Merging your Kernel Testing Code into KernelCI

How to test your own kernel project with KernelCI
Who am I

Hirotaka Motai

- Software engineer focuses on Embedded Linux, RT Linux.
- CIP representative in Cybertrust Japan Co., Ltd.
Collaborative development with OSS projects

- mainline
- debian
- KernelCI

Upstream first

Use the upstream code

Contributions

Patch Review
CVE Check
Testing

CIP Open Source Base Layer (OSBL)

Use for Linux Product

Development
Test infrastructure, CVE reports

https://www.cip-project.org/
Who am I:

Alice Ferrazzi

Gentoo:
- Gentoo Kernel Project Leader
- GKernelCI (Gentoo Kernel testing tool) creator

KernelCI:
- TSC (Technical Steering Committee) member

CIP (Civil Infrastructure Platform):
- CIP Testing Working Group member

MIRACLE LINUX powered by Cybertrust Japan Co., Ltd.
- Software Engineer
- EMlinux embedded Linux distribution - Lead CI system development
Agenda

- KernelCI
  - What’s KernelCI?
  - Who is doing it
  - Why KernelCI is needed
  - KernelCI composition
  - KernelCI tests labs are distributed
  - Current available KernelCI test labs
  - Some of the current kernel tree tested by KernelCI
  - Implementing a framework on KernelCI
  - Merging kernel testing code into upstream KernelCI

- KernelCI native implementation
  - What is KernelCI native
  - Merging CIP testing framework into KernelCI native
  - CIP (Civil Infrastructure Platform)
  - CIP testing framework
  - CIP testing merged into KernelCI
  - Add CIP Kernel tree to KernelCI
  - KernelCI results emails

- Add CIP rootfs
- Add tests
- CIP Web Dashboard
- Still left work

- KCIDB
  - What is
  - Pros
  - Who is using KCIDB
  - Implementation
  - Sending results with KCIDB
  - KCIDB Dashboard

- Future
- Conclusion

- More about KernelCI
- More about Gentoo Kernel CI
- More about Civil Infrastructure Project
What’s KernelCI?

KernelCI is a community-based open source distributed test automation system focused on upstream kernel development.
Who is doing it

- **TSC (Technical Steering Committee)**
  - Formed by KernelCI core developers and maintainer
  - KernelCI development and maintenance
- **Advisory Board**
  - Premium organizations representatives involved in KernelCI
  - Manage budgets and help coordinating tasks
Why KernelCI is needed

KernelCI can be used and checked by anyone from mail reports and website.

- Website: [https://linux.kernelci.org/](https://linux.kernelci.org/)
- Results mailing list: [https://groups.io/g/kernelci-results/](https://groups.io/g/kernelci-results/)

KernelCI is mostly useful to anyone involved in Kernel testing and Kernel development. Gives you a tool suite with already implemented tests cases and variety of different unique boards.
KernelCI composition

- **KernelCI-core** core-tools
  - The main configuration and tools of KernelCI.
- **Backend** (currently rework in progress)
  - Provides the KernelCI web API [https://api.kernelci.org/](https://api.kernelci.org/)
- **Frontend**
  - Web dashboard showing the data available from the backend
- **Test-definitions**
  - Keep lava jobs test definitions
  - You add code here if you want to add new lava jobs tests to KernelCI
- **Lava-docker**
  - For making your own KernelCI LAVA testing laboratory (works with docker)
- **Jenkins**
  - Orchestrates builds and tests
- **KCIDB**
  - Tool to submit kernel test data
KernelCI native

Native tests from kernelci-core

LAVA labs
Static checks

Custom labs

KernelCI Backend

kci_data

KCIDB

Independent testing frameworks

Intel 0-Day
Linaro
SONY Fuego
yocto PROJECT

Your system here

arm

Google syzbot

work in progress
KernelCI tests labs are distributed

- KernelCI test labs (currently only LAVA lab) can be connected to KernelCI for adding new boards under test
- Any LAVA lab with a publicly available API can be added to KernelCI
Current available KernelCI test labs

