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)

item	date
kernel 3.10 merge window open	2013.4.28
kernel 3.10 merge window close	2013.5.12
kernel 3.10 release	2013.6.30
Announce of 2013 LTS kernel version	2013.8.4
LTSI-3.10 git tree open	2013.9.11
3.10 becomes LTS (=3.12 release)	2013.11.15
LTSI-3.10 merge window open	2013.11.15
patch collection period	75 days
LTSI-3.10-rc1 (=merge window close)	2014.1.29
validation period	26 days
LTSI-3.10 release	2014.2.24

LTSI 3.14 development schedule (underway)

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

Please be sure not to late LTSI 3.14 merge window close!

Patch submission status after LTSI-3.10 release

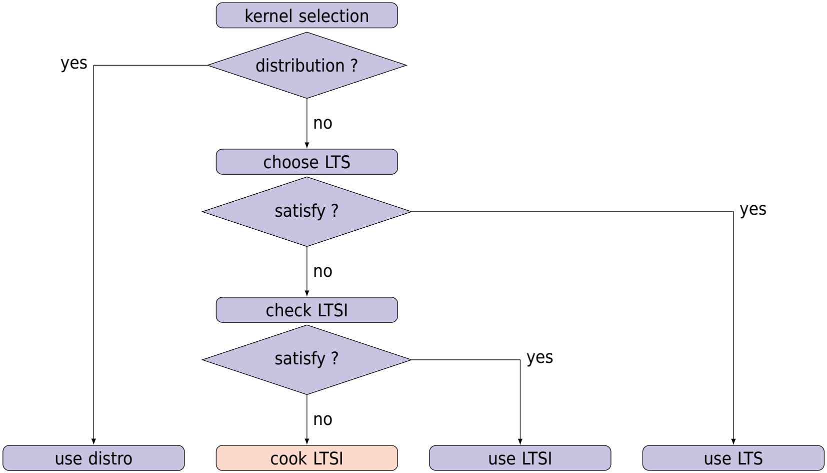
Month	Sender (company)	number	contents	target
March	Adrian Hunter (Intel)	2	MMC	3.10
March	Wei.sern.chan (Intel)	5	PWM	3.10
April	Simon Horman (Renesas)	10	serial	3.10
April	Simon Horman (Renesas)	1	USB	3.10
May	Geert Uytterhoeven (Renesas)	1	MDT	3.10
July	Dheeraj Jamwal (Intel)	50	DRM/i915	3.10
July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

