Google

Task migration at Google using CRIU

Linux Plumbers Conference 2018

Andy Tucker
agtucker@google.com
November 13, 2018

Basics

Google's internal cloud managed by "Borg" resource management system

- Heavy resource overcommit for high utilization

Applications submitted as *jobs*

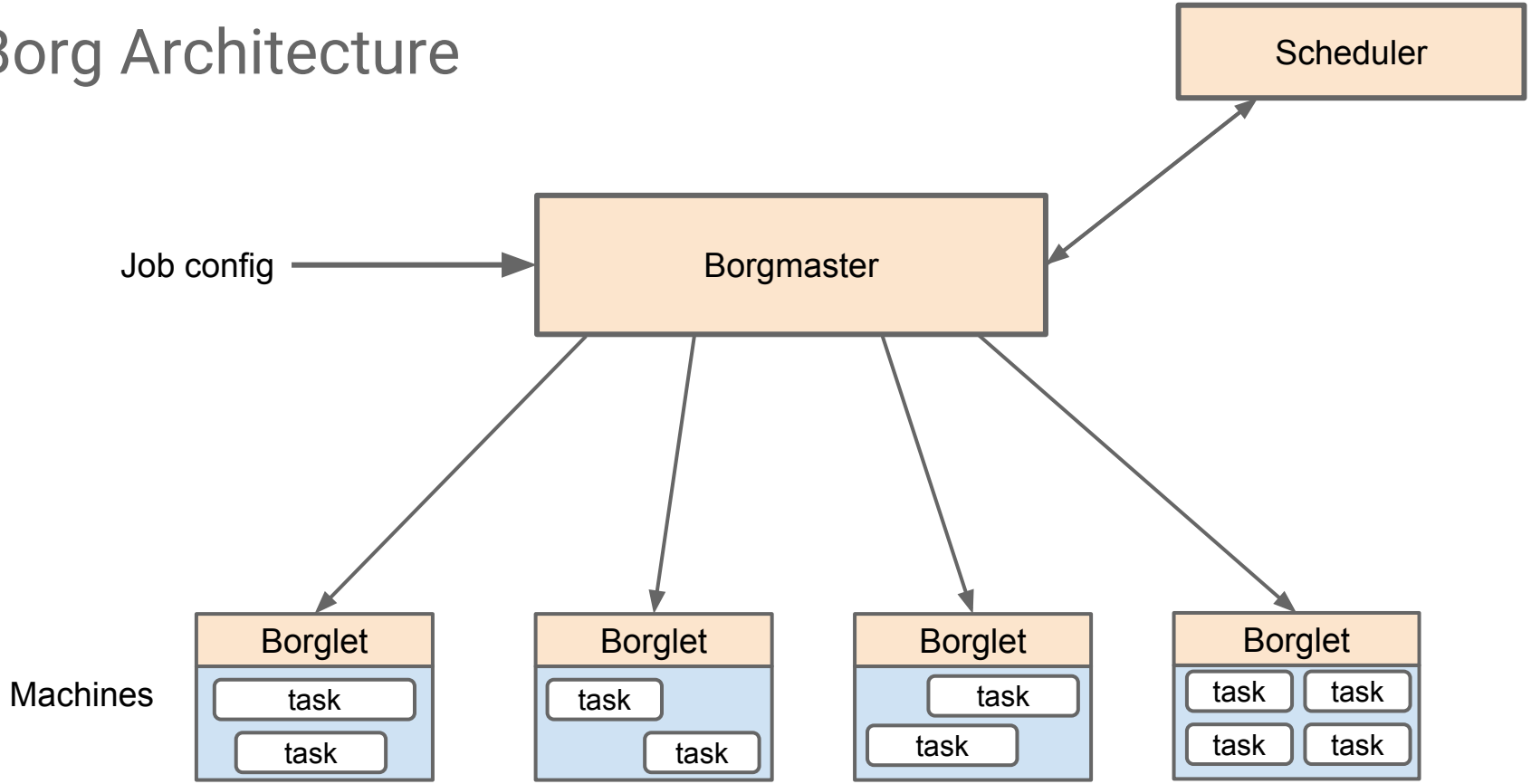
- Job config defines priority, requirements, number of replicas

Each replica is a *task* that can run on a different machine

Tasks are *preemptable* - high priority tasks can preempt lower ones

- Preemption causes task to *restart* on new machine

Borg Architecture



Why migrate?

