LTSI Project update

Long Term Support Initiative

Tsugikazu SHIBATA, NEC
Hisao Munakata, Renessas
14, October 2014
Embedded Linux Conference Europe@Dusseldorf
Who am I

Tsugikazu Shibata, NEC
Project lead of LTSI
Member of CE Working group
Member of the board of the Linux Foundation
Key activities of LTSI
Long Term Support Initiative

- LTSI had been started 2011 in Prague
- After long discussion of industry demand and how to solved

- Provide a industry managed kernel and maintain Long Term
- Provide a common place for embedded industry to share information: Mailing list, Workshop ...
- Provide place to support upstream activity : Help industry developer
Contents

• Linux Development process and status
• LTSI Development process
• LTSI New challenges : TEST Project
Status of Latest Linux Kernel

• Latest Kernel : 3.17
  – Released October 5th
  – Lines of code : 18,864,486
  – Files : 47,490

• Current Stable Kernels: 3.16.5

• Current development kernel: 3.18-rc1 is not yet released
Linux Development process

• Just after the release of 3.n, two weeks of merge window will be opened for proposal of new features
• After 2 weeks of merge window, -rc1 will be released and the stabilization will be started
• 3.n+1 will be released when it becomes reasonably stable by some of –rcX released
Kernel release cycle

- Release cycle of Linux kernel is about 60-70 days

<table>
<thead>
<tr>
<th>Version</th>
<th>Release</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.0</td>
<td>2011-7-21</td>
<td></td>
</tr>
<tr>
<td>3.1</td>
<td>2011-10-24</td>
<td>95</td>
</tr>
<tr>
<td>3.2</td>
<td>2012-1-4</td>
<td>72</td>
</tr>
<tr>
<td>3.3</td>
<td>2012-3-18</td>
<td>74</td>
</tr>
<tr>
<td>3.4</td>
<td>2012-5-20</td>
<td>63</td>
</tr>
<tr>
<td>3.5</td>
<td>2012-7-21</td>
<td>62</td>
</tr>
<tr>
<td>3.6</td>
<td>2012-9-30</td>
<td>71</td>
</tr>
<tr>
<td>3.5</td>
<td>2012-12-10</td>
<td>71</td>
</tr>
<tr>
<td>3.8</td>
<td>2013-2-18</td>
<td>70</td>
</tr>
<tr>
<td>3.9</td>
<td>2013-4-28</td>
<td>69</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Version</th>
<th>Release</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.10</td>
<td>2013-6-30</td>
<td>63</td>
</tr>
<tr>
<td>3.11</td>
<td>2013-9-2</td>
<td>64</td>
</tr>
<tr>
<td>3.12</td>
<td>2013-11-15</td>
<td>74</td>
</tr>
<tr>
<td>3.13</td>
<td>2014-1-21</td>
<td>67</td>
</tr>
<tr>
<td>3.14</td>
<td>2014-3-30</td>
<td>68</td>
</tr>
<tr>
<td>3.15</td>
<td>2014-6-8</td>
<td>70</td>
</tr>
<tr>
<td>3.16</td>
<td>2014-8-3</td>
<td>56</td>
</tr>
<tr>
<td>3.17</td>
<td>2014-10-5</td>
<td>63</td>
</tr>
</tbody>
</table>
Stable kernel release

• 3 part version like 3.x.y
• Recommended branch for users who want the most recent stable kernel
• Contain small and critical fixes for security problems or significant regressions discovered in a latest development version
• Bug fixes/Security fixes are back ported from latest version
• Drop when next stable kernel were released
Stable_kernel_rules.txt

• Published in 2006. Strict rule to back port from latest version

See Documentation/stable_kernel_rules.txt

- It must be obviously correct and tested.
- It cannot be bigger than 100 lines, with context.
- It must fix only one thing.
- It must fix a real bug that bothers people.
- It must fix a problem that causes a build error, an oops, a hang, data corruption, a real security issue, In short, something critical.
- Serious issues as reported by a user may also be considered.
- New device IDs and quirks are also accepted.
- No "theoretical race condition" issues, unless an explanation of how the race can be exploited is also provided.
- It cannot contain any "trivial" fixes in it.
- It or an equivalent fix must already exist in Linus' tree (upstream). ……
LongTerm Stable Kernel

• Extended maintenance period for stable kernel
• Pick one version per year and maintain 2 years
• Bug and security fixes found in latest version are continued to back ported
LTS includes large number of fixes

- Stable release only back port important bug and security fixes
- Thousands of fixes are there