- Lab-baylibre
- Lab-broonie
- Lab-cip
- Lab-clabbe
- Lab-collabora
- Lab-collabora-staging
- Lab-kontron
- Lab-linaro-lkft
- Lab-mhart
- Lab-nxp
- Lab-pengutronix
- Lab-theobroma-systems
- **Add your lab here!**
Some of the current kernel tree tested by KernelCI

- cip
- efi
- kselftest
- mainline
- net-next
- next
- rt-stable
- pm
- arm64
- amlogic

- clk
- chromeos
- android
- and more!
Framework definition

Testing framework including Kernel building, booting and testing code

For example the CIP (Civil Infrastructure Platform) project have its own testing framework for testing the CIP SLTS kernel tree.
Merging kernel testing code into upstream KernelCI

- **KernelCI native**
  - Directly collaborating with KernelCI code
  - For example, in the next slide we will explain how we merged CIP framework code into KernelCI native

- **KCIDB client**
  - Sharing kernel testing results
  - For example, we will explain how we manage to send Gentoo Linux kernel tests results into KernelCI common database

- KernelCI local alternative (please refer to [local development setup](#))
  - Fully local KernelCI deployment
KernelCI native implementation
What is KernelCI native

- KernelCI native is CI for automating building, booting and testing kernel trees
- KernelCI native repository are managed by the KernelCI TSC team
- KernelCI native tests jobs are in lava jobs format and generalized to run on different KernelCI LAVA lab environments
Pros

- Can reuse KernelCI tests tools and regressions check
- Usage of boards already connected through lava labs to KernelCI
Merging CIP testing framework into KernelCI native

- CIP decided to merge the current CIP testing framework into KernelCI native because of the KernelCI native pros
- In the next slides, we will explain what is CIP and how CIP managed to merge it
CIP (Civil Infrastructure Platform)

CIP is a Linux Foundation project that aims to establish a “base layer” of industrial-grade tooling using the Linux kernel and other open source projects.

https://www.cip-project.org/
CIP testing framework

- Using GitLab pipeline with LAVA CIP lab for building, booting and testing the SLTS and SLTS-RT Kernel
- LAVA tests like SMC (spectre meltdown testing) and IEC-62443-4-2 standard
- CIP rootfs isar-cip-core for user space testing
- Testing kernels with the configuration from cip-kernel-config
CIP merged into KernelCI

- Regressions tests mail and release testing mail
- Testing kernels with the configuration from cip-kernel-config
- Testing CIP core rootfs (isar-cip-core) with KernelCI
- Run CIP and KernelCI tests on CIP kernel (like Kselftest and LTP)
- CIP tests merged in KernelCI tests (like SMC spectre meltdown check)
- Automatic bisection of regression
- Testing CIP kernel in all the KernelCI testing boards
Add CIP Kernel tree to KernelCI

```yaml
trees:
  cip:
    url: "https://git.kernel.org/pub/scm/linux/kernel/git/cip/linux-cip.git"
```
Adding build configs allow the tree branch to be monitored and tested

```yaml
build_configs:
  cip_4.4:
    tree: cip
    branch: 'linux-4.4.y-cip'
    variants: *cip_variants_kselftest
  cip_4.4-rt:
    tree: cip
    branch: 'linux-4.4.y-cip-rt'
    variants: *cip_variants
```
KernelCI results emails

cip/linux-5.10.y-cip build: 184 builds: 3 failed, 181 passed, 4 errors, 10 warnings (v5.10.83-cip1)

Full Build Summary: https://kernelci.org/build/cip/branch/linux-5.10.y-cip/kernel/v5.10.83-cip1/

Tree: cip
Branch: linux-5.10.y-cip
Git Describe: v5.10.83-cip1
Git Commit: 2332f07a324fd78d7c7436deeed23cd7db441ea7
Git URL: https://git.kernel.org/pub/scm/linux/kernel/git/cip/linux-cip.git
Built: 7 unique architectures
If something broke -> regression email

cip/linux-4.19.y-cip baseline: 121 runs, 1 regressions (v4.19.217-cip62)

kernelci.org bot <bot@...>

cip/linux-4.19.y-cip baseline: 121 runs, 1 regressions (v4.19.217-cip62)

