Speeding up Kernel Testing and Debugging with virtme-ng
Problem

- Testing kernels can be painful and slow
- Lots of re-deployments and reboots involved
- Wait time
- Unpredictable results
- Lack of a fast edit/compile/test cycle
Proposed solution

- Create a virtual copy of your entire system on-the-fly
- Run your kernel inside this ephemeral system
- No re-deployments involved
- Extremely fast reboots
State of the art: virtme

- Written by Andrew Lutomirski
- Tool that allows to virtualize your running system
- Boot qemu/kvm instance with a custom kernel
- Export host rootfs to the guest (9p fs) in read-only mode
- Writes allowed in a tmpfs $HOME
virtme: limitations

- Limited testing capabilities
- Performance
  - Poor filesystem performance with 9p-fs
    - *(9p improvements with v5.15)*
    - Boot time not ideal
- Maintenance
  - Project not maintained anymore :(
• virtiofs + overlayfs
  • Improve filesystem performance
  • CoW live snapshot of the entire host filesystem

• qemu/kvm microVM
  • Lightweight virtual platform

• virtme-ng-init
  • Custom init script written in Rust
Shared file system that lets virtual machines access a directory tree on the host using FUSE / vhost-user

https://virtio-fs.gitlab.io/design.html
Replace 9p-fs with virtiofs

• $ time git diff
  • Before: 284.5s
  • After: 1.7s

• Boot time
  • Before: 6.2s
  • After: 5.2s
Overlayfs to handle writes (CoW)

- Use overlayfs to handle writes
  - upperdir/workdir \rightarrow tmpfs

- Automatically create overlays for the standard system paths at boot (/usr, /etc, /var, ...)

- EPERM issue with implicit overlayfs O_NOATIME (now fixed in virtiofsd upstream)
Qemu ‘microvm’ architecture

- microvm
  - virtual platform (inspired by firecraker)
  - Minimalist machine type (without PCI nor ACPI)
  - Optimized for boot time and memory footprint

- Boot time
  - Before: 5.2s
  - After: 3.8s

Kudos to Fejes Ferenc (@spyff0)
• virtme-ng-init
  • Custom init script implemented in Rust
  • Replace original virtme’s init script written in bash

• Boot time
  • Before: 3.8s
  • After: 1.2s
Result: boot time

![Boot time graph](image)
Demo

• https://youtu.be/3sDkVuXVw9A
Conclusion

- virtme-ng can provide a fast **edit/compile/test** workflow for kernel development
- Testing a kernel in 1.2s-1.3s is nice
- Easy to use by everyone (e.g., students, junior devs)
- Reduce power consumption required to do kernel testing
What’s next?

• Increase user base / collect feedbacks and potentially become a standard tool for kernel dev

• systemd support

• Better support across distro

• Better snaps/flatpack support
References

- virtme-ng
  https://github.com/arighi/virtme-ng
- Eco-friendly Linux kernel development: minimizing energy consumption during CI/CD
  https://lwn.net/Articles/935773/
- virtiofs
  https://virtio-fs.gitlab.io/
- Qemu microVM
  https://www.qemu.org/docs/master/system/i386/microvm.html
Questions?

Andrea Righi / @arighi
andrea.righi@canonical.com
github.com/arighi