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

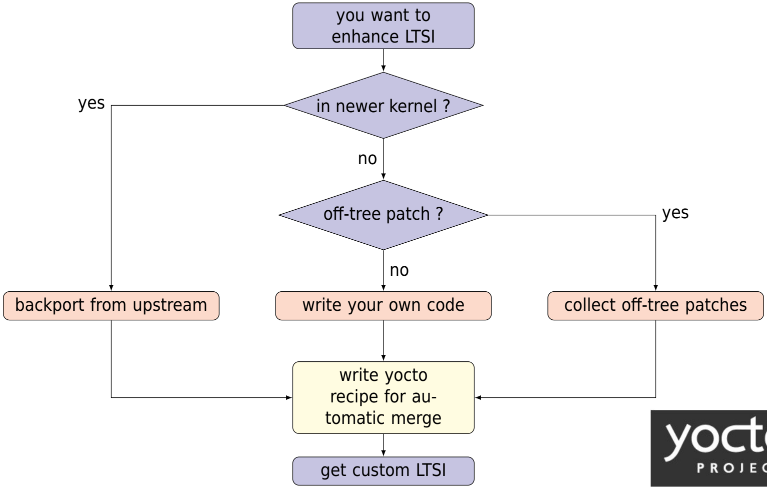
LONG TERM SUPPORT INITIATIVE

LTSI kernel release cadence, current shape and the future

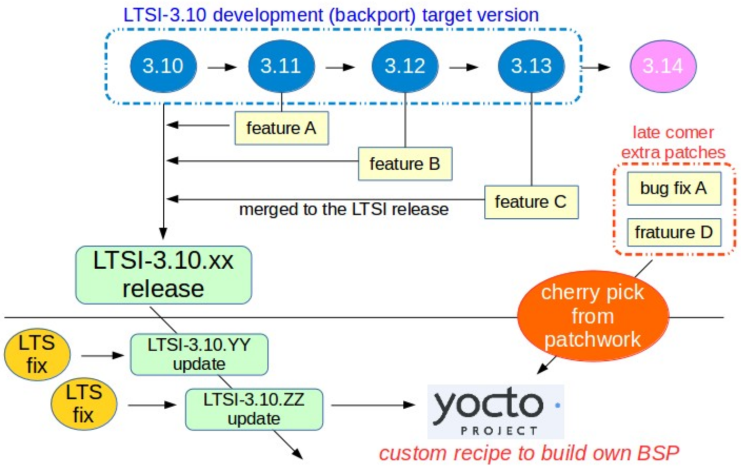
kernel selection procedure (distro, LTS and LTSI)



LTSI kernel cooking

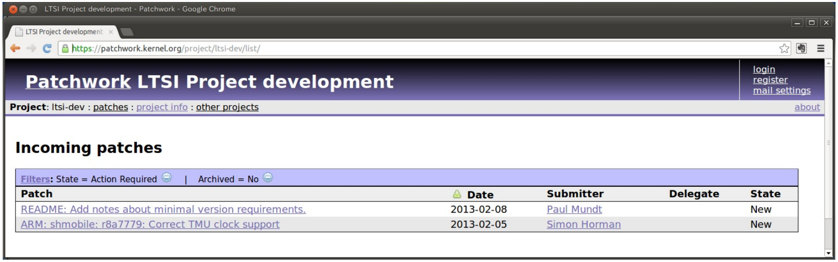


Adding some late comer LTSI patches to create BSP



LTSI patchwork

- You may want to add new patches to released LTSI.
- Then you sent a patch to LTSI, but it can not be merged.
- Patchwork can be the way to collect such off-tree patches.



LTSI Project development - Patchwork - Google Chrome

https://patchwork.kernel.org/project/ltsi-dev/list/

Patchwork LTSI Project development

login
register
mail settings

Project: ltsi-dev : patches : project info : other projects — about

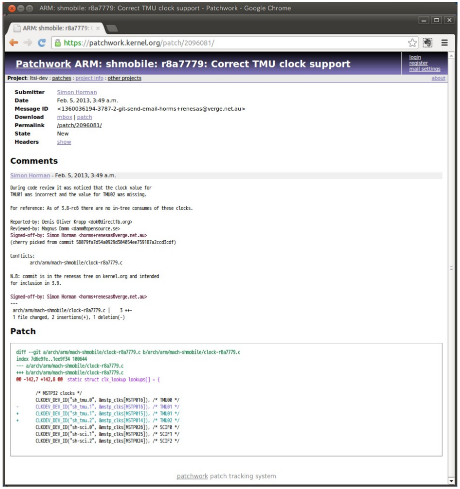
Incoming patches

Filters: State = Action Required | Archived = No

Patch	Date	Submitter	Delegate	State
README: Add notes about minimal version requirements.	2013-02-08	Paul Mundt		New
ARM: shmobile: r8a7779: Correct TMU clock support	2013-02-05	Simon Horman		New

https://patchwork.kernel.org/project/ltsi-dev/list/

You can cherry pick patch from LTSI-patchwork site


ARM: shmobile: r8a7779: Correct TMU clock support - Patchwork - Google Chrome

https://patchwork.kernel.org/patch/2096081/

Patchwork ARM: shmobile: r8a7779: Correct TMU clock support

login
register
mail settings

Project: ltsi-dev : patches : project info : other projects — about

Submitter	Simon Horman
Date	Feb. 5, 2013, 3:49 a.m.
Message ID	<1360036194-3787-2-git-send-email-horms+renesas@verge.net.au>
Download	mbox | patch
Permalink	/patch/2096081/
State	New
Headers	show

