Fuego
Test System Status Summary
June 2018
Tim Bird
Fuego Maintainer
Sony Electronics
Outline

Introduction

Feature List

Feature Status
Micro-Introduction

Fuego =

(Fuego Linux distribution + host scripts + pre-packaged tests + Jenkins)

all inside a container
Architecture Diagram

Host machine:

Container build system

Docker container:
- Fuego Linux distribution
- Jenkins
- Test programs
- Scripts

Volume Mount

Toolchains
Config
Builds
Logs

Web control interface

Target board
Core features

- Distribution of Linux for testing
- Build system
  - Architecture-neutral & inherently cross-platform
- Includes a collection of tests
  - Scripts for test execution
  - Results parsing, analysis, and visualization
- Report generation
- Multiple transports
- Jenkins front end
  - Also has a command line tool
Outline

Introduction

Feature List

Feature Status
Slide key

*Italic header = reference material*
Feature Areas

- Installation
- Test Execution
- Results Analysis
- User Interface
- Visualization
- Reports
- Tests

- Fuego Distribution
- Infrastructure
- Usability
- Documentation
- Marketing
Installation

- Docker build
  - Pre-built Docker image (partially completed)
    - Problems with container customization
- Adding a board
- Adding a toolchain
- Populating Jenkins artifacts
- Provisioning setup (none)
Pre-built docker image

- Ability to use Fuego without building the docker image
  - Create a pre-built Fuego docker image, and host it at docker.io
  - e.g. “docker run fuego”
- Requires automatic container customization
  - Network proxy
  - User and group
  - Volume mount customization
- Includes refactoring the Fuego directory layout
  - Turned out to be too intrusive for 1.3 release
Execution

- Launch – jenkins job or command line
  - main.sh, ftc run-test
- Test customization (plans, specs, criteria)
- Overlay system (ovgen.py)
- Test dependency system (need_check.sh)
- Transports (serial, ssh, local, ttc, lava)
- Build system
  - Source retrieval
- Deployment
- Remote execution
- Log retrieval
- Error handling/Abort
- Timeouts
Results Analysis

- Log parsing
  - Unified Output Format
- log_compare
- Pass-criteria
  - per-board pass-criteria
- Jenkins history
User Interface

- Jenkins UI
- Job launching
- ftc to manage Jenkins objects
  - ftc add-node, add-job, add-view
- History, status, result indicators
  - Not very rich (no icon for abort or skip)
- log links, results description
- Failure reason (missing)
Visualization

- Charts
- Tables
- Plots (jQuery flot module)
- chart_config
  - A few items missing from JTA (hide/unhide data sets)
- Log splitting by testcase
Log splitting by testcase

- Ability to split the test log into pieces, according to testcase boundaries
- Only works for some logs
- Requires slight modification to test’s parser.py
- Addition to UI
  - Can click on testcase in Jenkins UI, and see section of log related to that testcase
- Very handy for examining results details
Reports

- ftc list-runs, gen-report
  - field control
  - output formats
  - --where (filtering)
    - (time comparisons are a bit confusing)
    - (could use a batch id)

- Dynamic documents
  - chart insertion (missing)
  - coverage (very little)
Report generation improvements

- More output formats
  - html, rst, pdf, excel, csv
- Control of fields displayed
  - header_fields
  - report fields
- More filtering (--where options)
  - especially tguid:result
  - Try this:
    - ftc gen-report –where “tguid:result!=PASS”
Tests

- LTP – see next slide
- kselftest
- Realtime, Filesystem, Network
- POSIX
- System benchmarks
- Security/vulnerability
- autopkg
- ptest
- Fuego self-tests / release test
LTP

- lots of specs
- skiplists
  - Autodetect skiplist items
- supports separate build and manual deploy
- one_test
  - execute a single test program

Problems:
- realtime parsing is not good
- should separate POSIX from other tests
New tests

• Realtime benchmarks:
  • backfire, deadlinetest, migratetest, pmqtest, ptsematetest, sigwaittest, svsemateast

• Other tests (updated or new):
  • dbench4, dd, iperf3, vuls, autopkgtest, year2038
  • LTP_one_test
  • fuego_board_status

• Fuego self-tests:
  • fuego_lint, fuego_tguid_check, fuego_ftc_check
  • fuego_release_test
Fuego release test

- Complicated test to do a full release test
  - Builds docker container
  - Runs docker container for "release under test", alongside "test-runner" container
- Checks Jenkins web interface
  - Using HTML DOM element checks
  - Using comparisons of web page rendered images
- Adds capabilities to Fuego distribution for testing of other DUT web or image features
Fuego Distribution Additions

- **clitest**
  - Tests expected output of commands
- **Selenium**
  - Web page HTML testing
- **Chromium, imagemagick, python-pillow**
  - Web page image testing
- **pexpect, flake8 (python lint)**
  - Fuego self-test
- **python-reportlab**
  - Report formats
- **iperf3**
  - Network testing
Web page and image comparison tools

