Linux Plumbers Conference 2024

Report of Contributions

Contribution ID: **23** Type: **not specified**

Android MC

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.

Some highlights of progress made since last year's MC:

- For fw_devlink, got post-init-providers accepted into DT schema, as proposed and discussed at LPC. Additionally, as proposed at LPC, fw_devlink=rpm was made the default, so fw_devlink now enforces runtime PM ordering too.
- After discussions last year on board-id property being used to pick a DTB by the bootloader, patches for a shared solution were submitted upstream.
- Initial Pixel6 support has landed upstream, such that it can boot to console with 6.9-rc kernels.
- Having the chance to connect with the right glibc people facilitating a consensus between the bionic folks and the clang driver/lld ELF owners on an approach to mitigate the VMA (Virtual Memory Area) slab memory increase caused by the dynamic linker in devices supporting larger than 4KB page sizes.
- Discussion with the BPF ring buffer maintainer led to the event driven memory notifications from the kernel for low memory killer daemon (lmkd).

Also, listening to feedback from last year, we are planning to have slightly longer slots, so talks are not so rushed, but that also means we will have to be even more selective with topics.

Potential discussion topics for this year include:

- Device Longevity
- Power usage modeling and simulations
- Unified bootloader efforts
- The Power of Perfetto
- Using & tuning with the (soon to be) upstream Dynamic Energy Model
- Android Storage features: ublk, ureadhead, FUSE-BPF
- AVF updates&plans / pVM firmware
- More discussion on 16k pages
- RISCV updates

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

Presenters: PUNDIR, Amit; TABBA, Fuad (Google); STULTZ, John (Google); YAGHMOUR, Karim (Opersys inc.); LUBA, Lukasz; SEMWAL, Sumit (Linaro)

Contribution ID: **11** Type: **not specified**

Graphics & DRM MC

The Graphics & DRM Microconference welcomes the community to discuss topics around the Linux graphics stack and the DRM subsystem, with the goal of solving long standing and complex problems together.

The MC CfP is open to all proposals related to graphics, including the following potential topics:

- Rust and DRM
- Color management and HDR
- Automated tests of GPUs and the stack
- cgroups support
- Device reset management
- DRM and IA accelerators

MC Leads:
- André Almeida
- Daniel Stone

Primary authors: ALMEIDA, André (Igalia); STONE, Daniel (Collabora)

Presenter: ALMEIDA, André (Igalia)

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

Kernel Memory Management MC

Memory management has become exciting again. Some controversial subjects which might merit discussion:

- Should we add memory policy zones?
- How far should we go to support CXL?
- How do we handle page allocation in a memdesc world?
- Should we switch the slab allocator from partial slabs to sheaves?
- Can we get rid of non-compound multi-page allocations?
- What other improvements might we see from mTHP?
- How might we make allocations guaranteed to not fail?
- Can we share the pagecache between reflinked files?
- Is there a better way to share page tables between processes than hugetlb? -

Primary authors: WILCOX, Matthew (Oracle); BABKA, Vlastimil (SUSE Labs)

Contribution ID: **6** Type: **not specified**

Kernel Testing & Dependability MC

The Linux Plumbers 2024 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 kernel testing and dependability. This will help move the conversation forward from where we left off at the LPC 2023 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: Improving user experience and new web dashboard (https://github.com/kernelci/kernelci-project/discussions/28)

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:

Storing and Outputting Test Information: KUnit Attributes and KTAPv2 has been upstreamed.

KUnit APIs for managing devices has been upstreamed.

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

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

Presenters: TUCKER, Guillaume; LEVIN, Sasha; LEVIN, Sasha; KHAN, Shuah (The Linux Foundation); KHAN, Shuah

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

KVM Microconference

KVM (Kernel-based Virtual Machine) enables the use of hardware features to improve the efficiency, performance, and security of virtual machines (VMs) created and managed by userspace. KVM was originally developed to accelerate VMs running a traditional kernel and operating system, in a world where the host kernel and userspace are part of the VM's trusted computing base (TCB).

KVM has long since expanded to cover a wide (and growing) array of use cases, e.g. sandboxing untrusted workloads, deprivileging third party code, reducing the TCB of security sensitive workloads, etc. The expectations placed on KVM have also matured accordingly, e.g. functionality that once was "good enough" no longer meets the needs and demands of KVM users.

The KVM Microconference will focus on how to evolve KVM and adjacent subsystems in order to satisfy new and upcoming requirements. Of particular interest is extending and enhancing guest_memfd, a guest-first memory API that was heavily discussed at the 2023 KVM Microconference, and merged in v6.8.

The KVM MC is expected to have strong representation from maintainers (KVM and non-KVM), hardware vendors (Intel, AMD, ARM, RISC-V, etc), cloud (AWS, Google, Oracle, etc), client (Android, ChromeOS), and open source stalwarts such as Red Hat and SUSE.

