LTSI Status Update
LTSI kernel release cadence, current shape and the future
LTSI use case confirmation
LTSI Test Discussion
Resources

LTSI workshop @ ELCE2014
project update (focused on release scheme and testing)

Tsugikazu Shibata, Hisao Munakata

Linux Foundation Consumer Electronics working group

15:30 - 17:30 October 13th 2014
Room 10, Congress Center, Dusseldorf
LTSI Status Update
LTSI kernel update @ February 24, 2014

LTSI 3.0.79 --> 3.0.101 (EOL)
LTSI 3.4.46 --> 3.4.81 (update)
## LTSI 3.10 development history (result)

<table>
<thead>
<tr>
<th>item</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel 3.10 merge window open</td>
<td>2013.4.28</td>
</tr>
<tr>
<td>kernel 3.10 merge window close</td>
<td>2013.5.12</td>
</tr>
<tr>
<td>kernel 3.10 release</td>
<td>2013.6.30</td>
</tr>
<tr>
<td>Announce of 2013 LTS kernel version</td>
<td>2013.8.4</td>
</tr>
<tr>
<td>LTSI-3.10 git tree open</td>
<td>2013.9.11</td>
</tr>
<tr>
<td>3.10 becomes LTS (=3.12 release)</td>
<td>2013.11.15</td>
</tr>
<tr>
<td>LTSI-3.10 merge window open</td>
<td>2013.11.15</td>
</tr>
<tr>
<td>patch collection period</td>
<td>75 days</td>
</tr>
<tr>
<td>LTSI-3.10-rc1 (=merge window close)</td>
<td>2014.1.29</td>
</tr>
<tr>
<td>validation period</td>
<td>26 days</td>
</tr>
<tr>
<td>LTSI-3.10 release</td>
<td>2014.2.24</td>
</tr>
</tbody>
</table>
# LTSI 3.14 development schedule (underway)

<table>
<thead>
<tr>
<th>item</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel 3.14 merge window open</td>
<td>2014.1.9</td>
</tr>
<tr>
<td>kernel 3.14 merge window close</td>
<td>2014.2.2</td>
</tr>
<tr>
<td>kernel 3.14 release</td>
<td>2014.3.30</td>
</tr>
<tr>
<td>Greg announced that 3.14 is next LTS(I)</td>
<td>ELC2014</td>
</tr>
<tr>
<td>LTSI-3.14 merge window open</td>
<td>2014.8.21</td>
</tr>
<tr>
<td>patch collection period</td>
<td>70 days</td>
</tr>
<tr>
<td>LTSI-3.14-rc1 (=merge window close, target)</td>
<td>2014.10.30</td>
</tr>
<tr>
<td>validation period</td>
<td>50+ days</td>
</tr>
<tr>
<td>LTSI-3.14 release (target)</td>
<td>2014.12.25?</td>
</tr>
</tbody>
</table>

Please be sure not to late LTSI 3.14 merge window close!
## Patch submission status after LTSI-3.10 release

<table>
<thead>
<tr>
<th>Month</th>
<th>Sender (company)</th>
<th>number</th>
<th>contents</th>
<th>target</th>
</tr>
</thead>
<tbody>
<tr>
<td>March</td>
<td>Adrian Hunter (Intel)</td>
<td>2</td>
<td>MMC</td>
<td>3.10</td>
</tr>
<tr>
<td>March</td>
<td>Wei.sern.chan (Intel)</td>
<td>5</td>
<td>PWM</td>
<td>3.10</td>
</tr>
<tr>
<td>April</td>
<td>Simon Horman (Renesas)</td>
<td>10</td>
<td>serial</td>
<td>3.10</td>
</tr>
<tr>
<td>April</td>
<td>Simon Horman (Renesas)</td>
<td>1</td>
<td>USB</td>
<td>3.10</td>
</tr>
<tr>
<td>May</td>
<td>Geert Uytterhoeven (Renesas)</td>
<td>1</td>
<td>MDT</td>
<td>3.10</td>
</tr>
<tr>
<td>July</td>
<td>Dheeraj Jamwal (Intel)</td>
<td>50</td>
<td>DRM/i915</td>
<td>3.10</td>
</tr>
<tr>
<td>July</td>
<td>Damian (Renesas)</td>
<td>16</td>
<td>smack, security</td>
<td>3.10</td>
</tr>
<tr>
<td>August</td>
<td>Dheeraj Jamwal (Intel)</td>
<td>50</td>
<td>drm/i915</td>
<td>3.10</td>
</tr>
<tr>
<td>August</td>
<td>Dheeraj Jamwal (Intel)</td>
<td>41</td>
<td>drm/i915</td>
<td>3.10</td>
</tr>
<tr>
<td>August</td>
<td>Simon Horman (Renesas)</td>
<td>814</td>
<td>backport for LTSI-3.14</td>
<td>3.14</td>
</tr>
</tbody>
</table>

