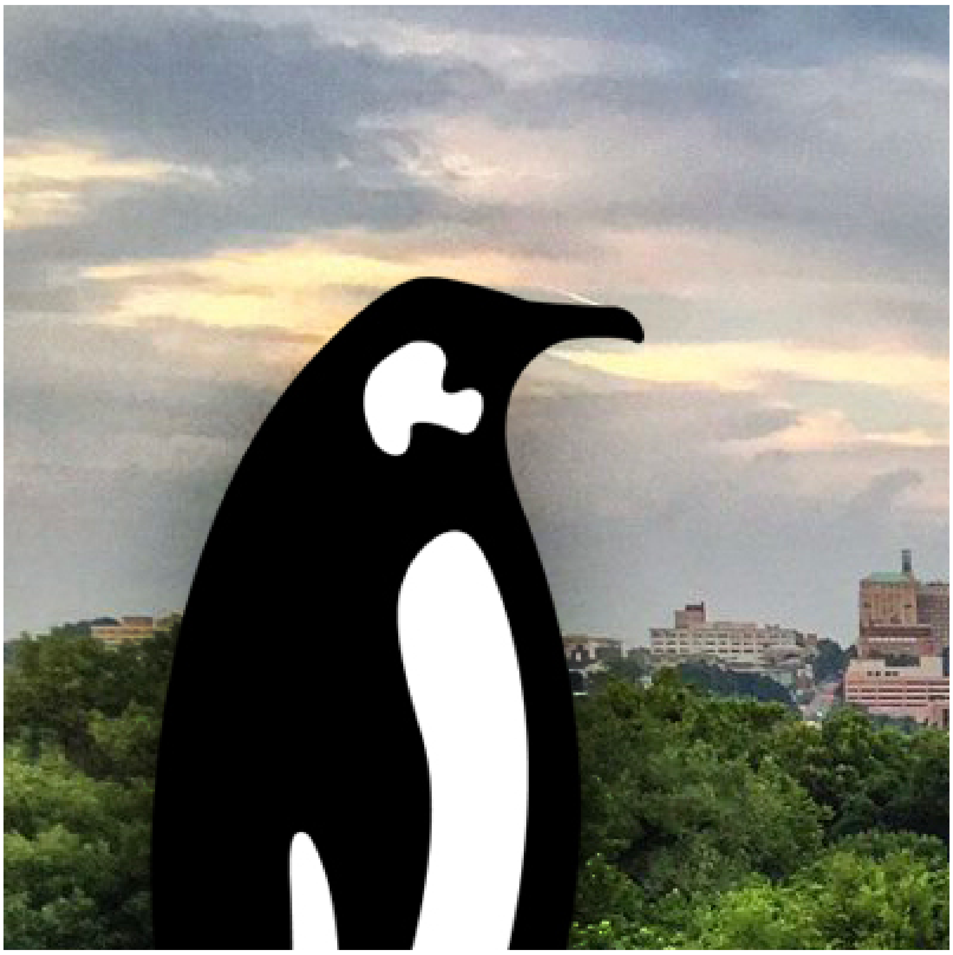
Linux Plumbers Conference 2023



Report of Contributions

https://lpc.events/e/17

Contribution ID: **26** Type: **not specified**

Android Microconference

The Android Micro Conference brings the upstream community and Android systems developers together to discuss issues and changes to the Android platform and their dependencies and interactions with the Linux kernel, allowing for collaboration on solutions for upstream.

Since last year's conference, there has been quite a bit of progress, specifically around:
- MGLRU enabled on Android14 kernels
- Upstream submissions of Fuse-bpf work, with a number of iterations of review feedback.
- Lots of patches resolving fw_devlink issues

Currently planned discussion topics for this year include:
- 16k Pages
- RISC-V
- android-mainline on Pixel6
- Updates on Binder
- BPF usage w/ Android
- Kernel and platform integration testing
- Vendor Hook Usage
- Building Modules for Android GKI Kernels
- Resolving Priority Inversion w/ Proxy Execution
- AOSP Devboards
- And likely more…

Key Attendees:
- Alistair Delva
- Betty Zhou
- Carlos Llamas
- John Moon
- Kalesh Singh
- Leo Yan
- Lukasz Luba
- Matthias Männich
- Neill Kapron
- Qais Yousef
- Sumit Semwal
- Suren Baghdasaryan
- William McVicker

MC leads:
John Stultz jstultz@google.com
Karim Yaghmour karim.yaghmour@opersys.com
Amit Pundir amit.pundir@linaro.org
Sumit Semwal sumit.semwal@linaro.org

Primary authors: PUNDIR, Amit; STULTZ, John (Google); YAGHMOUR, Karim (Opersys inc.); SEMWAL, Sumit (Linaro)

Track Classification: LPC Microconference Proposals

Contribution ID: **25** Type: **not specified**

Build Systems Microconference

In the Linux ecosystems there are many ways to build all the software used to put together a running system. Whether it's building all the binary packages for a binary Linux distribution, using a source-based distribution, or building an embedded system from scratch, there are a lot of shared challenges which each system solves in their own way.

This microconference is a way to get people who work on disparate build systems to discuss common problems and possible shared solutions across the entire problem space.

Suggested topics:

- Bootstrapping the build system
- Cross building software
- Make, autoconf, and other similar software build tools
- Package build systems, bitbake, emerge/portage, pacman, etc
- Packaging formats
- Managing software with language specific package managers
- Patch sharing
- License gathering and verification
- Security updates
- SBOMS
- Software chain-of-trust
- Repeatable builds
- Documentation and education
- Finding the next generation of maintainers
- Build-system visibility within the wider Plumbers attendees

Audience

Developers and maintainers in projects such as (though not a definitive list):

- Arch Linux
- Buildroot
- ChromeOS build
- Gentoo
- OpenEmbedded
- OpenWRT/LEDE
- Yocto Project
- Other traditional Binary Packaged distributions

Primary author: Mr WEBSTER, Behan (The Linux Foundation)

Co-author: BALISTER, Philip (OpenEmbedded)

Track Classification: LPC Microconference Proposals

Contribution ID: **31** Type: **not specified**

Compute eXpress Link (CXL)

Compute Express Link is a cache coherent fabric that in recent years has been gaining momentum in the industry. CXL 3.0 launched just before Plumbers 2022 (where very early discussions were had), bringing new challenges such as dynamic capacity devices and large scale fabrics, two features that bring significant challenges to Linux. There has also been controversy and confusion in the Linux kernel community about the state and future of CXL, regarding its usage and integration into, for example, the core memory management subsystem. Many concerns have been put to rest through proper clarification and setting of expectations.

The Compute Express Link microconference focuses on how to evolve the Linux CXL kernel driver and userspace components for support of the CXL 2.0 spec (and beyond). The microconference provides a space to open the discussion, incorporate more perspectives, and grow the CXL community with a goal that the CXL Linux plumbing serves the needs of the CXL ecosystem while balancing the needs of the Linux project. Specifically, this microconference welcomes submissions detailing industry and academia use cases in order to develop usage model scenarios. Finally, it will be a good opportunity to have existing upstream CXL developers available in a forum to discuss current CXL support and to communicate areas that need additional involvement.

Suggested topics:
- Ecosystem & Architectural review
- Dynamic Capacity Devices
- Fabric Management
- QEMU support
- Security (ie: IDE/SPDM)
- Managing vendor specificity
- Type 2 accelerator support (bias flip management)
- Coherence management of type2/3 memory (back-invalidation)
- P2P (UIO)
- RAS (GPF, AER)
- Hotplug (qos policies, daxctl)
- Hot remove
- Documentation
- Memory tiering topics that can relate to cxl (out of scope of MM/performance MCs)
- Industry and academia use cases

MC Leaders:
Dan Williams
Adam Manzanares
Jonathan Cameron
Davidlohr Bueso

Primary authors: MANZANARES, Adam (Samsung Electronics); WILLIAMS, Dan (Intel Open Source Technology Center); Mr BUESO, Davidlohr (Samsung Semiconductor); CAMERON, Jonathan (Huawei Technologies R&D (UK))

Presenter: Mr PRICE, Gregory (MemVerge Inc)

Track Classification: LPC Microconference Proposals

Contribution ID: **22** Type: **not specified**

Confidential Computing Microconference

The Confidential Computing microconferences in the past years brought together developers working secure execution features in hypervisors, firmware, Linux Kernel, over low-level user space up to container runtimes. A broad range of topics was discussed ranging from enablement for hardware features up to generic attestation workflows.

Over the last year there was progress on the development of Confidential Computing in the Linux kernel and user-space. The patch-sets for Intel TDX guest support and AMD SEV-SNP guest support were merged into the Linux kernel. Support for running as a CVM under Hyper-V has also been partially merged.

But there is still some way to go and problems to solve before a secure Confidential Computing stack with open source software and Linux as the hypervisor becomes a reality. The most pressing problems right now are:

- Support for restricted memory (Unmapped Private Memory patch-set, UPM) is still under development and discussion
- AMD SEV-SNP and Intel TDX host support

Other potential problems to discuss are:

- Support for un-accepted memory
- Confidential Computing support for ARM64
- Attestation workflows
- Secure VM service module (SVSM) and paravisor architecture and implementation (LINUX-SVSM vs. COCONUT-SVSM)
- Confidential Computing threat model
- Secure IO and device attestation
- Intel TDX Connect
- AMD SEV-TIO
- RISC-V CoVE (Task group and patches)
- Debuggability and live migration of confidential virtual machines

The Confidential Computing Microconference wants to bring developers
working on confidential computing together again to discuss these and other open problems.

Primary authors: GIANI, Dhaval (AMD); ROEDEL, Joerg (SUSE)

Track Classification: LPC Microconference Proposals

Contribution ID: **12** Type: **not specified**

Containers and checkpoint/restore

The usual containers and checkpoint/restore micro-conference.