Lower priority tasks experience frequent preemptions

- Also evictions for machine shutdown (kernel upgrades, hardware maintenance)
- All in-memory computation lost on a task restart
- Depending on job, rebuilding state may be expensive

Migration allows memory/process state to be preserved

- Avoids restart cost
- Allow jobs to run at "natural" priority
- No need for application-specific checkpointing

Prerequisites

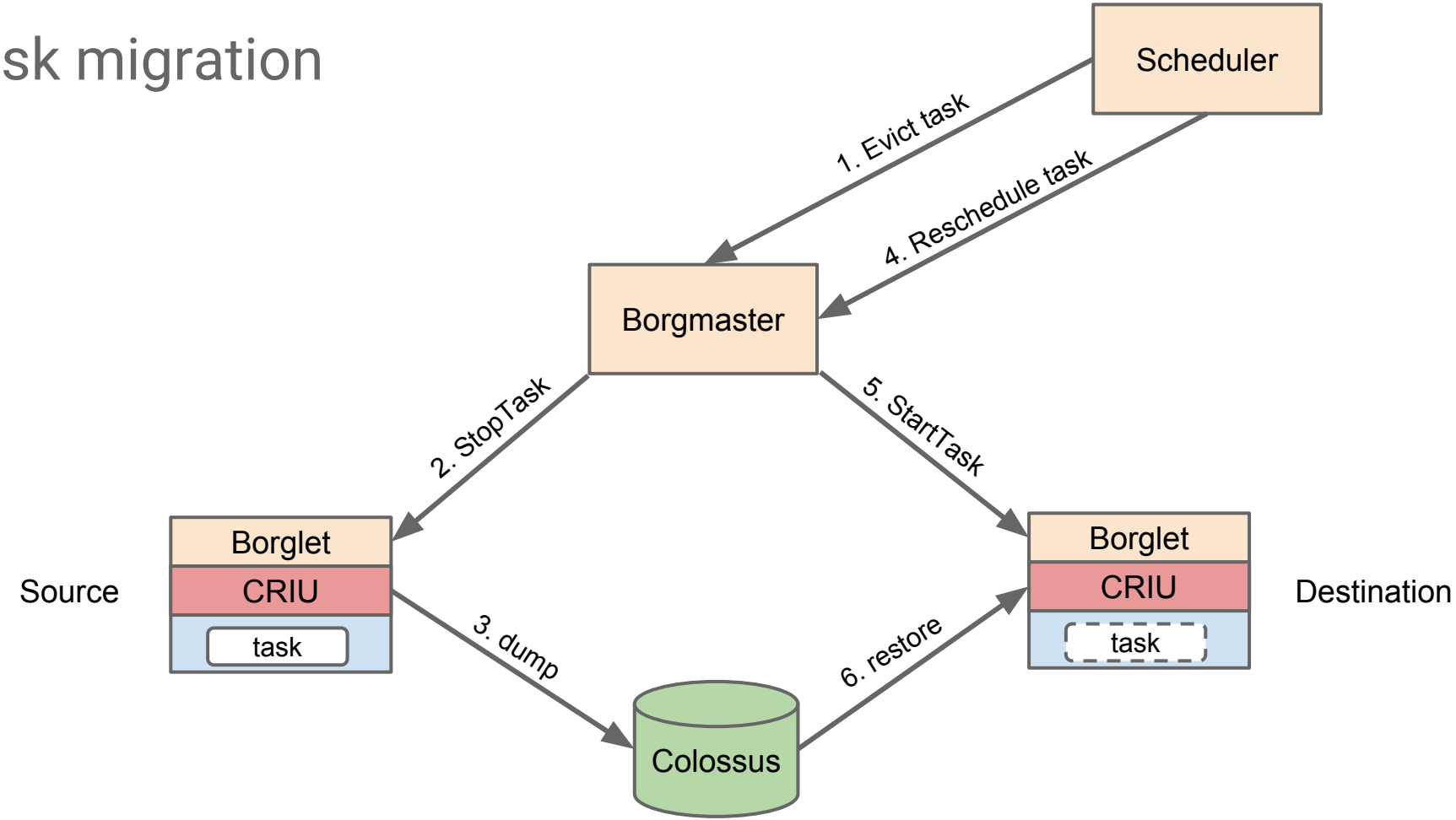
Tasks have few machine dependencies

- Running inside namespaces (pid, network, mount, UTS)
- No local disk
- Avoid use of hostname/IP (mostly...)

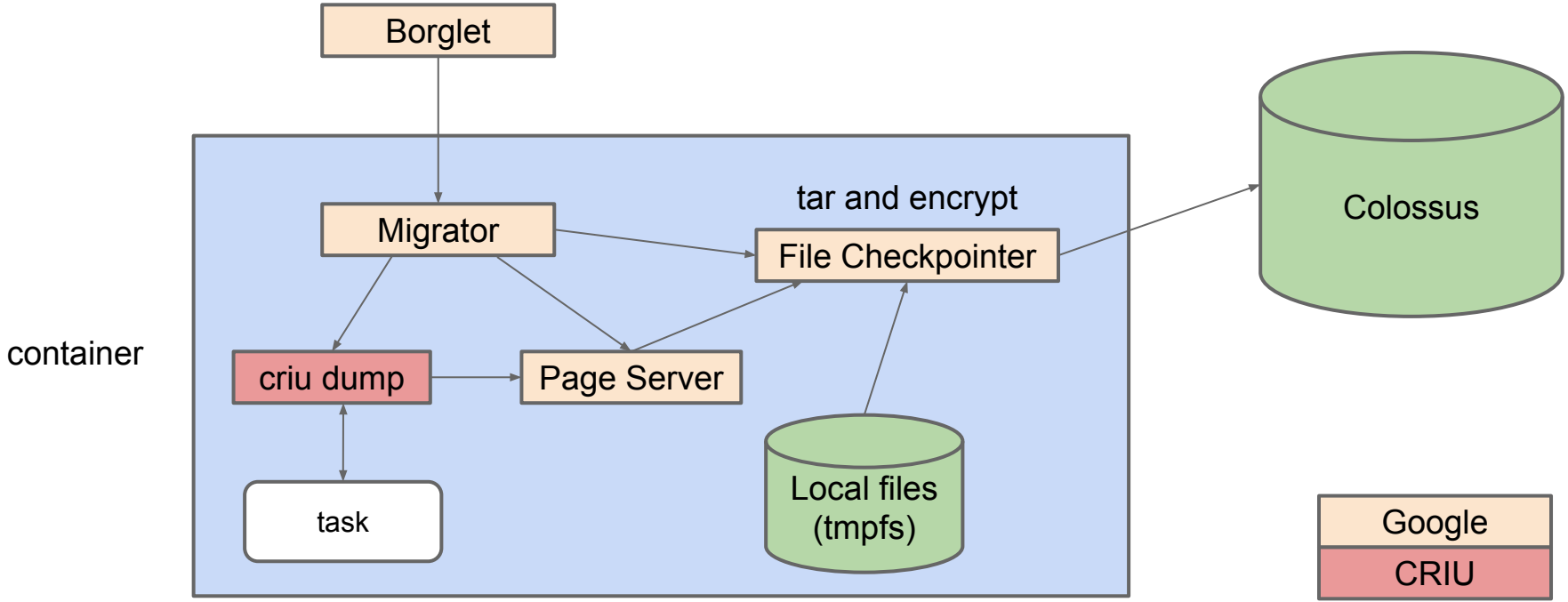
Clients are tolerant of network failures