- Added Selenium to Fuego distribution
  - For web page testing
- Added Chromium to Fuego distribution
  - For web page rendering automation
- Added tools for:
  - Comparison of returned HTML
  - Web page image capture
  - Image comparison
    - With support for masked regions
- **Note:** This is not generalized yet
  - Need to read Functional.fuego_release_test scripts and use as example
Infrastructure

- bitbucket git repositories
- Hardware board control
- Individual test phases
- ftc outside the docker container
Hardware board control

- General feature is ability to control board under automation
  - Added in 1.3:
    - Hook for hardware board reboot
    - Shows method for adding board control hooks
  - Goal is to support provisioning and other hardware functions, as well
    - ex. off-DUT test hardware control and multiplexing
- Would rather re-use some other board control layer
  - e.g. LAVA, labgrid, Dryad (from SLAV), ttc, etc.
Individual test phases

- Ability to run test phases individually
  - Main purpose is to allow separation of:
    - Test program build (on host)
    - Test execution on target
- Can use ftc option:
  - `ftc run-test -p 'pcb'`
    - Executes pre_test, pre_check and build phases, then stops
- Can use environment variable
  - `FUEGO_TEST_PHASES="pre_test pre_check build"`
ftc outside the docker container

- docker adds unneeded overhead to some commands
- Some commands can now be run outside the docker container
  - New fuego.conf file to specify directory locations
  - list-runs, gen-report can be done directly on host
- Partial step towards use of low-level Fuego functionality with alternate UIs and frameworks
Usability

• End user
  • License
  • Web site
  • Ease of installation/use
    • Consistency
    • Obviousness
  • Robustness
    • Error Handling

• Developer support
  • Coding style
  • Languages (bash, python, json)
  • Debugging
  • Documentation
Contributor guidelines

- Recently added to wiki
- Coding style guide
  - Mostly indentation (4 spaces, no tabs)
  - See http://fuegotest.org/wiki/Coding_style
- License guide
- Patch submission tips
  - See http://fuegotest.org/wiki/License_And_Contribution_Policy
Documentation

- Installation (quickstart guide)
- End-user docs (wiki, pdf)
  - User’s guide (adding...)
  - Walkthroughs (none)
  - Jenkins help (none)
    - Setting up triggers (needed)
    - Setting up notifications (needed)
- Developer docs (wiki)
  - Architecture
  - API reference
- Tutorials ??
  - (Some presentations exist, but they’re old)
Documentation conversion

• Conversion of docs to reStructuredText
  • Replace PDF and wiki docs with rst
  • Move all docs under source repository
  • Use sphynx to create multiple formats
  • Publish on readthedocs.io

• Made some progress
  • Have sphynx templates in place

• Got stuck on markup conversion
  • Considered automation, but hit some hurdles

• See http://fuegotest.org/wiki/rst_docs
Marketing

- Outreach
- Visibility
- User growth
  - (slower than expected)
Test server (prototype)

- Test store (put-test, get-test)
- Distributed test requests (put-request)
- Aggregation
  - Results sharing (put-run)
- Data mining (nothing)
Outline

Introduction
Feature List
Feature Status
## Feature Status

<table>
<thead>
<tr>
<th>Feature Area</th>
<th>Status</th>
<th>Potential Improvements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installation</td>
<td>Pretty Good</td>
<td>Pre-built image, provisioning help</td>
</tr>
<tr>
<td>Test Execution</td>
<td>Good</td>
<td>Handle timeouts better</td>
</tr>
<tr>
<td>Results Analysis</td>
<td>Very Good</td>
<td>Report of failure reason</td>
</tr>
<tr>
<td>User Interface</td>
<td>Good</td>
<td>??</td>
</tr>
<tr>
<td>Visualization</td>
<td>Good</td>
<td>Add more chart_config options</td>
</tr>
<tr>
<td>Reports</td>
<td>OK</td>
<td>Integration into dynamic docs</td>
</tr>
<tr>
<td>Tests</td>
<td>OK</td>
<td>More tests</td>
</tr>
<tr>
<td>Fuego Distribution</td>
<td>Good</td>
<td>Ability to load packages as needed</td>
</tr>
<tr>
<td>Infrastructure</td>
<td>OK</td>
<td>Move to gitlab</td>
</tr>
<tr>
<td>Usability</td>
<td>OK</td>
<td>??</td>
</tr>
<tr>
<td>Documentation</td>
<td>OK</td>
<td>Need tutorials and walkthroughs</td>
</tr>
<tr>
<td>Marketing</td>
<td>Good</td>
<td>??</td>
</tr>
<tr>
<td>Test server</td>
<td>Not Good</td>
<td>Finish implementation, provide server code</td>
</tr>
</tbody>
</table>
Final Thoughts

• Don’t want to focus on negatives
• Fuego is a very rich, capable, flexible system
• An amazing amount of features
  • It’s often easy to implement a new feature, because of rich existing infrastructure
  • Can usually support what you want to do
• Very friendly community