to have something as a build cache, but make sure not to lost ability to build from source
  • Don’t allow “magic binaries” that someone can’t rebuild
Notes from ELC 2017 BOF (2)

- Bisect
  - Should be a tool outside Fuego to bisect based on Fuego test result
  - Ftc needs to return proper error code
  - Maybe provide an example for how to do it

- Image Deploy, re-flash
  - Since LAVA does these, and AGL already uses LAVA, these are not high priority at the moment
Ideas from ELCE 2017

• Greg KH (and other mainline developers) need localhost board
  • Maybe can use something simpler than ssh to localhost?
  • We have transport=local, but that only works inside the container (is that right?)
Miscellaneous ideas

- **HealthCheck test**
  - Ftc target-status
  - Already have:
    - `ftc run-test -b <board> -t Functional.fuego_board_check`
- **Automatic board installation /wizard**
  - `ftc find-board`
- **Use “ftc run-test” in Jenkins, instead of direct invocation of main.sh**
  - This is what avocado does
- **Send results to a centralized repository**
Agenda
Unknown Features
Features Brainstorming
Roadmap Priorities
Hacking
Roadmap priorities

- See http://fuegotest.org/wiki/JFH17_Discussion_Notes
Agenda
Unknown Features
Features Brainstorming
Roadmap Priorities
Hacking
Hacking

• Survey of room
  • How many experienced vs new Fuego users?

• Projects
  • Project leaders
  • Type:
    • training/issue detection
    • problem investigation
    • feature development
  • Project list:
    • See http://fuegotest.org/wiki/JFH17_Hacking_Guide

• Setup
Fuego reference materials
Micro-Introduction

Fuego = (Jenkins + host scripts + pre-packaged tests) inside a container
Architecture Diagram

Host machine:
- Container build system

Docker container:
- Jenkins
- Test programs
- Scripts

Volume Mount
- Toolchains
- Config
- Builds
- Logs

Web control interface

Target board

- Test programs
- Scripts
Outline

Introduction

Vision

Core Principles

Diagrams

Resources
Vision – super high level

Do for testing what open source has done for coding

- Significant parts of the test process are unshared, ad hoc, private, etc.
  - For no good reason – most QA doesn’t need to be proprietary
    - There are OSS frameworks and test programs but parts are missing to create a open testing community.

- Fuego Goal:
  - Promote the sharing of tests, test methods, and results, the way code is shared now
    - Make it easy to create, share and discover tests
    - Make test results easy to share and evaluate
Core principles

- Actually finds bugs
- Allows sharing
- Usable by wide audience
  - Minimal requirements
  - Customizable
- Applicable to embedded
- Easy to use
- Scalability via decentralization
Outline

- Introduction
- Vision
- Core Principles
- Diagrams
- Resources
Overlay processing

Fuego functions
- main.sh
  - functions.sh
  - common.sh
  - overlays.sh
  - reports.sh
  etc.

<board>.conf

base-distrib.fuegoclass

base-board.fuegoclass

testplan

test specs

ovgen.py

Test variables
- prolog.sh

Base script
- fuego_script.sh
  - test_build()
  - test_deploy()
  - test_run()

main.sh

Functions

Base script

## Comparison of Fuego and Lava

<table>
<thead>
<tr>
<th>Assumption</th>
<th>Fuego</th>
<th>LAVA</th>
<th>Jenkins</th>
</tr>
</thead>
<tbody>
<tr>
<td>Board starting status</td>
<td>Board is running</td>
<td>Board will be provisioned and booted</td>
<td>Node is running</td>
</tr>
<tr>
<td>Test initiated by:</td>
<td>Manual, Jenkins trigger</td>
<td>External job insertion?</td>
<td>Jenkins trigger</td>
</tr>
<tr>
<td>Test software availability:</td>
<td>Source included, test binary is built and deployed to target</td>
<td>Is in distro or on target, or is installed during test</td>
<td>Builds software – no built-in deploy - left as exercise for test developer</td>
</tr>
<tr>
<td>Test scheduling</td>
<td>By Jenkins, cli has none, no target reservation system</td>
<td>By LAVA</td>
<td>By Jenkins</td>
</tr>
<tr>
<td>Results processing</td>
<td>Log parsing, send results to server (prototype)</td>
<td>Collect results?</td>
<td>Visualization for common formats (TAP, junit, xunit)</td>
</tr>
</tbody>
</table>
Outline

Introduction
Vision
Core Principles
Diagrams
Resources
Resources

• Fuego web server:
  • http://fuegotest.org/
  • wiki: http://fuegotest.org/wiki

• Mailing list:
  • https://lists.linuxfoundation.org/mailman/listinfo/fuego

• Repositories:
  • https://bitbucket.org/tbird20d/fuego
  • https://bitbucket.org/tbird20d/fuego-core
Fuego