Comments

Simon Horman - Feb. 5, 2013, 3:49 a.m.

```
During code review it was noticed that the clock value for
TMU01 was incorrect and the value for TMU02 was missing.

For reference: As of 3.8-rc6 there are no in-tree consumes of these clocks.

Reported-by: Denis Oliver Kropp <dok@directfb.org>
Reviewed-by: Magnus Damm <damm@opensource.se>
Signed-off-by: Simon Horman <horms+renesas@verge.net.au>
(cherry picked from commit 58079fa7d54a0929d304054ee759187a2ccd3cdf)

Conflicts:
	arch/arm/mach-shmobile/clock-r8a7779.c

N.B: commit is in the renesas tree on kernel.org and intended
for inclusion in 3.9.

Signed-off-by: Simon Horman <horms+renesas@verge.net.au>
---
 arch/arm/mach-shmobile/clock-r8a7779.c |    3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
```

Patch

```
diff --git a/arch/arm/mach-shmobile/clock-r8a7779.c b/arch/arm/mach-shmobile/clock-r8a7779.c
index 7d6e9fe..1ee9f34 100644
--- a/arch/arm/mach-shmobile/clock-r8a7779.c
+++ b/arch/arm/mach-shmobile/clock-r8a7779.c
@@ -142,7 +142,8 @@ static struct clk_lookup lookups[] = {

 	/* MSTP32 clocks */
 	CLKDEV_DEV_ID("sh_tmu.0", &mstp_clks[MSTP016]), /* TMU00 */
-	CLKDEV_DEV_ID("sh_tmu.1", &mstp_clks[MSTP016]), /* TMU01 */
+	CLKDEV_DEV_ID("sh_tmu.1", &mstp_clks[MSTP015]), /* TMU01 */
+	CLKDEV_DEV_ID("sh_tmu.2", &mstp_clks[MSTP014]), /* TMU02 */
 	CLKDEV_DEV_ID("sh-sci.0", &mstp_clks[MSTP026]), /* SCIF0 */
 	CLKDEV_DEV_ID("sh-sci.1", &mstp_clks[MSTP025]), /* SCIF1 */
 	CLKDEV_DEV_ID("sh-sci.2", &mstp_clks[MSTP024]), /* SCIF2 */
```

patchwork patch tracking system

- 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''
```

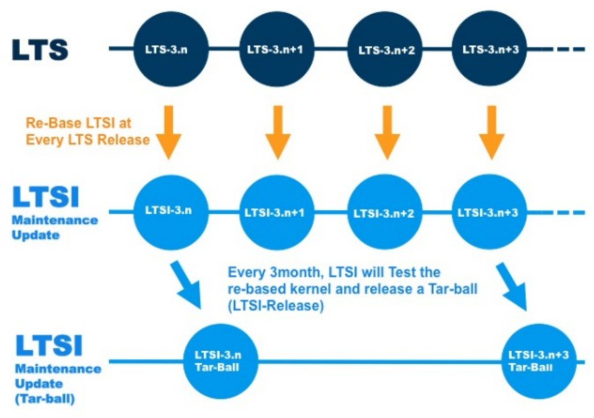
download off-tree patch from patchwork site and apply

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] =
```

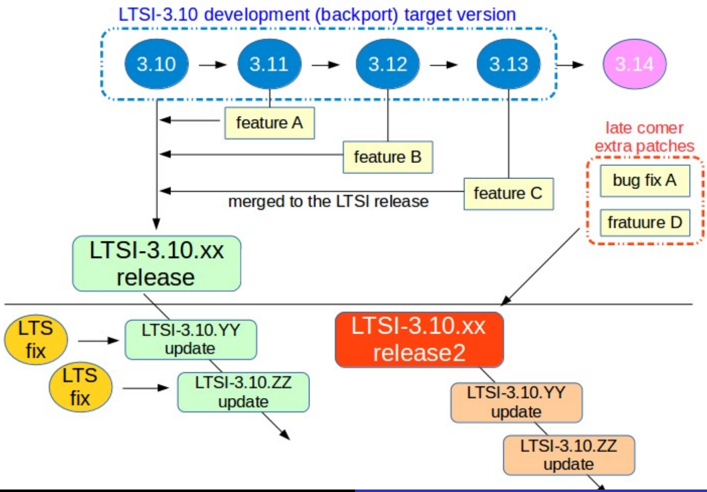
- Define patchwork URI
- You need to calculate SUM after file download (md5 and sha256)

