Task Migration at Scale Using CRIU

Linux Plumbers Conference 2018

Victor Marmol
vmarmol@google.com

Andy Tucker
agtucker@google.com

2018-11-15
Who we are

Outside of Google, we’ve worked on open source cluster management and containers
Who we are

Inside Google: we’re part of the **Borg** team

- Manages *all* compute jobs
- Runs on *every* server
What is Borg?

Google’s cluster management system

- **Borgmaster**: Cluster control and main API entrypoint
- **Borglet**: On-machine management daemon
- Suite of tools and UIs for managing jobs
- Many purpose-built platforms created on top of Borg
- Everything runs on Borg and everything runs in containers
Borg basics

Base compute primitive: **Task**

- A **priority** signals how quickly a task should schedule
- It’s **appclass** describes a task as either serving (latency sensitive) or batch
- Static content/binaries provided by **packages**
- A **container** isolates a task’s resources
  - Native Linux processes
  - Share an IP with the machine, ports are allocated for each task
Borg basics: evictions

When a task is **forcefully terminated** by Borg
- Typically receive a notification: 1-5min
- Our SLO allows for quite a few evictions
- Applications must handle them

Reasons for evictions
- **Preemption**: a higher priority task needs the resources
- Software **upgrades** (e.g.: kernel, firmware)
- Re-balancing for availability or performance
Evictions are impactful and hard to handle

Technical Complexity
- Handling evictions requires state management
  - How and what state to serialize and where to store it
- Application-specific and not very reusable

Lost Compute
- Batch jobs run at lower priorities and get preempted often
- Even platforms that handle them for users, don’t do a great job

Compute is lost...
Migrations to avoid evictions

*Transparently replace evictions with migration*

Native **task migration** offering in Borg

- Borg controls the eviction → always knows when to migrate
- Native management of state allows reuse for all workloads

Various possible mechanisms

- **Checkpoint/restore**
  - Pause application, transfer state, resume
  - Long blackout period, no brownout

- **Live**
  - Very short blackout, but with a longer brownout
  - Very low impact to applications
Challenges with task migration

Migrating network connections
Port collisions and port use
Storage migration is slow
Must virtualize machine-local resources
Linux process state hard to migrate
Challenges with task migration

- Migrating network connections: Drop the connection, user handles reconnections
- Port collisions and port use: NET namespaces and IPv6 per-container
- Storage migration is slow: Little to no local storage
- Must virtualize machine local resources: Linux namespaces
- Linux process state hard to migrate: CRIU!
Migration Workflow
Migration
Checkpoint/Restore
Isolated Task Environment

- Linux namespaces
- Little local storage
- IPv6
- Google libraries (e.g.: Stubby/gRPC)
Machine A

Task

Checkpoint
- Pause task
Task

Machine A

Checkpoint
- Pause task
- Serialize state
Task

Machine A

Checkpoint
- Pause task
- Serialize state
- Upload to distributed storage

Colossus
Migration
- Borgmaster chooses new machine to schedule the task.
Restore

- Download from distributed storage
Restore
- Download from distributed storage
- Deserialize state
Restore

- Download from distributed storage
- Deserialize state
- Continue running task
Isolated Task Environment

- Machine is opaque to the task
- Your local data travels with the task
- Your IP changes
- Google libraries re-establish connections
Networking

Networking @ Google
- Standardized RPC implementation: Stubby/gRPC
- Nearly all communication is RPC
- Unique IPv6 address per task
- BNS: Borg DNS, used by RPC layer

Task Migration
- Stubby/gRPC automatically reconnects
- Reconnect is transparent to users
- IP address changes, but this is rarely a problem
Storage

Storage @ Google: Minimized local storage
- Most tasks are **stateless**, few require local SSD/HDD
- Those that require state use our **remote storage stacks** (e.g.: Colossus, Spanner)
- Small local storage is offered via tmpfs

Task migration
- Lack of local storage greatly simplifies work
- Remote storage stacks use RPC and thus recover gracefully
- Small local storage is migrated with task
Task environment

Container
- Primarily used for resource isolation
- Full namespaces applied

Security
- Root is not mapped into user namespace
- Capabilities are strictly limited

Root filesystem
- Separate from the host machine’s
- Built and bundled by the task as a package
CRIU

Checkpoint/Restore in User Space
- Used to serialize/deserialize the task’s process

Security and isolation
- Run inside a task’s container
- Run with minimal privileges

The Migrator
- Injected into task during a migration, orchestrates the migration
- Manages execution of CRIU
- Encrypts and compresses checkpoint on the fly
  - Pretends to be a CRIU pageserver
In practice today

Migrations take **1-2min** and succeed **90%+** of the time

Where the time goes
- Checkpoint/restore is relatively fast for well-behaved tasks
- Writing/reading to remote storage dominates checkpoint/restore
- Scheduling delays are also a large source of latency

Causes of failures
- Timeouts from high task resource usage (e.g.: threads, memory)
- Different host environments
- Misc failures in serialization (e.g.: unsupported features)
Users

Works well for **batch** jobs

- Latency tolerant, longer-running, and lower priority
- Some are highly sharded and see many evictions
- Long pipelines suffer when some parts are evicted

User feedback

- They love it! Super **simple to adopt**
- Desire for advanced features
  - Migration notifications
  - User-controlled pause/resume

Not a great offering for latency-sensitive jobs
Adoption challenges

Handling connection failures
- In theory: users are taught to expect failures
- In practice: users don’t handle failures well
  - Expect them not to occur and reset their state when they do

Isolating task environment
- Users make assumptions about the underlying host
  - Services are available via localhost
  - Expecting host:port to work
- Users don’t expect the underlying host to change at runtime
  - Certain features detected at startup and never refreshed (e.g.: kernel, CPU, location)
Experience with CRIU

In one word? **AMAZING!**
- Mostly worked out of the box with few changes
- Reliability and performance have been great in production
- Community has been helpful and quick to fix issues

Our changes
- Performance improvements for checkpoint/restore
- Increasing/improving some limitations (see next slide)
- Most patches sent upstream
CRIU security

CRIU suggested to run as root
- Security auditing found a series of bugs
- A malicious task can hijack a CRIU process

Recommendation
- Run CRIU as the task’s user
- Run in user namespace without root mapped in
- Trim privileges to minimal set
What could do with improvements

Performance
- Some expensive operations remain, some have kernel limitations
  - e.g.: waitpid on all threads is $O(n^2)$

Security
- Reducing need for root and elevated capabilities
- Not well tested in this setup

Misc
- Contributing patches back is a bit hard
What could do with improvements

Live migration
- Parts of incremental restore are very, very difficult
- Lots of work ahead to do the type of brownout used in VM live migration today

Handling time
- Hard to abstract away many of the time HW counters
- Time namespaces to the rescue?
Future work

Increasing adoption internally
- Reduce lost compute and simplify user tasks
- Targeting on-by-default for large batch workloads

Machine-to-machine migration
- Skip the distributed storage of the checkpoint
- Reduces migration times to ~30s

Live migration
- Able to address latency sensitive workloads
- Will require some work in our stack and in CRIU
Questions?

Native task migration offering in Borg
- Reduces compute lost to evictions
- Simplifies task handling of preemptions
- Addresses most batch workloads
- Serving workloads need live migration

CRIU
- Works amazingly well out of the box
- Security an area of investment
- We are excited about and look forward to live migration!

Victor Marmol
vmarmol@google.com

Andy Tucker
agtucker@google.com