We noticed still many patches are sent to released LTSI-3.10. OK, we have created a method to merge late comer patches, however...we may need to discuss this further.
LTSI kernel release cadence, current shape and the future
kernel selection procedure (distro, LTS and LTSI)

- kernel selection
  - distribution?
    - yes
    - choose LTS
      - satisfy?
        - yes
        - use LTS
        - no
        - check LTSI
          - satisfy?
            - yes
            - use LTSI
            - no
            - use distro
    - no
  - no
    - cook LTSI
    - use LTSI
LTSI kernel cooking

- **you want to enhance LTSI**
  - **in newer kernel?**
    - yes
      - backport from upstream
    - no
      - off-tree patch?
        - yes
          - collect off-tree patches
        - no
          - write your own code
          - write yocto recipe for automatic merge
          - get custom LTSI
Adding some late comer LTSI patches to create BSP
You may want to **add new patches to released LTSI**.

Then you sent a patch to LTSI, but it cannot be merged.

Patchwork can be the way to **collect such off-tree patches**.

https://patchwork.kernel.org/project/ltsi-dev/list/
You can cherry pick patch from LTSI-patchwork site

- Patchwork automatically collect message that contains source code (patch)
- Each patch has unique tag and you can identify patch by tag
- You can write yocto recipe to collect patches in patchwork
Yocto meta file contains .bb (recipe) file

munakata@mythen:/Download/meta-renesas-20130204$ tree recipes-kernel/
recipes-kernel/
|-- linux
|  |-- files
|  |  |-- linux-yocto
|  |     |-- armadillo800eva
|  |     |     |-- armadillo800eva-non_hardware.cfg
|  |     |     |-- armadillo800eva-preempt-rt.scc
|  |     |     |-- armadillo800eva-standard.scc
|  |     |     |-- armadillo800eva.cfg
|  |     |     |-- armadillo800eva.scc
|  |     |     |-- defconfig
|  |     |     |-- missing_required.cfg
|  |     |     |-- required_redefinition.txt
|  |     |     |-- specified_non_hdw.cfg
|  |     |     |-- user-config.cfg
|  |     |     |-- user-patches.scc
|  |     |-- linux-yocto_3.4.bbappend
|  |-- linux-libc-headers
|     |-- linux-libc-headers-rmobile_git.bb

.bbappend can contain a pointer to LTSI off-tree patch
Edit recipe to merge LTSI-patchwork off-tree patch

diff --git a/recipes-kernel/linux/linux-yocto_3.4.bbappend b/recipes-kernel/linux/linux-yocto_3.4.bbappend
index 819c65a..0b89004 100644
--- a/recipes-kernel/linux/linux-yocto_3.4.bbappend
+++ b/recipes-kernel/linux/linux-yocto_3.4.bbappend
@@ -19,7 +19,10 @@ SRC_URI_append_armadillo800eva = ``
file://missing_required.cfg
file://required_redefinition.txt
file://specified_non_hdw.cfg

+ https://patchwork.kernel.org/patch/1132821/mbox/;
name=patch1;
downloadfilename=patch-1132821.patch;
apply=yes;
striplevel=1 \
''
+SRC_URI[patch1.md5sum] = ``c5e868f90629a56964c2c6ee731ba1cf''
+SRC_URI[patch1.sha256sum] = ``ea5f81ba7b91c0a1086f7c58f92a9818bae46615c5826aacba842c2aac5222''

COMPATIBLE_MACHINE_armadillo800eva= ``armadillo800eva''
KBRANCH_DEFAULT_armadillo800eva = ``armadillo800eva''

download off-tree patch from patchwork site and apply
Description of patchwork integration recipe

+https://patchwork.kernel.org/patch/1132821/mbox/;

name=patch1;
downloadfilename=patch-1132821.patch;
apply=yes;
striplevel=1

+SRC_URI[patch1.md5sum] =
+SRC_URI[patch1.sha256sum] =

- Define patchwork URI
- You need to calculate SUM after file download (md5 and sha256)
New LTS to LTSI update reflection cycle

Every stable update will be ported to existing LTSI code
(Discussion) Shall we have more merge periods?
LTSI use case confirmation
(Discussion) We want to hear user's voice

- Demanded feature
  - RT patches set
  - off tree utilities
  - (part of) safety features (like IEC61506, ISO2626)

- Others
LTSI Test Discussion
Why **LTSI kernel validation** becomes important?