New LTS to LTSI update reflection cycle


LTS
LTS-3.n
LTS-3.n+1
LTS-3.n+2
LTS-3.n+3
Re-Base LTSI at Every LTS Release
LTSI Maintenance Update
LTSI-3.n
LTSI-3.n+1
LTSI-3.n+2
LTSI-3.n+3
Every 3month, LTSI will Test the re-based kernel and release a Tar-ball (LTSI-Release)
LTSI Maintenance Update (Tar-ball)
LTSI-3.n Tar-Ball
LTSI-3.n+3 Tar-Ball

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 pacheset
  - off tree utilities
  - (part of) safty 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

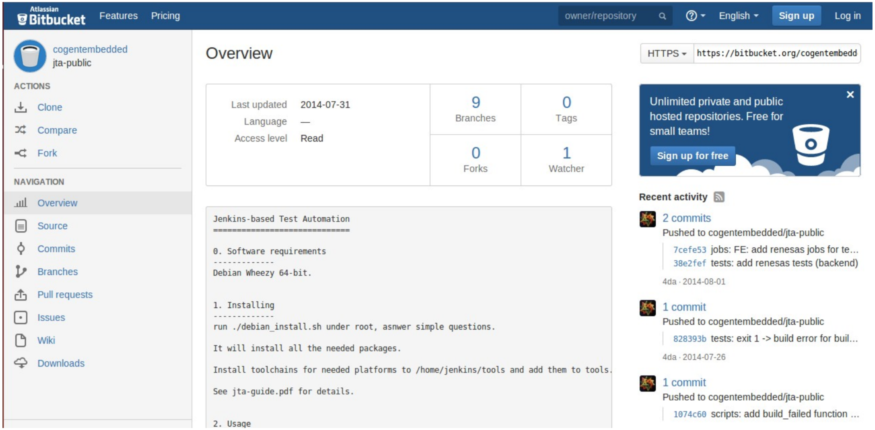
current shape -1/2

- Public tree to download whole test environment
  - [link]
  - https://bitbucket.org/cogentembedded/jta-public/
- Initial documentation
  - [link]
  - https://bytebucket.org/cogentembedded/jta-public/raw/7cefe53a09b5028bf2c99663d81ecde39b486713/docs/jta-guide.pdf
- Reports (automated)
  - [link]
  - http://145.255.234.170:8080/view/batch%20runs/job/Run%20SELECTED%20tests%20on%20SELECTED%20targets/ws/pdf_reports/minnow.2014-07-10_23-32-36.7.json.xml.pdf

current shape -2/2

- 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



Last updated	2014-07-31
Language	—
Access level	Read

9 Branches	0 Tags
0 Forks	1 Watcher

```
Jenkins-based Test Automation
==============================

0. Software requirements
-------------
Debian Wheezy 64-bit.


1. Installing
-------------
run ./debian_install.sh under root, asnwer simple questions.

It will install all the needed packages.

Install toolchains for needed platforms to /home/jenkins/tools and add them to tools.

See jta-guide.pdf for details.


2. Usage
```

Recent activity

2 commits
Pushed to cogentembedded/jta-public
7cefe53 jobs: FE: add renesas jobs for te...
38e2fef tests: add renesas tests (backend)
4da · 2014-08-01

1 commit
Pushed to cogentembedded/jta-public
828393b tests: exit 1 -> build error for buil...
4da · 2014-07-26

1 commit
Pushed to cogentembedded/jta-public
1074c60 scripts: add build_failed function ...

https://bitbucket.org/cogentembedded/jta-public/

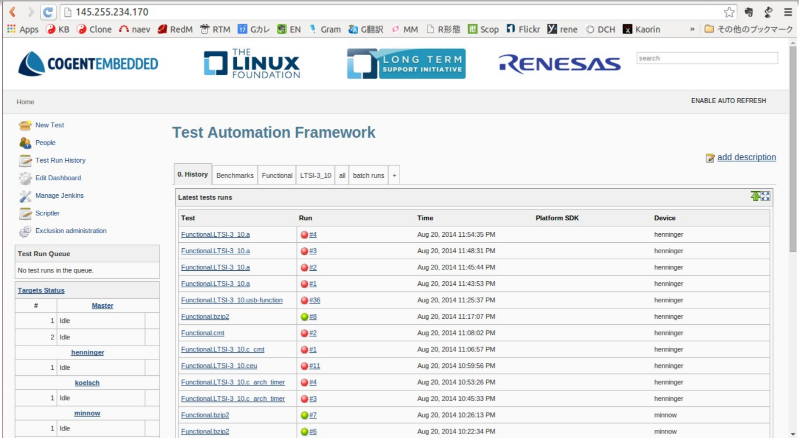
LTSI kernel testing (new/interesting bugs)