Potential Topics:
- Removing guest memory from the host kernel's direct map[1]
- Mapping guest_memfd into host userspace[2]
- Hugepage support for guest_memfd[3]
- Eliminating "struct page" for guest_memfd
- Passthrough/mediated PMU virtualization[4]
- Pagetable-based Virtual Machine (PVM)[5]
- Optimizing/hardening KVM usage of GUP[6][7]
- Live migration support for guest_memfd
- Defining KVM requirements for hardware vendors
- Utilizing "fault" injection to increase test coverage of edge cases

[1] https://lore.kernel.org/all/cc1bb8e9bc3e1ab637700a4d3defeec95b55060a.camel@amazon.com
[2] https://lore.kernel.org/all/20240222161047.402609-1-tabba@google.com
[3] https://lore.kernel.org/all/CABgObfa=DH7FySBviF63OS9sVog_wt-AqYgtUAGKqnY5Bizivw@mail.gmail.com
[4] https://lore.kernel.org/all/20240126085444.324918-1-xiong.y.zhang@linux.intel.com
[5] https://lore.kernel.org/all/20240226143630.33643-1-jiangshanlai@gmail.com
[6] https://lore.kernel.org/all/CABgObfZCay5-zaZd9mCYGMeS106L055CxsdOWWvRTUk2TPYycg@mail.gmail.com
[7] https://lore.kernel.org/all/20240320005024.3216282-1-seanjc@google.com

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

Presenters: BONZINI, Paolo (Red Hat); CHRISTOPHERSON, Sean (Google)

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

Rust MC

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, gccrs…).
- Other tooling and new ideas (Coccinelle for Rust, bindgen, Compiler Explorer, Cargo, Clippy, Miri…).
- Educational material.
- Any other Rust topic within the Linux ecosystem.

Last year was the second edition of the Rust MC and the focus was on presenting and discussing the ongoing efforts by different parties that are using and upstreaming new Rust abstractions and drivers (Using Rust in the binder driver, Block Layer Rust API, Rust in V4L2: a status report and Converting a DRM driver to Rust) as well as those that are improving the ergonomics and tooling around it (Klint: Compile-time Detection of Atomic Context Violations for Kernel Rust Code, pin-init: Solving Address Stability in Rust and Coccinelle for Rust).

Since the MC last year, there has been continued progress from users (e.g. the Android Binder Driver getting closer to upstreaming all its dependencies) as well as new project announcements (e.g. Nova), the first Rust reference driver merged together with its abstractions (the Rust Asix PHY driver), Rust support for new architectures mainlined (LoongArch and arm64)…

Primary authors: OJEDA, Miguel; ALMEIDA FILHO, Wedson

Presenters: OJEDA, Miguel; ALMEIDA FILHO, Wedson

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

Safe Systems with Linux

As Linux is increasingly deployed in systems with varying criticality constraints, distro providers are being expected to ensure that security fixes in their offerings do not introduce regressions for customer products that have safety considerations. The key question arises: How can they establish consistent linkage between code, tests, and the requirements that the code satisfies?

This MC addresses critical challenges in requirements tracking, documentation, testing, and artifact sharing within the Linux kernel ecosystem. Functionality has historically been added to the kernel with requirements explained in the email justifications for adding, but not formalized as "requirements"in the kernel documentation. While tests are contributed for the code, the underlying requirement that the tests satisfies is likewise not documented in a consistent manner.

Potential topics to be discussed:

- where should requirements that the kernel code and testing satisfies be tracked? In kernel documentation, in the code, etc.
- incorporating requirement linkage to the kernel code and tests that minimizes the impact to kernel maintainers and contributors.
- examples and strategies for enhancing documentation quality and level of detail within the Linux kernel so that effective safety analysis can be performed for products. Some starting points have been started [1], but what else is needed.
- connecting artifacts in a shareable format: how to effectively link and share testing, documentation, bug reports, and CVE information across multiple projects, infrastructures, and contribution processes.
- traceability and change identification in requirements to keep in sync with the evolving kernel code functionality and security fixes.
- increasing code test coverage of the Linux kernel to satisfy the higher safety assurance considerations. There's been some recent studies conducted by Boeing and the University of Illinois on various coverage types, that should be considered.
- requirements introduced by the Cyber Resilience Act in the EU [2] on product manufacturers might have on the Linux Kernel development process and documentation.
- improving systematic error responses when using Linux as well as runtime verification monitoring.

Last year, we had several talks on the need for safe systems [3][4] in various domains with Linux as a component (with varying safety criticality levels). This miniconference is targetted at getting those interested together, and working up a framework for collecting relevant evidence and sharing it.