- Designed for resiliency
- Most use gRPC/Stubby - automatic retry on connection failure
- IP+port lookup based on task ID (BNS)

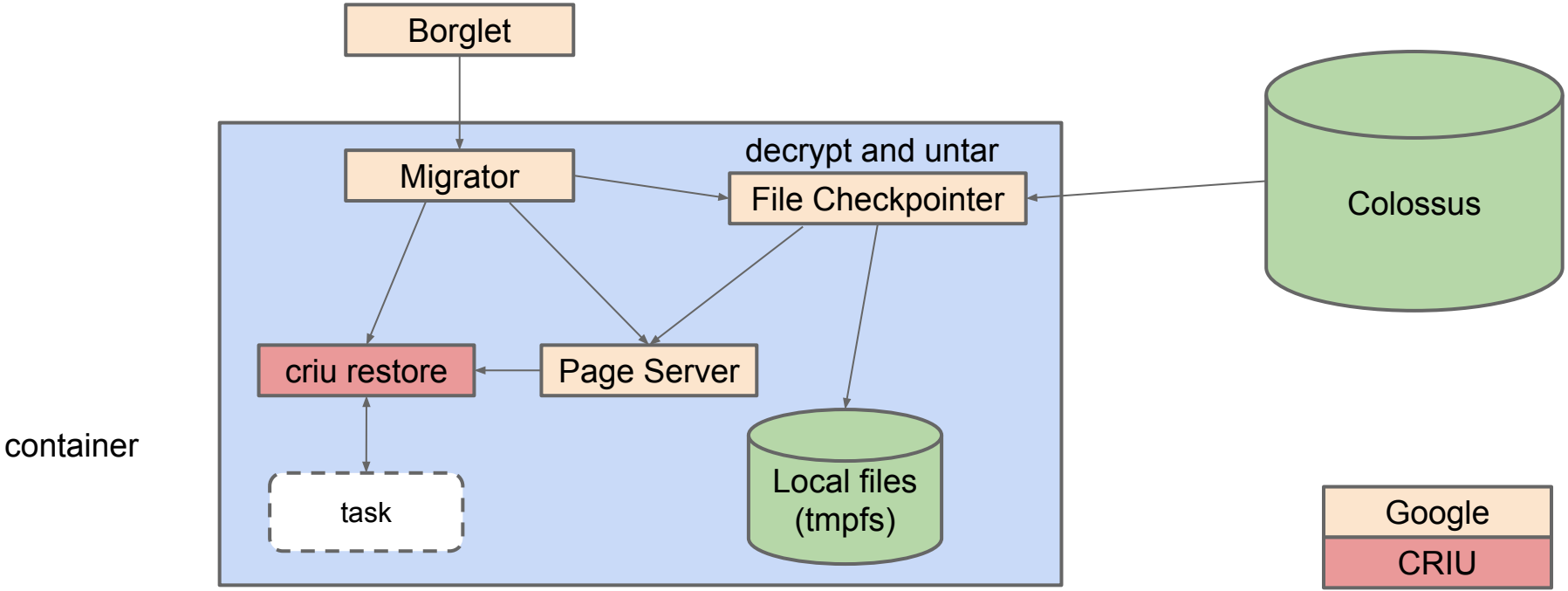
Task migration



Dump


Borglet
Migrator
tar and encrypt
File Checkpointer
Colossus
container
criu dump
Page Server
task
Local files
(tmpfs)
Google
CRIU

Restore


Borglet
Migrator
decrypt and untar
File Checkpointer
Colossus
criu restore
Page Server
container
task
Local files
(tmpfs)
Google
CRIU

Challenges

- Performance
- Time handling
- Security (separate talk coming later)