- 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 fsstress test 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


145.255.234.170

Home

ENABLE AUTO REFRESH

- New Test
- People
- Test Run History
- Edit Dashboard
- Manage Jenkins
- Scriptler
- Exclusion administration

Test Run Queue

No test runs in the queue.

Targets Status

#	Master	
1	Idle	
2	Idle	
	henninger	
1	Idle	
	koelsch	
1	Idle	
	minnow	
1	Idle	

Test Automation Framework

add description

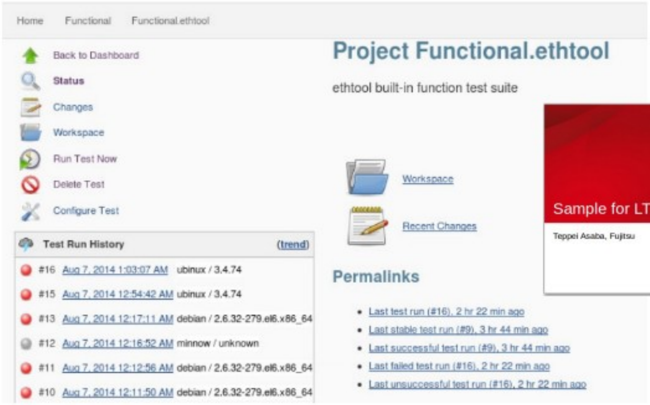
0. History | Benchmarks | Functional | LTSI-3_10 | all | batch runs | +

Latest tests runs

Test	Run	Time	Platform SDK	Device
Functional.LTSI-3_10.a	🔴 #4	Aug 20, 2014 11:54:35 PM		henninger
Functional.LTSI-3_10.a	🔴 #3	Aug 20, 2014 11:48:31 PM		henninger
Functional.LTSI-3_10.a	🔴 #2	Aug 20, 2014 11:45:44 PM		henninger
Functional.LTSI-3_10.a	🔴 #1	Aug 20, 2014 11:43:53 PM		henninger
Functional.LTSI-3_10.usb-function	🔴 #36	Aug 20, 2014 11:25:37 PM		henninger
Functional.bzip2	🟢 #8	Aug 20, 2014 11:17:07 PM		henninger
Functional.cmt	🔴 #2	Aug 20, 2014 11:08:02 PM		henninger
Functional.LTSI-3_10.c_cmt	🔴 #1	Aug 20, 2014 11:06:57 PM		henninger
Functional.LTSI-3_10.ceu	🔴 #11	Aug 20, 2014 10:59:56 PM		henninger
Functional.LTSI-3_10.c_arch_timer	🔴 #4	Aug 20, 2014 10:53:26 PM		henninger
Functional.LTSI-3_10.c_arch_timer	🔴 #3	Aug 20, 2014 10:45:33 PM		henninger
Functional.bzip2	🟢 #7	Aug 20, 2014 10:26:13 PM		minnow
Functional.bzip2	🟢 #6	Aug 20, 2014 10:22:34 PM		minnow

http://145.255.234.170/

Test case enhancement case-1 (Fujitsu : Ethertool)



Project Functional.ethtool

ethtool built-in function test suite

Test Run History (trend)

●	#16	Aug 7, 2014 1:03:07 AM	ubinux / 3.4.74
●	#15	Aug 7, 2014 12:54:42 AM	ubinux / 3.4.74
●	#13	Aug 7, 2014 12:17:11 AM	debian / 2.6.32-279.el6.x86_64
●	#12	Aug 7, 2014 12:16:52 AM	minnow / unknown
●	#11	Aug 7, 2014 12:12:56 AM	debian / 2.6.32-279.el6.x86_64
●	#10	Aug 7, 2014 12:11:50 AM	debian / 2.6.32-279.el6.x86_64