<table>
<thead>
<tr>
<th>Version</th>
<th># of commits</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td>To</td>
</tr>
<tr>
<td>3.0</td>
<td>3.0.101</td>
</tr>
<tr>
<td>3.1</td>
<td>3.1.10</td>
</tr>
<tr>
<td>3.2</td>
<td>3.2.62</td>
</tr>
<tr>
<td>3.3</td>
<td>3.3.8</td>
</tr>
<tr>
<td>3.4</td>
<td>3.4.103</td>
</tr>
<tr>
<td>3.5</td>
<td>3.5.7</td>
</tr>
<tr>
<td>3.6</td>
<td>3.6.11</td>
</tr>
<tr>
<td>3.7</td>
<td>3.7.10</td>
</tr>
<tr>
<td>3.8</td>
<td>3.8.13</td>
</tr>
<tr>
<td>3.9</td>
<td>3.9.11</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Version</th>
<th># of commits</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td>To</td>
</tr>
<tr>
<td>3.10</td>
<td>3.10.53</td>
</tr>
<tr>
<td>3.11</td>
<td>3.11.10</td>
</tr>
<tr>
<td>3.12</td>
<td>3.12.26</td>
</tr>
<tr>
<td>3.13</td>
<td>3.13.11</td>
</tr>
<tr>
<td>3.14</td>
<td>3.14.17</td>
</tr>
<tr>
<td>3.15</td>
<td>3.15.10</td>
</tr>
<tr>
<td>3.16</td>
<td>3.16.1</td>
</tr>
</tbody>
</table>

2014/10/11
# Current LTS versions

<table>
<thead>
<tr>
<th>Version</th>
<th>Maintainer</th>
<th>Released</th>
<th>Projected EOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.10</td>
<td>Greg Kroah-Hartman</td>
<td>2013-6-30</td>
<td>Sep, 2015</td>
</tr>
<tr>
<td>3.4</td>
<td>Li Zefan</td>
<td>2012-5-20</td>
<td>Sep, 2016</td>
</tr>
<tr>
<td>3.2</td>
<td>Ben Hutchings</td>
<td>2012-1-4</td>
<td>2016</td>
</tr>
<tr>
<td>2.6.32</td>
<td>Willy Tarreau</td>
<td>2009-12-3</td>
<td>Mid-2015</td>
</tr>
</tbody>
</table>

[https://kernel.googlesource.com/pub/scm/docs/kernel/website/](https://kernel.googlesource.com/pub/scm/docs/kernel/website/)
LTSI DEVELOPMENT PROCESS
Discussing about process as;

- Preparation
- Merge window
- Validation
- Maintenance
Decide LTS version

Decide LTSI development schedule

Before the merge window, developer should;

- Merge own patches into upstream to prepare merge into LTSI. That will be easier to maintain production kernel (ie. Driver)
- Prepare to back port important features in the upstream to LTSI
• Merge window will open after 4-5 month of LTS release
• Anyone can propose patches in the merge window period
• Patches should be in the upstream or at least in the –next
• Exceptional case: beneficial for the wide range of users, or projected to be in the upstream
• Merge window period is about two month and –rc1 will be released right after the window was closed
• No additional patches are accepted after the window closed
LTSI process: Validation

- After release of –rc1, developer should validate whether its features were correctly merged and worked fine
- Patches to fix problem should be sent
- Validation period is about a month or more
- LTSI development will be finished when maintainer recognized reasonably stable and then new LTSI kernel will be released
- LTS is continue to maintain in two years since that version was released
- LTSI will be rebased at the same time each new LTS was released
  - If you want to create own kernel based on LTS such as 3.10.x, you will be able to find same version in LTSI git tree
  - There are tool to create LTSI full kernel source in “scripts” directory
- LTSI full source tarball will be released every 3month after QA team have done its test as a volunteer work
LTSI: maintenance stream

Every 3 months, LTSI will test the re-based kernel and release a Tar-ball (LTSI-Release)

Re-Base LTSI at Every LTS Release

LTS

LTSI

LTSI

LTSI

LTSI-3.n

LTSI-3.n+1

LTSI-3.n+2

LTSI-3.n+3

LTSI-3.n

LTSI-3.n+1

LTSI-3.n+2

LTSI-3.n+3

LTSI-3.n Tar-Ball

LTSI-3.n+3 Tar-Ball
Patchwork is ready

- Patchwork is a tool to cherry pick patches posted on ML.
- LTSI is providing patchwork.
- In case you still have patches after closed the merge window, you can send patch to LTSI ML anytime.
- Using Patchwork, you can share patches even after the release.