Regressions Summary

-------------------------
<table>
<thead>
<tr>
<th>platform</th>
<th>arch</th>
<th>lab</th>
<th>compiler</th>
<th>defconfig</th>
<th>regressions</th>
</tr>
</thead>
<tbody>
<tr>
<td>panda</td>
<td>arm</td>
<td>lab-collabora</td>
<td>gcc-10</td>
<td>omap2plus_defconfig</td>
<td>1</td>
</tr>
</tbody>
</table>


Test: baseline
Tree: cip
Branch: linux-4.19.y-cip
Describe: v4.19.217-cip62
URL: https://git.kernel.org/pub/scm/linux/kernel/git/cip/linux-cip.git
SHA: dc62e26e3be875a7324b85b8274c13a335e610dd
## Test Regressions

<table>
<thead>
<tr>
<th>platform</th>
<th>arch</th>
<th>lab-collabora</th>
<th>compiler</th>
<th>defconfig</th>
<th>regressions</th>
</tr>
</thead>
<tbody>
<tr>
<td>panda</td>
<td>arm</td>
<td></td>
<td>gcc-10</td>
<td>omap2plus_defconfig</td>
<td>1</td>
</tr>
</tbody>
</table>

### Details:
- [https://kernelci.org/test/plan/id/61a587a0ab3b0079bd18f6d7](https://kernelci.org/test/plan/id/61a587a0ab3b0079bd18f6d7)

### Results:
- 5 PASS, 1 FAIL, 0 SKIP

### Full config:
- omap2plus_defconfig

### Compiler:
- gcc-10 (arm-linux-gnueabihf-gcc (Debian 10.2.1-6) 10.2.1 20210110)

### Plain log:

### HTML log:

### Rootfs:
- [http://storage.kernelci.org/images/rootfs/buildroot/kci-2020.05-6-g8983f3b738df/armel/baseline/rootfs.cpio.gz](http://storage.kernelci.org/images/rootfs/buildroot/kci-2020.05-6-g8983f3b738df/armel/baseline/rootfs.cpio.gz)

### Relevant error message:
- * baseline.dmesg.emerg: [https://kernelci.org/test/case/id/61a587a0ab3b0079bd18f6dd](https://kernelci.org/test/case/id/61a587a0ab3b0079bd18f6dd)
  - new failure (last pass: v4.19.216-cip61)
  - 2 lines

```plaintext
2021-11-30T02:08:20.120003 kern :emerg : BUG: spinlock bad magic on CPU#0, udevd/110
2021-11-30T02:08:20.129545 kern :emerg : lock: emif_lock+0x0/0xffffecfc [emif] .magic: dead4ead, owner: <none>/-1, owner_cpu: -1
2021-11-30T02:08:20.143995 <8>[ 21.198120] <LAVA_SIGNAL_TESTCASE TEST_CASE_ID=emerg RESULT=fail UNITS=lines MEASUREMENT=2>
```
Add CIP rootfs

- Rootfs are created with isar (Integration System for Automated Root filesystem generation) on Gitlab pipeline
- Created RootFS storage are pushed with KernelCI upload API (old API) to the KernelCI storage server
  - https://storage.kernelci.org/images/rootfs/cip/
- Storage RootFS image are used by KernelCI

Index of /images/rootfs/cip/

<table>
<thead>
<tr>
<th>File Name</th>
<th>File Size</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent directory/</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>20211105/</td>
<td>-</td>
<td>2021-Nov-05 17:41</td>
</tr>
<tr>
<td>20211115/</td>
<td>-</td>
<td>2021-Nov-15 14:06</td>
</tr>
<tr>
<td>latest/</td>
<td>-</td>
<td>2021-Nov-05 17:41</td>
</tr>
</tbody>
</table>
kernelci-core/config/core/test-configs.yaml