We will be discussing recent advancements in container technologies with some of the usual candidates being:

- CGroupV2 feature parity with CGroupV1
- Emulation of various files and system calls through FUSE and/or Seccomp
- Dealing with the eBPF-ification of the world
- Making user namespaces more accessible
- VFS idmap improvements

On the checkpoint/restore front, some of the potential topics include:

- Restoring FUSE services
- Handling GPUs
- Dealing with restartable sequences

And quite likely a variety of other container and checkpoint/restore topics as things evolve between now and the event.

Past editions of this micro-conference have been the source of many developments in the Linux kernel, including:

- PIDfds
- VFS idmap
- FUSE in user namespace
- Unprivileged overlayfs
- Time namespace
- A variety of CRIU features and checkpoint/restore kernel interfaces

Primary authors: REBER, Adrian (Red Hat); Mr BRAUNER, Christian; RAPOPORT, Mike (IBM); GRABER, Stéphane (Canonical Ltd.)

Track Classification: LPC Microconference Proposals

Contribution ID: **33** Type: **not specified**

Internet of Things MC

The IoT Microconference is a forum for developers to discuss all things IoT. Topics include tools, telemetry, device drivers, and protocols in not only the Linux kernel but also Real-Time Operating Systems such as Zephyr.

Since last year, there have been a number of new technical topics with significant updates.

- Opportunities in IoT and Edge computing with the Linux /dev/accel API
- Using the Thrift RPC framework between Linux and Zephyr
- Zephyr's new HTTP Server (a GSoC project)
- Rust in the Zephyr RTOS: Benefits, Challenges and Missing Pieces
- BeagleConnect Freedom Updates, Greybus, and the Linux Interface
- Linux-wpan updates on 6lowpan, 802.15.4 PAN coordinators and UWB

Current Problems that require attention (stakeholders):

- IEEE 802.15.4 SubGHz improvement areas in Zephyr, Linux (Florian Grandel, Stefan Schmidt, BeagleBoard.org)
- WpanUSB upstreaming in the Linux kernel, potentially dropping Zephyr support (Andrei Emeltchenko, BeagleBoard.org)
- IEEE 802.15.4 Linux subsystem device association handling (Miquel Raynal, Alexander Aring, Stefan Schmidt)
- Zephyr potentially dropping Bluetooth IPSP?

On a slightly less technical topic.

- Reflections after Two Years of Zephyr LTSv2

We hope you will join us either in-person or remote for what is shaping up to be another great event full of collaboration, discussion, and interesting perspectives.

Primary authors: FRIEDT, Christopher (Friedt Professional Engineering Services); Mr SCHMIDT, Stefan

Track Classification: LPC Microconference Proposals

Contribution ID: **8** Type: **not specified**

Kernel Testing & Dependability MC

The Linux Plumbers 2023 Kernel Testing & Dependability track focuses on advancing the current state of testing of the Linux Kernel and its related infrastructure. The main purpose is to improve software quality and dependability for applications that require predictability and trust. We aim to create connections between folks working on similar projects, and help individual projects make progress.

This track is intended to promote collaboration between all the communities and people interested in the Kernel testing & dependability. This will help move the conversation forward from where we left off at the LPC 2022 Kernel Testing & Dependability MC.

We ask that any topic discussions focus on issues/problems they are facing and possible alternatives to resolving them. The Microconference is open to all topics related to testing on Linux, not necessarily in the kernel space.

Potential testing and dependability topics:

- KernelCI: Topics on improvements and enhancements for test coverage
- Growing KCIDB, integrating more sources (https://kernelci.org/docs/kcidb/)
- Better sanitizers: KFENCE, improving KCSAN. (https://lwn.net/Articles/835367/)
- Using Clang for better testing coverage: Now that the kernel fully supports building with clang, how can all that work be leveraged into using clang's features?
- How to spread KUnit throughout the kernel?
- Building and testing in-kernel Rust code.
- Identify missing features that will provide assurance in safety critical systems.
- Which test coverage infrastructures are most effective to provide evidence for kernel quality assurance? How should it be measured?
- Explore ways to improve testing framework and tests in the kernel with a specific goal to increase traceability and code coverage.
- Regression Testing for safety: Prioritize configurations and tests critical and important for quality and dependability
- Transitioning to test-driven kernel release cycles for mainline and stable: How to start relying on passing tests before releasing a new version?
- Explore how do SBOMs figure into dependability?

Things accomplished from last year:

- Developing a new, modern API for KernelCI with Pub/Sub interface
- Adding Rust coverage in KernelCI https://linux.kernelci.org/job/rust-for-linux/branch/rust/
- KCIDB is continuing to gather results from many test systems: KernelCI, Red Hat's CKI, syzbot, ARM, Gentoo, Linaro's TuxSuite etc. The current focus is on generating common email reports based on this data and dealing with known issues.
- KFENCE is continuing to aid in detecting Out-of-bound OOB accesses, use-after-free errors (UAF), Double free and Invalid free and so on.
- Clang: CFI, weeding out issues upstream, etc.

- Kselftest continues to add coverage for new and existing features and subsystems.
- KUnit is continuing to act as the standard for some drivers and a de facto unit testing framework in the kernel . (https://www.youtube.com/watch?v=78gioY7VYxc)
- The Runtime Verification (RV) interface from Daniel Bristot de Oliveira was successfully merged.

MC Leads:

- Sasha Levin
- Guillaume Tucker
- Shuah Khan

Unconfirmed to-be attendees:

- Sasha Levin
- Kevin Hilman
- Guillaume Tucker
- Alice Ferrazzi
- Veronika Kabatova
- Nikolai Kondrashov
- Antonio Terceiro
- Mark Brown
- Don Zickus
- Enric Balletbo
- Tim Orling
- Gustavo Padovan
- Bjorn Andersson
- Milosz Wasilewski
- Shuah Khan
- Martin Peres
- Arnd Bergmann
- Remi Duraffort
- Peter Zijlstra
- Daniel Stone
- Jan Lübbe
- Dmitry Vyukov
- Brendan Higgins
- Greg KH
- Anders Roxell
- Guenter Roeck
- Jesse Barnes
- Kees Cook

- Kate Stewart
- Sudip Mukherjee
- Daniel Bristot de Oliveira
- Gabriele Paoloni
- Paul Albertella
- Elana Copperman
- Lukas Bulwahn

Primary authors: TOCKER, Guillaume; LEVIN, Sasha; KHAN, Shuah; KHAN, Shuah (The Linux Foundation)

Track Classification: LPC Microconference Proposals

Contribution ID: **194** Type: **not specified**

Detecting failed device probes

Monday 13 November 2023 12:00 (25 minutes)

Regressions that cause a device to no longer be probed by a driver can have a big impact on the platform's functionality, and despite being relatively common there isn't currently any generic way to detect them.

By enabling the community to catch device probe regressions in a way that doesn't require additional work for every new platform, and that can catch issues from config changes to driver probe errors, such regressions can be addressed much quicker and improve the kernel's stability.

This session will present the approach that is currently being proposed to detect device probe regressions and open the floor for feedback and discussion.

Primary authors: NAO, Laura; PRADO, Nicolas (Collabora)

Presenters: NAO, Laura; PRADO, Nicolas (Collabora)

Session Classification: Kernel Testing & Dependability MC

Track Classification: LPC Microconference: Kernel Testing & Dependability MC

Contribution ID: **225** Type: **not specified**

Quality in embargoed patches

Monday 13 November 2023 11:30 (25 minutes)

The bar on the quality of code that fixes embargoed issues is pretty low: usually the case is that the code is only tested by the author, and possibly a handful of other folks who are part of working on the fix.

This session is a discussion to help draft a proposal for a testing story that could be presented to HW vendors that with to publish embargoed code without going through the kernel's usual review process.

For more context: https://lore.kernel.org/ksummit/ZNuuvS5BtmjcazIv@sashalap/

Primary author: LEVIN, Sasha

Presenter: LEVIN, Sasha

Session Classification: Kernel Testing & Dependability MC

Track Classification: LPC Microconference: Kernel Testing & Dependability MC

Contribution ID: **114** Type: **not specified**

Storing and Outputting Test Information: KUnit Attributes and KTAPv2

Monday 13 November 2023 10:05 (25 minutes)

Current kernel testing frameworks save basic test information including test names, results, and even some diagnostic data. But to what extent should frameworks store supplemental test information? This could include test speed, module name, file path, and even parameters for parameterized tests.

Storing this information could greatly improve the kernel developer experience by allowing test frameworks to filter out unwanted tests, include helpful information in KTAP results, and possibly populate auto-generated documentation. I have been working on the new KUnit Test Attributes feature that could be part of this solution to store and access test-associated data.

But what test attributes should we be saving? How should test-associated data be formatted in KTAP? And what possibilities does this open with parameterized tests (filtering based on parameters or even parameter injection)?

Primary author: MOAR, Rae

Presenter: MOAR, Rae

Session Classification: Kernel Testing & Dependability MC

Track Classification: LPC Microconference: Kernel Testing & Dependability MC

Contribution ID: **115** Type: **not specified**

Testing Drivers with KUnit (Does hardware have to be hard?)

Monday 13 November 2023 10:30 (25 minutes)

Unit testing common library code is (relatively) easy, but drivers often deal with a lot of global state, both in code and in hardware. New features like static stubbing go some way towards making this easier, but a lot of work still goes into making "fake devices".

There are still many open questions, however:
- Are the existing tools helping? Is there something obviously missing?
- Are UML features like LOGIC_IOMEM a good path forward?
- How should drivers make a fake 'struct device'? Via a platform_device (possibly with devicetree support), root_device, or a new kunit_device?
- There are lots of ways of managing resources for tests (kunit_resource, KUnit actions, devres/devm_ APIs). What should we use, when?
- How do we deal with callbacks, threads, etc, with KUnit contexts?
- How to support other safety/reliability/testing opportunities like hardware fuzzing and Rust?

Primary author: GOW, David (Google)

Presenter: GOW, David (Google)

Session Classification: Kernel Testing & Dependability MC

Track Classification: LPC Microconference: Kernel Testing & Dependability MC

Contribution ID: **215** Type: **not specified**

the path to achieve a bug-free build on the mainline

Monday 13 November 2023 09:40 (25 minutes)

There're a lot of focused testing effort across Linux kernel community to guarantee the quality of kernel from build to runtime. Nowadays, not only the test process has moved towards formalization but also the test coverage has been increased to discover more issues in an earlier time. On the other side, some issues are still escaped to mainline.

