FastFreeze

Unprivileged checkpoint/restore for containerized applications

LPC 2020, Aug 24

Nicolas Viennot
Two Sigma

Important Legal Information

This document is being distributed for informational and educational purposes only and is not an offer to sell or the solicitation of an offer to buy any securities or other instruments. The information contained herein is not intended to provide, and should not be relied upon for, investment advice. The views expressed herein are not necessarily the views of Two Sigma Investments, LP or any of its affiliates (collectively, “Two Sigma”). Such views reflect the assumptions of the author(s) of the document and are subject to change without notice. The document may employ data derived from third-party sources. No representation is made by Two Sigma as to the accuracy of such information and the use of such information in no way implies an endorsement of the source of such information or its validity.

The copyrights and/or trademarks in some of the images, logos or other material used herein may be owned by entities other than Two Sigma. If so, such copyrights and/or trademarks are most likely owned by the entity that created the material and are used purely for identification and comment as fair use under international copyright and/or trademark laws. Use of such image, copyright or trademark does not imply any association with such organization (or endorsement of such organization) by Two Sigma, nor vice versa

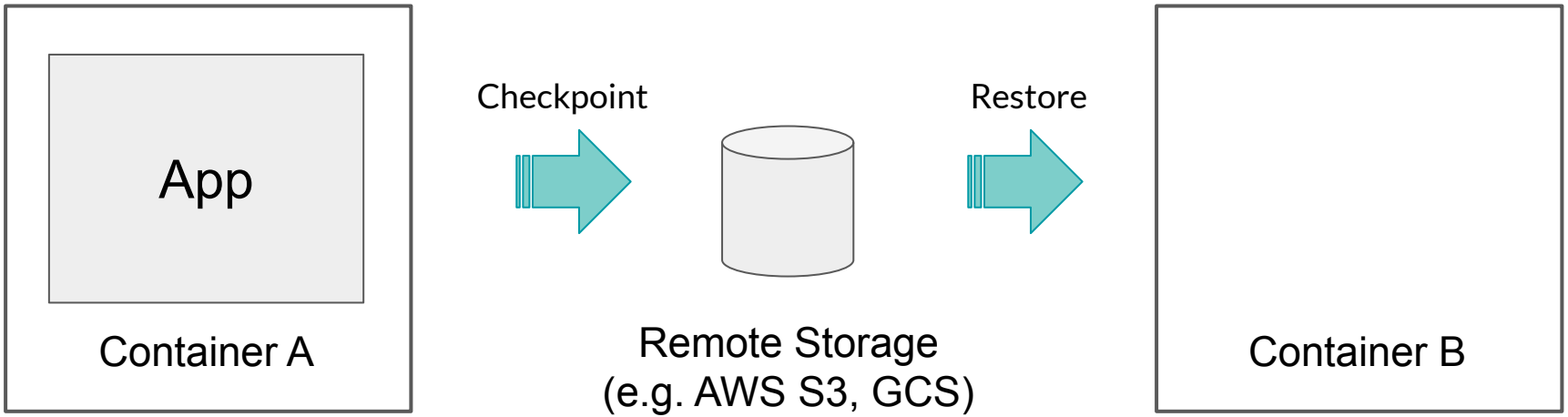
Scheduling Platforms

- Kubernetes
  - `command: app.sh [args]` in yaml file
- Job scheduler
  - `job_submit app.sh [args]`

Retry the same command in case of failures

Tight security, no root

Goal: C/R in containerized environments



No access to host

CRIU is only an engine

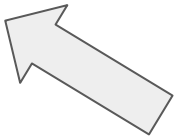
- Not ergonomic for the end-user. Need to be knowledgeable about the options
  - --cpu-cap, --shell-job, --tcp-close, --empty-ns net, --skip-in-flight, --ext-unix-sk
- Compilation/installation can be difficult on certain platforms
  - 10+ dependencies
- Needs root
- No file system C/R
  - Need synchronization with CRIU
- No image compression, upload/download

FastFreeze: Checkpoint/Restore, batteries included

- Easy to use, compatible with scheduling platforms
- Non-root CRIU
- Virtualize CPUID
- Upload/Download of images to/from remote storage
- C/R file system

FastFreeze: Checkpoint/Restore, batteries included

- Easy to use, compatible with scheduling platforms
- Non-root CRIU
- Virtualize CPUID
- Upload/Download of images to/from remote storage
- C/R file system



See talk on Fast Checkpointing with criu-image-streamer

Easy to use

Installation

Add this to a Dockerfile

```
RUN curl -SL https://github.com/.../fastfreeze.tar.xz | tar xJf - -C /opt; \
    ln -s /opt/fastfreeze/fastfreeze /usr/local/bin; \
    fastfreeze install
```

Usage

1. Run application through fastfreeze:
    - `fastfreeze run --image-url s3://ff/exec1.img -- ./app.sh [args…]`
2. Checkpoint the running app:
    - `fastfreeze checkpoint`

    Saves the app state in S3

    E.g. Kubernetes pre-stop hook, periodic checkpoints
3. Restore the app:

    Same command as 1.