- Upstream LTS is managed to be completely safe.
- LTSI can based on community LTS kernel, and
- LTSI is the place to add various NEW things
  - Feature back port from latest mainline (relatively safe)
  - Industry demanded not-mainlined (but commonly used) open source project code
  - Privately maintained bug-fix code (may be valuable)
  - Privately developed feature code

We want to validate LTSI kernel does not include any bug or regression against the community LTS code
Beyond the LTS(I) kernel use, share the test case!

New value opportunity of sharing the kernel test case

- Now many industry start using LTS and LTSI kernel.
- Each company may spend a lot of time for validation.
- Some of fundamental kernel feature test might be duplicated
  - common kernel function test (detail later)
  - common kernel benchmark test (detail later)
  - common compatibility conformance test
- Now we can consider sharing the (part of) kernel test case on top of LTS(I) kernel across the industry.
- We need to assign appropriate OSS license to each test case itself so the we can share them.
# Design target of shared LTSI test environment

## Feature
- Fully automated execution (nightly run)
- Easy to manage operation (add/edit test case)
- Trend monitoring capability (to catch the regression)
- User friendly interface (web access, GUI front end)

## Operation
- Local text execution *(can install to your computer)*
- Test case sharing mechanism
- Test result sharing mechanism (future work)
- Can penetrate to the upstream kernel development use
current shape -1/2

- Public tree to download whole test environment
  - [link]
  - https://bitbucket.org/cogentembedded/jta-public/

- Initial documentation
  - [link]

- Reports (automated)
  - [link]
Installation and update scripts (Debian only)

More tests integrated (including Renesas evaluation board-specific tests)

Misc. enhancements (e.g. error reporting)
Releasing beta version test suit

https://bitbucket.org/cogentembedded/jta-public/
LTSI kernel testing *(new/interesting bugs)*

- File system robustness/power-cycle tolerance test
  - ext3 (with misc combination of options, e.g. data=journal) behaves better than ext4, btrfs, etc. (with misc. options evaluated)
  - Example: ext4 failures occurred after power outages during fsstresstest run

- Need to pay attention for file system robustness and tolerance
Next steps

- Public server 24h/7d up/running with LTSI kernels for selected hardware (Intel Minnow, Renesas Henninger)
- More I/O and platform-specific tests
- Polished docs, deployment/installation scripts
Public server 24h/7d up/running with LTSI kernels

http://145.255.234.170/
Test case enhancement case-1 *(Fujitsu : Ethertool)*

Fujitsu added **Ethertool** test in their environment
ethtool - utility for controlling network drivers and hardware

Introduction

ethtool is the standard Linux utility for controlling network drivers and hardware, particularly for wired Ethernet devices. It can be used to:

- Get identification and diagnostic information
- Get extended device statistics
- Control speed, duplex, autonegotiation and flow control for Ethernet devices
- Control checksum offload and other hardware offload features
- Control DMA ring sizes and interrupt moderation
- Control receive queue selection for multiqueue devices
- Upgrade firmware in flash memory

Most features are dependent on support in the specific driver. See the manual page for full information.

Bugs

Bug reports should be sent to the maintainer, Ben Hutchings <bwh@kernel.org>, and to the netdev mailing list <netdev@vger.kernel.org>.

Development

See the development page.

https://www.kernel.org/pub/software/network/ethtool/
Test case enhancement case-2 (Renesas: driver)

Renesas added device driver test in our environment
Test case enhancement case-2 (Renesas: driver)

Target Script work log

write test for /dev/mmcbk0p2 (bs=1k count=1)
Test that device exists
Write random data to test file
1+0 records in
1+0 records out
1024 bytes (1.0 kB) copied, 0.0005902 s, 1.7 MB/s
Write test data to device
1+0 records in
1+0 records out
1024 bytes (1.0 kB) copied, 0.0054583 s, 188 kB/s
Read test data from device
1+0 records in
1+0 records out
1024 bytes (1.0 kB) copied, 0.0028504 s, 359 kB/s
Compare data written to data read
Test passed

Parser.py work log

raw_values[9] = 1024 bytes (1.0 kB) copied, 0.0054583 s, 188 kB/s
cur_dict['write'] = 188 kB/s
raw_values[13] = 1024 bytes (1.0 kB) copied, 0.0028504 s, 359 kB/s
cur_dict['read'] = 359 kB/s

Reference.log work log

Write
['188', 'kB/s']
188 0
For test write current value is 188, reference value - 0. Result - OK.
Comparison criteria is "greater or equal".