MC Leads:
Kate Stewart, Philipp Ahmann

Potential Participants (not confirmed yet):

Syed Mohammed Khasim
Jonathan Corbet
Shuah Khan
Greg Kroah-Hartman
Chuck Wobler
Daniel Bristot de Oliveira

Thomas Gleixner
Gabrielle Paoloni
Olivier Charrier
Jiri Kosina
Joachim Werner
Paul Albertela
Bertrand Boisseau

[1] https://docs.kernel.org/admin-guide/workload-tracing.html
[2] https://digital-strategy.ec.europa.eu/en/policies/cyber-resilience-act
[3] https://lpc.events/event/17/contributions/1499/
[4] https://lpc.events/event/17/contributions/1518/

Primary authors: STEWART, Kate (Linux Foundation); AHMANN, Philipp (Robert Bosch GmbH)

Presenters: STEWART, Kate (Linux Foundation); AHMANN, Philipp (Robert Bosch GmbH)

Contribution ID: **13** Type: **not specified**

Sched MC

The scheduler is at 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.
The following accomplishments have been made as a result of last year's micro-conference:

- Progress on proxy execution https://lore.kernel.org/lkml/20240224001153.2584030-1-jstultz@google.com/
- Progress on system pressure https://lore.kernel.org/lkml/170073688055.398.12687414937207369825.tip-bot2@tip-bot2/ https://lore.kernel.org/lkml/20240220145947.1107937-1-vincent.guittot@linaro.org/
- Progress in the DL server
- The EEVDF scheduler and improvements in latency nice
- Progress on adding tracepoints for IPI

Ideas of topics to be discussed include (but are not limited to):

- Improve responsiveness for CFS tasks
- The improvements on the EEVDF scheduler proposal
- Impact of new topology on CFS, including hybrid or heterogeneous system
- Taking into account task profile with IPCC or uclamp
- Locking improvements –e.g., proxy execution
- Improvements on SCHED_DEADLINE
- Tooling for debugging scheduling

It is fine if you have a new topic not on the list. People are encouraged to submit any topic related to real-time and scheduling.
The goal is to discuss open problems, preferably with patch set submissions already in discussion on LKML. The presentations are concise, and the central portion of the time should be given to the debate –thus, the importance of having an open and relevant problem with people in the community engaged in the solution.

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

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

Contribution ID: **27** Type: **not specified**

Tracing / Perf events 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.

This year, focus will also be on perf events:

Perf events are a mechanism for presenting performance counters and software events that occur running Linux to users. There are kernel and userland components to perf events, with the kernel presenting or extending APIs and the perf tool presenting this to users

Results and accomplishments from the last time (2023):

- Masami's work on accessing function entry data from function *return* probes (kprobe and fprobe) was merged for v6.9.
- eventfs is now dynamically created and fully working following *robust* discussions with Linus.
- Work on sframes was paused due to other priorities but is still a topic of interest.
- Discussions on integrating User events with libside are ongoing.
- User events added multi-format events.

Topics for this year:

- Feedback about the tracing/perf subsystems overall (e.g. how can people help the maintainers).
- Reboot persistent in-memory tracing buffers, this would make ftrace a very powerful debugging and performance analysis tool for kexec and could also be used for post crash debugging.
- Dynamic change of ftrace events to improve symbolic printing.
- Userspace instrumentation (libside), including discussion of its impacts on the User events ABI.
- Collect state dump events from kernel drivers (e.g. dump wifi interfaces configuration at a given point in time through trace buffers).
- Current work implementing performance monitoring in the kernel,
- User land profiling and analysis tools using the perf event API,
- Improving the kernel perf event and PMU APIs,
- Interaction between perf events and subsystems like cgroups, kvm, drm, bpf, etc.,
- Improving the perf tool and its interfaces in particular w.r.t. to scalability of the tool,
- Implementation of new perf features and tools using eBPF, like the ones in tools/perf/util/bpf_skel/.
- Further use of type information to augment the perf tools,

- Novel uses of perf events for debugging and correctness,
- New challenges in performance monitoring for the Linux kernel,
- Regression testing/CI integration for the perf kernel infrastructure and tools,
- Improving documentation,
- Security aspects of using tracing/perf tools,

Key attendees:

- Steven Rostedt
- Masami Hiramatsu
- Mathieu Desnoyers
- Alexei Starovoitov
- Peter Zijlstra
- Mark Rutland
- Beau Belgrave
- Daniel Bristot de Oliveira
- Florent Revest
- Jiri Olsa
- Tom Zanussi
- Alexander Graf
- Johannes Berg
- Arnaldo Carvalho de Melo
- Ian Rogers
- Namhyung Kim
- Stephane Eranian

Primary authors: CARVALHO DE MELO, Arnaldo (Red Hat Inc.); ROGERS, Ian (Google); DESNOYERS, Mathieu (EfficiOS Inc.); JEANSON, Michael (EfficiOS); KIM, Namhyung (Google); ROSTEDT, Steven