In this talk, we will dive into the build issues and look for a possible path to achieve build issue free on mainline. We will share
* the status quo of mainline including the trend of the issues found by different tools
* the profile of known community testing effort regarding the coverage and focus
* the newly adopted practices in 0-Day CI in past year

Then we want to exchange the ideas and have the discussion around
* what are the challenges that community currently faces, like identifying the build coverage of a new patch
* what kind of practices can be propagated to various testing sides
* the possible timeline (stages) for such achievement

We look forward to having more collaboration with other players in the community to jointly achieve this goal.

Primary author: LI, philip

Presenter: LI, philip

Session Classification: Kernel Testing & Dependability MC

Track Classification: LPC Microconference: Kernel Testing & Dependability MC

Contribution ID: **197** Type: **not specified**

Unifying and improving test regression reporting and tracking

Monday 13 November 2023 12:25 (35 minutes)

The current CI systems for the kernel offer basic and low-level
regression detection and handling capabilities based on test results, but they do that in their own specific way. We wonder if we can find more common ways of tackling the problem through post-processing the data provided by the different CI systems. We could then extract additional "hidden" information, look at failure trends, analyze and compare logs, as well as understand the impact of the test infra setup in a failure. This would allow us to address the need to track the reported regressions status, creating a standard workflow across the different tools.

Regzbot is proving itself as part of this standard workflow, but reporting the regressions from CI systems there is hard as the volume could be high, the relevance of some regressions could be low, and false positives do exist (a lot - specially for hardware testing). Better post-processing of the data and more standard workflows can help more reports to get to the community without destroying the quality of the data provided by regzbot.

We'd like to briefly talk about the current status and limitations of the testing ecosystem regarding regressions, exhibit the current efforts and strategies being developed, and start a discussion about contributions and plans for the future.

Primary authors: PADOVAN, Gustavo (Collabora); CAÑUELO, Ricardo

Presenters: PADOVAN, Gustavo (Collabora); CAÑUELO, Ricardo

Session Classification: Kernel Testing & Dependability MC

Track Classification: LPC Microconference: Linux Kernel Debugging MC

Contribution ID: **272** Type: **not specified**

Welcome!

Monday 13 November 2023 09:30 (10 minutes)

Presenters: LEVIN, Sasha; KHAN, Shuah

Session Classification: Kernel Testing & Dependability MC

Contribution ID: **37** Type: **not specified**

KVM

KVM (Kernel-based Virtual Machine) enables the use of hardware features to improve the efficiency, performance, and security of virtual machines created and managed by userspace. KVM was originally developed to host and accelerate "full" virtual machines running a traditional kernel and operating system, but has long since expanded to cover a wide array of use cases, e.g. hosting real time workloads, sandboxing untrusted workloads, deprivileging third party code, reducing the trusted computed base of security sensitive workloads, etc. As KVM's use cases have grown, so too have the requirements placed on KVM and the interactions between it and other kernel subsystems.

<p>The KVM Microconference will focus on how to evolve KVM and adjacent subsystems in order to satisfy new and upcoming requirements. Potential topics include:

<ul> <li> Serving inaccessible/unmappable memory for KVM guests (protected VMs); fine-grain permission updates of IOMMU and MMU page tables <li> Optimizing mmu$_notifiers, e.g. reducingTLBflushesandspuri$
$li > ImprovingandhardeningKVM + perfinteractions < li > Implementingarch -$
$agnosticabstractionsinKVM(e.g. MMU) < li > Utilizing"fault"injectiontoincreasetestcoverageofedgecases <$
$li > KVMvsVFIO(e.g. memorytypes, aratherhottopicontheARMside) < li > Persistenceofguestmemoryandke$
$/ul >$

<p>Key Attendees: <ul> <li> Paolo Bonzini (pbonzini@redhat.com), KVM Maintainer <li> Sean Christopherson (seanjc@google.com), KVM x86 Co-Maintainer <li> Alexander Graf (graf@amazon.de) <li> James Gowans (jgowans@amazon.com) <li> Mickaël Salaün (mic@digikod.net) </ul>

Primary authors: BONZINI, Paolo (Red Hat, Inc.); Mr CHRISTOPHERSON, Sean (Google)

Presenter: GOWANS, James (Amazon EC2)

Track Classification: LPC Microconference Proposals

Contribution ID: **14** Type: **not specified**

Linux Kernel Debugging Microconference

When things go wrong, we need to debug the kernel. There are about as many ways to do that as you can imagine: printk, kdb/kgdb over serial, tracing, attaching debuggers to /proc/kcore, and post-mortem debugging using core dumps, just to name a few. Frequently, tools and approaches used by userspace debuggers aren't enough for the requirements of the kernel, so special tools are created to handle them: crash, drgn, makedumpfile, libkdumpfile, and many, many others.

With the variety of tools and approaches available, it's important to collaborate on whatever shared problems we may have. This microconference is an opportunity to discuss these problems and come up with shared approaches to resolve them. Some examples of potential topic areas:

- Many debuggers understand core subsystems such as tasks, slab caches, mm_structs, etc, and provide information about them. But as the kernel evolves, code changes can break these tools, which need to contain decades of cruft to handle a variety of versions. How can we improve processes and tools so that the future decades of evolution can be handled without crippling our debuggers with more technical debt?

How can we share logic between debuggers to reduce duplicate effort in interpreting core kernel data structures?

Please see Philipp Rudo's excellent talk from LPC 2022 regarding this very topic.

- Kernel core dumps can come from a variety of sources: some are generated via kexec and /proc/vmcore, then makedumpfile. Others may be created by a variety of hypervisors including Qemu, Xen, and Hyper-V. The core dumps can use ELF, or more commonly, the compressed diskdump family of formats. With the variety of core dump producers and consumers, along with the variation in formats, it's not uncommon to encounter "broken" core dumps which need tweaks or additional tools to be read. How can we build tools to handle the diversity of core dumps, more easily fix broken ones, and guide the community to a better documented standard?
- Kernel debuggers rely on debuginfo such as DWARF, which can be bulky and is not commonly distributed alongside the kernel. How can we enable lightweight debugging options that run everywhere?
- When debugging kernel-related issues on live systems, stack unwinding of both kernel and userspace tasks is important. As it is, stack unwinding in the kernel can be done via frame pointers and ORC on x86_64, but userspace stack unwinding is more difficult, since many applications and libraries are compiled without frame pointers, and the kernel lacks a DWARF-based unwinder. What can the kernel debugging and tracing community do to improve this situation?

Below are key people who may be interested in joining the discussions and/or presenting on relevant debugging topics:

- Omar Sandoval
- Petr Tesarik
- Philipp Rudo

Primary author: BRENNAN, Stephen (Oracle)

Track Classification: LPC Microconference Proposals

Contribution ID: **15** Type: **not specified**

Live Patching Microconference

The Live Patching microconference at Linux Plumbers 2023 aims to gather stakeholders and interested parties to discuss proposed features and outstanding issues in live patching.

Live patching is a critical tool for maintaining system uptime and security by enabling fixes to be applied to running systems without the need for a reboot. The development of the infrastructure is an ongoing effort and while many problems have been resolved and features implemented, there are still open questions, some with already submitted patch sets, which need to be discussed.

Live Patching microconferences at the previous Linux Plumbers conferences proved to be useful in this regard and helped us to find final solutions or at least promising directions to push the development forward. It includes for example a support for several architectures (ppc64le and s390x were added after x86_64), a late module patching and module dependencies and user space live patching.

Currently proposed topics follow. The list is open though and more will be added during the regular Call for Topics.

- klp-convert (as means to fix CET IBT limitations) and its upstreamability
- shadow variables, global state transition
- kselftests and the future direction of development
- arm64 live patching
- livepatch author guide (a documentation of a live patch creation from source code)
- live patching and Rust

Key people

- Josh Poimboeuf jpoimboe@kernel.org
- Jiri Kosina jikos@kernel.org
- Miroslav Benes mbenes@suse.cz
- Petr Mladek pmladek@suse.com
- Joe Lawrence joe.lawrence@redhat.com
- Nicolai Stange nstange@suse.de
- Marcos Paulo de Souza mpdesouza@suse.de
- Mark Rutland mark.rutland@arm.com
- Mark Brown broonie@kernel.org

We encourage all attendees to actively participate in the microconference by sharing their ideas, experiences, and insights.

Primary authors: LAWRENCE, Joe (Red Hat); BENEŠ, Miroslav

Track Classification: LPC Microconference Proposals

Contribution ID: **36** Type: **not specified**

Power Management and Thermal Control Micro-conference

The Power Management and Thermal Control Microconference focuses on power management and thermal control infrastructure, CPU and device power-management mechanisms, and thermal control methods. In particular, we are interested in improving the thermal control infrastructure in the kernel to cover more use cases and utilizing energy-saving opportunities offered by modern hardware in new ways.

The goal is to facilitate cross-framework and cross-platform discussions that can help improve energy-awareness and thermal control in Linux.