https://patchwork.kernel.org/project/ltsi-dev/list/
How to create own BSP

LTSI-3.10 development (backport) target version

3.10 → 3.11 → 3.12 → 3.13 → 3.14

- feature A
- feature B
- feature C

merged to the LTSI release

LTSI-3.10.xx release

- LTS fix
- LTS update

LTSI-3.10.YY update

LTSI-3.10.ZZ update

cherry pick from patchwork

custom recipe to build own BSP

latecomer extra patches

bug fix A
fratumure D
<table>
<thead>
<tr>
<th>Event</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 3.10 becomes LTS</td>
<td>2013.8.4</td>
</tr>
<tr>
<td>LTSI-3.10 merge window open</td>
<td>2013.11.13</td>
</tr>
<tr>
<td>Merge Window 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>
# Major contributors for LTSI-3.10

<table>
<thead>
<tr>
<th>Contributor</th>
<th>Patch count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Darren Hart (Intel)</td>
<td>1,197</td>
</tr>
<tr>
<td>Simon Horman (for Renesas)</td>
<td>1,122</td>
</tr>
<tr>
<td>Daniel Sangorlin (Toshiba)</td>
<td>123</td>
</tr>
<tr>
<td>Patrik Jakobsson (for Intel)</td>
<td>46</td>
</tr>
<tr>
<td>Mark Brown (linaro)</td>
<td>11</td>
</tr>
<tr>
<td>Greg Kroah-Hartman (Linux Foundation)</td>
<td>11</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>2,510</strong></td>
</tr>
</tbody>
</table>
Major achievement of LTSI-3.10

• LTTng
• Power efficient work queues
• Intel BayTrail support
• Intel Minnow board support
• Renesas R-Car H2/M2 series support
• Xilinx Zinq board support
# LTSI-3.14 Release plan

<table>
<thead>
<tr>
<th>Event</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel 3.14 merge window open</td>
<td>2014.1.19</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>3.14 becomes LTS</td>
<td>2014.7.3</td>
</tr>
<tr>
<td><strong>LTSI-3.14 merge window open</strong></td>
<td><strong>2014.8.21</strong></td>
</tr>
<tr>
<td><strong>Merge window period</strong></td>
<td><strong>75 days</strong></td>
</tr>
<tr>
<td><strong>LTSI-3.14-rc1</strong> (=merge window close)</td>
<td>2014.10.30?</td>
</tr>
<tr>
<td><strong>Validation period</strong></td>
<td><strong>50+ days</strong></td>
</tr>
<tr>
<td><strong>LTSI-3.14 release</strong></td>
<td>2014.12.25?</td>
</tr>
</tbody>
</table>

*Please send patch to LTSI mailing list now!*
LTSI NEW CHALLENGES
TEST PROJECT
• LTSI is patch set added on top of LTS
• LTS is quite stable because of picking patches with strict rule (stable_kernel_rules.txt)
• However, LTSI will include
  – Back port features in latest upstream
  – Industry demanded non mainlined code

We want to validate LTSI kernel that does not include any bug or regression against LTS
LTSI Test : Share test case!

- Now LTS and LTSI is used by lots of companies
- Each companies may spend a lot of time for its validation
- Some of fundamental “feature test” might be duplicated and such portion might be shared

We can consider sharing some part of kernel test across the industry
- Test environment
- Test case
LTSI Test environment

• We have developed LTSI test environment
  – Fully automated execution
  – Easy to manage
  – Monitoring what have happened
  – GUI
  – Deliverable (Be able to install each companies)
  – 28 benchmarks and 33 test programs are integrated in initial test environment
LTSI Test environment

Current stage is just started but You can down load from:
https://bitbucket.org/cogentembedded/jta-public/

https://bitbucket.org/cogentembedded/jta-public/
LTSI Test environment

• Initial documentation:
  
  https://bytebucket.org/cogentembedded/jta-public/raw/7cefe53a09b5028bf2c99663d81ecde39b486713/docs/jta-guide.pdf

Your feedback is welcome!
NEXT STEP for LTSI Test environment

• Daily/Nightly test of LTSI for specific hardware
  – Public server 24h/7d up/running for LTSI testing with Intel Minnow, Renesas Henninger
• More I/O and platform-specific tests
• Polished docs, deployment/installation scripts
You will be able to refer its result

http://145.255.234.170/
Test case sharing

• Held discussion at LTSI workshop in Chicago
• What we have discussed:
  – License to deliver; strong or weak?
  – How the test cases are maintained; maintainer or just gather
  – How we manage test cases;
    • Collect test case and Referencing test cases

Further discussion will be happen in our workshop, please join us!
THANK YOU
You can participate LTSI

• Follow on Twitter account:
  @LinuxLTSI

• Web:
  http://ltsi.linuxfoundation.org

• Mailing list:
  https://lists.linuxfoundation.org/mailman/listinfo/ltsi-dev

• Git tree:
  http://git.linuxfoundation.org/?p=ltsi-kernel.git;a=summary