    `run` checks for the image presence, and restore the app if found

CRIU without privileges

(in the root user namespace)

Previous Work

LPC 2018 Radoslaw Burny: Securely migrating untrusted workloads with CRIU

LPC 2019 Adrian Reber: CRIU and the PID dance

LPC 2019 Kamil Yurtsever et al: Update on Task Migration at Google Using CRIU

CAP_SYS_ADMIN

- /proc/sys/kernel/ns_last_pid / clone3()
- /proc/pid/map_files/*
- /proc/self/exe
- Time namespace

/proc/sys/kernel/ns_last_pid / clone3()

- Control PID of next process
- Requires CAP_SYS_ADMIN in associated PID namespace
- Workaround:
  - clone()+exit() until needed, cycle through 100,000 PID/sec
    https://github.com/twosigma/set_ns_last_pid
- What if we could create a user namespace?
  - Create user+PID namespace, no privileges needed
  - Mount /proc
    - Bad news, if /proc/sys is mounted read-only, we can’t mount /proc
    - Standard setup in Docker and Kubernetes

/proc/pid/map_files/*

- Allow to open an mmap file that has been deleted
- open() requires CAP_SYS_ADMIN in the root user namespace
- Workaround:
  - We have ptrace access to the process that has the file mmaped
    We can dump the file content anyways
  - Don’t delete files
- Lengthy email conversation to remove the security check
  - /proc/pid/fd/* doesn’t need CAP_SYS_ADMIN
  - Possibly because dma-buf being mmaped, kdbus (2015)
  - https://lore.kernel.org/patchwork/patch/1252086/

/proc/self/exe

- Points to the execve() first argument
- Changing it requires CAP_SYS_ADMIN in the local user namespace
- Workaround:
  - Use fork+execve+ptrace to spoof /proc/self/exe as demonstrated with https://github.com/nviennot/run_as_exe
  - App can cache /proc/self/exe on boot
- Long email history starting in 2012 to remove the check
  - See commit message ebd6de681238
  - /proc/self/exe offers no guarantees but security/{audit, tomoyo} uses it. Careful.

Time namespace

- `Thread.sleep(1000)` in the JVM:
  - Sleep until absolute time `clock_gettime(CLOCK_MONOTONIC) + 1000`
- Requires CAP_SYS_ADMIN in the local user namespace
- Workaround:
  - Interpose with LD_PRELOAD all calls that use CLOCK_MONOTONIC abs time
  - https://github.com/twosigma/libvirttime
  - Harder than you think

CAP_SYS_ADMIN

- /proc/sys/kernel/ns_last_pid / clone3()
- /proc/PID/map_files
- /proc/self/exe
- Time namespace

CAP_CHECKPOINT_RESTORE

- /proc/sys/kernel/ns_last_pid / clone3()
- /proc/PID/map_files
- /proc/self/exe
- ~~Time namespace~~

- Landed Aug 4th, in Linux 5.9-rc1
- Efforts by: Adrian Reber, Christian Brauner, myself

CPUID Virtualization

CPU feature virtualization

- Problem:
  - App boots on a AVX-512 capable host
  - Checkpoint/Restore app on a non-AVX capable host
  - Crash
- Solution:
  - Hide advanced CPU features
  - Don’t mix instance types

CPU feature virtualization

- Interpose ELF loader /lib64/ld-linux-x86-64.so.2
- Tell the kernel to trap CPUID instruction and generate a SIGSEGV
- When app invokes CPUID, emulate it

https://github.com/twosigma/libvirtcpuid

Future work: CPUID namespace

FastFreeze benefits and vision

- Make C/R mainstream
- Make cheap preemptible VMs more accessible
  Avoid wasting resources
- **Vision**: make migration part of the stack
  - Warm boots (layer on top of a docker image)
  - Memory ballooning
  - Topology optimization at runtime
    Collocate compute and storage at runtime

https://github.com/twosigma/fastfreeze

Nicolas.Viennot@twosigma.com