Prospective topics:

- Idle injection and soft IRQs. (Srinivas Pandruvada)
  https://lore.kernel.org/linux-pm/20221215184300.1592872-1-srinivas.pandruvada@linux.intel.com
  https://lore.kernel.org/linux-pm/abcc6689c283bbe91b6fd16572cfd4a8d5e78cc6.camel@linux.intel.com
- Thermal sysfs/API update: are we happy with the current framework? (Srinivas Pandruvada)
  https://lore.kernel.org/linux-pm/abcc6689c283bbe91b6fd16572cfd4a8d5e78cc6.camel@linux.intel.com
- A way to define additional private attributes for a thermal zone. (Srinivas Pandruvada)
  https://lore.kernel.org/linux-pm/abcc6689c283bbe91b6fd16572cfd4a8d5e78cc6.camel@linux.intel.com
- intel_lpmd - Intel Low Power Mode Daemon. (Zhang Rui)
  https://lore.kernel.org/linux-pm/717fd5f97da6fd3ac6fa323220ab4a948db1a174.camel@intel.com
- Energy-efficiency API for scheduling tasks. (Len Brown)
  https://lpc.events/event/16/contributions/1276/
- Thermal infrastructure for debugfs + clean up the sysfs debug-related information. (Daniel Lezcano)
  https://lore.kernel.org/linux-pm/745b8b17-af4d-e8e1-83c1-89d600e7cd19@linaro.org
- New thermal trip types. (Daniel Lezcano)
  https://lore.kernel.org/linux-pm/745b8b17-af4d-e8e1-83c1-89d600e7cd19@linaro.org
- Thermal management with the time dimension taken into account. (Daniel Lezcano)
  https://lore.kernel.org/linux-pm/745b8b17-af4d-e8e1-83c1-89d600e7cd19@linaro.org
- Step-wise thermal governor improvements. (Daniel Lezcano)
  https://lore.kernel.org/linux-pm/745b8b17-af4d-e8e1-83c1-89d600e7cd19@linaro.org
- ACPI extensions for device DVFS. (Sudeep Holla)
  https://lore.kernel.org/linux-pm/20230525091844.tbxrk5gcwr2lppfo@bogus
- uclamp in CFS: Fairness, latency, and energy efficiency. (Morten Rasmussen)
  https://lore.kernel.org/linux-pm/ZHcebApf6WCPMxPa@R5WKVNH4JW
- Support multiple system wide low power states. (Ulf Hansson)
  https://lore.kernel.org/linux-pm/CAPDyKFpeLcH53A+jRwU41aHs48zhSA54aSrTmHASWxPofaU-tg@mail.gmail.com

All of the topics listed above can benefit from face-to-face discussions at the LPC, especially as far as the time needed to reach the consensus is concerned for at least some of them.

The specific list of topics will be determined through the CfP for this microconference.

The list of key participants for this microconference is as follows: Srinivas Pandruvada, Zhang Rui, Daniel Lezcano, Len Brown, Morten Rasmussen, Ulf Hansson, Sudeep Holla, Peter Zijlstra, Rafael Wysocki.

Since LPC 2022 we have been focusing mostly on the thermal zone device registration improvements (Unified structure for thermal zone device registration) which comprised several patch series submissions and pull requests:

- https://lore.kernel.org/linux-pm/72fcddd3-0429-4e23-ab68-2a502f451966@linaro.org
- https://lore.kernel.org/linux-pm/20230120231530.2368330-1-daniel.lezcano@linaro.org
- https://lore.kernel.org/linux-pm/20230123152756.4031574-1-daniel.lezcano@linaro.org
- https://lore.kernel.org/linux-pm/5916342.lOV4Wx5bFT@kreacher
- https://lore.kernel.org/linux-pm/20230203173331.3322089-1-daniel.lezcano@linaro.org
- https://lore.kernel.org/linux-pm/CAJZ5v0jXnjq+zRcsvUfuS=-5brPEdXw-BrFvkD8jR7kQ_ob_ww@mail.gmail.com
- https://lore.kernel.org/linux-pm/20230223224844.3491251-1-daniel.lezcano@linaro.org
- https://lore.kernel.org/linux-pm/ab323c72-61f9-9ac6-48ce-366f62e82091@linaro.org
- https://lore.kernel.org/linux-pm/CAJZ5v0h7z2iy5M+eWoA6M23rYfZ+OS54FVDjWmGNze4fR45EmA@mail.gmail.c

We have also done some work on scale-invariance improvements in accordance with an LPC 2022 topic Frequency-invariance gaps in current kernel:

- https://lore.kernel.org/linux-pm/20220407234258.569681-1-yu.c.chen@intel.com
- https://lore.kernel.org/linux-pm/20220415133356.179706384@linutronix.de

Moreover, following an LPC 2022 topic Linux per cpu idle injection, the Intel powerclamp driver has been improved to allow more fine-grained idle injection based on the generic power-capping idle-injection mechanism:

- https://lore.kernel.org/linux-pm/20230201182854.2158535-1-srinivas.pandruvada@linux.intel.com
- https://lore.kernel.org/linux-pm/20230207173219.4190013-1-srinivas.pandruvada@linux.intel.com
- https://lore.kernel.org/linux-pm/20230117182240.2817822-1-srinivas.pandruvada@linux.intel.com
- https://lore.kernel.org/linux-pm/CAJZ5v0jXnjq+zRcsvUfuS=-5brPEdXw-BrFvkD8jR7kQ_ob_ww@mail.gmail.com

Primary author: WYSOCKI, Rafael (Intel Open Source Technology Center)

Track Classification: LPC Microconference Proposals

Contribution ID: **16** Type: **not specified**

Real-time and Scheduling MC

The real-time and scheduling micro-conference joins these two intrinsically connected communities to discuss the next steps together.

Over the past decade, many parts of PREEMPT_RT have been included in the official Linux codebase. Examples include real-time mutexes, high-resolution timers, lockdep, ftrace, RCU_PREEMPT, threaded interrupt handlers, and more. The number of patches that need integration has been significantly reduced, and the rest is mature enough to make their way into mainline Linux.

The scheduler is the core of Linux performance. With different topologies and workloads, giving the user the best experience possible is challenging, from low latency to high throughput and from small power-constrained devices to HPC, where CPU isolation is critical.

The following accomplishments have been made as a result of last year's microconference:

- Progress on rtla/osnoise to support any workload 1
- Progress on adding tracepoints for IPI 2
- Improvements in RCU to reduce noise
- Progress on the latency-nice patch set 3

This year's topics to be discussed include:

- Improve responsiveness for CFS tasks - e.g., latency-nice patch
- The new EVVDF scheduler proposal 4
- Improvements in CPU Isolation
- The status of PREEMPT_RT Locking improvements - e.g., proxy execution 5
- Improvements on SCHED_DEADLINE
- Tooling for debugging scheduling and real-time 6

Links:

1 https://lore.kernel.org/lkml/cover.1669115208.git.bristot@kernel.org
2 https://lore.kernel.org/lkml/20230307143558.294354-1-vschneid@redhat.com/T/
3 https://lore.kernel.org/lkml/20230224093454.956298-3-vincent.guittot@linaro.org/
4 https://lore.kernel.org/lkml/20230328092622.062917921@infradead.org/T/
5 https://lore.kernel.org/lkml/20230411042511.1606592-1-jstultz@google.com/
6 https://lore.kernel.org/lkml/cover.1675179318.git.bristot@kernel.org/

Organizers:
- Daniel Bristot de Oliveira
- Juri Lelli
- Vincent Guittot
- Steven Rostedt

Primary authors: BRISTOT DE OLIVEIRA, Daniel (Red Hat, Inc.); LELLI, Juri (Red Hat); ROSTEDT, Steven; GUITTOT, Vincent (Linaro)

Track Classification: LPC Microconference Proposals

Contribution ID: **62** Type: **not specified**

Adaptive userspace spinlocks with rseq

Monday 13 November 2023 11:45 (20 minutes)

Implementing efficient spinlocks in userspace is not possible yet in Linux, even after years of different approaches and proposed solutions.The main gap to achieve it is the lack of ABI providing an easy and low-overhead way to check if the current lock holder is running or not.

In this session, we are going to present the problem, and to propose a solution for it using the restartable sequences infrastructure as means to expose the thread state to userspace in a cheap way, without requiring system calls.

RFC: https://lore.kernel.org/lkml/20230529191416.53955-1-mathieu.desnoyers@efficios.com/

Primary authors: ALMEIDA, André (Igalia); DESNOYERS, Mathieu (EfficiOS Inc.)

Presenters: ALMEIDA, André (Igalia); DESNOYERS, Mathieu (EfficiOS Inc.)

Session Classification: Real-time and Scheduling MC

Track Classification: LPC Microconference: Real-time and Scheduling MC

Contribution ID: **235** Type: **not specified**

CPU Isolation state of the art

Monday 13 November 2023 12:05 (20 minutes)

Here's a tour of what has been done in the front of CPU isolation
this year and what still need to be achieved. Among which topics will include examples such as:

- Memcg cache drain
- Vmstat
- Disable per-CPU buffer_head cache
- IPI deferrals
- cpusets v2 improvements
- Osnoise tracer
- Need for a nohz_full cpuset interface?
- Sysidle (energy optimization)

Primary author: WEISBECKER, Frederic (Suse)

Presenter: WEISBECKER, Frederic (Suse)

Session Classification: Real-time and Scheduling MC

Track Classification: LPC Microconference: Real-time and Scheduling MC

Contribution ID: **220** Type: **not specified**

Do nothing fast: How to scale idle cpus ?

Monday 13 November 2023 09:55 (20 minutes)

Following surprising benchmark results showing that adding a global raw spinlock in the idle loop significantly improves performance of the scheduler-heavy hackbench benchmark on a 192 core AMD EPYC, a month-long investigation followed to understand the root cause of this behavior.

This presentation is meant to walk the audience through the findings and the resulting solution, opening discussion on some of the still unexplained behaviors with respect to wakeup-queueing, going-to-idle frequency, task runqueue selection and migration frequency.

Primary author: DESNOYERS, Mathieu (EfficiOS Inc.)

Presenter: DESNOYERS, Mathieu (EfficiOS Inc.)

Session Classification: Real-time and Scheduling MC

Track Classification: LPC Microconference: Real-time and Scheduling MC

Contribution ID: **66** Type: **not specified**

How to reduce complexity in Proxy Execution

Monday 13 November 2023 11:25 (20 minutes)

The proxy execution patch series continues to be worked on to stabilize and get it ready for validation for use in products.

But its complexity is high.

I want to have a discussion for ideas on how we might break things up into more fine grained patches to iteratively get upstream, without making it an epic effort (hello, PREEMPT_RT!), or overwhelming reviewers (”[PATCH 1/628] sched:…”)

What initial half-steps might make sense? Is there value in proxy execution if we only boost locally (boost lockholder only if its on the same cpu as the selected blocked task), skipping migration initially?

I’ll also outline whatever the current state of the patch series is as of Nov.

