Fast checkpointing
with criu-image-streamer

Nicolas Viennot
Two Sigma

LPC 2020, Aug 24
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.
Google Preemptible VMs
Up to 5x cheaper
30sec eviction deadline
16 CPUs, 128GB RAM, 2GB/s Network
Goal: **Checkpoint in 30s**
16 CPUs, 128GB RAM, 2GB/s Network

Goal: **Checkpoint in 30s**

\[
\frac{128}{30} = 4.2\text{GB/s}
\]
Problem: Fast checkpointing

Running Application

App

Host

Checkpoint

Remote Storage

AWS S3, GCS
Checkpointing: the old way

App
Checkpointing: the old way

Diagram:
- App
  - Parasite
- CRIU
  - ptrace
Checkpointing: the old way

App
Parasite

pipe

CRIU

files

Image
Checkpointing: the old way

App
Parasite

pipe

CRIU

files

Image

fs

App files

Image

Archive

tar

pipe

Compress

lz4

pipe

Upload

gsutil cp -

network
Checkpointing: the old way

App → Parasite → CRIU → Image

Pipe

Image files

100MB/s

Image

App files

100MB/s

Archive

tar

Pipe

Compress

lz4

Pipe

Upload

gsutil cp -

Network

500MB/s

100MB/s
Streaming Checkpointing

App files

files

Archive

tar

pipe

CRIU

pipe

Compress Upload

network
Problem: Compress and upload stream

- UNIX pipes
- Multi-core compression
- Sharded upload
- Sockets
- Stragglers
Problem: Compress and upload stream

1 pipe

Multi-core compression
Sharded upload
Problem: Compress and upload stream
Problem: Compress and upload stream
Problem: Compress and upload stream

1GB/s max
Problem: Compress and upload stream

Contention

1GB/s max
Problem: Compress and upload stream

1GB/s max
Problem: Compress and upload stream
Problem: Compress and upload stream

Pipe flow diagram:

- Thread 1
  - liblz4
  - libzstd

- Thread 2

- Thread 3

- Thread 4

- Thread 5
  - libaws3
  - libgoogle

- Thread 6
  - ...

Libraries:
- liblz4
- libzstd
- libaws3
- libgoogle
Problem: Compress and upload stream

Shell command
lz4 - - | gsutil cp - gs://bucket/file
Problem: Compress and upload stream

Shell command
lz4 - - | openssl ... | gsutil cp - gs://bucket/file
Problem: Compress and upload stream

Shell command
```bash
lz4 - - | openssl ... | gsutil cp - gs://bucket/file
```
Problem: Split pipe and load-balance outputs

1 pipe

High performance
Load balancing

UNIX pipes
Fast
Fast
Slow
Split pipe and load-balance outputs

Kernel pipe buffers
Split pipe and load-balance outputs

Which output?

Fast

Fast

Slow
Split pipe and load-balance outputs

select()/epoll() => ioctl(FIONREAD)
Probe for amount of data in pipe

Fast

Fast

Slow
Split pipe and load-balance outputs

- `select()/epoll()` => `ioctl(FIONREAD)`
  - Probe for amount of data in pipe

- `read()/write()` => `splice()`
  - Zero-copy pipe-to-pipe transfer
Streaming Checkpointing

1. App files
2. Archive with tar
3. CRIU
4. CRIU Image Streamer
5. Compress with lz4
6. Upload

Flow:
- App files to Archive
- Archive to CRIU
- CRIU to CRIU Image Streamer
- Compress
- Upload
Streaming Checkpointing

App files

App
Parasite

files

Archive
tar

pipe

CRIU

pipe

CRIU
Image
Streamer

pipes

Compress+
Upload

pipe

vmsplice

splice

splice

splice

15 GB/s
## Performance Trade-off: CPU vs Network

<table>
<thead>
<tr>
<th></th>
<th>HTTPS</th>
<th>HTTPS</th>
<th>HTTPS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Upload Protocol</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Compression Algo</strong></td>
<td>None</td>
<td>lz4</td>
<td>zstd</td>
</tr>
<tr>
<td><strong>Compression Ratio</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(typical Java app)</td>
<td>1x</td>
<td>3x</td>
<td>5x</td>
</tr>
<tr>
<td><strong>Throughput per CPU</strong></td>
<td>550 MB/s</td>
<td>350 MB/s</td>
<td>220 MB/s</td>
</tr>
<tr>
<td>(before compression)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Time to checkpoint</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(16 CPUs, 128GB RAM, 2GB/s NIC)</td>
<td>64s</td>
<td>24s</td>
<td>37s</td>
</tr>
</tbody>
</table>
Streaming Checkpoint

- App files
- CRIU
- Parasite
- Archive tar
- CRIU Image Streamer
- Compress + Upload

Files are passed through the system with pipes, ending with compression and upload.
Streaming Restore

App files → files → Archive tar → pipe → CRIU → pipe → CRIU Image Streamer → pipes → Compress + Upload

App Parasite → pipe → CRIU → pipe → CRIU Image Streamer
Problem: CRIU reads image out-of-order

- Writes inventory.img at the end of the stream
- Reads inventory.img at the beginning of the stream
Problem: CRIU reads image out-of-order

- Writes inventory.img at the end of the stream
- Reads inventory.img at the beginning of the stream

Solution

- Buffer the entire image in memory
- Let CRIU access image in arbitrary order
- 2x memory problem solved by deallocating after transferring data to CRIU
Checkpoint example, it’s Unix

```bash
exec 10> >(lz4 - - | gsutil cp - gs://bucket/img-1.lz4)
exec 11> >(lz4 - - | gsutil cp - gs://bucket/img-2.lz4)
exec 20< <(tar -C / -cpf - /scratch/app)

criu-image-streamer \
   --shard-fds 10,11 --ext-file-fds fs.tar:20 \
   --images-dir /tmp capture &
criu dump --images-dir /tmp --tree $APP_PID --stream
```
Live Migration

criu-image-streamer capture | ssh remote criu-image-streamer serve
Acknowledgements

● Two Sigma
  ○ Vitaly Davidovich
  ○ Peter Burka

● CRIU team
  ○ Andrei Vagin, Google
  ○ Radostin Stoyanov, Cambridge
  ○ Mike Rapoport, IBM
Future Work

- Speed-up restore: zero-copy
- Support pre-dump
- Perform dedup
- Adaptive compression
- Add kernel pipe statistics
Conclusion

- We can checkpoint at 15GB/s
- Unix philosophy with performance
- ioctl(FIONREAD) and splice for zero-copy load-balancing

https://github.com/checkpoint-restore/criu-image-streamer

Nicolas.Viennot@twosigma.com