Read
['359', 'kB/s']
359 0
For test read current value is 359, reference value - 0. Result - OK.
Comparison criteria is "greater or equal".

capture -> transfer -> evaluate -> report
Test result aggregation

- **Multiple instances** of test frameworks
  (+targets, tests, configurations, parameters, bootcode/kernel/userspace combinations)

- How to **aggregate/process**
  (e.g. compare results, identify anomalies, remove duplicates)?

  - Step 1. Local anomalies/bugs can be handled/stored in centralized bugzilla-like system
  
  - Step 2. Test results can be processed/converted into a database, with proper indexing/parameterization (e.g. company/node reporting results, kernel version/patch level, tag/branch of test repository, etc)
Sharing the validation result (option 1)

- So far we have identified similar project/solution `openbenchmarking.org` that may be reused (still not 100% sure)

- Which came from Phoronix project (nice set of benchmarks)

- We will study if openbenchmarking.org infrastructure could be reused

- And, contact maintainer
MORE THAN 694,973 TEST RESULTS
AN OPEN, COLLABORATIVE TESTING PLATFORM

OPEN BENCHMARKING

AUTOMATED TESTING & BENCHMARKING ON AN OPEN PLATFORM

OpenBenchmarking.org is an open, collaborative testing platform designed by Phoronix Media and the developers behind the Phoronix Test Suite, the most comprehensive benchmarking platform for Linux and other operating systems. OpenBenchmarking.org makes the Phoronix Test Suite an even more extensible platform for conducting automated tests with complete integration into Phoronix Test Suite 3.0-Ireland as well as within Phoronomatic, an online test remote management system designed for managing test farms within enterprise environments.

The OpenBenchmarking.org infrastructure provides public and private storage of test result data (including system logs) and effective collaboration tools for sharing results and efficiently comparing multiple test result sets. OpenBenchmarking.org also provides a package management system for making accessible new, updated, and third-party test profiles / suites to the users of the Phoronix Test Suite.

Read more about the advanced capabilities provided by OpenBenchmarking.org on the features page. There is also a welcome letter from Michael Larabel, the lead developer of the Phoronix Test Suite and OpenBenchmarking.org, and the OpenBenchmarking.org blog.
Open-Source Benchmarking

The Phoronix Test Suite is the most comprehensive testing and benchmarking platform available that provides an extensible framework for which new tests can be easily added. The software is designed to effectively carry out both qualitative and quantitative benchmarks in a clean, reproducible, and easy-to-use manner.

The Phoronix Test Suite is based upon the extensive testing and internal tools developed by Phoronix.com since 2004 along with support from leading tier-one computer hardware and software vendors. This software is open-source and licensed under the GNU GPL.

Originally developed for automated Linux testing, support to the Phoronix Test Suite has since been added for Apple OS X, Microsoft Windows, BSD, and Solaris operating systems, among other POSIX compliant platforms such as GNU Hurd. The Phoronix Test Suite consists of a lightweight processing core (*pts-core*) with each benchmark consisting of an XML-based profile and related resource scripts. The process from the benchmark installation, to the actual benchmarking, to the parsing of important hardware and software components is heavily automated and completely repeatable, asking users only for confirmation of actions.

The Phoronix Test Suite interfaces with OpenBenchmarking.org as a collaborative web platform for the centralized storage of test results, sharing of test profiles and results, advanced analytical features, and other functionality. Phoromatic is an enterprise component to orchestrate test execution across multiple systems with remote management capabilities.
Sharing the validation result (option 2)

- Alternatively - we could just start with a database, that is filled in (in automated way) based on reports (xml reports) coming from each test environment setup/system.

- As for front-end/easy search/visualization - could be simple html front-end, tied with database search (there are open source frameworks available for that)

- If everyone has its own test version, test name, configuration, etc. (kernel version, patch/level, board/soc/ipblock revision, etc.), we would need to create formal identifiers/parameters for integration database (e.g. for search, index, etc.)
(Discussion) Sharing the test case and result
Resources
Resources = ltsi.linuxfoundation.org

- ML
  - ML subscription = https://lists.linuxfoundation.org/mailman/listinfo/ltsi-dev
  - ML archives = http://lists.linuxfoundation.org/pipermail/ltsi-dev/
  - ML patchwork = https://patchwork.kernel.org/project/ltsi-dev/list/
- git(each patch) = http://git.linuxfoundation.org/?p=ltsi-kernel.git;a=summary
- download (tar ball) = http://ltsi.linuxfoundation.org/downloads/releases
- twitter = @LinuxLTSI
- document archives = http://ltsi.linuxfoundation.org/resources