Primary author: STULTZ, John (Google)

Presenter: STULTZ, John (Google)

Session Classification: Real-time and Scheduling MC

Track Classification: LPC Microconference: Real-time and Scheduling MC

Contribution ID: **76** Type: **not specified**

Improving CPU Isolation with per-cpu spinlocks: performance cost and analysis

Monday 13 November 2023 12:25 (15 minutes)

What do we want?
- Better CPU isolation, in order to run time-sensitive tasks without interruption

What is (one of the things) preventing this?
- queue_work_on(isolated_cpu)

While working on those, an interesting parallel programming strategy was noticed:
- Use per-cpu structures with local_lock, when a remote CPU needs any action performed, use queue_work_on(target_cpu).
- Works great for rare remote-cpu interactions, but is terrible for CPU isolation
Previous works (Mel Gorman, 01b44456) propose the usage of per-cpu spinlocks instead. But aren't spinlocks expensive?

The objective of this presentation is to show a performance analysis done on per-cpu spinlocks, presenting base info such as:
- Cache coherence & contention: why spinlocks can be expensive
- How per-cpu spinlocks prevent most of this cost?- How does that impact isolated cpus ?

And then showing the numbers:
- How many clock cycles does per-cpu spinlocks actually cost?
- What else can we do to save more cycles, and how those impact performance?
- How much of the impact can be 'hidden' by OOO execution?
- What about contention?
- How does that compare with the current solution?

Primary author: Mr SOARES PASSOS, Leonardo Bras (Red Hat)

Presenter: Mr SOARES PASSOS, Leonardo Bras (Red Hat)

Session Classification: Real-time and Scheduling MC

Track Classification: LPC Microconference: Real-time and Scheduling MC

Contribution ID: **226** Type: **not specified**

newidle load balance optimization on high core count system

Before a CPU becomes idle, it kicks off idle load balance to pull tasks from other run queues to utilize the CPU and prevent it from idling. However, this search has a potential scalability problem when the number of CPUs and sched groups in the sched domain increases.

Idle load balance potentially traverses all sched domains and calculates the statistics one by one. The time cost on idle load balance could be O(n^2), where n is the number of sched groups in the domain. Overhead becomes high if there are many groups within 1 domain. For example, a system with many Cores in a LLC domain. And there are CPUs on the horizon that this is the case. This is not good for latency, performance, or power. We'll share several proposals on statistic calculation optimization and the test results to mitigate this issue. We hope to get advice/feedback on tuning these proposals and making them viable for upstreaming.

Primary author: YU, Chen

Presenter: YU, Chen

Session Classification: Real-time and Scheduling MC

Track Classification: LPC Microconference: Real-time and Scheduling MC

Contribution ID: **178** Type: **not specified**

Optimizing Chromium Low-Power Workloads on Intel Notebooks

Monday 13 November 2023 10:35 (20 minutes)

At LPC 2022 we hosted an Energy Quality of Service (EQOS) API discussion. The proposed API enables user-space to inform the kernel about something it is expert in: itself. Callers do not require any knowledge of the hardware, unrelated tasks, or the internal workings of the scheduler. The session sparked a lot of follow-on discussions, with the main take-away being "okay, so prototype it and demonstrate value."

We will present a prototype with measurements demonstrating value. A web browser, updated to invoke the EQOS API, can save significant power during video playback and video conferencing workloads, without impacting performance when required.

The EQOS API achieves this by differentiating tasks that care about maximum performance, from those that are willing to accept performance impact in the name of energy efficiency (EE tasks). The EQOS API simply tells the kernel which is which. As the entire purpose of QOS is to differentiate tasks, the kernel saves the EQOS class in task_struct. This prototype uses that per-task EQOS class in two ways.

First, it takes advantage of Intel's Hardware Performance-State "Energy Performance Preference" setting, effectively transforming this hardware knob from per-CPU to per-task. This allows the hardware to aggressively use high frequency for performance tasks, while being mindful of the energy cost of frequency for EE tasks.

Second, we tell the scheduler's ASYM-PACKING feature to continue preferring high priority CPUs for performance tasks, but to prefer low priority CPUs for EE tasks. This has the double benefit of getting the EE tasks off the limited high-performance CPUs, at the same time as retiring the EE tasks at more energy-efficient operating points.

The concept of per-task EQOS class is agnostic to the underlying hardware architecture, and the scheduler is free to use (or ignore) this hint differently on different hardware.

Unaddressed… Latency preferences are orthogonal to EQOS. Perhaps if we had a Latency-QOS hint, wecould tell the scheduler when a task wants to run efficiently, but promptly.

Primary authors: BROWN, Len (Intel Corporation); NERI, Ricardo (Intel Corporation); Mr SHANKAR, Vaibhav (Intel Corporation)

Presenters: BROWN, Len (Intel Corporation); NERI, Ricardo (Intel Corporation); Mr SHANKAR, Vaibhav (Intel Corporation)

Session Classification: Real-time and Scheduling MC

Track Classification: LPC Microconference: Real-time and Scheduling MC

Contribution ID: **136** Type: **not specified**

Q&A about PREEMP_RT

Monday 13 November 2023 12:40 (20 minutes)

Thomas will be open to people's questions about PREEMPT RT and other topics.

Primary author: GLEIXNER, Thomas

Presenter: GLEIXNER, Thomas

Session Classification: Real-time and Scheduling MC

Track Classification: LPC Microconference: Real-time and Scheduling MC

Contribution ID: **211** Type: **not specified**

system pressure on CPUs capacity and feedback to scheduler

Monday 13 November 2023 10:15 (20 minutes)

How to reflect better the pressure that can be applied on the CPUs compute capacity into the scheduler to improve task placement deciion and load balancing.
This is a follow-up of the talk at OSPM and patchset will be published before LPC

Primary author: GUITTOT, Vincent (Linaro)

Presenter: GUITTOT, Vincent (Linaro)

Session Classification: Real-time and Scheduling MC

Track Classification: LPC Microconference: Real-time and Scheduling MC

Contribution ID: **217** Type: **not specified**

Welcome message and DL Server

Monday 13 November 2023 09:30 (25 minutes)

The DL server is a method that allows the usage of a SCHED_DEADLINE to schedule an entire scheduler. This mechanism can be used for multiple purposes. The base case is to

For example, to schedule the CFS scheduler, avoiding the starvation from SCHED_FIFO. The server's base was presented by peterz some years ago, but it raised the points. For example, the inversion of priority of CFS and FIFO tasks happening at the same time and the CFS task being scheduled before the FIFO one, breaking the main PREEMPT_RT metric (scheduling latency).

Progress was made in the last months with the addition of deferrable servers: a deferable server defers the server activation for the future.

Towards the implementation of the DL server, it is important to discuss it with the broader scheduler community to define some topics as:

- The deferable server
- The interface
- How to expand it further

Primary author: BRISTOT DE OLIVEIRA, Daniel (Red Hat, Inc.)

Presenter: BRISTOT DE OLIVEIRA, Daniel (Red Hat, Inc.)

Session Classification: Real-time and Scheduling MC

Track Classification: LPC Microconference: Real-time and Scheduling MC

Contribution ID: **18** Type: **not specified**

RISC-V MC

We'd like to propose another edition of the RISC-V microconference for Plumbers at 2023. Broadly speaking anything related to both Linux and RISC-V is on topic, but discussion tend to involve the following categories:

- How to support new RISC-V ISA features in Linux, both for the standards are for vendor-specific extensions.
- Discussions related to RISC-V based SOCs, which frequently include interactions with other Linux subsystems as well as core arch/riscv code.
- Coordination with distributions and toolchains on userspace-visible behavior.

Likely Topics for Discussion Sections

The actual list of topics tends to be hard to pin down this early, but here's a few topics that have been floating around the mailing lists and may be easier to resolve real-time:

- Do we even bother with generic optimized lib routines, or just go vendor-specific?
- When can we start deprecating stuff? Likely-unused bits include: rv32, nommu, xip, old toolchains.
- Is it time to give up on profiles and just set a base ourselves?
- CI: Hosting PW-NIPA (currently hosted by Conor/Microchip), hosting "upstream kernel ci" on Github w/ sponsored runners?
- Hardware assisted control-flow integrity on RISC-V CPUs.
- Handling text patching on RISC-V systems.
- How do we deal with vendor-specific memory management?

Key Stakeholders

Apologies if I've missed anyone, but I've tried to list a handful of the people who frequently show up and help drive discussions at the RISC-V microconferences we've held at past Plumbers:

- Palmer, Atish, Anup, Conor, Bjorn: all regular attendees and key contributors/maintainers of various RISC-V related subsystems.
- Arnd, Conor, Heiko, Emil: There are many new SOC families showing up with RISC-V ports, and much of the new
- We usually have attendance from a handful of the arm/arm64/ppc/mips/loongarch contributors/maintainers, as we share a lot of code and thus find many cross-arch issues. There's probably going to be even more now that we've got many shared SOC families.
- Carlos/Nick: Due to the nature of RISC-V we end up with many complicated toolchain interactions, so it's always good to have some time to discuss toolchain topics.

Accomplishments post 2022 Microconference

All the talks at the 2022 Plumbers microconference have made at least some progress, with many of them resulting in big chunks of merged code. Specifically:

- The riscv_hwprobe() syscall has been merged. 1
- Support for ACPI has been merged 2.
- Kconfig.socs is in the process of being refactored.
- Preliminary patches for the RISC-V TEE have been posted. 3
- Some optimized routines have been merged, but
- Text patching is still up in the air, but we've been working through many of the issues pointed out during the discussions.