Performance

Staging in local disk or tmpfs is slow - moved to streaming design

- Also avoids 2x memory requirements

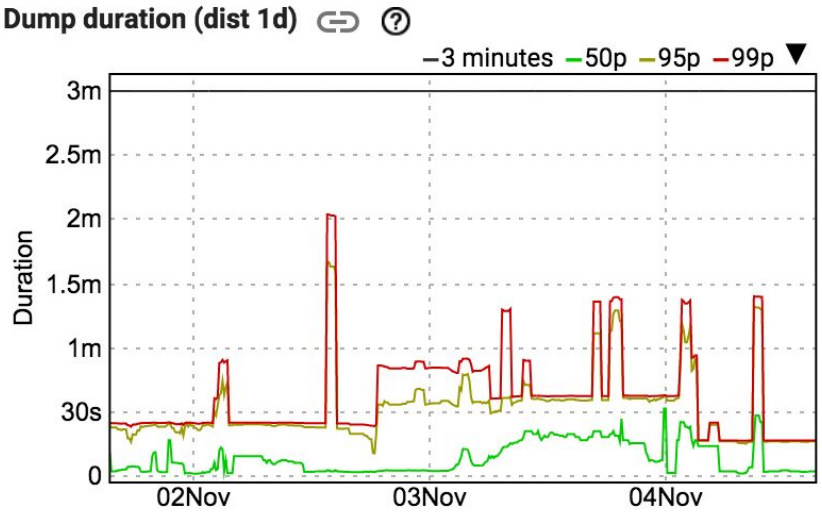
Scaling issues for tasks with large numbers (~1000s) of threads and sockets

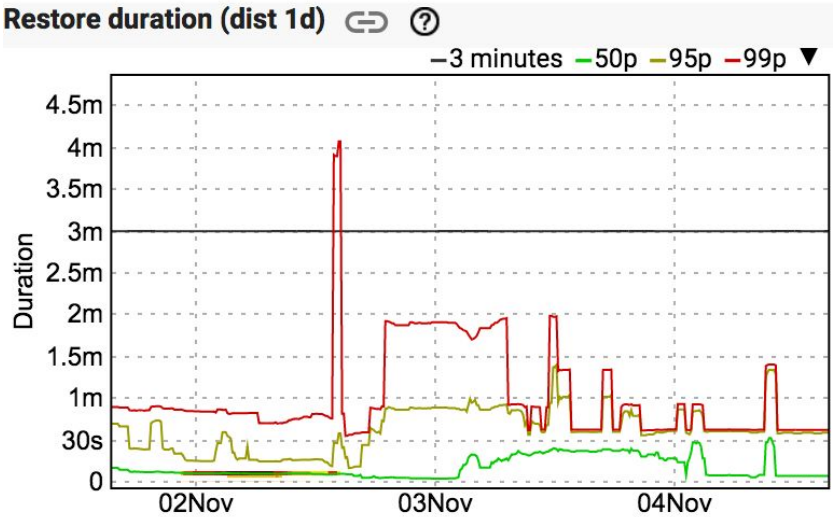
- `wait4`/`waitpid` traversing linked list of threads
- Walking lists of fds on restore (fixed in HEAD)
- Small socket hash tables

Generally 1-2 minute blackout time for most tasks

Live migration would provide further improvements (overlap dump and restore)

Performance data





Outliers are very large (~100 GB, 100s of threads, 1000s of file descriptors)

Time handling

TSC (x86 TimeStamp Counter) values not comparable across machines

- Can go "backwards" (non-monotonic) or jump forward
- Similar for `CLOCK_MONOTONIC` and `CLOCK_BOOTTIME`

Compensating with offset applied by low level libraries

- Current (virtual) TSC, etc. sent as metadata from source to destination
- Used to compute offset after migration, applied in wrapper library
- Migrations limited to CPUs of same TSC frequency (or could scale...)

Time namespace (kernel support) would be valuable (especially if extended to TSC)

Conclusions

Migration working well within Google, solving real problems

Performance within reasonable bounds (but working on improvements)

CRIU is stable and working well in production environments