Permalinks

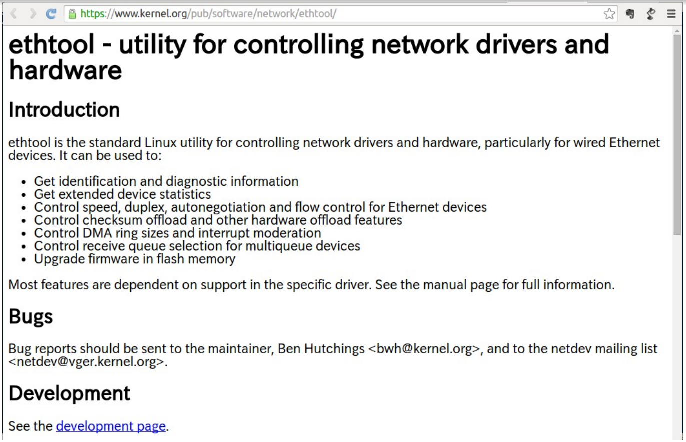
- Last test run (#16), 2 hr 22 min ago
- Last stable test run (#9), 3 hr 44 min ago
- Last successful test run (#9), 3 hr 44 min ago
- Last failed test run (#16), 2 hr 22 min ago
- Last unsuccessful test run (#16), 2 hr 22 min ago

Sample for LTSI workshop

Teppei Asaba, Fujitsu

Fujitsu added Ethertool test in their environment

ethtool


https://www.kernel.org/pub/software/network/ethtool/

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

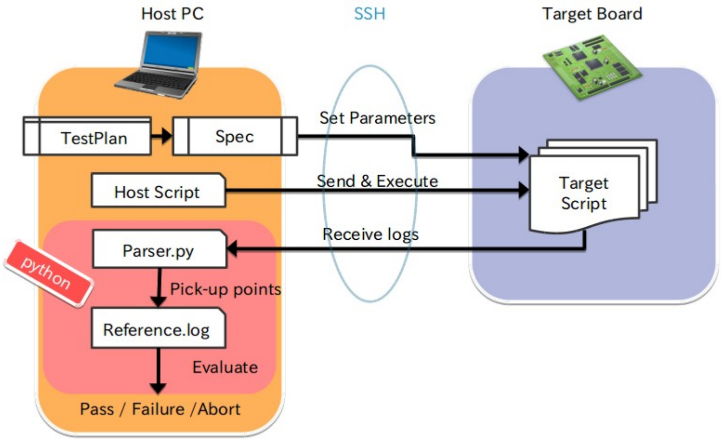
Bug reports should be sent to the maintainer, Ben Hutchings <bwh@kernel.org>, and to the netdev mailing list <netdev@vger.kernel.org>.

Development

See the development page.

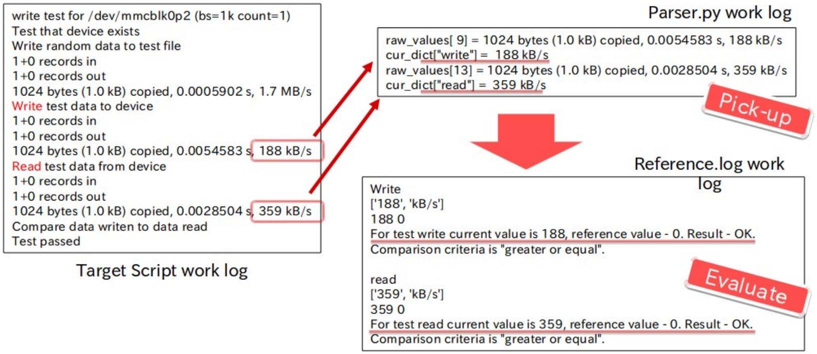
https://www.kernel.org/pub/software/network/ethtool/

Test case enhancement case-2 (Renesas: driver)



Renesas added device driver test in our environment

Test case enhancement case-2 (Renesas: driver)



```
write test for /dev/mmcblk0p2 (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 writen to data read
Test passed
```