1 https://lore.kernel.org/lkml/168201218504.13763.1031176103296142331.b4-ty@rivosinc.com/
2 https://lore.kernel.org/lkml/168571802526.11683.10109882495660507850.git-patchwork-notify@kernel.org/T/#m63a10dffc
3 https://lore.kernel.org/lkml/20230421153514.tpqzvdu7zt7pe7hs@amd.com/T/

Primary authors: PATRA, ATISH (Rivos); DABBELT, Palmer (Google)

Track Classification: LPC Microconference Proposals

Contribution ID: **271** Type: **not specified**

Control Flow Integrity on RISCV

Monday 13 November 2023 12:25 (20 minutes)

Memory safety issues impact program safety and integrity. One of the implications of such issues is subversion of programmer intended control flow of the program and thus violation of control flow integrity of program. There has been various software (and hardware) mechanisms using which one can enforce control flow integrity of the program. One such mechanism is using hardware assisted shadow stacks (for backward edge or return flow) and landing pad instruction (for forward edge or calls/jmps). Mainstream instruction set architectures (ISA) have extensions that can be leveraged by software to assist in enforcing control flow integrity. RISC-V has ongoing effort on similar lines to ratify an extension 1. Additionally there has been RFC patch series 2.
This talk is mostly going to be around approach and design decisions around CFI patch series seeking input from community members. Additionally this talk will go into details of how to use CFI extension to enforce kernel control flow integrity on risc kernel.

1 - https://github.com/riscv/riscv-cfi

2 - https://lore.kernel.org/lkml/20230213045351.3945824-1-debug@rivosinc.com/T/

Primary author: GUPTA, Deepak

Presenter: GUPTA, Deepak

Session Classification: RISC-V MC

Track Classification: LPC Microconference: RISC-V MC

Contribution ID: **208** Type: **not specified**

Deprecating Stuff

Monday 13 November 2023 09:35 (20 minutes)

Let's talk about what we can deprecate and when.

Primary author: DABBELT, Palmer (Google)

Presenter: DABBELT, Palmer (Google)

Session Classification: RISC-V MC

Track Classification: LPC Microconference: RISC-V MC

Contribution ID: **323** Type: **not specified**

Introduction

Monday 13 November 2023 09:30 (5 minutes)

Presenter: DABBELT, Palmer (Google)

Session Classification: RISC-V MC

Contribution ID: **198** Type: **not specified**

Perf feature improvements in RISC-V

Monday 13 November 2023 11:55 (15 minutes)

RISC-V Linux kernel has some basic perf support with counter overflow and stat until now. This has its own limitations and multiple perf related ISA extensions are being drafted to address these concerns. We would like discuss few of the existing challenges and new issues related to implementation for new ISA extensions. For example, counter event mapping, event encoding, host + guest usage for perf, kernel blind spot profiling with SSE, restricted user space access for cycle/instret.

The objective is to get an early feedback from the community to figure out the best path forward.

Primary author: PATRA, ATISH (Rivos)

Presenter: PATRA, ATISH (Rivos)

Session Classification: RISC-V MC

Track Classification: LPC Microconference: RISC-V MC

Contribution ID: **200** Type: **not specified**

Proposal of porting Trusted Execution Environment Provisioning (TEEP) Protocol with WorldGuard

Monday 13 November 2023 10:40 (20 minutes)

The objective of TEEP Protocol is to install and update the target device or server to have the latest critical software and data which is called Trusted Component (TC) at the IETF.
In the procedure, the server checks the trustworthiness of the target devices remotely whether it is compromised or not, and only installs and updates the software components if confirmed it is not compromised.

I would like to propose possibilities of porting from existing TEEP Protocol implementation on RISC-V relies on only PMP to using WorldGuard which was recently donated to RISC-V International from SiFive.
The current TEEP implementation only uses PMP hardware feature and the Keystone software stack.
It will have much practical value to have TEEP working on better hardware support which WorldGuard provides for the hardware isolated region.

I am standardizing the TEEP protocol and released the open-source reference implementation of it at the end of April 2023 at the Internet Engineering Task Force (IETF). I am also an Author which enables life cycle management of software packages as a Trusted Components (TC). The TC consists of Trusted Applications and Personalization data covers from IoT/Edge/Network devices, to drone, automotive and heavy industry equipment.

Maintaining the security bugs in the software packages is becoming challenging, while the number of software packages required for developing products is dramatically increasing every year.

The development of TEEP Protocol implementation started in late 2018.
It was a deadly psychological difficulty to repeat the approval request for releasing the source code for four years which took until April 2023.

It was initially planned to disclose in March 2019, so I could collaboratively develop with the people with Keystone project at UC Berkeley and Open Enclave project by Confidential Computing Consortium at the Linux Foundation. And making it worse, every time Keystone project releases new versions or the TEEP Protocol evolves at every IETF meeting, I had to revise the internal implementation only with limited engineers in AIST because the people of Keystone and Open Enclave do not know we was developing TEEP Protocol on top of their software stack. If I could have collaborated the implementation with the people of Keystone and the Open Enclave, then the development efforts would be much less, and the quality of the implementation would be improved by having their expertise. It was a typical bad example of working on open collaboration community like RISC-V without using the benefit of collaboration.
I hope Japanese organization will learn the good practice sometime in the future.

Primary author: TSUKAMOTO, Akira

Presenter: TSUKAMOTO, Akira

Session Classification: RISC-V MC

Track Classification: LPC Microconference: RISC-V MC

Contribution ID: **269** Type: **not specified**

RISC-V irqbypass with KVM

Monday 13 November 2023 12:45 (20 minutes)

KVM and VFIO provide an architecture-neutral irqbypass framework, but its enablement requires an implementation of an architecture-specific function, kvm_arch_irq_bypass_add_producer(). The RISC-V AIA and IOMMU specifications provide novel support for guest interrupt delivery (most notably MRIFs), which must be considered for RISC-V KVM's irqbypass implementation. We have an initial proposal which includes the RISC-V IOMMU driver implementing an IRQ domain in order to provide irq_set_vcpu_affinity(). This discussion is seeking feedback on that approach. Additionally, the RISC-V IOMMU will send notice MSIs when guest vIMSICs are backed by MRIFs, requiring a policy to select where the notice MSIs are delivered. This means we need to define new uAPI in order to involve the user. Feedback on the uAPI proposals would also be welcome. Also, we acknowledge that irqbypass performance will differ for guests with assigned interrupt files vs. those with MRIFs and we would like to discuss how best to modify or extend accounting in order to improve accuracy of measurements. Finally, the PoC is just getting started, by the time Plumbers meets, there should be enough done to have made other design decisions which could be discussed.

Primary author: JONES, Andrew (Ventana Micro Systems)

Presenter: JONES, Andrew (Ventana Micro Systems)

Session Classification: RISC-V MC

Track Classification: LPC Microconference: RISC-V MC

Contribution ID: **250** Type: **not specified**

RISC-V patchwork CI

Monday 13 November 2023 10:15 (25 minutes)

The RISC-V kernel has a number of different continous integration (CI) instances in the wild. This session covers the "patchwork CI", which pulls patches from patchwork, and reports build/test results back to the submitter. We will be presenting how the CI is setup, what builds are done, and how tests are performed. Further, we will discuss current limitations, and outline a "patchwork CI"-next plan.

We would like like to discuss gaps, ideas, concerns around the patchwork CI. Is anything missing? Can anything be removed? How can we improve the developer experience, and improve the quality of the submissions, pre-merge.

Primary author: TÖPEL, Björn (N/A)

Presenter: TÖPEL, Björn (N/A)

Session Classification: RISC-V MC

Track Classification: LPC Microconference: RISC-V MC

Contribution ID: **169** Type: **not specified**

RISC-V Vector: Current Status and Next?

Monday 13 November 2023 12:10 (15 minutes)

In this talk we are going to briefly share the status of Vector extension support and focus our discussion on the use of Vector in the kernel-mode. We will do it by reviewing others arch approaches and seeking if there is anything we may carry or improve as risc-v.

Most architectures provide SIMD instruction set to improve throughput of some operations. However, the use of SIMD instructions in their kernel mode is often restricted due to latency considerations for extra state-keeping. For example, it is not uncommon to see an arch that disables preemption when using kernel-mode SIMD. Also, msot ban the use of SIMD in interrupt context.

Among these architectures, seldom provide SIMD-optimized common sub-routines (mem/str ops). PowerPC provides vsx/vmx-optimized common routines with a precondition and a side-effect. First, it cannot leverage those routines in interrupt context. And, it must disable kernel preemption while using these subroutines. Meanwhile, though the same side effect applies to x86, it provides irq_fpu_usable() to allow some level of SIMD uses in interrupt context.

On risc-v, should we follow the path of powerPC? If yes, how do we decide when to use it? Vendors may have varies performance characteristic of V and using SIMD for small inputs may not gain. Should we do runtime detection for this or just enable V whenever the hardware supports?

Further, should risc-v take a step forward, by enabling preemption for its kernel-mode SIMD <a href="https://lore.kernel.org/all/20230721112855.1006-1-andy.chiu@sifive.com/">1</a>? Supporting kernel preemption during Vector execution may enable us to widely use Vector in kernel thread or syscalls while remain same level of responsiveness. Historically, the reason for disabling kernel preemption while using kernel-mode SIMD is the per-cpu variable consideration <a href="https://yarchive.net/comp/linux/kernel$_fp.html" > 2 < /a > .However, theper - cpuFPUcacheisphasingoutonx86andisnotpresentonrisc-v'sapproach.So, itmightbeagoodtimingfordiscussingif$

The talk will cover the following topics:

- The current status of Vector extension support
- Basic idea of supporting kernel-mode Vector
- The use of Vector optimized sub-routines for us and other arch
- Should we support running kernel-mode Vector with preemption?

1 https://lore.kernel.org/all/20230721112855.1006-1-andy.chiu@sifive.com/
2 https://yarchive.net/comp/linux/kernel_fp.html

Primary author: CHIU, Tao

Presenter: CHIU, Tao

Session Classification: RISC-V MC

Track Classification: LPC Microconference: RISC-V MC

Contribution ID: **174** Type: **not specified**

Run ILP32 on RV64 ISA (RV64ILP32)

Monday 13 November 2023 09:55 (20 minutes)

The 64ilp32 ABI is not a fresh topic; x86-x32, mips-n32, and arm64-ilp32 have all appeared for many years but have yet to succeed in wide usage. But running ILP32 on 64-bit ISA still has a magic power to abstract people for continuous trying; now, this is our turn. The rv64ilp32 patch series has iterated to the second version, combining u64ilp32 (User) & s64ilp32 (kernel), supporting the 64ilp32 kernel for the first time. In the presentation, we will show the advantages of 64ILP32 from different views:
- ILP32 v.s. LP64 (SPECint 2006 & 2017 scores)
- RV64 v.s. RV32 ISA (memcpy, ebpf, atomic64, DCAS …)
- Run 64ilp32 Fedora on all popular platforms.

