Boot2container

An initramfs for reproducible infrastructures

Martin Roukala (Valve contractor)

Who am I?

- Martin Roukala (né Peres), AKA mupuf
- Freelancer at MuPuF TMI and Valve contractor

- Most active in the graphics subsystem

My mission

Production-ready upstream Linux graphics drivers

- Usable
- Reliable
- Available

- Best compability
- Best performance
- Worst reliability

- Nice-looking games
- High FPS, Low latency
- Super complex beasts

Contradictions? I think not!

Solving the contradictions

Automated testing to the rescue!

- But this ain't for the feint of heart…
  - Every GFX component needs its own test environment
  - Test suites are enormous (~1M unit tests for Vulkan)
  - Games are even harder to test automatically
  - Test results need to be stable, and reproducible by developers
  - Developers need feedback ASAP, but test content takes ~6h
  - Tens of machines running unreliable kernels and crash-happy GPUs

How do we make such a CI/test system?

By using/creating blocks with ***great*** interfaces!

Case study: creation & deployment of the test environment

Generating the test environment

Comparing the Embedded vs Web ways

Rootfs

- Traditional method of deploying the test environment in the embedded world
- Can be created using:
  - Yocto / buildroot / Debos / ...
- Generates a full disk image
  - Self-contained
  - Slower: The full image needs flashing
  - Low portability (modules, firmwares)
- Interface:
  - Provides platform setup and shared test environment for all test suites

OCI containers

- Traditionally used for unit testing, and in the web world
- Can be created using:
  - docker / podman / buildah / ...
- Generates a set of overlays (layers)
  - Requires platform setup
  - Faster: the base OS is cached
  - High portability
- Interface:
  - Provides isolated test env. for each test suite (composable)

How to start your container?

- Containers require platform initialization to be done
- Do we need another rootfs for this?

No!

What is boot2container?

- Small (< 20 MB) / net-bootable initramfs (url)
- Declarative configuration via the kernel command line
- Features:
  - Network: DHCP & NTP
  - Cache drive:
    - Auto-selection, or configurable
    - Auto-formating, if needed
    - Swap file
  - Volumes:
    - mirroring from an S3-compatible storage
    - local encryption (fscrypt)
    - expiration
  - Multi-architecture: Based on u-root, podman, and shell scripts

How to use boot2container?

- Directly:

  -
    ```
    qemu-system-x86_64 -kernel bzImage -initrd boot2container.cpio -nographic -m 512M -append 'console=ttyS0 b2c.container="-ti docker://alpine:latest"'
    ```
  - or using your favorite bootloader: grub, uboot, ...
- Netbooted:
  - PXE/HTTP: For machines inside a trusted local network
  - iPXE/HTTPS: For standalone machines on the other side of the planet

Quick demo

```
$ wget -O b2c-v0.9.3.cpio.xz https://gitlab.freedesktop.org/mupuf/boot2container/-/releases/v0.9.3/downloads/initramfs.linux_amd64.cpio.xz

$ wget https://gitlab.freedesktop.org/mupuf/boot2container/-/releases/v0.9.3/downloads/bzImage

$ fallocate -l 1G disk.img

$ qemu-system-x86_64 -drive file=disk.img,format=raw,if=virtio -kernel bzImage -initrd b2c-v0.9.3.cpio.xz -nographic -m 384M -enable-kvm -append 'console=ttyS0 b2c.cache_device=auto b2c.ntp_peer=auto b2c.container="-ti docker.io/library/alpine:latest"'
```

Real-world example

Linux cmdline to run a test suite (IGT) and download results

- b2c.cache_device=auto b2c.ntp_peer=auto
- b2c.minio="job,{{ minio_url }},{{ job_bucket_access_key }},{{ job_bucket_secret_key }}"
- b2c.volume="job,mirror=job/{{ job_bucket }},pull_on=pipeline_start,push_on=changes,expiration=pipeline_end"
- b2c.container="-ti registry.freedesktop.org/mupuf/valve-infra/machine_registration check"
- b2c.container="-t -v job:/results registry.freedesktop.org/drm/igt-gpu-tools/igt:master igt_runner -o /results"
- console={{ local_tty_device }},115200 earlyprintk=vga,keep loglevel=6

Potential use cases for b2c

- Fleet of automated systems local or deployed in remote places:
  - Netbooting is feasible (~50MB per boot + initial download of the layers)
  - Every boot behaves as if it were the first boot
  - No local IT needed aside from replacing misbehaving hardware
  - Deployments: public transport screens, chains of shops, ...
- Server provisioning in the cloud
- Let me know if you have other uses in mind!

Conclusion

- For our GFX CI needs, we need:
  - Reproducibility of results/environment/CI infrastructure
  - Reliability
  - Simplicity
- Boot2container delivered on the requirements, and more:
  - Easy to deploy anywhere (locally, or remotely)
  - Low maintenance cost (just bump the b2c version regularly)

Future work

- Add support for the most common architectures (WIP)
- Rewrite the initscript in Go
- Reduce the size of the initramfs by merging all Go binaries in one
- Add support for downloadable modules to reduce the kernel's size
- Finalize the interface in the v1.0

Links

- boot2container:
  - https://gitlab.freedesktop.org/mupuf/boot2container
- Story about the creation of boot2container:
  - https://mupuf.org/blog/2021/02/10/setting-up-a-ci-system-part-2-generating-and-deploying-your-test-environment/
- Netboot locally or over the internet:
  - https://mupuf.org/blog/2022/01/10/setting-up-a-ci-system-part-3-provisioning-your-ci-gateway/
- Setting up a desktop PC for testing:
  - https://mupuf.org/blog/2021/02/08/setting-up-a-ci-system-preparing-your-test-machine/

Thanks for listening!