17   cip:
18     url: 'https://storage.kernelci.org/images/rootfs/cip/20211105'
19     arch_map:
20       amd64: [{arch: x86_64}]
Adding test SMC (spectre meltdown)

kernelci-core/config/lava/smc/smc.jinja2

```yaml
- test:
  name: {{ plan }}
  description: "Spectre meltdown test plan"
  timeout:
    minutes: 15
  definitions:
    - repository: https://github.com/kernelci/test-definitions
      from: git
      revision: kernelci.org
      path: automated/linux/spectre-meltdown-checker-test/spectre-meltdown-checker-test.yaml
      name: {{ plan }}
      lava-signal: kmsg
```
Lava test job

metadata:
  name: spectre-meltdown-checker
  format: "Lava-Test Test Definition 1.0"
  description: "Run spectre meltdown checker"
  maintainer:
    - naresh.kamboju@linaro.org

environment:
  - lava-test-shell

run:
  steps:
    - cd ./automated/linux/spectre-meltdown-checker-test
    - ./spectre-meltdown-checker-test.sh -s "${SKIP_INSTALL}" -v "${SMC_VERSION}" -w "${WGET_UPSTREAM}" 
      - ../../../utils/send-to-lava.sh ./output/result.txt
## Available Kernels

<table>
<thead>
<tr>
<th>Branch</th>
<th>Kernel</th>
<th>Commit</th>
<th>Build Status</th>
<th>Test Results</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>linux-5.10.y-cip</td>
<td>v5.10.83-cip1</td>
<td>2332f07a324fd7...</td>
<td>174</td>
<td>2479</td>
<td>2021-12-05</td>
</tr>
<tr>
<td>linux-4.19.y-cip-rt</td>
<td>v4.19.217-cip62...</td>
<td>59a33e49411615...</td>
<td>177</td>
<td>1032</td>
<td>2021-12-01</td>
</tr>
<tr>
<td>linux-4.19.y-cip</td>
<td>v4.19.217-cip62</td>
<td>dc62e26e3be87...</td>
<td>112</td>
<td>1939</td>
<td>2021-11-30</td>
</tr>
<tr>
<td>linux-4.19.y-cip</td>
<td>v4.19.216-cip61</td>
<td>6ecdd66903013...</td>
<td>113</td>
<td>3458</td>
<td>2021-11-13</td>
</tr>
<tr>
<td>linux-4.4.y-cip</td>
<td>v4.4.291-cip65</td>
<td>65ed894ba1119b...</td>
<td>103</td>
<td>1395</td>
<td>2021-11-13</td>
</tr>
<tr>
<td>linux-5.10.y-cip</td>
<td>v5.10.8-8301-g3...</td>
<td>3d6168cb89f653...</td>
<td>118</td>
<td>5482</td>
<td>2021-10-29</td>
</tr>
<tr>
<td>linux-4.19.y-cip</td>
<td>v4.19.213-cip60</td>
<td>7f69205acfe12...</td>
<td>113</td>
<td>3733</td>
<td>2021-10-24</td>
</tr>
<tr>
<td>linux-4.4.y-cip</td>
<td>v4.4.287-cip64</td>
<td>18599fbc737113...</td>
<td>114</td>
<td>1053</td>
<td>2021-10-13</td>
</tr>
</tbody>
</table>
Still left work

From the CIP - KernelCI organization board

- Cleaning the results (as KernelCI large number of boards and KernelCI problems sometime produce false results) work on this is here
- Implementing tests for IEC-62443-4-2 standard (there is already some work on this [1][2])
What is KCIDB

KernelCI Database service and tools

That tools is a package for submitting and querying Linux Kernel CI reports, coming from independent CI systems, and for maintaining the service behind that

https://kcidb.kernelci.org
Pros

- Can be easily implemented in your current workflow
- Useful if you already have a kernel testing framework that differs from KernelCI scope
- You can tailorize your system for your own personalized tests or environment
- Tool for unifying results from different CI systems in KernelCI
- Standardize CI reports and kernel upstream notifications
Who is using KCIDB

- Gentoo Linux
- Red Hat CKI
- Arm
- Google syzbot
- KernelCI
- Linaro Tuxsuite
Implementation

- Just adding a step to push results to your Linux kernel testing framework is enough to enable KCIDB.
- Any change is local to the testing framework and doesn’t have to be part of KernelCI native implementation.
Sending results with KCIDB

kcidb-submit -p kernelci-production -t kernelci_new < data_file.json
Welcome to the KernelCI web dashboard for the upstream Linux kernel.