The conclusion is that 64ilp32 would help the existing RISC-V hardware platforms by increasing performance and decreasing cost.

Primary author: Mr GUO, Ren

Presenter: Mr GUO, Ren

Session Classification: RISC-V MC

Track Classification: LPC Microconference: RISC-V MC

Contribution ID: **228** Type: **not specified**

SBI Supervisor Software Events

Monday 13 November 2023 11:30 (25 minutes)

The Supervisor Software Events (SSE) extension provides a
mechanism to inject software events from an SBI implementation
to supervisor software such that it preempts all other traps and
interrupts. This brings interesting challenges for the SBI implementation (OpenSBI,KVM RISC-V, etc) and supervisor software (Linux).

Primary authors: PATEL, Anup (Ventana Micro Systems); Mr LÉGER, Clément (Rivos Inc); CHAUHAN, Himanshu (Ventana Micro Systems)

Presenter: Mr LÉGER, Clément (Rivos Inc)

Session Classification: RISC-V MC

Track Classification: LPC Microconference: RISC-V MC

Contribution ID: **24** Type: **not specified**

Rust

Rust is a systems programming language that is making great strides in becoming the next big one in the domain.

Rust for Linux is the project adding support for the Rust language to the Linux kernel. Rust has a key property that makes it very interesting as the second language in the kernel: it guarantees no undefined behavior takes place (as long as unsafe code is sound). This includes no use-after-free mistakes, no double frees, no data races, etc. It also provides other important benefits, such as improved error handling, stricter typing, sum types, pattern matching, privacy, closures, generics, etc.

This microconference intends to cover talks and discussions on both Rust for Linux as well as other non-kernel Rust topics.

Possible Rust for Linux topics:

- Rust in the kernel (e.g. status update, next steps…).
- Use cases for Rust around the kernel (e.g. subsystems, drivers, other modules…).
- Discussions on how to abstract existing subsystems safely, on API design, on coding guidelines…
- Integration with kernel systems and other infrastructure (e.g. build system, documentation, testing and CIs, maintenance, unstable features, architecture support, stable/LTS releases, Rust versioning, third-party crates…).
- Updates on its subprojects (e.g. klint, pinned-init…).

Possible Rust topics:

- Language and standard library (e.g. upcoming features, stabilization of the remaining features the kernel needs, memory model…).
- Compilers and codegen (e.g. rustc improvements, LLVM and Rust, rustc_codegen_gcc, Rust GCC…).
- Other tooling and new ideas (bindgen, Cargo, Miri, Clippy, Compiler Explorer, Coccinelle for Rust…).
- Educational material.
- Any other Rust topic within the Linux ecosystem.

Last year was the first edition of the Rust MC and the focus was on showing the ongoing efforts by different parties (compilers, Rust for Linux, CI, eBPF…). Shortly after the Rust MC, Rust got merged into the Linux kernel. Abstractions are getting upstreamed, with the first major drivers looking to be merged soon: Android Binder, the Asahi GPU driver and the NVMe driver (presented in that MC).

Primary authors: OJEDA, Miguel; ALMEIDA FILHO, Wedson

Track Classification: LPC Microconference Proposals

Contribution ID: **35** Type: **not specified**

Tracing Microconference

The Linux kernel has grown in complexity over the years. Complete understanding of how it works via code inspection has become virtually impossible. Today, tracing is used to follow the kernel as it performs its complex tasks. Tracing is used today for much more than simply debugging. Its framework has become the way for other parts of the Linux kernel to enhance and even make possible new features. Live kernel patching is based on the infrastructure of function tracing, as well as BPF. It is now even possible to model the behavior and correctness of the system via runtime verification which attaches to trace points. There is still much more that is happening in this space, and this microconference will be the forum to explore current and new ideas.

Results and accomplishments from the last time (2021):

- User events were introduced, and have finally made it into the kernel
- The discussion around trace events to handle user faults initiated the event probe work around to the problem. That was to add probes on existing trace events to change their types. This works on synthetic events that can pass the user space file name of the entry of a system call to the exit of the system call which would have faulted in the file and make it available to the trace event.
- Dynamically creating the events directory is currently being worked on with the eventfs patch set. This will save memory as the dentries and inodes will only be allocated when accessed.
- The discussion about function tracing with arguments has helped inspire both fprobes and function graph return value tracing.
- There's still ongoing effort in merging the return path tracers of function graph and kretprobes and fprobes.

Topics for this year:

- Use of sframes. How to get user space stack traces without requiring frame pointers.
- Updating perf and ftrace to extract user space stack frames from a schedulable context (as requested by NMI).
- Extending user events. Now that they are in the kernel, how to make them more accessible to users and applications.
- Getting more use cases with the runtime verifier. Now that the runtime verifier is in the kernel (uses tracepoints to model against), what else can it be used for.
- Wider use of ftrace_regs in fprobes and rethook from fprobes because rethook may not fill all registers in pt_regs too. How BPF handles this will also be discussed.
- Removing kretprobes from kprobes so that kprobe can focus on handling software breakpoint.
- Object tracing (following a variable throughout each function call). This has had several patches out, but has stopped due to hard issues to overcome. A live discussion could possibly come up with a proper solution.
- Hardware breakpoints and tracing memory changes. Object tracing follows a variable when it changes between function calls. But if the hardware supports it, tracing a variable when it actually changes would be more useful albeit more complex. Discussion around this may come up with a easier answer.

- MMIO tracer being used in SMP. Currently the MMIO tracer does not handle race conditions. Instead, it offlines all but one CPU when it is enabled. It would be great if this could be used in normal SMP environments. There's nothing technically preventing that from happening. It only needs some clever thinking to come up with a design to do so.
- Getting perf counters onto the ftrace ring buffer. Ftrace is designed for fast tracing, and perf is a great profiler. Over the years it has been asked to have perf counters along side ftrace trace events. Perhaps its time to finally accomplish that. It could be that each function can show the perf cache misses of that function.

Key attendees:
- Steven Rostedt
- Masami Hiramtsu
- Mathieu Desnoyers
- Alexei Starovoitov
- Peter Zijlstra
- Mark Rutland
- Beau Belgrave
- Daniel Bristot de Oliveira
- Florent Revest
- Jiri Olsa
- Tom Zanussi

Primary authors: Mr HIRAMATSU, Masami (Google); ROSTEDT, Steven

Track Classification: LPC Microconference Proposals

Contribution ID: **29** Type: **not specified**

VFIO/IOMMU/PCI MC

The PCI interconnect specification, the devices that implement it, and the system IOMMUs that provide memory and access control to them are nowadays a de-facto standard for connecting high-speed components, incorporating more and more features such as:

- Address Translation Service (ATS)/Page Request Interface (PRI)
- Single-root I/O Virtualization (SR-IOV)/Process Address Space ID (PASID)
- Shared Virtual Addressing (SVA)
- Remote Direct Memory Access (RDMA)
- Peer-to-Peer DMA (P2PDMA)
- Cache Coherent Interconnect for Accelerators (CCIX)
- Compute Express Link (CXL)
- Data Object Exchange (DOE)
- Component Measurement and Authentication (CMA)
- Integrity and Data Encryption (IDE)
- Security Protocol and Data Model (SPDM)
- Gen-Z

These features are aimed at high-performance systems, server and desktop computing, embedded and SoC platforms, virtualisation, and ubiquitous IoT devices.

The kernel code that enables these new system features focuses on coordination between the PCI devices, the IOMMUs they are connected to, and the VFIO layer used to manage them (for userspace access and device passthrough) with related kernel interfaces and userspace APIs to be designed in-sync and in a clean way for all three sub-systems.

The VFIO/IOMMU/PCI MC focuses on the kernel code that enables these new system features, often requiring coordination between the VFIO, IOMMU and PCI sub-systems.

Following the success of LPC 2017, 2019, 2020, 2021, and 2022 VFIO/IOMMU/PCI MC, the Linux Plumbers Conference 2023 VFIO/IOMMU/PCI track will focus on promoting discussions on the PCI core but also current kernel patches aimed at VFIO/IOMMU/PCI sub-systems with specific sessions targeting discussions requiring the three sub-systems coordination.

See the following video recordings from 2022: LPC 2022 - VFIO/IOMMU/PCI MC

Older recordings can be accessed through our official YouTube channel at @linux-pci and the archived LPC 2017 VFIO/IOMMU/PCI MC web page at Linux Plumbers Conference 2017, where the audio recordings from the MC track and links to presentation materials are available.

The tentative schedule will provide an update on the current state of VFIO/IOMMU/PCI kernel sub-systems, followed by a discussion of current issues in the proposed topics.

The following was a result of last year's successful Linux Plumbers MC:

- Support for the /dev/iommufd device has been merged into the mainline kernel
- A discussion has been kicked off around the topic of the Instant Detection of Virtual Devices

- The work on the PCIe Endpoint Notifier has been completed and merged into the mainline kernel

Tentative topics that are under consideration for this year include (but are not limited to):

- PCI
  - Cache Coherent Interconnect for Accelerators (CCIX)/Compute Express Link (CXL) expansion memory and accelerators management
  - Data Object Exchange (DOE)
  - Integrity and Data Encryption (IDE)
  - Component Measurement and Authentication (CMA)
  - Security Protocol and Data Model (SPDM)
  - I/O Address Space ID Allocator (IOASID)
  - INTX/MSI IRQ domain consolidation
  - Gen-Z interconnect fabric
  - ARM64 architecture and hardware
  - PCI native host controllers/endpoints drivers current challenges and improvements (e.g., state of PCI quirks, etc.)
  - PCI error handling and management, e.g., Advanced Error Reporting (AER), Downstream Port Containment (DPC), ACPI Platform Error Interface (APEI) and Error Disconnect Recover (EDR)
  - Power management and devices supporting Active-state Power Management (ASPM)
  - Peer-to-Peer DMA (P2PDMA)
  - Resources claiming/assignment consolidation
  - Probing of native PCIe controllers and general reset implementation
  - Prefetchable vs non-prefetchable BAR address mappings
  - Untrusted/external devices management
  - DMA ownership models
  - Thunderbolt, DMA, RDMA and USB4 security
- VFIO
  - Write-combine on non-x86 architectures
  - I/O Page Fault (IOPF) for passthrough devices
  - Shared Virtual Addressing (SVA) interface
  - Single-root I/O Virtualization(SRIOV)/Process Address Space ID (PASID) integration
  - PASID in SRIOV virtual functions
  - Device assignment/sub-assignment
- IOMMU
  - /dev/iommufd development
  - IOMMU virtualisation
  - IOMMU drivers SVA interface
  - DMA-API layer interactions and the move towards generic dma-ops for IOMMU drivers
  - Possible IOMMU core changes (e.g., better integration with the device-driver core, etc.)