Target Script work log

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

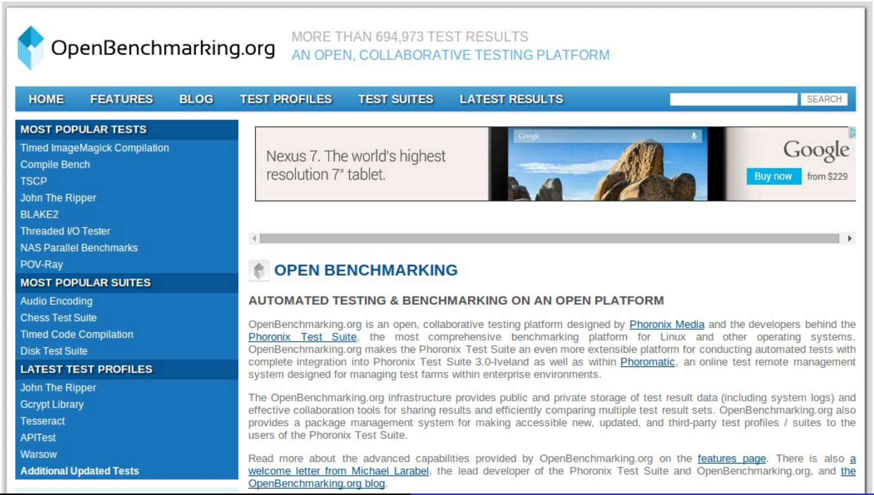
Test result aggregation

- 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

http://openbenchmarking.org/



MORE THAN 694,973 TEST RESULTS
AN OPEN, COLLABORATIVE TESTING PLATFORM

MOST POPULAR TESTS
- Timed ImageMagick Compilation
- Compile Bench
- TSCP
- John The Ripper
- BLAKE2
- Threaded I/O Tester
- NAS Parallel Benchmarks
- POV-Ray

MOST POPULAR SUITES
- Audio Encoding
- Chess Test Suite
- Timed Code Compilation
- Disk Test Suite

LATEST TEST PROFILES
- John The Ripper
- Gcrypt Library
- Tesseract
- APITest
- Warsow
- **Additional Updated Tests**

OPEN BENCHMARKING

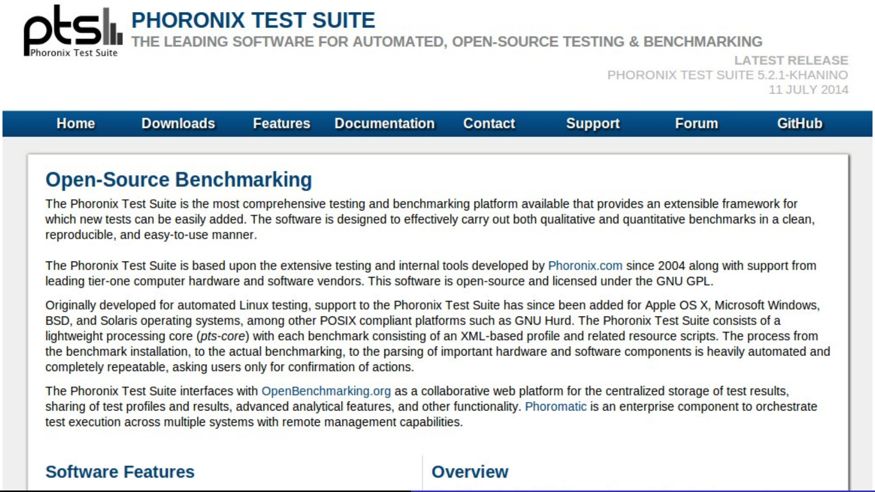
AUTOMATED TESTING & BENCHMARKING ON AN OPEN PLATFORM

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-Iveland as well as within Phoromatic, 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.

http://www.phoronix-test-suite.com/



PHORONIX TEST SUITE

THE LEADING SOFTWARE FOR AUTOMATED, OPEN-SOURCE TESTING & BENCHMARKING

LATEST RELEASE
PHORONIX TEST SUITE 5.2.1-KHANINO
11 JULY 2014

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.

Software Features

Overview

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

- LTSI process document (new) = http://ltsi.linuxfoundation.org/participate-in-ltsi/ltsi-development-guide
- 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