You'll find here all the results for kernel builds and tests run natively by KernelCI on mainline, linux-next, stable and a variety of maintainer branches.

To find out more about the project, see the main kernelci.org website.

- View kernel branches and their latest build and test results
- View latest kernel build results
- View latest test results
- View results per SoC or hardware family
- View statistics about all the data
Welcome to Linux Kernel CI report database (KCIDB) dashboard. Here you can find testing reports aggregated from multiple CI systems ("origins"). This is a work in progress, and we welcome contributions and participation. See the source code for the collection system, and for this dashboard. Write to kernelci@groups.io if you want to start submitting results from your CI system, or if you want to receive feedback.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Repository</th>
<th>Branch Name</th>
<th>Hash</th>
<th>Discovery time</th>
<th>Status</th>
<th>Builds</th>
<th>Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>gkernelci</td>
<td>git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git</td>
<td>5.4</td>
<td>219d54332a09</td>
<td>2021-01-21 19:34:58.569610+00</td>
<td>✔</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>gkernelci</td>
<td>git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git</td>
<td>4.4</td>
<td>af42f9b7e1b</td>
<td>2021-01-21 18:40:37.942345+00</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>gkernelci</td>
<td>git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git</td>
<td>4.9</td>
<td>69973b830859</td>
<td>2021-01-21 18:40:37.916154+00</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>gkernelci</td>
<td>git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git</td>
<td>5.10</td>
<td>2c95ebc5753b3e</td>
<td>2021-01-21 17:37:10.902543+00</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>gkernelci</td>
<td>git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git</td>
<td>4.19</td>
<td>84df9525b0c2</td>
<td>2021-01-21 16:03:36.629884+00</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>gkernelci</td>
<td>git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git</td>
<td>4.14</td>
<td>bebc6082da0a</td>
<td>2021-01-21 16:00:49.546258+00</td>
<td>✔</td>
<td>✗</td>
<td>✔</td>
</tr>
</tbody>
</table>
Future

- KernelCI core -> new KernelCI API (still in early phase)
- Jenkins -> Buildbot (still in decision phase)
  - Complete API
  - Both command line and UI interface
    - Command line can be used for sending patch test diff to be tested without commit
    - Can give developer authorization token for each developer
  - Maximum flexibility with “multi master” - “multi worker” modality
- Wrote in Python
- Reporters for mostly everything
- Modularity and Plugins
- Using twisted for asynchronous methods
- 19 years of development currently at version 3.5.0

- Buildbot KCIDB plugin (still in development phase)
Conclusion

KernelCI is a great tool that can be part of your daily Kernel development workflow.

Having a way of getting multiple tests results for each kernel change without going around with boards and data centers, can easily improve your life.

Having a way of automatically testing kernel sources and patches was also the reasoning behind the creation of GKernelCI.
More about

Civil Infrastructure Platform project
More about KernelCI

KernelCI documentation
More about Gentoo Kernel CI
GKernelCI
Civil Infrastructure Project Technical Channels

- IRC
  - #CIP on libera.chat

- Groups.io
  - https://lists.cip-project.org/g/cip-dev/
KernelCI Maintainers Channels

- IRC
  - #KernelCI on libera.chat

- Slack
  - https://kernelci.slack.com/

- Twitter
  - https://twitter.com/kernelci

- Groups.io
  - https://groups.io/g/kernelci
GkernelCI Maintainers Channels

- IRC
  - #gentoo-kernelci on libera.chat