If you are interested in participating in this MC and have topics to propose, please use the Call for Proposals (CfP) process. More topics might be added based on CfP for this MC.

Otherwise, join us to discuss helping Linux keep up with the new features added to the PCI interconnect specification. We hope to see you there!

Key Attendees:

- Alex Williamson
- Arnd Bergmann
- Ashok Raj
- Benjamin Herrenschmidt
- Bjorn Helgaas
- Dan Williams
- Eric Auger
- Jacob Pan
- Jason Gunthorpe
- Jean-Philippe Brucker
- Jonathan Cameron
- Jörg Rödel
- Kevin Tian
- Lorenzo Pieralisi
- Lu Baolu
- Marc Zyngier
- Pali Rohár
- Peter Zijlstra
- Thomas Gleixner

Contacts:

- Alex Williamson (alex.williamson@redhat.com)
- Bjorn Helgaas (bhelgaas@google.com)
- Jörg Roedel (jroedel@suse.de)
- Lorenzo Pieralisi (lorenzo.pieralisi@linaro.org)
- Krzysztof Wilczyński (kw@linux.com)

Primary authors: WILLIAMSON, Alex; HELGAAS, Bjorn (Google); ROEDEL, Joerg (SUSE); Mr WILCZYŃSKI, Krzysztof; PIERALISI, Lorenzo

Track Classification: LPC Microconference Proposals

Contribution ID: **183** Type: **not specified**

Improve Xeon IRQ throughput with posted interrupt

Wednesday 15 November 2023 09:30 (30 minutes)

Server SoCs today offer more PCIe lanes as well as the ability to stack more IO devices on a single port. Out of the box, devices such as high-speed NVMe drives can generate a significant number of interrupts at high frequencies. Due to microarchitecture choice and PCIe strong ordering requirements, limited IRQ throughput on Intel Xeon has also become a limiting factor for DMA throughput. IOPS can drop more than 50% as evidenced by the FIO test on multiple NVMe disks. This talk describes a scheme and RFC patch that optimize IRQ throughput by enabling posted interrupts on bare metal (beyond the VM usage today). The result is that much of the performance loss can be recovered without any new hardware or driver changes.

Primary author: PAN, Jacob

Presenter: PAN, Jacob

Session Classification: VFIO/IOMMU/PCI MC

Track Classification: LPC Microconference: VFIO/IOMMU/PCI MC

Contribution ID: **256** Type: **not specified**

IOMMU overhead optimizations and observability

Wednesday 15 November 2023 11:30 (45 minutes)

IOMMU overhead memory, which is primarily page table memory, is allocated directly from the buddy allocator, and is not charged or accounted for. Also, there is no easy way to debug IOMMU translations as there are no user interfaces that allow walking through IOMMU page tables. Below are the proposals to solve the problems.

Add an observability for IOMMU page table memory into /proc/meminfo:

```
PageTables:        XXX kB
SecPageTables:     XXX kB
IOMMUPageTables:   XXX kB
```

This would allow users to see how much IOMMU page table memory is
being used, which could help them identify and troubleshoot performance problems.

**Charge the IOMMU page table memory to the proper owner when DMA
mappings are established:**

This would allow users to control and limit the amount of IOMMU page table memory that is used by each process.

Allow walking through IOMMU page tables on live systems and in kdumps:

This would allow users to debug IOMMU translations and identify problems.

For live systems the interface should be similar to /proc/PID/pagemap, so users could walk through IOMMU page tables, and study which physical pages are currently mapped into page tables.

For kdumps, it should be a crash-utility extension to dump IOMMU page tables.

Limit the growth of page tables:

Currently, when pages are removed unmapped from the page table, the
free page table levels are not returned back to the system, see 1
for example. This can cause substantial overheads in cases where VA
addresses are not recycled. On the other hand, recycling VA addresses in order to save memory can be a security risk, and in general a bad practice.

We propose to limit the maximum number of empty page table levels to a certain amount.

Add iova_stress1 into kernel selftest:

This would allow us to verify that page table overhead does not
regress in the future.

1 https://github.com/soleen/iova_stress

Primary authors: TATASHIN, Pasha; ZHAO, Yu (Google)

Presenters: TATASHIN, Pasha; ZHAO, Yu (Google)

Session Classification: VFIO/IOMMU/PCI MC

Track Classification: LPC Microconference: VFIO/IOMMU/PCI MC

Contribution ID: **110** Type: **not specified**

iommufd discussion

Wednesday 15 November 2023 12:15 (45 minutes)

Open discussion on iommufd topics that have not been settled on the mailing list prior to the conference:

- IOMMU based dirty tracking
- IOMMU nested translation
- IOMMU userspace command queue
- Unique driver features
- iommufd support of SVA/PRI/PASID
- ARM interrupt handling in VMs
- Driver enablement for iommufd features

Primary authors: Mr GUNTHORPE, Jason (NVIDIA Networking); TIAN, Kevin (Intel)

Presenters: Mr GUNTHORPE, Jason (NVIDIA Networking); TIAN, Kevin (Intel)

Session Classification: VFIO/IOMMU/PCI MC

Track Classification: LPC Microconference: VFIO/IOMMU/PCI MC

Contribution ID: **203** Type: **not specified**

Non-discoverable devices in PCI devices

Wednesday 15 November 2023 10:30 (30 minutes)

Modern PCI devices can expose a whole slew of hardware behind a single PCI "device". While the PCI device itself is discoverable, everything behind it (via BARs) is not. These devices aren't fixed in what downstream devices are exposed nor their configuration. There's already a solution for discovering devices and their configuration which is Devicetree. There's also already a mechanism to dynamically add to a DT with DT overlays.

Specific use cases:

- AMD/Xilinx Alveo Accelerator cards - FPGA based PCIe card. Multiple downstream devices (hardware peripherals) exposed through PCI BARs. The downstream devices may be unrelated and already have a driver (platform bus) in the kernel.
- Microchip LAN9662 Ethernet controller - An SoC already supported upstream using Devicetree. This device also has a PCIe endpoint which can expose the entire SoC to Linux. Using DT to describe the SoC allows reusing all the drivers as-is.
- roadtest - A testing framework which exposes platform devices behind a virtual PCI device in UML. See https://lore.kernel.org/all/20230120-simple-mfd-pci-v1-1-c46b3d6601ef@axis.com/

What's needed in the kernel:

- Generating PCI device DT nodes - PCI devices can already be described in DT, but usually are not. In order to have a base to apply DT overlays to, the PCI device needs a DT node. That can be solved by generating the DT node (and parent nodes) for the device. With this, the PCI driver for the device can load and apply DT overlays to its node. See https://lore.kernel.org/all/1692120000-46900-1-git-send-email-lizhi.hou@amd.com/ (Should be upstream in 6.6)
- Enable Devicetree on Non-DT systems (x86_64) - A solution for ACPI based systems is desired as well. There's SSDT overlays for ACPI which might work?, but many of the devices to support don't use ACPI. With the generated PCI nodes, we just need the PCI host bridge and hooking up a few host resources (interrupts, MSI) to reuse the same DT overlay mechanism. A skeleton base DT is needed as well. That already exists as the DT unittest creates one for non-DT systems. There may be other issues lurking where we make DT vs. ACPI decisions in the kernel.

Questions:
- What about swnode and/or auxiliary bus?

Primary authors: HOU, Lizhi (AMD); HERRING, Rob (Arm)

Presenters: HOU, Lizhi (AMD); HERRING, Rob (Arm)

Session Classification: VFIO/IOMMU/PCI MC

Track Classification: LPC Microconference: VFIO/IOMMU/PCI MC

Contribution ID: **119** Type: **not specified**

PCI Endpoint Subsystem Open Items Discussion

Wednesday 15 November 2023 10:00 (30 minutes)

PCI Endpoint subsystem allows Linux Kernel to run on the PCI endpoint devices thereby establishing communication with the PCI host for data transfer. There are 3 open items to discuss for the PCI Endpoint subsystem:

1. The heart of the PCI Endpoint subsystem is the Endpoint Function (EPF) driver that describes the Physical and Virtual functions inside the Endpoint device. So far 3 EPF drivers were supported in the upstream Linux kernel. Recently, there are attempts to add VIRTIO-based EPF drivers for interoperability. This discussion aims at presenting current and past proposals and getting feedback on the desired approach.
2. Most of the EPF drivers that exist today are virtual function drivers (i.e. not backed by a hardware entity) except Modem Host Interface (MHI) for Qcom platforms. So there is a requirement to describe those functions in devicetree and binding and also allow the EPF drivers to bind with EPC during boot without ConfigFS intervention (using devicetree link between EPC and EPF).
3. The PCI Endpoint subsystem uses a custom memory allocator for managing the PCI outbound window memory. But it could make use of the Linux kernel's generic "genalloc/genpool subsystem".

Primary author: SADHASIVAM, Manivannan

Presenter: SADHASIVAM, Manivannan

Session Classification: VFIO/IOMMU/PCI MC

Track Classification: LPC Microconference: VFIO/IOMMU/PCI MC